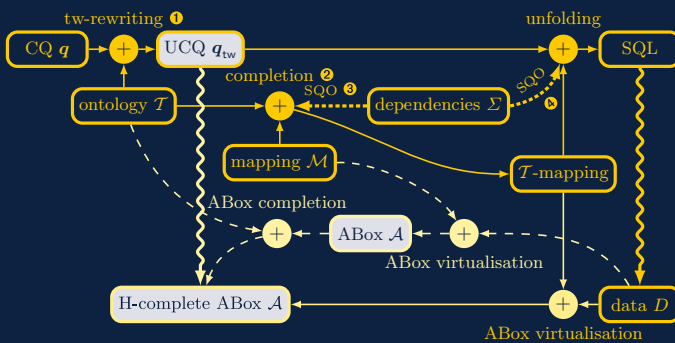


Sebastian Rudolph
 Georg Gottlob
 Ian Horrocks
 Frank van Harmelen (Eds.)

Reasoning Web

Semantic Technologies
 for Intelligent Data Access

9th International Summer School 2013
 Mannheim, Germany, July/August 2013
 Proceedings



Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Sebastian Rudolph Georg Gottlob
Ian Horrocks Frank van Harmelen (Eds.)

Reasoning Web

Semantic Technologies
for Intelligent Data Access

9th International Summer School 2013
Mannheim, Germany, July 30 – August 2, 2013
Proceedings



Springer

Volume Editors

Sebastian Rudolph

Technische Universität Dresden, Fakultät Informatik

Nöthnitzer Str. 46, 01062 Dresden, Germany

E-mail: sebastian.rudolph@tu-dresden.de

Georg Gottlob

University of Oxford, Department of Computer Science

Wolfson Building, Parks Road, Oxford, OX1 3 QD, UK

E-mail: georg.gottlob@cs.ox.ac.uk

Ian Horrocks

University of Oxford, Department of Computer Science

Wolfson Building, Parks Road, Oxford, OX1 3 QD, UK

E-mail: ian.horrocks@cs.ox.ac.uk

Frank van Harmelen

Vrije Universiteit Amsterdam, Department of Computer Science

de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

E-mail: frank.van.harmelen@cs.vu.nl

ISSN 0302-9743

e-ISSN 1611-3349

ISBN 978-3-642-39783-7

e-ISBN 978-3-642-39784-4

DOI 10.1007/978-3-642-39784-4

Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013943284

CR Subject Classification (1998): I.2, H.2, F.4, H.3, J.1

LNCS Sublibrary: SL 3 – Information Systems and Application,
incl. Internet/Web and HCI

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the lecture notes of the 9th Reasoning Web Summer School 2013, held from July 30 to August 2, 2013, in Mannheim, Germany.

Over the years, the Reasoning Web Summer School series has become the premier educational event for the active field of reasoning methods for the Web, attracting both young and well-established researchers. Previous editions of the school were held in Malta (2005), Lisbon (2006), Dresden (2007 and 2010), Venice (2008), Bressanone-Brixen (2009), Galway (2011), and Vienna (2012).

The 2013 edition was hosted by the Data and Web Science Research Group at the University of Mannheim (<http://dws.informatik.uni-mannheim.de>). The group, co-headed by Heiner Stuckenschmidt and Chris Bizer, currently consists of four professors, five post-doctoral researchers and 15 PhD students. Covering the whole life cycle of Web Data including information extraction and integration, data management, reasoning and mining as well as user interaction with data, the group is the perfect host for the summer school. It is well known for related community efforts such as DBpedia and the ontology alignment evaluation initiative and has a strong background in Web reasoning, working mainly on non-standard reasoning for information integration and information retrieval, scalable reasoning by parallelization and the combination of logical and probabilistic inference.

Following the tradition of the two previous years, the summer school has been co-located with the International Conference on Web Reasoning and Rule Systems (RR). Moreover, these two events were directly preceded by the Second OWL Reasoner Evaluation Workshop (ORE 2013) on July 22 and the 26th International Workshop on Description Logics (DL 2013) July 23–26, both held close by in Ulm, Germany.

The 2013 summer school program covered diverse aspects of Web reasoning, ranging from scalable lightweight formalisms such as RDF to more expressive ontology languages based on description logics. It also featured foundational reasoning techniques used in answer set programming and ontology-based data access as well as emerging topics like geo-spatial information handling and reasoning-driven information extraction and integration.

The tutorial articles were prepared as accompanying material for the students of the summer school, to deepen the understanding and to provide pointers to the relevant literature in the field. However, we expect these lecture notes to be of use beyond that for colleagues and interested readers with sufficient background in computer science.

The accompanying lecture slides and teaching material of all tutorials are made available on the summer school website at <http://reasoningweb.org/2013/>.

We would like to thank everybody who helped make this event possible. Above all, we would like to thank all the lecturers and their co-authors, whose

hard work and devoted commitment created the solid foundation for this event. We are grateful to all the reviewers: your timely feedback on the submitted articles helped the authors to further increase the quality of their excellent tutorials.

Furthermore, we express our gratitude to the local organization team led by Heiner Stuckenschmidt for putting great organizational and administrative effort into making the summer school both a pleasant and a insightful experience.

Last but not least, we thank all sponsors of this event, who provided support financially or in terms of resources: Fluid Operations, Semafora, Association for Logic Programming, NSF, LOD2, Artificial Intelligence Journal, IOS Press, SOFTPLANT, the University of Mannheim, Siemens AG Österreich, and Springer. We particularly thank Marco Maratea, who – once again – did an excellent job in acquiring sponsors, which allowed us to keep the registration fees on a moderate level and support students with travel grants.

July 2013

Sebastian Rudolph
Georg Gottlob
Ian Horrocks
Frank van Harmelen

Organization

General Chair

Sebastian Rudolph Technische Universität Dresden, Germany

Scientific Advisory Board

Georg Gottlob University of Oxford, UK
Ian Horrocks University of Oxford, UK
Frank van Harmelen Vrije Universiteit Amsterdam, The Netherlands

Local Chair

Heiner Stuckenschmidt University of Mannheim, Germany

Local Organization

Stephanie Keil University of Mannheim, Germany
Nicole Nowak University of Mannheim, Germany
Sebastian Kotthoff University of Mannheim, Germany

Sponsorship Chair

Marco Maratea Università di Genova, Italy

Additional Reviewers

Michael Morak
Rafael Peñaloza-Nyssen
Andreas Pieris
Linda van den Brink

Sponsors

Platinum Sponsors



Association for Logic Programming



Semafora



Fluid Operations



National Science Foundation

Gold Sponsors



Artificial Intelligence Journal



LOD2

Silver Sponsors

IOS Press

SOFTPLANT

Invited Partners

Siemens AG Österreich
DERI Galway

Springer

Table of Contents

Introduction to Linked Data and Its Lifecycle on the Web	1
<i>Sören Auer, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, and Amrapali Zaveri</i>	
RDFS and OWL Reasoning for Linked Data	91
<i>Axel Polleres, Aidan Hogan, Renaud Delbru, and Jürgen Umbrich</i>	
Introductions to Description Logics – A Guided Tour	150
<i>Anni-Yasmin Turhan</i>	
Answer Set Programming	162
<i>Wolfgang Faber</i>	
Ontology-Based Data Access with Databases: A Short Course	194
<i>Roman Kontchakov, Mariano Rodríguez-Muro, and Michael Zakharyashev</i>	
A Geo-semantics Flyby	230
<i>Krzysztof Janowicz, Simon Scheider, and Benjamin Adams</i>	
Statistical Relational Data Integration for Information Extraction	251
<i>Mathias Niepert</i>	
Author Index	285

Introduction to Linked Data and Its Lifecycle on the Web

Sören Auer, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, and Amrapali Zaveri

AKSW, Institut für Informatik, Universität Leipzig, Pf 100920, 04009 Leipzig
lastname@informatik.uni-leipzig.de
<http://aksw.org>

Abstract. With Linked Data, a very pragmatic approach towards achieving the vision of the Semantic Web has gained some traction in the last years. The term Linked Data refers to a set of best practices for publishing and interlinking structured data on the Web. While many standards, methods and technologies developed within by the Semantic Web community are applicable for Linked Data, there are also a number of specific characteristics of Linked Data, which have to be considered. In this article we introduce the main concepts of Linked Data. We present an overview of the Linked Data lifecycle and discuss individual approaches as well as the state-of-the-art with regard to extraction, authoring, linking, enrichment as well as quality of Linked Data. We conclude the chapter with a discussion of issues, limitations and further research and development challenges of Linked Data. This article is an updated version of a similar lecture given at Reasoning Web Summer School 2011.

1 Introduction

One of the biggest challenges in the area of intelligent information management is the exploitation of the Web as a platform for data and information integration as well as for search and querying. Just as we publish unstructured textual information on the Web as HTML pages and search such information by using keyword-based search engines, we are already able to easily publish structured information, reliably interlink this information with other data published on the Web and search the resulting data space by using more expressive querying beyond simple keyword searches.

The Linked Data paradigm has evolved as a powerful enabler for the transition of the current document-oriented Web into a Web of interlinked Data and, ultimately, into the Semantic Web. The term Linked Data here refers to a set of best practices for publishing and connecting structured data on the Web. These best practices have been adopted by an increasing number of data providers over the past three years, leading to the creation of a global data space that contains many billions of assertions – the Web of Linked Data (cf. Figure 1).

In this chapter we give an overview of recent development in the area of Linked Data management. The different stages in the linked data life-cycle [11,7] are depicted in Figure 2.

Information represented in unstructured form or adhering to other structured or semi-structured representation formalisms must be mapped to the RDF data model (*Extraction*). Once there is a critical mass of RDF data, mechanisms have to be in place to

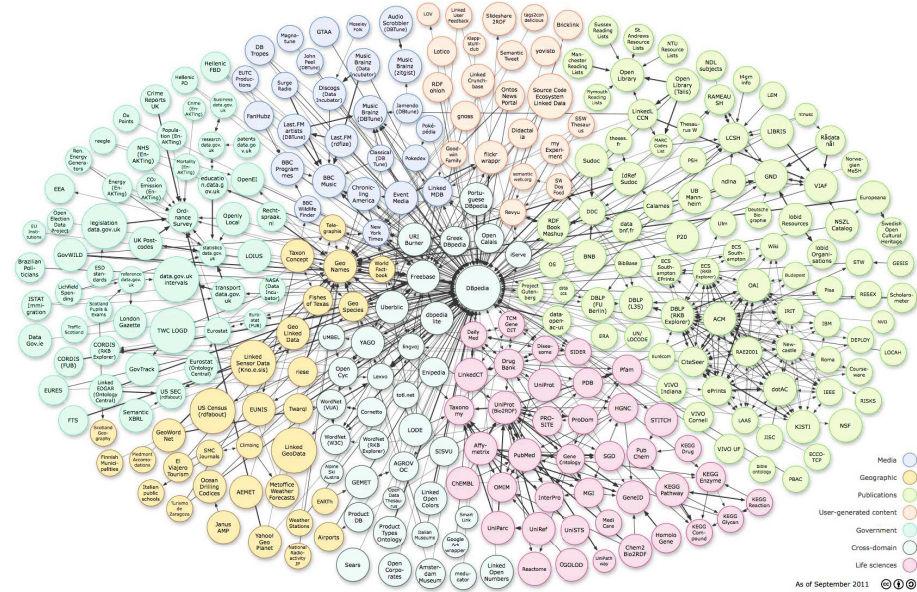


Fig. 1. Overview of some of the main Linked Data knowledge bases and their interlinks available on the Web. (This overview is published regularly at <http://lod-cloud.net> and generated from the Linked Data packages described at the dataset metadata repository ckan.net.)

store, index and query this RDF data efficiently (*Storage & Querying*). Users must have the opportunity to create new structured information or to correct and extend existing ones (*Authoring*). If different data publishers provide information about the same or related entities, links between those different information assets have to be established (*Linking*). Since Linked Data primarily comprises instance data we observe a lack of classification, structure and schema information. This deficiency can be tackled by approaches for enriching data with higher-level structures in order to be able to aggregate and query the data more efficiently (*Enrichment*). As with the Document Web, the Data Web contains a variety of information of different quality. Hence, it is important to devise strategies for assessing the quality of data published on the Data Web (*Quality Analysis*). Once problems are detected, strategies for repairing these problems and supporting the evolution of Linked Data are required (*Evolution & Repair*). Last but not least, users have to be empowered to browse, search and explore the structure information available on the Data Web in a fast and user friendly manner (*Search, Browsing & Exploration*).

These different stages of the linked data life-cycle do not exist in isolation or are passed in a strict sequence, but mutually fertilize themselves. Examples include the following:

- The detection of mappings on the schema level, will directly affect instance level matching and vice versa.

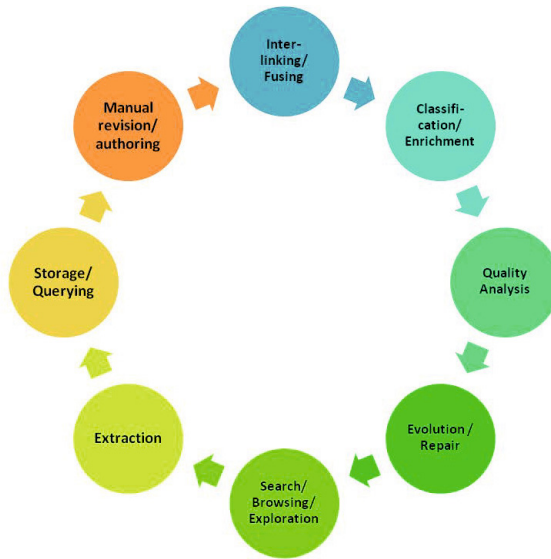


Fig. 2. The Linked Data life-cycle

- Ontology schema mismatches between knowledge bases can be compensated for by learning which concepts of one are equivalent to which concepts of the other knowledge base.
- Feedback and input from end users can be taken as training input (i.e. as positive or negative examples) for machine learning techniques in order to perform inductive reasoning on larger knowledge bases, whose results can again be assessed by end users for iterative refinement.
- Semantically-enriched knowledge bases improve the detection of inconsistencies and modelling problems, which in turn results in benefits for interlinking, fusion, and classification.
- The querying performance of the RDF data management directly affects all other components and the nature of queries issued by the components affects the RDF data management.

As a result of such interdependence, we envision the Web of Linked Data to realize an improvement cycle for knowledge bases, in which an improvement of a knowledge base with regard to one aspect (e.g. a new alignment with another interlinking hub) triggers a number of possible further improvements (e.g. additional instance matches).

The use of Linked Data offers a number of significant benefits:

- *Uniformity.* All datasets published as Linked Data share a uniform data model, the RDF statement data model. With this data model all information is represented in facts expressed as triples consisting of a subject, predicate and object. The elements used in subject, predicate or object positions are mainly globally unique IRI/URI entity identifiers. At the object position also literals, i.e. typed data values can be used.

Table 1. Juxtaposition of the concepts Linked Data, Linked Open Data and Open Data

Representation \ degree of openness	Possibly closed	Open (cf. opendefinition.org)
<i>Structured data model</i> (i.e. XML, CSV, SQL etc.)	Data	Open Data
<i>RDF data model</i> (published as Linked Data)	Linked Data (LD)	Linked Open Data (LOD)

- *De-referencability.* URIs are not just used for identifying entities, but since they can be used in the same way as URLs they also enable locating and retrieving resources describing and representing these entities on the Web.
- *Coherence.* When an RDF triple contains URIs from different namespaces in subject and object position, this triple basically establishes a link between the entity identified by the subject (and described in the source dataset using namespace A) with the entity identified by the object (described in the target dataset using namespace B). Through the typed RDF links, data items are effectively interlinked.
- *Integrability.* Since all Linked Data sources share the RDF data model, which is based on a single mechanism for representing information, it is very easy to attain a syntactic and simple semantic integration of different Linked Data sets. A higher level semantic integration can be achieved by employing schema and instance matching techniques and expressing found matches again as alignments of RDF vocabularies and ontologies in terms of additional triple facts.
- *Timeliness.* Publishing and updating Linked Data is relatively simple thus facilitating a timely availability. In addition, once a Linked Data source is updated it is straightforward to access and use the updated data source, since time consuming and error prone extraction, transformation and loading is not required.

The development of research approaches, standards, technology and tools for supporting the Linked Data lifecycle data is one of the main challenges. Developing adequate and pragmatic solutions to these problems can have a substantial impact on science, economy, culture and society in general. The publishing, integration and aggregation of statistical and economic data, for example, can help to obtain a more precise and timely picture of the state of our economy. In the domain of health care and life sciences making sense of the wealth of structured information already available on the Web can help to improve medical information systems and thus make health care more adequate and efficient. For the media and news industry, using structured background information from the Data Web for enriching and repurposing the quality content can facilitate the creation of new publishing products and services. Linked Data technologies can help to increase the flexibility, adaptability and efficiency of information management in organizations, be it companies, governments and public administrations or online communities. For end-users and society in general, the Data Web will help to obtain and integrate required information more efficiently and thus successfully manage the transition towards a knowledge-based economy and an information society.

Structure of this chapter. This chapter aims to explain the foundations of Linked Data and introducing the different aspects of the Linked Data lifecycle by highlighting a particular approach and providing references to related work and further reading. We start by briefly explaining the principles underlying the Linked Data paradigm in Section 2. The first aspect of the Linked Data lifecycle is the extraction of information from unstructured, semi-structured and structured sources and their representation according to the RDF data model (Section 3). We present the user friendly authoring and manual revision aspect of Linked Data with the example of Semantic Wikis in Section 4. The interlinking aspect is tackled in Section 5 and gives an overview on the LIMES framework. We describe how the instance data published and commonly found on the Data Web can be enriched with higher level structures in Section 6. We present an overview of the various data quality dimensions and metrics along with currently existing tools for data quality assessment of Linked Data in Section 7. Due to space limitations we omit a detailed discussion of the evolution as well as search, browsing and exploration aspects of the Linked Data lifecycle in this chapter. The chapter is concluded by several sections on promising applications of Linked Data and semantic technologies, in particular Open Governmental Data, Semantic Business Intelligence and Statistical and Economic Data.

Overall, this is an updated version of a similar lecture given at Reasoning Web Summer School 2011 [13].

2 The Linked Data Paradigm

In this section we introduce the basic principles of Linked Data. The section is partially based on the Section 2 from [54]. The term Linked Data refers to a set of best practices for publishing and interlinking structured data on the Web. These best practices were introduced by Tim Berners-Lee in his Web architecture note Linked Data¹ and have become known as the Linked Data principles. These principles are:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
- Include links to other URIs, so that they can discover more things.

The basic idea of Linked Data is to apply the general architecture of the World Wide Web [71] to the task of sharing structured data on global scale. The Document Web is built on the idea of setting hyperlinks between Web documents that may reside on different Web servers. It is built on a small set of simple standards: Uniform Resource Identifiers (URIs) and their extension Internationalized Resource Identifiers (IRIs) as globally unique identification mechanism [21], the Hypertext Transfer Protocol (HTTP) as universal access mechanism [44], and the Hypertext Markup Language (HTML) as a widely used content format [65]. Linked Data builds directly on Web architecture and applies this architecture to the task of sharing data on global scale.

¹ <http://www.w3.org/DesignIssues/LinkedData.html>

2.1 Resource Identification with IRIs

To publish data on the Web, the data items in a domain of interest must first be identified. These are the things whose properties and relationships will be described in the data, and may include Web documents as well as real-world entities and abstract concepts. As Linked Data builds directly on Web architecture, the Web architecture term *resource* is used to refer to these *things of interest*, which are in turn identified by HTTP URIs. Linked Data uses only HTTP URIs, avoiding other URI schemes such as URNs [101] and DOIs². The structure of HTTP URIs looks as follows:

[scheme:][//authority][path][?query][#fragment]

A URI for identifying Shakespeare's 'Othello', for example, could look as follows:

`http://de.wikipedia.org/wiki/Othello#id`

HTTP URIs make good names for two reasons:

1. They provide a simple way to create globally unique names in a decentralized fashion, as every owner of a domain name or delegate of the domain name owner may create new URI references.
2. They serve not just as a name but also as a means of accessing information describing the identified entity.

2.2 De-referencability

Any HTTP URI should be de-referencable, meaning that HTTP clients can look up the URI using the HTTP protocol and retrieve a description of the resource that is identified by the URI. This applies to URIs that are used to identify classic HTML documents, as well as URIs that are used in the Linked Data context to identify real-world objects and abstract concepts. Descriptions of resources are embodied in the form of Web documents. Descriptions that are intended to be read by humans are often represented as HTML. Descriptions that are intended for consumption by machines are represented as RDF data. Where URIs identify real-world objects, it is essential to not confuse the objects themselves with the Web documents that describe them. It is therefore common practice to use different URIs to identify the real-world object and the document that describes it, in order to be unambiguous. This practice allows separate statements to be made about an object and about a document that describes that object. For example, the creation year of a painting may be rather different to the creation year of an article about this painting. Being able to distinguish the two through use of different URIs is critical to the consistency of the Web of Data.

The Web is intended to be an information space that may be used by humans as well as by machines. Both should be able to retrieve representations of resources in a form that meets their needs, such as HTML for humans and RDF for machines. This can be achieved using an HTTP mechanism called content negotiation [44]. The basic

² <http://www.doi.org/hb.html>

idea of content negotiation is that HTTP clients send HTTP headers with each request to indicate what kinds of documents they prefer. Servers can inspect these headers and select an appropriate response. If the headers indicate that the client prefers HTML then the server will respond by sending an HTML document. If the client prefers RDF, then the server will send the client an RDF document.

There are two different strategies to make URIs that identify real-world objects de-referencable [137]. Both strategies ensure that objects and the documents that describe them are not confused and that humans as well as machines can retrieve appropriate representations.

303 URIs. Real-world objects can not be transmitted over the wire using the HTTP protocol. Thus, it is also not possible to directly de-reference URIs that identify real-world objects. Therefore, in the 303 URI strategy, instead of sending the object itself over the network, the server responds to the client with the HTTP response code 303 See Other and the URI of a Web document which describes the real-world object. This is called a *303 redirect*. In a second step, the client de-references this new URI and retrieves a Web document describing the real-world object.

Hash URIs. A widespread criticism of the 303 URI strategy is that it requires two HTTP requests to retrieve a single description of a real-world object. One option for avoiding these two requests is provided by the hash URI strategy. The hash URI strategy builds on the characteristic that URIs may contain a special part that is separated from the base part of the URI by a hash symbol (#). This special part is called the fragment identifier. When a client wants to retrieve a hash URI the HTTP protocol requires the fragment part to be stripped off before requesting the URI from the server. This means a URI that includes a hash cannot be retrieved directly, and therefore does not necessarily identify a Web document. This enables such URIs to be used to identify real-world objects and abstract concepts, without creating ambiguity [137].

Both approaches have their advantages and disadvantages. Section 4.4. of the W3C Interest Group Note *Cool URIs for the Semantic Web* compares the two approaches [137]: Hash URIs have the advantage of reducing the number of necessary HTTP round-trips, which in turn reduces access latency. The downside of the hash URI approach is that the descriptions of all resources that share the same non-fragment URI part are always returned to the client together, irrespective of whether the client is interested in only one URI or all. If these descriptions consist of a large number of triples, the hash URI approach can lead to large amounts of data being unnecessarily transmitted to the client. 303 URIs, on the other hand, are very flexible because the redirection target can be configured separately for each resource. There could be one describing document for each resource, or one large document for all of them, or any combination in between. It is also possible to change the policy later on.

2.3 RDF Data Model

The RDF data model [1] represents information as sets of statements, which can be visualized as node-and-arc-labeled directed graphs. The data model is designed for the

integrated representation of information that originates from multiple sources, is heterogeneously structured, and is represented using different schemata. RDF can be viewed as a *lingua franca*, capable of moderating between other data models that are used on the Web.

In RDF, information is represented in statements, called RDF triples. The three parts of each triple are called its subject, predicate, and object. A triple mimics the basic structure of a simple sentence, such as for example:

Burkhard Jung	is the mayor of	Leipzig
(subject)	(predicate)	(object)

The following is the formal definition of RDF triples as it can be found in the W3C RDF standard [1].

Definition 1 (RDF Triple). Assume there are pairwise disjoint infinite sets I , B , and L representing IRIs, blank nodes, and RDF literals, respectively. A triple $(v_1, v_2, v_3) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an RDF triple. In this tuple, v_1 is the subject, v_2 the predicate and v_3 the object. We denote the union $I \cup B \cup L$ by T called RDF terms.

The main idea is to use IRIs as identifiers for entities in the subject, predicate and object positions in a triple. Data values can be represented in the object position as literals. Furthermore, the RDF data model also allows in subject and object positions the use of identifiers for unnamed entities (called blank nodes), which are not globally unique and can thus only be referenced locally. However, the use of blank nodes is discouraged in the Linked Data context as we discuss below. Our example fact sentence about Leipzig's mayor would now look as follows:

<code><http://leipzig.de/id></code>	<code><http://example.org/p/hasMayor></code>	<code><http://Burkhard-Jung.de/id></code> .
(subject)	(predicate)	(object)

This example shows, that IRIs used within a triple can originate from different namespaces thus effectively facilitating the mixing and mashing of different RDF vocabularies and entities from different Linked Data knowledge bases. A triple having identifiers from different knowledge bases at subject and object position can be also viewed as an typed link between the entities identified by subject and object. The predicate then identifies the type of link. If we combine different triples we obtain an RDF graph.

Definition 2 (RDF Graph). A finite set of RDF triples is called RDF graph. The RDF graph itself represents an resource, which is located at a certain location on the Web and thus has an associated IRI, the graph IRI.

An example of an RDF graph is depicted in Figure 3. Each unique subject or object contained in the graph is visualized as a node (i.e. oval for resources and rectangle for literals). Predicates are visualized as labeled arcs connecting the respective nodes. There are a number of synonyms being used for RDF graphs, all meaning essentially the same but stressing different aspects of an RDF graph, such as *RDF document* (file perspective), *knowledge base* (collection of facts), *vocabulary* (shared terminology), *ontology* (shared logical conceptualization).

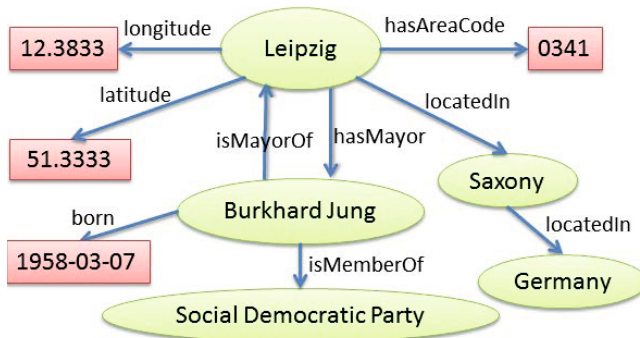


Fig. 3. Example RDF graph containing 9 triples describing the city of Leipzig and its mayor

Problematic RDF features in the Linked Data Context. Besides the features mentioned above, the RDF Recommendation [1] also specifies some other features. In order to make it easier for clients to consume data only the subset of the RDF data model described above should be used. In particular, the following features are problematic when publishing RDF as Linked Data:

- *RDF reification* (for making statements about statements) should be avoided if possible, as reified statements are rather cumbersome to query with the SPARQL query language. In many cases using reification to publish metadata about individual RDF statements can be avoided by attaching the respective metadata to the RDF document containing the relevant triples.
- *RDF collections* and *RDF containers* are also problematic if the data needs to be queried with SPARQL. Therefore, in cases where the relative ordering of items in a set is not significant, the use of multiple triples with the same predicate is recommended.
- The scope of *blank nodes* is limited to the document in which they appear, meaning it is not possible to create links to them from external documents. In addition, it is more difficult to merge data from different sources when blank nodes are used, as there is no URI to serve as a common key. Therefore, all resources in a data set should be named using IRI references.

2.4 RDF Serializations

The initial official W3C RDF standard [1] comprised a serialization of the RDF data model in XML called *RDF/XML*. Its rationale was to integrate RDF with the existing XML standard, so it could be used smoothly in conjunction with the existing XML technology landscape. Unfortunately, *RDF/XML* turned out to be rather difficult to understand for the majority of potential users, since it requires to be familiar with two data models (i.e. the tree-oriented XML data model as well as the statement oriented RDF datamodel) and interactions between them, since RDF statements are represented in XML. As a consequence, with *N-Triples*, *Turtle* and *N3* a family of alternative

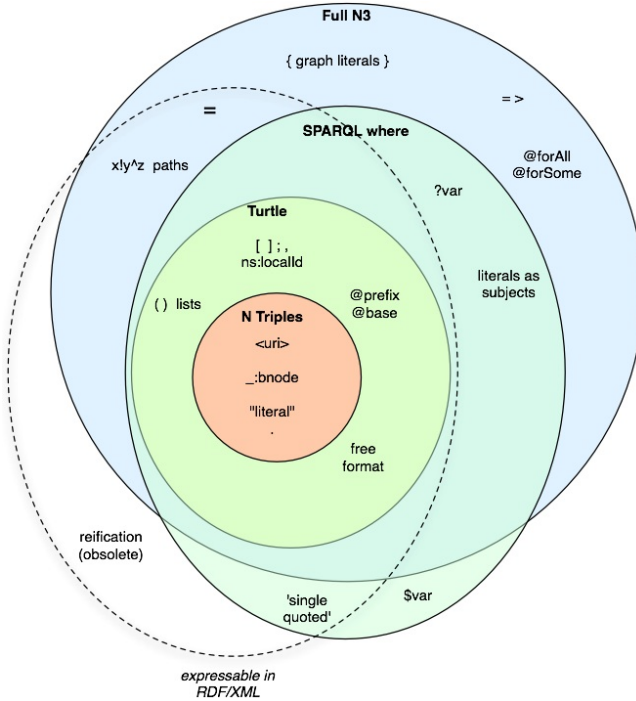


Fig. 4. Various textual RDF serializations as subsets of N3 (from [20])

text-based RDF serializations was developed, whose members have the same origin, but balance differently between readability for humans and machines. Later in 2009, *RDFa* (RDF Annotations, [2]) was standardized by the W3C in order to simplify the integration of HTML and RDF and to allow the joint representation of structured and unstructured content within a single source HTML document. Another RDF serialization, which is particularly beneficial in the context of JavaScript web applications and mashups is the serialization of RDF in JSON. In the sequel we present each of these RDF serializations in some more detail. Figure 5 presents an example serialized in the most popular serializations.

N-Triples. This serialization format was developed specifically for RDF graphs. The goal was to create a serialization format which is very simple. N-Triples are easy to parse and generate by software. An N-Triples document consists of a set of triples, which are separated ‘.’ (lines 1-2, 3-4 and 5-6 in Figure 5 contain one triple each). URI components of a triple are written in full and enclosed by ‘<’ and ‘>’. Literals are enclosed in quotes, datatypes can be appended to a literal using ‘g’ (line 6), language tags using ‘@’ (line 4). They are a subset of *Notation 3* and *Turtle* but lack, for example, shortcuts such as CURIEs. This makes them less readable and more difficult to create manually. Another disadvantage is that N-triples use only the 7-bit US-ASCII character encoding instead of UTF-8.

Turtle. Turtle (Terse RDF Triple Language) is a subset of, and compatible with, Notation 3 and a superset of the minimal N-Triples format (cf. Figure 4). The goal was to use the essential parts of Notation 3 for the serialization of RDF models and omit everything else. Turtle became part of the SPARQL query language for expressing graph patterns. Compared to N-Triples, Turtle introduces a number of shortcuts, such as namespace definitions (lines 1-5 in Figure 5), the semicolon as a separator between triples sharing the same subject (which then does not have to be repeated in subsequent triples) and the comma as a separator between triples sharing the same subject and predicate. Turtle, just like Notation 3, is human-readable, and can handle the "%" character in URIs (required for encoding special characters) as well as IRIs due to its UTF-8 encoding.

Notation 3. N3 (Notation 3) was devised by Tim Berners-Lee and developed for the purpose of serializing RDF. The main aim was to create a very human-readable serialization. Hence, an RDF model serialized in N3 is much more compact than the same model in RDF/XML but still allows a great deal of expressiveness even going beyond the RDF data model in some aspects. Since, the encoding for N3 files is UTF-8 the use of IRIs does not pose a problem.

RDF/XML. The RDF/XML syntax [98] is standardized by the W3C and is widely used to publish Linked Data on the Web. However, the syntax is also viewed as difficult for humans to read and write, and therefore consideration should be given to using other serializations in data management and curation workflows that involve human intervention, and to the provision of alternative serializations for consumers who may wish to eyeball the data. The MIME type that should be used for RDF/XML within HTTP content negotiation is `application/rdf+xml`.

RDFa. RDF in Attributes (RDFa, [2]) was developed for embedding RDF into XHTML pages. Since it is an extension to the XML based XHTML, UTF-8 and UTF-16 are used for encoding. The "%" character for URIs in triples can be used because RDFa tags are not used for a part of a RDF statement. Thus IRIs are usable, too. Because RDFa is embedded in XHTML, the overhead is higher compared to other serialization technologies and also reduces the readability. The basic idea of RDFa is enable an RDFa processor to extract RDF statements from an RDFa enriched HTML document. This is achieved by defining the scope of a certain resource description, for example, using the 'about' attribute (cf. line 10 in Figure 5). Within this scope, triples can now be extracted from links having an additional 'rel' attribute (line 13) or other tags having a 'property attribute' (lines 11 and 14).

JSON-LD. JavaScript Object Notation (JSON) was developed for easy data interchange between applications. JSON, although carrying JavaScript in its name and being a subset of JavaScript, meanwhile became a language independent format which can be used for exchanging all kinds of data structures and is widely supported in different programming languages. Compared to XML, JSON-LD requires less overhead with regard to parsing and serializing. JSON-LD has been developed by the JSON for Linking Data Community Group and been transferred to the RDF Working Group for review, improvement, and publication along the Recommendation track. JSON-LD's design goals

N-Triples

```

1 <http://dbpedia.org/resource/Leipzig> <http://dbpedia.org/property/hasMayor>
2 <http://dbpedia.org/resource/Burkhard_Jung> .
3 <http://dbpedia.org/resource/Leipzig> <http://www.w3.org/2000/01/rdf-schema#label>
4 "Leipzig"@de .
5 <http://dbpedia.org/resource/Leipzig> <http://www.w3.org/2003/01/geo/wgs84_pos#lat>
6 "51.333332"^^<http://www.w3.org/2001/XMLSchema#float> .

```

Turtle

```

1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix rdfs="http://www.w3.org/2000/01/rdf-schema#" .
3 @prefix dbp="http://dbpedia.org/resource/" .
4 @prefix dbpp="http://dbpedia.org/property/" .
5 @prefix geo="http://www.w3.org/2003/01/geo/wgs84_pos#" .
6
7 dbp:Leipzig dbpp:hasMayor dbp:Burkhard_Jung ;
8 rdfs:label "Leipzig"@de ;
9 geo:lat "51.333332"^^xsd:float .

```

RDF/XML

```

1 <?xml version="1.0"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4 xmlns:dbpp="http://dbpedia.org/property/"
5 xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
6 <rdf:Description rdf:about="http://dbpedia.org/resource/Leipzig">
7 <property:hasMayor rdf:resource="http://dbpedia.org/resource/Burkhard_Jung" />
8 <rdfs:label xml:lang="de">Leipzig</rdfs:label>
9 <geo:lat rdf:datatype="http://www.w3.org/2001/XMLSchema#float">51.3333</geo:lat>
10 </rdf:Description>
11 </rdf:RDF>

```

RDFa

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
3 "http://www.w3.org/Markup/DTD/xhtml-rdfa-1.dtd">
4 <html version="XHTML+RDFa 1.0" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml"
5 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
6 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
7 xmlns:dbpp="http://dbpedia.org/property/"
8 xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
9 <head><title>Leipzig</title></head>
10 <body about="http://dbpedia.org/resource/Leipzig">
11 <h1 property="rdfs:label" xml:lang="de">Leipzig</h1>
12 <p>Leipzig is a city in Germany. Leipzig's mayor is
13 <a href="Burkhard_Jung" rel="dbpp:hasMayor">Burkhard Jung</a>. It is located
14 at latitude <span property="geo:lat" datatype="xsd:float">51.3333</span>.</p>
15 </body>
16 </html>

```

JSON-LD

```

1 {
2   "@context": {
3     "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
4     "hasMayor": { "@id": "http://dbpedia.org/property/hasMayor", "@type": "@id" },
5     "Person": "http://xmlns.com/foaf/0.1/Person",
6     "lat": "http://www.w3.org/2003/01/geo/wgs84_pos#lat"
7   },
8   "@id": "http://dbpedia.org/resource/Leipzig",
9   "rdfs:label": "Leipzig",
10  "hasMayor": "http://dbpedia.org/resource/Burkhard_Jung",
11  "lat": { "@value": "51.3333", "@type": "http://www.w3.org/2001/XMLSchema#float"
12 }

```

Fig. 5. Different RDF serializations of three triples from Figure 3

are simplicity, compatibility, expressiveness, terseness, zero edits and one-pass processing. As a result, JSON-LD documents are basically standard attribute-value JSON documents with an additional context section (lines 2-7 in Figure 5) establishing mappings to RDF vocabularies. Text in JSON and, thus, also RDF resource identifiers are encoded in Unicode and hence can contain IRIs.

3 Extraction

Information represented in unstructured form or adhering to a different structured representation formalism must be mapped to the RDF data model in order to be used within the Linked Data life-cycle. In this section, we give an overview on some relevant approaches for extracting RDF from unstructured and structured sources.

3.1 From Unstructured Sources

The extraction of structured information from unstructured data sources (especially text) has been a central pillar of *natural language processing* (NLP) and *Information Extraction* (IE) for several decades. With respect to the extraction of RDF data from unstructured data, three sub-disciplines of NLP play a central role: *Named Entity Recognition* (NER) for the extraction of entity labels from text, *Keyword/Keyphrase Extraction* (KE) for the recognition of central topics and *Relationship Extraction* (RE, also called relation mining) for mining the properties which link the entities and keywords described in the data source. A noticeable additional task during the migration of these techniques to Linked Data is the extraction of suitable IRIs for the discovered entities and relations, a requirement that was not needed before. In this section, we give a short overview of approaches that implement the required NLP functionality. Then we present a framework that applies machine learning to boost the quality of the RDF extraction from unstructured data by merging the results of NLP tools. As an orthogonal activity, we want to mention the NLP2RDF project [58], which provides RDF serialisation for NLP tools solving the above mentioned tasks.

Named Entity Recognition. The goal of NER is to discover instances of a predefined classes of entities (e.g., persons, locations, organizations) in text. NER tools and frameworks implement a broad spectrum of approaches, which can be subdivided into three main categories: dictionary-based, rule-based, and machine-learning approaches. The first systems for NER implemented dictionary-based approaches, which relied on a list of NEs and tried to identify these in text [157,5]. Following work that showed that these approaches did not perform well for NER tasks such as recognizing proper names [136], rule-based approaches were introduced. These approaches rely on hand-crafted rules [32,146] to recognize NEs. Most rule-based approaches combine dictionary and rule-based algorithms to extend the list of known entities. Nowadays, handcrafted rules for recognizing NEs are usually implemented when no training examples are available for the domain or language to process [106].

When training examples are available, the methods of choice are borrowed from supervised machine learning. Approaches such as Hidden Markov Models [168], Maximum Entropy Models [35] and Conditional Random Fields [45] have been applied to the NER task. Due to scarcity of large training corpora as necessitated by machine learning approaches, semi-supervised [125,105] and unsupervised machine learning approaches [107,41] have also been used for extracting NER from text. [105] gives an exhaustive overview of approaches for NER.

Keyphrase Extraction. Keyphrases/Keywords are multi-word units (MWUs) which capture the main topics of a document. The automatic detection of such MWUs has been an important task of NLP for decades but due to the very ambiguous definition of what an appropriate keyword should be, current approaches to the extraction of keyphrases still display low F-scores [75]. From the point of view of the Semantic Web, the extraction of keyphrases is a very similar task to that of finding tags for a given document. Several categories of approaches have been adapted to enable KE, of which some originate from research areas such as summarization and information retrieval (IR). Still, according to [74], the majority of the approaches to KE implement combinations of statistical, rule-based or heuristic methods [48,120] on mostly document [97], keyphrase [149] or term cohesion features [124]. [75] gives a overview of current tools for KE.

Relation Extraction. The extraction of relations from unstructured data builds upon work for NER and KE to determine the entities between which relations might exist. Most tools for RE rely on pattern-based approaches. Some early work on pattern extraction relied on supervised machine learning [51]. Yet, such approaches demanded large amount of training data, making them difficult to adapt to new relations. The subsequent generation of approaches to RE aimed at bootstrapping patterns based on a small number of input patterns and instances. For example, [28] presents the Dual Iterative Pattern Relation Expansion (DIPRE) and applies it to the detection of relations between authors and titles of books. This approach relies on a small set of seed patterns to maximize the precision of the patterns for a given relation while minimizing their error rate of the same patterns. Snowball [3] extends DIPRE by a new approach to the generation of seed tuples. Newer approaches aim to either collect redundancy information from the whole Web [123] or Wikipedia [158,164] in an unsupervised manner or to use linguistic analysis [53,119] to harvest generic patterns for relations.

URI Disambiguation. One important problem for the integration of NER tools for Linked Data is the retrieval of IRIs for the entities to be manipulated. In most cases, the URIs can be extracted from generic knowledge bases such as DBpedia [104,83] by comparing the label found in the input data with the `rdfs:label` or `dc:title` of the entities found in the knowledge base. Furthermore, information such as the type of NEs can be used to filter the retrieved IRIs via a comparison of the `rdfs:label` of the `rdf:type` of the URIs with the name of class of the NEs. Still in many cases (e.g., Leipzig, Paris), several entities might bear the same label.

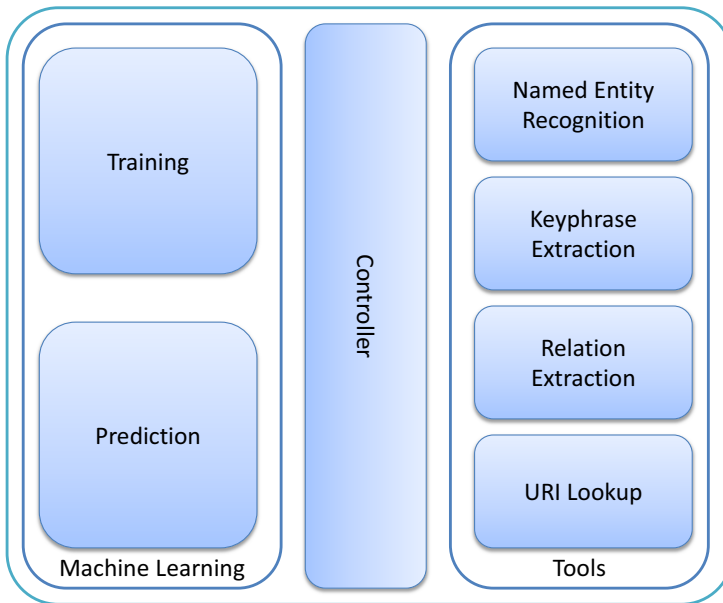


Fig. 6. FOX Architecture

Unsupervised Extraction Example: The FOX Framework

Several frameworks have been developed to implement the functionality above for the Data Web including OpenCalais³ and Alchemy⁴. Yet, these tools rely mostly on one approach to perform the different tasks at hand. In this section, we present the FOX (Federated knOwledge eXtraction) framework⁵, which makes use of the diversity of the algorithms available for NER, KE and RE to generate high-quality RDF.

The architecture of FOX consists of *three main layers* as shown in Figure 6. The *machine learning* layer implements interfaces for accommodating ensemble learning techniques such as simple veto algorithms but also neural networks. It consists of *two main modules*. The *training module* allows to load training data so as to enable FOX to learn the best combination of tools and categories for achieving superior recall and precision on the input training data. Depending on the training algorithm used, the user can choose to tune the system for either precision or recall. When using neural networks for example, the user can decide to apply a higher threshold for the output neurons, thus improving the precision but potentially limiting the recall. The *prediction module* allows to run FOX by loading the result of a training session and processing the input data according to the tool-category combination learned during the training phase. Note that the same learning approach can be applied to NER, KE, RE and URI lookup as they all can be modelled as classification tasks.

³ <http://www.opencalais.com>

⁴ <http://www.alchemyapi.com>

⁵ <http://aksw.org/projects/fox>

The second layer of FOX is the *controller*, which coordinates the access to the modules that carry out the language processing. The controller is aware of each of the modules in its backend and carries out the initialisation of these modules once FOX is started. Furthermore, it collects the results from the backend modules and invokes the results of a training instance to merge the results of these tools.

The final layer of FOX is the *tool layer*, wherein all NLP tools and services integrated in FOX can be found. It is important to notice that the tools per se are not trained during the learning phase of FOX. Rather, we learn of the models already loaded in the tools to allow for the best prediction of named entities in a given domain.

The ensemble learning implemented by FOX was evaluated in the task of NER by integrating three NER tools (Stanford NER, Illinois NER and a commercial tool) and shown to lead to an improvement of more than 13% in F-Score (see Figure 7) when combining three tools, therewith even outperforming commercial systems.

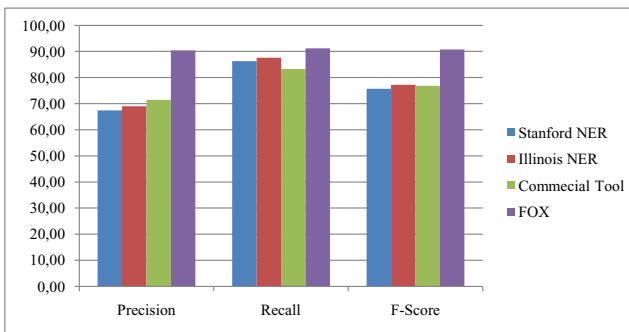


Fig. 7. Comparison of precision, recall and F-score of the best runs of FOX and its components on NER

3.2 From Structured Sources

Structured knowledge, e.g. relational databases and XML, is the backbone of many (web) applications. Extracting or converting this knowledge to RDF is a long-standing research goal in the Semantic Web community. A conversion to RDF allows to integrate the data with other sources and perform queries over it. In this lecture, we focus on the conversion of relational databases to RDF (see Figure 8). In the first part, we summarize material from a recent relational database to RDF (RDB2RDF) project report. After that, we describe the mapping language R2RML, which is a language for expressing database to RDF conversion mappings. While we focus on relational data, we also want to note that extraction from CSV files is also highly important as illustrated in use cases in the financial [96] and health sector [165,166].

Triplify and RDB2RDF Survey Report. The table displayed in Figure 9 is taken from the Triplify WWW paper [8]. The survey report [135] furthermore contained a chart (see Figure 10) showing the reference framework for classifying the approaches and an extensive table classifying the approaches (see Figure 11). Another recent survey is [144].

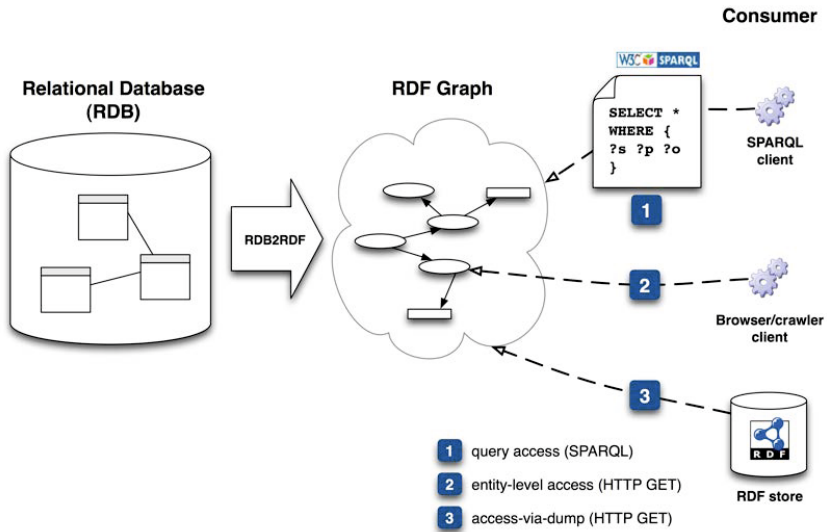


Fig. 8. Illustration of RDB to RDF conversion.

Source: <http://www.w3.org/2001/sw/rdb2rdf/use-cases/>.

The following criteria can be extracted:

Automation Degree. Degree of mapping creation automation.

Values: Manual, Automatic, Semi-Automatic.

Domain or Database Semantics Driven. Some approaches are tailored to model a domain, sometimes with the help of existing ontologies, while others attempt to extract domain information primarily from the given database schema with few other resources used (domain or database semantics-driven). The latter often results in a table-to-class, column-to-predicate mapping. Some approaches also use a (semi) automatic approach based on the database, but allow manual customization to model domain semantics.

Approach	Automation (a)	Domain or database semantics-driven (b)	Access paradigm (c)	Mapping language (d)	Domain reliance (e)
Dartgrid [17]	Manual	Domain	SPARQL	Visual Tool	dependent
Hu et al. [11]	Auto	Both	ETL	intern	dependent
Tirmizi et al. [16]	Auto	DB	ETL	FOL	general
Li et al. [12]	Semi	DB	ETL	n/a	general
DB2OWL[10]	Semi	DB	SPARQL	R2O	general/dependent
RDBToOnto [6]	Semi	DB+M	ETL	Visual Tool	general
Sahoo et al. [15]	Manual	Domain	ETL	XSLT	dependent
R2O[13]	Manual	DB+M	SPARQL	R2O	dependent
D2RQ[4]	Auto	DB+M	LD, SPARQL	D2RQ	general
Virtuoso RDF View [5, 9]	Semi	DB+M	SPARQL	own	general
Triplify	Manual	Domain	LD	SQL	general

Table 4: An integrated overview of mapping approaches. Criteria for classification were merged, some removed, fields were completed, when missing. DB+M means that the semi-automatic approach can later be customized manually

Fig. 9. Table comparing relevant approaches from [8]

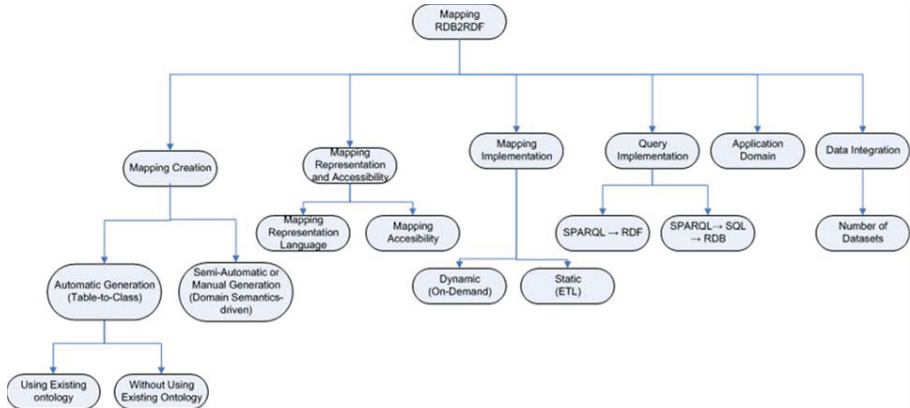


Fig. 10. Reference framework by [135]

Values: Domain, DB (database), DB+M (database and later manual customisation), Both (Domain and DB)

Access Paradigm. Resulting access paradigm (ETL [extract transform load], Linked Data, SPARQL access). Note that the access paradigm also determines whether the resulting RDF model updates automatically. ETL means a one time conversion, while Linked Data and SPARQL always process queries versus the original database.

Values: SPARQL, ETL, LD

Mapping Language. The used mapping language as an important factor for reusability and initial learning cost.

Values: Visual Tool, intern (internal self-designed language), FOL, n/a (no information available), R2O, XSLT, D2RQ, proprietary, SQL

Domain reliance. Domain reliance (general or domain-dependent): requiring a pre-defined ontology is a clear indicator of domain dependency.

Values: Dependent, General

Type. Although not used in the table the paper discusses four different classes:

Values: Alignment, Database Mining, Integration, Languages/Servers

R2RML - RDB to RDF Mapping Language. The R2RML W3C recommendation⁶ specifies an RDF notation for mapping relational tables, views or queries into RDF. The primary area of applicability of this is extracting RDF from relational databases, but in special cases R2RML could lend itself to on-the-fly translation of SPARQL into SQL or to converting RDF data to a relational form. The latter application is not the

⁶ <http://www.w3.org/TR/r2rml/>

PROJECTS	MAPPING CREATION	MAPPING REPRESENTATION AND ACCESSIBILITY		MAPPING IMPLEMENTATION	QUERY IMPLEMENTATION	APPLICATION DOMAIN	DATA INTEGRATION	
	Automatic (Table-to-Class) or Manual/Semi-Automatic (Domain Semantics-driven)	Representation Language	Mapping Access	Static (ETL) or Dynamic	SPARQL→RDF or SPARQL→SQL→RDB		Yes/No	Number of Datasets
1. Hu et al, 2007	Automatic (with use of existing ontology)	First Order Logic formulae or Horn Clauses	Files	None Specified	None Specified	Generic	Enables (through contextual mappings)	Potentially Multiple
2. Kashyap et al, 2007	Manual/Semi-Automatic (Domain Semantics-driven)	Mediator Framework Classes	Mapping mediator	Dynamic	SPARQL→SQL→RDB	Life Sciences	Enables	Potentially Multiple
3. DB2OWL (Cullot et al, 2007)	Automatic (Table to Class)	R2O language	R2O mapping document		SPARQL→SQL→RDB	Generic	Enables	Potentially Multiple
4. Tirmizi et al 2008	Automatic (Table to Class, SQL-DDL to RDF)	First Order Logic	None specified	Static	None Specified	Generic	No	None
5. SOAM (Li et al, 2005)	Automatic (Table to Class) with user input	Logic Rules	Implemented as part of system	Static	Potentially SPARQL (on generated populated ontology)	Generic (Case Study: Economics)	No	None
6. Sahoo et al, 2008	Manual/Semi-Automatic (Domain Semantics-driven)	XPath expressions	XSLT document	Static	SPARQL	Life Sciences	Yes	Test included five (Gene, Biological Pathway)
7. Byrne, 2008	Manual/Semi-Automatic (Domain Semantics-driven)	SKOS vocabulary	RDF document	Static	SPARQL	Cultural Heritage	No	None
8. Green et al, 2008	Manual/Semi-Automatic (Domain Semantics-driven)	D2RQ language	D2RQ mapping file	Dynamic	SPARQL→SQL→RDB	Ordnance Survey	Yes	Multiple
9. Virtuoso RDF View (Blakeley, 2007)	Both (user-specified)	SPASQL-based Meta Schema Language	Quad Storage	Both	Both	Generic	Enables	Potentially Multiple
10. D2RQ (Bizer et al, 2007)	Both (user-specified)	D2RQ language	D2RQ mapping file	Both	Both	Generic	Enables	Potentially Multiple
11. R2O (Barrasa et al, 2006)	Both (user-specified)	R2O language	R2O mapping document	Both	Both	Generic	Enables	Potentially Multiple
12. Dartnoid (Wu et al, 2006)	Automatic (Table to Class)	XML File	Visualized Mapping tool	Dynamic	SPARQL→SQL→RDB (Provide search and query interface)	Life Science (Traditional Chinese Medicine, TCM)	Yes	Test included databases for herb, compound formulas, disease, drug, TCM treatment.
13. RDBtoOnto (Cebah, 2008)	Automatic (Table to Class, allows user intervention)	Constraint rules	Not explicitly stored	Static	Potentially SPARQL (on generated populated ontology)	Generic	No	None
14. Asio Tools	Automatic (Table to	OWL Full based language	File based	Both	SPARQL→SQL→RDB	Generic	Enables	Potentially Multiple

Fig. 11. Comparison of approaches from [135]

primary intended use of R2RML but may be desirable for importing linked data into relational stores. This is possible if the constituent mappings and underlying SQL objects constitute updateable views in the SQL sense.

Data integration is often mentioned as a motivating use case for the adoption of RDF. This integration will very often be between relational databases which have logical entities in common, each with its local schema and identifiers. Thus, we expect to see relational to RDF mapping use cases involving the possibility of a triple coming from multiple sources. This does not present any problem if RDF is being extracted but does lead to complications if SPARQL queries are mapped into SQL. In specific, one will end up with potentially very long queries consisting of joins of unions. Most of the joins between terms of the unions will often be provably empty and can thus be optimized away. This capability however requires the mapping language to be able to express metadata about mappings, i.e. that IRIs coming from one place are always disjoint from IRIs coming from another place. Without such metadata optimizing SPARQL to SQL translation is not possible, which will significantly limit the possibility of querying collections of SQL databases through a SPARQL end point without ETL-ing the mapped RDF into an RDF store.

RDF is emerging as a format for interoperable data publishing. This does not entail that RDF were preferable as a data warehousing model. Besides, for large warehouses, RDF is not cost competitive with relational technology, even though projects such as LOD2 and LDBC expect to narrow this gap (see, e.g., [102,103] for recent SPARQL benchmarks). Thus it follows that on the fly mapping of SPARQL to SQL will be important. Regardless of the relative cost or performance of relational or RDF technology, it is not a feasible proposition to convert relational warehouses to RDF in general, rather existing investments must be protected and reused. Due to these reasons, R2RML will have to evolve in the direction of facilitating querying of federated relational resources.

Supervised Extraction Example: Sparqlify. The challenges encountered with large scale relational data sources LinkedGeoData [12,145] indicate that ETL style approaches based on the conversion of all underlying data to RDF have severe deficiencies. For instance, the RDF conversion process is very time consuming for large-scale, crowdsourced data. Furthermore, changes in data modelling require many changes in the extracted RDF data or the creation of a completely new dump. In summary, the ETL approach is not sufficiently flexible for very large and frequently changing data. It seems preferable to establish virtual RDF views over the existing relational database. In contrast to other tools, such as *D2R* and *Virtuoso RDF views*, Sparqlify converts each SPARQL query to a single SQL query. This allows all optimisations of the underlying database to be applied and can lead to better scalability.

Figure 12 shows the query rewriting workflow in Sparqlify. The rationale of Sparqlify is to leave the schema of the underlying relational database schema unmodified and define RDF views over it. SPARQL queries can then be written against those views, which are expressed in the *Sparqlify-ML* (mapping language). Sparqlify-ML is easy to learn for users, who are experienced in SPARQL and SQL and more compact than other syntactic variants such as R2RML. The left part of Figure 12 shows all steps, which are performed to answer a query. First, the query is converted into an algebra expression.

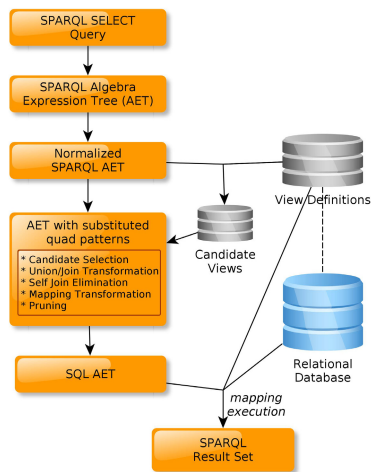


Fig. 12. The Sparqlify concepts and query rewriting workflow

This expression is subsequently converted to a normal form. Given the query patterns, relevant Sparqlify-ML views need to be detected. After this is done, the algebra expression is rewritten to include those relevant views. In a next step, optimisations on the algebra expression are performed to improve efficiency. Finally, this algebra expression can be transformed to an SQL algebra expression. For accomplishing this, we define a general relational algebra for RDB-to-RDF mappings. The SQL query, which was obtained, is executed against the relational database. Using the defined mappings, the SQL result set returned by the relational database can be converted to a SPARQL result set.

All of the above steps are explained in detail throughout the next sections. The main contribution of the Sparqlify project is a formalization, which goes beyond previous work by being capable to push the complete query execution using a single SQL query into the DBMS.

4 Authoring with Semantic Wikis

Semantic Wikis are an extension to conventional, text-based Wikis. While in conventional Wikis pages are stored as blocks of text using a special Wiki markup for structuring the display of the text and adding links to other pages, semantic Wikis aim at adding rich structure to the information itself. To this end, two initially orthogonal approaches have been used: a) extending the markup language to allow semantic annotations and links with meaning or b) building the Wiki software directly with structured information in mind. Nowadays, both approaches have somewhat converged, for instance Semantic MediaWiki [77] also provides forms for entering structured data (see Figure 13). Characteristics of both approaches are summarized in Table 2 for the two prototypical representatives of both approaches, i.e. Semantic MediaWiki and OntoWiki.

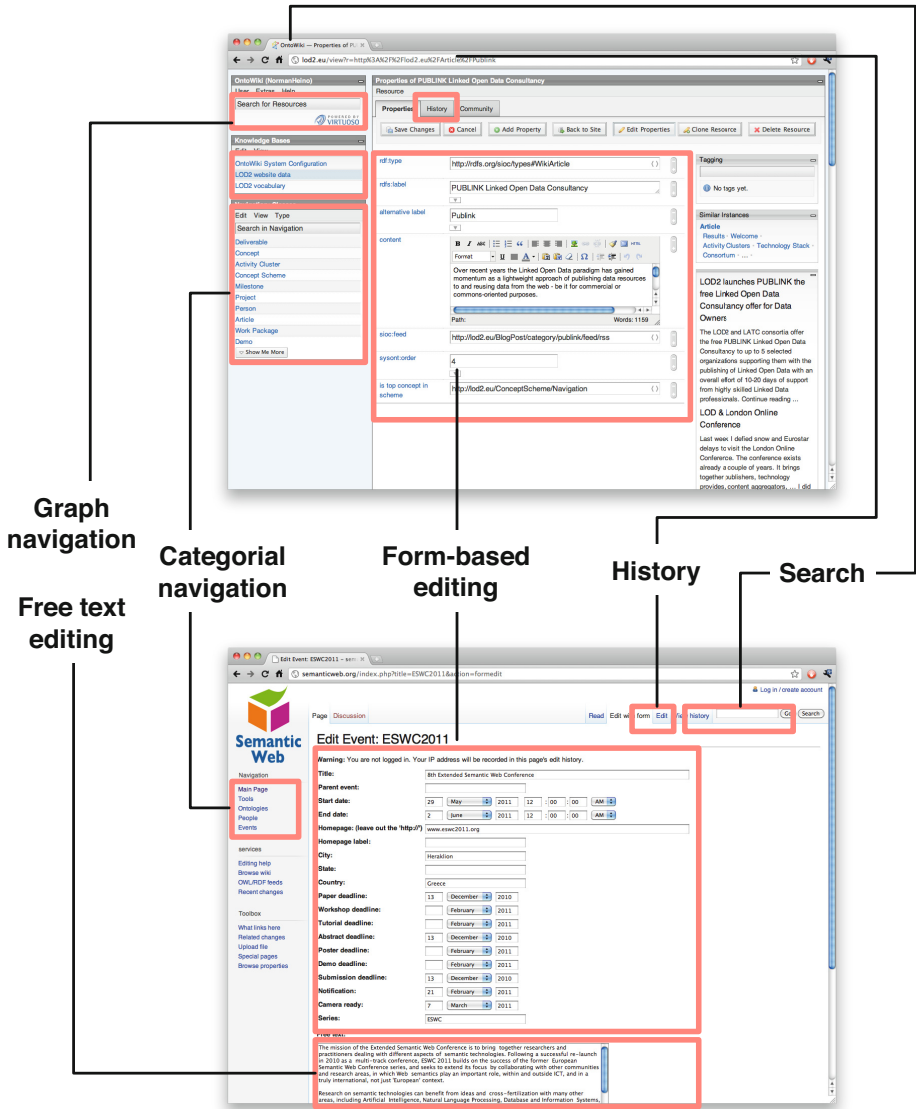


Fig. 13. Comparison of Semantic MediaWiki and OntoWiki GUI building blocks

Table 2. Conceptual differences between Semantic MediaWiki and OntoWiki

	Semantic MediaWiki	OntoWiki
<i>Managed entities</i>	Articles	Resources
<i>Editing</i>	Wiki markup	Forms
<i>Atomic element</i>	Text blob	Statement

Extending Wikis with Semantic Markup. The benefit of a Wiki system comes from the amount of interlinking between Wiki pages. Those links clearly state a relationship between the linked-to and the linking page. However, in conventional Wiki systems this relationship cannot be made explicit. Semantic Wiki systems therefore add a means to specify typed relations by extending the Wiki markup with semantic (i.e. typed) links. Once in place, those links form a knowledge base underlying the Wiki which can be used to improve search, browsing or automatically generated lists and category pages. Examples of approaches for extending Wikis with semantic markup can be found in [77,138,14,122,143]. They represent a straightforward combination of existing Wiki systems and the Semantic Web knowledge representation paradigms. Yet, we see the following obstacles:

Usability: The main advantage of Wiki systems is their unbeatable usability. Adding more and more syntactic possibilities counteracts ease of use for editors.

Redundancy: To allow the answering of real-time queries to the knowledge base, statements have to be additionally kept in a triple store. This introduces a redundancy, which complicates the implementation.

Evolution: As a result of storing information in both Wiki texts and triple store, supporting evolution of knowledge is difficult.

Wikis for Editing Structured Data. In contrast to text-based systems, Wikis for structured data – also called Data Wikis – are built on a structured model of the data being edited. The Wiki software can be used to add instances according to the schema or (in some systems) edit the schema itself. One of those systems is OntoWiki⁷ [9] which bases its data model on RDF. This way, both schema and instance data are represented using the same low-level model (i.e. statements) and can therefore be handled identically by the Wiki.

4.1 OntoWiki - A Semantic Data Wiki

OntoWiki started as an RDF-based data wiki with emphasis on collaboration but has meanwhile evolved into a comprehensive framework for developing Semantic Web applications [55]. This involved not only the development of a sophisticated extension interface allowing for a wide range of customizations but also the addition of several access and consumption interfaces allowing OntoWiki installations to play both a provider and a consumer role in the emerging Web of Data.

⁷ Available at: <http://ontowiki.net>

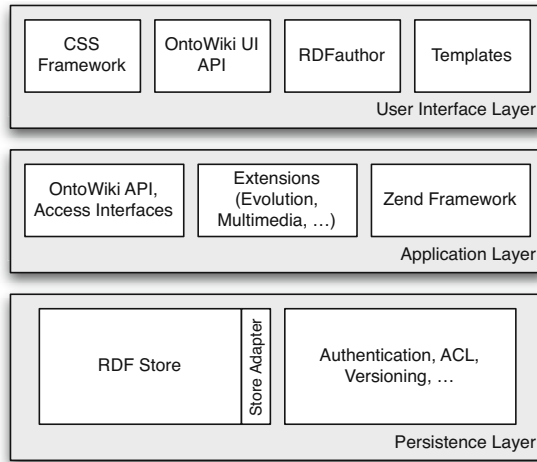


Fig. 14. Overview of OntoWiki’s architecture with extension API and Zend web framework (modified according to [55])

OntoWiki is inspired by classical Wiki systems, its design, however, (as mentioned above) is independent and complementary to conventional Wiki technologies. In contrast to other semantic Wiki approaches, in OntoWiki text editing and knowledge engineering (i. e. working with structured knowledge bases) are not mixed. Instead, OntoWiki directly applies the Wiki paradigm of “making it easy to correct mistakes, rather than making it hard to make them” [90] to collaborative management of structured knowledge. This paradigm is achieved by interpreting knowledge bases as *information maps* where every node is represented visually and interlinked to related resources. Furthermore, it is possible to enhance the knowledge schema gradually as well as the related instance data agreeing on it. As a result, the following requirements and corresponding features characterize OntoWiki:

Intuitive display and editing of instance data should be provided in generic ways, yet enabling domain-specific presentation of knowledge.

Semantic views allow the generation of different views and aggregations of the knowledge base.

Versioning and evolution provides the opportunity to track, review and roll-back changes selectively.

Semantic search facilitates easy-to-use full-text searches on all literal data, search results can be filtered and sorted (using semantic relations).

Community support enables discussions about small information chunks. Users are encouraged to vote about distinct facts or prospective changes.

Online statistics interactively measures the popularity of content and activity of users.

Semantic syndication supports the distribution of information and their integration into desktop applications.

OntoWiki enables the easy creation of highly structured content by distributed communities. The following points summarize some limitations and weaknesses of OntoWiki and thus characterize the application domain:

Environment: OntoWiki is a Web application and presumes all collaborators to work in a Web environment, possibly distributed.

Usage Scenario: OntoWiki focuses on knowledge engineering projects where a single, precise usage scenario is either initially (yet) unknown or not (easily) definable.

Reasoning: Application of reasoning services was (initially) not the primary focus.

4.2 Generic and Domain-Specific Views

OntoWiki can be used as a tool for presenting, authoring and managing knowledge bases adhering to the RDF data model. As such, it provides generic methods and views, independent of the domain concerned. Two generic views included in OntoWiki are the resource view and the list view. While the former is generally used for displaying all known information about a resource, the latter can present a set of resources, typically instances of a certain concept. That concept must not necessarily be explicitly defined as `rdfs:Class` or `owl:Class` in the knowledge base. Via its faceted browsing, OntoWiki allows the construction of complex concept definitions, with a pre-defined class as a starting point by means of property value restrictions. These two views are sufficient for browsing and editing all information contained in a knowledge base in a generic way. For domain-specific use cases, OntoWiki provides an easy-to-use extension interface that enables the integration of custom components. By providing such a custom view, it is even possible to hide completely the fact that an RDF knowledge base is worked on. This permits OntoWiki to be used as a data-entry frontend for users with a less profound knowledge of Semantic Web technologies.

4.3 Workflow

With the use of RDFS [27] and OWL [126] as ontology languages, resource definition is divisible into different layers: a terminology box for conceptual information (i. e. classes and properties) and an assertion box for entities using the concepts defined (i. e. instances). There are characteristics of RDF which, for end users, are not easy to comprehend (e. g. *classes* can be defined as *instances* of `owl:Class`). OntoWiki's user interface, therefore, provides elements for these two layers, simultaneously increasing usability and improving a user's comprehension for the structure of the data. After starting and logging in into OntoWiki with registered user credentials, it is possible to select one of the existing ontologies. The user is then presented with general information about the ontology (i. e. all statements expressed about the knowledge base as a resource) and a list of defined classes, as part of the conceptual layer.

After starting and logging in into OntoWiki with registered user credentials, it is possible to select one of the existing knowledge bases. The user is then presented with general information about the ontology (i. e. all statements expressed about the knowledge base as a resource) and a list of defined classes, as part of the conceptual layer. By selecting one of these classes, the user obtains a list of the class' instances. OntoWiki

applies basic `rdfs:subClassOf` reasoning automatically. After selecting an instance from the list – or alternatively creating a new one – it is possible to manage (i. e. insert, edit and update) information in the details view. OntoWiki focuses primarily on the assertion layer, but also provides ways to manage resources on the conceptual layer. By enabling the visualization of schema elements, called *System Classes* in the OntoWiki nomenclature, conceptual resources can be managed in a similar fashion as instance data.

4.4 Authoring

Semantic content in OntoWiki is represented as resource descriptions. Following the RDF data model representing one of the foundations of the Semantic Web vision, resource descriptions are represented (at the lowest level) in the form of *statements*. Each of these statements (or triples) consist of a *subject* which identifies a resource as well as a *predicate* and an *object* which together represent data about said resource in a fashion reminiscent of key-value pairs. By means of RDFa [2], these statements are retained in the HTML view (i.e. user interface) part and are thus accessible to client-side techniques like JavaScript.

Authoring of such content is based on said client-side representation by employing the RDFauthor approach [148]: views are declared in terms of the model language (RDF) which allows the underlying model be restored. Based on this model, a user interface can be generated with the model being providing all the domain knowledge required to do so. The RDFauthor system provides an extensible set of authoring widgets specialized for certain editing tasks. RDFauthor was also extended by adding capabilities for automatically translating literal object values between different languages. Since the semantic context is known to the system, these translation functionality can be bound to arbitrary characteristics of the data (e. g. to a certain property or a missing language).

Versioning & Evolution. As outlined in the wiki principles, keeping track of all changes is an important task in order to encourage user participation. OntoWiki applies this concept to RDF-based knowledge engineering in that all changes are tracked on the statement level [10]. These low-level changes can be grouped to reflect application- and domain-specific tasks involving modifications to several statements as a single versioned item. Provenance information as well as other metadata (such as time, user or context) of a particular changeset can be attached to each individual changeset. All changes on the knowledge base can be easily reviewed and rolled-back if needed. The loosely typed data model of RDF encourages continuous evolution and refinement of knowledge bases. With *EvoPat*, OntoWiki supports this in a declarative, pattern-based manner (cf.[132]).

4.5 Access Interfaces

In addition to human-targeted graphical user interfaces, OntoWiki supports a number of machine-accessible data interfaces. These are based on established Semantic Web standards like SPARQL or accepted best practices like publication and consumption of Linked Data.



Fig. 15. OntoWiki views: (background) A tabular list view, which contains a filtered list of resources highlighting some specific properties of those resources and (foreground) a resource view which allows to tag and comment a specific resource as well as editing all property values

SPARQL Endpoint. The SPARQL recommendation not only defines a query language for RDF but also a protocol for sending queries to and receiving results from remote endpoints⁸. OntoWiki implements this specification, allowing all resources managed in an OntoWiki be queried over the Web. In fact, the aforementioned RDFauthor authoring interface makes use of SPARQL to query for additional schema-related information, treating OntoWiki as a remote endpoint in that case.

Linked Data. Each OntoWiki installation can be part of the emerging Linked Data Web. According to the Linked Data publication principles (cf. section 2), OntoWiki makes all resources accessible by its IRI (provided, the resource's IRI is in the same namespace as the OntoWiki instance). Furthermore, for each resource used in OntoWiki additional triples can be fetched if the resource is de-referenceable.

Semantic Pingback. Pingback is an established notification system that gained wide popularity in the blogosphere. With Semantic Pingback [147], OntoWiki adapts this idea to Linked Data providing a *notification mechanism* for resource usage. If a

⁸ <http://www.w3.org/TR/rdf-sparql-protocol/>

Pingback-enabled resource is mentioned (i. e. linked to) by another party, its pingback server is notified of the usage. Provided, the Semantic Pingback extension is enabled all resources used in OntoWiki are pinged automatically and all resources defined in OntoWiki are Pingback-enabled.

4.6 Exploration Interfaces

For exploring semantic content, OntoWiki provides several exploration interfaces that range from generic views over search interfaces to sophisticated querying capabilities for more RDF-knowledgeable users. The subsequent paragraphs give an overview of each of them.

Knowledge base as an information map. The compromise between, on the one hand, providing a generic user interface for arbitrary RDF knowledge bases and, on the other hand, aiming at being as intuitive as possible is tackled by regarding knowledge bases as *information maps*. Each node at the information map, i. e. RDF resource, is represented as a Web accessible page and interlinked to related digital resources. These Web pages representing nodes in the information map are divided into three parts: a left sidebar, a main content section and a right sidebar. The left sidebar offers the selection of content to display in the main content section. Selection opportunities include the set of available knowledge bases, a hierarchical browser and a full-text search.

Full-text search. The full-text search makes use of special indexes (mapped to proprietary extensions to the SPARQL syntax) if the underlying knowledge store provides this feature, else, plain SPARQL string matching is used. In both cases, the resulting SPARQL query is stored as an object which can later be modified (e. g. have its filter clauses refined). Thus, full-text search is seamlessly integrated with faceted browsing (see below).

Content specific browsing interfaces. For domain-specific use cases, OntoWiki provides an easy-to-use extension interface that enables the integration of custom components. By providing such a custom view, it is even possible to hide completely the fact that an RDF knowledge base is worked on. This permits OntoWiki to be used as a data-entry frontend for users with a less profound knowledge of Semantic Web technologies.

Faceted-browsing. Via its faceted browsing, OntoWiki allows the construction of complex concept definitions, with a pre-defined class as a starting point by means of property value restrictions. These two views are sufficient for browsing and editing all information contained in a knowledge base in a generic way.

Query-builder. OntoWiki serves as a SPARQL endpoint, however, it quickly turned out that formulating SPARQL queries is too tedious for end users. In order to simplify the creation of queries, we developed the *Visual Query Builder*⁹ (VQB) as an OntoWiki extension, which is implemented in JavaScript and communicates with the triple store

⁹ <http://aksw.org/Projects/OntoWiki/Extension/VQB>

using the SPARQL language and protocol. VQB allows to visually create queries to the stored knowledge base and supports domain experts with an intuitive visual representation of query and data. Developed queries can be stored and added via drag-and-drop to the current query. This enables the reuse of existing queries as building blocks for more complex ones.

4.7 Applications

Catalogous Professorum. The World Wide Web, as an ubiquitous medium for publication and exchange, already significantly influenced the way historians work: the online availability of catalogs and bibliographies allows to efficiently search for content relevant for a certain investigation; the increasing digitization of works from historical archives and libraries, in addition, enables historians to directly access historical sources remotely. The capabilities of the Web as a medium for collaboration, however, are only starting to be explored. Many, historical questions can only be answered by combining information from different sources, from different researchers and organizations. Also, after original sources are analyzed, the derived information is often much richer, than can be captured by simple keyword indexing. These factors pave the way for the successful application of knowledge engineering techniques in historical research communities.

In [131] we report about the application of an adaptive, semantics-based knowledge engineering approach using OntoWiki for the development of a prosopographical knowledge base. In prosopographical research, historians analyze common characteristics of historical groups by studying statistically relevant quantities of individual biographies. Untraceable periods of biographies can be determined on the basis of such accomplished analyses in combination with statistically examinations as well as patterns of relationships between individuals and their activities.

In our case, researchers from the historical seminar at Universität Leipzig aimed at creating a prosopographical knowledge base about the life and work of professors in the 600 years history of Universität Leipzig ranging from the year 1409 till 2009 - the *Catalogus Professorum Lipsiensis* (CPL). In order to enable historians to collect, structure and publish this prosopographical knowledge an ontological knowledge model was developed and incrementally refined over a period of three years. The community of historians working on the project was enabled to add information to the knowledge base using an adapted version of OntoWiki. For the general public, a simplified user interface¹⁰ is dynamically generated based on the content of the knowledge base. For access and exploration of the knowledge base by other historians a number of access interfaces was developed and deployed, such as a graphical SPARQL query builder, a relationship finder and plain RDF and Linked Data interfaces. As a result, a group of 10 historians supported by a much larger group of volunteers and external contributors collected information about 1,300 professors, 10,000 associated periods of life, 400 institutions and many more related entities.

The benefits of the developed knowledge engineering platform for historians are twofold: Firstly, the collaboration between the participating historians has significantly

¹⁰ Available at: <http://www.uni-leipzig.de/unigeschichte/professorenkatalog/>

improved: The ontological structuring helped to quickly establish a common understanding of the domain. Collaborators within the project, peers in the historic community as well as the general public were enabled to directly observe the progress, thus facilitating peer-review, feedback and giving direct benefits to the contributors. Secondly, the ontological representation of the knowledge facilitated original historical investigations, such as historical social network analysis, professor appointment analysis (e.g. with regard to the influence of cousin-hood or political influence) or the relation between religion and university. The use of the developed model and knowledge engineering techniques is easily transferable to other prosopographical research projects and with adaptations to the ontology model to other historical research in general. In the long term, the use of collaborative knowledge engineering in historian research communities can facilitate the transition from largely individual-driven research (where one historian investigates a certain research question solitarily) to more community-oriented research (where many participants contribute pieces of information in order to enlighten a larger research question). Also, this will improve the reusability of the results of historic research, since knowledge represented in structured ways can be used for previously not anticipated research questions.

OntoWiki Mobile. As comparatively powerful mobile computing devices are becoming more common, mobile web applications have started gaining in popularity. An important feature of these applications is their ability to provide *offline functionality* with local updates for later synchronization with a web server. The key problem here is the reconciliation, i. e. the problem of potentially *conflicting updates* from *disconnected clients*. Another problem current mobile application developers face is the plethora of mobile application development platforms as well as the incompatibilities between them. *Android* (Google), *iOS* (Apple), *Blackberry OS* (RIM), *WebOS* (HP/Palm), *Symbian* (Nokia) are popular and currently widely deployed platforms, with many more proprietary ones being available as well. As a consequence of this fragmentation, realizing a special purpose application, which works with many or all of these platforms is extremely time consuming and inefficient due to the large amount of duplicate work required.

The W3C addressed this problem, by enriching HTML in its 5th revision with access interfaces to local storage (beyond simple cookies) as well as a number of devices and sensors commonly found on mobile devices (e. g. GPS, camera, compass etc.). We argue, that in combination with semantic technologies these features can be used to realize a *general purpose*, mobile collaboration platform, which can support the long tail of mobile special interest applications, for which the development of individual tools would not be (economically) feasible.

In [39] we present the *OntoWiki Mobile* approach realizing a mobile semantic collaboration platform based on the OntoWiki. It comprises specifically adopted user interfaces for browsing, faceted navigation as well as authoring of knowledge bases. It allows users to collect instance data and refine the structured knowledge bases on-the-go. OntoWiki Mobile is implemented as an *HTML5 web application*, thus being completely mobile device platform independent. In order to allow offline use in cases with restricted network coverage (or in order to avoid roaming charges) it uses the novel HTML5 local

storage feature for replicating parts of the knowledge base on the mobile device. Hence, a crucial part of OntoWiki Mobile is the advanced conflict resolution for RDF stores. The approach is based on a combination of the EvoPat [132] method for data evolution and ontology refactoring along with a versioning system inspired by distributed version control systems like Git. OntoWiki Mobile is a generic, application domain agnostic tool, which can be utilized in a wide range of very different usage scenarios ranging from instance acquisition to browsing of semantic data on the go. Typical OntoWiki Mobile usage scenarios are settings where users need to author and access semantically structured information on the go or in settings where users are away from regular power supply and restricted to light-weight equipment (e. g. scientific expeditions).

Semantics-Based Requirements Engineering. Semantic interoperability, linked data, and a shared conceptual foundation become increasingly important prerequisites in software development projects that are characterized by spatial dispersion, large numbers of stakeholders, and heterogeneous development tools. The SoftWiki OntoWiki extension [93] focuses specifically on semantic collaboration with respect to requirements engineering. Potentially very large and spatially distributed groups of stakeholders, including developers, experts, managers, and average users, shall be enabled to collect, semantically enrich, classify, and aggregate software requirements. OntoWiki is used to support collaboration as well as interlinking and exchange of requirements data. To ensure a shared conceptual foundation and semantic interoperability, we developed the SoftWiki Ontology for Requirements Engineering (SWORE) that defines core concepts of requirement engineering and the way they are interrelated. For instance, the ontology defines frequent relation types to describe requirements interdependencies such as details, conflicts, related to, depends on, etc. The flexible SWORE design allows for easy extension. Moreover, the requirements can be linked to external resources, such as publicly available domain knowledge or company-specific policies. The whole process is called semantification of requirements. It is envisioned as an evolutionary process: The requirements are successively linked to each other and to further concepts in a collaborative way, jointly by all stakeholders. Whenever a requirement is formulated, reformulated, analyzed, or exchanged, it might be semantically enriched by the respective participant.

5 Linking

The fourth Linked Data Principle, i.e., “Include links to other URIs, so that they can discover more things” (cf. section 2) is the most important Linked Data principle as it enables the paradigm change from data silos to interoperable data distributed across the Web. Furthermore, it plays a key role in important tasks such as cross-ontology question answering [22,94], large-scale inferences [151,99] and data integration [95,19]. Yet, while the number of triples in Linked Data sources increases steadily and has surpassed 31 billions¹¹, links between knowledge bases still constitute less than 5% of these triples. The goal of linking is to tackle this sparseness so as to transform the Web into a platform for data and information integration as well as for search and querying.

¹¹ <http://lod-cloud.net/>

5.1 Link Discovery

Linking can be generally defined as *connecting things that are somehow related*. In the context of Linked Data, the idea of linking is especially concerned with establishing typed links between entities (i.e., classes, properties or instances) contained in knowledge bases. Over the last years, several frameworks have been developed to address the lack of typed links between the different knowledge bases on the Linked Data web. Overall, two main categories of frameworks that aim to achieve this goal can be differentiated. The first category implements *ontology matching* techniques and aims to establish links between the ontologies underlying two data sources. The second and more prominent category of approaches, dubbed *instance matching* approaches (also called linking or link discovery approaches), aims to discover links between instances contained in two data sources. It is important to notice that while ontology and instance matching are similar to schema matching [128,127] and record linkage [162,38,25] respectively (as known in the research area of databases), linking on the Web of Data is a more generic and thus more complex task, as it is not limited to finding equivalent entities in two knowledge bases. Rather, it aims at finding semantically related entities and establishing typed links between them, most of these links being imbued with formal properties (e.g., transitivity, symmetry, etc.) that can be used by reasoners and other application to infer novel knowledge. In this section, we will focus on the discovery of links between instances and use the term link discovery as name for this process. An overview of ontology matching techniques is given in [42].

Formally, link discovery can be defined as follows:

Definition 3 (Link Discovery). *Given two sets S (source) and T (target) of instances, a (complex) semantic similarity measure $\sigma : S \times T \rightarrow [0, 1]$ and a threshold $\theta \in [0, 1]$, the goal of link discovery task is to compute the set $M = \{(s, t), \sigma(s, t) \geq \theta\}$.*

In general, the similarity function used to carry out a link discovery task is described by using a *link specification* (sometimes called *linkage decision rule* [68]).

5.2 Challenges

Two key challenges arise when trying to discover links between two sets of instances: the computational complexity of the matching task *per se* and the selection of an appropriate link specification. The first challenge is intrinsically related to the link discovery process. The time complexity of a matching task can be measured by the number of comparisons necessary to complete this task. When comparing a source knowledge base S with a target knowledge base T , the completion of a matching task requires a-priori $O(|S||T|)$ comparisons, an impractical proposition as soon as the source and target knowledge bases become large. For example, discovering duplicate cities in DBpedia [6] alone would necessitate approximately 0.15×10^9 similarity computations. Hence, the provision of time-efficient approaches for the reduction of the time complexity of link discovery is a key requirement to instance linking frameworks for Linked Data.

The second challenge of the link discovery process lies in the selection of an appropriate link specification. The configuration of link discovery frameworks is usually

carried out manually, in most cases simply by guessing. Yet, the choice of a suitable link specification measure is central for the discovery of satisfactory links. The large number of properties of instances and the large spectrum of measures available in literature underline the complexity of choosing the right specification manually¹². Supporting the user during the process of finding the appropriate similarity measure and the right properties for each mapping task is a problem that still needs to be addressed by the Linked Data community. Methods such as supervised and active learning can be used to guide the user in need of mapping to a suitable linking configuration for his matching task. In the following, we give a short overview of existing frameworks for Link Discovery on the Web of Data. Subsequently, we present a time-efficient framework for link discovery in more detail and show how it can detect link specifications using active learning.

5.3 Approaches to Link Discovery

Current frameworks for link discovery can be subdivided into two main categories: *domain-specific* and *universal* frameworks. Domain-specific link discovery frameworks aim at discovering links between knowledge bases from a particular domain. One of the first domain-specific approaches to carry out instance linking for Linked Data was implemented in the *RKBExplorer*¹³ [50] with the aim of discovering links between entities from the domain of academics. Due to the lack of data available as Linked Data, the *RKBExplorer* had to extract RDF from heterogeneous data source so as to populate its knowledge bases with instances according to the AKT ontology¹⁴. Especially, instances of persons, publications and institutions were retrieved from several major metadata websites such as ACM and DBLP. The linking was implemented by the so-called Consistent Reference Service (CRS) which linked equivalent entities by comparing properties including their type and label. So far, the CRS is limited to linking objects in the knowledge bases underlying the *RKBExplorer* and cannot be used for other tasks without further implementation.

Another domain-specific tool is GNAT [129], which was developed for the music domain. It implements several instance matching algorithms of which the most sophisticated, the online graph matching algorithm (OGMA), applies a similarity propagation approach to discover equivalent resources. The basic approach implemented by OGMA starts with a single resource $s \in S$. Then, it retrieves candidate matching resources $t \in T$ by comparing properties such as `foaf:name` for artists and `dc:title` for albums. If $\sigma(s, t) \geq \theta$, then the algorithm terminates. In case a disambiguation is needed, the resource related to s and t in their respective knowledge bases are compared and their similarity value is cumulated to recompute $\sigma(s, t)$. This process is iterated until a mapping resource for s is found in T or no resource matches.

Universal link discovery frameworks are designed to carry out mapping tasks independently from the domain of the source and target knowledge bases. For example, RDF-AI [139], a framework for the integration of RDF data sets, implements a

¹² The SimMetrics project (<http://simmetrics.sf.net>) provides an overview of strings similarity measures.

¹³ <http://www.rkbexplorer.com>

¹⁴ <http://www.aktors.org/publications/ontology/>

five-step approach that comprises the preprocessing, matching, fusion, interlinking and post-processing of data sets. RDF-AI contains a series of modules that allow for computing instances matches by comparing their properties. Especially, it contains translation modules that allow to process the information contained in data sources before mapping. By these means, it can boost the precision of the mapping process. These modules can be configured by means of XML-files. RDF-AI does not comprise means for querying distributed data sets via SPARQL¹⁵. In addition, it suffers from not being time-optimized. Thus, mapping by using this tool can be very time-consuming.

A time-optimized approach to link discovery is implemented by the LIMES framework [112,111,116] (Link Discovery Framework for metric spaces).¹⁶ The idea behind the LIMES framework is to use the mathematical characteristics of similarity and distance measures to reduce the number of computations that have to be carried out by the system without losing any link. For example, LIMES can make use of the fact that the edit distance is a distance metric to approximate distances without having to compute them [112]. Moreover, it implements the reductio-ratio-optimal space tiling algorithm \mathcal{HR}^3 to compute similarities in affine spaces with Minkowski measures [110]. In contrast to other frameworks (of which most rely on blocking), LIMES relies on time-efficient set operators to combine the results of these algorithms efficiently and has been shown to outperform the state of the art by these means [111]. Moreover, LIMES implements unsupervised and supervised machine learning approaches for detecting high-quality link specifications [116,117].

Another link discovery framework is SILK [156]. SILK implements several approaches to minimize the time necessary for mapping instances from knowledge bases. In addition to implementing rough index pre-matching to reach a quasi-linear time-complexity, SILK also implements a lossless blocking algorithm called MultiBlock [69] to reduce its overall runtime. The approach relies on generating overlapping blocks of instances and only comparing pairs of instances that are located in the same block. Moreover, SILK provides supervised machine learning approaches for link discovery [70].

It is important to notice that the task of discovering links between knowledge bases is related with record linkage [162,38] and de-duplication [25]. The database community has produced a vast amount of literature on efficient algorithms for solving these problems. Different blocking techniques such as standard blocking, sorted-neighborhood, bigram indexing, canopy clustering and adaptive blocking [18,23,76] have been developed to address the problem of the quadratic time complexity of brute force comparison methods. The idea is to filter out obvious non-matches efficiently before executing the more detailed and time-consuming comparisons. In the following, we present a state-of-the-art framework that implements lossless instance matching based on a similar idea in detail.

5.4 The LIMES Algorithm

The original LIMES algorithm described in [112] addresses the scalability problem of link discovery by utilizing the *triangle inequality* in metric spaces to compute

¹⁵ <http://www.w3.org/TR/rdf-sparql-query/>

¹⁶ <http://limes.sf.net>. A graphical user interface can be found at <http://saim.aksw.org>.

pessimistic estimates of instance similarities. Based on these approximations, LIMES can filter out a large number of instance pairs that cannot suffice the matching condition set by the user. The real similarities of the remaining instances pairs are then computed and the matching instances are returned.

Mathematical Framework. In the remainder of this section, we use the following notations:

1. A is an affine space,
2. m, m_1, m_2, m_3 symbolize metrics on A ,
3. x, y and z represent points from A and
4. α, β, γ and δ are scalars, i.e., elements of \mathbb{R} .

Definition 4 (Metric space). A metric space is a pair (A, m) such that A is an affine space and $m : A \times A \rightarrow \mathbb{R}$ is a function such that for all x, y and $z \in A$

1. $m(x, y) \geq 0$ (M_1) (non-negativity),
2. $m(x, y) = 0 \Leftrightarrow x = y$ (M_2) (identity of indiscernibles),
3. $m(x, y) = m(y, x)$ (M_3) (symmetry) and
4. $m(x, z) \leq m(x, y) + m(y, z)$ (M_4) (triangle inequality).

Note that the definition of a matching based on a similarity function σ can be rewritten for metrics m as follows:

Definition 5 (Instance Matching in Metric Spaces). Given two sets S (source) and T (target) of instances, a metric $m : S \times T \rightarrow [0, \infty[$ and a threshold $\theta \in [0, \infty[$, the goal of instance matching task is to compute the set $M = \{(s, t) | m(s, t) \leq \theta\}$.

Example of metrics on strings include the Levenshtein distance and the block distance. However, some popular measures such as JaroWinkler [161] do not satisfy the triangle inequality and are consequently not metrics. The rationale behind the LIMES framework is to make use of the boundary conditions entailed by the triangle inequality (TI) to reduce the number of comparisons (and thus the time complexity) necessary to complete a matching task. Given a metric space (A, m) and three points x, y and z in A , the TI entails that

$$m(x, y) \leq m(x, z) + m(z, y). \quad (1)$$

Without restriction of generality, the TI also entails that

$$m(x, z) \leq m(x, y) + m(y, z), \quad (2)$$

thus leading to the following boundary conditions in metric spaces:

$$m(x, y) - m(y, z) \leq m(x, z) \leq m(x, y) + m(y, z). \quad (3)$$

Inequality 3 has two major implications. The first is that the distance from a point x to any point z in a metric space can be approximated given the distance from x to a reference point y and the distance from the reference point y to z . Such a reference point is called an *exemplar* following [49]. The role of an exemplar is to be used as a sample

of a portion of the metric space A . Given an input point x , knowing the distance from x to an exemplar y allows to compute lower and upper bounds of the distance from x to any other point z at a known distance from y . An example of such an approximation is shown in Figure 21. In this figure, all the points on the circle are subject to the same distance approximation. The distance from x to z is close to the lower bound of inequality 3, while the distance from x to z' is close to the upper bound of the same inequality.

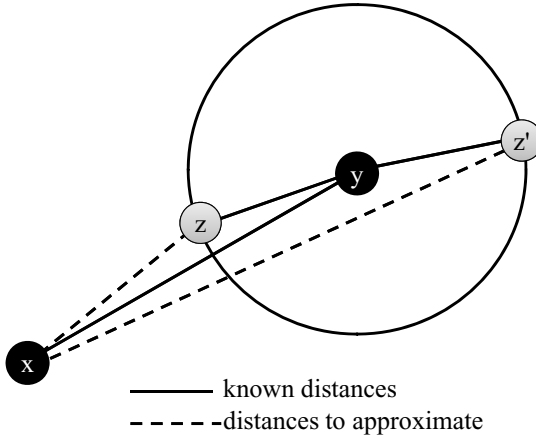


Fig. 16. Approximation of distances via exemplars. The lower bound of the distance from x to z can be approximated by $m(x, y) - m(y, z)$.

The second implication of inequality 3 is that the distance from x to z can only be smaller than θ if the lower bound of the approximation of the distance from x to z via *any exemplar* y is also smaller than θ . Thus, if the lower bound of the approximation of the distance $m(x, z)$ is larger than θ , then $m(x, z)$ itself must be larger than θ . Formally,

$$m(x, y) - m(y, z) > \theta \Rightarrow m(x, z) > \theta. \quad (4)$$

Supposing that all distances from instances $t \in T$ to exemplars are known, reducing the number of comparisons simply consists of using inequality 4 to compute an approximation of the distance from all $s \in S$ to all $t \in T$ and computing the real distance only for the (s, t) pairs for which the first term of inequality 4 does not hold. This is the core of the approach implemented by LIMES.

Computation of Exemplars. The core idea underlying the computation of exemplars in LIMES is to select a set of exemplars in the metric space underlying the matching task in such a way that they are distributed uniformly in the metric space. One way to achieve this goal is by ensuring that the exemplars display a high dissimilarity. The approach used by LIMES to generate exemplars with this characteristic is shown in Algorithm 1.

Algorithm 1. Computation of Exemplars**Require:** Number of exemplars n **Require:** Target knowledge base T

1. Pick random point $e_1 \in T$
2. Set $E = E \cup \{e_1\}$;
3. Compute the distance from e_1 to all $t \in T$
- while** $|E| < n$ **do**
 4. Get a random point e' such that $e' \in \operatorname{argmax}_t \sum_{e \in E} m(t, e)$
 5. $E = E \cup \{e'\}$;
 6. Compute the distance from e' to all $t \in T$

end while

7. Map each point in $t \in T$ to one of the exemplars $e \in E$ such that $m(t, e)$ is minimal

return E

Let n be the desired number of exemplars and E the set of all exemplars. In step 1 and 2, LIMES initializes E by picking a random point e_1 in the metric space (T, m) and setting $E = \{e_1\}$. Then, it computes the similarity from the exemplar e_1 to every other point in T (step 3). As long as the size of E has not reached n , LIMES repeats steps 4 to 6: In step 4, a point $e' \in T$ such that the sum of the distances from e' to the exemplars $e \in E$ is maximal (there can be many of these points) is chosen randomly. This point is chosen as new exemplar and consequently added to E (step 5). Then, the distance from e' to all other points in T is computed (step 6). Once E has reached the size n , LIMES terminates the iteration. Finally, each point is mapped to the exemplar to which it is most similar (step 7) and the exemplar computation terminates (step 8). This algorithm has a constant time complexity of $O(|E||T|)$.

An example of the results of the exemplar computation algorithm ($|E| = 3$) is shown in Figure 17. The initial exemplar was the leftmost exemplar in the figure.

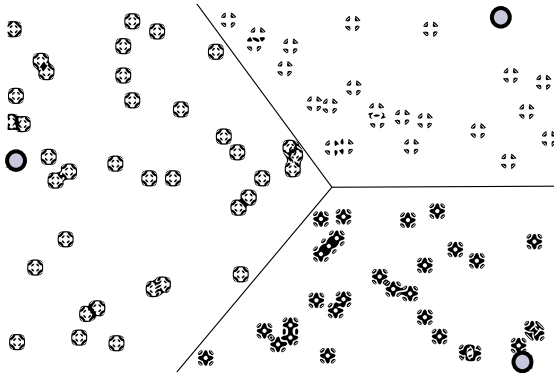


Fig. 17. Mapping of points to three exemplars in a metric space. The exemplars are displayed as gray disks.

Matching Based on Exemplars. The instances associated with an exemplar $e \in E$ in step 7 of Algorithm 1 are stored in a list L_e sorted in descending order with respect to the distance to e . Let $\lambda_1^e \dots \lambda_m^e$ be the elements of the list L_e . The goal of matching an instance s from a source knowledge base to a target knowledge base w.r.t. a metric m is to find all instances t of the target knowledge source such that $m(s, t) \leq \theta$, where θ is a given threshold. LIMES achieves this goal by using the matching algorithm based on exemplars shown in Algorithm 2.

Algorithm 2. LIMES' Matching algorithm

Require: Set of exemplars E

Require: Instance $s \in S$

Require: Metric m

Require: threshold θ

```

1.  $M = \emptyset$ 
for  $e \in |E|$  do
  if  $m(s, e) \leq \theta$  then
    2.  $M = M \cup \{e\}$ 
    for  $i = 1 \dots |L_e|$  do
      if  $(m(s, e) - m(e, \lambda_i^e)) \leq \theta$  then
        if  $m(s, \lambda_i^e) \leq \theta$  then
          3.  $M = M \cup \{(s, \lambda_i^e)\}$ 
        end if
      else
        break
      end if
    end for
  end if
end for
return  $M$ 

```

LIMES only carries out a comparison when the approximation of the distance is less than the threshold. Moreover, it terminates the similarity computation for an exemplar e as soon as the first λ^e is found such that the lower bound of the distance is larger than θ . This break can be carried out because the list L_e is sorted, i.e., if $m(s, e) - m(e, \lambda_i^e) > \theta$ holds for the i^{th} element of L_e , then the same inequality holds for all $\lambda_j^e \in L_e$ with $j > i$. In the worst case, LIMES' matching algorithm has the time complexity $O(|S||T|)$, leading to a total worst time complexity of $O((|E| + |S|)|T|)$, which is larger than that of brute force approaches. However, as the results displayed in Figure 18 show, a correct parameterization of LIMES leads to significantly smaller numbers of comparisons and runtimes.

5.5 The \mathcal{HR}^3 Algorithm

Let S resp. T be the source and target of a Link Discovery task. One of the key ideas behind time-efficient Link Discovery algorithms \mathcal{A} is to reduce the number of

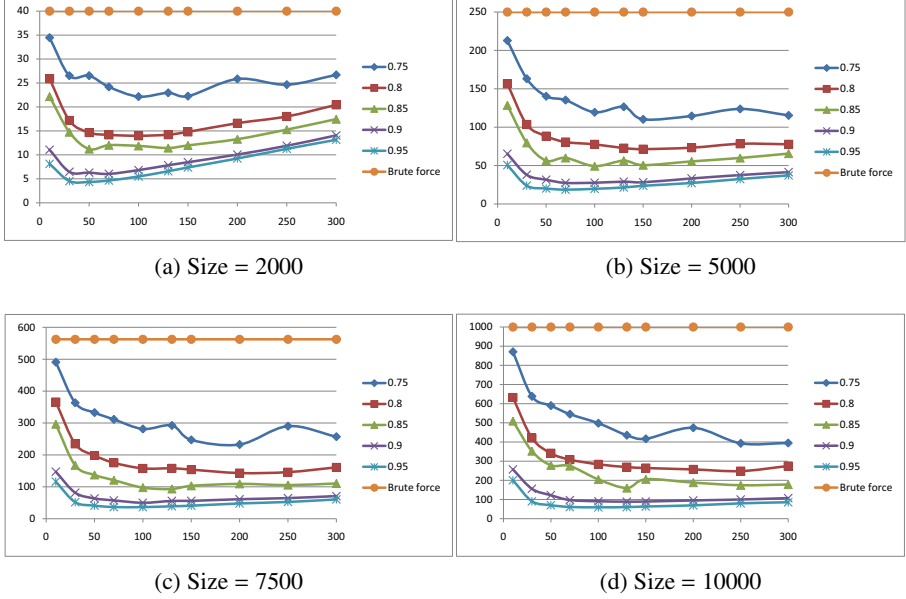


Fig. 18. Comparisons required by LIMES for different numbers of exemplars on knowledge bases of different sizes. The x-axis shows the number of exemplars, the y-axis the number of comparisons in multiples of 10^5 .

comparisons that are effectively carried out to a number $C(\mathcal{A}) < |S||T|$. The reduction ratio RR of an algorithm \mathcal{A} is given by

$$RR(\mathcal{A}) = 1 - \frac{C(\mathcal{A})}{|S||T|}. \quad (5)$$

$RR(\mathcal{A})$ captures how much of the Cartesian product $|S||T|$ was not explored before the output of \mathcal{A} was reached. It is obvious that even an optimal lossless solution which performs only the necessary comparisons cannot achieve a RR of 1. Let C_{min} be the minimal number of comparisons necessary to complete the Link Discovery task without losing recall, i.e., $C_{min} = |\mathcal{M}|$. The relative reduction ratio $RRR(\mathcal{A})$ is defined as the portion of the minimal number of comparisons that was carried out by the algorithm \mathcal{A} before it terminated. Formally

$$RRR(\mathcal{A}) = \frac{1 - \frac{C_{min}}{|S||T|}}{1 - \frac{C(\mathcal{A})}{|S||T|}} = \frac{|S||T| - C_{min}}{|S||T| - C(\mathcal{A})}. \quad (6)$$

$RRR(\mathcal{A})$ indicates how close \mathcal{A} is to the optimal solution with respect to the number of candidates it tests. Given that $C(\mathcal{A}) \geq C_{min}$, $RRR(\mathcal{A}) \geq 1$. Note that the larger the value of $RRR(\mathcal{A})$, the poorer the performance of \mathcal{A} with respect to the task at hand.

The main observation that led \mathcal{HR}^3 is that while most algorithms aim to optimize their RR (and consequently their RRR), most approaches do not provide any guarantee

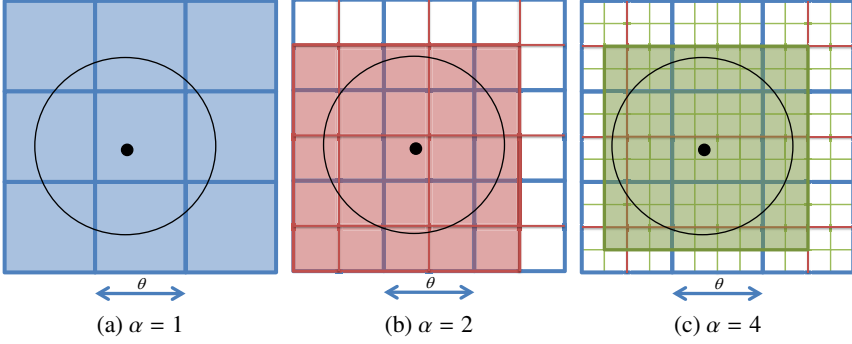


Fig. 19. Space tiling for different values of α . The colored squares show the set of elements that must be compared with the instance located at the black dot. The points within the circle lie within the distance θ of the black dot. Note that higher values of α lead to a better approximation of the hypersphere but also to more hypercubes.

with respect to the RR (and consequently the RRR) that they can achieve. The approach is the first mathematically optimal algorithm w.r.t the reduction ratio that it can achieve, i.e., the first approach such that given any relative reduction ratio r , there is always a setting that leads \mathcal{HR}^3 achieving a relative reduction ratio r' with $r' \leq r$. To achieve this goal \mathcal{HR}^3 relies on space tiling as introduced by the HYPPO algorithm [109].

Space Tiling for Link Discovery.

HYPPO addresses the problem of efficiently mapping instance pairs $(s, t) \in S \times T$ described by using exclusively numeric values in a n -dimensional metric space and has been shown to outperform the state of the art in previous work [109]. The observation behind space tiling is that in spaces (Ω, δ) with orthogonal, (i.e., uncorrelated) dimensions¹⁷, common metrics for Link Discovery can be decomposed into the combination of functions $\phi_{i, i \in \{1 \dots n\}}$ which operate on exactly one dimension of Ω : $\delta = f(\phi_1, \dots, \phi_n)$. For Minkowski distances of order p , $\phi_i(x, \omega) = |x_i - \omega_i|$ for all values of i and $\delta(x, \omega) = \sqrt[p]{\sum_{i=1}^n \phi_i^p(x, \omega)}$. A direct consequence of this observation is the inequality $\phi_i(x, \omega) \leq \delta(x, \omega)$. The basic insight that results this observation is that the hypersphere $H(\omega, \theta) = \{x \in \Omega : \delta(x, \omega) \leq \theta\}$ is a subset of the hypercube V defined as $V(\omega, \theta) = \{x \in \Omega : \forall i \in \{1 \dots n\}, \phi_i(x_i, \omega_i) \leq \theta\}$. Consequently, one can reduce the number of comparisons necessary to detect all elements of $H(\omega, \theta)$ by discarding all elements which are not in $V(\omega, \theta)$ as non-matches. Let $\Delta = \theta/\alpha$, where $\alpha \in \mathbb{N}$ is the *granularity parameter* that controls how fine-grained the space tiling should be (see Figure 19 for an example). We first tile Ω into the adjacent hypercubes (short: cubes) C that contain all the points ω such that

$$\forall i \in \{1 \dots n\}, c_i \Delta \leq \omega_i < (c_i + 1)\Delta \text{ with } (c_1, \dots, c_n) \in \mathbb{N}^n. \quad (7)$$

¹⁷ Note that in all cases, a space transformation exists that can map a space with correlated dimensions to a space with uncorrelated dimensions.

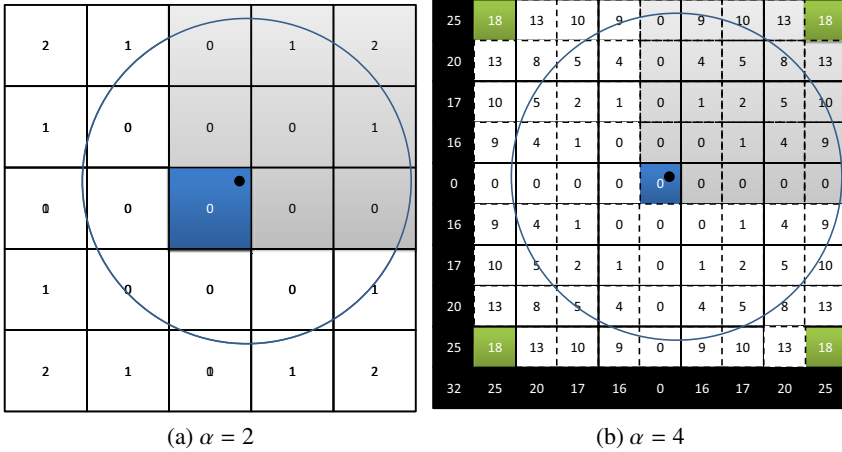


Fig. 20. Space tiling and resulting index for a two-dimensional example. Note that the index in both subfigures was generated for exactly the same portion of space. The black dot stands for the position of ω .

We call the vector (c_1, \dots, c_n) the coordinates of the cube C . Each point $\omega \in \Omega$ lies in the cube $C(\omega)$ with coordinates $(\lfloor \omega_i / \Delta \rfloor)_{i=1 \dots n}$. Given such a space tiling, it is obvious that $V(\omega, \theta)$ consists of the union of the cubes such that $\forall i \in \{1 \dots n\} : |c_i - c(\omega)_i| \leq \alpha$.

\mathcal{HR}^3 's Indexing Scheme. Let $\omega \in \Omega = S \cup T$ be an arbitrary reference point. Furthermore, let δ be the Minkowski distance of order p . The *index* function is defined as follows:

$$index(C, \omega) = \begin{cases} 0 & \text{if } \exists i : |c_i - c(\omega)_i| \leq 1 \text{ with } i \in \{1, \dots, n\}, \\ \sum_{i=1}^n (|c_i - c(\omega)_i| - 1)^p & \text{else,} \end{cases} \quad (8)$$

where C is a hypercube resulting from a space tiling and $\omega \in \Omega$. Figure 20 shows an example of such indexes for $p = 2$ with $\alpha = 2$ (Figure 20a) and $\alpha = 4$ (Figure 20b). Note that the blue square with index 0 contains the reference point ω . All elements of C must only be compared with the elements of cubes C' such that $index(C, C') \leq \alpha^p$. The authors of [110] prove formally that given this approach to space tiling, the following theorem holds:

Theorem 1. $\lim_{\alpha \rightarrow \infty} RRR(\mathcal{HR}^3, \alpha) = 1$.

This conclusion is illustrated by Figure 21, which shows the space tiling computed by \mathcal{HR}^3 for different values of α with $p = 2$ and $n = 2$. The higher α , the closer the approximation is to a circle. Note that these results allow to conclude that for any RRR -value r larger than 1, there is a setting of \mathcal{HR}^3 that can compute links with a RRR smaller or equal to r .

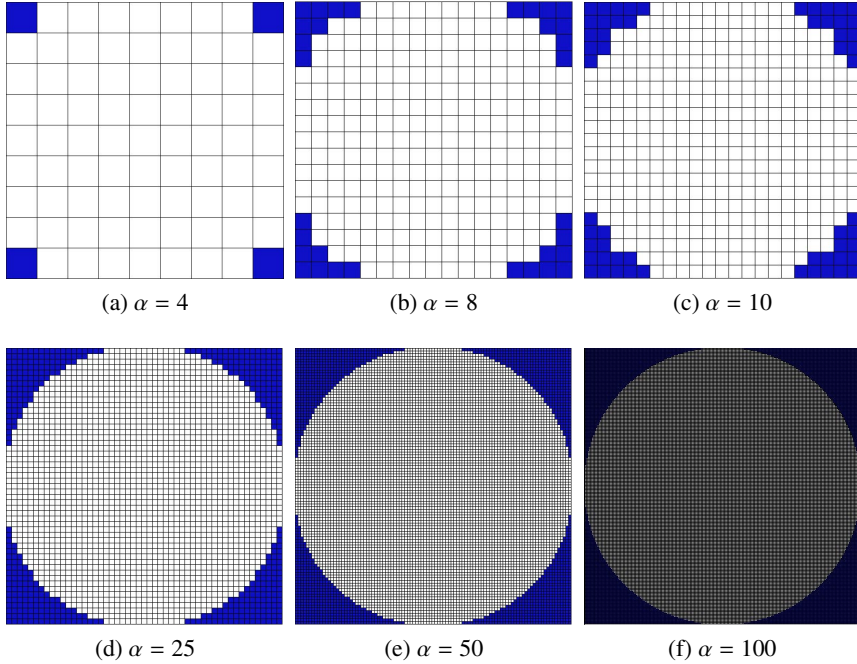


Fig. 21. Space tilings generated by \mathcal{HR}^3 for different values of α . The white squares are selected for comparisons with the elements of the square in the middle of the figure whilst the colored ones are discarded.

Evaluation. \mathcal{HR}^3 was evaluated against HYPPO w.r.t. to the number of comparisons that it has to carry out in several settings. In the first and second experiments, the goal was to deduplicate DBpedia places by comparing their names (`rdfs:label`), minimum elevation, elevation and maximum elevation. 2988 entities possessed all four properties. The Euclidean metric was applied to the last three values with the thresholds 49 meters resp. 99 meters for the first resp. second experiment. The third and fourth experiments aimed to discover links between Geonames and LinkedGeoData. This experiment was of considerably larger scale than the first one, as we compared 74458 entities in Geonames with 50031 entities from LinkedGeoData. Again, the number of comparisons necessary to complete the task by using the Euclidean metric was measured. The distance thresholds were set to 1 resp. 9° in experiment 3 resp. 4. We ran all experiments on the same Windows 7 Enterprise 64-bit computer with a 2.8GHz i7 processor with 8GB RAM. The JVM was allocated 7GB RAM to ensure that the runtimes were not influenced by swapping. Only one of the kernels of the processors was used.

The results (see Figure 22) show that \mathcal{HR}^3 can reduce the overhead in comparisons (i.e., the number of unnecessary comparisons divided by the number of necessary comparisons) from approximately 24% for HYPPO to approximately 6% (granularity = 32). In experiment 2, the overhead is reduced from 4.1% to 2%. This difference in overhead reduction is mainly due to the data clustering around certain values and the clusters

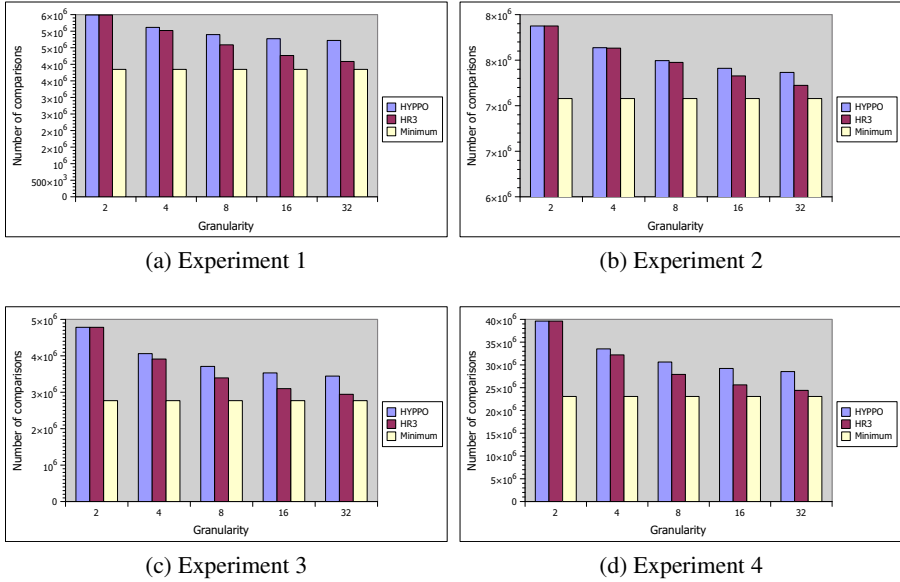


Fig. 22. Number of comparisons for \mathcal{HR}^3 and HYPPO

having a radius between 49 meters and 99 meters. Thus, running the algorithms with a threshold of 99 meters led to only a small a-priori overhead and HYPPO performing remarkably well. Still, even on such data distributions, \mathcal{HR}^3 was able to discard even more data and to reduce the number of unnecessary computations by more than 50% relative. In the best case (Exp. 4, $\alpha = 32$, see Figure 22d), \mathcal{HR}^3 required approximately 4.13×10^6 less comparisons than HYPPO for $\alpha = 32$. Even for the smallest setting (Exp. 1, see Figure 22a), \mathcal{HR}^3 still required 0.64×10^6 less comparisons.

5.6 Active Learning of Link Specifications

The second challenge of Link Discovery is the time-efficient discovery of link specifications for a particular linking task. Several approaches have been proposed to achieve this goal, of which most rely on genetic programming [70,117,115]. The COALA (Correlation-Aware Active Learning) approach was implemented on top of the genetic programming approach EAGLE [116] with the aim of improving the selection of positive and negative examples during active learning. In the following, we give an overview of COALA.

Intuition. Let \mathcal{N} be the set of most informative negative and \mathcal{P} the set of most informative positive examples w.r.t. an informativeness function ifm (e.g., the distance from the decision boundary). used by a curious classifier [141] The basic insight

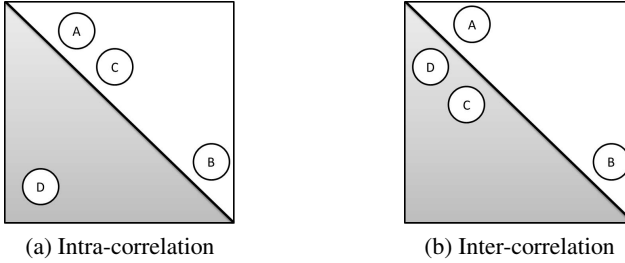


Fig. 23. Examples of correlations within classes and between classes. In each subfigure, the gray surface represent \mathcal{N} while the white surface stands for \mathcal{P} . The oblique line is C 's boundary.

behind COALA is that the correlation between the features of the elements of \mathcal{N} and \mathcal{P} should play a role when computing the sets \mathcal{I}^+ and \mathcal{I}^- of positive resp. negative queries for the oracle. In particular, two main factors affect the information content of a link candidate: its similarity to elements of its presumed class and to elements of the other class. For the sake of simplicity, we will assume that the presumed class of the link candidate of interest is $+1$, i.e., that the link candidate was classified as positive by the current curious classifier. Our insights yet hold symmetrically for link candidates whose presumed class is -1 .

Let $A = (s_A, t_A), B = (s_B, t_B) \in \mathcal{P}$ to be two link candidates which are equidistant from C 's boundary. Consider Figure 23a, where $\mathcal{P} = \{A, B, C\}$ and $\mathcal{N} = \{D\}$. The link candidate B is on on average most distant from any other elements of \mathcal{P} . Thus, it is more likely to be a statistical outlier than A . Hence, making a classification error on B should not have the same impact as an erroneous classification of link candidate A , which is close to another presumably positive link candidate, C . Consequently, B should be considered less informative than A . Approaches that make use of this information are said to exploit the *intra-class correlation*. Now, consider Figure 23b, where $\mathcal{P} = \{A, B\}$ and $\mathcal{N} = \{C, D\}$. While the probability of A being an outlier is the same as B 's, A is still to be considered more informative than B as it is located closer to elements of \mathcal{N} and can thus provide more information on where to set the classifier boundary. This information is dubbed *inter-class correlation*. Several approaches that make use of these two types of correlations can be envisaged. In the following, we present two approaches for these purposes. The first makes use of intra-class correlations and relies on graph clustering. The second approach relies on the spreading activation principle in combination with weight decay. We assume that the complex similarity function σ underlying C is computed by combining n atomic similarity functions $\sigma_1, \dots, \sigma_n$. This combination is most commonly carried out by using metric operators such as min, max or linear combinations.¹⁸ Consequently, each link candidate (s, t) can be described by a vector $(\sigma_1(s, t), \dots, \sigma_n(s, t)) \in [0, 1]^n$. We define the *similarity of link candidates* $\text{sim} : (S \times T)^2 \rightarrow [0, 1]$ to be the inverse of the Euclidean distance in the space spanned by the similarities σ_1 to σ_n . Hence, the similarity of two link candidates (s, t) and (s', t') is given by:

¹⁸ See [111] for a more complete description of a grammar for link specifications.

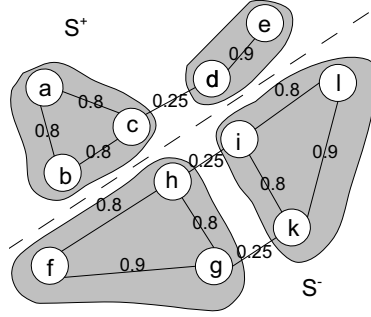


Fig. 24. Example of clustering. One of the most informative single link candidate is selected from each cluster. For example, d is selected from the cluster $\{d, e\}$.

$$\text{sim}((s, t), (s', t')) = \frac{1}{1 + \sqrt{\sum_{i=1}^n (\sigma_i(s, t) - \sigma_i(s', t'))^2}}. \quad (9)$$

Note that we added 1 to the denominator to prevent divisions by 0.

Graph Clustering. The basic intuition behind using clustering for COALA is that groups of very similar link candidates can be represented by a single link candidate. Consequently, once a representative of a group has been chosen, all other elements of the group become less informative. An example that illustrates this intuition is given in Figure 24. We implemented COALA based on clustering as follows: In each iteration, we begin by first selecting two sets $\mathcal{S}^+ \subseteq \mathcal{P}$ resp. $\mathcal{S}^- \subseteq \mathcal{N}$ that contain the positive resp. negative link candidates that are most informative for the classifier at hand. Formally, \mathcal{S}^+ fulfills

$$\forall x \in \mathcal{S}^+ \forall y \in \mathcal{P}, y \notin \mathcal{S}^+ \rightarrow \text{ifm}(y) \leq \text{ifm}(x). \quad (10)$$

The analogous equation holds for \mathcal{S}^- . In the following, we will explain the further steps of the algorithm for \mathcal{S}^+ . The same steps are carried out for \mathcal{S}^- .

First, we compute the similarity of all elements of \mathcal{S}^+ by using the similarity function shown in Equation 9. In the resulting similarity matrix, we set all elements of the diagonal to 0. Then, for each $x \in \mathcal{S}^+$, we only retain a fixed number ec of highest similarity values and set all others to 0. The resulting similarity matrix is regarded as the adjacency matrix of an undirected weighted graph $G = (V, E, \text{sim})$. G 's set of nodes V is equal to \mathcal{S}^+ . The set of edges E is a set of 2-sets¹⁹ of link candidates. Finally, the weighted function is the similarity function sim . Note that ec is the minimal degree of nodes in G .

In a second step, we use the graph G as input for a graph clustering approach. The resulting clustering is assumed to be a partition \mathcal{V} of the set V of vertices of G . The informativeness of partition $V_i \in \mathcal{V}$ is set to $\max_{x \in V_i} \text{ifm}(x)$. The final step of our approach

¹⁹ A n -set is a set of magnitude n .

consists of selecting the most informative node from each of the k most informative partitions. These are merged to generate I^+ , which is sent as query to the oracle. The computation of I^- is carried out analogously. Note that this approach is generic in the sense that it can be combined with any graph clustering algorithm that can process weighted graphs as well as with any informativeness function ifm. Here, we use Border-Flow [118] as clustering algorithm because (1) it has been used successfully in several other applications such as the creation of SPARQL benchmarks [102] and the analysis of protein-protein interactions [108]. and (2) it is parameter-free and does not require any tuning.

Spreading Activation with Weight Decay. The idea behind spreading activation with weight decay (WD) is to combine the intra- and inter-class correlation to determine the informativeness of each link candidate. Here, we begin by computing the set $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$, where \mathcal{S}^+ and \mathcal{S}^- are described as above. Let s_i and s_j be the i^{th} and j^{th} elements of \mathcal{S} . We then compute the quadratic similarity matrix \mathcal{M} with entries $m_{ij} = \text{sim}(s_i, s_j)$ for $i \neq j$ and 0 else. Note that both negative and positive link candidates belong to \mathcal{S} . Thus, \mathcal{M} encodes both inter- and intra-class correlation. In addition to \mathcal{M} , we compute the activation vector \mathcal{A} by setting its entries to $a_i = \text{ifm}(s_i)$. In the following, \mathcal{A} is considered to be a column vector.

In a first step, we normalize the activation vector \mathcal{A} to ensure that the values contained therein do not grow indefinitely. Then, in a second step, we set $\mathcal{A} = \mathcal{A} + \mathcal{M} \times \mathcal{A}$. This has the effect of propagating the activation of each s to all its neighbors according to the weights of the edges between s and its neighbors. Note that elements of \mathcal{S}^+ that are close to elements of \mathcal{S}^- get a higher activation than elements of \mathcal{S}^+ that are further away from \mathcal{S}^- and vice-versa. Moreover, elements at the center of node clusters (i.e., elements that are probably no statistical outliers) also get a higher activation than elements that are probably outliers. The idea behind the weight decay step is to update the matrix by setting each m_{ij} to m_{ij}^r , where $r > 1$ is a fix exponent. This is the third step of the algorithm. Given that $\forall i \forall j m_{ij} \leq 1$, the entries in the matrix get smaller with time. By these means, the amount of activation transferred across long paths is reduced. We run this three-step procedure iteratively until all non-1 entries of the matrix are less or equal to a threshold $\epsilon = 10^{-2}$. The k elements of \mathcal{S}^+ resp. \mathcal{S}^- with maximal activation are returned as I^+ resp. I^- . In the example shown in Figure 25, while all nodes from \mathcal{S}^+ and \mathcal{S}^- start with the same activation, two nodes get the highest activation after only 3 iterations.

Evaluation. COALA was evaluated by running weight decay and clustering in combination with EAGLE, an active genetic programming approach for learning link specifications. Throughout the following experiments, EAGLE’s mutation and crossover rates were set to 0.6. Individuals were given a 70% chance to get selected for reproduction. The population sizes were set to 20 and 100. We set $k = 5$ and ran our experiments for 10 iterations. Between each iteration we evolved the populations for 50 generations. We ran our experiments on two real-world datasets and three synthetic datasets. The synthetic datasets consisted of the Persons1, Person2 and Restaurants datasets from the

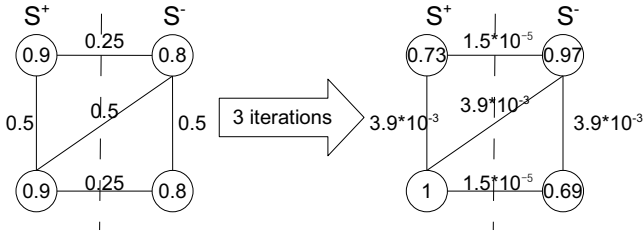


Fig. 25. Example of weight decay. Here r was set to 2. The left picture shows the initial activations and similarity scores while the right picture shows the results after 3 iterations. Note that for the sake of completeness the weights of the edges were not set to 0 when they reached ϵ .

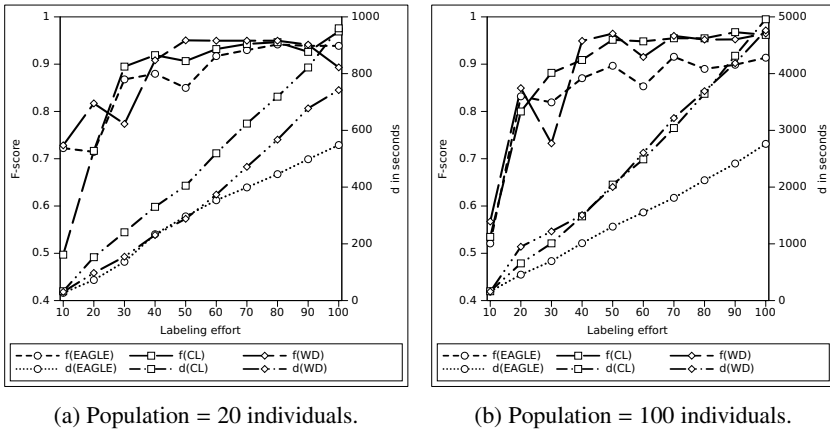


Fig. 26. F-score and runtime on the ACM-DBLP dataset. $f(X)$ stands for the F-score achieved by algorithm X, while $d(X)$ stands for the total duration required by the algorithm.

OAEI 2010 benchmark²⁰. The real-world datasets consisted of the ACM-DBLP and Abt-Buy datasets, which were extracted from websites or databases [76]²¹. Given that genetic programming is non-deterministic, all results presented below are the means of 5 runs. Each experiment was ran on a single thread of a server running JDK1.7 on Ubuntu 10.0.4 and was allocated maximally 2GB of RAM. The processors were 2.0GHz Quadcore AMD Opterons. An excerpt of the results is shown in Figure 26. While the results show that COALA outperform EAGLE, it remains unclear whether WD or CL is the best approach to achieving a faster convergence towards the optimal solution.

²⁰ <http://oaei.ontologymatching.org/2010/>

²¹ http://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution

5.7 Conclusion

We presented and discussed linking approaches for Linked Data and the challenges they face. In addition, we gave an overview of several state-of-the-art approaches for instance matching for Linked Data. We then presented time-efficient approaches for link discovery. Finally, we presented a state-of-the-art approach for the active learning of link specifications. This approach can be easily extended to learn specifications automatically.²² Novel challenges that need to be addressed include the automatic management of resources for link specifications. First works on running link discovery in parallel have shown that using massively parallel hardware such as GPUs can lead to better results than using cloud implementations even on considerably large datasets [114]. Detecting the right resources for linking automatically given a hardware landscape is yet still a dream to achieve.

6 Enrichment

The term *enrichment* in this chapter refers to the (semi-)automatic extension of a knowledge base schema. It describes the process of increasing the expressiveness and semantic richness of a knowledge base. Usually, this is achieved by adding or refining terminological axioms.

Enrichment methods can typically be applied in a *grass-roots* approach to knowledge base creation. In such an approach, the whole ontological structure is not created upfront, but evolves with the data in a knowledge base. Ideally, this enables a more agile development of knowledge bases. In particular, in the context of the Web of Linked Data such an approach appears to be an interesting alternative to more traditional ontology engineering methods. Amongst others, Tim Berners-Lee advocates to get “raw data now”²³ and worry about the more complex issues later.

Knowledge base enrichment can be seen as a sub-discipline of ontology learning. Ontology learning is more general in that it can rely on external sources, e.g. written text, to create an ontology. The term knowledge base enrichment is typically used when already existing data in the knowledge base is analysed to improve its schema.

Enrichment methods span several research areas like knowledge representation and reasoning, machine learning, statistics, natural language processing, formal concept analysis and game playing. Considering the variety of methods, we structure this section as follows: First, we give an overview of different types of enrichment and list some typical methods and give pointers to references, which allow the reader to obtain more information on a topic. In the second part, we describe a specific software – the ORE tool – in more detail.

6.1 State of the Art and Types of Enrichment

Ontology enrichment usually involves applying heuristics or machine learning techniques to find axioms, which can be added to an existing ontology. Naturally, different techniques have been applied depending on the specific type of axiom.

²² Such an extension is the basis of the self-configuration algorithm of the SAIM framework.

²³ http://www.ted.com/talks/tim_berniers_lee_on_the_next_web.html

One of the most complex tasks in ontology enrichment is to find *definitions* of classes. This is strongly related to Inductive Logic Programming (ILP) [121] and more specifically supervised learning in description logics. Research in those fields has many applications apart from being applied to enrich ontologies. For instance, it is used in the life sciences to detect whether drugs are likely to be efficient for particular diseases. Work on learning in description logics goes back to e.g. [33,34], which used so-called *least common subsumers* to solve the learning problem (a modified variant of the problem defined in this article). Later, [17] invented a refinement operator for \mathcal{ALER} and proposed to solve the problem by using a top-down approach. [40,66,67] combine both techniques and implement them in the YINYANG tool. However, those algorithms tend to produce very long and hard-to-understand class expressions. The algorithms implemented in DL-Learner [86,87,78,88,80] overcome this problem and investigate the learning problem and the use of top down refinement in detail. DL-FOIL [43] is a similar approach, which is based on a mixture of upward and downward refinement of class expressions. They use alternative measures in their evaluation, which take the open world assumption into account, which was not done in ILP previously. Most recently, [82] implements appropriate heuristics and adaptations for learning definitions in ontologies. The focus in this work is efficiency and practical application of learning methods. The article presents plugins for two ontology editors (Protégé and OntoWiki) as well stochastic methods, which improve previous methods by an order of magnitude. For this reason, we will analyse it in more detail in the next subsection. The algorithms presented in the article can also learn *super class axioms*.

A different approach to learning the definition of a named class is to compute the so called *most specific concept* (msc) for all instances of the class. The most specific concept of an individual is the most specific class expression, such that the individual is instance of the expression. One can then compute the *least common subsumer* (lcs) [16] of those expressions to obtain a description of the named class. However, in expressive description logics, an msc does not need to exist and the lcs is simply the disjunction of all expressions. For light-weight logics, such as \mathcal{EL} , the approach appears to be promising.

Other approaches, e.g. [91] focus on learning in hybrid knowledge bases combining ontologies and *rules*. Ontology evolution [92] has been discussed in this context. Usually, hybrid approaches are a generalisation of concept learning methods, which enable powerful rules at the cost of efficiency (because of the larger search space). Similar as in knowledge representation, the tradeoff between expressiveness of the target language and efficiency of learning algorithms is a critical choice in symbolic machine learning.

Another enrichment task is *knowledge base completion*. The goal of such a task is to make the knowledge base complete in a particular well-defined sense. For instance, a goal could be to ensure that all subclass relationships between named classes can be inferred. The line of work starting in [133] and further pursued in e.g. [15] investigates the use of *formal concept analysis* for completing knowledge bases. It is promising, although it may not be able to handle noise as well as a machine learning technique. A Protégé plugin [140] is available. [154] proposes to improve knowledge bases through relational exploration and implemented it in the *RELEXO framework*²⁴. It focuses on

²⁴ <http://code.google.com/p/relexo/>

simple relationships and the knowledge engineer is asked a series of questions. The knowledge engineer either must positively answer the question or provide a counterexample.

[155] focuses on learning *disjointness* between classes in an ontology to allow for more powerful reasoning and consistency checking. To achieve this, it can use the ontology itself, but also texts, e.g. Wikipedia articles corresponding to a concept. The article includes an extensive study, which shows that proper modelling disjointness is actually a difficult task, which can be simplified via this ontology enrichment method.

Another type of ontology enrichment is schema mapping. This task has been widely studied and will not be discussed in depth within this chapter. Instead, we refer to [31] for a survey on ontology mapping.

There are further more light-weight ontology enrichment methods. For instance, *taxonomies* can be learned from simple tag structures via heuristics [29,153]. Similarly, “properties of properties” can be derived via simple statistical analysis. This includes the detection whether a particular property might be symmetric, function, reflexive, inverse functional etc. Similarly, domains and ranges of properties can be determined from existing data. Enriching the schema with domain and range axioms allows to find cases, where properties are misused via OWL reasoning.

Table 3. Work in ontology enrichment grouped by type or aim of learned structures

Type/Aim	References
Taxonomies	[163,29,153]
Definitions	often done via ILP approaches such as [86,87,88,82,43,40,66,67,17], genetic approaches [78] have also been used
Super Class Axioms	[82,153,29]
Rules in Ontologies	[91,92]
Disjointness	[155]
Properties of properties	[29,46]
Alignment	challenges: [142], recent survey: [31]
Completion	formal concept analysis and relational exploration [15,154,140]

In the following subsection, we describe an enrichment approach for learning definitions and super class axioms in more detail. The algorithm was recently developed by the first authors and is described in full detail in [82].

6.2 Class Expression Learning in DL-Learner

The Semantic Web has recently seen a rise in the availability and usage of knowledge bases, as can be observed within the Linking Open Data Initiative, the TONES and Protégé ontology repositories, or the Watson search engine. Despite this growth, there is still a lack of knowledge bases that consist of sophisticated schema information and instance data adhering to this schema. Several knowledge bases, e.g. in the life sciences, only consist of schema information, while others are, to a large extent, a collection of facts without a clear structure, e.g. information extracted from data bases or texts.

The combination of sophisticated schema and instance data allows powerful reasoning, consistency checking, and improved querying possibilities. We argue that being able to learn OWL class expressions²⁵ is a step towards achieving this goal.

Example 1. As an example, consider a knowledge base containing a class `Capital` and instances of this class, e.g. London, Paris, Washington, Canberra etc. A machine learning algorithm could, then, suggest that the class `Capital` may be equivalent to one of the following OWL class expressions in Manchester OWL syntax²⁶:

```
City and isCapitalOf at least one GeopoliticalRegion
City and isCapitalOf at least one Country
```

Both suggestions could be plausible: The first one is more general and includes cities that are capitals of states, whereas the latter one is stricter and limits the instances to capitals of countries. A knowledge engineer can decide which one is more appropriate, i.e. a semi-automatic approach is used, and the machine learning algorithm should guide her by pointing out which one fits the existing instances better. Assuming the knowledge engineer decides for the latter, an algorithm can show her whether there are instances of the class `Capital` which are neither instances of `City` nor related via the property `isCapitalOf` to an instance of `Country`.²⁷ The knowledge engineer can then continue to look at those instances and assign them to a different class as well as provide more complete information; thus improving the quality of the knowledge base. After adding the definition of `Capital`, an OWL reasoner can compute further instances of the class which have not been explicitly assigned before.

Using machine learning for the generation of suggestions instead of entering them manually has the advantage that 1.) the given suggestions fit the instance data, i.e. schema and instances are developed in concordance, and 2.) the entrance barrier for knowledge engineers is significantly lower, since understanding an OWL class expression is easier than analysing the structure of the knowledge base and creating a class expression manually. Disadvantages of the approach are the dependency on the availability of instance data in the knowledge base and requirements on the quality of the ontology, i.e. modelling errors in the ontology can reduce the quality of results.

Overall, we describe the following in this chapter:

- extension of an existing learning algorithm for learning class expressions to the ontology engineering scenario,
- presentation and evaluation of different heuristics,
- showcase how the enhanced ontology engineering process can be supported with plugins for Protégé and OntoWiki,
- evaluation of the presented algorithm with several real ontologies from various domains.

²⁵ http://www.w3.org/TR/owl2-syntax/#Class_Expressions

²⁶ For details on Manchester OWL syntax (e.g. used in Protégé, OntoWiki) see [63].

²⁷ This is not an inconsistency under the standard OWL open world assumption, but rather a hint towards a potential modelling error.

The adapted algorithm for solving the learning problems, which occur in the ontology engineering process, is called *CELOE* (*Class Expression Learning for Ontology Engineering*). It was implemented within the open-source framework DL-Learner.²⁸ DL-Learner [79,80] leverages a modular architecture, which allows to define different types of components: knowledge sources (e.g. OWL files), reasoners (e.g. DIG²⁹ or OWL API based), learning problems, and learning algorithms. In this overview, we focus on the latter two component types, i.e. we define the class expression learning problem in ontology engineering and provide an algorithm for solving it.

Learning Problem. The process of learning in logics, i.e. trying to find high-level explanations for given data, is also called *inductive reasoning* as opposed to *inference* or *deductive reasoning*. The main difference is that in deductive reasoning it is formally shown whether a statement follows from a knowledge base, whereas in inductive learning new statements are invented. Learning problems, which are similar to the one we will analyse, have been investigated in *Inductive Logic Programming* [121] and, in fact, the method presented here can be used to solve a variety of machine learning tasks apart from ontology engineering.

In the ontology learning problem we consider, we want to learn a formal description of a class A , which has (inferred or asserted) instances in the considered ontology. In the case that A is already described by a class expression C via axioms of the form $A \sqsubseteq C$ or $A \equiv C$, those can be either refined, i.e. specialised/generalised, or relearned from scratch by the learning algorithm. To define the class learning problem, we need the notion of a *retrieval* reasoner operation $R_{\mathcal{K}}(C)$. $R_{\mathcal{K}}(C)$ returns the set of all instances of C in a knowledge base \mathcal{K} . If \mathcal{K} is clear from the context, the subscript can be omitted.

Definition 6 (class learning problem). *Let an existing named class A in a knowledge base \mathcal{K} be given. The class learning problem is to find an expression C such that $R_{\mathcal{K}}(C) = R_{\mathcal{K}}(A)$.*

Clearly, the learned expression C is a description of (the instances of) A . Such an expression is a candidate for adding an axiom of the form $A \equiv C$ or $A \sqsubseteq C$ to the knowledge base \mathcal{K} . If a solution of the learning problem exists, then the used base learning algorithm (as presented in the following subsection) is complete, i.e. guaranteed to find a correct solution if one exists in the target language and there are no time and memory constraints (see [87,88] for the proof). In most cases, we will not find a solution to the learning problem, but rather an approximation. This is natural, since a knowledge base may contain false class assignments or some objects in the knowledge base are described at different levels of detail. For instance, in Example 1, the city “Apia” might be typed as “Capital” in a knowledge base, but not related to the country “Samoa”. However, if most of the other cities are related to countries via a role `isCapitalOf`, then the learning algorithm may still suggest City and `isCapitalOf` at least one Country since this describes the majority of capitals in the knowledge base well. If the knowledge engineer agrees with such a definition, then a tool can assist him in completing missing information about some capitals.

²⁸ <http://dl-learner.org>

²⁹ <http://dl.kr.org/dig/>

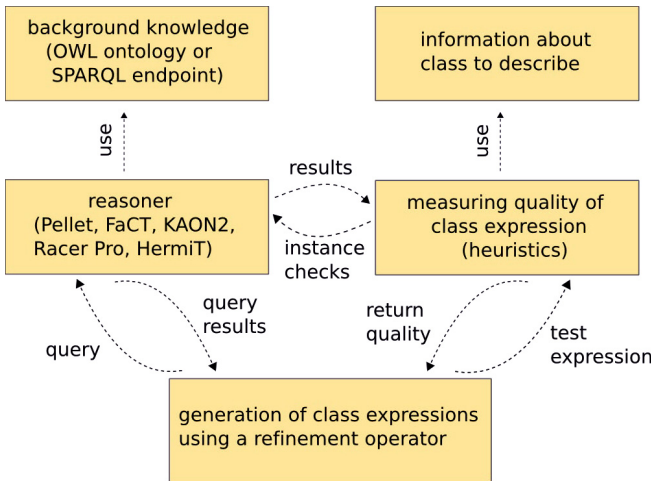


Fig. 27. Outline of the general learning approach in CELOE: One part of the algorithm is the generation of promising class expressions taking the available background knowledge into account. Another part is a heuristic measure of how close an expression is to being a solution of the learning problem. Figure adapted from [56].

According to Occam’s razor [26] simple solutions of the learning problem are to be preferred over more complex ones, because they are more readable. This is even more important in the ontology engineering context, where it is essential to suggest simple expressions to the knowledge engineer. We measure simplicity as the *length* of an expression, which is defined in a straightforward way, namely as the sum of the numbers of concept, role, quantifier, and connective symbols occurring in the expression. The algorithm is biased towards shorter expressions. Also note that, for simplicity the definition of the learning problem itself does enforce coverage, but not prediction, i.e. correct classification of objects which are added to the knowledge base in the future. Concepts with high coverage and poor prediction are said to *overfit* the data. However, due to the strong bias towards short expressions this problem occurs empirically rarely in description logics [88].

Base Learning Algorithm. Figure 27 gives a brief overview of the *CELOE* algorithm, which follows the common “generate and test” approach in ILP. This means that learning is seen as a search process and several class expressions are generated and tested against a background knowledge base. Each of those class expressions is evaluated using a heuristic, which is described in the next section. A challenging part of a learning algorithm is to decide which expressions to test. In particular, such a decision should take the computed heuristic values and the structure of the background knowledge into account. For *CELOE*, we use the approach described in [87,88] as base, where this problem has already been analysed, implemented, and evaluated in depth. It is based on the idea of *refinement operators*:

Definition 7 (refinement operator). A quasi-ordering is a reflexive and transitive relation. In a quasi-ordered space (S, \leq) a downward (upward) refinement operator ρ is a mapping from S to 2^S , such that for any $C \in S$ we have that $C' \in \rho(C)$ implies $C' \leq C$ ($C \leq C'$). C' is called a specialisation (generalisation) of C .

Refinement operators can be used for searching in the space of expressions. As ordering we can use subsumption. (Note that the subsumption relation \sqsubseteq is a quasi-ordering.) If an expression C subsumes an expression D ($D \sqsubseteq C$), then C will cover all examples which are covered by D . This makes subsumption a suitable order for searching in expressions as it allows to prune parts of the search space without losing possible solutions.

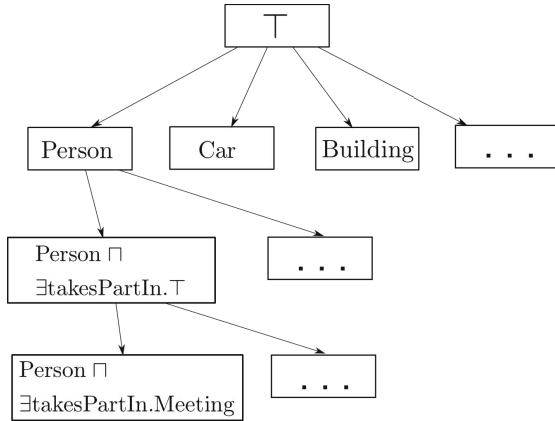


Fig. 28. Illustration of a search tree in a top down refinement approach

The approach we used is a top-down algorithm based on refinement operators as illustrated in Figure 28 (more detailed schemata can be found in the slides³⁰ of the ontology learning lecture of Reasoning Web 2010 [81]). This means that the first class expression, which will be tested is the most general expression (\top), which is then mapped to a set of more specific expressions by means of a downward refinement operator. Naturally, the refinement operator can be applied to the obtained expressions again, thereby spanning a *search tree*. The search tree can be pruned when an expression does not cover sufficiently many instances of the class A we want to describe. One example for a path in a search tree spanned up by a downward refinement operator is the following (\rightsquigarrow denotes a refinement step):

$$\begin{aligned} \top &\rightsquigarrow \text{Person} \rightsquigarrow \text{Person} \sqcap \text{takesPartIn} \top \\ &\rightsquigarrow \text{Person} \sqcap \text{takesPartIn.Meeting} \end{aligned}$$

The heart of such a learning strategy is to define a suitable refinement operator and an appropriate search heuristics for deciding which nodes in the search tree should be expanded. The refinement operator in the considered algorithm is defined in [88]. It is based on earlier work in [87] which in turn is built on the theoretical foundations of [86]. It has been shown to be the best achievable operator with respect to a set of properties (not further described here), which are used to assess the performance of

³⁰ <http://reasoningweb.org/2010/teaching-material/lehmann.pdf>

refinement operators. The learning algorithm supports conjunction, disjunction, negation, existential and universal quantifiers, cardinality restrictions, hasValue restrictions as well as boolean and double datatypes.

6.3 Finding a Suitable Heuristic

A heuristic measures how well a given class expression fits a learning problem and is used to guide the search in a learning process. To define a suitable heuristic, we first need to address the question of how to measure the accuracy of a class expression. We introduce several heuristics, which can be used for *CELOE* and later evaluate them.

We cannot simply use supervised learning from examples directly, since we do not have positive and negative examples available. We can try to tackle this problem by using the existing instances of the class as positive examples and the remaining instances as negative examples. This is illustrated in Figure 29, where \mathcal{K} stands for the knowledge base and A for the class to describe. We can then measure accuracy as the number of correctly classified examples divided by the number of all examples. This can be computed as follows for a class expression C and is known as *predictive accuracy* in Machine Learning:

$$\text{predacc}(C) = 1 - \frac{|R(A) \setminus R(C)| + |R(C) \setminus R(A)|}{n} \quad n = |\text{Ind}(\mathcal{K})|$$

Here, $\text{Ind}(\mathcal{K})$ stands for the set of individuals occurring in the knowledge base. $R(A) \setminus R(C)$ are the false negatives whereas $R(C) \setminus R(A)$ are false positives. n is the number of all examples, which is equal to the number of individuals in the knowledge base in this case. Apart from learning definitions, we also want to be able to learn super class axioms ($A \sqsubseteq C$). Naturally, in this scenario $R(C)$ should be a superset of $R(A)$. However, we still do want $R(C)$ to be as small as possible, otherwise \top would always be a solution. To reflect this in our accuracy computation, we penalise false negatives more than false positives by a factor of t ($t > 1$) and map the result to the interval $[0, 1]$:

$$\text{predacc}(C, t) = 1 - 2 \cdot \frac{t \cdot |R(A) \setminus R(C)| + |R(C) \setminus R(A)|}{(t + 1) \cdot n} \quad n = |\text{Ind}(\mathcal{K})|$$

While being straightforward, the outlined approach of casting class learning into a standard learning problem with positive and negative examples has the disadvantage that the number of negative examples will usually be much higher than the number of positive examples. As shown in Table 4, this may lead to overly optimistic estimates. More importantly, this accuracy measure has the drawback of having a dependency on the number of instances in the knowledge base.

Therefore, we investigated further heuristics, which overcome this problem, in particular by transferring common heuristics from information retrieval to the class learning problem:

1. *F-Measure*: F_β -Measure is based on *precision* and *recall* weighted by β . They can be computed for the class learning problem without having negative examples. Instead, we perform a retrieval for the expression C , which we want to evaluate.

We can then define precision as the percentage of instances of C , which are also instances of A and recall as percentage of instances of A , which are also instances of C . This is visualised in Figure 29. F-Measure is defined as harmonic mean of precision and recall. For learning super classes, we use F_3 measure by default, which gives recall a higher weight than precision.

2. *A-Measure*: We denote the arithmetic mean of precision and recall as A-Measure. Super class learning is achieved by assigning a higher weight to recall. Using the arithmetic mean of precision and recall is uncommon in Machine Learning, since it results in too optimistic estimates. However, we found that it is useful in super class learning, where F_n is often too pessimistic even for higher n .
3. *Generalised F-Measure*: Generalised F-Measure has been published in [36] and extends the idea of F-measure by taking the three valued nature of classification in OWL/DLs into account: An individual can either belong to a class, the negation of a class or none of both cases can be proven. This differs from common binary classification tasks and, therefore, appropriate measures have been introduced (see [36] for details). Adaption for super class learning can be done in a similar fashion as for F-Measure itself.
4. *Jaccard Distance*: Since $R(A)$ and $R(C)$ are sets, we can use the well-known Jaccard coefficient to measure the similarity between both sets.

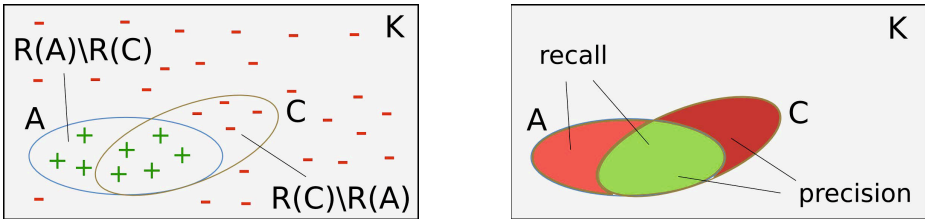
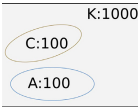
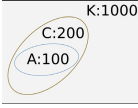
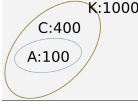
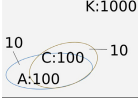



Fig. 29. Visualisation of different accuracy measurement approaches. \mathcal{K} is the knowledge base, A the class to describe and C a class expression to be tested. Left side: Standard supervised approach based on using positive (instances of A) and negative (remaining instances) examples. Here, the accuracy of C depends on the number of individuals in the knowledge base. Right side: Evaluation based on two criteria: recall (*Which fraction of $R(A)$ is in $R(C)$?*) and precision (*Which fraction of $R(C)$ is in $R(A)$?*).

We argue that those four measures are more appropriate than predictive accuracy when applying standard learning algorithms to the ontology engineering use case. Table 4 provides some example calculations, which allow the reader to compare the different heuristics.

Efficient Heuristic Computation. Several optimisations for computing the heuristics are described in [82]. In particular, adapted approximate reasoning and stochastic approximations are discussed. Those improvements have shown to lead to order of magnitude gains in efficiency for many ontologies. We refrain from describing those methods in this chapter.

Table 4. Example accuracies for selected cases (eq = equivalence class axiom, sc = super class axiom). The images on the left represent an imaginary knowledge base \mathcal{K} with 1000 individuals, where we want to describe the class A by using expression C . It is apparent that using predictive accuracy leads to impractical accuracies, e.g. in the first row C cannot possibly be a good description of A , but we still get 80% accuracy, since all the negative examples outside of A and C are correctly classified.

illustration	pred. acc.		F-Measure		A-Measure		Jaccard
	eq	sc	eq	sc	eq	sc	
	80%	67%	0%	0%	0%	0%	0%
	90%	92%	67%	73%	75%	88%	50%
	70%	75%	40%	48%	63%	82%	25%
	98%	97%	90%	90%	90%	90%	82%
	95%	88%	67%	61%	75%	63%	50%

The Protégé Plugin. After implementing and testing the described learning algorithm, we integrated it into *Protégé* and *OntoWiki*. Together with the Protégé developers, we extended the Protégé 4 plugin mechanism to be able to seamlessly integrate the DL-Learner plugin as an additional method to create class expressions. This means that the knowledge engineer can use the algorithm exactly where it is needed without any additional configuration steps. The plugin has also become part of the official Protégé 4 repository, i.e. it can be directly installed from within Protégé.

A screenshot of the plugin is shown in Figure 30. To use the plugin, the knowledge engineer is only required to press a button, which then starts a new thread in the background. This thread executes the learning algorithm. The used algorithm is an *anytime algorithm*, i.e. at each point in time we can always see the currently best suggestions. The GUI updates the suggestion list each second until the maximum runtime – 10 seconds by default – is reached. This means that the perceived runtime, i.e. the time after which only minor updates occur in the suggestion list, is often only one or two seconds for small ontologies. For each suggestion, the plugin displays its accuracy.

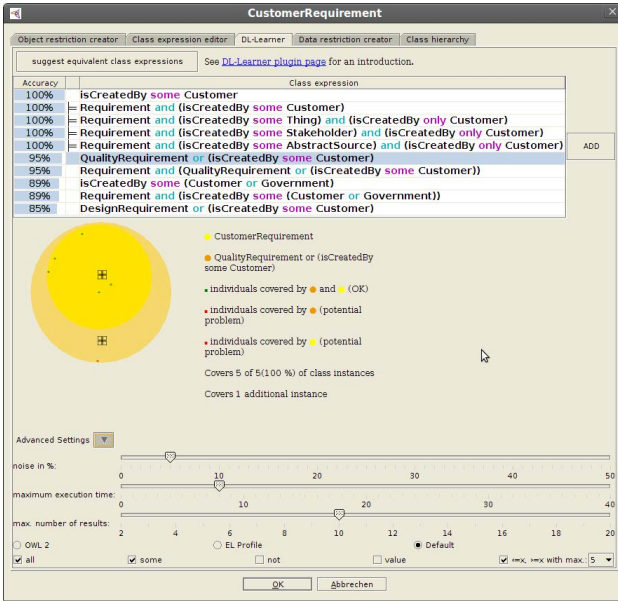


Fig. 30. A screenshot of the DL-Learner Protégé plugin. It is integrated as additional tab to create class expressions in Protégé. The user is only required to press the “suggest equivalent class expressions” button and within a few seconds they will be displayed ordered by accuracy. If desired, the knowledge engineer can visualize the instances of the expression to detect potential problems. At the bottom, optional expert configuration settings can be adopted.

When clicking on a suggestion, it is visualized by displaying two circles: One stands for the instances of the class to describe and another circle for the instances of the suggested class expression. Ideally, both circles overlap completely, but in practice this will often not be the case. Clicking on the plus symbol in each circle shows its list of individuals. Those individuals are also presented as points in the circles and moving the mouse over such a point shows information about the respective individual. Red points show potential problems detected by the plugin. Please note that we use closed world reasoning to detect those problems. For instance, in our initial example, a capital which is not related via the property `isCapitalOf` to an instance of `Country` is marked red. If there is not only a potential problem, but adding the expression would render the ontology inconsistent, the suggestion is marked red and a warning message is displayed. Accepting such a suggestion can still be a good choice, because the problem often lies elsewhere in the knowledge base, but was not obvious before, since the ontology was not sufficiently expressive for reasoners to detect it. This is illustrated by a screencast available from the plugin homepage,³¹ where the ontology becomes inconsistent after adding the axiom, and the real source of the problem is fixed afterwards. Being able to make such suggestions can be seen as a strength of the plugin.

The plugin allows the knowledge engineer to change expert settings. Those settings include the maximum suggestion search time, the number of results returned and settings related to the desired target language, e.g. the knowledge engineer can choose to stay within the OWL 2 EL profile or enable/disable certain class expression constructors. The learning algorithm is designed to be able to handle noisy data and the visualisation of the suggestions will reveal false class assignments so that they can be fixed afterwards.

³¹ <http://dl-learner.org/wiki/ProtegePlugin>

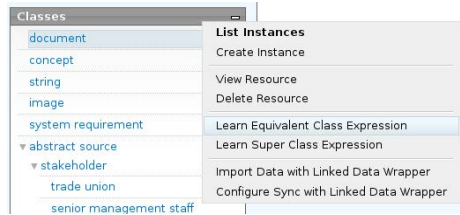


Fig. 31. The DL-Learner plugin can be invoked from the context menu of a class in OntoWiki

The OntoWiki Plugin. Analogous to Protégé, we created a similar plugin for OntoWiki (cf. section 4). OntoWiki is a lightweight ontology editor, which allows distributed and collaborative editing of knowledge bases. It focuses on wiki-like, simple and intuitive authoring of semantic content, e.g. through inline editing of RDF content, and provides different views on instance data.

Recently, a fine-grained plugin mechanism and extensions architecture was added to OntoWiki. The DL-Learner plugin is technically realised by implementing an OntoWiki component, which contains the core functionality, and a module, which implements the UI embedding. The DL-Learner plugin can be invoked from several places in OntoWiki, for instance through the context menu of classes as shown in Figure 31.

The plugin accesses DL-Learner functionality through its WSDL-based web service interface. Jar files containing all necessary libraries are provided by the plugin. If a user invokes the plugin, it scans whether the web service is online at its default address. If not, it is started automatically.

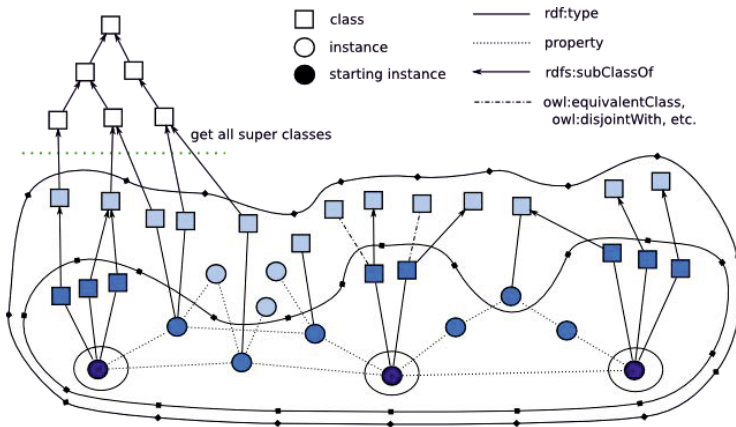


Fig. 32. Extraction with three starting instances. The circles represent different recursion depths. The circles around the starting instances signify recursion depth 0. The larger inner circle represents the fragment with recursion depth 1 and the largest outer circle with recursion depth 2. Figure taken from [56].

A major technical difference compared to the Protégé plugin is that the knowledge base is accessed via SPARQL, since OntoWiki is a SPARQL-based web application. In Protégé, the current state of the knowledge base is stored in memory in a Java object. As a result, we cannot easily apply a reasoner on an OntoWiki knowledge base. To overcome this problem, we use the DL-Learner fragment selection mechanism described in [56,57,30]. Starting from a set of instances, the mechanism extracts a relevant fragment from the underlying knowledge base up to some specified recursion depth. Figure 32 provides an overview of the fragment selection process. The fragment has the property that learning results on it are similar to those on the complete knowledge base. For a detailed description we refer the reader to the full article.

The fragment selection is only performed for medium to large-sized knowledge bases. Small knowledge bases are retrieved completely and loaded into the reasoner. While the fragment selection can cause a delay of several seconds before the learning algorithm starts, it also offers flexibility and scalability. For instance, we can learn class expressions in large knowledge bases such as DBpedia in OntoWiki.³²

The screenshot displays the DL-Learner interface within the OntoWiki environment. On the left, the OntoWiki navigation pane shows the 'Navigation: Classes' section with 'requirement' selected. The main window, titled 'DL-Learner - Learnt Class Expressions', shows a table of 'Suggested Equivalence Classes for customer requirement'. The table lists several suggestions with their accuracy values and toggle buttons for details. A context menu is open over the 84% accuracy entry, showing options like 'View Resource', 'Delete Resource', and 'Learn Equivalent Class Expression'. A circular progress indicator on the right shows the current state of learning.

accuracy	toggle all	suggested class expressions
100%	toggle details	is created by some Customer
100%	toggle details	Requirement and is created by some Customer
93%	toggle details	QualityRequirement or is created by some Customer
93%	toggle details	Requirement and (QualityRequirement or is created by some Customer)
84%	toggle details	Requirement and is created by some (Customer or Government)
79%	toggle details	every User Activity - User Can Access Data From Every Computer
79%	toggle details	requirement or is created by some Customer)

Fig. 33. Screenshot of the result table of the DL-Learner plugin in OntoWiki

Figure 33 shows a screenshot of the OntoWiki plugin applied to the SWORE [130] ontology. Suggestions for learning the class “customer requirement” are shown in Manchester OWL Syntax. Similar to the Protégé plugin, the user is presented a table of suggestions along with their accuracy value. Additional details about the instances of “customer requirement”, covered by a suggested class expressions and additionally contained instances can be viewed via a toggle button. The modular design of OntoWiki allows rich user interaction: Each resource, e.g. a class, property, or individual, can be viewed and subsequently modified directly from the result table as shown for “design

³² OntoWiki is undergoing an extensive development, aiming to support handling such large knowledge bases. A release supporting this is expected for the first half of 2012.

Table 5. Statistics about test ontologies

Ontology	#logical axioms	#classes	#object properties	#data properties	#individuals	DL expressivity
SC Ontology ³³	20081	28	8	5	3542	$\mathcal{AL}(\mathcal{D})$
Adhesome ³⁴	12043	40	33	37	2032	$\mathcal{ALCHN}(\mathcal{D})$
GeoSkills ³⁵	14966	613	23	21	2620	$\mathcal{ALCHOIN}(\mathcal{D})$
Eukariotic ³⁶	38	11	1	0	11	\mathcal{ALCON}
Breast Cancer ³⁷	878	196	22	3	113	$\mathcal{ALCROF}(\mathcal{D})$
Economy ³⁸	1625	339	45	8	482	$\mathcal{ALCH}(\mathcal{D})$
Resist ³⁹	239	349	134	38	75	$\mathcal{ALUF}(\mathcal{D})$
Finance ⁴⁰	16014	323	247	74	2466	$\mathcal{ALCROIQ}(\mathcal{D})$
Earthrealm ⁴¹	931	2364	215	36	171	$\mathcal{ALCHO}(\mathcal{D})$

requirement” in the screenshot. For instance, a knowledge engineer could decide to import additional information available as Linked Data and run the CELOE algorithm again to see whether different suggestions are provided with additional background knowledge.

Evaluation. To evaluate the suggestions made by our learning algorithm, we tested it on a variety of real-world ontologies of different sizes and domains. Please note that we intentionally do not perform an evaluation of the machine learning technique as such on existing benchmarks, since we build on the base algorithm already evaluated in detail in [88]. It was shown that this algorithm is superior to other supervised learning algorithms for OWL and at least competitive with the state of the art in ILP. Instead, we focus on its use within the ontology engineering scenario. The goals of the evaluation are to 1. determine the influence of reasoning and heuristics on suggestions, 2. to evaluate whether the method is sufficiently efficient to work on large real-world ontologies.

To perform the evaluation, we wrote a dedicated plugin for the Protégé ontology editor. This allows the evaluators to browse the ontology while deciding whether the suggestions made are reasonable. The plugin works as follows: First, all classes with at least 5 inferred instances are determined. For each such class, we run CELOE with different settings to generate suggestions for definitions. Specifically, we tested two

³³ <http://www.mindswap.org/ontologies/SC.owl>

³⁴ <http://www.sbcny.org/datasets/adhesome.owl>

³⁵ <http://i2geo.net/ontologies/current/GeoSkills.owl>

³⁶ <http://www.co-ode.org/ontologies/eukariotic/2005/06/01/eukariotic.owl>

³⁷ <http://acl.icnet.uk/%7Emw/MDM0.73.owl>

³⁸ <http://reliant.teknowledge.com/DAML/Economy.owl>

³⁹ <http://www.ecs.soton.ac.uk/~aoj04r/resist.owl>

⁴⁰ <http://www.fadyart.com/Finance.owl>

⁴¹ <http://sweet.jp1.nasa.gov/1.1/earthrealm.owl>

reasoners and five different heuristics. The two reasoners are standard Pellet and Pellet combined with approximate reasoning (not described in detail here). The five heuristics are those described in Section 6.3. For each configuration of CELOE, we generate at most 10 suggestions exceeding a heuristic threshold of 90%. Overall, this means that there can be at most $2 * 5 * 10 = 100$ suggestions per class – usually less, because different settings of CELOE will still result in similar suggestions. This list is shuffled and presented to the evaluators. For each suggestion, the evaluators can choose between 6 options (see Table 6):

- 1 the suggestion improves the ontology (improvement)
- 2 the suggestion is no improvement and should not be included (not acceptable) and
- 3 adding the suggestion would be a modelling error (error)

In the case of existing definitions for class A, we removed them prior to learning. In this case, the evaluator could choose between three further options:

- 4 the learned definition is equal to the previous one and both are good (equal +)
- 5 the learned definition is equal to the previous one and both are bad (equal -) and
- 6 the learned definition is inferior to the previous one (inferior).

We used the default settings of CELOE, e.g. a maximum execution time of 10 seconds for the algorithm. The knowledge engineers were five experienced members of our research group, who made themselves familiar with the domain of the test ontologies. Each researcher worked independently and had to make 998 decisions for 92 classes between one of the options. The time required to make those decisions was approximately 40 working hours per researcher. The raw agreement value of all evaluators is 0.535 (see e.g. [4] for details) with 4 out of 5 evaluators in strong pairwise agreement (90%). The evaluation machine was a notebook with a 2 GHz CPU and 3 GB RAM.

Table 6 shows the evaluation results. All ontologies were taken from the Protégé OWL⁴² and TONES⁴³ repositories. We randomly selected 5 ontologies comprising instance data from these two repositories, specifically the Earthrealm, Finance, Resist, Economy and Breast Cancer ontologies (see Table 5).

The results in Table 6 show which options were selected by the evaluators. It clearly indicates that the usage of approximate reasoning is sensible. The results are, however, more difficult to interpret with regard to the different employed heuristics. Using predictive accuracy did not yield good results and, surprisingly, generalised F-Measure also had a lower percentage of cases where option 1 was selected. The other three heuristics generated very similar results. One reason is that those heuristics are all based on precision and recall, but in addition the low quality of some of the randomly selected test ontologies posed a problem. In cases of too many very severe modelling errors, e.g. conjunctions and disjunctions mixed up in an ontology or inappropriate domain and range restrictions, the quality of suggestions decreases for each of the heuristics. This is the main reason why the results for the different heuristics are very close. Particularly, generalised F-Measure can show its strengths mainly for properly designed ontologies. For instance, column 2 of Table 6 shows that it missed 7% of possible improvements.

⁴² http://protegewiki.stanford.edu/index.php/Protege_Ontology_Library

⁴³ <http://owl.cs.manchester.ac.uk/repository/>

Table 6. Options chosen by evaluators aggregated by class. FIC stands for the fast instance checker, which is an approximate reasoning procedure.

reasoner/heuristic	improvement	equal quality (+)	equal quality (-)	inferior	not acceptable	error	missed improvements in %	selected position on suggestion list (incl. std. deviation)	avg. accuracy of selected suggestion in %
Pellet/F-Measure	16.70	0.44	0.66	0.00	64.66	17.54	14.95	2.82 ± 2.93	96.91
Pellet/Gen. F-Measure	15.24	0.44	0.66	0.11	66.60	16.95	16.30	2.78 ± 3.01	92.76
Pellet/A-Measure	16.70	0.44	0.66	0.00	64.66	17.54	14.95	2.84 ± 2.93	98.59
Pellet/pred. acc.	16.59	0.44	0.66	0.00	64.83	17.48	15.22	2.69 ± 2.82	98.05
Pellet/Jaccard	16.81	0.44	0.66	0.00	64.66	17.43	14.67	2.80 ± 2.91	95.26
Pellet FIC/F-Measure	36.30	0.55	0.55	0.11	52.62	9.87	1.90	2.25 ± 2.74	95.01
Pellet FIC/Gen. F-M.	33.41	0.44	0.66	0.00	53.41	12.09	7.07	1.77 ± 2.69	89.42
Pellet FIC/A-Measure	36.19	0.55	0.55	0.00	52.84	9.87	1.63	2.21 ± 2.71	98.65
Pellet FIC/pred. acc.	32.99	0.55	0.55	0.11	55.58	10.22	4.35	2.17 ± 2.55	98.92
Pellet FIC/Jaccard	36.30	0.55	0.55	0.11	52.62	9.87	1.90	2.25 ± 2.74	94.07

This means that for 7% of all classes, one of the other four heuristics was able to find an appropriate definition, which was not suggested when employing generalised F-Measure. The last column in this table shows that the average value of generalised F-Measure is quite low. As explained previously, it distinguishes between cases when an individual is instance of the observed class expression, its negation, or none of both. In many cases, the reasoner could not detect that an individual is instance of the negation of a class expression, because of the absence of disjointness axioms and negation in the knowledge base, which explains the low average values of generalised F-Measure. Column 4 of Table 6 shows that many selected expressions are amongst the top 5 (out of 10) in the suggestion list, i.e. providing 10 suggestions appears to be a reasonable choice.

In general, the improvement rate is only at about 35% according to Table 6 whereas it usually exceeded 50% in preliminary experiments with other real-world ontologies with fewer or less severe modelling errors. Since CELOE is based on OWL reasoning, it is clear that schema modelling errors will have an impact on the quality of suggestions. As a consequence, we believe that the CELOE algorithm should be combined with ontology debugging techniques. We have obtained first positive results in this direction and plan to pursue it in future work. However, the evaluation also showed that CELOE does still work in ontologies, which probably were never verified by an OWL reasoner.

Summary. We presented the CELOE learning method specifically designed for extending OWL ontologies. Five heuristics were implemented and analysed in conjunction with CELOE along with several performance improvements. A method for approximating heuristic values has been introduced, which is useful beyond the ontology engineering scenario to solve the challenge of dealing with a large number of examples

in ILP [160]. Furthermore, we biased the algorithm towards short solutions and implemented optimisations to increase readability of the suggestions made. The resulting algorithm was implemented in the open source DL-Learner framework. We argue that CELOE is the first ILP based algorithm, which turns the idea of learning class expressions for extending ontologies into practice. CELOE is integrated into two plugins for the ontology editors Protégé and OntoWiki and can be invoked using just a few mouse clicks.

7 Linked Data Quality

Linked Open Data (LOD) has provided, over the past several years, an unprecedented volume of structured data currently amount to 50 billion facts, represented as RDF triples. Although publishing large amounts of data on the Web is certainly a step in the right direction, the published data is only as useful as its quality. On the Data Web we have very varying quality of information covering various domains since data is merged together from different autonomous evolving data sources on the Web. For example, data extracted from semi-structured or even unstructured sources, such as DBpedia, often contains inconsistencies as well as mis-represented and incomplete information. Despite data quality in LOD being an essential concept, the autonomy and openness of the information providers makes the web vulnerable to missing, inaccurate, incomplete, inconsistent or outdated information.

Data quality is commonly conceived as *fitness for use* [72,159] for a certain application or use case. However, even datasets with quality problems might be useful for certain applications, as long as the quality is in the required range. For example, in the case of DBpedia the data quality is perfectly sufficient for enriching Web search with facts or suggestions about common sense information, such as entertainment topics. In such a scenario, DBpedia can be used to show related movies and personal information, when a user searches for an actor. In this case, it is rather neglectable, when in relatively few cases, a related movie or some personal fact is missing. For developing a medical application, on the other hand, the quality of DBpedia is probably completely insufficient. It should be noted that even the traditional, document-oriented Web has content of varying quality and is still perceived to be extremely useful by most people.

Consequently, one of the key challenges is to determine the quality of datasets published on the Web and making this quality information explicitly available. Assuring data quality is particularly a challenge in LOD as it involves a set of autonomously evolving data sources. Other than on the document Web, where information quality can be only indirectly (e.g. via page rank) or vaguely defined, there are much more concrete and measurable data quality metrics available for structured information such as accuracy of facts, completeness, adequacy of semantic representation or degree of understandability.

In this chapter, we first define the basic concepts of data quality, then report the formal definitions of a set of 26 different dimensions along with their respective metrics identified in [167].

Thereafter, we describe a set of currently available tools specially designed to assess the quality of Linked Data.

7.1 Data Quality Concepts

In this section, we introduce the basic concepts of data quality to help the readers understand these terminologies in their consequent usage.

Data Quality. The term *data quality* is commonly conceived as a multi-dimensional construct with a popular definition as the "fitness for use" [72]. In case of the Semantic Web, there are varying concepts of data quality such as the semantic metadata on the one hand and the notion of link quality on the other. There are several characteristics of data quality that should be considered i.e. the completeness, accuracy, consistency and validity on the one hand and the representational consistency, conciseness as well as the timeliness, understandability, availability and verifiability on the other hand.

Data Quality Problems. A set of issues that can affect the potentiality of the applications that use the data are termed as data quality problems. The problems may vary from the incompleteness of data, inconsistency in representation, invalid syntax or inaccuracy.

Data Quality Dimensions and Metrics. Data quality assessment involves the measurement of quality *dimensions* (or *criteria*) that are relevant to the user. A data quality assessment *metric* (or *measure*) is a procedure for measuring an information quality dimension [24]. The metrics are basically heuristics designed to fit a specific assessment situation [89]. Since the dimensions are rather abstract concepts, the assessment metrics rely on quality *indicators* that can be used for the assessment of the quality of a data source w.r.t the criteria [47].

Data Quality Assessment Method. A data quality assessment methodology is the process of evaluating if a piece of data meets the information consumers need for a specific use case [24]. The process involves measuring the quality dimensions that are relevant to the user and comparing the assessment results with the users quality requirements.

7.2 Linked Data Quality Dimensions

In [167], a core set of 26 different data quality dimensions were reported that can be applied to assess the quality of Linked Data. These dimensions are divided into the following groups:

- Contextual dimensions
- Trust dimensions
- Intrinsic dimensions
- Accessibility dimensions
- Representational dimensions
- Dataset dynamicity

Figure 34 shows the classification of the dimensions into these 6 different groups as well as the relations between them.

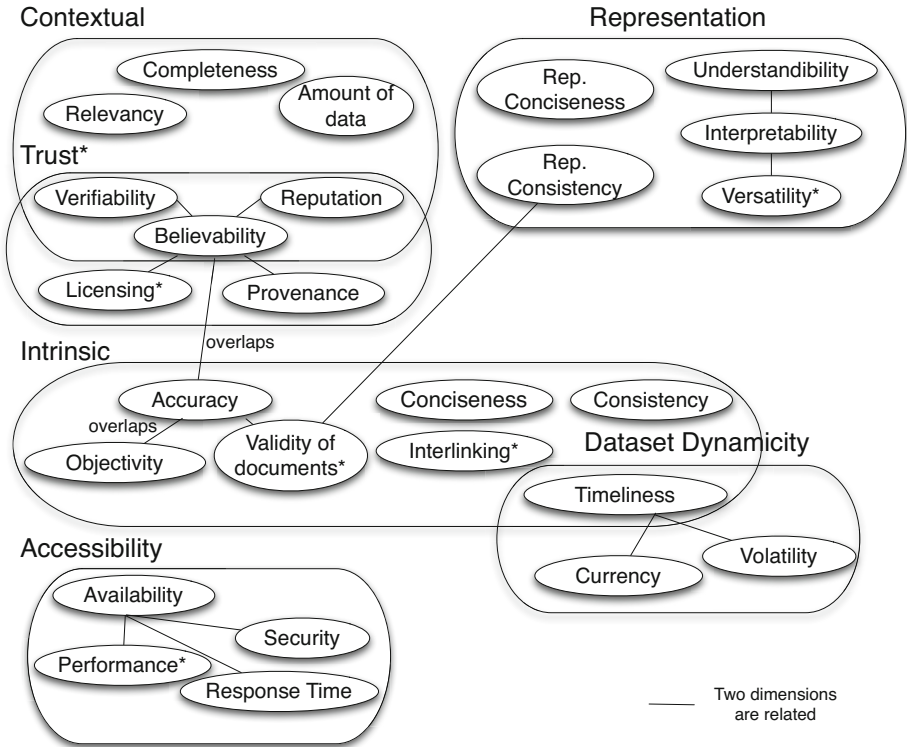


Fig. 34. Linked data quality dimensions and the relations between them [Source: [167]]

Use Case Scenario. Since data quality is described as "fitness to use", we introduce a specific use case that will allow us to illustrate the importance of each dimension with the help of an example. Our use case is about an intelligent flight search engine, which relies on acquiring (aggregating) data from several datasets. It obtains information about airports and airlines from an airline dataset (e.g., OurAirports⁴⁴, OpenFlights⁴⁵). Information about the location of countries, cities and particular addresses is obtained from a spatial dataset (e.g., LinkedGeoData⁴⁶). Additionally, aggregators in RDF pull all the information related to airlines from different data sources (e.g., Expedia⁴⁷, Tripadvisor⁴⁸, Skyscanner⁴⁹ etc.) that allows a user to query the integrated dataset for a flight from any start and end destination for any time period. We will use this scenario throughout this section as an example of how data quality influences fitness to use.

⁴⁴ <http://thedatahub.org/dataset/ourairports>

⁴⁵ <http://thedatahub.org/dataset/open-flights>

⁴⁶ linkedgeo.org

⁴⁷ <http://www.expedia.com/>

⁴⁸ <http://www.tripadvisor.com/>

⁴⁹ <http://www.skyscanner.de/>

Contextual Dimensions. Contextual dimensions are those that highly depend on the context of the task at hand as well as on the subjective preferences of the data consumer. There are three dimensions *completeness*, *amount-of-data* and *relevancy* that are part of this group.

Definition 8 (Completeness). *Completeness refers to the degree to which all required information is present in a particular dataset. In general, completeness is the extent to which data is of sufficient depth, breadth and scope for the task at hand. In terms of Linked Data, we classify completeness as follows:*

- *Schema completeness, the degree to which the classes and properties of an ontology are represented, thus can be called "ontology completeness",*
- *Property completeness, measure of the missing values for a specific property,*
- *Population completeness, the percentage of all real-world objects of a particular type that are represented in the datasets and*
- *Interlinking completeness, which has to be considered especially in Linked Data and refers to the degree to which instances in the dataset are interlinked.*

Metrics. Completeness can be measured by detecting the number of classes, properties, values and interlinks that are present in the dataset by comparing it to the original dataset (or gold standard dataset). It should be noted that in this case, users should assume a closed-world-assumption where a gold standard dataset is available and can be used to compare against.

Example. In the use case, the flight search engine should contain complete information so that it includes all offers for flights (population completeness). For example, a user residing in Germany wants to visit her friend in America. Since the user is a student, low price is of high importance. But, looking for flights individually on the airlines websites shows her flights with very expensive fares. However, using our flight search engine she finds all offers, even the less expensive ones and is also able to compare fares from different airlines and choose the most affordable one. The more complete the information for flights is, including cheap flights, the more visitors the site attracts. Moreover, sufficient interlinks between the datasets will allow her to query the integrated dataset so as to find an optimal route from the start to the end destination (interlinking completeness) in cases when there is no direct flight.

Definition 9 (Amount-of-data). *Amount-of-data refers to the quantity and volume of data that is appropriate for a particular task.*

Metrics. The amount-of-data can be measured in terms of bytes (most coarse-grained), triples, instances, and/or links present in the dataset. This amount should represent an appropriate volume of data for a particular task, that is, appropriate scope and level of detail.

Example. In the use case, the flight search engine acquires enough amount of data so as to cover all, even small, airports. In addition, the data also covers alternative means of transportation. This helps to provide the user with better travel plans, which includes smaller cities (airports). For example, when a user wants to travel from Connecticut to Santa Barbara, she might not find direct or indirect flights by searching individual

flights websites. But, using our example search engine, she is suggested convenient flight connections between the two destinations, because it contains a large amount of data so as to cover all the airports. She is also offered convenient combinations of planes, trains and buses. The provision of such information also necessitates the presence of a large amount of internal as well as external links between the datasets so as to provide a fine grained search for flights between specific places.

Definition 10 (Relevancy). *Relevancy refers to the provision of information which is in accordance with the task at hand and important to the users' query.*

Metrics. Relevancy is highly context dependent and can be measured by using meta-information attributes for assessing whether the content is relevant for a particular task. Additionally, retrieval of relevant documents can be performed using a combination of hyperlink analysis and information retrieval methods.

Example. When a user is looking for flights between any two cities, only relevant information i.e. start and end times, duration and cost per person should be provided. If a lot of irrelevant data is included in the spatial data, e.g. post offices, trees etc. (present in LinkedGeoData), query performance can decrease. The user may thus get lost in the silos of information and may not be able to browse through it efficiently to get only what she requires.

Trust Dimensions. Trust dimensions are those that focus on the trustworthiness of the dataset. There are five dimensions that are part of this group, namely, *provenance*, *verifiability*, *believability*, *reputation* and *licensing*.

Definition 11 (Provenance). *Provenance refers to the contextual metadata that focuses on how to represent, manage and use information about the origin of the source. Provenance helps to describe entities to enable trust, assess authenticity and allow reproducibility.*

Metrics. Provenance can be measured by analyzing the metadata associated with the source. This provenance information can in turn be used to assess the trustworthiness, reliability and credibility of a data source, an entity, a publishers or even individual RDF statements. There exists an inter-dependancy between the data provider and the data itself. On the one hand, data is likely to be accepted as true if it is provided by a trustworthy provider. On the other hand, the data provider is trustworthy if it provides true data. Thus, both can be checked to measure the trustworthiness.

Example. The example flight search engine constitutes information from several airline providers. In order to verify the reliability of these different airline data providers, provenance information from the aggregators can be analyzed and re-used so as enable users of the flight search engine to trust the authenticity of the data provided to them.

Definition 12 (Verifiability). *Verifiability refers to the degree by which a data consumer can assess the correctness of a dataset and as a consequence its trustworthiness.*

Metrics. Verifiability can be measured either by an unbiased third party, if the dataset itself points to the source or by the presence of a digital signature.

Example. In the use case, if we assume that the flight search engine crawls information from arbitrary airline websites, which publish flight information according to a standard vocabulary, there is a risk for receiving incorrect information from malicious websites. For instance, such a website publishes cheap flights just to attract a large number of visitors. In that case, the use of digital signatures for published RDF data allows to restrict crawling only to verified datasets.

Definition 13 (Reputation). *Reputation is a judgement made by a user to determine the integrity of a source. It is mainly associated with a data publisher, a person, organisation, group of people or community of practice rather than being a characteristic of a dataset. The data publisher should be identifiable for a certain (part of a) dataset.*

Metrics. Reputation is usually a score, for example, a real value between 0 (low) and 1 (high). There are different possibilities to determine reputation and can be classified into manual or (semi-)automated approaches. The manual approach is via a survey in a community or by questioning other members who can help to determine the reputation of a source or by the person who published a dataset. The (semi-)automated approach can be performed by the use of external links or page ranks.

Example. The provision of information on the reputation of data sources allows conflict resolution. For instance, several data sources report conflicting prices (or times) for a particular flight number. In that case, the search engine can decide to trust only the source with a higher reputation.

Definition 14 (Believability). *Believability is defined as the degree to which the information is accepted to be correct, true, real and credible.*

Metrics. Believability is measured by checking whether the contributor is contained in a list of trusted providers. In Linked Data, believability can be subjectively measured by analyzing the provenance information of the dataset.

Example. In the flight search engine use case, if the flight information is provided by trusted and well-known flights companies such as Lufthansa, British Airways, etc. then the user believes the information provided by their websites. She does not need to verify their credibility since these are well-known international flight companies. On the other hand, if the user retrieves information about an airline previously unknown, she can decide whether to believe this information by checking whether the airline is well-known or if it is contained in a list of trusted providers. Moreover, she will need to check the source website from which this information was obtained.

Definition 15 (Licensing). *Licensing is defined as a granting of the permission for a consumer to re-use a dataset under defined conditions.*

Metrics. Licensing can be checked by the indication of machine and human readable information associated with the dataset clearly indicating the permissions of data re-use.

Example. Since the example flight search engine aggregates data from several data sources, a clear indication of the license allows the search engine to re-use the data from the airlines websites. For example, the LinkedGeoData dataset is licensed under the Open Database License⁵⁰, which allows others to copy, distribute and use the data

⁵⁰ <http://opendatacommons.org/licenses/odbl/>

and produce work from the data allowing modifications and transformations. Due to the presence of this specific license, the flight search engine is able to re-use this dataset to pull geo-spatial information and feed it to the search engine.

Intrinsic Dimensions. Intrinsic dimensions are those that are independent of the user's context. These dimensions focus on whether information correctly represents the real world and whether information is logically consistent in itself. There are five dimensions that are part of this group, namely, *accuracy*, *objectivity*, *validity-of-documents*, *interlinking*, *consistency* and *conciseness*.

Definition 16 (Accuracy). *Accuracy can be defined as the extent to which data is correct, that is, the degree to which it correctly represents the real world facts and is also free of error. In particular, we associate accuracy mainly to semantic accuracy which relates to the correctness of a value to the actual real world value, that is, accuracy of the meaning.*

Metrics. Accuracy can be measured by checking the correctness of the data in a data source. That is, the detection of outliers or identification of semantically incorrect values through the violation of functional dependency rules. Accuracy is one of the dimensions, which is affected by assuming a closed or open world. When assuming an open world, it is more challenging to assess accuracy, since more logical constraints need to be specified for inferring logical contradictions.

Example. In the use case, let us suppose that a user is looking for flights between Paris and New York. Instead of returning flights starting from Paris, France, the search returns flights between Paris in Texas and New York. This kind of semantic inaccuracy in terms of labelling as well as classification can lead to erroneous results.

Definition 17 (Objectivity). *Objectivity is defined as the degree to which the interpretation and usage of data is unbiased, unprejudiced and impartial. This dimension highly depends on the type of information and therefore is classified as a subjective dimension.*

Metrics. Objectivity can not be measured qualitatively but indirectly by checking the authenticity of the source responsible for the information, whether the dataset is neutral or the publisher has a personal influence on the data provided. Additionally, it can be measured by checking whether independent sources can confirm a single fact.

Example. In the example flight search engine, consider the reviews available for each airline regarding the safety, comfort and prices. It may happen that an airline belonging to a particular alliance is ranked higher than others when in reality it is not so. This could be an indication of a bias where the review is falsified due to the providers preference or intentions. This kind of bias or partiality affects the user as she might be provided with incorrect information from expensive flights or from malicious websites.

Definition 18 (Validity-of-documents). *Validity-of-documents refers to the valid usage of the underlying vocabularies and the valid syntax of the documents (syntactic accuracy).*

Metrics. A syntax validator can be employed to assess the validity of a document, i.e. its syntactic correctness. The syntactic accuracy of entities can be measured by detecting

the erroneous or inaccurate annotations, classifications or representations. An RDF validator can be used to parse the RDF document and ensure that it is syntactically valid, that is, to check whether the document is in accordance with the RDF specification.

Example. Let us assume that the user is looking for flights between two specific locations, for instance Paris (France) and New York (United States) using the example flight search engine. However, the user is returned with no results. A possible reason for this is that one of the data sources incorrectly uses the property `geo:lon` to specify the longitude of "Paris" instead of using `geo:long`. This causes a query to retrieve no data when querying for flights starting near to a particular location.

Definition 19 (Interlinking). *Interlinking refers to the degree to which entities that represent the same concept are linked to each other.*

Metrics. Interlinking can be measured by using network measures that calculate the interlinking degree, cluster coefficient, sameAs chains, centrality and description richness through sameAs links.

Example. In the example flight search engine, the instance of the country "United States" in the airline dataset should be interlinked with the instance "America" in the spatial dataset. This interlinking can help when a user queries for a flight as the search engine can display the correct route from the start destination to the end destination by correctly combining information for the same country from both the datasets. Since names of various entities can have different URIs in different datasets, their interlinking can help in disambiguation.

Definition 20 (Consistency). *Consistency means that a knowledge base is free of (logical/formal) contradictions with respect to particular knowledge representation and inference mechanisms.*

Metrics. On the Linked Data Web, semantic knowledge representation techniques are employed, which come with certain inference and reasoning strategies for revealing implicit knowledge, which then might render a contradiction. Consistency is relative to a particular logic (set of inference rules) for identifying contradictions. A consequence of our definition of consistency is that a dataset can be consistent wrt. the RDF inference rules, but inconsistent when taking the OWL2-QL reasoning profile into account. For assessing consistency, we can employ an inference engine or a reasoner, which supports the respective expressivity of the underlying knowledge representation formalism. Additionally, we can detect functional dependency violations such as domain/range violations. In practice, RDF-Schema inference and reasoning with regard to the different OWL profiles can be used to measure consistency in a dataset. For domain specific applications, consistency rules can be defined, for example, according to the SWRL [64] or RIF standards [73] and processed using a rule engine.

Example. Let us assume a user is looking for flights between Paris and New York on the 21st of December, 2012. Her query returns the following results:

Flight	From	To	Arrival	Departure
A123	Paris	NewYork	14:50	22:35
A123	Paris	Singapore	14:50	22:35

The results show that the flight number A123 has two different destinations at the same date and same time of arrival and departure, which is inconsistent with the ontology definition that one flight can only have one destination at a specific time and date. This contradiction arises due to inconsistency in data representation, which can be detected by using inference and reasoning.

Definition 21 (Conciseness). *Conciseness refers to the redundancy of entities, be it at the schema or the data level. Thus, conciseness can be classified into:*

- *intensional conciseness (schema level) which refers to the redundant attributes and*
- *extensional conciseness (data level) which refers to the redundant objects.*

Metrics. As conciseness is classified in two categories, it can be measured by as the ratio between the number of unique attributes (properties) or unique objects (instances) compared to the overall number of attributes or objects respectively present in a dataset.

Example. In the example flight search engine, since data is fused from different datasets, an example of intensional conciseness would be a particular flight, say A123, being represented by two different identifiers in different datasets, such as <http://airlines.org/A123> and <http://flights.org/A123>. This redundancy can ideally be solved by fusing the two and keeping only one unique identifier. On the other hand, an example of extensional conciseness is when both these different identifiers of the same flight have the same information associated with them in both the datasets, thus duplicating the information.

Accessibility Dimensions. The dimensions belonging to this category involve aspects related to the way data can be accessed and retrieved. There are four dimensions part of this group, which are *availability*, *performance*, *security* and *response-time*.

Definition 22 (Availability). *Availability of a dataset is the extent to which information is present, obtainable and ready for use.*

Metrics. Availability of a dataset can be measured in terms of accessibility of the server, SPARQL endpoints or RDF dumps and also by the dereferencability of the URIs.

Example. Let us consider the case in which the user looks up a flight in the example flight search engine. However, instead of retrieving the results, she is presented with an error response code such as `4xx client error`. This is an indication that a requested resource is unavailable. In particular, when the returned error code is `404 Not Found` code, she may assume that either there is no information present at that specified URI or the information is unavailable. Naturally, an apparently unreliable system is less likely to be used, in which case the user may not book flights after encountering such issues.

Definition 23 (Performance). *Performance refers to the efficiency of a system that binds to a large dataset, that is, the more performant a data source the more efficiently a system can process data.*

Metrics. Performance is measured based on the scalability of the data source, that is a query should be answered in a reasonable amount of time. Also, detection of the usage

of prolix RDF features or usage of slash-URIs can help determine the performance of a dataset. Additional metrics are low latency and high throughput of the services provided for the dataset.

Example. In the use case example, the target performance may depend on the number of users, i.e. it may be required to be able to server 100 simultaneous users. Our flight search engine will not be scalable if the time required to answer to all queries is similar to the time required when querying the individual datasets. In that case, satisfying performance needs requires caching mechanisms.

Definition 24 (Security). *Security can be defined as the extent to which access to data can be restricted and hence protected against its illegal alteration and misuse. It refers to the degree to which information is passed securely from users to the information source and back.*

Metrics. Security can be measured based on whether the data has a proprietor or requires web security techniques (e.g. SSL or SSH) for users to access, acquire or re-use the data. The importance of security depends on whether the data needs to be protected and whether there is a cost of data becoming unintentionally available. For open data the protection aspect of security can be often neglected but the non-repudiation of the data is still an important issue. Digital signatures based on private-public key infrastructures can be employed to guarantee the authenticity of the data.

Example. In the example scenario, let us consider a user who wants to book a flight from a city A to a city B. The search engine should ensure a secure environment to the user during the payment transaction since her personal data is highly sensitive. If there is enough identifiable public information of the user, then she can be potentially targeted by private businesses, insurance companies etc. which she is unlikely to want. Thus, the use of SSL can be used to keep the information safe.

Definition 25 (Response-time). *Response-time measures the delay, usually in seconds, between submission of a query by the user and reception of the complete response from the dataset.*

Metrics. Response-time can be assessed by measuring the delay between submission of a request by the user and reception of the response from the dataset.

Example. A user is looking for a flight which includes multiple destinations using the example flight search engine. She wants to fly from Milan to Boston, Boston to New York and New York to Milan. In spite of the complexity of the query the search engine should respond quickly with all the flight details (including connections) for all the destinations.

Representational Dimensions. Representational dimensions capture aspects related to the design of the data such as the *representational-conciseness*, *representational-consistency*, *understandability*, *versatility* as well as the *interpretability* of the data.

Definition 26 (Representational-conciseness). *Representational-conciseness refers to the representation of the data which is compact and well formatted on the one hand but also clear and complete on the other hand.*

Metrics. Representational-conciseness is measured by qualitatively verifying whether the RDF model that is used to represent the data is concise enough in order to be self-descriptive and unambiguous.

Example. A user, after booking her flight, is interested in additional information about the destination airport such as its location. Our flight search engine should provide only that information related to the location rather than returning a chain of other properties.

Definition 27 (Representational-consistency). *Representational-consistency is the degree to which the format and structure of the information conform to previously returned information. Since Linked Data involves aggregation of data from multiple sources, we extend this definition to not only imply compatibility with previous data but also with data from other sources.*

Metrics. Representational-consistency can be assessed by detecting whether the dataset re-uses existing vocabularies or terms from existing established vocabularies to represent its entities.

Example. In the use case, consider different airlines companies using different notation for representing their data, e.g. some use RDF data and some others use turtle. In order to avoid interoperability issue, we provide data based on the Linked Data principle which is designed to support heterogeneous description models, which is necessary to handle different format of data. The exchange of information in different formats will not be a big deal in our search engine since strong links are created between datasets.

Definition 28 (Understandability). *Understandability refers to the ease with which data can be comprehended, without ambiguity, and used by a human consumer. Thus, this dimension can also be referred to as the comprehensibility of the information where the data should be of sufficient clarity in order to be used.*

Metrics. Understandability can be measured by detecting whether human-readable labels for classes, properties and entities are provided. Provision of the metadata of a dataset can also contribute towards assessing its understandability. The dataset should also clearly provide exemplary URIs and SPARQL queries along with the vocabularies used so that can users can understand how it can be used.

Example. Let us assume that the example flight search engine allows a user to enter a start and destination address. In that case, strings entered by the user need to be matched to entities in the spatial dataset⁴, probably via string similarity. Understandable labels for cities, places etc. improve search performance in that case. For instance, when a user looks for a flight to U.S (label), then the search engine should return the flights to the United States or America.

Definition 29 (Interpretability). *Interpretability refers to technical aspects of the data, that is whether information is represented using an appropriate notation and whether it conforms to the technical ability of the consumer.*

Metrics. Interpretability can be measured by the use of globally unique identifiers for objects and terms or by the use of appropriate language, symbols, units and clear definitions.

Example. Consider the example flight search engine wherein a user that is looking for a flight from Milan to Boston. Data related to Boston in the integrated data, for the required flight, contains the following entities:

- <http://rdf.freebase.com/ns/m.049jnng>
- <http://rdf.freebase.com/ns/m.043j22x>
- Boston Logan Airport

For the first two items no human-readable label is available, therefore the URI is displayed, which does not represent anything meaningful to the user besides the information that Freebase contains information about Boston Logan Airport. The third, however, contains a human-readable label, which the user can easily interpret.

Definition 30 (Versatility). *Versatility mainly refers to the alternative representations of data and its subsequent handling. Additionally, versatility also corresponds to the provision of alternative access methods for a dataset.*

Metrics. Versatility can be measured by the availability of the dataset in different serialisation formats, different languages as well as different access methods.

Example. Consider a user from a non-English speaking country who wants to use the example flight search engine. In order to cater to the needs of such users, our flight search engine should be available in different languages so that any user has the capability to understand it.

Dataset Dynamicity. An important aspect of data is its update or change over time. The main dimensions related to the dynamicity of a dataset proposed in the literature are *currency*, *volatility*, and *timeliness*.

Definition 31 (Currency). *Currency refers to the speed with which the information (state) is updated after the real-world information changes.*

Metrics. The measurement of currency relies on two components: (i) delivery time (the time when the data was last modified) and (ii) the current time, both possibly present in the data models.

Example. Consider a user who is looking for the price of a flight from Milan to Boston using the example search engine and she receives the updates via email. The currency of the price information is measured with respect to the last update of the price information. The email service sends the email with a delay of about 1 hour with respect to the time interval in which the information is determined to hold. In this way, if the currency value exceeds the time interval of the validity of the information, the result is said to be not current. If we suppose validity of the price information (the frequency of change of the price information) to be 2 hours, the price list that is not changed within this time is considered to be out-dated. To determine whether the information is out-dated or not we need to have temporal metadata available and represented by one of the data models proposed in the literature [134].

Definition 32 (Volatility). *Volatility can be defined as the length of time during which the data remains valid.*

Metrics. Volatility can be measured by two components: (i) the expiry time (the time when the data becomes invalid) and (ii) the input time (the time when the data was first published on the Web). Both these metrics are combined together to measure the distance between the expiry time and the input time of the published data.

Example. Let us consider the aforementioned use case where a user wants to book a flight from Milan to Boston and she is interested in the price information. The price is considered as volatile information as it changes frequently, for instance it is estimated that the flight price changes each minute (data remains valid for one minute). In order to have an updated price list, the price should be updated within the time interval pre-defined by the system (one minute). In case the user observes that the value is not re-calculated by the system within the last few minutes, the values are considered to be out-dated. Notice that volatility is estimated based on changes that are observed related to a specific data value.

Definition 33 (Timeliness). *Timeliness refers to the time point at which the data is actually used. This can be interpreted as whether the information is available in time to be useful.*

Metrics. Timeliness is measured by combining the two dimensions: currency and volatility. Additionally timeliness states the recency and frequency of data validation and does not include outdated data.

Example. A user wants to book a flight from Milan to Boston and the example flight search engine will return a list of different flight connections. She picks one of them and follows all the steps to book her flight. The data contained in the airline dataset shows a flight company that is available according to the user requirements. In terms of time-related quality dimension, the information related to the flight is recorded and reported to the user every two minutes which fulfils the requirement decided by our search engine that corresponds to the volatility of a flight information. Although the flight values are updated on time, the information received to the user about the flight's availability is not on time. In other words, the user did not perceive that the availability of the flight was depleted because the change was provided to the system a moment after her search.

7.3 Data Quality Assessment Frameworks

There are several efforts in developing data quality assessment frameworks (methodologies) in order to assess the data quality of LOD. These efforts are either semi-automated [47], automated [52] or manual [24,100,152]. Moreover, there is a lot of research performed extensively to assess the quality of LOD [61,62] and report commonly occurring problems that plague the existing datasets. In this section, we describe the various existing data quality assessment tools in terms of their usability, level of user interaction and applicability in terms of data quality assessment, also discussing their pros and cons.

Semi-automated. *Flemming's data quality assessment tool* [47], a semi-automated framework, is a simple user interface⁵¹. The tool presents an evaluation system to the

⁵¹ Available in German only at:

<http://linkeddata.informatik.hu-berlin.de/LDSrcAss/datenquelle.php>

user where she needs to only enter the SPARQL endpoint and a few example resources from a dataset. After specifying dataset details (endpoint, graphs, example URIs), the user is given an option for assigning weights to each of the pre-defined data quality metrics. Two options are available for assigning weights: (a) assigning a weight of 1 to all the metrics or (b) choosing the pre-defined exemplary weight of the metrics defined for a semantic data source. In the next step, the user is asked to answer a series of questions regarding the datasets, which are important indicators of the data quality for Linked Datasets and those which cannot be quantified. These include, for example, questions about the use of stable URIs, the number of obsolete classes and properties, and whether the dataset provides a mailing list. Next, the user is presented with a list of dimensions and metrics, for each of which weights can be specified again. Each metric is provided with two input fields: one showing the assigned weights and another with the calculated value.

Qualitätsbewertung von Datenquellen

Ausgabe der Ergebnisse

Auf dieser Seite erfolgt die Ausgabe der ermittelten Ergebnisse. Dabei werden die Bewertungen der Datenquelle bzgl. aller Indikatoren ausgewertet, um ihre Bewertung bzgl. der Merkmale sowie bzgl. des gesamten Bewertungssystems zu bestimmen. Sämtliche Zwischenergebnisse werden ausgegeben. Wurden die Bewertungen der Indikatoren durch eine Berechnungsmethode ermittelt, so wird zusätzlich eine Anmerkung zur Entstehung der Bewertung angegeben.

Der Qualitätswert der zu bewertenden Datenquelle LinkedCT, erreichbar unter der URI <http://linkedct.org>, beträgt **15 von 100** Punkten.

Die Stichprobe wurde anhand der folgenden URIs erstellt:

- <http://linkedct.org/resource/trial/nct00003202/>
- <http://linkedct.org/resource/trial/nct01208948/>
- <http://linkedct.org/resource/trial/nct01723163/>

Fig. 35. Excerpt of the Flemmings Data Quality Assessment tool showing the result of assessing the quality of LinkedCT with a score of 15 out of 100

In the end, the user is presented with a score ranging from 0 to 100, where 100 represents the best quality, representing the final data quality score. Additionally, the rating of each dimension and the total weight (based on 11 dimensions) is calculated using the user input from the previous steps. Figure 35 shows an excerpt of the tool showing the result of assessing the quality of LinkedCT with a score of 15 out of 100.

On one hand, the tool is easy to use with the form-based questions and adequate explanation for each step. Also, the assigning of the weights for each metric and the calculation of the score is straightforward and easy to adjust for each of the metrics. However, this tool has a few drawbacks: (1) the user needs to have adequate knowledge about the dataset in order to correctly assign weights for each of the metrics; (2) it does not drill down to the root cause of the data quality problem and (3) some of the main quality dimensions are missing from the analysis such as accuracy, completeness,

provenance, consistency, conciseness and relevancy as some could not be quantified and were not perceived to be true quality indicators.

Automated. The *LINK-QA framework* [52] takes a set of resources, SPARQL endpoints and/or dereferencable resources and a set of triples as input to automatically perform quality assessment and generates an HTML report. However, using this tool, the user cannot choose the dataset that she is interested in assessing.

Manual. The *WIQA* [24] and *Sieve* [100] frameworks also assess the quality of datasets but require a considerable amount of user involvement and are therefore considered manual tools. For instance, WIQA provides the user a wide range of policies to filter information and a set of quality criteria to assess the quality of the information.

Sieve, on the other hand, assists not only in the assessment of the quality of datasets but also in their fusion. It aims to use the data integration task as a means to increase completeness, conciseness and consistency in any chosen dataset. Sieve is a component of the *Linked Data Integration Framework (LDIF)*⁵² used first to assess the quality between two or more data sources and second to fuse (integrate) the data from the data sources based on their quality assessment.

In order to use this tool, a user needs to be conversant with programming. The input of Sieve is an LDIF provenance metadata graph generated from a data source. Based on this information the user needs to set the configuration property in an XML file known as *integration properties*. The quality assessment procedure relies on the measurement of metrics chosen by the user where each metric applies a scoring function having a value from 0 to 1.

Sieve implements only a few scoring functions such as *TimeCloseness*, *Preference*, *SetMembership*, *Threshold* and *Interval Membership* which are calculated based on the metadata provided as input along with the original data source. The configuration file is in XML format which should be modified based on the use case, as shown in Listing 1.1. The output scores are then used to fuse the data sources by applying one of the fusion functions, which are: *Filter*, *Average*, *Max*, *Min*, *First*, *KeepSingleValue* *ByQualityScore*, *Last*, *Random*, *PickMostFrequent*.

Listing 1.1. A configuration of Sieve: a data quality assessment and data fusion tool

```

1 <Sieve>
2 <QualityAssessment>
3 <AssessmentMetric id="sieve:recency">
4 <ScoringFunction class="TimeCloseness">
5 <Param name="timeSpan" value="7"/>
6 <Input path="?GRAPH/provenance:lasUpdated"/>
7 </ScoringFunction>
8 </AssessmentMetric>
9 <AssessmentMetric id="sieve:reputation">
10 <ScoringFunction class="ScoredList">
11 <Param name="priority" value="http://pt.wikipedia.org http://en.wikipedia.org"/>
12 <Input path="?GRAPH/provenance:lasUpdated"/>
13 </ScoringFunction>
14 </AssessmentMetric>
15 </Sieve>

```

⁵² <http://ldif.wbsg.de/>

In addition, users should specify the `DataSource` folder, the homepage element that refers to the data source from which the entities are going to be fused. Second, the XML file of the `ImportJobs` that downloads the data to the server should also be modified. In particular, the user should set up the `dumpLocation` element as the location of the dump file.

Although the tool is very useful overall, there are some drawbacks that decreases its usability: (1) the tool is not mainly used to assess data quality for a source, but instead to perform data fusion (integration) based on quality assessment. Therefore, the quality assessment can be considered as an accessory that leverages the process of data fusion through evaluation of few quality indicators; (2) it does not provide a user interface, ultimately limiting its usage to end-users with programming skills; (3) its usage is limited to domains providing provenance metadata associated with the data source.

LODGR`Refine`⁵³ [152], a LOD-enabled version of Google Refine, is an open-source tool for refining messy data. Although this tool is not focused on data quality assessment per se, it is powerful in performing preliminary cleaning or refining of raw data. Using this tool, one is able to import several different file types of data (CSV, Excel, XML, RDF/XML, N-Triples or even JSON) and then perform cleaning action via a browser-based interface. By using a diverse set of filters and facets on individual columns, LODGR`Refine` can help a user to semi-automate the cleaning of her data.

For example, this tool can help to detect duplicates, discover patterns (e.g. alternative forms of an abbreviation), spot inconsistencies (e.g. trailing white spaces) or find and replace blank cells. Additionally, this tool allows users to reconcile data, that is to connect a dataset to existing vocabularies such that it gives meaning to the values. Reconciliations to *Freebase*⁵⁴ helps mapping ambiguous textual values to precisely identified Freebase entities. Reconciling using *Sindice* or based on standard SPARQL or SPARQL with full-text search is also possible⁵⁵ using this tool. Moreover, it is also possible to extend the reconciled data with DBpedia as well as export the data as RDF, which adds to the uniformity and usability of the dataset.

These feature thus assists in assessing as well as improving the data quality of a dataset. Moreover, by providing external links, the interlinking of the dataset is considerably improved. LODGR`Refine` is easy to download and install as well as to upload and perform basic cleansing steps on raw data. The features of reconciliation, extending the data with DBpedia, transforming and exporting the data as RDF are added advantages. However, this tool has a few drawbacks: (1) the user is not able to perform detailed high level data quality analysis utilizing the various quality dimensions using this tool; (2) performing cleansing over a large dataset is time consuming as the tool follows a column data model and thus the user must perform transformations per column.

8 Outlook and Future Challenges

Although the different approaches for aspects of the Linked Data life-cycle as presented in this chapter are already working together, more effort must be done to further

⁵³ <http://code.zemanta.com/sparkica/>

⁵⁴ <http://www.freebase.com/>

⁵⁵ <http://refine.deri.ie/reconciliationDocs>

integrate them in ways that they mutually fertilize themselves. The discovery of new links or the authoring of new resource descriptions, for example, should automatically trigger the enrichment of the linked knowledge bases. The enrichment in turn can trigger the application of inconsistency detection and repair techniques. This leads to recognizing data quality problems and their consequent assessment and improvement. The browsing and exploration paths followed by end-users can be taken into account for machine learning techniques to refine the knowledge bases etc. Ultimately, when the different aspects of Linked Data management are fully integrated we envision the Web of Data becoming a washing machine for knowledge. A progress in one particular aspect will automatically trigger improvements in many other ones as well. In the following we outline some research challenges and promising research directions regarding some of the Linked Data management aspects.

Extraction. One promising research direction with regard to the extraction from unstructured sources is the development of standardized, LOD enabled integration interfaces between existing NLP tools. An open question is whether and how efficient bi-directional synchronization between extraction source and target knowledge base can be established. With regard to the extraction from structured sources (e.g. relational, XML) we need a declarative syntax and semantics for data model transformations. Some orthogonal challenges include the use of LOD as background knowledge and the representation and tracing of provenance information.

Authoring. Current Semantic Wikis still suffer from a lack of scalability. Hence, an important research and development target are large-scale Semantic Wikis, which include functionality for access control and provenance. In order to further flexibilize and simplify the authoring an adaptive choreography of editing widgets based on underlying data structures is needed. Also, the joint authoring of unstructured and structured sources (i.e. HTML/RDFa authoring) and better support for the integrated semantic annotation of other modalities such as images, audio, video is of paramount importance.

Natural Language Queries. One of the future challenges for Linked Data is to create user interfaces, which are able to hide the complexity of the underlying systems. A possible path towards this goal is question answering, e.g. converting natural language queries to SPARQL [150,85]. In order to allow users to interact with such systems, there is ongoing work on converting the created SPARQL queries back to natural language [113] and employ feedback mechanisms [84,59]. Ultimately, a goal is to provide users enhanced functionality without the need to adapt to different kinds of interface.

Automatic Management of Resources for Linking. With the growth of the Cloud and of the datasets that need to be interlinked, the use of parallel hardware has been studied over the last few years [60,114]. The comparative study of parallel hardware for link discovery yet shows surprising results and suggests that the use of massively parallel yet local hardware can lead to tremendous runtime improvements. Still, when result sets go beyond sizes of 10^{10} , the higher amount of resources available on remote devices in the Cloud is still to be used. Devising automatic solutions for selecting the right hardware to run a linking task is one of the most interesting research areas pertaining

to the efficient execution of link specifications. In addition, developing reduction-ratio optimal algorithms for spaces other than Minkowski spaces promises to ensure the best possible use of available hardware. Finally, devising more efficient means to combine single algorithms is the third open area of research in this domain. The challenges faces with regard to learning link specifications are also manifold and include devising approaches that can efficiently detected most informative positive and negative examples as well even running in a fully unsupervised manner on properties that are not one-to-one relations.

Linked Data Visualization. The potential of the vast amount of Linked Data on the Web is enormous but in most cases it is very difficult *and* cumbersome for users to visualize, explore and use this data, especially for lay-users [37] without experience with Semantic Web technologies. Visualizations are useful for obtaining an overview of the datasets, their main types, properties and the relationships between them. Compared to prior information visualization strategies, we have a unique opportunity on the Data Web. The unified RDF data model being prevalent on the Data Web enables us to bind data to visualizations in an *unforeseen* and *dynamic* way. An information visualization technique requires certain data structures to be present. When we can derive and generate these data structures automatically from reused vocabularies or semantic representations, we are able to realize a largely automatic visualization workflow. Ultimately, various visualizations techniques can develop an ecosystem of data extractions and visualizations, which can be bound together in a dynamic and unforeseen way. This will enable users to explore datasets even if the publisher of the data does not provide any exploration or visualization means. Yet, most existing work related to visualizing RDF is focused on concrete domains and concrete datatypes.

Acknowledgments. We would like to thank our colleagues from the AKSW research group in Leipzig as well as the LOD2 and GeoKnow project consortia, without whom writing this chapter would not have been possible. In particular, we would like to thank Christian Bizer and Tom Heath, whose Chapter 2 of the book ‘Linked Data – Evolving the Web into a Global Data Space’ [54] served as a blueprint for Section 2; Sebastian Hellmann, Claus Stadler, Jörg Unbehauen for their contributions to Section 3, Sebastian Tramp, Michael Martin, Norman Heino, Phillip Frischmuth and Thomas Riechert for their contributions to the development of OntoWiki as described in Section 4. This work was supported by a grant from the European Union’s 7th Framework Programme provided for the projects LOD2 (GA no. 257943), GeoKnow (GA no. 318159) and the Eureka project SCMS.

References

1. Resource description framework (RDF): Concepts and abstract syntax. Technical report, W3C 2 (2004)
2. Adida, B., Birbeck, M., McCarron, S., Pemberton, S.: RDFa in XHTML: Syntax and processing – a collection of attributes and processing rules for extending XHTML to support RDF. W3C Recommendation (October 2008), <http://www.w3.org/TR/rdfa-syntax/>

3. Agichtein, E., Gravano, L.: Snowball: Extracting relations from large plain-text collections. In: ACM DL, pp. 85–94 (2000)
4. Agresti, A.: An Introduction to Categorical Data Analysis, 2nd edn. Wiley-Interscience (1997)
5. Amsler, R.: Research towards the development of a lexical knowledge base for natural language processing. SIGIR Forum 23, 1–2 (1989)
6. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
7. Auer, S., et al.: Managing the life-cycle of linked data with the lod2 stack. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part II. LNCS, vol. 7650, pp. 1–16. Springer, Heidelberg (2012)
8. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: Triplify: Light-weight linked data publication from relational databases. In: Quemada, J., León, G., Maarek, Y.S., Nejdl, W. (eds.) Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20–24, pp. 621–630. ACM (2009)
9. Auer, S., Dietzold, S., Riechert, T.: OntoWiki – A Tool for Social, Semantic Collaboration. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 736–749. Springer, Heidelberg (2006)
10. Auer, S., Herre, H.: A versioning and evolution framework for RDF knowledge bases. In: Virbitskaite, I., Voronkov, A. (eds.) PSI 2006. LNCS, vol. 4378, pp. 55–69. Springer, Heidelberg (2007)
11. Auer, S., Lehmann, J.: Making the web a data washing machine - creating knowledge out of interlinked data. Semantic Web Journal (2010)
12. Auer, S., Lehmann, J., Hellmann, S.: LinkedGeoData - adding a spatial dimension to the web of data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 731–746. Springer, Heidelberg (2009)
13. Auer, S., Lehmann, J., Ngonga Ngomo, A.-C.: Introduction to linked data and its lifecycle on the web. In: Polleres, A., d’Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 1–75. Springer, Heidelberg (2011)
14. Aumüller, D.: Semantic Authoring and Retrieval within a Wiki (WikSAR). In: Demo Session at the Second European Semantic Web Conference, ESWC 2005 (May 2005), <http://wiksar.sf.net>
15. Baader, F., Ganter, B., Sattler, U., Sertkaya, B.: Completing description logic knowledge bases using formal concept analysis. In: IJCAI 2007. AAAI Press (2007)
16. Baader, F., Sertkaya, B., Turhan, A.-Y.: Computing the least common subsumer w.r.t. a background terminology. J. Applied Logic 5(3), 392–420 (2007)
17. Badea, L., Nienhuys-Cheng, S.-H.: A refinement operator for description logics. In: Cussens, J., Frisch, A.M. (eds.) ILP 2000. LNCS (LNAI), vol. 1866, pp. 40–59. Springer, Heidelberg (2000)
18. Baxter, R., Christen, P., Churches, T.: A comparison of fast blocking methods for record linkage. In: KDD 2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation (2003)
19. Ben-David, D., Domany, T., Tarem, A.: Enterprise data classification using semantic web technologies. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 66–81. Springer, Heidelberg (2010)

20. Berners-Lee, T.: Notation 3 (1998), <http://www.w3.org/DesignIssues/Notation3.html>
21. Berners-Lee, T., Fielding, R.T., Masinter, L.: Uniform resource identifiers (URI): Generic syntax. Internet RFC 2396 (August 1998)
22. Bhagdev, R., Chapman, S., Ciravegna, F., Lanfranchi, V., Petrelli, D.: Hybrid search: Effectively combining keywords and semantic searches. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) *ESWC 2008*. LNCS, vol. 5021, pp. 554–568. Springer, Heidelberg (2008)
23. Bilenko, M., Kamath, B., Mooney, R.J.: Adaptive blocking: Learning to scale up record linkage. In: *ICDM 2006*, pp. 87–96. IEEE (2006)
24. Bizer, C., Cyganiak, R.: Quality-driven information filtering using the wiqa policy framework. *Web Semantics* 7(1), 1–10 (2009)
25. Bleiholder, J., Naumann, F.: Data fusion. *ACM Comput. Surv.* 41(1), 1–41 (2008)
26. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Occam’s razor. In: *Readings in Machine Learning*, pp. 201–204. Morgan Kaufmann (1990)
27. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C recommendation, W3C (February 2004), <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>
28. Brin, S.: Extracting patterns and relations from the world wide web. In: Atzeni, P., Mendelson, A.O., Mecca, G. (eds.) *WebDB 1998*. LNCS, vol. 1590, pp. 172–183. Springer, Heidelberg (1999)
29. Bühmann, L., Lehmann, J.: Universal OWL axiom enrichment for large knowledge bases. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d’Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) *EKAW 2012*. LNCS, vol. 7603, pp. 57–71. Springer, Heidelberg (2012)
30. Cherix, D., Hellmann, S., Lehmann, J.: Improving the performance of a sparql component for semantic web applications. In: *JIST* (2012)
31. Choi, N., Song, I.-Y., Han, H.: A survey on ontology mapping. *SIGMOD Record* 35(3), 34–41 (2006)
32. Coates-Stephens, S.: The analysis and acquisition of proper names for the understanding of free text. *Computers and the Humanities* 26, 441–456 (1992), doi:10.1007/BF00136985
33. Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: *AAAI 1992*, pp. 754–760 (1992)
34. Cohen, W.W., Hirsh, H.: Learning the CLASSIC description logic: Theoretical and experimental results. In: *KR 1994*, pp. 121–133. Morgan Kaufmann (1994)
35. Curran, J.R., Clark, S.: Language independent ner using a maximum entropy tagger. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, vol. 4, pp. 164–167. Association for Computational Linguistics, Morristown (2003)
36. d’Amato, C., Fanizzi, N., Esposito, F.: A note on the evaluation of inductive concept classification procedures. In: Gangemi, A., Keizer, J., Presutti, V., Stoermer, H. (eds.) *SWAP 2008*. CEUR Workshop Proceedings, vol. 426, CEUR-WS.org (2008)
37. Dadzie, A.-S., Rowe, M.: Approaches to visualising Linked Data. *Semantic Web* 2(2), 89–124 (2011)
38. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering* 19, 1–16 (2007)
39. Ermilov, T., Heino, N., Tramp, S., Auer, S.: OntoWiki Mobile – Knowledge Management in your Pocket. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) *ESWC 2011, Part I*. LNCS, vol. 6643, pp. 185–199. Springer, Heidelberg (2011)

40. Esposito, F., Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Knowledge-intensive induction of terminologies from metadata. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 441–455. Springer, Heidelberg (2004)
41. Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.* 165, 91–134 (2005)
42. Euzenat, J., Shvaiko, P.: *Ontology matching*. Springer, Heidelberg (2007)
43. Fanizzi, N., d’Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: Železný, F., Lavrač, N. (eds.) ILP 2008. LNCS (LNAI), vol. 5194, pp. 107–121. Springer, Heidelberg (2008)
44. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext transfer protocol – http/1.1 (rfc 2616). Request For Comments (1999) <http://www.ietf.org/rfc/rfc2616.txt> (accessed July 7, 2006)
45. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL 2005, pp. 363–370. Association for Computational Linguistics, Morristown (2005)
46. Fleischhacker, D., Völker, J., Stuckenschmidt, H.: Mining RDF data for property axioms. In: Meersman, R., et al. (eds.) OTM 2012, Part II. LNCS, vol. 7566, pp. 718–735. Springer, Heidelberg (2012)
47. Flemming, A.: Quality characteristics of linked data publishing datasources. Master’s thesis, Humboldt-Universität zu Berlin (2010)
48. Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C., Nevill-Manning, C.G.: Domain-specific keyphrase extraction. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 1999, pp. 668–673. Morgan Kaufmann Publishers Inc., San Francisco (1999)
49. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. *Science* 315, 972–976 (2007)
50. Glaser, H., Millard, I.C., Sung, W.-K., Lee, S., Kim, P., You, B.-J.: Research on linked data and co-reference resolution. Technical report, University of Southampton (2009)
51. Grishman, R., Yangarber, R.: Nyu: Description of the Proteus/Pet system as used for MUC-7 ST. In: MUC-7. Morgan Kaufmann (1998)
52. Guéret, C., Groth, P., Stadler, C., Lehmann, J.: Assessing linked data mappings using network measures. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 87–102. Springer, Heidelberg (2012)
53. Harabagiu, S., Bejan, C.A., Morarescu, P.: Shallow semantics for relation extraction. In: IJCAI, pp. 1061–1066 (2005)
54. Heath, T., Bizer, C.: *Linked Data - Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology, vol. 1. Morgan & Claypool (2011)
55. Heino, N., Dietzold, S., Martin, M., Auer, S.: Developing semantic web applications with the ontowiki framework. In: Pellegrini, T., Auer, S., Tochtermann, K., Schaffert, S. (eds.) *Networked Knowledge - Networked Media*. SCI, vol. 221, pp. 61–77. Springer, Heidelberg (2009)
56. Hellmann, S., Lehmann, J., Auer, S.: Learning of OWL class descriptions on very large knowledge bases. *Int. J. Semantic Web Inf. Syst.* 5(2), 25–48 (2009)
57. Hellmann, S., Lehmann, J., Auer, S.: Learning of owl class expressions on very large knowledge bases and its applications. In: *Interoperability Semantic Services and Web Applications: Emerging Concepts* (ed.) *Learning of OWL Class Expressions on Very Large Knowledge Bases and its Applications*, ch. 5, pp. 104–130. IGI Global (2011)

58. Hellmann, S., Lehmann, J., Auer, S.: Linked-data aware URI schemes for referencing text fragments. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 175–184. Springer, Heidelberg (2012)
59. Hellmann, S., Lehmann, J., Unbehauen, J., Stadler, C., Lam, T.N., Strohmaier, M.: Navigation-induced knowledge engineering by example. In: Takeda, H., Qu, Y., Mizoguchi, R., Kitamura, Y. (eds.) JIST 2012. LNCS, vol. 7774, pp. 207–222. Springer, Heidelberg (2013)
60. Hillner, S., Ngonga Ngomo, A.-C.: Parallelizing limes for large-scale link discovery. In: I Semantics (2011)
61. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the pedantic web. In: LDOW (2010)
62. Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., Decker, S.: An empirical survey of linked data conformance. *Journal of Web Semantics* (2012)
63. Horridge, M., Patel-Schneider, P.F.: Manchester syntax for OWL 1.1. In: OWLED 2008 (2008)
64. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosz, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. Technical report, W3C (May 2004)
65. HTML 5: A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft (August 2009) <http://www.w3.org/TR/2009/WD-html5-20090825/>
66. Iannone, L., Palmisano, I.: An algorithm based on counterfactuals for concept learning in the semantic web. In: Ali, M., Esposito, F. (eds.) IEA/AIE 2005. LNCS (LNAI), vol. 3533, pp. 370–379. Springer, Heidelberg (2005)
67. Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence* 26(2), 139–159 (2007)
68. Inan, A., Kantarcioglu, M., Bertino, E., Scannapieco, M.: A hybrid approach to private record linkage. In: ICDE, pp. 496–505 (2008)
69. Isele, R., Jentzsch, A., Bizer, C.: Efficient multidimensional blocking for link discovery without losing recall. In: WebDB (2011)
70. Isele, R., Jentzsch, A., Bizer, C.: Active learning of expressive linkage rules for the web of data. In: Brambilla, M., Tokuda, T., Tolksdorf, R. (eds.) ICWE 2012. LNCS, vol. 7387, pp. 411–418. Springer, Heidelberg (2012)
71. Jacobs, I., Walsh, N.: Architecture of the world wide web, volume one. World Wide Web Consortium, Recommendation REC-webarch-20041215 (December 2004)
72. Juran, J.: *The Quality Control Handbook*. McGraw-Hill, New York (1974)
73. Kifer, M., Boley, H.: Rif overview. Technical report, W3C (June 2010), <http://www.w3.org/TR/2012/NOTE-rif-overview-20121211/>
74. Kim, S.N., Kan, M.-Y.: Re-examining automatic keyphrase extraction approaches in scientific articles. In: Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications, MWE 2009, pp. 9–16. Association for Computational Linguistics, Stroudsburg (2009)
75. Kim, S.N., Medelyan, O., Kan, M.-Y., Baldwin, T.: Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In: Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval 2010, pp. 21–26. Association for Computational Linguistics, Stroudsburg (2010)
76. Köpcke, H., Thor, A., Rahm, E.: Comparative evaluation of entity resolution approaches with fever. *Proc. VLDB Endow.* 2(2), 1574–1577 (2009)
77. Krötzsch, M., Vrandečić, D., Völkel, M., Haller, H., Studer, R.: Semantic wikipedia. *Journal of Web Semantics* 5, 251–261 (2007)
78. Lehmann, J.: Hybrid learning of ontology classes. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 883–898. Springer, Heidelberg (2007)

79. Lehmann, J.: DL-Learner: learning concepts in description logics. *Journal of Machine Learning Research (JMLR)* 10, 2639–2642 (2009)
80. Lehmann, J.: Learning OWL Class Expressions. PhD thesis, University of Leipzig. PhD in Computer Science (2010)
81. Lehmann, J.: Ontology learning. In: *Proceedings of Reasoning Web Summer School* (2010)
82. Lehmann, J., Auer, S., Bühmann, L., Tramp, S.: Class expression learning for ontology engineering. *Journal of Web Semantics* 9, 71–81 (2011)
83. Lehmann, J., Bizer, C., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. *Journal of Web Semantics* 7(3), 154–165 (2009)
84. Lehmann, J., Bühmann, L.: AutoSPARQL: Let users query your knowledge base. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) *ESWC 2011, Part I. LNCS*, vol. 6643, pp. 63–79. Springer, Heidelberg (2011)
85. Lehmann, J., et al.: DEQA: Deep web extraction for question answering. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) *ISWC 2012, Part II. LNCS*, vol. 7650, pp. 131–147. Springer, Heidelberg (2012)
86. Lehmann, J., Hitzler, P.: Foundations of refinement operators for description logics. In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) *ILP 2007. LNCS (LNAI)*, vol. 4894, pp. 161–174. Springer, Heidelberg (2008)
87. Lehmann, J., Hitzler, P.: A refinement operator based learning algorithm for the \mathcal{ALC} description logic. In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) *ILP 2007. LNCS (LNAI)*, vol. 4894, pp. 147–160. Springer, Heidelberg (2008)
88. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. *Machine Learning Journal* 78(1-2), 203–250 (2010)
89. Pipino, D.K.L., Wang, R., Rybold, W.: *Developing Measurement Scales for Data-Quality Dimensions*, vol. 1. M.E. Sharpe, New York (2005)
90. Leuf, B., Cunningham, W.: *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley Professional (2001)
91. Lisi, F.A.: Building rules on top of ontologies for the semantic web with inductive logic programming. *Theory and Practice of Logic Programming* 8(3), 271–300 (2008)
92. Lisi, F.A., Esposito, F.: Learning SHIQ+log rules for ontology evolution. In: *SWAP 2008. CEUR Workshop Proceedings*, vol. 426, CEUR-WS.org (2008)
93. Lohmann, S., Heim, P., Auer, S., Dietzold, S., Riechert, T.: Semantifying requirements engineering – the softwiki approach. In: *Proceedings of the 4th International Conference on Semantic Technologies (I-SEMANTICS 2008)*, pp. 182–185. J.UCS (2008)
94. Lopez, V., Uren, V., Sabou, M.R., Motta, E.: Cross ontology query answering on the semantic web: an initial evaluation. In: *K-CAP 2009*, pp. 17–24. ACM, New York (2009)
95. Ma, L., Sun, X., Cao, F., Wang, C., Wang, X., Kanellos, N., Wolfson, D., Pan, Y.: Semantic enhancement for enterprise data management. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) *ISWC 2009. LNCS*, vol. 5823, pp. 876–892. Springer, Heidelberg (2009)
96. Martin, M., Stadler, C., Frischmuth, P., Lehmann, J.: Increasing the financial transparency of european commission project funding. *Semantic Web Journal, Special Call for Linked Dataset descriptions* (2013)
97. Matsuo, Y., Ishizuka, M.: Keyword Extraction From A Single Document Using Word Co-Occurrence Statistical Information. *International Journal on Artificial Intelligence Tools* 13(1), 157–169 (2004)
98. McBride, B., Beckett, D.: Rdf/xml syntax specification. W3C Recommendation (February 2004)

99. McCusker, J., McGuinness, D.: Towards identity in linked data. In: Proceedings of OWL Experiences and Directions Seventh Annual Workshop (2010)
100. Mendes, P., Mühleisen, H., Bizer, C.: Sieve: Linked data quality assessment and fusion. In: LWDM (March 2012)
101. Moats, R.: Urn syntax. Internet RFC 2141 (May 1997)
102. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.-C.: DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 454–469. Springer, Heidelberg (2011)
103. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.-C.: Usage-Centric Benchmarking of RDF Triple Stores. In: Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012) (2012)
104. Morsey, M., Lehmann, J., Auer, S., Stadler, C., Hellmann, S.: DBpedia and the Live Extraction of Structured Data from Wikipedia. Program: Electronic Library and Information Systems 46, 27 (2012)
105. Nadeau, D.: Semi-Supervised Named Entity Recognition: Learning to Recognize 100 Entity Types with Little Supervision. PhD thesis, University of Ottawa (2007)
106. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. *Linguisticae Investigationes* 30(1), 3–26 (2007)
107. Nadeau, D., Turney, P., Matwin, S.: Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity, pp. 266–277 (2006)
108. Ngonga Ngomo, A.-C.: Parameter-free clustering of protein-protein interaction graphs. In: Proceedings of Symposium on Machine Learning in Systems Biology 2010 (2010)
109. Ngonga Ngomo, A.-C.: A time-efficient hybrid approach to link discovery. In: Proceedings of OM@ISWC (2011)
110. Ngonga Ngomo, A.-C.: Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 378–393. Springer, Heidelberg (2012)
111. Ngonga Ngomo, A.-C.: On link discovery using a hybrid approach. *Journal on Data Semantics* 1, 203–217 (2012)
112. Ngonga Ngomo, A.-C., Auer, S.: Limes - a time-efficient approach for large-scale link discovery on the web of data. In: Proceedings of IJCAI (2011)
113. Ngonga Ngomo, A.-C., Bühmann, L., Unger, C., Lehmann, J., Gerber, D.: Sorry, i don't speak sparql – translating sparql queries into natural language. In: Proceedings of WWW (2013)
114. Ngonga Ngomo, A.-C., Kolb, L., Heino, N., Hartung, M., Auer, S., Rahm, E.: When to reach for the cloud: Using parallel hardware for link discovery. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 275–289. Springer, Heidelberg (2013)
115. Ngonga Ngomo, A.-C., Lehmann, J., Auer, S., Höffner, K.: Raven – active learning of link specifications. In: Proceedings of OM@ISWC (2011)
116. Ngonga Ngomo, A.-C., Lyko, K.: EAGLE: Efficient active learning of link specifications using genetic programming. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 149–163. Springer, Heidelberg (2012)
117. Ngonga Ngomo, A.-C., Lyko, K., Christen, V.: COALA – correlation-aware active learning of link specifications. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 442–456. Springer, Heidelberg (2013)
118. Ngonga Ngomo, A.-C., Schumacher, F.: Border flow – a local graph clustering algorithm for natural language processing. In: Proceedings of the 10th International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2009), pp. 547–558. Best Presentation Award (2009)

119. Nguyen, D.P.T., Matsuo, Y., Ishizuka, M.: Relation extraction from wikipedia using subtree mining. In: AAAI, pp. 1414–1420 (2007)
120. Nguyen, T., Kan, M.-Y.: Keyphrase Extraction in Scientific Publications, pp. 317–326 (2007)
121. Nienhuys-Cheng, S.-H., de Wolf, R.: Foundations of Inductive Logic Programming. LNCS, vol. 1228. Springer, Heidelberg (1997)
122. Oren, E.: SemperWiki: A Semantic Personal Wiki. In: Decker, S., Park, J., Quan, D., Sauerermann, L. (eds.) Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6, vol. 175 (2005)
123. Pantel, P., Pennacchiotti, M.: Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In: ACL, pp. 113–120. ACL Press (2006)
124. Park, Y., Byrd, R.J., Boguraev, B.K.: Automatic glossary extraction: beyond terminology identification. In: Proceedings of the 19th International Conference on Computational Linguistics, COLING 2002, vol. 1, pp. 1–7. Association for Computational Linguistics, Stroudsburg (2002)
125. Pasca, M., Lin, D., Bigham, J., Lifchits, A., Jain, A.: Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In: Proceedings of the 21st National Conference on Artificial Intelligence, vol. 2, pp. 1400–1405. AAAI Press (2006)
126. Patel-Schneider, P.F., Hayes, P., Horrocks, I.: OWL Web Ontology Language - Semantics and Abstract Syntax. W3c:rec, W3C (February 10, 2004), <http://www.w3.org/TR/owl-semantics/>
127. Rahm, E.: Schema Matching and Mapping. Springer, Heidelberg (2011)
128. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal 10, 334–350 (2001)
129. Raimond, Y., Sutton, C., Sandler, M.: Automatic interlinking of music datasets on the semantic web. In: 1st Workshop about Linked Data on the Web (2008)
130. Riechert, T., Lauenroth, K., Lehmann, J., Auer, S.: Towards semantic based requirements engineering. In: Proceedings of the 7th International Conference on Knowledge Management (I-KNOW) (2007)
131. Riechert, T., Morgenstern, U., Auer, S., Tramp, S., Martin, M.: Knowledge engineering for historians on the example of the catalogus professorum lipsiensis. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 225–240. Springer, Heidelberg (2010)
132. Rieß, C., Heino, N., Tramp, S., Auer, S.: EvoPat – Pattern-Based Evolution and Refactoring of RDF Knowledge Bases. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 647–662. Springer, Heidelberg (2010)
133. Rudolph, S.: Exploring relational structures via FLE. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 196–212. Springer, Heidelberg (2004)
134. Rula, A., Palmonari, M., Harth, A., Stadtmüller, S., Maurino, A.: On the diversity and availability of temporal information in linked open data. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 492–507. Springer, Heidelberg (2012)
135. Sahoo, S.S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr., T., Auer, S., Sequeda, J., Ez-zat, A.: A survey of current approaches for mapping of relational databases to rdf (January 2009)
136. Sampson, G.: How fully does a machine-usable dictionary cover english text. Literary and Linguistic Computing 4(1) (1989)
137. Sauerermann, L., Cyganiak, R.: Cool uris for the semantic web. W3C Interest Group Note (December 2008)

138. Schaffert, S.: Ikwiki: A semantic wiki for collaborative knowledge management. In: Proceedings of the 1st International Workshop on Semantic Technologies in Collaborative Applications (STICA) (2006)
139. Scharffé, F., Liu, Y., Zhou, C.: Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In: Proc. IJCAI 2009 IR-KR Workshop (2009)
140. Sertkaya, B.: OntocomP system description. In: Grau, B.C., Horrocks, I., Motik, B., Sattler, U. (eds.) Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30. CEUR Workshop Proceedings, vol. 477, CEUR-WS.org (2009)
141. Settles, B.: Active Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers (2012)
142. Shvaiko, P., Euzenat, J.: Ten challenges for ontology matching. Technical report (August 01, 2008)
143. Souzis, A.: Building a Semantic Wiki. IEEE Intelligent Systems 20(5), 87–91 (2005)
144. Spanos, D.-E., Stavrou, P., Mitrou, N.: Bringing relational databases into the semantic web: A survey. Semantic Web 3(2), 169–209 (2012)
145. Stadler, C., Lehmann, J., Höffner, K., Auer, S.: Linkedgeodata: A core for a web of spatial open data. Semantic Web Journal 3(4), 333–354 (2012)
146. Thielen, C.: An approach to proper name tagging for german. In: Proceedings of the EACL 1995 SIGDAT Workshop (1995)
147. Tramp, S., Frischmuth, P., Ermilov, T., Auer, S.: Weaving a Social Data Web with Semantic Pingback. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS (LNAI), vol. 6317, pp. 135–149. Springer, Heidelberg (2010)
148. Tramp, S., Heino, N., Auer, S., Frischmuth, P.: RDFauthor: Employing RDFa for collaborative Knowledge Engineering. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS (LNAI), vol. 6317, pp. 90–104. Springer, Heidelberg (2010)
149. Turney, P.D.: Coherent keyphrase extraction via web mining. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, pp. 434–439. Morgan Kaufmann Publishers Inc., San Francisco (2003)
150. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D., Cimiano, P.: Sparql template-based question answering. In: Proceedings of WWW (2012)
151. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: OWL reasoning with webPIE: Calculating the closure of 100 billion triples. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 213–227. Springer, Heidelberg (2010)
152. Verlic, M.: Lodgrefine - lod-enabled google refine in action. In: I-SEMANTICS 2012 Posters and Demonstrations Track, pp. 31–27 (2012)
153. Völker, J., Niepert, M.: Statistical schema induction. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 124–138. Springer, Heidelberg (2011)
154. Völker, J., Rudolph, S.: Fostering web intelligence by semi-automatic OWL ontology refinement. In: Web Intelligence, pp. 454–460. IEEE (2008)
155. Völker, J., Vrandečić, D., Sure, Y., Hotho, A.: Learning disjointness. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 175–189. Springer, Heidelberg (2007)
156. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Discovering and maintaining links on the web of data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 650–665. Springer, Heidelberg (2009)
157. Walker, D., Amsler, R.: The use of machine-readable dictionaries in sublanguage analysis. Analysing Language in Restricted Domains (1986)

158. Wang, G., Yu, Y., Zhu, H.: PORE: Positive-only relation extraction from wikipedia text. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 580–594. Springer, Heidelberg (2007)
159. Wang, R.Y., Strong, D.M.: Beyond accuracy: what data quality means to data consumers. *Journal of Management Information Systems* 12(4), 5–33 (1996)
160. Watanabe, H., Muggleton, S.: Can ILP be applied to large datasets? In: De Raedt, L. (ed.) ILP 2009. LNCS, vol. 5989, pp. 249–256. Springer, Heidelberg (2010)
161. Winkler, W.: The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Bureau of the Census (1999)
162. Winkler, W.: Overview of record linkage and current research directions. Technical report, Bureau of the Census - Research Report Series (2006)
163. Wu, H., Zubair, M., Maly, K.: Harvesting social knowledge from folksonomies. In: Proceedings of the Seventeenth Conference on Hypertext and Hypermedia, HYPERTEXT 2006, pp. 111–114. ACM, New York (2006)
164. Yan, Y., Okazaki, N., Matsuo, Y., Yang, Z., Ishizuka, M.: Unsupervised relation extraction by mining wikipedia texts using information from the web. In: *ACL 2009*, pp. 1021–1029 (2009)
165. Zaveri, A., Lehmann, J., Auer, S., Hassan, M.M., Sherif, M.A., Martin, M.: Publishing and interlinking the global health observatory dataset. *Semantic Web Journal, Special Call for Linked Dataset descriptions* (2013)
166. Zaveri, A., Pietrobon, R., Auer, S., Lehmann, J., Martin, M., Ermilov, T.: Redd-observatory: Using the web of data for evaluating the research-disease disparity. In: *Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence* (2011)
167. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S.: Quality assessment methodologies for linked open data. Under review, <http://www.semantic-web-journal.net/content/quality-assessment-methodologies-linked-open-data>
168. Zhou, G., Su, J.: Named entity recognition using an hmm-based chunk tagger. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL 2002*, pp. 473–480. Association for Computational Linguistics, Morristown (2002)

RDFS and OWL Reasoning for Linked Data

Axel Polleres¹, Aidan Hogan², Renaud Delbru², and Jürgen Umbrich^{2,3}

¹ Siemens AG Österreich, Siemensstraße 90, 1210 Vienna, Austria

² Digital Enterprise Research Institute, National University of Ireland, Galway

³ Fujitsu (Ireland) Limited, Swords, Co. Dublin, Ireland

Abstract. Linked Data promises that a large portion of Web Data will be usable as one big interlinked RDF database against which structured queries can be answered. In this lecture we will show how reasoning – using RDF Schema (RDFS) and the Web Ontology Language (OWL) – can help to obtain more complete answers for such queries over Linked Data. We first look at the extent to which RDFS and OWL features are being adopted on the Web. We then introduce two high-level architectures for query answering over Linked Data and outline how these can be enriched by (lightweight) RDFS and OWL reasoning, enumerating the main challenges faced and discussing reasoning methods that make practical and theoretical trade-offs to address these challenges. In the end, we also ask whether or not RDFS and OWL are enough and discuss numeric reasoning methods that are beyond the scope of these standards but that are often important when integrating Linked Data from several, heterogeneous sources.

1 Introduction

Linked Data [36,2] denotes the emerging trend of publishing structured data on the Web as interlinked RDF graphs [49] following a number of simple principles: use HTTP URIs to name things, return RDF about those things when their URIs are looked up, and include links to related RDF documents elsewhere on the Web. These trends – and related trends with respect to embedding metadata into HTML pages (such as promoted by `schema.org`) – have led to a huge volume of RDF data being made openly available online, contributing to what is often called the “Web of Data”.

The Semantic Web provides the necessary tools to query this data, (i) firstly by defining RDF [49,35] as a universal data format; (ii) secondly by defining SPARQL [60,30], a standard query language for RDF; and (iii) lastly by providing schema languages such as RDF Schema (RDFS) [13] and OWL [37], which allow for adding rich semantics to the data.

It is the aim of this lecture to emphasise that all three components, and in particular the reasoning capabilities enabled by RDFS and OWL, are essential to enable usage of the Web of Data as “*one huge database*” as originally envisioned by Tim Berners-Lee [9].

As opposed to standard reasoning and query answering in OWL, for instance as described in the present volume in the chapter on Ontology-based Data Access [47], reasoning over Linked Data poses several unique challenges:

- C1 *Linked Data is huge*, that is, it needs highly scalable or modular reasoning techniques;
- C2 *Linked Data is not “pure” OWL*, that is, a lot of RDF Data published as Linked Data violates the strict syntactic corset of OWL (2) DL, and thus is not directly interpretable under OWL Direct Semantics;
- C3 *Linked Data is inconsistent*, that is, if you take the Web of Data in its entirety, it is quite normal to encounter inconsistencies – not only from accidental or malicious datasets – but also because publishers may express contradicting views;
- C4 *Linked Data is evolving*, that is, RDF Graphs on the Web evolve, they change, information is added and removed;
- C5 *Linked Data needs more than RDFS and OWL*, that is, there is more implicit data hidden in Linked Data than can be captured with the semantics of RDFS and OWL alone.

In this lecture, we will introduce and discuss robust and scalable reasoning techniques that are specifically tailored to deal with these challenges in diverse and large-scale Linked Data settings. We first recapitulate the basic concepts of RDF, Linked Data, RDFS, OWL and SPARQL, and, with reference to a practical real-world scenario, we exemplify the use of query-rewriting techniques and rule-based approaches for reasoning over Linked Data (Section 2). We then will reflect in more detail on challenges C1–C5 above and discuss basic architectures, namely data-warehousing and on-the-fly-traversal based approaches, to reason about and query over Linked Data (Section 3).

While the W3C has defined two standard OWL fragments tailored for both rule-based (OWL 2 RL) and query-rewriting (OWL 2 QL) techniques, as we will see, standard reasoning within these fragments may not be a perfect fit for the Linked Data use-case; we will thus discuss which OWL features are actually predominantly used in current Linked Data and how these features relate to various standard and non-standard OWL fragments (Section 4).

The remaining chapters will then introduce specific approaches to Linked Data reasoning, each of which addresses some of the above-named challenges:

Section 5 introduces two rule-based approaches for Linked Data warehouses, both of which apply a cautious form of materialisation that considers where the axioms in question were found on the Web:

- *Context-Dependent Reasoning* [17], which is deployed within the Sindice semantic web engine [54].
- *Authoritative Reasoning* [41,12], which is deployed within the Scalable Authoritative OWL Reasoner (SAOR) as part of the Semantic Web Search Engine (SWSE) [40].

Section 6 presents an alternative approach to enrich on-the-fly-traversal based approaches for Linked Data query processing with lightweight OWL Reasoning [67].

Section 7 presents a technique for reasoning with “attribute equations” [10] that models interdependencies between numerical properties expressible in terms of simple mathematical equations, which we argue is complementary

to RDFS/OWL techniques in that it allows for integrating numerical information embedded in Linked Data in a manner that RDFS and OWL do not.

We then briefly recap and conclude the lecture material in Section 8.

2 Preliminaries

As mentioned in the introduction, more and more Web sources provide readily accessible and interlinked RDF data. In the context of this paper, when speaking about structured data on the Web, we focus on data represented and published in the Resource Description Framework (RDF) according to the Linked Data principles. Hundreds of such datasets have been published in this manner, and have been collectively illustrated by Cyganiak and Jentzsch in the so-called “Linked Data Cloud”.¹ Among these DBpedia² plays a central role as an RDF extract of structured data from Wikipedia. As a further example, the New York Times³ provide RDF metadata about entities occurring in their articles as well as API access to the latest articles about these entities.

In terms of reasoning, our main considerations revolve around deductive inferences that enrich this RDF data with implicit information by means of exploiting (parts of) the formal semantics of RDFS and OWL. In particular, we aim at motivating how such additional inferences can contribute to query answering using SPARQL queries over the whole body of Web-accessible Linked Data (per Berners-Lee’s “one big database” vision [9]).

We begin by introducing the basic concepts of the relevant Web standards along with a running example from real-world Linked Data. As a sort of disclaimer upfront, we emphasise that these preliminaries are not intended to replace a fully-fledged introduction to RDF, SPARQL, OWL or Description Logics; we refer the interested reader to excellent introductory chapters published within the earlier editions of the Reasoning Web summer school or to the official current W3C standard specifications in the references for further details.

2.1 The Resource Description Framework – RDF

Informally, all RDF data can be understood as a set of subject–predicate–object triples, where all subjects and predicates are URIs, and in the object position both URIs and literal values (such as numbers, strings, etc.) are allowed. Furthermore, blank nodes can be used in the subject or object resource to denote an unnamed resource with local scope. Some sample RDF data in the popular Turtle syntax [5,6] are shown in Fig. 1.

¹ <http://lod-cloud.net>

² <http://dbpedia.org>

³ <http://data.nytimes.org>

More formally, 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 $C := UBL$.⁴ A triple $t := (s, p, o) \in UB \times U \times C$ is called an *RDF triple*, where s is called subject, p predicate, and o object. A triple $t := (s, p, o) \in Tr$, $Tr := C \times C \times C$ is called a *generalised triple* [25], 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 $G \subset Tr$ a *graph*.⁵

In the Turtle syntax, which we use for examples (see Fig. 1), URIs are often denoted either in full form using strings delimited by angle brackets (e.g. `<http://dbpedia.org/resource/Werner_von_Siemens>`) or as shorthand prefixed names (e.g. `dbr:Werner_von_Siemens`), where prefixes are as defined in Fig. 1. Literals are delimited by double quotes (e.g. `"SAP AG"`) with an optional trailing language tag (e.g. `@en`, `@de`) or a datatype which itself is again identified by a URI (e.g. `8sd:dateTime`). Blank nodes are (typically) scoped to a local document [35,48] and are denoted in Turtle either by the “prefix” `_:` or alternatively using square brackets `[]`. Furthermore, Turtle allows use of ‘`a`’ as a shortcut for `rdf:type` (denoting class membership) [5]. Finally, as shown in Fig. 1, RDF triples in Turtle notation are delimited by a trailing ‘`.`’, where predicate-object-pairs that share the same subject can be grouped using ‘`;`’ and several objects for the same subject–predicate pair can be grouped using ‘`,`’.

The data provided in Fig. 1 reflects real-world Linked Data available on the Web at the time of writing. However, we cheat slightly: in reality, revenue is not given in DBpedia using a dedicated property that includes the currency name (like `dbo:revenueUSD` or `dbo:revenueEUR`), but instead, e.g., the revenue data for IBM mentioned in Fig. 1(b) rather appears as follows:

```
dbr:IBM  dbp:revenue "US$ 106.916 billion";
         dbo:revenue "1.06916E11"^^dbdt:usDollar .
```

The use of different units (currencies in this case) and the fact that these units are often ill-defined for current Linked Data is a separate problem that we discuss later in Section 7.

2.2 Linked Data Principles and Provenance

In order to cope with the unique challenges of handling diverse and unverified RDF data spread over RDF graphs published at different URIs across the Web, many algorithms require inclusion of a notion of provenance, i.e., the consideration of the source of RDF data found on the Web. To this end, we provide some

⁴ Herein, we sometimes use the convention of concatenating set names to represent unions; e.g. $UBL = U \cup B \cup L$. Also, though blank nodes are not constants by standard, we consider them as such for convenience. For more details on blank nodes and Skolemisation, we refer the interested reader to [48]

⁵ Note that we use \subset instead of \subseteq here since we consider RDF graphs as *finite* subsets of the infinite set of triples Tr . We will also often use sans-serif (e.g., U) to denote an infinite set and use normal math font (e.g., U) to denote a finite proper subset.

<pre> @prefix nytimes: <http://data.nytimes.com/element/> . @prefix nyt: <http://data.nytimes.com/> . @prefix dbr: <http://dbpedia.org/resource/> . @prefix skos: <http://www.w3.org/2004/02/skos/core#> . @prefix xsd: <http://www.w3.org/2001/XMLSchema#> . nyt:75293219995342479362 owl:sameAs dbr:SAP_AG ; nytimes:associated_article_count 10 ; nytimes:first_use "2007-03-23"^^xsd:dateTime ; nytimes:latest_use "2010-05-13"^^xsd:dateTime ; skos:prefLabel "SAP AP"@en ; skos:inScheme nytimes:nytd_org ; rdf:type skos:Concept . nyt:N82918236209763785922 owl:sameAs dbr:Siemens ; nytimes:associated_article_count 4 ; nytimes:first_use "2008-12-21"^^xsd:dateTime ; nytimes:latest_use "2009-11-06"^^xsd:dateTime ; skos:inScheme nytimes:nytd_org ; skos:prefLabel "Siemens A.G"@en ; rdf:type skos:Concept . nyt:49586210195898795812 owl:sameAs dbr:IBM ; nytimes:associated_article_count 196 ; nytimes:first_use "2004-09-01"^^xsd:dateTime ; nytimes:latest_use "2010-04-27"^^xsd:dateTime ; skos:inScheme nytimes:nytd_org ; skos:prefLabel "International Business Machines Corporation"@en rdf:type skos:Concept . </pre>	<pre> @prefix dbr: <http://dbpedia.org/resource/> . @prefix dbp: <http://dbpedia.org/property/> . @prefix dbo: <http://dbpedia.org/ontology/> . @prefix foaf: <http://xmlns.com/foaf/0.1/> . dbr:SAP_AG dbo:revenueEUR 1.622E10 rdfs:label "SAP AG"@en, "SAP"@de ; foaf:name "SAP AG"@en ; dbo:foundedBy dbr:Claus_Wellenreuther, dbr:Hasso_Plattner , dbr:Dietmar_Hopp , dbr:Klaus_Tschira , dbr:Hans-Werner_Hector , dbr:Rajkumar_Asokan; rdf:type dbo:Company, dbo:Organisation, dbo:Agent, owl:Thing. dbr:Siemens dbo:revenueEUR 7.829E10 ; rdfs:label "Siemens AG"@en , "Siemens"@de ; foaf:name "Siemens AG"@en ; dbo:foundedBy dbr:Werner_von_Siemens ; rdf:type dbo:Company, dbo:Organisation, dbo:Agent, owl:Thing. dbr:IBM dbo:revenueUSD 1.06916E11 ; rdfs:label "IBM"@en, rdfs:label "IBM"@de ; foaf:name "International Business Machines Corporation"@en ; rdf:type dbo:Company, dbo:Organisation, dbo:Agent, owl:Thing; dbo:foundedBy dbr:Thomas_J._Watson ; </pre>
(a)	(b)

Fig. 1. Sample Data from DBpedia and NYT in Turtle Syntax (retrieved 10 March 2013)

formal preliminaries for the Linked Data principles, and HTTP mechanisms for retrieving RDF data from *dereferenceable URIs* (i.e., URIs that return content when looked up over HTTP).

Linked Data Principles Herein, we will refer to the four best practices of Linked Data publishing as follows [7]:

LDP1: use URIs as names for things;

LDP2: use HTTP URIs so those names can be dereferenced;

LDP3: return useful – herein we assume RDF – information upon dereferencing of those URIs; and

LDP4: include links using externally dereferenceable URIs.⁶

Data Source. We define the *http-download* function $\text{get} : U \rightarrow 2^{\text{Tr}}$ as the mapping from a URI to an RDF graph it provides by means of a given HTTP lookup [19] 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

⁶ That is, within your published RDF graph, use HTTP URIs pointing to other dereferenceable documents, that possibly contain further RDF graphs.

blank-node labels (based on the input URI) to ensure uniqueness when merging RDF graphs [35]. We define the set of *data sources* $S \subset U$ as the set of URIs $S := \{s \in U \mid \text{get}(s) \neq \emptyset\}$.⁷

RDF Triple in Context/RDF Quadruple. An ordered pair (t, c) with a triple $t := (s, p, o)$, and with a context $c \in S$ and $t \in \text{get}(c)$ 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 . In Fig. 2 we illustrate the RDF data from Fig. 1 as dereferenceable RDF graphs. Note that Fig. 2 contains more data than Fig. 1, where we include triples obtained from dereferencing the founders of organisations mentioned in Fig. 1, as well as data obtained from dereferencing class and property URIs.

HTTP Dereferencing/Redirects. We now give some notation relating to a “Linked Dataset”, which is inspired by the notion of named graphs in SPARQL, but where the dataset reflects the $\text{get}(\cdot)$ function that maps graph names to RDF graphs obtained from dereferencing URIs.

Definition 1 (Data Source and Linked Dataset)

We define a *Linked Dataset* as $\Gamma \subset \text{get}$; i.e., a finite set of pairs $(s, \text{get}(s))$ such that $s \in S$.⁸ The “global” RDF graph presented by a Linked Dataset is denoted as

$$\text{merge}(\Gamma) := \biguplus_{(u, G) \in \Gamma} G$$

where the operator ‘ \biguplus ’ denotes the RDF merge of RDF graphs: a set union where blank nodes are rewritten to ensure that no two input graphs contain the same blank node label [35]. (From the perspective of a SPARQL dataset, one may consider $\text{merge}(\Gamma)$ as the “default graph” and all such $(s, \text{get}(s))$ as named graph pairs.)

A URI may provide a HTTP redirect to another URI using a 30x response code [19]; we denote this function as $\text{redir} : U \rightarrow U$ which first removes any fragment identifier from the URI (a hash string suffix) and then follows a single redirect, mapping a URI to itself in the case of failure (e.g., where no redirect exists). We denote the fixpoint of redir as redirs , denoting traversal of a number of redirects (a limit may be set on this traversal to avoid cycles and artificially long redirect paths). We define *dereferencing* as the function $\text{deref} := \text{get} \circ \text{redirs}$ 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.

We denote the set of *dereferenceable URIs* as $D := \{d \in U \mid \text{deref}(d) \neq \emptyset\}$; note that $S \subset D$ and we place no expectations on what $\text{deref}(d)$ returns, other than returning some valid RDF.⁹ As a shortcut, we denote by $\text{derefs} : 2^U \rightarrow 2^{U \times 2^U}$;

⁷ Note again that we use \subset instead of \subseteq to emphasise that obviously not all URIs point to non-empty RDF graphs.

⁸ Γ is finite thus – again – we use \subset instead of \subseteq .

⁹ The URIs within D that are not in S are those that return (non-empty) RDF graphs by means of (an) intermediate redirect(s) and/or after having a fragment identifier stripped.

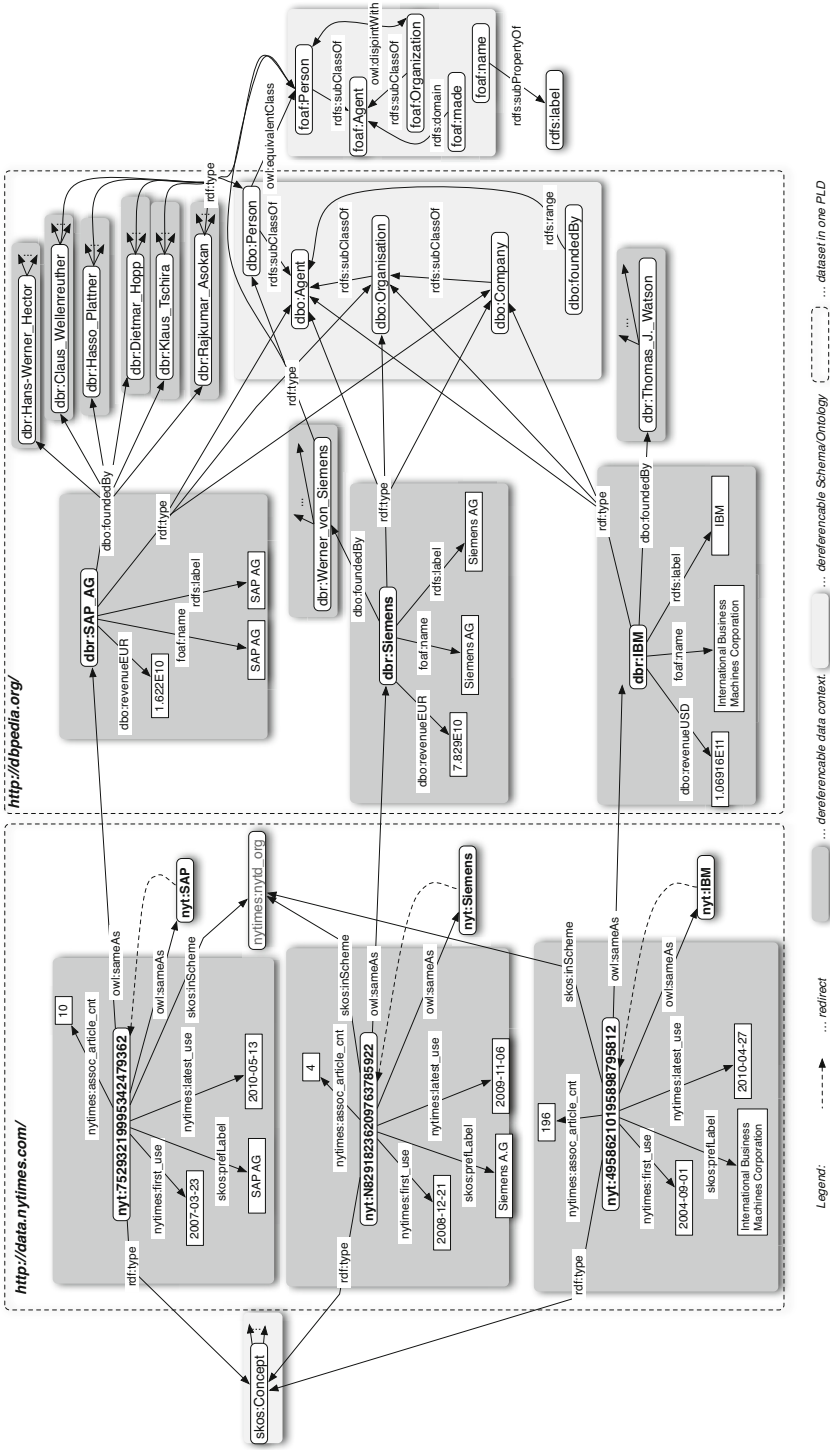


Fig. 2. Sample Data from DBpedia and NYT as Linked Data

$U \mapsto \{(\text{redirs}(u), \text{deref}(u)) \mid u \in U \cap D\}$ the mapping from a set of URIs to the Linked Dataset it represents by dereferencing all URIs (only including those in D which return some RDF).

Example 1

Taking Fig. 2, dashed arrows indicate redirects: NYT provides some “readable” URIs that redirect to its internal identifiers for entities, including $\text{redirs}(\text{nyt:Siemens}) = \text{nyt:N82918236209763785922}$. Dereferencing the URI nyt:Siemens yields the RDF graph for the source $\text{nyt:N82918236209763785922}$ (symbolised by the gray box in Fig. 2); in other words $\text{deref}(\text{nyt:Siemens}) = \text{get}(\text{redirs}(\text{nyt:Siemens})) = \text{get}(\text{nyt:N82918236209763785922})$. Conversely, we see $\text{deref}(\text{nytimes:nytd_org}) = \emptyset$; the URI is not dereferenceable. Thus we say that $\text{nyt:N82918236209763785922} \in D$ (and $\in S$) and $\text{nyt:Siemens} \in D$, whereas $\text{nytimes:nytd_org} \notin D$. \diamond

2.3 SPARQL, Conjunctive Queries and Rules

We now introduce some concepts relating to the query language SPARQL, where details can be found in [60,57,30]. We herein focus on evaluating simple, conjunctive, *basic graph patterns* (BGPs), we do not formally consider more complex patterns (e.g., UNION, FILTER, etc.) of the SPARQL language, which can be layered on top [57]. In addition, we consider URIs and not IRIs for convenience and to stay consistent with the RDF preliminaries.

Definition 2 (Variables, Triple Patterns & BGPs)

Let V be the set of variables ranging over UBL; we denote RDF variables as alphanumeric strings with a ‘?’ prefix (e.g. ?X). A triple pattern $tp := (s, p, o)$ is an element of the set $Q := VUL \times VU \times VUL$.¹⁰ For simplicity, we do not consider blank-nodes in triple patterns (they could be roughly emulated by an injective mapping from B to V). A finite (herein, non-empty) set of triple patterns $Q \subset Q$ is called a Basic Graph Pattern (BGP), or herein, simply a conjunctive query. We use $\text{vars}(Q) \subset V$ to denote the finite set of variables in Q and $\text{terms}(Q)$ to denote the set of all terms VUL in Q .

The solutions of a BGP for a dataset Γ are defined accordingly as follows.

Definition 3 (SPARQL solutions)

Call the partial function $\mu : \text{dom}(\mu) \cup UL \rightarrow UBL$ a solution mapping with a domain $\text{dom}(\mu) \subseteq V$. A solution mapping binds variables in $\text{dom}(\mu)$ to UBL and is the identify function for UL. Overloading notation, let $\mu : Q \rightarrow Tr$ and $\mu : 2^Q \rightarrow 2^{Tr}$ also resp. denote a solution mapping from triple patterns to RDF triples,

¹⁰ Though academic, SPARQL does allow for literals in the subject position.

and basic graph patterns to RDF graphs such that $\mu(tp) := (\mu(s), \mu(p), \mu(o))$ and $\mu(Q) := \{\mu(tp) \mid tp \in Q\}$. We now define the set of SPARQL solutions for a query Q over a (Linked) Dataset Γ as

$$\llbracket Q \rrbracket_{\Gamma} := \{\mu \mid \mu(Q) \subseteq \text{merge}(\Gamma) \wedge \text{dom}(\mu) = \text{vars}(Q)\}.$$

For brevity, and unlike SPARQL as according to the official W3C specification, solutions are herein given as sets (not multi-sets), implying a default *DISTINCT* semantics for queries, and we assume that answers are given over the default graph consisting of the merge of RDF graphs in the dataset.

Note that in the database literature, conjunctive queries are often rather modelled as conjunctions of (first-order) atoms, inspired by languages such as Datalog; let us introduce some basic concepts from Datalog to relate the notions of BGPs, conjunctive queries, and rules:

Definition 4 (Atom)

An atomic formula or atom is a formula of the form $p(e_1, \dots, e_n)$, where all such e_1, \dots, e_n are terms (i.e. in our case constants from \mathbf{C} or variables from \mathbf{V}) and where p is a predicate of arity n – we denote the set of all such atoms by **Atoms**. Atoms that do not contain variables are called *ground*. As such, this notation can be thought of as generalising that of RDF triples (or quadruples, resp.), where we use a standard RDF ternary predicate T to represent RDF triples in the form $T(s, p, o)$ – for example, $T(\text{Siemens}, \text{foundedby}, \text{Werner_von_Siemens})$ – where we will typically leave T implicit; analogously, we use the predicate $Q(s, p, o, c)$ synonymously for an RDF quadruple, again optionally leaving Q implicit.

In Logic Programming, atoms not containing variables are called *ground atoms*. The set of all possible ground atoms – denoted as the set **Facts** – can be viewed in our context as a generalisation of **Tr**. A finite set of ground atoms $I \subseteq \mathbf{Facts}$ (often simply called “*facts*”) – which in our context again can be viewed as a generalisation of a graph G – in Logic Programming is typically synonymous to a (Herbrand) *interpretation*.

Letting A and B be two atoms, we say that A *subsumes* B —denoted $A \triangleright B$ —if there exists a substitution $\theta \in \Theta$ of variables such that $A\theta = B$ (applying θ to the variables of A yields B); we may also say that B is an *instance* of A ; if B is ground, we say that it is a *ground instance*. Similarly, if we have a substitution $\theta \in \Theta$ such that $A\theta = B\theta$, we say that θ is a *unifier* of A and B ; we denote by $\text{mgu}(A, B)$ the *most general unifier* of A and B which provides the “minimal” variable substitution (up to variable renaming) required to unify A and B .

Definition 5 (Rule, Program). A rule R is given as follows:

$$H \leftarrow B_1, \dots, B_n (n \geq 0),$$

where H, B_1, \dots, B_n are atoms, H is called the head (conclusion/consequent) and B_1, \dots, B_n the body (premise/antecedent). We use $\text{Head}(R)$ to denote the head H of R and $\text{Body}(R)$ to denote the body B_1, \dots, B_n of R .¹¹ The variables of our rules are range restricted, also known as safe [66]: like Datalog, the variables appearing in the head of each rule must also appear in the body whereby a substitution that grounds the body must also ground the head. We denote the set of all such rules by **Rules**. A rule with an empty body can be 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 $\text{Ground}(R)$ the set of ground instantiations of a rule R and with $\text{Ground}(P)$ 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.

Definition 6 (Immediate Consequence Operator). We give the immediate consequence operator \mathfrak{T}_P of a program P under interpretation I as:¹²

$$\mathfrak{T}_P : 2^{\text{Facts}} \rightarrow 2^{\text{Facts}}$$

$$I \mapsto \left\{ \text{Head}(R)\theta \mid R \in P \wedge \exists I' \subseteq I \text{ s.t. } \theta = \text{mgu}(\text{Body}(R), I') \right\}$$

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

Since our rules are a syntactic subset of Datalog, \mathfrak{T}_P has a *least fixpoint*, denoted $\text{lfp}(\mathfrak{T}_P)$, which can be calculated in a bottom-up fashion, starting from the empty interpretation Δ and applying iteratively \mathfrak{T}_P [73] (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 $\text{lfp}(\mathfrak{T}_P) = \mathfrak{T}_P \uparrow \alpha$ for $\alpha < \omega$, where ω denotes the least infinite ordinal. In other words, the immediate consequence operator will reach a fixpoint in countable steps [66]. Thus, \mathfrak{T}_P is also *continuous*.

Definition 7 (least model, closure). We call $\text{lfp}(\mathfrak{T}_P)$ the *least model*, or the *closure* of P , which is given the more succinct notation $\text{lm}(P)$.

¹¹ Such a rule can be represented as a definite Horn clause.

¹² Recall again that in Herbrand semantics, an interpretation I can be thought of as simply a set of facts.

Obviously, using the notation of rules, any BGP can be straightforwardly seen as a conjunctive query (CQ) in the classical sense, and the more general concept of a union of conjunctive queries (UCQs) can be defined as follows.

Definition 8 (Union of Conjunctive Queries (classical notion)). A conjunctive query (CQ) is a special rule of the form

$$q(\vec{x}) \leftarrow \text{Body}(\vec{x}, \vec{y})$$

where \vec{x} is a sequence of variables called distinguished variables, \vec{y} is a sequence of variables called non-distinguished variables, and $\text{Body}(\vec{x}, \vec{y})$ is a conjunction of body atoms over these variables. A program P_q (where we just write q when P is implicit from the context) that consists of only rules with the same head $q(\vec{x})$ (such that q does not appear in any body) is a union of conjunctive queries (UCQ).

Given a (Linked) Dataset Γ and a SPARQL BGP query Q , the definition of solutions from Definition 3 thus corresponds to the notion of entailed answers for the corresponding classical conjunctive query q , written $\text{Ans}(q, \text{merge}(\Gamma))$ (and used in the literature, cf. for instance [14]).¹³ That is, in our context we can equate $\text{Ans}(q, \text{merge}(\Gamma))$ with the set of tuples \vec{a} such that $q(\vec{a}) \in \mathfrak{T}_{P_q}(\text{merge}(\Gamma))$. Note that as for SPARQL, all variables are considered to be distinguished, since there is no “real” projection in the sense of classical conjunctive queries: any BGP occurring inside a SPARQL query is evaluated as a conjunctive query without non-distinguished variables, whereupon the SPARQL algebra evaluates more complex patterns, such as SELECT clauses and so forth [30,21].

Example 2

The following SPARQL query asks for the labels and revenues (in EUR) of organisations:

```

QUERY 1
SELECT ?X ?L ?R
WHERE { ?X a dbo:Organisation ; rdfs:label ?L ; dbo:revenueEUR ?R .}

```

This query asks to evaluating the basic graph pattern:

$$\{(?X, \mathbf{a}, \text{dbo:Organisation}), (?X, \text{rdfs:label}, ?L), (?X, \text{dbo:revenueEUR}, ?R)\}$$

which, respectively, in the classical notation corresponds to the following conjunctive query:

$$q(?X, ?L, ?R) \leftarrow (?X, \mathbf{a}, \text{dbo:Organisation}), (?X, \text{rdfs:label}, ?L), (?X, \text{dbo:revenueEUR}, ?R).$$

¹³ We use $\text{merge}(\Gamma)$ here synonymously with the knowledge base consisting of the facts corresponding to the triples in $\text{merge}(\Gamma)$.

Given the data in Fig. 2, this pattern (and thus the query) would obtain the following solutions (writing solution mappings μ in tabular form):

?X	?L	?R
dbr:SAP_AG	"SAP AG"@en	1.622E10
dbr:Siemens	"Siemens"@de	7.829E10

Notably, the revenue for IBM is not returned (although it could be calculated from the EUR–USD exchange rate).

The next query, on the contrary, asks for the date of the latest article published about each element in the SKOS scheme `nytimes:nytd_org` (i.e., a different way of asking for “organisations”, but using the NYT RDF vocabulary):

QUERY 2

```
SELECT ?X ?D
WHERE { ?X skos:inScheme nytimes:nytd_org . ?X nytimes:latest_use ?D . }
```

with the following solutions:

?X	?D
nyt:75293219995342479362	2010-05-13
nyt:N82918236209763785922	2009-11-06
nyt:49586210195898795812	2010-04-27

Query 1 and Query 2 could each be answered by staying within a single site (that is, Query 1 only would obtain answers from data at `dbpedia.org`, whereas Query 2 would only produce answers with the data at `data.nytimes`, respectively) and – at least for our sample – answers for either query can be obtained from the individual dataset. However, the real power of Linked Data lies in combining data from multiple datasets to obtain answers over their combined content. For example Query 3 combines knowledge from both sites and asks for the latest NYT article dates of IBM (using its NYT identifier) and its revenue in USD:

QUERY 3

```
SELECT ?X ?D ?R
WHERE { nyt:49586210195898795812 nytimes:latest_use ?D .
        nyt:49586210195898795812 owl:sameAs ?X .
        ?X dbo:revenueUSD ?R . }
```

Again assuming the entire graph of Fig. 2 as input, this query would obtain the single result

?X	?D	?R
dbr:IBM	2010-04-27	1.06916E11

As a further example, let Query 4 ask for all `foaf:Agents`.

```

QUERY 4
SELECT ?X
WHERE { ?X a foaf:Agent .}

```

Clearly, for the RDF data in Fig. 2, this query would not return any solutions (despite many implicit instances of the class being available), whereas:

```

QUERY 5
SELECT ?X
WHERE { ?X a foaf:Person .}

```

would return all the company founders listed in DBpedia, since these are explicitly typed as `foaf:Persons`. ◇

We emphasise that these queries miss (implicit) results that would be intuitively expected as an answer and that could be achieved through reasoning. In the following we will substantiate this intuition referring to some of the challenges (C1–C5) mentioned in the introduction; before that, however, we need to clarify the importance of schema information and its semantics in RDF.

2.4 Inferring Additional Triples by Schema Information and Rules

In order to model schema information, which also allows to infer additional implicit information in RDF, the Semantic Web offers two ontology languages: the lightweight RDF Schema (RDFS) [13] and the expressive Web Ontology Language (OWL) [37]. Within this section, we will briefly cover an overview of the essential features of these languages in a Linked Data setting.

RDFS RDF itself already provides means to express class membership (`rdf:type`); RDF Schema (RDFS) additionally provides a special vocabulary, consisting primarily of RDF properties with a predefined semantics to model class hierarchies (`rdfs:subClassOf`), and property hierarchies (`rdfs:subPropertyOf`), as well as means to define domains and ranges that respectively allow for associating a class to the subject and object of a relation with a given property (`rdfs:domain`, `rdfs:range`). These core RDFS properties allow for describing and embedding the semantics of user-defined vocabularies in RDF itself.

OWL The Web Ontology Language (OWL) extends RDFS and allows for expressing further schema definitions in RDF, e.g., allowing to express equality of individuals (`owl:sameAs`), equivalence or disjointness of properties and classes (`owl:equivalentClass`, `owl:equivalentProperty`, `owl:disjointWith`, `owl:propertyDisjointWith`), or complex class definitions; while a full account is beyond our scope, more details on additional OWL features will be discussed in Section 4 below.

Some RDFS and OWL information expressed in RDF is shown in the light-gray boxes of Fig. 2: for instance, the classes and properties used on DBpedia are described using, amongst others, the `dbo:` and `foaf:` vocabularies (i.e., sets of terms described in their respective ontologies) which include the “schema” triples (aka. terminological triples) shown in Fig. 3.

<pre>@prefix dbo: <http://dbpedia.org/ontology/> . dbo:foundedBy rdfs:range dbo:Agent . dbo:Company rdfs:subClassOf dbo:Organisation. dbo:Organisation rdfs:subClassOf dbo:Agent. dbo:Person rdfs:subClassOf dbo:Agent. dbo:Person owl:equivalentClass foaf:Person.</pre>	<pre>@prefix foaf: <http://xmlns.com/foaf/0.1/> . foaf:name rdfs:subPropertyOf rdfs:label . foaf:made rdfs:domain foaf:Agent . foaf:Person rdfs:subClassOf foaf:Agent , geo:SpatialThing ; owl:disjointWith foaf:Organization . foaf:Organization rdfs:subClassOf foaf:Agent .</pre>
(a)	(b)

Fig. 3. Sample schema triples from the DBpedia ontology and from FOAF (retrieved 10 March 2013)

The meaning of these OWL definitions are given by two alternative (but related) semantics. The RDF-Based Semantics [64] can interpret arbitrary RDF graphs without restriction, but where common reasoning tasks are undecidable as a result. The Direct Semantics [52] is based on Description Logic (DL) theory [3,63], where a subset of RDF graphs following certain syntactic restrictions can be mapped to a set of DL axioms. A full translation from the OWL 2 DL language to the DL *SR₀IQ* is implicit in [56], where we list a small snippet of constructs used for our examples in Table 1. As such, according to Table 1, the RDF triples in Fig. 3 can be viewed as DL ontologies containing the axioms listed in Fig. 4.

Typical reasoning tasks over an expressive DL like *SR₀IQ* (e.g., using tableau methods to perform consistency checking, instance checking, satisfiability checking, etc.; see [3,63]) are in the worst case doubly-exponential even for a non-deterministic machine, and in practice are often likewise very expensive, especially at the types of scales encountered in a Linked Data setting. Furthermore,

Table 1. Mapping DL axioms to RDF using the RDFS and OWL vocabularies

	DL	RDFS
1	$A_1 \sqsubseteq A_2$	A_1 rdfs:subClassOf A_2
2	$P_1 \sqsubseteq P_2$	P_1 rdfs:subPropertyOf P_2
3	$\exists P \sqsubseteq A$	P rdfs:domain A
4	$\exists P^- \sqsubseteq A$	P rdfs:range A
5	$A_1 \equiv A_2$	A_1 owl:equivalentClass A_2
6	$A_1 \sqcap A_2 \sqsubseteq \perp$	A_1 owl:disjointWith A_2
7	$A(x)$	x rdf:type A
8	$R(x, y)$	x R y
9	$x = y$	x owl:sameAs y

$\exists \text{dbo:fundedby}^- \sqsubseteq \text{dbo:Agent}$	$\text{foaf:name} \sqsubseteq \text{rdfs:label}$
$\text{dbo:Company} \sqsubseteq \text{dbo:Organisation}$	$\exists \text{foaf:made} \sqsubseteq \text{foaf:Agent}$
$\text{dbo:Organisation} \sqsubseteq \text{dbo:Agent}$	$\text{foaf:Person} \sqsubseteq \text{foaf:Agent}$
$\text{dbo:Person} \sqsubseteq \text{dbo:Agent}$	$\text{foaf:Person} \sqsubseteq \text{geo:SpatialThing}$
$\text{dbo:Person} \equiv \text{foaf:Person}$	$\text{foaf:Person} \sqcap \text{foaf:Organisation} \sqsubseteq \perp$
	$\text{foaf:Organization} \sqsubseteq \text{foaf:Agent}$

(a)

(b)

Fig. 4. DL axioms corresponding to the DBpedia and FOAF ontology snippets from Fig. 3

the decidability of conjunctive query answering for *SR_QIQ* is still open [25].¹⁴ Thus, the W3C has identified three tractable profiles of OWL that are particularly relevant for query answering [25], where we will focus upon two such profiles in this lecture:

OWL 2 QL is designed as a language for which efficient (with respect to data complexity), sound and complete query answering can be achieved by rewriting techniques – i.e., extending conjunctive queries (such as SPARQL BGPs) to capture the semantics of the ontology.

OWL 2 RL is designed as a language for which sound and complete (with respect to assertional knowledge) reasoning can be applied by means of Datalog-style techniques – e.g., bottom-up or top-down rule-based inferencing. A standard OWL 2 RL ruleset, called OWL 2 RL/RDF, encodes part of the RDF-Based Semantics of OWL 2. The assertional entailments given by rule-based reasoning using OWL 2 RL/RDF over an OWL 2 RL ontology correspond with the assertional entailments under the Direct Semantics for that ontology [25, Theorem PR1]. The OWL 2 RL/RDF ruleset can also be applied over arbitrary RDF graphs but beyond OWL 2 RL ontologies, the aforementioned correspondence with the Direct Semantics no longer holds.

Query Rewriting. The OWL 2 QL fragment [25, Section 3] contains a combination of features that are tailored for efficient processing by query rewriting techniques. That is, given an OWL (or, respectively, its DL counterpart) ontology O in this fragment, one can answer conjunctive queries Q correctly by rewriting Q based on the axioms in O into a union of conjunctive queries (a UCQ), where we denote this process by $rewrite(Q, O)$.

To give the reader an idea, Algorithm 1 illustrates a very rudimentary version of a rewriting algorithm that implements $rewrite(Q, O)$ just for the basic RDFS axioms (axioms of the forms 1–4 from Table 1). We note that this algorithm can be viewed as a very downstripped version of the **PerfectRef** algorithm [14] which covers a much larger set of DL axioms; there have been a number of extensions and alternative query rewriting techniques proposed recently [58,62,61,46,24].

¹⁴ In this context, we note here that the case of SPARQL is decidable, since SPARQL’s BGP matching treats all variables as distinguished, see above; for further details, we refer to the SPARQL 1.1 Entailment Regimes specification [23] and a more detailed book chapter in an earlier edition of the Reasoning Web summer school [21].

Algorithm 1. Basic Query Rewriting algorithm

Input: Conjunctive query q , DL ontology \mathcal{O}
Output: Union (set) of conjunctive queries

```

1  $P := P_q$ 
2 repeat
3    $P' := P$ 
4   foreach  $q \in P'$  do
5     foreach  $g$  in  $q$  do // expansion
6       foreach axiom  $i$  of one of the forms 1–4 from Table 1 in  $\mathcal{O}$  do
7         if  $i$  is applicable to  $g$  then
8            $P := P \cup \{q[g/\text{gr}(g,i)]\}$  // see Table 2
9 until  $P' = P$ 
10 return  $P$ 

```

Table 2. Semantics of $\text{gr}(g, i)$ of Algorithm 1 (‘ $_$ ’ stands for a “fresh” variable)

g	i	$\text{gr}(g/i)$
$(x, \text{rdf:type}, A)$	$B \sqsubseteq A$	$(x, \text{rdf:type}, B)$
	$\exists P \sqsubseteq A$	$P(x, _)$
	$\exists P^- \sqsubseteq A$	$P(_, x)$
(x, P_1, y)	$P_2 \sqsubseteq P_1$	(x, P_2, y)

Example 3

Taking Query 4 again, the BGP $\{(?X, a, \text{foaf:Agent})\}$ corresponds to the conjunctive query:

$$q(?X) \leftarrow (?X, \text{rdf:type}, \text{foaf:Agent})$$

which expanded by Algorithm 1 with respect to the ontology in Fig. 4(b) results in the following UCQ:

$$\begin{aligned}
q(?X) &\leftarrow (?X, a, \text{foaf:Agent}) \\
q(?X) &\leftarrow (?X, a, \text{foaf:Person}) \\
q(?X) &\leftarrow (?X, a, \text{foaf:Organization}) \\
q(?X) &\leftarrow (?X, \text{foaf:made}, ?Y)
\end{aligned}$$

The resulting UCQ can again (using the UNION pattern) be translated back to SPARQL:

```

QUERY 4'
SELECT ?X
WHERE { { ?X a foaf:Agent } UNION
        { ?X a foaf:Person } UNION
        { ?X a foaf:Organization } UNION
        { ?X foaf:made ?Y } }

```

◇

With this small example, we have shown that the rewriting technique for OWL 2 QL can be partially applied to SPARQL queries. However, note that OWL 2 QL (and likewise the respective query rewriting algorithms from the DL literature) omit important OWL features for Linked Data (discussed later in Section 4), such as inferences from `owl:sameAs`, where rule-based inference as mentioned in the following section might be more suitable.

Rule-Based Reasoning. As an alternative to query rewriting based on the OWL 2 QL profile, another fragment of OWL – OWL 2 RL [25, Section 4] – has a normative set of rules called OWL 2 RL/RDF, which encode a subset of the semantics of RDFS and OWL 2 and can be directly used for (Datalog-style) reasoning (see also the informal RDFS entailment rules in [35, Section 7], which were later extended by OWL features in [65,53]).

Some sample OWL 2 RL/RDF rules are given in Table 3 implementing the basic features of RDFS and additionally supporting the semantics of equality for `owl:sameAs` missed by OWL 2 QL. A more detailed discussion on which OWL features (and respective rules encoding these) are particularly relevant for Linked Data Reasoning will be discussed in Section 4.

Table 3. Core RDFS and `owl:sameAs` rules

	ID	Head	Body
RDFS	prp-spo1	$(?s, ?p_2, ?o)$	$\leftarrow (?p_1, \text{rdfs:subPropertyOf}, ?p_2), (?s, ?p_1, ?o)$
	prp-dom	$(?p, \text{rdf:type}, ?c)$	$\leftarrow (?p, \text{rdfs:domain}, ?c), (?s, ?p, ?o)$
	prp-rng	$(?o, \text{rdf:type}, ?c)$	$\leftarrow (?p, \text{rdfs:range}, ?c), (?s, ?p, ?o)$
	cax-sco	$(?s, \text{rdf:type}, ?c_2)$	$\leftarrow (?c_1, \text{rdfs:subClassOf}, ?c_2), (?s, \text{rdf:type}, ?c_1)$
Same-As	eq-sym	$(?y, \text{owl:sameAs}, ?x)$	$\leftarrow (?x, \text{owl:sameAs}, ?y)$
	eq-trans	$(?x, \text{owl:sameAs}, ?z)$	$\leftarrow (?x, \text{owl:sameAs}, ?y), (?y, \text{owl:sameAs}, ?z)$
	eq-rep-s	$(?s', ?p, ?o)$	$\leftarrow (?s, \text{owl:sameAs}, ?s'), (?s, ?p, ?o)$
	eq-rep-p	$(?s, ?p', ?o)$	$\leftarrow (?p, \text{owl:sameAs}, ?p'), (?s, ?p, ?o)$
	eq-rep-o	$(?s, ?p, ?o')$	$\leftarrow (?o, \text{owl:sameAs}, ?o'), (?s, ?p, ?o)$

As opposed to query-rewriting (top-down evaluation), OWL 2 RL/RDF rules can also be applied in a bottom-up fashion for the purposes of a priori *materialization*: given a linked dataset Γ and a set of such inference rules R , pre-compute

and store the closure $\text{Im}(\text{merge}(I) \cup R)$,¹⁵ issuing queries as they are input directly against the closure. Caching the full closure thus mitigates the expense of reasoning during query-time, but may conversely incur huge pre-runtime costs, storage overheads, as well as the cost of truth maintenance in dynamic scenarios.

Example 4

As an example, let us consider the following variant of Query 3: (i) instead of explicitly following the `owl:sameAs` link, we assume the necessary inferences are supplied by reasoning; (ii) we ask for all `rdfs:labels` of the company (as opposed to just `skos:prefLabel`).

QUERY 3'

```
SELECT ?D ?R ?L
WHERE { nyt:49586210195898795812 nytimes:latest_use ?D ;
        dbo:revenueUSD ?R ;
        rdfs:label ?L }
```

while the first triple pattern is matched by explicitly stated data, the subsequent query-relevant triples must be obtained from the closure with respect to, e.g., the rules in Table 3, which contains (amongst other triples):

```
nyt:49586210195898795812 nytimes:latest_use "2010-04-27"^^xsd:date ;
dbo:revenueUSD 1.06916E11 ;
rdfs:label "IBM"@en ,
           "International Business Machines Corporation"@en .
```

leading to the following solutions:

?D	?R	?L
2010-04-27	1.06916E11	"IBM"@en
2010-04-27	1.06916E11	"International Business Machines Corporation"@en

◇

3 Overall Approaches and Challenges to Reason over and Query Linked Data

We identify two main approaches to reason and query over Linked Data:

1. Data-warehousing approaches for querying linked data are typically deployed for RDF search engines such as Sindice [54] or SWSE [40]. These engines provide query interfaces over a local centralised index of Linked Data harvested

¹⁵ That is, all RDF triples entailed by the RDF graph obtained from I (read as facts) and R .

from the Web and typically use rule-based materialisation (as presented in Section 10) to cautiously infer additional RDF triples; Section 5 will discuss such cautious reasoning techniques that are tailored to not infer too much information in this setting.

2. Rather than relying on a centralised index, Linked Data itself can be viewed as a database that can be queried directly and dynamically [33]. That is, starting from the URIs appearing in the query, or from a set of seed URIs, query-relevant data is navigated following Linked Data principles and retrieved dynamically. The main advantage of this approach is that results are much fresher and all query-relevant data do not need to be known locally. However, the main weaknesses of this approach are that performing remote lookups at query-time is costly, potentially a lot intermediate data are transferred, and the recall depends significantly on the seed URIs and conformance of the query-relevant sources with Linked Data principles. In Section 6, we will present such an approach and discuss how reasoning can be incorporated.

Getting back to the challenges enumerated in the introduction, let us now briefly discuss how these affect the architectural choice for a particular reasoning and querying infrastructure.

- C1 *Linked Data is huge.* Our example contains only sample data from two example datasets in the Linked Data Web. The most recent incarnation of the Linking Open Data cloud (from September 2011), is claimed to represent over 31 billion triples spread across 295 datasets.¹⁶ It is obvious that staying aware of and gathering this dynamically evolving data for query processing is an issue in terms of scale, but also in terms of deciding what parts of which datasets are relevant for the query at hand. For example, broad queries like Query 5, without further scope, are notoriously hard to answer, since instances of `foaf:Person` are spread over various datasets and individual RDF files spread right across the Web: while data-warehouses probably do not provide complete results on such queries, on-the-fly-traversal based approaches in the worst case can't answer such queries at all, or depending on the seed URIs, cause prohibitively many lookups during query processing.
- C2 *Linked Data is not "pure" OWL.* When all usage of the `rdfs:` and `owl:` vocabulary is within the "mappable" fragment for OWL (see, e.g., Table 1), the RDF graph in question is interpretable under the OWL Direct Semantics. However, arbitrary RDF published on the Web does not necessarily fall within these bounds. For example, the FOAF vocabulary defines inverse-functional datatype properties such as `foaf:mbox_sha1sum`, which is disallowed by the restrictions under which the Direct Semantics is defined. Even worse, one may find "harmful" RDF published online that makes reason-

¹⁶ While the LOD cloud was not updated since then, the source it is based on – <http://datahub.io/group/locloud> – listed 337 LOD datasets at the time of writing.

ing impossible or potentially uncontrollable, if done naively; for example, consider the triples:¹⁷

```

rdfs:subPropertyOf rdfs:subPropertyOf owl:sameAs .
rdf:type rdfs:subPropertyOf owl:sameAs .

```

that could have dramatic effects when fed into a rule-based reasoner (which the reader can easily convince herself when considering our example data along with the rules from Table 3). The improper use of URIs from the special `rdf:`, `rdfs:`, and `owl:` vocabularies, i.e., the use of properties and classes from these vocabularies in assertional “non-property” or “non-class” positions is also referred to as *non-standard vocabulary use* in the literature.¹⁸ Any reasoning approach for Linked Data has to ensure that harmful triples (be they erroneous or malicious) in published RDF are dealt with, either by being ignored or otherwise by keeping inferences “confined”.

- C3 *Linked Data is inconsistent.* While we have not explicitly mentioned inconsistent data in our examples – similar to non-standard use – inconsistencies often occur in published RDF data [12]. Indeed a single additional triple such as

```

dbr:Hasso_Plattner rdf:type foaf:Organization .

```

would render the example data from Fig. 2 inconsistent under OWL semantics, due to the fact that `dbr:Hasso_Plattner` is also an asserted member of the class `foaf:Person` which is declared disjoint with `foaf:Organization` in the FOAF ontology. Again, reasoning techniques need to be robust in the presence of such inconsistencies, trying to “isolate” them rather than falling for *ex falso quod libet* (anything follows from a contradiction).

- C4 *Linked Data is evolving.* As per the Web itself, Linked Data is inherently dynamic [45]. For instance, our example data is not necessarily up-to-date with Linked Data currently published by NYT: for example, an article about IBM’s “Watson” project was published in 2012 after the “Jeopardy” show,¹⁹ making the `nytimes:latest_use` date of for IBM from Query 3’ above stale. In fact, if NYT was about to update its Linked Data interface regularly with the most current articles, a query like Query 3’ would become significantly more challenging to deal with, particularly for data-warehousing approaches that do not guarantee fresh results. We discuss these issues further in Section Section 6 below.

¹⁷ A self-fulfilling prophecy: <http://axel.derri.ie/~axepol/nasty.rdf>.

¹⁸ That is, a property from these vocabularies may only be used as a predicate, and likewise classes (such as e.g. `rdfs:Resource`) may only be used as the object of `rdf:type` triples; see details within [16,38] where non-standard vocabulary use is formally defined. Though potentially very “dangerous”, non-standard triples are not always “bad”: e.g., many axiomatic triples for RDFS are non-standard.

¹⁹ Article available online at <https://www.nytimes.com/2012/01/08/jobs/building-the-watson-team-of-scientists.html> (retrieved 10 March 2013)

C5 *Linked Data needs more than RDFS and OWL.* There is more implicit knowledge hidden in Linked Data than can be captured with the semantics of RDFS and OWL alone; in fact, it may be considered quite unintuitive that Query 1 from p. 101 does not return IBM’s revenue: the exchange rate between USD and EUR is itself available as data on the Web, so why shouldn’t the Web of Data be able to make use of this knowledge? However, ontology languages like OWL and RDFS do not provide means to express mathematical conversions as necessary in this example. While not solvable with current Linked Data standards and techniques, we discuss a possible approach to tackle this problem in Section 7.

4 How Much OWL Is Needed for Linked Data?

Given the variety of combinations of techniques and RDFS/OWL profiles that can be applied for reasoning over Linked Data, an obvious question to ask is *which features of RDFS and OWL are most prominently used on the current Web of Data?*

Knowing which features are frequently used and which are infrequently used provides insights into how appropriate the coverage of various OWL profiles might be for the Linked Data use-case, and in particular, the relative costs of supporting or not supporting the semantics of a certain language primitive depending on its adoption in Web data for the setting of a given architectural choice. For example, a language feature that is costly to support or that otherwise adds complexity to a reasoning algorithm could potentially be “turned off”, with minimal practical effect, if it is found to be very infrequently used in real-world data.

In this section, we thus look at the features of RDFS and OWL that are most/least widely adopted on the Web of Data. For the purposes of this study, we take the Billion Triple Challenge 2011 corpus, which consists of 2.145 billion quadruples crawled from 7.411 million RDF/XML documents through an open crawl ran in May/June 2011 spanning 791 pay-level domains.²⁰ This corpus represents a broad *sample* of the Web of Data. We then look into the levels of adoption of individual RDFS and OWL features within this broad corpus.

4.1 Measures Used

In order to adequately characterise the uptake of various RDF(S) and OWL features used in this corpus, we present different measures to quantify their *prevalence* and *prominence*.

First, we look at the *prevalence* of use of different features, i.e., how often they are used. Here, we must take into account the diversity of the data under analysis, where few domains account for many documents and many domains account for few documents, and so forth [38]. We thus present two simple metrics:

²⁰ A pay-level domain is a direct sub-domain of a top-level domain (TLD) or a second-level country domain (ccSLD), e.g., `dbpedia.org`, `bbc.co.uk`. This gives us our notion of “domain”.

Doc: number of *documents* using the language feature

Dom: number of *pay-level-domains* (i.e., sites) using the language feature.

However, raw counts do not reflect the reality that the use of an OWL feature in one important ontology or vocabulary may often have greater practical impact than use in a thousand obscure documents. Thus, we also look at the *prominence* of use of different features. We use PageRank to quantify our notion of prominence: PageRank calculates a variant of the Eigenvector centrality of nodes (e.g., documents) in a graph, where taking the intuition of directed links as “positive votes”, the resulting scores help characterise the relative prominence (i.e., centrality) of particular documents on the Web [55,31].

In particular, we first rank documents in the corpus. To construct the graph, we follow Linked Data principles and consider sources as nodes, where a directed edge $(s_1, s_2) \in \mathcal{S} \times \mathcal{S}$ is extended from source s_1 to s_2 iff $\text{get}(s_1)$ contains (in any triple position) a URI that dereferences to document s_2 (i.e., there exists a $u \in \text{terms}(\text{get}(s_1))$ such that $\text{redirs}(u) = s_2$). We also prune edges to only consider (s_1, s_2) when s_1 and s_2 are non-empty sources in our corpus. We then apply a standard PageRank analysis over the resulting directed graph, using the power iteration method with ten iterations. For reasons of space, we refer the interested reader to [55] for more detail on PageRank, and to the following thesis [38] for more details on the particular algorithms used for this paper.

With PageRank scores computed for all documents in the corpus, for each RDFS and OWL language feature, we then present:

\sum **Rank** the *sum of PageRank scores* for documents in which the language feature is used.

With respect to \sum RANK, under the random surfer model of PageRank [55], given an agent starting from a random location and traversing documents on (our sample of) the Web of Data through randomly selected dereferenceable URIs, the \sum RANK value for a feature approximates the probability with which that agent will be at a document using that feature after traversing ten links. In other words, the score indicates the likelihood of an agent, operating over the Web of Data based on dereferenceable principles, to encounter a given feature during a random walk.

The graph extracted from the corpus consists of 7.411 million nodes and 198.6 million edges. Table 4 presents the top-10 ranked documents in our corpus, which are dominated by core meta-vocabularies, documents linked therefrom, and other popular vocabularies.²¹

4.2 Survey of RDF(S)/OWL Features

Table 5 presents the results of the survey of RDF(S) and OWL usage in our corpus, where for features with non-trivial semantics, we present the measures

²¹ We ran another similar analysis with links to and from core RDF(S) and OWL vocabularies disabled. The results for the feature analysis remained similar. Mainly `owl:sameAs` dropped several positions in terms of the sum of PageRank.

Table 4. Top ten ranked documents

№	Document URI	Rank
1	http://www.w3.org/1999/02/22-rdf-syntax-ns	0.121
2	http://www.w3.org/2000/01/rdf-schema	0.110
3	http://dublincore.org/2010/10/11/dcelements.rdf	0.096
4	http://www.w3.org/2002/07/owl	0.078
5	http://www.w3.org/2000/01/rdf-schema-more	0.049
6	http://dublincore.org/2010/10/11/dcterms.rdf	0.036
7	http://www.w3.org/2009/08/skos-reference/skos.rdf	0.026
8	http://xmlns.com/foaf/spec/	0.023
9	http://dublincore.org/DCMI.rdf	0.021
10	http://www.w3.org/2003/g/data-view	0.017

mentioned in the previous section, as well as support for the features in various RDFS/OWL profiles. Those titled EL, QL and RL refer intuitively to the standard OWL 2 profiles [25]. The RDFS standard is titled RDFS. All other profiles are non-standard proposals made in the literature. The profile titled RDFS+ refers to RDFS-Plus as proposed by Allemang and Hendler [1], which extends RDFS with lightweight OWL features. The profile titled L2 refers to a similar proposal by Fisher et al. [20] to extend RDFS with some lightweight OWL features. Description Logic Programs (DLP) was proposed by Grosz et al. [26] in order to support incomplete OWL reasoning using rules; this proposal was later built upon in Horst’s pD* profile [65]. The AL profile refers to features that can be supported with rules that do not require A-Box (i.e., assertional joins), which are expensive to compute at scale; the AL profile is used later in Section 5.

In column ‘ST’, we indicate which features have expressions that can be represented as a *single triple* in RDF, i.e., which features do not require auxiliary blank nodes of the form `_:x` or the SEQ production in Table 1 of the OWL 2 Mapping to RDF document [56]. This distinction is motivated by our initial observations that such features are typically the most widely used in Web data.

From the list of language features, we exclude `rdf:type`, which trivially appeared in 90.3% of documents. We present the table ordered by the sum of PageRank measure $[\sum \text{RANK}]$.

Regarding *prevalence*, we see from Table 5 that `owl:sameAs` is the most widely used axiom in terms of documents (1.778 million; 24%) and domains (117; 14.8%). Surprisingly (to us), RDF container membership properties (`rdf:_*`) are also heavily used (likely attributable to RSS 1.0 documents). Regarding *prominence*, we make the following observations:

1. The top six features are those that form the core of RDFS [53].
2. The RDF(S) declaration classes `rdfs:Class`, `rdf:Property` are used in fewer, but more prominent documents than OWL’s versions `owl:Class`, `owl:DatatypeProperty`, `owl:ObjectProperty`.

Table 5. Survey of RDFS/OWL primitives used on the Web of Data and support in different tractable profiles. ‘~’ denotes partial support.

Nº	Primitive	Σ Rank	Doc	Dom	RDFS	L2	RDFS+	DLP	pl*	EL	QL	RL	AL	ST
1	rdf:Property	5.74E-01	8,049	48	✓	X	X	X	✓	X	X	X	✓	✓
2	rdfs:range	4.67E-01	44,492	89	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
3	rdfs:domain	4.62E-01	43,247	89	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
4	rdfs:subClassOf	4.60E-01	115,608	109	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5	rdfs:Class	4.45E-01	19,904	43	✓	X	X	✓	✓	X	X	X	✓	✓
6	rdfs:subPropertyOf	2.35E-01	6,080	80	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
7	owl:Class	1.74E-01	302,701	111	X	X	✓	✓	✓	✓	✓	✓	✓	✓
8	owl:ObjectProperty	1.68E-01	285,412	92	X	X	✓	✓	X	✓	✓	✓	✓	✓
9	rdfs:Datatype	1.68E-01	23	9	~	X	X	~	~	~	~	~	~	~
10	owl:DatatypeProperty	1.65E-01	234,483	82	X	X	✓	✓	✓	✓	✓	✓	✓	✓
11	owl:AnnotationProperty	1.60E-01	172,290	55	X	X	X	✓	X	✓	✓	✓	✓	✓
12	owl:FunctionalProperty	9.18E-02	298	34	X	X	✓	✓	X	X	✓	X	✓	✓
13	owl:equivalentProperty	8.54E-02	141	23	X	✓	✓	✓	✓	✓	✓	✓	✓	✓
14	owl:inverseOf	7.91E-02	366	43	X	✓	✓	✓	✓	X	✓	✓	✓	✓
15	owl:disjointWith	7.65E-02	230	27	X	X	X	✓	X	✓	✓	✓	X	✓
16	owl:sameAs	7.29E-02	1,778,208	117	X	✓	✓	✓	✓	X	✓	~	~	~
17	owl:equivalentClass	5.24E-02	22,291	39	X	✓	✓	✓	✓	✓	✓	✓	✓	✓
18	owl:InverseFunctionalProperty	4.79E-02	111	24	X	X	✓	~	~	X	X	✓	✓	X
19	owl:unionOf	3.15E-02	15,162	30	X	X	X	~	X	X	X	~	~	X
20	owl:SymmetricProperty	3.13E-02	120	23	X	✓	✓	✓	✓	X	✓	✓	✓	✓
21	owl:TransitiveProperty	2.98E-02	150	30	X	✓	✓	✓	✓	X	✓	✓	X	✓
22	owl:someValuesFrom	2.13E-02	1,753	15	X	X	X	~	~	~	~	~	~	X
23	rdf:*	1.42E-02	293,022	62	✓	X	X	X	✓	X	X	✓	✓	✓
24	owl:allValuesFrom	2.98E-03	29,084	20	X	X	X	~	~	X	X	~	X	X
25	owl:minCardinality	2.43E-03	33,309	19	X	X	X	~	X	X	X	X	X	X
26	owl:maxCardinality	2.14E-03	10,413	24	X	X	X	~	X	X	X	~	X	X
27	owl:cardinality	1.75E-03	3,170	24	X	X	X	~	X	X	X	X	X	X
28	owl:oneOf	4.13E-04	74	11	X	X	X	~	X	~	X	~	~	X
29	owl:hasValue	3.91E-04	55	14	X	X	X	~	✓	✓	X	✓	✓	X
30	owl:intersectionOf	3.37E-04	186	13	X	X	X	✓	X	✓	~	~	~	X
31	owl:NamedIndividual ⁽²⁾	1.63E-04	3	2	X	X	X	X	X	✓	✓	✓	✓	✓
32	owl:AllDifferent	1.55E-04	21	8	X	X	X	X	X	✓	X	✓	X	X
33	owl:propertyChainAxiom ⁽²⁾	1.23E-04	14	6	X	X	X	X	X	✓	X	✓	X	X
34	owl:onDataRange	8.41E-05	3	1	X	X	X	X	X	X	X	X	X	X
35	owl:minQualifiedCardinality ⁽²⁾	8.40E-05	2	1	X	X	X	X	X	X	X	X	X	X
36	owl:qualifiedCardinality ⁽²⁾	4.02E-05	2	1	X	X	X	X	X	X	X	X	X	X
37	owl:AllDisjointClasses ⁽²⁾	4.01E-05	2	2	X	X	X	X	X	✓	✓	✓	X	X
38	owl:maxQualifiedCardinality ⁽²⁾	4.01E-05	1	1	X	X	X	X	X	X	X	~	X	X
39	owl:ReflexiveProperty ⁽²⁾	1.30E-05	2	1	X	X	X	X	X	✓	✓	X	✓	✓
40	owl:complementOf	1.96E-06	75	4	X	X	X	~	X	X	~	~	X	X
41	owl:differentFrom	7.18E-07	25	7	X	X	X	✓	X	✓	X	✓	✓	✓
42	owl:onDatatype	2.72E-07	1	1	X	X	X	X	X	X	X	X	X	X
43	owl:disjointUnionOf	6.31E-08	2	2	X	X	X	X	X	X	X	X	X	X
44	owl:hasKey ⁽²⁾	3.67E-08	1	1	X	X	X	X	X	✓	X	✓	X	X
45	owl:propertyDisjointWith ⁽²⁾	2.43E-08	1	1	X	X	X	X	X	✓	✓	X	✓	✓

Not Used: rdfs:ContainerMembershipProperty, owl:AllDisjointProperties⁽²⁾, owl:Annotation⁽²⁾, owl:AsymmetricProperty⁽²⁾, owl:Axiom⁽²⁾, owl:IrreflexiveProperty⁽²⁾, owl:NegativePropertyAssertion⁽²⁾, owl:datatypeComplementOf⁽²⁾, owl:hasSelf⁽²⁾

- owl:complementOf and owl:differentFrom are the least prominently used original OWL features.
- Of the features new to OWL 2, owl:NamedIndividual is the most prominently used in thirty-first position. Our crawl was conducted nineteen months

after OWL 2 became a W3C Recommendation (Oct. 2009), where we note that new OWL 2 features have had little penetration in prominent Web vocabularies during that interim. Further, several OWL 2 features were not used at all in our corpus.

5. The top eighteen features are expressible with a single RDF triple. The highest ranked primitive for which this is not the case is `owl:unionOf` in nineteenth position, which requires use of RDF collections (i.e., lists). Union classes are often specified as the domain or range of a given property: the most prominent such example is the SKOS vocabulary (the seventh highest-ranked document) which specifies the range of the `skos:member` property as the union of `skos:Concept` and `skos:Container`.

In terms of profile support, we observe that RDFS has good catchment for a few of the most prominent features, but otherwise has poor coverage. Aside from syntactic/declaration features, from the top-20 features (which cover 98% of the total cumulative rank), L2 misses functional properties_(pos=12), disjoint classes₍₁₅₎, inverse-functional properties₍₁₈₎ and union classes₍₁₉₎. RDFS-Plus omits support for disjoint₍₁₅₎ and union classes₍₁₉₎. DLP, as defined by Volz [70, §A], has coverage of all such features, but does not support inverse-functional₍₁₈₎ datatype properties. pD* does not support disjoint₍₁₅₎ or union classes₍₁₉₎.

Regarding the standard OWL 2 profiles, OWL 2 EL and OWL 2 QL both omit support for important top-20 features. Neither include functional₍₁₂₎ or inverse-functional properties₍₁₈₎, or union classes₍₁₉₎. OWL 2 EL further omits support for inverse₍₁₄₎ and symmetric properties₍₂₀₎. OWL 2 QL does not support the prevalent same-as₍₁₆₎ feature. Conversely, OWL 2 RL has much better coverage, albeit having only partial support for union classes₍₁₉₎.

Summing up, we acknowledge that such a survey cannot give a universal or definitive indication of the most important OWL features for Linked Data. First, we only survey a limited sample of the Web of Data. Second, the future may (or may not) see radical changes in how OWL is used on the Web; e.g., OWL 2 terms may soon enjoy more adoption. Still, Table 5 offers useful anecdotal insights into the extant *trends* of adoption of RDFS and OWL on the Web, and what features are the most crucial to support in a current Linked Data setting.

4.3 Survey of Datatype Use

Implementing the full range of RDF, XSD and OWL datatypes is often costly [18], with custom code (or an external library) required to support each one. We are thus interested to see which ones are most important to support.

Aside from plain literals, the RDF semantics defines a single datatype supported under RDF-entailment: `rdf:XMLLiteral` [35]. However, the RDF semantics also defines D-entailment, which provides interpretations over a datatype map that gives a mapping from lexical datatype strings into a value space. The datatype map may also impose disjointness constraints within its value space. These interpretations allow for determining which lexical strings are valid for a datatype, which different lexical strings refer to the same value and which to different values,

Table 6. Survey of standard datatypes used on the Web of Data

N ^o	Primitive	Σ Rank	Lit	Doc	Dom	D	O2
1	xsd:dateTime	4.18E-2	2,919,518	1,092,048	68	✓	✓
2	xsd:boolean	2.37E-2	75,215	41,680	22	✓	✓
3	xsd:integer	1.97E-2	1,015,235	716,904	41	✓	✓
4	xsd:string	1.90E-2	1,629,224	475,397	76	✓	✓
5	xsd:date	1.82E-2	965,647	550,257	39	✓	X
6	xsd:long	1.63E-2	1,143,351	357,723	6	✓	✓
7	xsd:anyURI	1.61E-2	1,407,283	339,731	16	✓	✓
8	xsd:int	1.52E-2	2,061,837	400,448	31	✓	✓
9	xsd:float	9.09E-3	671,613	341,156	21	✓	✓
10	xsd:gYear	4.63E-3	212,887	159,510	12	✓	X
11	xsd:nonNegativeInteger	3.35E-3	9,230	10,926	26	✓	✓
12	xsd:double	2.00E-3	137,908	68,682	31	✓	✓
13	xsd:decimal	1.11E-3	43,747	13,179	9	✓	✓
14	xsd:duration	6.99E-4	28,541	28,299	4	X	X
15	xsd:gMonthDay	5.98E-4	34,492	20,886	3	✓	X
16	xsd:short	5.71E-4	18,064	11,643	2	✓	✓
17	rdf:XMLLiteral	4.97E-4	1,580	791	11	✓	✓
18	xsd:gMonth	2.50E-4	2,250	1,132	3	✓	X
19	rdf:PlainLiteral	1.34E-4	109	19	2	X	✓
20	xsd:gYearMonth	8.49E-5	6,763	3,080	5	✓	X
21	xsd:positiveInteger	5.11E-5	1,423	1,890	2	✓	✓
22	xsd:gDay	4.26E-5	2,234	1,117	1	✓	X
23	xsd:token	3.56E-5	2,900	1,450	1	✓	✓
24	xsd:unsignedByte	2.62E-7	66	11	1	✓	✓
25	xsd:byte	2.60E-7	58	11	1	✓	✓
26	xsd:time	8.88E-8	23	4	3	✓	X
27	xsd:unsignedLong	6.71E-8	6	1	1	✓	✓
–	other xsd/owl dts. not used	–	–	–	–	–	–

and which sets of datatype values are disjoint from each other. An XSD-datatype map is then defined that extends the set of supported datatypes into those defined for XML Schema, including types for boolean, numeric, temporal, string and other forms of literals. Datatypes that are deemed to be ambiguously defined (viz. `xsd:duration`) or specific to XML (e.g., `xsd:QName`), etc. are omitted.

The original OWL specification recommends use of a similar set of datatypes to that for D-entailment, where compliant reasoners are required to support `xsd:string` and `xsd:integer`. Furthermore, OWL allows for defining enumerated datatypes.

With the standardisation of OWL 2 came two new datatypes: `owl:real` and `owl:rational`, along with novel support for `xsd:dateTimeStamp`. However, XSD datatypes relating to date, time and Gregorian calendar values are not supported. OWL 2 also introduced mechanisms for defining new datatypes by restricting facets of legacy datatypes; however, from Table 5 we note that `owl:onDatatype` (used for facet restrictions) has only very few occurrences in our corpus.

Given this broad collection of datatypes, it is interesting to see which ones are most commonly used on the Web of Data, and which ones are thus a priority to support, where we use a similar methodology as presented before. In our corpus, we found 278 different datatype URIs assigned to literals. Of these, 158 came from the DBpedia exporter which models SI units, currencies, etc., as datatypes. Using analogous measures as before, Table 6 lists the top *standard* RDF(S), OWL

and XSD datatypes as used to type literals in our corpus. We omit plain literals which were used in 6.609 million documents (89%). The LIT column indicates the number of literals with that datatype. D indicates the datatypes supported by D-entailment with the recommended XSD datatype map. O2 indicates the datatypes supported by OWL 2.

We make the following observations based on Table 6:

1. The top four standard datatypes are supported by both the traditional XSD datatype map and by OWL 2.
2. OWL 2 does not support `xsd:date`₍₅₎, `xsd:time`₍₂₆₎, or the various Gregorian datatypes_(10,15,18,20,22).
3. Despite not being supported by any standard entailment, `xsd:duration`₍₁₄₎ was used in 28 thousand documents across four different domains.
4. Various standard datatypes are not used at all in the data. For example, `xsd:dateTimeStamp`, the “new” OWL datatypes, binary datatypes and various normalised-string/token datatypes did not appear at all.²²

4.4 A Profile of OWL for Linked Data?

Our analysis of the adoption of RDFS and OWL has shown that while some features are broadly adopted on the Web of Data, others are barely adopted at all. Thus, it would seem possible, for example, to only support some fraction of the standard OWL features while capturing support for the broad majority of axioms present on the Web of Data. In Table 5, we already saw that the most frequently used features corresponds with the ability to represent their respective axioms in RDF as a single triple (and thus without blank nodes if being interpreted under the Direct Semantics).

In previous work, we thus proposed the OWL LD (Linked Data) profile, which is a proper subset of OWL 2 RL supporting only those features that are expressible with a single RDF triple [22]. The RDF-Based Semantics of the OWL LD profile can be (partly) supported by means of a subset of the OWL 2 RL/RDF rules relating to the supported features. We also provide a grammar under which the Direct Semantics of the profile can be supported, making (optionally) conformant documents compatible with the OWL Direct Semantics. We propose that OWL LD – as a practical, terse profile – facilitates greater ease of implementation for Linked Data reasoning applications, while maintaining high coverage of commonly used features.

5 Rule-Based Inference for Linked Data: Authoritative vs. Context-Dependent Reasoning

A common strategy for reasoning over multiple sources is to simply merge them together and compute the deductive closure over the resulting monolithic RDF

²² In fact, `owl:real` does not have a lexical space, and thus cannot be written down; irrational numbers are difficult to write down.

graph. However, when dealing with arbitrary sources from the Web, one cannot expect the data to always adhere to strict rules or to be universally infallible. Web data is highly heterogeneous and unexpected usage of data and data schema is common. For example, data can be erroneous or crafted for malicious purposes. As a consequence, there are risks for a reasoner to infer undesirable logical assertions, which can be harmful for the system. These assertions increase the noise in the data collection and decrease the precision of the querying system. In addition, such inferences add an unnecessary computational overhead, which augments the demand of computational resources and limits the performance of the system. Therefore, a requirement of inference engines for Web data is the ability to cope with disparate data quality, where, in fact, incompleteness (with respect to standard RDFS/OWL profiles) is thus a desirable feature.

In this section, we present two cautious approaches for applying rule-based inferencing over diverse Linked Data in a robust manner: Context-Dependent reasoning [17] and Authoritative Reasoning [12]. Both have been designed to cope with disparate data quality and to work at large scale. However, each approach has been optimised for different scenarios. Context-Dependent reasoning is optimised for reasoning over a large number of small graphs, whereas Authoritative Reasoning is optimised for reasoning over a large single graph.

5.1 Context-Dependent Reasoning

Context-Dependent reasoning has been developed to meet the requirements of the Sindice search engine project [54]. The Sindice search engine indexes a large number of Linked Data documents, each of which contains a small RDF graph. Reasoning over these graphs enables to make explicit what would otherwise be implicit knowledge, adding value to Sindice’s search engine results to ultimately be more competitive in terms of precision and recall [50].

The Context-Dependent reasoning approach has been designed to work on a multitude of small graphs in a distributed manner. Each computing node will perform the deductive closure of one graph. A *data source* is divided into small sub-graphs, e.g., on a per-entity basis or on a per-document basis such as in the Sindice search engine. Each of these graphs represents a *contextual graph*. Larger contextual graphs can be constructed from smaller ones depending on the needs. For example, one can aggregate all documents that are connected with `owl:sameAs` links into a single contextual graph if one needs to reason across `owl:sameAs` links.

A fundamental requirement in the design of the Context-Dependent reasoning approach has been to confine T-Box claims (aka., terminological claims, aka. schema claims, as per Table 3) and reasoning tasks into “contexts” in order to track the provenance of inference results. By tracking the provenance of each individual T-Box claim, we are able to prevent one ontology to alter the semantics of other ontologies on a global scale. In addition, such a context-dependent approach provides an efficient distributed computing model which scales linearly with the amount of data [17].

To reason over contexts, we assume that the ontologies that these contexts refer to are either included explicitly with `owl:imports` declarations or implicitly by using property and class URIs that dereference directly to the data describing the ontology itself. This later case should be the standard if the W3C best practices for publishing ontologies [51] and the Linked Data principles [7] are followed by data publishers. As ontologies might refer to other ontologies, the import process then needs to be recursively iterated as explained in the next section.

A naive approach would be to execute such a recursive fetching for each contextual graph and to create an *aggregate context* [28], i.e., the *RDF merge* of the contextual graph and of the imported ontologies. At this point the deductive closure of the aggregate context can be computed. Such a naive procedure is however obviously inefficient since a lot of processing time will be used to recalculate the T-Box deductions which could be instead reused for possibly large numbers of other contextual graphs. Thus an ontology base is used to store and reuse such deductions and is described next.

Reasoning with Contexts. The notions of context and lifting rules presented in the following are based on Guha’s context mechanism [28]. Its aim is to control the integration of data and ultimately avoid the aggregation of data that may result in undesirable inferred assertions.

Within his framework, a *Context* is a first class resource and denotes the scope of validity of a statement. The contents of the context are said to be true in that context. This scope is defined by the symbol *ist* (“is true in context”), introduced by Guha in [27]. The notation $ist(c, \varphi)$ states that a proposition φ is true in the context c . Since contexts are first class objects, it becomes possible to define expressive formulae whose domains and ranges are contexts. An example is the so called *Lifting Rule* that enables to *lift* axioms from one context to another.

An *Aggregate Context* is a subclass of *Context*. Its content is composed by the contents lifted from other contexts. An aggregate context must contain the full specification of what it imports. In our case, each contextual graph is considered an *Aggregate Context*, since it always contains the specification of what it imports through explicit or implicit import declarations, as explained next.

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

The `owl:imports` primitive is transitive. That is, an import declaration states that, when reasoning with an ontology O , one should consider not only the axioms of O , but the entire *import closure* of O . The *import closure* of an ontology O is the smallest set containing the axioms of O and all of the axioms from the ontologies that O (transitively) imports. For example, if ontology O_A imports O_B , and O_B imports O_C , then O_A imports both O_B and O_C .

Implicit Import Declaration. Most RDF models published on the Web do not contain explicit `owl:imports` declarations. For example, among the 228 million documents in Sindice, only 704 thousand declare at least one `owl:imports` link; also the example `dbo:` and `foaf: ontologies`²³ in our examples do not contain any explicit `owl:imports` links. Instead, many RDF models generally refer to existing ontologies by their classes or property URIs. For example, most FOAF profile documents do not explicitly import the FOAF ontology, but instead just directly use the classes and properties of the FOAF vocabulary, which dereference to the FOAF ontology. Following Linked Data principles, the URIs of the classes and properties defined in an ontology should be dereferenceable and should provide the machine-processable definition of the vocabulary (presumably given in RDFS/OWL).

That is, in the presence of dereferenceable class or property URIs, we perform what we call an *implicit import*. By dereferencing the URI, we attempt to retrieve a graph containing the description of the ontological entity identified by this URI and to include its content inside the source RDF model. The implicit import is also considered transitive.

Example 5

In Fig. 2, if a RDF model such as `dbr:Werner_von_Siemens` refers to an ontological entity such as `dbo:Person` from the ontology `dbo`, and if `dbo` refers to an ontological entity `foaf:Person` in an ontology `foaf`, then the model imports the two ontologies given by `dbo` and `foaf`. \diamond

Import Lifting Rules. Guha’s context mechanism defines the *importsFrom* lifting rule [28] which corresponds to the inclusion of one context into another. The *owl:imports* primitive and the implicit import declaration are easily mapped to the *importsFrom* rule.

A particular case is when import relations are cyclic. Importing an ontology into itself is considered a null action, so if ontology O_A imports O_B and O_B imports O_A , then the two ontologies are considered to be equivalent [4]. Based on this definition, we extend Guha’s definition to allow cycles in a graph of *importsFrom*. We introduce a new symbol *eq*, and the notation $eq(c_1, c_2)$ states that c_1 is equivalent to c_2 , i.e., that the set of propositions true in c_1 is identical to the set of propositions true in c_2 .

Definition 9 (Cyclic Import Rule). *Let c_1 and c_2 be two contexts. If c_1 contains the proposition $importsFrom(c_1, c_2)$ and c_2 the proposition $importsFrom(c_2, c_1)$, then the two contexts are considered equivalent:*

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

²³ In reality, the ontology defining the vocabulary in the `dbo` namespace is split over many documents: one per class and property term; however, this is not important for the current discussion.

Deductive Closure of RDF Models. In Context-Dependent reasoning, the deductive closure of a graph is the set of assertions that are entailed in the aggregate context, composed of the graph itself and its ontology import closure. We now explain how the deductive closure of an aggregate context is performed. Given two contexts c_1 and c_2 , for example a Linked Data document and an ontology, their axioms are lifted into an aggregate context labelled $c_1 \wedge c_2$. The deductive closure of the aggregate context is then computed using the rule-based inference engine.

It is to be noticed that the deductive closure of an aggregate context can lead to inferred statements that are not true in any of the source contexts alone.

Example 6

In Fig. 2, if a context c_1 contains an instance `dbr:Werner_von_Siemens` of the class `dbo:Person`, and a context c_2 contains a proposition stating that `dbo:Person` is equivalent to `foaf:Person`, then the entailed conclusion that `dbr:Werner_von_Siemens` is a `foaf:Person` is only true in the aggregate context $c_1 \wedge c_2$:

$$\begin{aligned} & \text{ist}(c_1, \text{dbo:Person}(x)) \wedge \\ & \text{ist}(c_2, \text{equivalentClass}(\text{dbo:Person}, \text{foaf:Person})) \rightarrow \\ & \text{ist}(c_1 \wedge c_2, \text{foaf:Person}(x)) \end{aligned}$$

◇

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

Definition 10 (Aggregate Entailment). *Let c_1 and c_2 be two contexts with respectively two propositions φ_1 and φ_2 , $\text{ist}(c_1, \varphi_1)$ and $\text{ist}(c_2, \varphi_2)$, and $\varphi_1 \wedge \varphi_2 \models \varphi_3$, such that $\varphi_2 \not\models \varphi_3$, $\varphi_1 \not\models \varphi_3$; then we call φ_3 a newly entailed proposition in the aggregate context $c_1 \wedge c_2$. We call the set of all newly defined propositions an aggregate entailment and denote it as Δ_{c_1, c_2} :*

$$\begin{aligned} \Delta_{c_1, c_2} = \{ & \text{ist}(c_1, \varphi_1) \wedge \text{ist}(c_2, \varphi_2) \models \text{ist}(c_1 \wedge c_2, \varphi_3) \\ & \text{and } \neg(\text{ist}(c_1, \varphi_3) \vee \text{ist}(c_2, \varphi_3)) \} \end{aligned}$$

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

When considering (in our case (Horn) rule-based) RDFS/OWL inferences only, aggregate contexts enjoy the following monotonicity property²⁴: if the aggregate context $c_1 \subseteq c_2$ then $ist(c_2, \phi)$ implies $ist(c_1, \phi)$, or respectively, for overlapping contexts, if $ist(c_1 \cap c_2, \phi)$ implies both $ist(c_1, \phi)$ and $ist(c_2, \phi)$. This property is exploited in the implementation of the ontology base, which is described next, to avoid storing duplicate inferred statements.

Context-Dependent Ontology Base. A problem when reasoning over a large number of contextual graphs independently is that the process of computing the ontology import closure and its deductive closure has to be repeated for each contextual graph. This is inefficient since the computation of the import closure and the deductive closure is resource demanding and can in fact be reused for other contextual graphs. The import closure necessitates executing multiple Web requests that place load on the network and take time, whereas the computation of the deductive closure is CPU bound. In addition, the computation of the T-Box closure is more CPU intensive than the computation of the A-Box closure [17]. This observation suggests to focus on the optimisation of the T-Box closure computation. Thanks to the smaller scale of the T-Box with respect to the A-Box, we can store the computed ontology import closure as well as the deductive closure in an *ontology base* in order to reuse them in later computation.

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

Details on the formalisation of the ontology base and of an optimised strategy to update the ontology base can be found in [17].

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

The reasoning engine that is used by the ontology base is specifically designed and optimised to compute entailments in memory using a standard bottom-up semi-naive evaluation approach. Each RDF term in a statement is mapped to a unique identifier (integer). Statements are indexed using in-memory data structures, similar to triple tables, in order to lookup any kind of statement patterns. Rules are then checked against the index in an iterative manner, with

²⁴ We remark here that under the addition of possibly non-monotonic rules to the Semantic Web architecture, this context monotonicity only holds under certain circumstances [59].

one rule being applied at a given iteration. The result of the rule is then added to the index before proceeding to the next iteration. Iterations continue until a fixpoint is reached. For rules that requires joins between multiple statements, since we are working with a small amount of data and a small number of elements, we rely on an efficient merge-join algorithm where both relations are sorted on the join attribute using bit arrays. The bit arrays are then intersected using bitwise operations.

The A-Box reasoning process is distributed by dividing up the large A-Box on a per-context basis. Each context provides a chunk of data that is distributed to different computing nodes. A computing node acts independently as an A-Box reasoner and has its own ontology base. The A-Box rule engine is based on the same rule engine used by the ontology base.

Since each chunk of data is relatively small, the deductive closure of the A-Box can be entirely performed in memory without relying on disk accesses. With respect to other distributed approaches that perform reasoning on the global model, we avoid reading and writing multiple times the data directly from the disk, and therefore we obtain better performance. Importantly, the distributed model scales linearly with the number of available nodes in the cluster since replicating the ontology base on each machine allows for embarrassingly parallel execution during A-Box reasoning.²⁵

The Context-Dependent reasoning implementation has been in use by the Sindice search engine since 2008. The reasoner supports the pD* profile [65], though the Context-Dependent approach generalises straightforwardly to any materialisation mechanism. It is running on a Hadoop cluster of 150 computing nodes as part of the indexing pipeline of Sindice. It has enabled Sindice to reason over more than 700 million documents, which represents a total of more than 50 billion triples.

5.2 Authoritative Reasoning

The Authoritative reasoning algorithm was developed to provide RDFS and OWL materialisation support in the context of the Semantic Web Search Engine (SWSE) project [40], with similar proposals made in the context of reasoning over class hierarchies for the Falcons search engine [15]. As opposed to the Context-Dependent method, which partitions the problem of reasoning into a large collection of (relatively) small contexts, the Authoritative reasoning algorithm rather considers a single large RDF graph (in line with its SWSE use-case). Tackling the fallibility of Web data, Authoritative reasoning builds a single global T-Box that only includes axioms from “trusted sources”. The core intuition of authoritative reasoning is that the T-Box axioms extracted from an ontology on the Web should only be able to affect reasoning over data that instantiates the terms in that ontology’s namespace.

²⁵ That is, no communication is required between machines, where each can thus process their own content independently

Example 7

In the data we merge from various Web sources, assume we find the following two triples, both of which we consider to be T-Box axioms and neither of which we initially know whether to trust or not:

```
foaf:Person rdfs:subClassOf geo:SpatialThing .
foaf:Person rdfs:subClassOf ex:EvilEntity .
```

Let's take three triples instantiating the classes involved:

```
ex:Fred a foaf:Person .
ex:Jill a geo:SpatialThing .
ex:Tim a ex:EvilEntity .
```

Under RDFS semantics, these triples have the following corresponding entailments:

```
ex:Fred a geo:SpatialThing .
ex:Fred a ex:EvilEntity .
```

According to the semantics of `rdfs:subClassOf`, the original T-Box axioms only affect the inferences possible over data instantiating the `foaf:Person` class. As to whether these T-Box axioms can be trusted, we thus ask: are either of these T-Box axioms given in the document dereferenced by `foaf:Person`? The first one is indeed in the FOAF ontology, and hence can be trusted (and is considered “authoritative” along with its inferences). The second one is not, and will not be considered by the authoritative reasoning process. ◇

This intuitive notion of which sources to trust for individual T-Box axioms then relies on two prerequisites:

T-Box distinguishable from A-Box: We assume that T-Box triples in the data (and triple patterns in the rules) can be distinguished from A-Box triples.

Authoritative relation: We assume an authoritative relation that maps from an RDF document to a set of RDF terms it can speak authoritatively about.

We now discuss these general prerequisites in more detail for the setting of applying RDFS/OWL reasoning over Linked Data.

T-Box distinguishable from A-Box. We discussed previously that T-Box data intuitively refers to ontological/schema definitions using the RDFS and OWL standards to define the semantics of classes and properties in a vocabulary. This intuition is sufficient for our purposes, where more precise definitions of T-Box and A-Box in the context of Authoritative reasoning are provided, e.g., in [12].

Loosely related to the notion of meta-modelling in OWL, our A-Box also contains the T-Box data (but not vice-versa). Thus, we can reason over schema triples analogous to if they were assertions. We also split the body of rules into a (possibly empty) A-Box and (possibly empty) T-Box graph pattern, where we define a T-Box triple pattern as any pattern that can only unify with a T-Box triple, and we define an A-Box triple pattern as the complement.

Example 8

Take the following OWL rule (cax-eqc1 in OWL 2 RL/RDF):

$$(?X, a, ?C_2) \leftarrow \underline{(?C_1, owl:equivalentClass, ?C_2)}, (?X, a, ?C_1)$$

The first (underlined) triple pattern in the body is considered T-Box since it can only be matched by T-Box triples. The second triple pattern in the body is considered A-Box because it is not a T-Box pattern. We do not need to categorise the head of the rule in this manner. Of course, the A-Box pattern in the body may also match a T-Box triple, as per the previous meta-modelling discussion. \diamond

Authoritative reasoning then involves checking the source of T-Box knowledge. Incorrect or malicious T-Box triples are the most “dangerous” in a reasoning context, where, for example, if a dataset contains millions of instances of `foaf:Person`, a single T-Box triple stated in an arbitrary location – such as one of the following

```
foaf:Person rdfs:subClassOf ex:EvilEntity .
foaf:Person rdfs:subClassOf owl:Nothing .
```

can affect inferences computed for all the millions of instances of `foaf:Person` defined in Linked Data.

Authoritative Relation. Next, we need to establish a relationship between RDF sources and the set of RDF terms they speak authoritatively for. In the Linked Data setting, we can establish this authoritative relation by directly using the notion of dereferencing.

Definition 11 (Authoritative sources for terms). *We denote a mapping from a source URI to the set of terms it speaks authoritatively for as follows:*

$$\text{auth} : S \rightarrow 2^C \\ s \mapsto \{c \in U \mid \text{redirs}(c) = s\} \cup (\text{terms}(\text{get}(s)) \cap B)$$

A source is thus authoritative for all URIs that dereference to it and all blank nodes it mentions. This formalises, for example, the intuitive relationship that exists between the FOAF ontology and the set of terms in the `foaf:*` namespace

that dereference to it.²⁶ No document is considered authoritative for literals, though this has little effect on the reasoning process.

Authoritative reasoning is applied over a Linked Dataset as given in Definition 1, which tracks the source associated with each RDF graph. Furthermore, the algorithm requires knowledge about redirects to establish the authoritative function. In practice, these data are replicated locally for the reasoning engine to access; the reasoner does not perform live lookups.

Authoritative Reasoning. The primary goal of the authoritative reasoning process is to safe-guard widely used vocabularies from redefinition in arbitrary locations. Precise definitions and guarantees for authoritative reasoning are given elsewhere in [38,12]. Here sketching the main intuition, given an ontology O providing a set of T-Box axioms and G an arbitrary RDF graph (e.g., a Web document or a merge of documents), if G does not mention any term for which O is authoritative, and O is not an implicit import of such a document, then we do not want the *T-Box axioms* provided by O to affect materialisation over G .

Thus, for example, if G instantiates vocabulary terms from the FOAF ontology but not from the DBpedia ontology, then the T-Box extracted from DBpedia should not affect inferencing over the A-Box of G . The implicit imports of the FOAF ontology can, however, affect inferencing over the T-Box of G , even if their terms are not explicitly mentioned. For example, the FOAF ontology states that `foaf:Person` is a sub-class of `geo:SpatialThing`; if G contains instances of `foaf:Person`, they will be inferred to be instances of `geo:SpatialThing` and it will then be the prerogative of the corresponding WGS84 Geo Ontology to define what inferences are possible over the latter class, even though the corresponding class is not explicitly referenced by G .

Whether or not a T-Box axiom is considered authoritative then directly depends on the rules being applied in the reasoning process. In fact, a T-Box axiom may be authoritative with respect to one rule and not another.

Example 9

Take the following T-Box triple from Fig. 1:

```
dbo:Person owl:equivalentClass foaf:Person .
```

This triple is given by the document that `dbo:Person` dereferences to. If we then take OWL 2 RL/RDF rule `cax-eqc1` mentioned in Example 8:

$$(?X, a, ?C_2) \leftarrow \underline{(?C_1, owl:equivalentClass, ?C_2)}, (?X, a, ?C_1)$$

the T-Box triple is authoritative for this rule since it translates data about `dbo:Person` instances into `foaf:Person` instances.

²⁶ The source URI will often not share the namespace of the URIs it is authoritative for since redirects (esp. PURLS) are commonly used for dereferencing schemes.

Now, if we take the same T-Box triple but instead take OWL 2 RL/RDF rule `cax-eqc2`:

$$(?X, a, ?C_1) \leftarrow (?C_1, \text{owl:equivalentClass}, ?C_2), (?X, a, ?C_2)$$

the T-Box triple is no longer authoritative since it translates instance data about `foaf:Person` into `dbo:Person` instances, and as justified before, we do not want the DBpedia ontology document to be able to affect inferences over (FOAF) data that do not contain any DBpedia terms for which the document is authoritative. \diamond

Thus, we see that T-Box axioms are only authoritative with respect to the rule(set) under consideration. When applying rules over the data, we can then apply a relatively straightforward (and slightly stricter) condition to ensure that the T-Box axiom matched in the body of the rule will lead to an authoritative inference. Recall that the document serving the T-Box axiom should be authoritative for at least one term mentioned in the A-Box being reasoned over. We thus look at the terms bound to variables that appear in both the T-Box and A-Box part of the rule body. For a given rule application, if the document providing the T-Box axiom is authoritative for at least one such term, we deem the rule application to be authoritative; otherwise we consider it to be non-authoritative.

Example 10

From the previous example, if we look at rule `cax-eqc1`, the only variable common to the T-Box and A-Box segments of the rule body is `?C1`. Taking the given T-Box axiom, `?C1` is bound to `dbo:Person` for which the T-Box source is authoritative. Hence, for any triple of the form `(?X, a, dbo:Person)` in our A-Box data, we can authoritatively infer the corresponding triple of the form `(?X, a, foaf:Person)`.

Instead taking rule `cax-eqc2`, the only variable common to the T-Box and A-Box segments of the rule body is `?C2`. For the given T-Box axiom, `?C2` is bound to `foaf:Person` for which the DBpedia ontology is not authoritative. Hence, for any A-Box triple of the form `(?X, a, foaf:Person)` in our data, authoritative reasoning will block the inference of `(?X, a, dbo:Person)` (unless the T-Box axiom is also given in the FOAF ontology, which in reality it is not). \diamond

If a rule contains only T-Box or only A-Box patterns in its body, authoritative reasoning does not block the inferences. Any inferences from T-Box level reasoning are assigned to the context of the source from which *all* of the premises originate; if a T-Box level inference involves premises from multiple documents,

it is not considered to have any fixed source and can never be authoritative.²⁷ Standard Authoritative reasoning does not affect rules consisting of only A-Box patterns, which includes rules that provide `owl:sameAs` entailments over instances (see Table 3).

Implementation and Scalability Unlike the Context-Dependent reasoning algorithm, Authoritative reasoning does not partition the problem of reasoning into small contexts. Instead, Authoritative reasoning requires applying inference over the entire dataset in “one go”. Thus, the methods of inference applied must scale to the entire dataset (and not just individual contexts). We implement such methods in the Scalable Authoritative OWL Reasoner (SAOR) [41], designed to apply lightweight OWL reasoning over large collections of diverse Linked Data. The reasoning process is divided into two distinct phases, as follows:

Compute T-Box. The T-Box is extracted from the main body of data and axioms are analysed for authoritativeness with respect to the given ruleset. If required, T-Box level reasoning is applied.

Reason over A-Box. The A-Box is reasoned over with respect to the global authoritative T-Box built in the previous phase.

In terms of scalability, when dealing with large collections of Linked Data, we observe that the T-Box is generally quite small (e.g., typically < 0.1% of the total triple count [41]) and is frequently accessed during the reasoning process; hence we load the T-Box into memory such that it can be accessed efficiently. Furthermore, a variety of papers have demonstrated that splitting the T-Box from the main body of data allows for effective distributed computation of materialised inferences [72,69,41,68], where (similar to Context-Dependent reasoning) the T-Box is replicated to each machine performing inference. Indeed, if the ruleset does not contain any rules with more than one A-Box pattern in the body (as is the case for, e.g., the RDFS inference rules [cf. Table 3] and for rules `cax-sco`, `cax-eqc1` and `cax-eqc2` introduced previously in the examples), then this form of distributed materialisation can reason over the A-Box in an embarrassingly parallel fashion for any arbitrary distributed partitioning of the A-Box data. Rules with multiple A-Box patterns in the body require joins over the very large A-Box (typically between machines), and in many cases, such rules can produce huge volumes of materialisations; for example, transitive-property reasoning is quadratic with respect to the extension of that property in the A-Box.

An important practical question then is how much is lost by not considering rules that have multiple A-Box patterns in the body? In Table 5, the AL profile lists the features that can be supporting using “A-Linear rules”: rules with only one assertional pattern [41]. In SAOR, we implement A-Linear OWL

²⁷ In any case, informally, we can conjecture that terminology-specific reasoning in rule-sets such as OWL 2 RL/RDF is (often) redundant with respect to assertional inferencing applied recursively; for example, performing the transitive closure of sub-class relations is only necessary to infer sub-class relations, where recursive application of `cax-sco` will infer all assertions without it.

2 RL/RDF rules: the intersection of AL and RL. One of the most prominent features we lose is the ability to reason over `owl:sameAs` relations; both to infer such relations through, e.g., functional properties and inverse-functional properties, and to support the semantics of equality as per the rules in Table 3 (only `eq-sym` is A-Linear).

In terms of completeness with respect to standard bottom-up rule-evaluation (i.e., without any distinction between T-Box or A-Box), the main limitation of considering a separate static T-Box while reasoning over the A-Box is that it can lead to incompleteness if new T-Box triples are found while reasoning over the A-Box [41] (these triples will not be reflected in the T-Box). Inference of T-Box triples during A-Box reasoning can occur due to non-standard use of the core RDFS or OWL vocabulary (see Section 3). Workarounds for this problem are possible: for example to recursively identify and reason over non-standard triples in a pre-processing step, etc. However, non-standard use of the RDF(S) and OWL vocabulary is not found often in Linked Data, with notable exceptions being, e.g., the RDFS axiomatic triples and the documents dereferenced by the RDF, RDFS and OWL terms themselves.

In the SAOR system, following previous papers [72,69], we also perform A-Box reasoning in a distributed setting. We have evaluated the applicability of SAOR over 1 billion Linked Data triples taken from 4 million Linked Data documents. Using a variety of optimisations for our A-Linear profile of OWL 2 RL/RDF, on a cluster of nine machines with 4GB of RAM and 2.2 GHz single-core processors, we computed 1 billion unique and authoritative inferences in about 3.5 hours [41], roughly doubling the input size. Without considering the authority of inferences, we estimated that the volume of materialisation would increase by $55\times$, even for the lightweight reasoning profile being considered [12].

5.3 Comparison and Open Issues

Meeting the Challenges Tackling C1 (scalability) in the list of challenges enumerated in Section 3, both Context-Dependent reasoning and Authoritative reasoning use distributed computing and partitioning techniques and various rule-based optimisations to enable high levels of scale.

Tackling C2 (impure and fallible OWL), both approaches analyse the source of input axioms and apply cautious materialisation, where incompleteness with respect to standard OWL profiles is thus a feature, not a “bug”.²⁸ Both approaches can use rule-based inferencing to support an incomplete RDF-Based semantics, which does not require input graphs to conform to OWL 2 DL restrictions enforced by OWL’s Direct Semantics.

Regarding C3 (inconsistencies), both approaches use monotonic rule-based reasoning techniques that do not reduce the deductive reasoning process to unsatisfiability checking, and thus do not fall into “ex falso quod libet”. Inconsistencies can be ignored. However, in the case of SAOR, we have also looked at resolving the contradictions presented by inconsistencies: we investigated using

²⁸ Importantly, a *non-standard* version of completeness can be rigorously defined in both cases. See, e.g., [41] for details in the SAOR case.

an annotated logic program framework to rank assertions under a PageRank model, where the marginal assertion in a contradiction is defeated [12].

With respect to C4 (dynamic Linked Data), Context-Dependent reasoning allows entailments to be updated on a context-by-context basis, where changes to the ontology base can also be efficiently supported (see [17]); Authoritative reasoning does not directly support incremental updates, where truth maintenance techniques would be required. (The following section presents an approach that better handles reasoning and querying over Linked Data in highly dynamic scenarios.)

With respect to C5 (more than RDFS/OWL required), both approaches generalise to the application of arbitrary rule-based reasoning, where the Context-Dependent framework – a means to manage contexts – generalises further to any form of deductive (or even inductive) reasoning process, as required.

Comparison of Both Approaches. In terms of the differences between both approaches, the Context-Dependent approach is designed to run over small contexts, typically involving one “assertional” document and its recursive ontology imports. Although the framework can handle aggregate contexts, the larger these aggregate contexts become, the closer Context-Dependent reasoning resembles the naïve case of standard reasoning over a monolithic graph. Thus, Context-Dependent reasoning is not well-suited to deriving entailments across assertional documents. The T-Box generated during Authoritative reasoning can be used to cautiously derive entailments across assertional documents (effectively reflecting a common consensus for a T-Box across all contexts); however, in practice, to achieve scalability, the A-Linear profile disables such inferences.

Conversely, Context-Dependent reasoning trusts all axioms in a local context, whereas Authoritative reasoning does not. In other words, Context-Dependent reasoning allows non-authoritative reasoning within contexts, which Authoritative reasoning never allows. With reference to Example 9, if a document imports the DBpedia ontology involved, Context-Dependent reasoning will permit translating `foaf:Person` instances into `dbo:Person` instances, whereas Authoritative reasoning will not.

Support for same-as? A primary limitation common to both approaches is the inability to effectively reason over `owl:sameAs` relations. Context-Dependent reasoning can only process such relations with a single context, which will miss the bulk of equivalence relations between assertional documents. In theory, Authoritative reasoning can support `owl:sameAs` inferences, but for scalability reasons, rules with A-Box joins are disabled in the SAOR implementation. However, in other more focussed works, we have looked at specialised methods for authoritative reasoning of `owl:sameAs` relations in a Linked Data setting [42].

Indeed, `owl:sameAs` can produce huge volumes of inferences: in previous work [42], we found 33,052 equivalent terms within a single (correct) `owl:sameAs` clique, which would require $33,052^2 = 1,092,434,704$ triples just to materialise the pair-wise and reflexive `owl:sameAs` relations between terms in this one group, even before applying any of the `eq-rep-*` rules for replacement. Given the importance of `owl:sameAs` reasoning for aligning entities in Linked Data, the

potential expense of such reasoning, and given that equivalence relations cannot be universally trusted on the Web [29], a number of works have tackled this issue with specialised techniques and optimisations [42,44,68]. For example, most systems supporting `owl:sameAs` reasoning at large scale use a single canonical identifier to represent each set of equivalent identifiers, avoiding the explosion of data that could otherwise occur [39,42,68,11]. In previous work, we applied authoritative reasoning to compute `owl:sameAs` relations from functional and inverse-functional properties and cardinality restrictions [42].

Interestingly, Hu et al. [42] investigate a notion of authority for `owl:sameAs` inferencing, assigning a level of trust to such a relation based on whether the given document is authoritative for the subject or object or both of a same-as relation (here applying authority on an A-Box level). In any case, we note that `owl:sameAs` is an important reasoning feature in the context of Linked Data, but similarly requires specialised techniques – that go beyond a generic reasoning framework – to handle effectively in scalable, real-world settings.

6 Enriching Link-Traversal Based Querying of Linked Data by Reasoning

As discussed previously, data-warehousing approaches – such as those introduced in the previous section – are not well suited for reasoning and querying over highly dynamic Linked Data. Content replicated in local indexes will quickly become out-of-date with respect to the current version of the respective sources on the Web. However, we referred in the introduction to the vision of the Web of Data itself as being a giant database spanning the Web, where certain types of queries can be posed and executed directly over the sources it contains. Such an approach for executing SPARQL queries directly over the Web of Data – called Link Traversal Based Query Execution (LTBQE) – was first proposed by Hartig et al. [33] (§ 6.1). However, the original approach did not offer any reasoning capabilities; indeed, no existing reasoning approaches at the time would seem suitable for such a scenario.

In this section, we first describe the LTBQE algorithm (a comprehensive study of the semantics and computability of LTBQE has been covered in [32]), complete with formal definitions and illustrative examples, motivate why RDFS/OWL reasoning is useful in such a setting, and then discuss methods we have ourselves since proposed to support such reasoning features.

6.1 Overview of Baseline LTBQE

Given a SPARQL query, the core operation of LTBQE is to identify and retrieve a focused set of query-relevant RDF documents from the Web of Data from which answers can be extracted. The approach begins by dereferencing URIs found in the query itself. The documents that are returned are parsed, and triples matching patterns of the query are processed; the URIs in these triples are also dereferenced to look for further information, and so forth. The process is recursive up to a fixpoint wherein no new query-relevant sources are found.

New answers for the query can be computed on-the-fly as new sources arrive. We now formally define the key notion of query-relevant documents in the context of LTBQE, and give an indication as to how these documents are derived.²⁹

Definition 12 (Query Relevant Sources & Answers). *First let $\text{uris}(\mu) := \{u \in \mathbf{U} \mid \exists v \text{ s.t. } (v, u) \in \mu\}$ denote the set of URIs in a solution mapping μ . Given a query Q and an intermediate dataset Γ , we define the function qrel , which extracts from Γ a set of URIs that can (potentially) be dereferenced to find further sources deemed relevant for Q :*

$$\text{qrel}(Q, \Gamma) := \bigcup_{tp \in Q} \bigcup_{\mu \in \llbracket \{tp\} \rrbracket_{\Gamma}} \text{uris}(\mu)$$

To begin the recursive process of finding query-relevant sources, LTBQE takes URIs in the query—denoted with $U_Q := \text{terms}(Q) \cap \mathbf{U}$ —as “seeds”, and builds an initial dataset by dereferencing these URIs: $\Gamma_0^Q := \text{derefs}(U_Q)$. Thereafter, for $i \in \mathbb{N}$, define:³⁰

$$\Gamma_{i+1}^Q := \text{derefs}(\text{qrel}(Q, \Gamma_i^Q)) \cup \Gamma_i^Q$$

The set of LTBQE query relevant sources for Q is given as the least n such that $\Gamma_n^Q = \Gamma_{n+1}^Q$, denoted simply Γ^Q . The set of LTBQE query answers for Q is given as $\llbracket Q \rrbracket_{\Gamma^Q}$, or simply denoted $\llbracket Q \rrbracket$.

Example 11

We illustrate this core concept of LTBQE query-relevant sources with a simple example based on Fig. 2. Let us consider our example Query 3.

First, the process extracts all raw query URIs:

$$U_Q = \{\text{nyt:4958--}, \text{nytimes:latest_use}, \text{owl:sameAs}, \text{dbo:revenueUSD}\}$$

and the engine dereferences these URIs. Second, LTBQE looks to extract additional query relevant URIs by seeing if any query patterns are matched in the current dataset. LTBQE repeats the above process until no new sources are found. When no other query-relevant URIs are found, a fix-point is reached and the process terminates with the results given over the retrieved “query-relevant documents”. \diamond

6.2 (In)Completeness of LTBQE

An open question is the *decidability* of collecting query-relevant sources: does it always terminate? This is dependent on whether one considers the Web of

²⁹ This is similar in principle to the generic notion of reachability introduced previously [34,32], but relies here on concrete HTTP specific operations.

³⁰ In practice, URIs need only be dereferenced once; *i.e.*, only URIs in $\text{qrel}(Q, \Gamma_i^Q) \setminus (\text{qrel}(Q, \Gamma_{i-1}^Q) \cup U_Q)$ need be dereferenced at each stage.

Data to be infinite or finite. For an infinite Web of Data, this process is indeed undecidable [32]. To illustrate this case, Hartig [32] uses the example of a Linked Data server describing all natural numbers³¹, where each $n \in \mathbb{N}$ is given a dereferenceable URI, each n has a link to $n + 1$ with the predicate `ex:next`, and a query with the pattern “`?n ex:next ?np1 .`” is given. In this case, the traversal of query-relevant sources will span the set of all natural numbers. However, if the (potential) Web of Data is finite, then LTBQE is decidable; in theory, it will terminate after processing all sources. The question of whether the Web (of Data) is infinite or not comes down to whether the set of URIs is infinite or not: though they may be infinite in theory [8] (individual URIs have no upper bound for length), they are finite in practice (machines can only process URIs up to some fixed length).³²

Of course, this is a somewhat academic distinction. In practice, the Web of Data is sufficiently large that LTBQE may end up traversing an infeasibly large number of documents before terminating. A simple worst case would be a query with an “open pattern” consisting of three variables.

Example 12

The following query asks for data on the founders of `dbr:SAP_AG`:

```
SELECT ?s ?p ?o WHERE {
  dbr:SAP_AG dbo:foundedBy ?s .
  ?s ?p ?o .
}
```

The first query-relevant sources will be identified as the documents dereferenced from `dbr:SAP_AG` and `dbo:foundedBy`. Thereafter, all triples in these documents will match the open pattern, and thus all URIs in these documents will be considered as potential query-relevant links. This will continue recursively, crawling the entire Web of Data. Of course, this problem does not occur only for open patterns. One could also consider the following query which asks for the friends of the founders of `dbr:SAP_AG`:

```
SELECT ?o WHERE {
  dbr:SAP_AG dbo:foundedBy ?s .
  ?s foaf:knows ?o .
}
```

This would end up crawling the connected Web of FOAF documents, as are linked together by dereferenceable `foaf:knows` links. ◇

³¹ Such a server has been made available by Vrandečić *et al.* [71], but unfortunately stops just shy of a billion. See, *e.g.*,

<http://km.aifb.kit.edu/projects/numbers/web/n42>.

³² It is not clear if URIs are (theoretically) finite strings. If so, they are countable [32].

Partly addressing this problem, Hartig *et al.* [33] defined an iterator-based execution model for LTBQE, which rather approximates the answers provided by Definition 12. This execution model defines an ordering of triple patterns in the query, similar to standard nested-loop join evaluation. The most selective patterns (those expected to return the fewest bindings) are executed first and initial bindings are propagated to bindings further up the tree. Crucially, later triple patterns are partially bound when looking for query-relevant sources. Thus, taking the previous example, the pattern “`?s foaf:knows ?o .`” will never be used to find query-relevant sources, but rather partially-bound patterns like “`dbr:Werner_Von_Siemens foaf:knows ?o .`” will be used. As such, instead of retrieving all possible query-relevant sources, the iterator-based execution model uses interim results to apply a more focused traversal of the Web of Data. This also makes the iterator-based implementation order-dependent: results may vary depending on which patterns are executed first and thus answers may be missed. However, it does solve the problem of traversing too many sources when low-selectivity patterns are present in the query.

Whether defined in an order-dependent or order-independent fashion, LTBQE will often not return complete answers with respect to the Web of Data [32]. We now enumerate some of the potential reasons LTBQE can miss answers.

Example 13

No dereferenceable query URIs: The LTBQE approach cannot return results in cases where the query does not contain dereferenceable URIs. For example, consider posing the following query against Fig. 2:

```
SELECT ?s ?p WHERE {
  ?s ?p nytimes:nytd_org .
}
```

As previously explained, the URI `nytimes:nytd_org` is not dereferenceable ($\text{deref}(\text{nytimes:nytd_org}) = \emptyset$) and thus, the query processor cannot compute and select relevant sources from interim results. \diamond

Example 14

Unconnected query-relevant documents: Similar to the previous case of reachability, the number of results might be affected if query relevant documents cannot be reached. This is the case if answers are “connected” by literals, blank-nodes or non-dereferenceable URIs. In such situations, the query engine cannot discover and dereference further query relevant data. The following query illustrates such a case:

```
SELECT ?comp ?name WHERE {
  dbr:SAP_AG foaf:name ?name .
  ?comp skos:prefLabel ?name .
}
```

Answers (other than `dbr:SAP_AG`) cannot be reached from the starting URI `dbr:SAP_AG` because the relevant documents are connected together by the literal "SAP AG", which cannot be traversed as a HTTP link. ◇

Example 15

Dereferencing partial information: In the general case, the effectiveness of LTBQE is heavily dependent on the amount of data returned by the `deref(u)` function. In an ideal case, dereferencing a URI u would return all triples mentioning u on the Web of Data. However, this is not always the case; for example:

```
SELECT ?s WHERE {
  ?s owl:sameAs dbr:SAP_AG .
}
```

This quite simple query cannot be answered by link-traversal techniques since the triple “`nyt:75293219995342479362 owl:sameAs dbr:SAP_AG .`” is not accessible by dereferencing `dbr:SAP_AG` or `owl:sameAs`. ◇

The assumption that all RDF available on the Web of Data about a URI u can be collected by dereferencing u is clearly idealised; hence, later in Section 6.4 we will empirically analyse how much the assumption holds in practice, giving insights into the potential recall of LTBQE on an infrastructural level.

6.3 LiDaQ: Extending LTBQE with Reasoning

Partly addressing some of the shortcomings of the LTBQE approach in terms of completeness (or, perhaps more fittingly, *recall*), Hartig *et al.* [33] proposed an extension of the set of query relevant sources to consider `rdfs:seeAlso` links, which sometimes overcomes the issue of URIs not being dereferenceable (as per `nytimes:nytd_org` in our example).

On top of this extension, we previously proposed a system called “LiDaQ” that extends the baseline LTBQE approach with components that leverage lightweight RDFS and `owl:sameAs` reasoning in order to improve recall. Formal

definitions of the extensions we propose are available in our paper describing LiDaQ [67]. Here we rather sketch our proposals and provide intuitive examples.

Considering owl:sameAs Links and Inferences. First, we propose following owl:sameAs links, which, in a Linked Data environment, are used to state that more information about the given resource can be found elsewhere under the target URI. Thus, to fully leverage owl:sameAs information, we first propose to follow relevant owl:sameAs links when gathering query-relevant sources and subsequently apply owl:sameAs reasoning, which supports the semantics of *replacement* for equality, meaning that information about equivalent resources is mapped to all available identifiers and made available for query answering. We illustrate the need for such an extension with the following example:

Example 16

Consider the following query asking for the revenue(s) of the company identified by the URI `nyt:75293219995342479362`.

```
SELECT ?rev WHERE {
  nyt:75293219995342479362 dbo:revenueEUR ?rev .
}
```

When applying this query over the data in Fig. 2, the owl:sameAs relationship between `nyt:75293219995342479362` and `dbr:SAP_AG` states that both URIs are equivalent and referring to the same real world entity, and hence that the information for one applies to the other. Hence, the revenue associated with `dbr:SAP_AG` should be returned as an answer according to OWL semantics. However, the baseline LTBQE approach will not return any answers since such equality relations are not considered. In summary, to answer this query, LTBQE must be extended to follow owl:sameAs links and apply reasoning to materialise inferences with respect to the semantics of replacement. ◇

To return answers for such examples, LTBQE needs to be extended to follow owl:sameAs links and apply reasoning. Thus, the set of query-relevant sources is extended to also consider documents dereferenced by looking up URIs that are equivalent to query-relevant URIs (such as `dbr:SAP_AG` in the previous example) and subsequently applying the *Same-As* subset of OWL 2 RL/RDF rules given in Table 3 over the merge of data, which performs replacement for equivalent terms related through owl:sameAs.

Considering RDFS Inferences. Second, we can extend LTBQE to consider some lightweight RDFS reasoning, which takes schema-level information from pertinent vocabularies and ontologies that describe the semantics of class and

property terms used in the query-relevant data and uses it to infer new knowledge. We motivate the need for RDFS reasoning with another straightforward example:

Example 17

Consider the following query asking for the label(s) of the company identified by the URI `dbr:IBM`.

```
SELECT ?label WHERE {
  dbr:IBM rdfs:label ?label .
}
```

When applying this query over the data in Fig. 2, baseline LTBQE will return the answer “IBM”. However, from the schema data (right-hand side of the example), we can see that the `foaf:name` property is defined in RDFS as a sub-property of `rdfs:label`. Hence, under RDFS semantics, we should also get “International Business Machines Corporation” as an additional answer. As such, considering RDFS inferencing can help find more answers under the LTBQE approach. ◇

Thus we can extend LTBQE to apply further reasoning and generate more answers. Although our LiDaQ proposal only considers the RDFS rules enumerated in Table 3, the approach can be generalised to other more comprehensive rule-sets, such as OWL 2 RL/RDF.

As a first step, we must make (RDF) schema data available to the query engine, where we propose three mechanisms:

1. a static collection of schema data are made available as input to the engine (e.g., the schema data from Fig. 2 are made available offline to the engine);
2. the properties and classes mentioned in the query-relevant sources are dereferenced to dynamically build a direct collection of schema data (e.g., since mentioned in the `dbr:IBM` query-relevant document, `foaf:name` is dereferenced to get the schema data at runtime); and
3. the direct collection of dynamic schema data is expanded by recursively following links on a schema level (e.g., not only is `foaf:name` dereferenced, but `rdfs:label` is also recursively dereferenced from that document).

The first approach follows the proposals for Authoritative reasoning laid out in the previous section, whereas the latter two approaches follow a “live” version of proposals for Context-Dependent reasoning, where the second mechanism includes direct implicit imports and the third mechanism includes recursive implicit imports (as argued in that section, since `owl:imports` is quite rare within a Linked Data setting, we omit its support for brevity).

Using the schema data collected by one of these methods, in the second step, we apply rule-based RDFS reasoning to materialise inferences and make them available for query-answering. Thus we can achieve the types of answers missing from the example.

6.4 Benefit of LTBQE Reasoning Extensions

Taken together, the two proposed reasoning extensions for LTBQE allow any client with a Web connection to answer queries, such as given in Example 4, and retrieve a full set of answers fresh from the original sources, potentially spanning multiple domains. The client does not need to index the sources in question.

With respect to generalising the benefits of reasoning, we now briefly summarise results of an empirical study we conducted to examine how LTBQE and its extensions can be expected to perform in practice; details can be found in [67]. The study took a large crawl of the Web of Data (the BTC'11 corpus) as a sample and surveyed the ratio of all triples mentioning a URI in our corpus against those returned in the dereferenceable document of that URI; this is done for different triple positions. In addition, the study also looks at the comparative amount of raw data about individual resources considering (1) explicit, dereferenceable information; (2) including `rdfs:seeAlso` links [33]; (3) including `owl:sameAs` links and inferences; (4) including RDFS inferences with respect to a static schema.

The study reports that, in the general case, LTBQE works best when a subject URI is provided in a query-pattern, works adequately when only (non-class) object URIs are provided, but works poorly when it must rely on property URIs bound to the predicate position or class URIs bound to the object position. Furthermore, we found that `rdfs:seeAlso` links are not so common (found in 2% of cases) and do not significantly extend the raw data made available to LTBQE for query-answering. Conversely, `owl:sameAs` links are a bit more common (found in 16% of cases) and can increase the raw data made available for answering queries significantly (2.5×). Furthermore, RDFS reasoning often (81% of the time) increases the amount of available raw data by a significant amount (1.8×).

We also tested the effect of these extensions for running queries live over Linked Data. We generated a large set of queries intended to be answerable using LTBQE by means of random walks across dereferenceable URIs available in our crawl. We then enabled and disabled the various configurations of the extensions proposed and ran the queries live over Linked Data sources on the Web. Summarising the results, we found that adding reasoning support to LTBQE allows for finding additional answers when directly querying Linked Data, but also introduces significant overhead. In particular, proposals to dynamically traverse implicit imports at runtime generate a huge overhead. Our conclusions were that reasoning was possible in such a setting, but is only practicable for simple queries involving few documents. Again, an excellent example of the type of query well-supported by such an approach is Query 3' listed earlier.

7 Extending Query Rewriting Techniques by Attribute Equations for Linked Data

In this section, we are getting back to challenge C5 from the Introduction, that is, the claim that *Linked Data needs more than RDFS and OWL*. To justify this position, we return to our running example.

Example 18

We already pointed to the example of Query 1 from p. 101, where it may be considered quite unintuitive that IBM's revenue is not returned. Given the exchange rate between EUR and USD (1 EUR = 1.30 USD as of 25 March, 2013), the value for `dbo:revenueEUR` should be computable from a value for `dbo:revenueUSD` and vice versa. In general, many numerical properties are related, not by `rdfs:subPropertyOf` relations or anything expressible in RDFS/OWL, but rather by the simple mathematical equations such as, for instance:

$$\text{revenueUSD} = \text{revenueEUR} * 1.3 \quad (1)$$

$$\text{profitEUR} = \text{revenueEUR} - \text{totalExpensesEUR} \quad (2)$$

◇

While such equations are not expressible in RDFS or OWL itself, lots of emerging Linked Data is composed of interdependent numerical properties assigned to resources. Lots of implicit information would be expressible in the form of such simple mathematical equations modelling these interdependencies. These dependencies include simple conversions, e.g., between currencies as in (1), or functional dependencies between multiple properties, such as exemplified in (2).

In this section we present an approach to extend RDFS and OWL by so-called attribute equations as part of the terminological knowledge in order to enable inclusion of additional numerical knowledge in the reasoning processes for integrating Linked Data. While an exhaustive discussion of the idea of attribute equations in all depth is beyond the scope of this paper, we refer the interested reader to [10] for more details.

7.1 Extending Ontologies by Attribute Equations

Attribute equations in [10] allow a very restricted form of simple numerical equations in multiple variables as follows.

Definition 13. Let $\{x_1, \dots, x_n\}$ be a set of variables. A simple equation E is an algebraic equation of the form $x_1 = f(x_2, \dots, x_n)$ such that $f(x_2, \dots, x_n)$

is an arithmetic expression over numerical constants and variables x_2, \dots, x_n where f uses the elementary algebraic operators $+$, $-$, \cdot , \div and contains each x_i exactly once. $\text{vars}(E)$ is the set of variables $\{x_1, \dots, x_n\}$ appearing in E .

That is, we allow non-polynomials for f – since divisions are permitted – but do not allow exponents (different from ± 1) for any variable; the idea here is that such equations can be solved uniquely for each x_i by only applying elementary transformations, assuming that all x_j for $j \neq i$ are known: i.e., for each x_i , such that $2 \leq i \leq n$, an equivalent equation E' of the form $x_i = f'(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$ is uniquely determined. Note that since each variable occurs only once and standard procedure for solving single variable equations can be used, we write $\text{solve}(x_1 = f(x_2, \dots, x_n), x_i)$ to denote E' .³³

In order to enable semantic support within OWL and RDFS for such simple equations where attribute URIs will be used as variable names, we need a syntactic representation. To this end, in addition to DL axioms encoded in the `rdfs:` and `owl:` vocabularies, we propose a new property (e.g., extending the `rdfs:` vocabulary) `rdfs:definedByEquation` to encode so called equation axioms. That is, we extend axioms as in Table 1 as follows:

Table 7. Mapping equation axioms to RDF

DL	RDFS
$P_0 = f(P_1, \dots, P_n)$	<code>P_0 rdfs:definedByEquation "f(P_1, ..., P_n)"</code>

Here, P_1, \dots, P_n are URIs of numerical properties (which we also call *attributes*); we write the respective arithmetic expressions $f(P_1, \dots, P_n)$ as plain literals in this terse encoding (instead of, e.g., breaking down the arithmetic expressions into RDF triples).

Example 19

The RDF encodings of (1) and (2) are

```
dbo:revenueUSD rdfs:definedByEquation "dbo:revenueEUR * 1.3" .
dbo:profitEUR rdfs:definedByEquation
    "dbo:revenueEUR - dbo:totalExpensesEUR" .
```

◇

³³ ... with analogy to notation used by computer algebra systems (such as Mathematica, cf. <http://www.wolfram.com/mathematica/>, or Maxima, cf. <http://maxima.sourceforge.net>).

As mentioned before in the context of Definition 13, we consider equations that result from just applying elementary transformations as equivalent. In order to define the semantics of equation axioms accordingly, we will make use of the following definition.

Definition 14. Let $E: P_0 = f(P_1, \dots, P_n)$ be an equation axiom then, for any P_i with $0 \leq i \leq n$ we call the equation axiom $\text{solve}(E, P_i)$ the P_i -variant of E .

As for the details on the formal semantics of attribute equations we refer the reader again to [10] and directly jump to two potential ways to implement reasoning and querying with such equations.

In principle, there are the same two potential ways to implement reasoning with property equations as already discussed in the context of RDFS and OWL in Section 2.4: rule-based inference and query rewriting. As we will see, the main problem in the context of attribute equations is that both of these approaches, when extended straightforwardly, would potentially not terminate. In the following, we present both of these implementation approaches and discuss their pros and cons.

7.2 Implementing Attribute Equations within Rules

Many rule-based approaches such as SWRL [43] offer additional support for mathematical built-ins.

Example 20

Using SWRL, (1) could be encoded as follows:

$$(?X, \text{dbo:revenueUSD}, ?USD) \leftarrow (?X, \text{dbo:revenueEUR}, ?EUR), ?USD = ?EUR * 1.3 \quad (3)$$

$$\begin{aligned} (?X, \text{dbo:profitEUR}, ?PEUR) &\leftarrow (?X, \text{dbo:revenueEUR}, ?REUR), \\ & (?X, \text{dbo:totalExpensesEUR}, ?TEUR), \\ & ?PEUR = ?REUR - ?TEUR \end{aligned} \quad (4)$$

◇

However, note that rules as exemplified above are not sufficient: (i) rule (3) is in the “wrong direction” for inferring additional results necessary for Query 1, that is, we would need different variants of the rule for converting from *EUR* to *USD* and vice versa; (ii) the above rules are not *DL safe* (i.e., we want to go beyond binding values only to explicitly named individuals where we also want to compute *new* values) which potentially leads to termination problems in rule-based approaches (and it actually does in existing systems).

Problem (i) could be solved in some cases by simply adding additional rules for each variant of each equation axiom, but the extended ruleset will, in turn, often give rise to Problem (ii), as shown.

Example 21

For the previous example, we can add more SWRL rules as follows:

$$(?X, \text{dbo:revenueEUR}, ?EUR) \leftarrow (?X, \text{dbo:revenueUSD}, ?USD), \quad (5)$$

$$?EUR = ?USD / 1.3$$

$$(?X, \text{dbo:revenueEUR}, ?REUR) \leftarrow (?X, \text{dbo:profitEUR}, ?PEUR), \quad (6)$$

$$(?X, \text{dbo:totalExpensesEUR}, ?TEUR),$$

$$?REUR = ?PEUR + ?TEUR$$

$$(?X, \text{dbo:totalExpensesEUR}, ?TEUR) \leftarrow (?X, \text{dbo:revenueEUR}, ?REUR), \quad (7)$$

$$(?X, \text{dbo:profitEUR}, ?PEUR),$$

$$?TEUR = ?REUR - ?PEUR$$

However, here problem (ii) comes into play, as it is easy to see that these rules potentially produce infinite results, which we leave as an exercise to the reader. As a hint, consider the following example data:

```
:company1 dbo:profitEUR 1;
          dbo:revenueEUR 1;
          dbo:totalExpensesEUR 1;
```

Obviously, this data is not coherent with the equation, in the sense that it is ambiguous and a rule-engine that tries to compute the closure would not terminate (such example could occur in reality due to, e.g., rounding errors). \diamond

Certain rule engines provide special built-ins to avoid running into non-termination problems as exemplified above. For instance, Jena provides a special built-in `noValue`, which returns sound but incomplete results whereby it only fires a rule if no value exists for a certain attribute on the inferences thus far or in the data – not unlike negation-as-failure.

Example 22

Using the `noValue` built-in, rule (4) (and analogously the other rule variants) could be encoded in Jena's rule syntax as follows:

```
[ (?X dbo:revenueEUR ?REUR) (?X dbo:totalExpensesEUR ?TEUR)
  difference(?REUR, ?TEUR, ?PEUR) noValue(?X, dbo:profitEUR)
  -> (?X dbo:profitEUR ?PEUR)]
```

Values for `?PEUR` will only be computed from the given equations if no such value for `dbo:profitEUR` already exists on the resource bound to `?X`. \diamond

7.3 Implementing Attribute Equations by Query Rewriting

An alternative implementation approach for reasoning with attribute equations (which according to initial experiments in [10] seems to work better than rules-based materialisation) is based on query rewriting – essentially extending Algorithm 1 from p. 106.

The idea here is that the expansion step in line 8 of Algorithm 1 is extended to also work with equation axioms as per Table 7. That is, informally, we extend the expansion function $gr(g, i)$ from Table 2 as follows:

g	i	$gr(g/i)$
(x, P_0, y)	$P_0 =; f(P_1, \dots, P_n)$	$(x, P_1, ?V_{P_1}), \dots, (x, P_n, ?V_{P_n}), A = f(?V_{P_1}, \dots, ?V_{P_n})$

where we consider any equation axiom that has a P_0 -variant. Similar to the rule-based approach, special care has to be taken such that equation axioms are not expanded infinitely; to this end, a simple blocking condition in the variant of Algorithm 1 presented in [10] avoids that the same equation axiom is used twice to compute the same value.

Example 23

To illustrate the approach, let us take a variation of Query 1 as an example, which only asks for the revenues of organisations, i.e., the SPARQL Query:

```
SELECT ?X ?R
WHERE { ?X a dbo:Organisation; dbo:revenueEUR ?R . }
```

We start with its formulation as a conjunctive query

$$q(?X, ?R) \leftarrow (?X, a, \text{dbo:Organisation}), (?X, \text{dbo:revenueEUR}, ?R) \quad (8)$$

which is expanded as follows:

$$q(?X, ?R) \leftarrow (?X, a, \text{dbo:Organisation}), (?X, \text{dbo:revenueEUR}, ?R) \quad (9)$$

$$q(?X, ?R) \leftarrow (?X, a, \text{dbo:Company}), (?X, \text{dbo:revenueEUR}, ?R) \quad (10)$$

$$q(?X, ?R) \leftarrow (?X, a, \text{dbo:Organisation}), (?X, \text{dbo:revenueUSD}, ?V_{\text{revenueUSD}}), \quad (11)$$

$$?R = ?V_{\text{revenueUSD}}/1.3$$

$$q(?X, ?R) \leftarrow (?X, a, \text{dbo:Company}), (?X, \text{dbo:revenueUSD}, ?V_{\text{revenueUSD}}), \quad (12)$$

$$?R = ?V_{\text{revenueUSD}}/1.3$$

◇

We note that translation back to SPARQL is not as straightforward here as it was without attribute equations, due to the fact that, as opposed to UCQs over only RDF triple predicates, we now are dealing with UCQs that also involve equality predicates and arithmetic operations such as $?R = ?V_{\text{revenueUSD}}/1.3$ in (11) and (12). Unions are again (like in Example 3) translated back to UNION

patterns in SPARQL, whereas equalities in query bodies are translated – depending on whether the left-hand side of these equalities is a variable or a constant – to either a BIND pattern³⁴, or a FILTER pattern.

Example 24

Following the previous example, this rewritten SPARQL 1.1 query will return the revenues of all three companies in our example data in EUR:

```
SELECT ?X ?R
WHERE { { ?X a dbo:Organisation; dbo:revenueEUR ?R .}
        UNION { ?X a dbo:Company; dbo:revenueEUR ?R .}
        UNION { ?X a dbo:Organisation; dbo:revenueUSD ?V_revenueUSD .
                BIND ( ?V_revenueUSD / 1.3 AS ?R ) }
        UNION { ?X a dbo:Company; dbo:revenueUSD ?V_revenueUSD .
                BIND ( ?V_revenueUSD / 1.3 AS ?R ) } }
```



What we would like to emphasise here then, is that RDFS and OWL may not be enough for the reasoning requirements of Linked Data, where we show how (and why), e.g., numerical data can also be axiomatised for reasoning.

8 Summary

In this lecture we have illustrated particular challenges, opportunities and obstacles for applying OWL and RDFS reasoning in the context of querying Linked Data. We discussed the use of the RDFS and OWL standards in the area of Linked Data publishing, showing the degree to which individual features have been adopted on the Web. Though our results fall well short of indicating universal adoption, encouragingly, we find that many “lightweight” features of OWL and in particular RDFS have been widely adopted. We also provided practical examples as to how these RDFS and OWL axioms embedded in Linked Data can help for querying diverse sources, and how reasoning can thus help to further realise the vision of the Web of Data as one giant database.

However, while reasoning helps to obtain additional useful results in many cases, caution is required and specifically tailored reasoning algorithms need to be applied. In Sections 5–6 we have presented such tailored reasoning approaches and discussed their pros and cons; none of these approaches provides a panacea for reasoning “in the wild”, the right approach depends a lot on the use case, particularly on the datasets considered and the query at hand. Still, the presented approaches have demonstrated that lightweight reasoning in

³⁴ BIND patterns are a new feature in SPARQL1.1 to assign values from an expression to a variable, see [30].

diverse Linked Data setting is not only useful, but possible in practice, despite the enumerated challenges relating to scale, fallible data, inconsistencies, etc.

On the other hand, for modelling certain integration scenarios in Linked Data, we have shown that OWL and RDFS alone do not suffice to model a lot of implicit information and have briefly discussed attribute equations as an extension of OWL; given the increasing amount of published numerical data in RDF on the emerging Web of data, we believe that this topic deserves increased attention within the Semantic Web reasoning community. Generalising, though we have shown RDFS and OWL reasoning to be useful for querying Linked Data, these standards only provide a *basis* – not a solution – for capturing the semantics of Linked Data and still fall far short of that required to properly realise the vision of the Web of Data as a global, integrated database.

Acknowledgements. The work presented herein has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2), by an IRCSET Scholarship. We thank our co-authors on joint works that flowed into this article, namely, Stefan Bischof, Piero Bonatti, Stefan Decker, Birte Glimm, Andreas Harth, Markus Krötzsch, Jeff Z. Pan, Luigi Sauro and Giovanni Tumarello.

References

1. Allemang, D., Hendler, J.A.: *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann/Elsevier (2008)
2. Auer, S., Lehmann, J., Ngonga Ngomo, A.-C., Zaveri, A.: Introduction to Linked Data and its Lifecycle on the Web. In: Rudolph, S., Gottlob, G., Horrocks, I., van Harmelen, F. (eds.) *Reasoning Web 2013*. LNCS, vol. 8067, pp. 1–90. Springer, Heidelberg (2013)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: *The Description Logic Handbook: Theory, Implementation and Application*. Cambridge University Press (2002)
4. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: *OWL Web Ontology Language Reference*. W3C Recommendation, W3C (February 2004)
5. Beckett, D., Berners-Lee, T.: *Turtle – Terse RDF Triple Language*. W3C Team Submission (January 2008), <http://www.w3.org/TeamSubmission/turtle/>
6. Beckett, D., Berners-Lee, T., Prud'hommeaux, E., Carothers, G.: *Turtle – Terse RDF Triple Language*. W3C Candidate Recommendation (February 2013), <http://www.w3.org/TR/2013/CR-turtle-20130219/>
7. Berners-Lee, T.: *Linked Data*. W3C Design Issues (July 2006), <http://www.w3.org/DesignIssues/LinkedData.html> (retr. October 27, 2010)
8. Berners-Lee, T., Fielding, R.T., Masinter, L.: *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986 (January 2005), <http://tools.ietf.org/html/rfc3986>
9. Berners-Lee, T., Fischetti, M.: *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper (1999)
10. Bischof, S., Polleres, A.: RDFS with attribute equations via SPARQL rewriting. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) *ESWC 2013*. LNCS, vol. 7882, pp. 335–350. Springer, Heidelberg (2013)

11. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: OWLIM: A family of scalable semantic repositories. *Semantic Web* 2(1), 33–42 (2011)
12. Bonatti, P.A., Hogan, A., Polleres, A., Sauro, L.: Robust and scalable Linked Data reasoning incorporating provenance and trust annotations. *J. Web Sem.* 9(2), 165–201 (2011)
13. Brickley, D., Guha, R.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation (February 2004), <http://www.w3.org/TR/rdf-schema/>
14. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning* 39(3), 385–429 (2007)
15. Cheng, G., Qu, Y.: Integrating lightweight reasoning into class-based query refinement for object search. In: Domingue, J., Anutariya, C. (eds.) ASWC 2008. LNCS, vol. 5367, pp. 449–463. Springer, Heidelberg (2008)
16. de Bruijn, J., Heymans, S.: Logical foundations of (e)RDF(S): Complexity and reasoning. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 86–99. Springer, Heidelberg (2007)
17. Delbru, R., Tummarello, G., Polleres, A.: Context-dependent OWL reasoning in Sindice – experiences and lessons learnt. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 46–60. Springer, Heidelberg (2011)
18. Emmons, I., Collier, S., Garlapati, M., Dean, M.: RDF literal data types in practice. In: Proceedings of Workshop on Scalable Semantic Web Systems (SSWS). LNCS, vol. 5947, Springer (2011)
19. Fielding, R.T., Gettys, J., Mogul, J.C., Frystyk, H., Masinter, L., Leach, P.J., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (June 1999), <http://www.ietf.org/rfc/rfc2616.txt>
20. Fischer, F., Unel, G., Bishop, B., Fensel, D.: Towards a scalable, pragmatic knowledge representation language for the Web. In: Pnueli, A., Virbitskaite, I., Voronkov, A. (eds.) PSI 2009. LNCS, vol. 5947, pp. 124–134. Springer, Heidelberg (2010)
21. Glimm, B.: Using SPARQL with RDFS and OWL entailment. In: Polleres, A., d’Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 137–201. Springer, Heidelberg (2011)
22. Glimm, B., Hogan, A., Krötzsch, M., Polleres, A.: OWL: Yet to arrive on the Web of Data? In: LDOW, vol. 937. CEUR-WS.org (2012)
23. Glimm, B., Ogbuji, C.: SPARQL 1.1 Entailment Regimes. W3C Recommendation (March 2013), <http://www.w3.org/TR/sparql11-entailment/>
24. Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive datalog programs. In: 13th Int’l Conf. on Principles of Knowledge Representation and Reasoning (KR 2012), Rome, Italy. AAAI Press (2012)
25. Grau, B.C., Motik, B., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language: Profiles. W3C Recommendation (October 2009), <http://www.w3.org/TR/owl2-profiles/>
26. Grosz, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: WWW, pp. 48–57 (2003)
27. Guha, R.V.: Contexts: a formalization and some applications. PhD thesis, Stanford University, Stanford, CA, USA (1992)
28. Guha, R., McCool, R., Fikes, R.: Contexts for the Semantic Web. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 32–46. Springer, Heidelberg (2004)

29. Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: When `owl:sameAs` Isn't the Same: An Analysis of Identity in Linked Data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 305–320. Springer, Heidelberg (2010)
30. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. W3C Recommendation (March 2013), <http://www.w3.org/TR/sparql11-query/>
31. Harth, A., Kinsella, S., Decker, S.: Using Naming Authority to Rank Data and Ontologies for Web Search. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 277–292. Springer, Heidelberg (2009)
32. Hartig, O.: SPARQL for a Web of Linked Data: Semantics and computability. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 8–23. Springer, Heidelberg (2012)
33. Hartig, O., Bizer, C., Freytag, J.-C.: Executing SPARQL queries over the Web of Linked Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 293–309. Springer, Heidelberg (2009)
34. Hartig, O., Freytag, J.-C.: Foundations of traversal based query execution over linked data. In: HT, pp. 43–52. ACM (2012)
35. Hayes, P.: RDF Semantics. W3C Recommendation (February 2004), <http://www.w3.org/TR/rdf-mt/>
36. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space, 1st edn. Synthesis Lectures on the Semantic Web: Theory and Technology, vol. 1. Morgan & Claypool (2011), <http://linkeddatabook.com/editions/1.0/>
37. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 Web Ontology Language Primer. W3C Recommendation (October 2009), <http://www.w3.org/TR/owl2-primer/>
38. Hogan, A.: Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora. PhD thesis, Digital Enterprise Research Institute, National University of Ireland, Galway (2011), <http://aidanhogan.com/docs/thesis/>
39. Hogan, A., Harth, A., Decker, S.: Performing Object Consolidation on the Semantic Web Data Graph. In: 1st I3 Workshop: Identity, Identifiers, Identification Workshop. CEUR Workshop Proceedings, vol. 249. CEUR-WS.org (2007)
40. Hogan, A., Harth, A., Umbrich, J., Kinsella, S., Polleres, A., Decker, S.: Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine. *J. Web Sem.* 9(4), 365–401 (2011)
41. Hogan, A., Pan, J.Z., Polleres, A., Decker, S.: SAOR: Template Rule Optimisations for Distributed Reasoning over 1 Billion Linked Data Triples. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 337–353. Springer, Heidelberg (2010)
42. Hogan, A., Zimmermann, A., Umbrich, J., Polleres, A., Decker, S.: Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora. *J. Web Sem.* 10, 76–110 (2012)
43. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosz, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. W3C member submission, W3C (2004)
44. Hu, W., Qu, Y., Sun, X.: Bootstrapping object coreferencing on the Semantic Web. *J. Comput. Sci. Technol.* 26(4), 663–675 (2011)

45. Käfer, T., Abdelrahman, A., Umbrich, J., O'Byrne, P., Hogan, A.: Observing Linked Data dynamics. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) *ESWC 2013*. LNCS, vol. 7882, pp. 213–227. Springer, Heidelberg (2013)
46. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyashev, M.: The combined approach to ontology-based data access. In: *22nd Int'l Joint Conf. on Artificial Intelligence (IJCAI 2011)*, Barcelona, Catalonia, Spain, pp. 2656–2661. IJCAI/AAAI (2011)
47. Kontchakov, R., Rodríguez-Muro, M., Zakharyashev, M.: Ontology-based data access with databases: A short course. In: Rudolph, S., Gottlob, G., Horrocks, I., van Harmelen, F. (eds.) *Reasoning Web 2013*. LNCS, vol. 8067, pp. 194–229. Springer, Heidelberg (2013)
48. Mallea, A., Arenas, M., Hogan, A., Polleres, A.: On Blank Nodes. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) *ISWC 2011, Part I*. LNCS, vol. 7031, pp. 421–437. Springer, Heidelberg (2011)
49. Manola, F., Miller, E., McBride, B.: *RDF Primer*. W3C Recommendation (February 2004), <http://www.w3.org/TR/rdf-primer/>
50. Mayfield, J., Finin, T.: Information retrieval on the Semantic Web: Integrating inference and retrieval. In: *Proceedings of the SIGIR Workshop on the Semantic Web* (August 2003)
51. Miles, A., Baker, T., Swick, R.: Best Practice Recipes for Publishing RDF Vocabularies. W3C working group note, W3C (2008)
52. Motik, B., Patel-Schneider, P.F., Grau, B.C.: *OWL 2 Web Ontology Language Direct Semantics*. W3C Recommendation (October 2009), <http://www.w3.org/TR/owl2-direct-semantics/>
53. Muñoz, S., Pérez, J., Gutierrez, C.: Simple and Efficient Minimal RDFS. *J. Web Sem.* 7(3), 220–234 (2009)
54. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: *Sindice.com: a document-oriented lookup index for open linked data*. *IJMSO* 3(1), 37–52 (2008)
55. Page, L., Brin, S., Motwani, R., Winograd, T.: *The PageRank Citation Ranking: Bringing Order to the Web*. Technical report, Stanford Digital Library Technologies Project (1998)
56. Patel-Schneider, P.F., Motik, B., Cuenca Grau, B., Horrocks, I., Parsia, B., Ruttenberg, A., Schneider, M.: *OWL 2 Web Ontology Language: Mapping to RDF Graphs*. W3C Recommendation (October 2009), <http://www.w3.org/TR/owl2-mapping-to-rdf/>
57. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. *ACM Transactions on Database Systems* 34(3):Article 16 (45 pages) (2009)
58. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. *Journal of Applied Logic* 8(2), 186–209 (2010)
59. Polleres, A., Feier, C., Harth, A.: Rules with contextually scoped negation. In: Sure, Y., Domingue, J. (eds.) *ESWC 2006*. LNCS, vol. 4011, pp. 332–347. Springer, Heidelberg (2006)
60. Prud'hommeaux, E., Seaborne, A.: *SPARQL Query Language for RDF*. W3C Recommendation (January 2008), <http://www.w3.org/TR/rdf-sparql-query/>
61. Rosati, R.: Prexto: Query rewriting under extensional constraints in DL-Lite. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) *ESWC 2012*. LNCS, vol. 7295, pp. 360–374. Springer, Heidelberg (2012)

62. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: 12th Int'l Conf. on Principles of Knowledge Representation and Reasoning (KR 2010). AAAI Press (2010)
63. Rudolph, S.: Foundations of Description Logics. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 76–136. Springer, Heidelberg (2011)
64. Schneider, M.: OWL 2 Web Ontology Language RDF-Based Semantics. W3C Recommendation (October 2009), <http://www.w3.org/TR/owl2-rdf-based-semantics/>
65. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics* 3(2-3), 79–115 (2005)
66. Ullman, J.D.: Principles of Database and Knowledge Base Systems. Computer Science Press (1989)
67. Umbrich, J., Hogan, A., Polleres, A., Decker, S.: Improving the recall of live linked data querying through reasoning. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 188–204. Springer, Heidelberg (2012)
68. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.E.: WebPIE: A Web-scale Parallel Inference Engine using MapReduce. *J. Web Sem.* 10, 59–75 (2012)
69. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable distributed reasoning using MapReduce. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 634–649. Springer, Heidelberg (2009)
70. Volz, R.: Web Ontology Reasoning with Logic Databases. PhD thesis, AIFB, Karlsruhe, Germany (2004)
71. Vrandečić, D., Krötzsch, M., Rudolph, S., Lösch, U.: Leveraging Non-Lexical Knowledge for the Linked Open Data Web. *Review of Fool's day Transactions (RAFT)* 5, 18–27 (2010)
72. Weaver, J., Hendler, J.A.: Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 682–697. Springer, Heidelberg (2009)
73. Yardeni, E., Shapiro, E.Y.: A Type System for Logic Programs. *J. Log. Program.* 10(1/2/3&4), 125–153 (1991)

Introductions to Description Logics – A Guided Tour

Anni-Yasmin Turhan

Theoretical Computer Science,
TU Dresden, Germany
turhan@tcs.inf.tu-dresden.de

Abstract. Description Logics (DLs) are the logical formalism underlying the standard web ontology language OWL 2. DLs have formal semantics which are the basis for many powerful reasoning services. This paper provides an overview of basic topics in the field of Description Logics by surveying the introductory literature and course material with a focus on DL reasoning services. The resulting compilation also gives a historical perspective on DLs as a research area.

1 Introduction

Description Logics (DLs) are a family of knowledge representation formalisms that have formal semantics. This family of logics is designed towards representing terminological knowledge of an application domain in a structured and formally well-understood way. DLs allow users to define important notions, such as classes or relations of their application domain in terms of concepts and roles. *Concepts* correspond to unary predicates and *roles* correspond to binary predicates in First Order Logic (FOL). They restrict the interpretations of the classes and relations, respectively.

Starting from a set of concept names and role names, complex concept descriptions can be built by means of *concept constructors*. For instance, expressive DLs offer the Boolean connectors as concept constructors. Concept descriptions are the main building blocks for capturing information in the *knowledge base* (or *ontology*). Typically, a DL knowledge base consists of two parts:

- the *TBox*, which contains the terminological knowledge, i.e., the knowledge on the categories and relations relevant in the application domain and
- the *ABox*, which contains the assertional knowledge, i.e., the knowledge on individual facts.

Knowledge that is captured only implicitly in the ontology can be *inferred* from the given descriptions of concepts and roles and also from the information on the individuals in the ABox, as for instance, sub-class or instance relationships.

DLs have been investigated and used since the late eighties. Of main interest are the reasoning problems defined for DLs and the corresponding reasoning algorithms. Many courses and tutorials have been given on the subject since. Here, instead of providing yet another standard introduction on this branch of logics, we rather survey existing course material. Mostly, these courses were designed and held by people, who are active in research on the subjects covered in those courses. The set of papers, tutorials

and courses covered here is certainly only a part of the existing material. We focus on material that is available on-line at the time of writing.

This paper is structured as follows. In the next section we give an overview of introductory literature and tutorials on Description Logics in general. In Section 3 we give short historic overview, followed by a commented list of course material on basic reasoning and newer reasoning services in DLs in Section 4. The last section provides references to material on application areas of DLs.

2 Introductory Material to DLs in General

Introductory readings. The standard reference for DLs is certainly the DL handbook [5]. For the DL novice with a bit of a background in logics, the two introductory chapters [22] and [10] are a good starting point to get a detailed and slow paced introduction to the basic notions of DLs. For a short reference of basic DL terms see [1].

The chapters on DLs in the *Handbook of Modal Logic* [9] and in the *Handbook on Ontologies* [8] provide also detailed and self-contained introductions. While the first emphasizes more the theoretical aspects of DLs, the latter rather highlights practical aspects of using DLs. A more recent and comprehensive introduction is given in [2]. For readers who are more interested in what current DL systems can do and how to employ them (and not so much in the theoretical foundations) [50] is a good reference. The most up-to-date 'all purpose' beginner's introduction is the *DL primer* [64].

Introductory courses. Probably the most detailed on-line course on DLs are the slides from Enrico Franconi's course [40] from 2002. This course lays the foundations by giving an introduction to Computational Logics in general in Module 1 and provides an introduction to simple, i.e., rather inexpressive DLs (with only a few concept constructors) in the Modules 2, 3 and 4. Other classic courses on introductions to DLs that are rather suitable for the reader with a little knowledge on logics, are [67,90].

2.1 Relation to Other Logics

DL are logics and as such closely related to other formal logics. In particular, most DLs are a fragment of First Order Logic. Concepts are simply FOL formulas with one free variable. Some DLs are simply syntactic variants of Hybrid Logics or Modal Logics, see [87]. The correspondences between DLs and other logics are explained in detail in the DL handbook chapter dedicated to this topic [86] and in the already mentioned on-line course by Franconi [40](Module 5). Most introductory tutorials provide translation functions from DL knowledge bases into FOL, such as [2].

3 A Short Historical Overview

In this section we give an overview over the main developments of the DL research. The papers cited here are mainly the original research papers (and rather of interest for readers with DL background). Typically, the following historical phases of DL research are distinguished:

Early knowledge representation systems. Historically, DLs originate from knowledge representation systems such as *semantic networks* [82,93] or *frame systems* [72]. Despite the fact that these systems lacked formal semantics, they offered methods to compute inheritance relations between the specified notions.

Early DL systems. In the late eighties, reasoning algorithms for DL systems were mostly sound, but incomplete, i.e., they would compute correct answers, but not necessarily all correct answers. At this time the belief was held that terminological reasoning is inherently intractable [76,77], and thus completeness was traded for tractability. These reasoning algorithms have been implemented in systems such as Back [76,78] and Classic [19,18,21].

DLs in the nineties. During the nineties, sound and complete reasoning methods were investigated for the core inferences of DL systems: consistency and subsumption. *Consistency* assures that the specification of the concepts, roles and individuals are free of contradictions. For *subsumption* one computes super- and sub-concept relations from the given specifications of concepts (and roles).

The underlying technique for computing the basic DL inferences is the tableau method [39,89]. The core idea of the tableau method is to construct a model, which then gives evidence that a particular knowledge base has a model. The tableau method was extended to more and more expressive DLs ([12,31]). The gain in expressiveness came at the cost of higher complexity for the reasoning procedures—reasoning for the DLs investigated is PSpace-complete or even ExpTime-complete [31].

Despite the high complexity, highly optimized DL reasoning systems—most prominently the FACT system [48]—were implemented based on the tableau method [74]. In fact, it turned out that these highly optimized implementations of the reasoning methods do perform surprisingly well on DL knowledge bases from practical applications.

DLs in the new millennium. The quest for more expressive DLs with decidable reasoning procedures for standard reasoning continued—allowing for more information that can be stated on roles, such as inverse of roles, for instance, see [55,56,52,57]. Around that time first initiatives emerged for standardizing DLs (such as DAML+OIL, see [33]) and the reasoner interfaces (such as the on from the DL implementers group [16,98]). Based on these initiatives, the development of a variety of ontology tools started. Early ontology editors such as OilEd [15], OntoTrack [65] PROTÉGÉ [41,61] or Swoop [59] were developed and user communities of DL systems started to grow.

In the last decade there were two main trends in DL research. First, the investigation of so-called ‘light-weight’ DLs, i.e., DL that are of fairly low expressivity, but have good computational complexity for reasoning. There are two ‘families’ of lightweight DLs: the \mathcal{EL} family [23,3,4], for which the subsumption and the instance problem are polynomial, and the DL Lite family [27,29], for which the instance problem and the answering of (unions of) conjunctive queries are polynomial¹. A member of each of

¹ If measured in the size of the data alone the complexity is even LogSpace.

these two families is the DL corresponding to one of the profiles of the OWL 2 standard [99]. The second trend is, that various new, so-called non-standard inferences are investigated for DLs. For instance,

- the generation of *explanations* of unexpected consequences that the DL reasoner detected [88,81,11,58,46],
- *answering of conjunctive queries* as a means to access the instance data of an ontology, [75,28,29,43,79,38,66],
- support for building ontologies by computing *generalizations* [24,13,37,101], and
- computing *modularizations* of ontologies to facilitate their reuse [42,69,35,71,62].

4 DL Reasoning

4.1 Standard DL Reasoning

The standard reasoning problems for DLs, such as satisfiability or subsumption are discussed in the introductions to DLs mentioned in Section 2. A rather detailed discussion of the model theoretic properties of basic DLs (such as the moderately expressive DL \mathcal{ALC}) were recently given in [68,90].

Solutions to DL reasoning problems are computed by different reasoning procedures. As mentioned earlier, for expressive DLs tableaux-based procedures are common, see [31,12,67]. Reasoning in the \mathcal{EL} -family underlying the OWL 2 EL profile is realized by *completion-based* or *consequence-driven* approaches, which are covered in the courses [96,63].

Another approach to obtain decision procedures for DL reasoning problems is the *automata-based* approach. Automata-based approaches are often more convenient for showing ExpTime complexity upper-bounds than tableau-based approaches. This approach is described in detail in [6,67,2].

The computational complexity of deciding standard reasoning problems is discussed in [36] and more recently in [67,2,90].

The reader interested in the implementations of DL systems is referred to [74] for an early—by now almost historic—overview. Fairly recent accounts on this ever changing subject can be found in [73,51].

4.2 On Non-standard Reasoning Tasks

Building DL ontologies. The subject of ontology engineering is addressed, for example, in [60]. However, reasoning based approaches that either employ standard reasoning [73] or inferences that compute generalizations of either collections of (complex) concept descriptions or of an individual are described in [9,95].

Explanation and Repair. When a reasoner computes a consequence of information represented in the ontology, the result might be un-intuitive to the user. Thus computing explanations of (or even plans how to repair) such unexpected consequences are helpful. Both tasks have been discussed in the tutorial [47]. The first task is described for the \mathcal{EL} -family in [14] and the course [96]. Implementations for this task are described in [47,51].

Modularity. Re-using a part of an ontology requires to identify that part of the ontology that does not ‘interact’ with the rest of the ontology when computing a given reasoning task. Furthermore, it should be ensured that the importing ontology does not entail unwanted consequences w.r.t. the given reasoning task when importing the new ontology module. These tasks are topics of the course material provided in [47] (with an emphasis on tool support) and in [68,91] with an emphasis on the theoretical background.

Answering Conjunctive Queries. As mentioned earlier, answering conjunctive queries allows for a much more expressive query language than concept-based querying, i.e., instance queries. This reasoning task is currently a very active research area of DLs. The DL Lite family of DLs is designed such that conjunctive query answering can be performed efficiently. Query answering in DL Lite is covered in [26]. Methods for query answering for both families of light-weight DLs is described in the course slides [32]. In addition to query answering for the light-weight DLs, [80] explains also the methods for expressive DLs.

5 Application Areas of DLs

This section gives pointers introductory reading and course material on prominent application areas for DLs.

Data integration. Since DLs can represent information on different levels of detail they are a good candidate for data integration. The core of the data integration approach via DLs is their ability to capture other modeling languages frequently used to specify database schemas—such as entity relationship diagrams (ER), see [40], module 2 or UML, see [25].

Biomedical ontologies. An account on the early use of DLs in the medical field is given in [83]. In [100] an application in protein classifications described.

As a matter of FACT, the medical ontology GALEN [84] was the prime motivation for the development of highly optimized reasoners [49]. Since then large biomedical ontologies [94,34,85,92] have been valuable benchmarks for DL reasoners. More recent overviews on medical ontologies written in DLs is given in [70,50].

Semantic Web. The semantic web was early spotted as a potential application area of DLs, since they allow to write ontologies in order to annotate web resources, see [7,45]. More importantly, the reasoning services defined and investigated for DLs support the querying of these ontologies. From the plethora of course material on DLs and the semantic web, we recommend to the reader [7,45] for early views on the subject and [54,53] for more recent and technical ones.

The potential application of the semantic web facilitated the standardization of DLs. An introduction to the original OWL standard can be found in [54] and OWL 2 and its profiles is covered in [63,44,97].

Ontology-based data access. Ontology-based data access (ODBA) exploits DLs to enrich the data in a database by information from the DL ontology. The initial ideas were described in [17]. The key idea is to employ query answering for this task. In the last years the topic received more attention due to low complexity for DL Lite for this task. An early course on this subject is [30], which includes descriptions of first tools. Recent courses on the topic [26,32] focus more on the algorithms behind ODBA. The most up-to-date resource for a course on ODBA is the one in this Reasoning Web summer School.

References

1. Baader, F.: Description logic terminology. In: [5], pp. 485–495. Cambridge University Press (2003)
2. Baader, F.: Description logics. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) Reasoning Web. LNCS, vol. 5689, pp. 1–39. Springer, Heidelberg (2009)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005). Morgan-Kaufmann Publishers, Edinburgh (2005)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope further. In: Clark, K., Patel-Schneider, P.F. (eds.) Proc. of the OWLED Workshop (2008)
5. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
6. Baader, F., Hladik, J., Lutz, C., Wolter, F.: From tableaux to automata for description logics. In: Vardi, M.Y., Voronkov, A. (eds.) LPAR 2003. LNCS, vol. 2850, pp. 1–32. Springer, Heidelberg (2003), Slides of the talk are available from <http://lat.inf.tu-dresden.de/~baader/Talks/LPAR03.pdf>
7. Baader, F., Horrocks, I., Sattler, U.: Description logics for the semantic web. KI 16(4), 57–59 (2002)
8. Baader, F., Horrocks, I., Sattler, U.: Description logics. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. International Handbooks on Information Systems, pp. 3–28. Springer (2004), <http://www.cs.ox.ac.uk/people/ian.horrocks/Publications/download/2004/BaHS04a.pdf>
9. Baader, F., Lutz, C.: Description logic. In: Blackburn, P., van Benthem, J., Wolter, F. (eds.) The Handbook of Modal Logic, pp. 757–820. Elsevier (2006)
10. Baader, F., Nutt, W.: Basic description logics. In: [5], ch. 2, pp. 43–96. Cambridge University Press (2003)
11. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic \mathcal{EL} . In: Calvanese, D., Franconi, E., Haarslev, V., Lembo, D., Motik, B., Tessaris, S., Turhan, A.-Y. (eds.) Proc. of the 2007 Description Logic Workshop (DL 2007), CEUR-WS (2007)
12. Baader, F., Sattler, U.: An overview of tableau algorithms for description logics. Studia Logica 69, 5–40 (2001)
13. Baader, F., Sertkaya, B., Turhan, A.-Y.: Computing the least common subsumer w.r.t. a background terminology. Journal of Applied Logics (2007)
14. Baader, F., Suntisrivaraporn, B.: Debugging SNOMED CT using axiom pinpointing in the description logic \mathcal{EL}^+ . In: Proceedings of the International Conference on Representing and Sharing Knowledge Using SNOMED (KR-MED 2008), Phoenix, Arizona (2008)

15. Bechhofer, S., Horrocks, I., Goble, C.A., Stevens, R.: OilEd: a Reason-able Ontology Editor for the Semantic Web. In: Baader, F., Brewka, G., Eiter, T. (eds.) KI 2001. LNCS (LNAI), vol. 2174, pp. 396–408. Springer, Heidelberg (2001), OilEd download page <http://oiled.man.ac.uk>
16. Bechhofer, S., Möller, R., Crowther, P.: The DIG Description Logic Interface. In: Proc. of the 2003 Description Logic Workshop (DL 2003), Rome, Italy (2003)
17. Borgida, A., Lenzerini, M., Rosati, R.: Description logics for databases. In: [5], pp. 462–484. Cambridge University Press (2003)
18. Borgida, A., Patel-Schneider, P.F.: A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research* 1, 277–308 (1994)
19. Brachman, R.J., Borgida, A., McGuinness, D.L., Alperin Resnick, L.: The CLASSIC knowledge representation system, or, KL-ONE: the next generation. Preprints of the Workshop on Formal Aspects of Semantic Networks, Two Harbors, Cal. (1989)
20. Brachman, R.J., Levesque, H.J.: *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos (1985)
21. Brachman, R.J., McGuinness, D.L., Patel-Schneider, P.F., Resnick, L.A., Borgida, A.: Living with classic: When and how to use a kl-one-like language. In: *Principles of Semantic Networks*, pp. 401–456. Morgan Kaufmann (1991)
22. Brachman, R.J., Nardi, D.: An introduction to description logics. In: [5], ch. 1, pp. 1–40. Cambridge University Press (2003)
23. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: de Mantáras, R.L., Saitta, L. (eds.) Proc. of the 16th European Conf. on Artificial Intelligence (ECAI 2004), pp. 298–302. IOS Press (2004)
24. Brandt, S., Küsters, R., Turhan, A.-Y.: Approximation and difference in description logics. In: Fensel, D., McGuinness, D., Williams, M.-A. (eds.) Proc. of the 8th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2002). Morgan Kaufmann Publishers, San Francisco (2002)
25. Calvanese, D., De Giacomo, G.: Description logics for conceptual data modeling in uml. Course held at ESSLLI 2003 (2003), <http://www.inf.unibz.it/~calvanese/teaching/2003-08-ESSLLI-UML>
26. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R.: Ontologies and databases: The *DL-lite* approach. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) *Reasoning Web*. LNCS, vol. 5689, pp. 255–356. Springer, Heidelberg (2009), Course slides available from <http://www.inf.unibz.it/calvanese/teaching/2009-09-ReasoningWeb-school-ontologies-dbs/ReasoningWeb-2009-ontologies-dbs.pdf>
27. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: Veloso, M.M., Kambhampati, S. (eds.) Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005), pp. 602–607. AAAI Press/The MIT Press (2005)
28. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), pp. 260–270 (2006)
29. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning* 39(3), 385–429 (2007)

30. Calvanese, D., De Giacomo, G., Rodriguez-Muro, M.: Integrating data into ontologies. Part of the ISWC 2008 tutorial 'Reasoning for Ontology Engineering and Usage' (2008), <http://www.inf.unibz.it/~calvanese/teaching/2008-10-ISWC-tutorial-tones/TonesTutorial08-4.pdf>
31. Calvanese, D., Giacomo, G.D.: Expressive description logics. In: [5], ch. 5, pp. 178–218. Cambridge University Press (2003)
32. Calvanese, D., Zakharyashev, M.: Answering queries in description logics: Theory and applications to data management. Course held at ESSLLI 2010 (2010), <http://www.inf.unibz.it/~calvanese/teaching/2010-08-ESSLLI-DL-QA/>
33. Connolly, D., van Harmelen, F., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: DAML+OIL reference description. W3C Note (March 2001), <http://www.w3.org/TR/daml+oil-reference>
34. Consortium, T.G.O.: Gene Ontology: Tool for the unification of biology. *Nature Genetics* 25, 25–29 (2000)
35. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research* 31, 273–318 (2008)
36. Donini, F.M.: Complexity of reasoning. In: [5], ch. 3, pp. 96–136. Cambridge University Press (2003)
37. Donini, F.M., Colucci, S., Di Noia, T., Di Sciascio, E.: A tableaux-based method for computing least common subsumers for expressive description logics. In: Proc. of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 739–745. AAAI (July 2009)
38. Eiter, T., Lutz, C., Ortiz, M., Simkus, M.: Query answering in description logics with transitive roles. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009). AAAI Press (2009)
39. Fitting, M.: Basic modal logic. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 1, pp. 365–448. Oxford Science Publications (1993)
40. Franconi, E.: Description logics – tutorial course information (2002), <http://www.inf.unibz.it/~franconi/dl/course>
41. Gennari, J., Musen, M., Ferguson, R., Grosso, W., Crubézy, M., Eriksson, H., Noy, N., Tu, S.: The evolution of PROTÉGÉ-2000: An environment for knowledge-based system development. *International Journal of Human-Computer Studies* 58, 89–123 (2003)
42. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? a case for conservative extensions in description logics. In: Doherty, P., Mylopoulos, J., Welty, C. (eds.) Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), pp. 187–197. AAAI Press (2006)
43. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic \mathcal{SHIQ} . In: Veloso, M.M. (ed.) Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007), Hyderabad, India, pp. 399–404 (2007)
44. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 Web Ontology Language: Primer (October 27, 2009), <http://www.w3.org/TR/owl2-primer/>
45. Horrocks, I., Sattler, U.: Logical foundations for the semantic web. Slides of the ESSLLI 2003 tutorial (2003), <http://www.cs.man.ac.uk/~horrocks/ESSLLI2003>
46. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in OWL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 323–338. Springer, Heidelberg (2008)

47. Horridge, M., Sattler, U.: Understanding and repairing inferences (includes modularisation). Part of the ISWC 2008 tutorial 'Reasoning for Ontology Engineering and Usage' (2008), <http://www.inf.unibz.it/calvanese/teaching/2008-10-ISWC-tutorial-tones/TonesTutorial08-3.pdf>
48. Horrocks, I.: Using an expressive description logic: FaCT or fiction? In: Cohn, A., Schubert, L., Shapiro, S. (eds.) Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 1998), pp. 636–647 (1998)
49. Horrocks, I.: Ontology engineering: Tools and methodologies. Slides from the tutorial at the Norwegian Semantic Days (April 2007), <http://www.cs.ox.ac.uk/people/ian.horrocks/Seminars/download/SemanticDays07-tutorial.ppt>
50. Horrocks, I.: Description logic: a formal foundation for languages and tools. Slides from the tutorial at the Semantic Technology Conference (SemTech) (2010), <http://www.cs.ox.ac.uk/people/ian.horrocks/Publications/download/2010/HoPa10a.pdf>
51. Horrocks, I.: Tool support for ontology engineering. In: Fensel, D. (ed.) Foundations for the Web of Information and Services, pp. 103–112. Springer (2011)
52. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *SHOIQ*. In: Doherty, P., Mylopoulos, J., Welty, C. (eds.) Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67. AAAI Press (2006)
53. Horrocks, I., Patel-Schneider, P.F.: KR and reasoning on the semantic web: OWL. In: Domingue, J., Fensel, D., Hendler, J.A. (eds.) Handbook of Semantic Web Technologies, ch. 9, pp. 365–398. Springer (2011), <http://www.cs.ox.ac.uk/people/ian.horrocks/Publications/download/2010/HoPa10a.pdf>
54. Horrocks, I., Patel-Schneider, P.F., McGuinness, D.L., Welty, C.A.: OWL: a Description Logic Based Ontology Language for the Semantic Web. In: Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.) The Description Logic Handbook: Theory, Implementation, and Applications, ch. 14, 2nd edn., Cambridge University Press (2007), <http://www.cs.ox.ac.uk/people/ian.horrocks/Publications/download/2003/HPMW07.pdf>
55. Horrocks, I., Sattler, U.: Optimised reasoning for *SHIQ*. In: Proc. of the 15th European Conference on Artificial Intelligence (2002)
56. Horrocks, I., Sattler, U.: A tableaux decision procedure for *SHOIQ*. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005). Morgan Kaufmann (January 2005)
57. Horrocks, I., Sattler, U.: A tableau decision procedure for *SHOIQ*. J. of Automated Reasoning 39(3), 249–276 (2007)
58. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007)
59. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca Grau, B., Hendler, J.: Swoop: A web ontology editing browser. J. Web Sem. 4(2), 144–153 (2006)
60. Keet, M.: Introduction to ontology engineering, with emphasis on semantic web technologies. Meraka Institute, South Africa. Course in the Masters Ontology Winter School 2010 (2010), <http://www.meteck.org/teaching/SA/MOWS10OntoEngCouse.html>
61. Knublauch, H., Horridge, M., Musen, M.A., Rector, A.L., Stevens, R., Drummond, N., Lord, P.W., Noy, N.F., Seidenberg, J., Wang, H.: The protege owl experience. In: Grau, B.C., Horrocks, I., Parsia, B., Patel-Schneider, P.F. (eds.) Proceedings of the OWLED Workshop. CEUR Workshop Proceedings, vol. 188, CEUR-WS.org. (2005)

62. Konev, B., Ludwig, M., Walther, D., Wolter, F.: The logical difference for the lightweight description logic \mathcal{EL} . *J. Artif. Intell. Res. (JAIR)* 44, 633–708 (2012)
63. Krötzsch, M.: OWL 2 Profiles: An introduction to lightweight ontology languages. In: Eiter, T., Krennwallner, T. (eds.) *Reasoning Web 2012*. LNCS, vol. 7487, pp. 112–183. Springer, Heidelberg (2012), Course slides available from http://korrekt.org/talks/2012/OWL2_Profiles_Reasoning-Web-2012.pdf
64. Krötzsch, M., Simančík, F., Horrocks, I.: A description logic primer. *CoRR*, abs/1201.4089 (2012), <http://arxiv.org/abs/1201.4089>
65. Liebig, T., Noppens, O.: ONTOTRACK: Combining browsing and editing with reasoning and explaining for OWL lite ontologies. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) *ISWC 2004*. LNCS, vol. 3298, pp. 244–258. Springer, Heidelberg (2004), OntoTrack download page <http://www.informatik.uni-ulm.de/ki/ontotrack/>
66. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) *IJCAR 2008*. LNCS (LNAD), vol. 5195, pp. 179–193. Springer, Heidelberg (2008)
67. Lutz, C., Sattler, U.: Description logics (2005), Slides are available from <http://www.computational-logic.org/content/events/iccl-ss-2005/lectures/lutz/index.php?id=24>
68. Lutz, C., Sattler, U., Wolter, F.: Modularity in logical theories and ontologies. Slides of the ESSLI 2008 tutorial (2008), <http://cgi.csc.liv.ac.uk/~frank/publ/esslli08.html>
69. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007)*. AAAI Press (2007)
70. Lutz, C., Wolter, F.: Mathematical logic for life science ontologies. In: Ono, H., Kanazawa, M., de Queiroz, R. (eds.) *WoLLIC 2009*. LNCS, vol. 5514, pp. 37–47. Springer, Heidelberg (2009)
71. Lutz, C., Wolter, F.: Deciding inseparability and conservative extensions in the description logic \mathcal{EL} . *Journal of Symbolic Computation* 45(2), 194–228 (2010)
72. Minsky, M.: A framework for representing knowledge. Technical report, MIT-AI Laboratory, Cambridge, MA, USA (1974)
73. Möller, R.: Introduction to standard reasoning. Part of the ISWC 2008 tutorial 'Reasoning for Ontology Engineering and Usage' (2008), <http://www.inf.unibz.it/calvanese/teaching/2008-10-ISWC-tutorial-tones/TonesTutorial08-1.pdf>
74. Möller, R., Haarslev, V.: Description logic systems. In: [5], pp. 282–305. Cambridge University Press (2003)
75. Motik, B., Sattler, U.: A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In: Hermann, M., Voronkov, A. (eds.) *LPAR 2006*. LNCS (LNAI), vol. 4246, pp. 227–241. Springer, Heidelberg (2006), KAON2 download page <http://kaon2.semanticweb.org/>
76. Nebel, B.: Computational complexity of terminological reasoning in BACK. *Artificial Intelligence Journal* 34(3), 371–383 (1988)
77. Nebel, B.: Terminological reasoning is inherently intractable. *Artificial Intelligence Journal* 43, 235–249 (1990)
78. Nebel, B., von Luck, K.: Hybrid reasoning in BACK. In: *Proc. of the 3rd Int. Sym. on Methodologies for Intelligent Systems (ISMIS 1988)*, pp. 260–269. North-Holland Publ. Co., Amsterdam (1988)

79. Ortiz, M., Calvanese, D., Eiter, T.: Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning* 41(1), 61–98 (2008)
80. Ortiz, M., Simkus, M.: Reasoning and query answering in description logics. In: Eiter, T., Krennwallner, T. (eds.) *Reasoning Web 2012*. LNCS, vol. 7487, pp. 1–53. Springer, Heidelberg (2012), Course slides available from <http://www.kr.tuwien.ac.at/events/rw2012/teaching-material/RW12-Tutorial-Ortiz-Simkus-1p.pdf>
81. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL Ontologies. In: Ellis, A., Hagino, T. (eds.) *Proc. of the 14th Int. World Wide Web Conference (WWW 2005)*, Chiba, Japan, pp. 633–640 (May 2005)
82. Quillian, M.R.: Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science* 12, 410–430 (1967), Republished in [20]
83. Rector, A.: Medical informatics. In: [5], pp. 406–426. Cambridge University Press (2003)
84. Rector, A.L., Rogers, J.: Ontological and practical issues in using a description logic to represent medical concept systems: Experience from GALEN. In: Barahona, P., Bry, F., Franconi, E., Henze, N., Sattler, U. (eds.) *Reasoning Web 2006*. LNCS, vol. 4126, pp. 197–231. Springer, Heidelberg (2006)
85. Rosse, C., Mejino, J.L.V.: A reference ontology for biomedical informatics: the foundational model of anatomy. *Journal of Biomedical Informatics* 36, 478–500 (2003)
86. Sattler, U., Calvanese, D., Molitor, R.: Relationships with other formalisms. In: [5], ch. 4, pp. 137–177. Cambridge University Press (2003)
87. Schild, K.: A correspondence theory for terminological logics: preliminary report. In: Mylopoulos, J., Reiter, R. (eds.) *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI 1991)*, Sydney, Australia (1991)
88. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Gottlob, G., Walsh, T. (eds.) *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, Acapulco, Mexico, pp. 355–362. Morgan Kaufmann, Los Altos (2003)
89. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with unions and complements. Technical Report SR-88-21, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany (1988)
90. Schneider, T., Sattler, U.: Description logics: an introductory course on a nice family of logics. Course held at ESSLLI 2012 (August 2012), http://www.informatik.uni-bremen.de/~ts/teaching/2012_dl
91. Schneider, T., Walther, D.: Modularity in ontologies. Course held at ESSLLI 2011 (August 2011), http://www.informatik.uni-bremen.de/~ts/teaching/2011_modularity
92. Sioutos, N., de Coronado, S., Haber, M.W., Hartel, F.W., Shaiu, W.-L., Wright, L.W.: NCI thesaurus: A semantic model integrating cancer-related clinical and molecular information. *J. of Biomedical Informatics* 40(1), 30–43 (2007)
93. Sowa, J.F. (ed.): *Principles of Semantic Networks*. Morgan Kaufmann, Los Altos (1991)
94. Spackman, K.: Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *Journal of the American Medical Informatics Assoc.* (2000) (Fall Symposium Special Issue)
95. Turhan, A.-Y.: Bottom-up construction of ontologies. Part of the ISWC 2008 tutorial 'Reasoning for Ontology Engineering and Usage' (2008), <http://www.inf.unibz.it/calvanese/teaching/2008-10-ISWC-tutorial-tones/TonesTutorial08-2.pdf>
96. Turhan, A.-Y.: Reasoning and explanation in \mathcal{EL} and in expressive description logics. In: Aßmann, U., Bartho, A., Wende, C. (eds.) *Reasoning Web*. LNCS, vol. 6325, pp. 1–27. Springer, Heidelberg (2010)

97. Turhan, A.-Y.: Description logic reasoning for semantic web ontologies. In: Akerkar, R. (ed.) Proc. of the International Conference on Web Intelligence, Mining and Semantics, WIMS 2011, p. 6. ACM (2011)
98. Turhan, A.-Y., Bechhofer, S., Kaplunova, A., Liebig, T., Luther, M., Möller, R., Noppens, O., Patel-Schneider, P., Suntisrivaraporn, B., Weithöner, T.: DIG 2.0 – Towards a flexible interface for description logic reasoners. In: Cuenca Grau, B., Hitzler, P., Shankey, C., Wallace, E. (eds.) Proceedings of the Second International Workshop OWL: Experiences and Directions (November 2006)
99. W3C OWL Working Group. OWL 2 web ontology language document overview. W3C Recommendation (October 27, 2009),
<http://www.w3.org/TR/2009/REC-owl2-overview-20091027/>
100. Wolstencroft, K., Brass, A., Horrocks, I., Lord, P., Sattler, U., Turi, D., Stevens, R.: A little semantic web goes a long way in biology. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 786–800. Springer, Heidelberg (2005)
101. Zarriß, B., Turhan, A.-Y.: Most specific generalizations w.r.t. general \mathcal{EL} -TBoxes. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013). AAAI Press, Beijing (to appear, 2013)

Answer Set Programming

Wolfgang Faber

Department of Mathematics
University of Calabria
87030 Rende (CS), Italy
wf@wfaber.com

Abstract. Answer Set Programming (ASP) evolved from various fields such as Logic Programming, Deductive Databases, Knowledge Representation, and Non-monotonic Reasoning, and serves as a flexible language for declarative problem solving. There are two main tasks in problem solving, representation and reasoning, which are clearly separated in the declarative paradigm. In ASP, representation is done using a rule-based language, while reasoning is performed using implementations of general-purpose algorithms, referred to as ASP solvers. Rules in ASP are interpreted according to common sense principles, including a variant of the closed-world-assumption (CWA) and the unique-name-assumption (UNA). Collections of ASP rules are referred to as ASP programs, which represent the modelled knowledge. To each ASP program a collection of answer sets, or intended models, is associated, which stand for the solutions to the modelled problem; this collection can also be empty, meaning that the modelled problem does not admit a solution. Several reasoning tasks exist: the classical ASP task is enumerating all answer sets or determining whether an answer set exists, but ASP also allows for query answering in brave or cautious modes. This article provides an introduction to the field, starting with historical perspectives, followed by a definition of the core language, a guideline to knowledge representation, an overview of existing ASP solvers, and a panorama of current research topics in the field.

1 Introduction

Traditional software engineering is focussed on an imperative, algorithmic approach, in which a computer is basically being told what steps should be followed in order to solve a given problem. Finding good algorithms for hard problems is often not obvious and requires substantial skill and knowledge. Despite these efforts, solutions provided in this way typically lack flexibility: when the specification of the given problem changes only slightly, for example because additional information on the nature of the problem becomes available, major re-engineering is often necessary, in the worst case from scratch. The main problem with this approach is that the knowledge about the problem and its solutions is not represented explicitly, but only implicitly, hidden inside the algorithm. Especially in Artificial Intelligence, it does not seem like the best idea to solve problems in this way, as that would yield inflexible, and hence arguably unintelligent agents. John McCarthy has coined the term *elaboration tolerance* in the 1980s for the ease of modifying problem representations to take into account new phenomena.

He argues that natural languages fare well regarding elaboration tolerance in “human” use; clearly, algorithmic representations are not very elaboration tolerant.

Declarative programming is an alternative, which suits elaboration tolerance much better. In this approach, the problem and its solutions are specified explicitly. That is, the features of the problem and its solution are represented, rather than a method for how solutions are to be obtained. Methods like this actually come natural in science, but also in everyday life. Before we try to work out how to solve a problem, we usually first try to understand it and figure out how a solution would like, before trying to find a method to obtain a solution. One of the first to put this approach into perspective in computer science was the same John McCarthy mentioned earlier in the 1950s [1]. He also postulated that the most natural language for specifying problems and solutions would be logic, and in particular predicate logic.

In fact, logic is an excellent candidate for declarative programming: It provides a simple and abstract formalism, and in addition, it has the potential for automation. Similar to an abstract or electronic machine which can execute an algorithm, computational logic has produced tools that allow for automatically obtaining solutions, given a declarative specification in logic. Indeed, many people nowadays use this way of solving problems: Queries to relational databases together with the database schemata are examples of declarative specifications that allow for obtaining answers to the queries. Indeed, the probably most widely used database query language, SQL, essentially is predicate logic written in a particular way [2].

However, in many cases one wants to go beyond databases as they are used today. It has been shown that relational databases and query languages like SQL can only represent fairly simple problems. For instance, problems like finding the cheapest tour of a number of cities, or filling a container with items of different size, such that the value transported in the container is maximized, are typical problems that can probably not be solved using SQL. It might seem unusual to use the word “probably” here, but underlying this conjecture is one of the most famous open problems in computer science—the question whether P equals NP. These are complexity classes; basically every problem has some intrinsic complexity, indicating how many resources are required to solve it on a standard machine model, in terms of the size of the problem input. P is the class of problems that require at most polynomial amount of time in the input size (which is variable). NP is just a slight alteration, in which one also gets a polynomial-size certificate that can be verified in time polynomial in the input. The crucial question is how to obtain this certificate for a given input, and indeed so far nobody has succeeded to prove convincingly either that this can always be done in polynomial time or that it cannot be done in polynomial time. In other words, it is unknown whether P and NP are different or equal, see for example [3,4].

Logic Programming is an attempt to use declarative programming with logic that goes beyond SQL, and often beyond P. The main construct in logic programming is a rule, a expression that looks like $Head \leftarrow Body$, where $Body$ is a logic conjunction possibly involving negation, and $Head$ is either an atomic formula or a logic disjunction. This can be seen as a logic formula (\leftarrow denoting implication, and universal quantification being implicit), with the special meaning that $Head$ is defined by that rule. In the beginning of the field (as described in the following section), logic

programming actually attempted to become a full-scale programming language. Its most famous language, Prolog [5], aimed at this, but had to renounce to full declarativeness in order to achieve that goal. For instance, in Prolog rules the order inside *Body* matters, as does the order among rules (most notably for termination). Moreover, Prolog also has a number of non-logical constructs.

Answer Set Programming (ASP) is a branch of logic programming, which does not aspire to create a full general-purpose language. In this respect, it is influenced by database languages, as also these are not general-purpose languages, but suffice for a particular class of problems. ASP does however attempt to enlarge the class of problems which can be expressed by the language. While, as mentioned earlier, SQL probably cannot express hard problems in NP, ASP definitely can. Actually, ASP can uniformly express all problems in the complexity class Σ_2^P and its complement Π_2^P , which are similar to NP and co-NP, respectively, with an NP-hard oracle, that is, at each step of the computation a problem in NP might have to be solved. If it turns out that $P = NP$, then all of these classes coincide. For details on these complexity classes, see [3,4], for a survey of expressivity and complexity in ASP, see [6].

In ASP, the rule construct $Head \leftarrow Body$ (where *Head* can be a disjunction) is read like a formula in nonmonotonic logics, rather than classical logic. Nonmonotonic logics are an effort to formulate a logic of common sense that is adapting the semantics of logic such that it corresponds better to our everyday reasoning, which is characterized by the presence of incomplete knowledge, hypothetical reasoning and default assumptions. It can be argued that nonmonotonic logics are much better suited in such a setting than classical logic. One salient feature in ASP is that the truth of elements needs justifications, that is, for an atom to be true there must be a rule in the program that supports its truth. In this respect, the formula is related to intuitionistic or constructive logic, and indeed ASP has been characterized by means of an intermediate logic that is situated between classical and intuitionistic logic.

Summarizing, ASP is a formalism that has emerged syntactically from logic programming, with strong influences from database theory, nonmonotonic logics, knowledge representation, symbolic artificial intelligence, and more. Its main representation feature is the rule, which is interpreted according to common sense principles. It allows for declarative specifications of a rich class of programs, generalizing the declarative approach of databases. In ASP, one writes a program (a collection of rules), which represents a problem to be solved. This program, together with some input, which is also expressed by a collection of factual rules, possesses a collection of solutions (possibly also no solution), which correspond to the solutions of the modelled problem. Since these solutions are usually sets, the term Answer Set has been coined.

Concerning terminology, ASP is sometimes used in a somewhat broader sense, referring to any declarative formalism representing a solution as a set. However, the more frequent understanding is the narrower reading adopted in this article, which has been coined in [7]. Moreover, since ASP is the most prominent branch of logic programming in which rule heads may be disjunctive, sometimes the term Disjunctive Logic Programming can be found referring explicitly to ASP. Yet other terms for ASP are A-Prolog, AnsProlog, and Stable Logic Programming. For complementary introductory material on ASP, we refer to [8] and [9].

2 History of ASP from a Logic Programming Perspective

The roots of Answer Set Programming lie predominantly in Logic Programming, Non-monotonic Reasoning and Databases. In this section we give an overview on the history of Logic Programming from the perspective of Answer Set Programming. It therefore does not cover several important subfields of Logic Programming, such as Constraint Logic Programming [10] or Abductive Logic Programming [11].

As mentioned in the introduction, one of the first to suggest logic, and in particular predicate logic, as a programming language was John McCarthy in the 1950s [1]. McCarthy's motivating example was set in Artificial Intelligence, and involved planning as its main task, an agenda that was continuously elaborated, see for instance [12].

Developments in Computational Logic, most notably the specification of the Resolution principle and unification as a computational method by J. Alan Robinson in 1965 [13], acted as a catalyst for the rise of Logic Programming, which eventually got really started, when Prolog, a working system developed by a group around Alain Colmerauer in Marseilles, became available [5]. A few other, somewhat more restricted systems had been available before, but Prolog was to make the breakthrough for Logic Programming.

One of the prime advocates of what would become known as the Logic Programming paradigm has been Robert Kowalski, who provided the philosophical basis and concretizations of the Logic programming paradigm, for instance in [14] and [15]. Kowalski also collaborated with Colmerauer on Prolog, and within his group in Edinburgh, alternative implementations of Prolog were created. There has also been a standardization effort for the language, which would become known as Edinburgh Prolog and served as the de facto specification of Prolog for many years until the definition of ISO Prolog in 1995 [16].

However, Logic Programming, and Prolog in particular, was inspired by, but not the same as classical first-order logic. Initially the differences were not entirely clear. The first effort to provide a formal definition for the semantics of Logic Programming is also due to Kowalski, who together with Maarten van Emden gave a semantics based on fix-points of operators for a restricted class of logic programs (Horn programs, also called positive programs) in [17]. This fixpoint semantics essentially coincided with minimal Herbrand models and with resolution-based query answering on Horn programs. The major feature missing in Horn programs is negation—however Prolog did have a negation operator.

Indeed, the quest for finding a suitable semantics in the spirit of minimal models for programs containing negation turned out to be far from straightforward. A first attempt was made by Keith Clark in [18] by defining a transformation of the programs to formulas in classical logic, which are then interpreted using the classical model semantics. However, the approach gave arguably unintuitive results for programs with positive recursion. In particular, the obtained semantics does not coincide with the minimal model semantics on positive programs.

At about the same time, Raymond Reiter formulated the Closed World Assumption in [19], which can be seen as the philosophical basis of the treatment of negation. A further milestone in the research on the intended semantics for programs with negation has been the definition of what later became known uniformly as perfect model

semantics for programs that can be stratified on negation, in [20] and [21]. The basic idea of stratification is that programs can be partitioned into subprograms (strata) such that the rules of each stratum contain negative predicates only if they are defined in lower strata. In this way, it is possible to evaluate the program by separately evaluating its partitions in such a way that a given “stratum” is processed whenever the ones from which it (negatively) depends have already been processed.

While an important step forward, it is obvious that not all logic programs are stratified. In particular, programs which are recursive through negation are never stratified, and the problem of assigning a semantics to non-stratified programs still remained open. There were basically two approaches for finding suitable definitions.

The first approach was giving up the classical setting of models which assign two truth values, and introduce a third value, intuitively representing unknown. This approach required a somewhat different definition, because in the two-valued approach one would give a definition only for positive values, implicitly stating that all other constructs are considered to be negative. For instance, for minimal models, one minimizes the true elements, implicitly stating that all elements not contained in the minimal model will be false. With three truth values this strategy is no longer applicable, as elements which are not true can be either false or undefined. In order to resolve this issue, Allen Van Gelder, Kenneth Ross, and John Schlipf introduced the notion of unfounded sets in [22], in order to define which elements of the program should be considered false given some partial valuation. Combining existing techniques for defining the minimal model with unfounded sets, they defined the notion of a well-founded model. In this way, any program would still be guaranteed to have a single model, just like there is a unique minimal model for positive programs and a unique perfect model for stratified programs.

The second approach consisted of viewing logic programs as formulas in nonmonotonic logics (see for instance [23] for an overview) rather than formulas of classical logic (with an additional minimality criterion), and as a corollary, abandoning the unique model property. Among the first to concretize this were Michael Gelfond in [24], who proposed to view logic programs as formulas of autoepistemic logic, and Nicole Bidoit and Christine Froidevaux in [25], who proposed to view logic programs as formulas of default logic. Both of these developments have been picked up by Michael Gelfond and Vladimir Lifschitz, who in [26] defined the notion of stable models, which is inspired by nonmonotonic logics, however does not refer explicitly to these, but rather relies on a reduct which effectively emulates nonmonotonic inference. It was this surprisingly simple formulation, which did not require previous knowledge on non-classical logics that has become well-known, and forms the basis of ASP. While any program admits exactly one well-founded model, programs may admit no, one or many stable models. However, well-founded and stable models are closely related, for instance the well-founded model of a program is contained in each stable model, cf. [27]. Moreover, both approaches coincide with perfect models on stratified programs.

Yet another, somewhat orthogonal line of research concerned the use of disjunction in rule heads. This construct is appealing, because it allows for direct nondeterministic definitions. Prolog and many other logic programming languages traditionally do not provide such a feature, being restricted to so-called definite rules. Jack Minker has been

a pioneer and advocate of having disjunctions in programs. In [28] he formulated the Generalized Closed World Assumption (GCWA), which gave a simple and intuitive semantics for disjunctive logic programs. This concept has been elaborated on over the years, most notably by the Extended GCWA defined in [29]. Eventually, also the stable model semantics has been extended to disjunctive programs in [30] by just minimally altering the definition of [26]. On the other hand, defining an extension of well-founded models for disjunctive programs remains a controversial matter to this date with various rivalling definitions, cf. [31].

The final step towards Answer Set Programming in the traditional sense has been the addition of a second kind of negation, which has a more classical reading than negation as failure. This second negation is often known as true, strong, or classical negation. Combining this feature with disjunctive stable models of [30] led to the definition of answer sets in [7].

3 ASP Language

In what follows, we provide a formal definition of the syntax and semantics of the core Answer Set Programming language in the sense of [7], that is, disjunctive logic programming involving two kinds of negation (referred to as strong negation and negation as failure), under the answer sets semantics.

3.1 Core ASP

Syntax. Following a convention dating back to Prolog, strings starting with uppercase letters denote logical variables, while strings starting with lower case letters denote constants. A *term* is either a variable or a constant. Note that, as common in ASP, function symbols are not considered.

An *atom* is an expression $p(t_1, \dots, t_n)$, where p is a *predicate* of arity n and t_1, \dots, t_n are terms. A *classical literal* l is either an atom p (in this case, it is *positive*), or a negated atom $\neg p$ (in this case, it is *negative*). A *negation as failure (NAF) literal* ℓ is of the form l or $\text{not } l$, where l is a classical literal; in the former case ℓ is *positive*, and in the latter case *negative*. Unless stated otherwise, by *literal* we mean a classical literal.

Given a classical literal l , its *complementary literal* $\neg l$ is defined as $\neg p$ if $l = p$ and p if $l = \neg p$. A set L of literals is said to be *consistent* if, for every literal $l \in L$, its complementary literal is not contained in L .

A *disjunctive rule (rule, for short)* r is a construct

$$a_1 \vee \dots \vee a_n \leftarrow b_1, \dots, b_k, \text{not } b_{k+1}, \dots, \text{not } b_m. \quad (1)$$

where $a_1, \dots, a_n, b_1, \dots, b_m$ are classical literals and $n \geq 0, m \geq k \geq 0$. The disjunction $a_1 \vee \dots \vee a_n$ is called the *head* of r , while the conjunction $b_1, \dots, b_k, \text{not } b_{k+1}, \dots, \text{not } b_m$ is referred to as the *body* of r . A rule without head literals (i.e., $n = 0$) is usually referred to as an *integrity constraint*. A rule having precisely one head literal (i.e., $n = 1$) is called a *normal rule*. If the body is empty (i.e., $k = m = 0$), it is called a *fact*, and in this case the “ \leftarrow ” sign is usually omitted.

The following notation will be useful for further discussion. If r is a rule of form (1), then $H(r) = \{a_1, \dots, a_n\}$ is the set of literals in the head and $B(r) = B^+(r) \cup B^-(r)$ is the set of the body literals, where $B^+(r)$ (the *positive body*) is $\{b_1, \dots, b_k\}$ and $B^-(r)$ (the *negative body*) is $\{b_{k+1}, \dots, b_m\}$.

An ASP program \mathcal{P} is a finite set of rules. A not-free program \mathcal{P} (i.e., such that $\forall r \in \mathcal{P} : B^-(r) = \emptyset$) is called *positive* or *Horn*,¹ and a \vee -free program \mathcal{P} (i.e., such that $\forall r \in \mathcal{P} : |H(r)| \leq 1$) is called *normal logic program*.

In ASP, rules in programs are usually required to be safe. The motivation of safety comes from the field of databases, where safety has been introduced as a means to guarantee that queries (programs in the case of ASP) do not depend on the universe (the set of constants) considered. As an example, a fact $p(X)$. gives rise to the truth of $p(a)$ when the universe $\{a\}$ is considered, while it gives rise to the truth of $p(a)$ and $p(b)$ when the universe $\{a, b\}$ is considered. Safe programs do not suffer from this problem when at least the constants occurring in the program are considered. For a detailed discussion, we refer to [2].

A rule is *safe* if each variable in that rule also appears in at least one positive literal in the body of that rule. An ASP program is safe, if each of its rules is safe, and in the following we will only consider safe programs.

A term (an atom, a rule, a program, etc.) is called *ground*, if no variable appears in it. Sometimes a ground program is also called *propositional* program.

Example 1. Consider the following program:

$r_1: a(X) \vee b(X) \leftarrow c(X, Y), d(Y), \text{not } e(X).$

$r_2: \leftarrow c(X, Y), k(Y), e(X), \text{not } b(X)$

$r_3: m \leftarrow n, o, a(1).$

$r_4: c(1, 2).$

r_1 is a disjunctive rule with $H(r_1) = \{a(X), b(X)\}$, $B^+(r_1) = \{c(X, Y), d(Y)\}$, and $B^-(r_1) = \{e(X)\}$. r_2 is an integrity constraint with $B^+(r_2) = \{c(X, Y), k(Y), e(X)\}$, and $B^-(r_2) = \{b(X)\}$. r_3 is a ground, positive, and non-disjunctive rule with $H(r_3) = \{m\}$, $B^+(r_3) = \{n, o, a(1)\}$, and $B^-(r_3) = \emptyset$. r_4 , finally, is a fact (note that \leftarrow is omitted). Moreover, all of the rules are safe. \square

Semantics. We next describe the semantics of ASP programs, which is based on the answer set semantics originally defined in [7]. However, different to [7] only consistent answer sets are considered, as it is now standard practice.

We note that in ASP the availability of some pre-interpreted predicates is assumed, such as $=$, $<$, $>$. However, it would also be possible to define them explicitly as facts, so they are not treated in a special way here.

Herbrand Universe and Literal Base For any program \mathcal{P} , the *Herbrand universe*, denoted by $U_{\mathcal{P}}$, is the set of all constants occurring in \mathcal{P} . If no constant occurs in \mathcal{P} , $U_{\mathcal{P}}$

¹ In positive programs negation as failure (not) does not occur, while strong negation (\neg) may be present.

consists of one arbitrary constant². The *Herbrand literal base* $B_{\mathcal{P}}$ is the set of all ground (classical) literals constructable from predicate symbols appearing in \mathcal{P} and constants in $U_{\mathcal{P}}$ (note that, for each atom p , $B_{\mathcal{P}}$ contains also the strongly negated literal $\neg p$).

Example 2. Consider the following program:

$$\mathcal{P}_0 = \left\{ \begin{array}{l} r_1: a(X) \vee b(X) \leftarrow c(X,Y). \\ r_2: e(X) \leftarrow c(X,Y), \text{ not } b(X). \\ r_4: c(1,2). \end{array} \right\}$$

then, the universe is $U_{\mathcal{P}_0} = \{1, 2\}$, and the base is $B_{\mathcal{P}_0} = \{a(1), a(2), b(1), b(2), e(1), e(2), c(1,1), c(1,2), c(2,1), c(2,2), \neg a(1), \neg a(2), \neg b(1), \neg b(2), \neg e(1), \neg e(2), \neg c(1,1), \neg c(1,2), \neg c(2,1), \neg c(2,2)\}$. \square

Ground Instantiation. For any rule r , $Ground(r)$ denotes the set of rules obtained by replacing each variable in r by constants in $U_{\mathcal{P}}$ in all possible ways. For any program \mathcal{P} , its ground instantiation is the set $Ground(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} Ground(r)$. Note that for propositional programs, $\mathcal{P} = Ground(\mathcal{P})$ holds.

Example 3. Consider again program \mathcal{P}_0 of Example 2. Its ground instantiation is:

$$Ground(\mathcal{P}_0) = \left\{ \begin{array}{l} g_1: a(1) \vee b(1) \leftarrow c(1,1). \quad g_2: a(1) \vee b(1) \leftarrow c(1,2). \\ g_3: a(2) \vee b(2) \leftarrow c(2,1). \quad g_4: a(2) \vee b(2) \leftarrow c(2,2). \\ g_5: e(1) \leftarrow c(1,1), \text{ not } b(1). \quad g_6: e(1) \leftarrow c(1,2), \text{ not } b(1). \\ g_7: e(2) \leftarrow c(2,1), \text{ not } b(2). \quad g_8: e(2) \leftarrow c(2,2), \text{ not } b(2). \\ g_9: c(1,2). \end{array} \right\}$$

Note that, the atom $c(1, 2)$ was already ground in \mathcal{P}_0 , while the rules g_1, \dots, g_4 (resp. g_5, \dots, g_8) are obtained by replacing the variables in r_1 (resp. r_2) with constants in $U_{\mathcal{P}_0}$. \square

Answer Sets. For every program \mathcal{P} , its answer sets are defined using its ground instantiation $Ground(\mathcal{P})$ in two steps: First the answer sets of positive disjunctive programs are defined, then the answer sets of general programs are defined by a reduction to positive disjunctive programs and a stability condition.

An interpretation I is a consistent³ set of ground classical literals $I \subseteq B_{\mathcal{P}}$ w.r.t. a program \mathcal{P} . A consistent interpretation $X \subseteq B_{\mathcal{P}}$ is called *closed under* \mathcal{P} (where \mathcal{P} is a positive disjunctive Datalog program), if, for every $r \in Ground(\mathcal{P})$, $H(r) \cap X \neq \emptyset$ whenever $B(r) \subseteq X$. An interpretation which is closed under \mathcal{P} is also called *model* of \mathcal{P} . An interpretation $X \subseteq B_{\mathcal{P}}$ is an *answer set* for a positive disjunctive program \mathcal{P} , if

² Actually, since the language does not contain function symbols and since rules are required to be safe, this extra constant is not needed. However, we have kept the classic definition in order to avoid confusion.

³ A set $I \subseteq B_{\mathcal{P}}$ is *consistent* if for each positive classical literal such that $l \in I$ it holds that $\neg l \notin I$.

it is minimal (under set inclusion) among all (consistent) interpretations that are closed under \mathcal{P} .

Example 4. The positive program $\mathcal{P}_1 = \{a \vee \neg b \vee c.\}$ has the answer sets $\{a\}$, $\{\neg b\}$, and $\{c\}$; note that they are minimal and correspond to the multiple ways of satisfying the disjunction. Its extension $\mathcal{P}_2 = \mathcal{P}_1 \cup \{\leftarrow a.\}$ has the answer sets $\{\neg b\}$ and $\{c\}$, since the additional constraint is not satisfied by interpretation $\{a\}$. Moreover, the positive program $\mathcal{P}_3 = \mathcal{P}_2 \cup \{\neg b \leftarrow c. ; c \leftarrow \neg b.\}$ has the single answer set $\{\neg b, c\}$ (indeed, the remaining consistent closed interpretation $\{a, \neg b, c\}$ is not minimal). while, it is easy to see that, $\mathcal{P}_4 = \mathcal{P}_3 \cup \{\leftarrow c\}$ has no answer set. \square

The *reduct* or *Gelfond-Lifschitz transform* of a ground program \mathcal{P} w.r.t. a set $X \subseteq B_{\mathcal{P}}$ is the positive ground program \mathcal{P}^X , obtained from \mathcal{P} by

- deleting all rules $r \in \mathcal{P}$ for which $B^-(r) \cap X \neq \emptyset$ holds;
- deleting the negative body from the remaining rules.

An *answer set* of a program \mathcal{P} is a set $X \subseteq B_{\mathcal{P}}$ such that X is an answer set of $\text{Ground}(\mathcal{P})^X$.

Example 5. For the negative ground program $\mathcal{P}_5 = \{a \leftarrow \text{not } b.\}$, $A = \{a\}$ is the only answer set, as $\mathcal{P}_5^A = \{a.\}$. For example for $B = \{b\}$, $\mathcal{P}_5^B = \emptyset$, and so B is not an answer set. \square

Example 6. Consider again program \mathcal{P}_0 of Example 2, whose ground instantiation $\text{Ground}(\mathcal{P}_0)$ has been reported in Example 3. A naïve way to compute the answer sets of \mathcal{P}_0 is to consider all possible interpretations, checking whether they are answer sets of $\text{Ground}(\mathcal{P}_0)$.

For instance, consider the interpretation $I_0 = \{c(1, 2), a(1), e(1)\}$. The corresponding reduct $\text{Ground}(\mathcal{P}_0)^{I_0}$ contains the rules g_1, g_2, g_3, g_4, g_9 , plus $e(1) \leftarrow c(1, 1)$, $e(1) \leftarrow c(1, 2)$, $e(2) \leftarrow c(2, 1)$ and $e(2) \leftarrow c(2, 2)$, obtained by canceling the negative literals from g_5, g_6, g_7 and g_8 , respectively. We can thus verify that I_0 is an answer set for $\text{Ground}(\mathcal{P}_0)^{I_0}$ and therefore also an answer set for $\text{Ground}(\mathcal{P}_0)$ and \mathcal{P}_0 .

Let us now consider the interpretation $I_1 = \{c(1, 2), b(1), e(1)\}$, which is a model of $\text{Ground}(\mathcal{P}_0)$. The reduct $\text{Ground}(\mathcal{P}_0)^{I_1}$ contains the rules g_1, g_2, g_3, g_4, g_9 plus both $e(2) \leftarrow c(2, 1)$ and $e(2) \leftarrow c(2, 2)$ (note that both g_5 and g_6 are deleted because $b(1) \in I_1$). I_1 is not an answer set of $\text{Ground}(\mathcal{P}_0)^{I_1}$ because $\{c(1, 2), b(1)\} \subset I_1$ is. As a consequence I_1 is not an answer set of \mathcal{P}_0 .

It can be verified that \mathcal{P}_0 has two answer sets, I_0 and $\{c(1, 2), b(1)\}$. \square

Example 7. Given the ASP program $\mathcal{P}_5 = \{a \vee \neg b \leftarrow c. ; \neg b \leftarrow \text{not } a, \text{not } c. ; a \vee c \leftarrow \text{not } \neg b.\}$ and $I = \{\neg b\}$, the reduct \mathcal{P}_5^I is $\{a \vee \neg b \leftarrow c. ; \neg b.\}$. It is easy to see that I is an answer set of \mathcal{P}_5^I , and for this reason it is also an answer set of \mathcal{P}_5 .

Now consider $J = \{a\}$. The reduct \mathcal{P}_5^J is $\{a \vee \neg b \leftarrow c. ; a \vee c.\}$ and it can be easily verified that J is an answer set of \mathcal{P}_5^J , so it is also an answer set of \mathcal{P}_5 .

If, on the other hand, we take $K = \{c\}$, the reduct \mathcal{P}_5^K is equal to \mathcal{P}_5^J , but K is not an answer set of \mathcal{P}_5^K : for the rule $r : a \vee \neg b \leftarrow c$, $B(r) \subseteq K$ holds, but $H(r) \cap K \neq \emptyset$ does not. Indeed, it can be verified that I and J are the only answer sets of \mathcal{P}_5 .

In some cases, it is possible to emulate disjunction by unstratified normal rules by “shifting” the disjunction to the body [32,33,34], as shown in the following example. Consider $\mathcal{P}_5 = \{a \vee b.\}$ and its “shifted version” $\mathcal{P}'_5 = \{a \leftarrow \text{not } b. ; b \leftarrow \text{not } a.\}$. Both programs have the same answer sets, namely $\{a\}$ and $\{b\}$.

However, this is not possible in general. For example, consider $\mathcal{P}_6 = \{a \vee b. ; a \leftarrow b. ; b \leftarrow a.\}$. It has $\{a, b\}$ as its single answer set, while its “shifted version” $\mathcal{P}'_6 = \{a \leftarrow \text{not } b. ; b \leftarrow \text{not } a. ; a \leftarrow b. ; b \leftarrow a.\}$ has no answer set at all.

Note that these considerations prove that \mathcal{P}_5 and \mathcal{P}'_5 are not strongly equivalent [35]. However, there is no deep relationship between “shifted” programs and strong equivalence: They can be strongly equivalent (e.g. $\mathcal{P}_5 \cup \{\leftarrow a, b.\}$ and $\mathcal{P}'_5 \cup \{\leftarrow a, b.\}$), equivalent (e.g. \mathcal{P}_5 and \mathcal{P}'_5), or not equivalent at all (e.g. \mathcal{P}_6 and \mathcal{P}'_6).

ASP inherits its main two reasoning tasks from nonmonotonic reasoning: *brave reasoning* (also called credulous reasoning) and *cautious reasoning* (also called skeptical reasoning). Given a program \mathcal{P} and a formula ϕ without variables, $\mathcal{P} \models_b \phi$ if ϕ is true in some answer set of \mathcal{P} , $\mathcal{P} \models_c \phi$ if ϕ is true in all answer sets of \mathcal{P} . While these definitions work for any formula, in ASP ϕ is often restricted to an atom, a literal, or a conjunction of literals. When ϕ contains variables, one is asking for those substitutions σ for which $\mathcal{P} \models_b \phi\sigma$ (or $\mathcal{P} \models_c \phi\sigma$).

3.2 Semantic Characterizations

While the semantic definition given in the previous section is the one originally given in [7], several other definitions have been shown to be equivalent to it.

For instance, Pearce showed in [36] how to link stable models and answer sets to the intermediate logic HT (“Here and There”) proposed by Heyting [37] (corresponding also to the three-valued Gödel Logic G_3) by essentially adding an equilibrium criterion.

Another characterization has been provided by means of a simplified reduct that became known as the *FLP reduct* [38,39]. The FLP reduct of a ground program \mathcal{P} w.r.t. a set $X \subseteq B_{\mathcal{P}}$ is the positive ground program \mathcal{P}^X , obtained from \mathcal{P} by

- deleting all rules $r \in \mathcal{P}$ for which the body is false (i.e., $B^+(r) \not\subseteq X$ or $B^-(r) \cap X \neq \emptyset$ holds).

The definition of an answer set then remains equal modulo the replacement of Gelfond-Lifschitz reduct by FLP reduct. The motivation for this reduct was a language extension by aggregates (cf. the next section), but it also significantly simplifies the handling of core programs, as rules that are not deleted by the reduct remain intact.

Further characterizations have been provided in [40].

3.3 Language Extensions

The work on ASP started with standard rules, but fairly soon implementations extending the basic language started to emerge. The most important extensions to the ASP language can be grouped in three main classes:

- *Optimization Constructs*
- *Aggregates and External Atoms*

- *Function Symbols and Existential Quantifiers*
- *Arbitrary Formulas as Programs*
- *Preference Handling*

Optimization Constructs. The basic ASP language can be used to solve complex search problems, but does not natively provide constructs for specifying optimization problems (i.e., problems where some goal function must be minimized or maximized). Two extensions of ASP have been conceived for solving optimization problems: *Weak Constraints* [41,42] and *Optimize Statements* [43].

In the basic language, constraints are rules with empty head, and represent a condition that *must* be satisfied, and for this reason they are also called *strong* constraints. Contrary to strong constraints, *weak constraints* allow us to express desiderata, that is, conditions that *should* be satisfied. Thus, they may be violated, and their semantics involves minimizing the number of violated instances of weak constraints. In other words, the presence of strong constraints modifies the semantics of a program by discarding all models which do not satisfy some of them; while weak constraints identify an approximate solution, that is, one in which (weak) constraints are satisfied as much as possible.

From a syntactic point of view, a weak constraint is like a strong one where the implication symbol \leftarrow is replaced by \rightsquigarrow . The informal meaning of a weak constraint $\rightsquigarrow B$ is “try to falsify B ,” or “ B should preferably be false.” Additionally, a weight and a priority level for the weak constraint may be specified after the constraint enclosed in brackets (by means of positive integers or variables). When not specified, the weak constraint is assumed to have weight 1 and priority level 1, respectively.

In this case, we are interested in the answer sets which minimize the sum of weights of the violated (unsatisfied) weak constraints in the highest priority level and, among them, those which minimize the sum of weights of the violated weak constraints in the next lower level, and so on. In other words, the answer sets are considered along a lexicographic ordering along the priority levels over the sum of weights of violated weak constraints. Therefore, higher values for weights and priority levels allow for marking weak constraints of higher importance (e.g., the most important constraints are those having the highest weight among those with the highest priority level).

As an example consider the Traveling Salesman Problem (TSP). TSP is a variant of the Hamiltonian Cycle problem, which amounts to finding the shortest (minimal cost) Hamiltonian cycle (that is, a cycle including all vertices of the graph) in a directed *numerically labeled* graph. This problem can be solved by adapting the encoding of the Hamiltonian Path problem given in Section 4 (and discussed in detail there) in order to deal with labels, by adding only one weak constraint.

Suppose that the graph G is specified by predicates *node* (unary) and *arc* (ternary), and that one starting node is specified by the predicate *start* (unary).

The ASP program with weak constraints solving the TSP problem is thus as follows:

$$\begin{aligned} r_1: & \text{inPath}(X, Y) \vee \text{outPath}(X, Y) \leftarrow \text{arc}(X, Y). \\ r_2: & \text{reached}(X) \leftarrow \text{start}(X). \\ r_3: & \text{reached}(X) \leftarrow \text{reached}(Y), \text{inPath}(Y, X). \\ r_4: & \leftarrow \text{inPath}(X, Y, -), \text{inPath}(X, Y_1, -), Y <> Y_1. \end{aligned}$$

$$\begin{aligned}
r_5: & \leftarrow \text{inPath}(X, Y, _), \text{inPath}(X_1, Y, _), X \neq X_1. \\
r_6: & \leftarrow \text{node}(X), \text{not reached}(X). \\
r_7: & \leftarrow \text{start}(X), \text{not inPath}(Y, X). \\
r_8: & \leftarrow \text{inPath}(X, Y, C)[C, 1].
\end{aligned}$$

The only differences with respect to the Hamiltonian Path program of Section 4 are the constraint r_7 , which ensures cyclicity of the path, and the weak constraint r_8 , which states the preference to avoid taking arcs with high cost in the path, and has the effect of selecting those answer sets for which the total cost of arcs selected by *inPath* (which coincides with the length of the path) is the minimum (i.e., the path is the shortest).

The TSP encoding provided above is an example of the “guess, check and optimize” programming pattern [42], which extends the original “guess and check” (see Section 4) by adding an additional “optimization part” which mainly contains weak constraints. In the example above, the optimization part contains only the weak constraint r_8 .

Optimize Statements are syntactically somewhat simpler. They assign numeric values to a set of ground literals, and thereby select those answer sets for which the sum of the values assigned to literals which are true in the respective answer sets are maximal or minimal. It is not hard to see that Weak Constraints can emulate Optimize Statements, but not vice versa.

Aggregates and External Atoms. There are some simple properties, often arising in real-world applications, which cannot be encoded in a simple and natural manner using ASP. Especially properties that require the use of arithmetic operators on a set of elements satisfying some conditions (like sum, count, or maximum) require rather cumbersome encodings (often requiring an “external” ordering relation over terms), if one is confined to classic ASP. On an abstract level, this can be viewed like having an “external atom,” which use some external evaluator to determine truth or falsity. The best-known type of programs with such external atoms are HEX programs, which evolved from dl-programs that use description logics as external evaluators [44]. In the following, we will stick with the more traditional numeric aggregates for simplicity.

Similar observations have also been made in related domains, notably database systems, which led to the definition of aggregate functions. Especially in database systems this concept is by now both theoretically and practically fully integrated. When ASP systems became used in real applications, it became apparent that aggregates are needed also here. First, cardinality and weight constraints [43], which are special cases of aggregates, have been introduced. However, in general one might want to use also other aggregates (like minimum, maximum, or average), and it is not clear how to generalize the framework of cardinality and weight constraints to allow for arbitrary aggregates. To overcome this deficiency, ASP has been extended with special atoms handling aggregate functions [45,46,47,48,38,49,50]. Intuitively, an aggregate function can be thought of as a (possibly partial) function mapping multisets of constants to a constant.

An *aggregate function* is of the form $f(S)$, where S is a set term of the form $\{Vars : Conj\}$, where $Vars$ is a list of variables and $Conj$ is a conjunction of standard atoms, and f is an *aggregate function symbol*.

The most common aggregate functions compute the number of terms, the sum of non-negative integers, and minimum/maximum term in a set.

Aggregates are especially useful when real-world problems have to be dealt with. Consider the following example application:⁴ A project team has to be built from a set of employees according to the following specifications:

1. At least a given number of different skills must be present in the team.
2. The sum of the salaries of the employees working in the team must not exceed the given budget.

Suppose that our employees are provided by a number of facts of the form $emp(EmpId, Skill, Salary)$; the minimum number of different skills and the budget are specified by the facts $nSkill(N)$ and $budget(B)$. We then encode each property stated above by an aggregate atom, and enforce it by an integrity constraint:

$$\begin{aligned} r_1: in(I) \vee out(I) &\leftarrow emp(I, Sk, Sa). \\ r_3: \leftarrow nSkill(M), \text{not } \#count\{Sk : emp(I, Sk, Sa), in(I)\} &\geq M. \\ r_4: \leftarrow budget(B), \text{not } \#sum\{Sa, I : emp(I, Sk, Sa), in(I)\} &\leq B. \end{aligned}$$

Intuitively, the disjunctive rule “guesses” whether an employee is included in the team or not, while the two constraints correspond one-to-one to the requirements. Indeed, the function $\#count$ counts the number of employees in the team, while $\#sum$ sums the salaries of the employees which are part of the team.

Note that thanks to the aggregates the translation of the specifications is straightforward.

Function Symbols and Existential Quantifiers. Since ASP evolved from Datalog, traditional ASP languages do not allow for function symbols or existential quantification. However, often it is convenient to use function symbols for simple reasons like grouping arguments together; they are also needed for creating and managing more complex data structures like lists, as available in standard logic programming languages like Prolog. Also in applications concerning ontologies or the Semantic Web, often existential quantifications in rule heads would be required in order to model unknown entities. Of course, the two approaches are related, as usually it is possible to eliminate existential quantifiers by introducing Skolem functions.

A major difficulty is that extensions either by function symbols or existential quantifiers immediately lead to undecidability of the respective computational problems. A major quest is therefore the identification of significant decidable subclasses of the language.

Concerning extensions with (uninterpreted) function symbols, there are essentially two groups of proposals:

Syntactically restricted fragments, such as ω -restricted programs [51], λ -restricted programs [52], *finite-domain programs* [53], *argument-restricted programs* [54], *FDNC programs* [55], *bidirectional programs* [56], and the proposal of [57]; these approaches introduce syntactic constraints (which can be easily checked at small computational

⁴ In the example, we adopted the syntax of the DLV system, the same aggregate functions can be specified also by exploiting other ASP dialects.

cost) or explicit domain restrictions, thus allowing computability of answer sets and/or decidability of querying.

Semantically restricted fragments, such as *finitely ground programs* [53], *finitary programs* [58,59], *disjunctive finitely-recursive programs* [60] and *queries* [61,62]; with respect to syntactically restricted fragments, these approaches aim at identifying broader classes of programs for which computational tasks such as querying are decidable. However, deciding the membership of a given program in these fragments is undecidable in general.

There have been a few other proposals that treat function symbols not in the traditional logic programming sense, but as in classical logic, where most prominently the unique names assumption does not hold. We refer to [63] for an overview.

Concerning existential quantifiers in rule heads, most proposals are confined to positive, non-disjunctive programs, often referred to as Datalog[±]. The decidable subclasses in this case rely on four main syntactic paradigms, called *guardedness* [64], *weak-acyclicity* [65], *stickiness* [66,67], and *shyness* [68]. Extensions of these decidable classes to positive, but disjunctive programs have been proposed in [69], [70], and [71]. A condition which combines weak acyclicity and guardedness has been proposed in [72]. Guardedness has been extended to stratified negation in [73].

Arbitrary Formulas as Programs. There have been several attempts at generalizing the input language to arbitrary formulas, rather than rules, with the intention of identifying the “ASP logic.” Pearce had provided equilibrium logic in the 1990s [36], but it was confined to propositional logic and had only one kind of negation (negation as failure). This logic has more recently been extended to quantified equilibrium logic in [74]. Moreover, there were attempts at defining semantics of arbitrary formulas by means of a second-order sentence reminiscent of circumscription. This had first been done for propositional formulas [75], and was later extended to first-order formulas [76]. In contrast to the languages discussed in the previous sections, the goal here is to define an ASP semantics for languages that are as general as possible. Reasoning tasks for first-order theories clearly suffer from undecidability issues.

Preference Handling. ASP programs usually follow a “guess and check” programming pattern (see Section 4), where a set of rules (the guessing part) is used to guess a solution (or equivalently, to generate answer set candidates); while another set of rules, called checking part, is added to discard solutions which are not admissible. This methodology allows the programmer to distinguish between solutions and non-solutions. However, in many realistic applications the possibility to make more fine grained distinctions is required; and, in particular, distinctions between more and less preferred solutions are needed (see [77] for a discussion). For this reason, there has been a substantial amount of work on extending ASP programs with preferences, and in particular, the major focus has been on qualitative approaches. This stems from the fact that for a variety of applications numerical information is hard to obtain (preference elicitation is rather difficult) and often turns out to be unnecessary (see [77]). Still, language extensions based on quantitative information, such as weak constraints mentioned above, emulate qualitative preferences under certain conditions, and vice versa.

There are two basic possibilities for representing qualitative preferences. In one approach, the preference is specified among rules, mirroring the fact that some rules may be more reliable than others, and striving to use a set of rules that is as preferred as possible for giving reason to an answer. In the second approach, the preferences are specified among literals, reflecting information on either the likelihood or the desirability of the affirmations represented by the literals.

In the first kind of formalisms, preferences are specified by means of an ordering among rules. Formally, an *ordered logic program* is a pair $(\Pi, <)$ where Π is a logic program and $< \subseteq (\Pi \times \Pi)$ is a strict partial order. Given, $r_1, r_2 \in \Pi$, the relation $r_1 < r_2$ expresses that r_2 has higher priority than r_1 . For example, consider the following program:

$$r_1: \neg a. \qquad r_2: b \leftarrow \neg a, \text{ not } c. \qquad r_3: c \leftarrow \text{not } b.$$

This program has two answer sets, one given by $\{\neg a, b\}$ and the other given by $\{\neg a, c\}$. For the first answer set, rules r_1 and r_2 are applied; for the second, r_1 and r_3 . However, assume that we have reason to prefer r_2 to r_3 , expressed by $r_3 < r_2$. In this case we would want to obtain just the first answer set. In this case we say that the first is a *preferred answer set*.

In general, defining which answer sets should be the preferred ones in this setting is not always as obvious as in the example above, and indeed several approaches have been proposed. A comprehensive comparison of three major semantics, defined by Delgrande, Schaub, Tompits [78], by Brewka and Eiter [79], and by Wang, Zhou, Lin [80], has been presented in [81].

In the second representational approach, preferences are represented among atoms, literals or formulas.

One way of specifying this has been proposed in [82] is the use of *ordered disjunction* in rule heads. In particular, the operator \times in rule heads acts as a disjunction specifying also preferences. The meaning of a rule $a_1 \times \dots \times a_n \leftarrow \text{body}$ is that if the body is satisfied then some a_i must be in the answer set, most preferably a_1 , if this is impossible then a_2 , and so on. The formal semantics is defined by means of answer sets of split programs and of rule satisfaction degrees. There are some degrees of freedom when aggregating the satisfaction degrees of several rules, leading to different semantics, the main ones being cardinality-based, set-inclusion-based, and Pareto-based.

In the ordered disjunction approach the construction of answer sets is amalgamated with the expression of preferences. *Optimization programs* [83], on the other hand, strictly separate these two aspects. An optimization program is a pair (P_{gen}, P_{pref}) . Here, P_{gen} is an arbitrary logic program used to generate answer sets. All we require is that it produces sets of literals as its answer sets. P_{pref} is a preference program. Preference programs consist of preference rules of the form $c_1 > \dots > c_n \leftarrow \text{body}$, where the c_i are Boolean combinations of literals built from \vee, \wedge, \neg and *not*. As in the case of ordered disjunction, the semantics of these programs is based on the degree of satisfaction of preference rules, and as in the case of ordered disjunctions, there are several options for aggregating these satisfaction degrees for defining semantics.

Another ASP extension suitable for preference handling has been presented in [84]. There, standard ASP has been enriched by introducing consistency-restoring rules (cr-rules) and preferences, leading to the CR-Prolog language. Basically, in this language, besides standard ASP rules one may specify CR-rules, that are expressions of the form: $r:a_1 \vee \dots \vee a_n \leftarrow^+ \text{body}$ ($n \geq 1$). The intuitive meaning of CR-ruler is: if *body* is true then one of a_1, \dots, a_n is “possibly” believed to be true. Importantly, the name of CR-prolog rules can be directly exploited to specify preferences among them. In particular, if the fact $\text{prefer}(r_1, r_2)$ is added to a CR-program, then rule r_1 is preferred over rule r_2 . This allows one to encode partial orderings among preferred answer sets by explicitly writing preferences among CR-rules.

Other Extensions. ASP has been extended in other directions in order to meet requirements of different application domains, hence there is a number of interesting languages having the roots on ASP. For instance, ASP has been exploited for defining and implementing *action languages* (i.e., languages conceived for dealing with actions and change) \mathcal{K} [85], and \mathcal{E} [86]; while, in [87] a framework for *abduction with penalization* has been proposed and implemented as a front-end for the ASP system DLV. A logic language called *ID-Logic* [88] has been introduced to deal with classical logic with inductive definitions (which correspond semantically to logic rules). Other ASP extensions have been conceived to deal with *Ontologies* (i.e., abstract models of a complex domain). In particular, in [89] an ASP-based language for ontology specification and reasoning has been proposed, which extends ASP in order to deal with complex real-world entities, like classes, objects, compound objects, axioms, and taxonomies. In [90] an open world semantics for ASP programs has been proposed. Moreover, in [91] an extension of ASP, called *HEX-Programs*, which supports higher-order atoms as well as external atoms has been proposed. External atoms allows for embedding external sources of computation in a logic program. Thus, HEX-programs are useful for various tasks, including meta-reasoning, data type manipulations, and reasoning on top of Description Logic (DL) [92] ontologies. *Template predicates* have been introduced in [93]. Template predicates are special intensional predicates defined by means of generic reusable subprograms, which have been conceived for easing coding and improving readability and compactness of programs. Finally, nested programs, allowing for nested logical expressions to occur in rules have also been studied [94,95].

4 Knowledge Representation and Reasoning in ASP

ASP has been used in several domains, ranging from artificial intelligence over traditional databases to semantic web applications. ASP can be used to encode problems in a declarative fashion; indeed, the power of disjunctive rules allows for expressing problems which are more complex than NP, and the (optional) separation of a fixed, non-ground program from an input database allows one to obtain uniform solutions over varying instances.

More in detail, many problems of comparatively high computational complexity can be solved in a natural manner by following a “Guess&Check” programming methodology, originally introduced in [96] and refined in [42]. The idea behind this method

can be summarized as follows: a database of facts is used to specify an instance of the problem, while a set of (usually disjunctive⁵) rules, called “guessing part”, is used to define the search space; solutions are then identified in the search space by another (optional) set of rules, called “checking part”, which impose some admissibility constraint. Basically, the answer sets of the program, which combines the input database with the guessing part, represent “solution candidates”; those candidates are then filtered, by adding the checking part, which guarantee that the answer sets of the resulting program represent precisely the admissible solutions for the input instance. To grasp the intuition behind the role of both the guessing and checking parts, consider the following example.

Example 8. Suppose that we want to partition a set of persons in two groups, while avoiding that father and children belong to the same group. Following the guess&check methodology, we use a disjunctive rule to “guess” all the possible assignments of persons to groups as follows:

$$\text{group}(P, 1) \vee \text{group}(P, 2) \leftarrow \text{person}(P).$$

To understand what this rule does, consider a simple instance of the problem, in which there are two persons: joe and his father john. This instance is represented by four facts

$$\text{person}(\text{john}). \text{person}(\text{joe}). \text{father}(\text{john}, \text{joe}).$$

We can verify that the answer sets of the resulting program (facts plus disjunctive rule) correspond to all possible assignments of the three persons to two groups:

$$\begin{aligned} &\{\text{person}(\text{john}), \text{person}(\text{joe}), \text{father}(\text{john}, \text{joe}), \text{group}(\text{john}, 1), \text{group}(\text{joe}, 1)\} \\ &\{\text{person}(\text{john}), \text{person}(\text{joe}), \text{father}(\text{john}, \text{joe}), \text{group}(\text{john}, 1), \text{group}(\text{joe}, 2)\} \\ &\{\text{person}(\text{john}), \text{person}(\text{joe}), \text{father}(\text{john}, \text{joe}), \text{group}(\text{john}, 2), \text{group}(\text{joe}, 1)\} \\ &\{\text{person}(\text{john}), \text{person}(\text{joe}), \text{father}(\text{john}, \text{joe}), \text{group}(\text{john}, 2), \text{group}(\text{joe}, 2)\} \end{aligned}$$

However, we want to discard assignments in which father and children belong to the same group. To this end, we add the checking part by writing the following constraint:

$$\leftarrow \text{group}(P1, G), \text{group}(P2, G), \text{father}(P1, P2).$$

The answer sets of the augmented program are then the intending ones, where the checking part has acted as a sort of filter:

$$\begin{aligned} &\{\text{person}(\text{john}), \text{person}(\text{joe}), \text{father}(\text{john}, \text{joe}), \text{group}(\text{john}, 1), \text{group}(\text{joe}, 2)\} \\ &\{\text{person}(\text{john}), \text{person}(\text{joe}), \text{father}(\text{john}, \text{joe}), \text{group}(\text{john}, 2), \text{group}(\text{joe}, 1)\} \end{aligned}$$

□

⁵ Some ASP variants use *choice rules* as guessing part (see [97,98,43]). Moreover, in some cases, it is possible to emulate disjunction by unstratified normal rules by “shifting” the disjunction to the body [32,33,34], but this is not possible in general.

In the following, we illustrate the usage of ASP as a tool for knowledge representation and reasoning by example. In particular, we first deal with a problem motivated by classical deductive database applications; then we exploit the “Guess&Check” programming style to show how a number of well-known harder problems can be encoded in ASP.

Reachability. Given a finite directed graph $G = (V, A)$, we want to compute all pairs of nodes $(a, b) \in V \times V$ such that b is reachable from a through a nonempty sequence of arcs in A . In different terms, the problem amounts to computing the transitive closure of the relation A .

The input graph is encoded by assuming that A is represented by the binary relation $arc(X, Y)$, where a fact $arc(a, b)$ means that G contains an arc from a to b , i.e., $(a, b) \in A$; while, the set of nodes V is not explicitly represented, since the nodes appearing in the transitive closure are implicitly given by these facts.

The following program then defines a relation $reachable(X, Y)$ containing all facts $reachable(a, b)$ such that b is reachable from a through the arcs of the input graph G :

$$\begin{aligned} r_1: & \text{reachable}(X, Y) \leftarrow \text{arc}(X, Y). \\ r_2: & \text{reachable}(X, Y) \leftarrow \text{arc}(X, U), \text{reachable}(U, Y). \end{aligned}$$

The first rule states that that node Y is reachable from node X if there is an arc in the graph from X to Y , while the second rule represents the transitive closure by stating that node Y is reachable from node X if there exists a node U such that U is directly reachable from X (there is an arc from X to U) and Y is reachable from U .

As an example, consider a graph represented by the following facts:

$$\text{arc}(1, 2). \quad \text{arc}(2, 3). \quad \text{arc}(3, 4).$$

The single answer set of the program reported above together with these three facts program is

$\{\text{reachable}(1, 2), \text{reachable}(2, 3), \text{reachable}(3, 4), \text{reachable}(1, 3), \text{reachable}(2, 4), \text{reachable}(1, 4), \text{arc}(1, 2), \text{arc}(2, 3), \text{arc}(3, 4)\}$. The first three reported literals are justified by the rule r_1 , while the other literals containing the predicate $reachable$ are justified by rule r_2 .

In the following, we describe the usage of the “Guess&Check” methodology.

Hamiltonian Path. Given a finite directed graph $G = (V, A)$ and a node $a \in V$ of this graph, does there exist a path in G starting at a and passing through each node in V exactly once?

This is a classical NP-complete problem in graph theory. Suppose that the graph G is specified by using facts over predicates $node$ (unary) and arc (binary), and the starting node a is specified by the predicate $start$ (unary). Then, the following program \mathcal{P}_{hp} solves the *Hamiltonian Path* problem:

$$\begin{aligned} r_1: & \text{inPath}(X, Y) \vee \text{outPath}(X, Y) \leftarrow \text{arc}(X, Y). \\ r_2: & \text{reached}(X) \leftarrow \text{start}(X). \end{aligned}$$

$$\begin{aligned}
r_3: & \text{reached}(X) \leftarrow \text{reached}(Y), \text{inPath}(Y, X). \\
r_4: & \leftarrow \text{inPath}(X, Y), \text{inPath}(X, Y_1), Y \langle \rangle Y_1. \\
r_5: & \leftarrow \text{inPath}(X, Y), \text{inPath}(X_1, Y), X \langle \rangle X_1. \\
r_6: & \leftarrow \text{node}(X), \text{not reached}(X), \text{not start}(X).
\end{aligned}$$

The disjunctive rule (r_1) guesses a subset S of the arcs to be in the path, while the rest of the program checks whether S constitutes a Hamiltonian Path. Here, an auxiliary predicate *reached* is defined, which specifies the set of nodes which are reached from the starting node. Doing this is very similar to reachability, but the transitivity is defined over the guessed predicate *inPath* using rule r_3 . Note that *reached* is completely determined by the guess for *inPath*, no further guessing is needed.

In the checking part, the first two constraints (namely, r_4 and r_5) ensure that the set of arcs S selected by *inPath* meets the following requirements, which any Hamiltonian Path must satisfy: (i) there must not be two arcs starting at the same node, and (ii) there must not be two arcs ending in the same node. The third constraint enforces that all nodes in the graph are reached from the starting node in the subgraph induced by S .

Let us next consider an alternative program \mathcal{P}'_{hp} , which also solves the *Hamiltonian Path* problem, but intertwines the reachability with the guess:

$$\begin{aligned}
r_1: & \text{inPath}(X, Y) \vee \text{outPath}(X, Y) \leftarrow \text{reached}(X), \text{arc}(X, Y). \\
r_2: & \text{inPath}(X, Y) \vee \text{outPath}(X, Y) \leftarrow \text{start}(X), \text{arc}(X, Y). \\
r_3: & \text{reached}(X) \leftarrow \text{inPath}(Y, X). \\
r_4: & \leftarrow \text{inPath}(X, Y), \text{inPath}(X, Y_1), Y \langle \rangle Y_1. \\
r_5: & \leftarrow \text{inPath}(X, Y), \text{inPath}(X_1, Y), X \langle \rangle X_1. \\
r_6: & \leftarrow \text{node}(X), \text{not reached}(X), \text{not start}(X).
\end{aligned}$$

Here, the two disjunctive rules (r_1 and r_2), together with the auxiliary rule r_3 , guess a subset S of the arcs to be in the path, while the rest of the program checks whether S constitutes a Hamiltonian Path. Here, *reached* is defined in a different way. In fact, *inPath* is already defined in a way that only arcs reachable from the starting node will be guessed. The remainder of the checking part is the same as in \mathcal{P}_{hp} .

Ramsey Numbers. In the previous example, we have seen how a search problem can be encoded in an ASP program whose answer sets correspond to the problem solutions. We now build a program whose answer sets witness that a property does not hold, i.e., the property at hand holds if and only if the program has no answer set. We next apply the above programming scheme to a well-known problem of number and graph theory.

The Ramsey number $R(k, m)$ is the smallest integer n such that, no matter how we color the arcs of the complete undirected graph (clique) with n nodes using two colors, say red and blue, there is a red clique with k nodes (a red k -clique) or a blue clique with m nodes (a blue m -clique).

Ramsey numbers exist for all pairs of positive integers k and m [99]. We next show a program \mathcal{P}_{ra} that allows us to decide whether a given integer n is *not* the Ramsey Number $R(3, 4)$. By varying the input number n , we can determine $R(3, 4)$, as described

below. Let \mathcal{F}_{ra} be the collection of facts for input predicates *node* and *arc* encoding a complete graph with n nodes. \mathcal{P}_{ra} is the following program:

$$\begin{aligned} r_1: & \text{blue}(X, Y) \vee \text{red}(X, Y) \leftarrow \text{arc}(X, Y). \\ r_2: & \leftarrow \text{red}(X, Y), \text{red}(X, Z), \text{red}(Y, Z). \\ r_3: & \leftarrow \text{blue}(X, Y), \text{blue}(X, Z), \text{blue}(Y, Z), \text{blue}(X, W), \\ & \text{blue}(Y, W), \text{blue}(Z, W). \end{aligned}$$

Intuitively, the disjunctive rule r_1 guesses a color for each edge. The first constraint (r_2) eliminates the colorings containing a red clique (i.e., a complete graph) with 3 nodes, and the second constraint (r_3) eliminates the colorings containing a blue clique with 4 nodes. The program $\mathcal{P}_{ra} \cup \mathcal{F}_{ra}$ has an answer set if and only if there is a coloring of the edges of the complete graph on n nodes containing no red clique of size 3 and no blue clique of size 4. Thus, if there is an answer set for a particular n , then n is *not* $R(3, 4)$, that is, $n < R(3, 4)$. On the other hand, if $\mathcal{P}_{ra} \cup \mathcal{F}_{ra}$ has no answer set, then $n \geq R(3, 4)$. Thus, the smallest n such that no answer set is found is the Ramsey number $R(3, 4)$.

Strategic Companies. In the examples considered so far, the complexity of the problems is located at most on the first level of the Polynomial Hierarchy [100] (in NP or co-NP). We next demonstrate that also more complex problems, located at the second level of the Polynomial Hierarchy, can be encoded in ASP. To this end, we now consider a knowledge representation problem, inspired by a common business situation, which is known under the name *Strategic Companies* [101].

Suppose there is a collection $C = \{c_1, \dots, c_m\}$ of companies c_i owned by a holding, a set $G = \{g_1, \dots, g_n\}$ of goods, and for each c_i we have a set $G_i \subseteq G$ of goods produced by c_i and a set $O_i \subseteq C$ of companies controlling (owning) c_i . O_i is referred to as the *controlling set* of c_i . This control can be thought of as a majority in shares; companies not in C , which we do not model here, might have shares in companies as well. Note that, in general, a company might have more than one controlling set. Let the holding produce all goods in G , i.e., $G = \bigcup_{c_i \in C} G_i$.

A subset of the companies $C' \subseteq C$ is a *production-preserving* set if the following conditions hold: (1) The companies in C' produce all goods in G , i.e., $\bigcup_{c_i \in C'} G_i = G$. (2) The companies in C' are closed under the controlling relation, i.e., if $O_i \subseteq C'$ for some $i = 1, \dots, m$ then $c_i \in C'$ must hold.

A subset-minimal set C' , which is *production-preserving*, is called a *strategic set*. A company $c_i \in C$ is called *strategic*, if it belongs to some strategic set of C .

This notion is relevant when companies should be sold. Indeed, intuitively, selling any non-strategic company does not reduce the economic power of the holding. Computing strategic companies is on the second level of the Polynomial Hierarchy [101].

In the following, we consider a simplified setting as considered in [101], where each product is produced by at most two companies (for each $g \in G$ $|\{c_i \mid g \in G_i\}| \leq 2$) and each company is jointly controlled by at most three other companies, i.e., $|O_i| \leq 3$ for $i = 1, \dots, m$. Assume that for a given instance of Strategic Companies, \mathcal{F}_{st} contains the following facts:

- *company*(c) for each $c \in C$,
- *prod_by*(g, c_j, c_k), if $\{c_i \mid g \in G_i\} = \{c_j, c_k\}$, where c_j and c_k may possibly coincide,
- *contr_by*(c_i, c_k, c_m, c_n), if $c_i \in C$ and $O_i = \{c_k, c_m, c_n\}$, where c_k, c_m , and c_n are not necessarily distinct.

We next present a program \mathcal{P}_{st} , which characterizes this hard problem using only two rules:

$$\begin{aligned} r_1: & \textit{strat}(Y) \vee \textit{strat}(Z) \leftarrow \textit{prod_by}(X, Y, Z). \\ r_2: & \textit{strat}(W) \leftarrow \textit{contr_by}(W, X, Y, Z), \textit{strat}(X), \textit{strat}(Y), \textit{strat}(Z). \end{aligned}$$

Here *strat*(X) means that company X is a strategic company. The guessing part of the program consists of the disjunctive rule r_1 , and the checking part consists of the normal rule r_2 . The program \mathcal{P}_{st} is surprisingly succinct, given that Strategic Companies is a hard problem.

The program \mathcal{P}_{st} exploits the minimization which is inherent to the semantics of answer sets for the check whether a candidate set C' of companies that produces all goods and obeys company control is also minimal with respect to this property.

The guessing rule r_1 intuitively selects one of the companies c_1 and c_2 that produce some item g , which is described by *prod_by*(g, c_1, c_2). If there was no company control information, minimality of answer sets would naturally ensure that the answer sets of $\mathcal{F}_{st} \cup \{r_1\}$ correspond to the strategic sets; no further checking would be needed. However, in case control information is available, the rule r_2 checks that no company is sold that would be controlled by other companies in the strategic set, by simply requesting that this company must be strategic as well. The minimality of the strategic sets is automatically ensured by the minimality of answer sets.

The answer sets of $\mathcal{F}_{st} \cup \mathcal{P}_{st}$ correspond one-to-one to the strategic sets of the holding described in \mathcal{F}_{st} ; a company c is thus strategic iff *strat*(c) is in some answer set of $\mathcal{F}_{st} \cup \mathcal{P}_{st}$.

An important note here is that the checking “constraint” r_2 interferes with the guessing rule r_1 : applying r_2 may “spoil” the minimal answer set generated by r_1 . For example, suppose the guessing part gives rise to a ground rule

$$r_3: \textit{strat}(c1) \vee \textit{strat}(c2) \leftarrow \textit{prod_by}(g, c1, c2)$$

and the fact *prod_by*($g, c1, c2$) is given in \mathcal{F}_{st} . Now suppose the rule is satisfied in the guessing part by making *strat*($c1$) true. If, however, in the checking part an instance of rule r_2 is applied which derives *strat*($c2$), then the application of the rule r_3 to derive *strat*($c1$) is invalidated, as the minimality of answer sets implies that rule r_3 cannot justify the truth of *strat*($c1$), if another atom in its head is true.

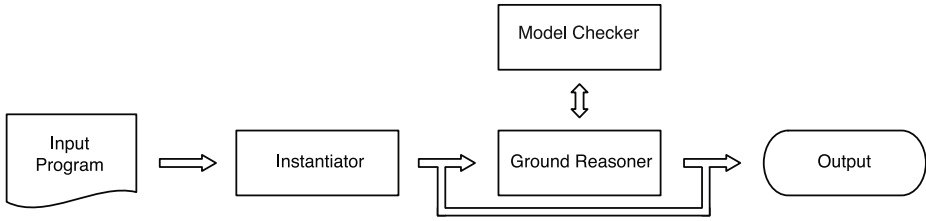


Fig. 1. General architecture of an ASP system

5 Implementations and Applications

In this section we consider some additional topics that allow the reader to have a broader picture of ASP. In particular, we introduce the general architecture of ASP systems, and we briefly describe several applications of ASP.

5.1 System Algorithms

Initially somewhat impeded by complexity considerations, reasonable algorithms and systems supporting ASP became available in the second half of the 1990s. The first widely used ones were *Smodels* [102,43], supporting non-disjunctive ASP, and *DLV* [42], supporting ASP (with disjunction) as defined in [7]. These two systems have been improved over the years and are still in widespread use. Later-on, more systems for non-disjunctive ASP, like *ASSAT* [103,104], *Cmodels* [105], and *Clasp* [106] became available, and also more disjunctive ASP systems became available with the advent of *GnT* [107], *cmodels-3* [108], and *ClaspD* [109].

While, as discussed below, the systems do not use the same techniques, they basically agree on the general architecture depicted in Figure 1.

The evaluation flow of the computation is outlined in detail. Upon startup, the input specified by the user is parsed and transformed into the internal data structures of the system.⁶

In general, an input program \mathcal{P} contains variables, and the first step of a computation of an ASP system is to eliminate these variables, generating a ground instantiation $ground(\mathcal{P})$ of \mathcal{P} . This variable-elimination process is called *instantiation* of the program (or *grounding*), and is performed by the *Instantiator* module (see Figure 1).

A naïve Instantiator would produce the full ground instantiation $Ground(\mathcal{P})$ of the input, which is, however, undesirable from a computational point of view, as in general many useless ground rules would be generated. All of the systems therefore employ different procedures, which are geared towards keeping the instantiated program as small as possible. A necessary condition is, of course, that the instantiated program must have the same answer sets as the original program. However, it should be noted that the Instantiator solves a problem, which is in general EXPTIME-hard, the produced

⁶ The input is usually read from text files, but some systems also interface to relational databases for retrieving facts stored in relational tables.

ground program being potentially of exponential size with respect to the input program. Optimizations in the Instantiator therefore often have a big impact, as its output is the input for the following modules, which implement computationally hard algorithms. Moreover, if the input program is normal and stratified, the Instantiator module is, in some cases, able to directly compute its stable model (if it exists).

The subsequent computations, which constitute the non-deterministic part of an ASP system, are then performed on $ground(\mathcal{P})$ by both the *Ground Reasoner* and the *Model Checker*. Roughly, the former produces some “candidate” answer set, whose stability is subsequently verified by the latter.

The existing ASP systems mainly differ in the technique employed for implementing the *Ground Reasoner*. There are basically two approaches, that we will refer to as *search-based* and *rewriting-based*. In the search-based approach, the *Ground Reasoner* implements a backtracking search algorithm, which works directly on the ground instantiation of the input program. Search-based systems, like e.g. DLV and Smodels, are often referred to as “native” ASP systems, because the employed algorithms directly manipulate logic programs and are optimized for those. In the rewriting-based approach, the *Ground Reasoner* transforms the ground program into a propositional formula and then invokes a Boolean satisfiability solver for finding answer set candidates.

As previously pointed out, the *Model Checker* verifies whether an answer set candidate at hand is an answer set for the input program. This task is as hard as the problem solved by the Ground Reasoner for disjunctive programs, while it is trivial for non-disjunctive programs. However, there is also a class of disjunctive programs, called Head-Cycle-Free programs [32], for which the task solved by the Model Checker is provably simpler, which is exploited in the system algorithms.

Finally, once an answer set has been found, ASP systems typically print it in text format, and possibly the *Ground Reasoner* resumes in order to look for further answer sets.

In order to implement query answering, an important technique is a generalization of the Magic Set algorithm, originally proposed for Datalog programs. It has been extended to ASP without negation and integrity constraints in [110] and [111], and extended to certain classes of programs containing negation but no disjunction in [112,113]. In [114] the technique is described for programs with both disjunction and negation (but in a limited form, also integrity constraints are not permitted), and in [115] a large fragment of programs has been identified, for which this technique is correct. It has been implemented in the ASP system DLV, and also inside the system KAON2 [116,117].

5.2 Applications

Answer Set Programming has been successfully applied to many areas including:

- *Information integration*. ASP has been exploited for supporting consistent query answering, in information integration systems under the so-called Global-as-View approach [118,119,120], also in presence of data inconsistencies and data incompleteness.

- *Configuration and Verification management.* In product configuration [121], ASP has been used as a declarative semantics providing formal definitions for main concepts in product configuration, including configuration models, requirements, and valid configurations. And, in particular, in the field of software configuration, a prototype configurator for the complete Debian Linux system distribution has been implemented by using ASP [122].
- *Knowledge Management.* ASP has a strong potential for exploitation in the area of knowledge management and semantic technologies. An ASP-based system for ontology representation and reasoning, called OntoDLV [89], is employed in many real-world applications, ranging from e-learning to enterprise ontologies and agent-based applications. In [123] an ASP-based approach to the problem of recognizing and extracting information from unstructured documents has been presented. While, in [124,125] a system for content classification, called OLEX, is presented, which exploits ASP to extract concepts and semantic metadata from documents.
- *Security engineering.* In [126] it is shown how security protocols can be specified and verified efficiently and effectively by embedding reasoning about actions into logic programming. In particular, two significant case studies in protocol verification have been modeled: the classical Needham-Schroeder public-key protocol, and the Aziz-Diffie key agreement protocol for mobile communication.

Moreover, applications from various areas can be found in the literature, including auctions [127], scheduling [128], policy description [129], workflow management [130], outlier detection [131], linguistics [132], multi agent systems [133,134,135], and E-learning [135].

Concluding, ASP is an appealing tool for knowledge representation and reasoning, and thanks to the applicability the implementations of ASP solvers to real-world problems, it is tackling many industrially-relevant applications.

It is worth noting that, ASP systems are currently away from comfortably enabling the development of industry-level applications; and, like any other programming language, ASP needs tools and development methodologies to facilitate and improve the coding process. At the time of this writing, the field of software engineering for ASP has been already settled by the ASP community [136], and it is currently evolving. Indeed, both methodologies (see Section 4) and prototype tools are already available (see [137,93,136,138,139,89]).

Acknowledgements. The author thanks all his co-authors over the years, for the discussions and the work that laid the foundation to this overview; in particular Nicola Leone and Francesco Ricca, with whom I co-authored a description, on which the present work has been based.

References

1. McCarthy, J.: Programs with Common Sense. In: Proceedings of the Teddington Conference on the Mechanization of Thought Processes, Her Majesty's Stationery Office, pp. 75–91 (1959)

2. Abiteboul, S., Hull, R., Vianu, V.: *Foundations of Databases*. Addison-Wesley (1995)
3. Goldreich, O.: *Computational Complexity: A Conceptual Perspective*. Cambridge University Press (2008)
4. Arora, S., Barak, B.: *Computational Complexity: A Modern Approach*. Cambridge University Press (2009)
5. Colmerauer, A., Roussel, P.: *The Birth of Prolog*. ACM, New York (1996)
6. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: *Complexity and Expressive Power of Logic Programming*. *ACM Computing Surveys* 33(3), 374–425 (2001)
7. Gelfond, M., Lifschitz, V.: *Classical Negation in Logic Programs and Disjunctive Databases*. *New Generation Computing* 9, 365–385 (1991)
8. Baral, C.: *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press (2003)
9. Gelfond, M., Leone, N.: *Logic Programming and Knowledge Representation – the A-Prolog perspective*. *Artificial Intelligence* 138(1-2), 3–38 (2002)
10. Marriott, K., Stuckey, P.J.: *Programming with Constraints: An Introduction*. MIT Press (1998)
11. Kakas, A.C., Kowalski, R.A., Toni, F.: *Abductive Logic Programming*. *Journal of Logic and Computation* 2(6), 719–770 (1992)
12. McCarthy, J., Hayes, P.J.: *Some Philosophical Problems from the Standpoint of Artificial Intelligence*. In: Meltzer, B., Michie, D. (eds.) *Machine Intelligence* 4, pp. 463–502. Edinburgh University Press (1969) reprinted in [140]
13. Robinson, J.A.: *A Machine-Oriented Logic Based on the Resolution Principle*. *Journal of the ACM* 12(1), 23–41 (1965)
14. Kowalski, R.A.: *Predicate Logic as Programming Language*. In: *IFIP Congress*, pp. 569–574 (1974)
15. Kowalski, R.A.: *Algorithm = Logic + Control*. *Communications of the ACM* 22(7), 424–436 (1979)
16. International Organization for Standardization: *ISO/IEC 13211-1:1995: Information technology — Programming languages — Prolog — Part 1: General core*. International Organization for Standardization, Geneva, Switzerland (1995)
17. van Emden, M.H., Kowalski, R.A.: *The Semantics of Predicate Logic as a Programming Language*. *Journal of the ACM* 23(4), 733–742 (1976)
18. Clark, K.L.: *Negation as Failure*. In: Gallaire, H., Minker, J. (eds.) *Logic and Data Bases*, pp. 293–322. Plenum Press, New York (1978)
19. Reiter, R.: *On Closed World Data Bases*. In: Gallaire, H., Minker, J. (eds.) *Logic and Data Bases*, pp. 55–76. Plenum Press, New York (1978)
20. Apt, K.R., Blair, H.A., Walker, A.: *Towards a Theory of Declarative Knowledge*. In: [141], pp. 89–148
21. Van Gelder, A.: *Negation as Failure Using Tight Derivations for General Logic Programs*. In: [141], pp. 1149–1176
22. Van Gelder, A., Ross, K.A., Schlipf, J.S.: *Unfounded Sets and Well-Founded Semantics for General Logic Programs*. In: *Proceedings of the Seventh Symposium on Principles of Database Systems (PODS 1988)*, pp. 221–230 (1988)
23. Marek, V.W., Truszczyński, M.: *Nonmonotonic Logics – Context-Dependent Reasoning*. Springer (1993)
24. Gelfond, M.: *On Stratified Autoepistemic Theories*. In: *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI 1987)*, pp. 207–211 (1987)
25. Bidoit, N., Froidevaux, C.: *Minimalism subsumes Default Logic and Circumscription in Stratified Logic Programming*. In: *Proceedings of the Symposium on Logic in Computer Science (LICS 1987)*, pp. 89–97. IEEE (June 1987)

26. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In: *Logic Programming: Proceedings Fifth Intl Conference and Symposium*, pp. 1070–1080. MIT Press, Cambridge (1988)
27. Van Gelder, A., Ross, K.A., Schlipf, J.S.: The Well-Founded Semantics for General Logic Programs. *Journal of the ACM* 38(3), 620–650 (1991)
28. Minker, J.: On Indefinite Data Bases and the Closed World Assumption. In: Loveland, D.W. (ed.) *CADE 1982. LNCS*, vol. 138, pp. 292–308. Springer, Heidelberg (1982)
29. Yahya, A.H., Henschen, L.J.: Deduction in Non-Horn Databases. *Journal of Automated Reasoning* 1(2), 141–160 (1985)
30. Przymusiński, T.C.: Stable Semantics for Disjunctive Programs. *New Generation Computing* 9, 401–424 (1991)
31. Wang, K., Zhou, L.: Comparisons and Computation of Well-founded Semantics for Disjunctive Logic Programs. *ACM Transactions on Computational Logic* 6(2) (April 2005)
32. Ben-Eliyahu, R., Dechter, R.: Propositional Semantics for Disjunctive Logic Programs. *Annals of Mathematics and Artificial Intelligence* 12, 53–87 (1994)
33. Dix, J., Gottlob, G., Marek, V.W.: Reducing Disjunctive to Non-Disjunctive Semantics by Shift-Operations. *Fundamenta Informaticae* 28, 87–100 (1996) (This is a full version of [142])
34. Leone, N., Rullo, P., Scarcello, F.: Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics and Computation. *Information and Computation* 135(2), 69–112 (1997)
35. Lifschitz, V., Pearce, D., Valverde, A.: Strongly Equivalent Logic Programs. *ACM Transactions on Computational Logic* 2(4), 526–541 (2001)
36. Pearce, D.: Equilibrium logic. *Annals of Mathematics and Artificial Intelligence* 47(1-2), 3–41 (2006)
37. Heyting, A.: Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-Mathematische Klasse*, 42–56 (1930)
38. Faber, W., Leone, N., Pfeifer, G.: Recursive aggregates in disjunctive logic programs: Semantics and complexity. In: Alferes, J.J., Leite, J. (eds.) *JELIA 2004. LNCS (LNAI)*, vol. 3229, pp. 200–212. Springer, Heidelberg (2004)
39. Faber, W., Leone, N., Pfeifer, G.: Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence* 175(1), 278–298 (2011); Special Issue: John McCarthy’s Legacy
40. Lifschitz, V.: Twelve definitions of a stable model. In: Garcia de la Banda, M., Pontelli, E. (eds.) *ICLP 2008. LNCS*, vol. 5366, pp. 37–51. Springer, Heidelberg (2008)
41. Buccafurri, F., Leone, N., Rullo, P.: Enhancing Disjunctive Datalog by Constraints. *IEEE Transactions on Knowledge and Data Engineering* 12(5), 845–860 (2000)
42. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic* 7(3), 499–562 (2006)
43. Simons, P., Niemelä, I., Sooinen, T.: Extending and Implementing the Stable Model Semantics. *Artificial Intelligence* 138, 181–234 (2002)
44. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining Answer Set Programming with Description Logics for the Semantic Web. In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR 2004)*, Whistler, Canada, pp. 141–151 (2004); Extended Report RR-1843-03-13, Institut für Informationssysteme, TU Wien (2003)
45. Calimeri, F., Faber, W., Leone, N., Perri, S.: Declarative and Computational Properties of Logic Programs with Aggregates. In: *Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pp. 406–411 (August 2005)

46. Dell'Armi, T., Faber, W., Ielpa, G., Leone, N., Pfeifer, G.: Aggregate Functions in Disjunctive Logic Programming: Semantics, Complexity, and Implementation in DLV. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, pp. 847–852. Morgan Kaufmann Publishers (August 2003)
47. Denecker, M., Pelov, N., Bruynooghe, M.: Ultimate Well-Founded and Stable Model Semantics for Logic Programs with Aggregates. In: Codognet, P. (ed.) ICLP 2001. LNCS, vol. 2237, pp. 212–226. Springer, Heidelberg (2001)
48. Faber, W., Leone, N.: On the Complexity of Answer Set Programming with Aggregates. In: Baral, C., Brewka, G., Schlipf, J. (eds.) LPNMR 2007. LNCS (LNAI), vol. 4483, pp. 97–109. Springer, Heidelberg (2007)
49. Hella, L., Libkin, L., Nurmonen, J., Wong, L.: Logics with aggregate operators. *Journal of the ACM* 48(4), 880–907 (2001)
50. Pelov, N., Denecker, M., Bruynooghe, M.: Well-founded and Stable Semantics of Logic Programs with Aggregates. *Theory and Practice of Logic Programming* 7(3), 301–353 (2007)
51. Syrjänen, T.: Omega-Restricted Logic Programs. In: Eiter, T., Faber, W., Truszczyński, M. (eds.) LPNMR 2001. LNCS (LNAI), vol. 2173, pp. 267–279. Springer, Heidelberg (2001)
52. Gebser, M., Schaub, T., Thiele, S.: GrinGo: A new grounder for answer set programming. In: Baral, C., Brewka, G., Schlipf, J. (eds.) LPNMR 2007. LNCS (LNAI), vol. 4483, pp. 266–271. Springer, Heidelberg (2007)
53. Calimeri, F., Cozza, S., Ianni, G., Leone, N.: Computable Functions in ASP: Theory and Implementation. In: Garcia de la Banda, M., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 407–424. Springer, Heidelberg (2008)
54. Lierler, Y., Lifschitz, V.: One More Decidable Class of Finitely Ground Programs. In: Hill, P.M., Warren, D.S. (eds.) ICLP 2009. LNCS, vol. 5649, pp. 489–493. Springer, Heidelberg (2009)
55. Šimkus, M., Eiter, T.: FDNC: Decidable Non-monotonic Disjunctive Logic Programs with Function Symbols. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 514–530. Springer, Heidelberg (2007)
56. Eiter, T., Šimkus, M.: Bidirectional Answer Set Programs with Function Symbols. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), Pasadena, CA, USA, pp. 765–771 (July 2009)
57. Lin, F., Wang, Y.: Answer Set Programming with Functions. In: Proceedings of Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), Sydney, Australia, pp. 454–465. AAAI Press (September 2008)
58. Bonatti, P.A.: Reasoning with infinite stable models II: Disjunctive programs. In: Stuckey, P.J. (ed.) ICLP 2002. LNCS, vol. 2401, pp. 333–346. Springer, Heidelberg (2002)
59. Bonatti, P.A.: Reasoning with infinite stable models. *Artificial Intelligence* 156(1), 75–111 (2004)
60. Baselice, S., Bonatti, P.A., Crisculo, G.: On Finitely Recursive Programs. *Theory and Practice of Logic Programming* 9(2), 213–238 (2009)
61. Calimeri, F., Cozza, S., Ianni, G., Leone, N.: Magic Sets for the Bottom-Up Evaluation of Finitely Recursive Programs. In: Erdem, E., Lin, F., Schaub, T. (eds.) LPNMR 2009. LNCS, vol. 5753, pp. 71–86. Springer, Heidelberg (2009)
62. Alviano, M., Faber, W., Leone, N.: Disjunctive asp with functions: Decidable queries and effective computation. In: *Theory and Practice of Logic Programming*, 26th Int'l. Conference on Logic Programming (ICLP 2010) (2010); Special Issue 10(4–6), 497–512 (2010)
63. Cabalar, P.: Partial Functions and Equality in Answer Set Programming. In: Garcia de la Banda, M., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 392–406. Springer, Heidelberg (2008)

64. Cali, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Brewka, G., Lang, J. (eds.) *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*, pp. 70–80. AAAI Press (2008)
65. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. *Theoretical Computer Science* 336(1), 89–124 (2005)
66. Cali, A., Gottlob, G., Pieris, A.: Advanced processing for ontological queries. *Proceedings of the VLDB Endowment* 3(1), 554–565 (2010)
67. Cali, A., Gottlob, G., Pieris, A.: Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence* 193, 87–128 (2012)
68. Leone, N., Manna, M., Terracina, G., Veltri, P.: Efficiently computable datalog[∃] programs. In: Brewka, G., Eiter, T., McIlraith, S. (eds.) *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*. AAAI Press (2012)
69. Alviano, M., Faber, W., Leone, N., Manna, M.: Disjunctive datalog with existential quantifiers: Semantics, decidability, and complexity issues. In: *Theory and Practice of Logic Programming, 28th Int'l. Conference on Logic Programming (ICLP 2012)* (2012); Special Issue 12(4-5), 701–718 (July 2012)
70. Gottlob, G., Manna, M., Morak, M., Pieris, A.: On the complexity of ontological reasoning under disjunctive existential rules. In: Rovan, B., Sassone, V., Widmayer, P. (eds.) *MFCS 2012*. LNCS, vol. 7464, pp. 1–18. Springer, Heidelberg (2012)
71. Pierre Bourhis, M.M., Pieris, A.: The impact of disjunction on ontological query answering under guarded-based existential rules. In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)* (2013)
72. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: Walsh, T. (ed.) *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pp. 963–968 (2011)
73. Cali, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantics* 14, 57–83 (2012)
74. Pearce, D., Valverde, A.: Quantified equilibrium logic and foundations for answer set programs. In: Garcia de la Banda, M., Pontelli, E. (eds.) *ICLP 2008*. LNCS, vol. 5366, pp. 546–560. Springer, Heidelberg (2008)
75. Ferraris, P.: Answer Sets for Propositional Theories. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) *LPNMR 2005*. LNCS (LNAI), vol. 3662, pp. 119–131. Springer, Heidelberg (2005)
76. Ferraris, P., Lee, J., Lifschitz, V.: A new perspective on stable models. In: *Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pp. 372–379 (January 2007)
77. Brewka, G.: Answer Sets: From Constraint Programming Towards Qualitative Optimization. In: Lifschitz, V., Niemelä, I. (eds.) *LPNMR 2004*. LNCS (LNAI), vol. 2923, pp. 34–46. Springer, Heidelberg (2003)
78. Delgrande, J.P., Schaub, T., Tompits, H.: A Framework for Compiling Preferences in Logic Programs. *Theory and Practice of Logic Programming* 3(2), 129–187 (2003)
79. Brewka, G., Eiter, T.: Preferred Answer Sets for Extended Logic Programs. *Artificial Intelligence* 109(1-2), 297–356 (1999)
80. Wang, K., Zhou, L., Lin, F.: Alternating Fixpoint Theory for Logic Programs with Priority. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) *CL 2000*. LNCS (LNAI), vol. 1861, pp. 164–178. Springer, Heidelberg (2000)

81. Schaub, T., Wang, K.: A Comparative Study of Logic Programs with Preference. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001), Seattle, WA, USA, pp. 597–602. Morgan Kaufmann Publishers (August 2001)
82. Brewka, G.: Logic Programming with Ordered Disjunction. In: Proceedings of the 9th International Workshop on Non-Monotonic Reasoning (NMR 2002), pp. 67–76 (April 2002)
83. Brewka, G., Niemelä, I., Truszczyński, M.: Answer Set Optimization. In: Gottlob, G., Walsh, T. (eds.) Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, pp. 867–872. Morgan Kaufmann (August 2003)
84. Balduccini, M., Gelfond, M.: Logic Programs with Consistency-Restoring Rules. In: Doherty, P., McCarthy, J., Williams, M. (eds.) International Symposium on Logical Formalization of Commonsense Reasoning, AAAI 2003 Spring Symposium Series (2003)
85. Eiter, T., Faber, W., Leone, N., Pfeifer, G., Polleres, A.: A Logic Programming Approach to Knowledge-State Planning: Semantics and Complexity. *ACM Transactions on Computational Logic* 5(2), 206–263 (2004)
86. Dimopoulos, Y., Kakas, A.C., Michael, L.: Reasoning About Actions and Change in Answer Set Programming Programs. In: Lifschitz, V., Niemelä, I. (eds.) LPNMR 2004. LNCS (LNAI), vol. 2923, pp. 61–73. Springer, Heidelberg (2003)
87. Perri, S., Scarcello, F., Leone, N.: Abductive Logic Programs with Penalization: Semantics, Complexity and Implementation. *Theory and Practice of Logic Programming* 5(1-2), 123–159 (2005)
88. Mariën, M., Gilis, D., Denecker, M.: On the Relation Between ID-Logic and Answer Set Programming. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 108–120. Springer, Heidelberg (2004)
89. Ricca, F., Leone, N.: Disjunctive Logic Programming with types and objects: The DLV⁺ System. *Journal of Applied Logics* 5(3), 545–573 (2007)
90. Heymans, S., Van Nieuwenborgh, D., Hadavandi, E.: Semantic web reasoning with conceptual logic programs. In: Antoniou, G., Boley, H. (eds.) RuleML 2004. LNCS, vol. 3323, pp. 113–127. Springer, Heidelberg (2004)
91. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer Set Programming. In: International Joint Conference on Artificial Intelligence (IJCAI 2005), Edinburgh, UK, pp. 90–96 (August 2005)
92. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press (2003)
93. Calimeri, F., Ianni, G., Ielpa, G., Pietramala, A., Santoro, M.C.: A system with template answer set programs. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 693–697. Springer, Heidelberg (2004)
94. Pearce, D., Tompits, H., Woltran, S.: Encodings for Equilibrium Logic and Logic Programs with Nested Expressions. In: Brazdil, P.B., Jorge, A.M. (eds.) EPIA 2001. LNCS (LNAI), vol. 2258, pp. 306–320. Springer, Heidelberg (2001)
95. Pearce, D., Sarsakov, V., Schaub, T., Tompits, H., Woltran, S.: A Polynomial Translation of Logic Programs with Nested Expressions into Disjunctive Logic Programs: Preliminary Report. In: Proceedings of the 9th International Workshop on Non-Monotonic Reasoning (NMR 2002) (2002)
96. Eiter, T., Faber, W., Leone, N., Pfeifer, G.: Declarative Problem-Solving Using the DLV System. In: Minker, J. (ed.) *Logic-Based Artificial Intelligence*, pp. 79–103. Kluwer Academic Publishers (2000)

97. Niemelä, I., Simons, P.: Smodels – An Implementation of the Stable Model and Well-founded Semantics for Normal Logic Programs. In: Fuhrbach, U., Dix, J., Nerode, A. (eds.) LPNMR 1997. LNCS (LNAI), vol. 1265, pp. 420–429. Springer, Heidelberg (1997)
98. Syrjänen, T.: Lparse 1.0 User's Manual (2002), <http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz>
99. Radziszowski, S.P.: Small Ramsey Numbers. *The Electronic Journal of Combinatorics* 1 (1994) (revision 9: July 15, 2002)
100. Papadimitriou, C.H.: *Computational Complexity*. Addison-Wesley (1994)
101. Cadoli, M., Eiter, T., Gottlob, G.: Default Logic as a Query Language. *IEEE Transactions on Knowledge and Data Engineering* 9(3), 448–463 (1997)
102. Simons, P.: Smodels Homepage (since 1996), <http://www.tcs.hut.fi/Software/smodels/>
103. Zhao, Y.: ASSAT homepage (since 2002), <http://assat.cs.ust.hk/>
104. Lin, F., Zhao, Y.: ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers. In: *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 2002)*, Edmonton, Alberta, Canada. AAAI Press/MIT Press (2002)
105. Babovich, Y., Maratea, M.: Cmodels-2: Sat-based answer sets solver enhanced to non-tight programs (2003), <http://www.cs.utexas.edu/users/tag/cmodels.html>
106. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-driven answer set solving. In: *Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pp. 386–392. Morgan Kaufmann Publishers (January 2007)
107. Janhunen, T., Niemelä, I., Seipel, D., Simons, P., You, J.H.: Unfolding Partiality and Disjunctions in Stable Model Semantics. Technical Report cs.AI/0303009, arXiv.org (March 2003)
108. Lierler, Y.: CMODELS – SAT-Based Disjunctive Answer Set Solver. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 447–451. Springer, Heidelberg (2005)
109. Drescher, C., Gebser, M., Grote, T., Kaufmann, B., König, A., Ostrowski, M., Schaub, T.: Conflict-Driven Disjunctive Answer Set Solving. In: Brewka, G., Lang, J. (eds.) *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*, Sydney, Australia, pp. 422–432. AAAI Press (2008)
110. Greco, S.: Binding Propagation Techniques for the Optimization of Bound Disjunctive Queries. *IEEE Transactions on Knowledge and Data Engineering* 15(2), 368–385 (2003)
111. Cumbo, C., Faber, W., Greco, G., Leone, N.: Enhancing the magic-set method for disjunctive datalog programs. In: Demoen, B., Lifschitz, V. (eds.) *ICLP 2004*. LNCS, vol. 3132, pp. 371–385. Springer, Heidelberg (2004)
112. Faber, W., Greco, G., Leone, N.: Magic sets and their application to data integration. In: Eiter, T., Libkin, L. (eds.) *ICDT 2005*. LNCS, vol. 3363, pp. 306–320. Springer, Heidelberg (2005)
113. Faber, W., Greco, G., Leone, N.: Magic Sets and their Application to Data Integration. *Journal of Computer and System Sciences* 73(4), 584–609 (2007)
114. Alviano, M., Faber, W., Greco, G., Leone, N.: Magic sets for disjunctive datalog programs. *Artificial Intelligence* 187–187, 156–192 (2012)
115. Alviano, M., Faber, W.: Dynamic magic sets and super-coherent answer set programs. *AI Communications – The European Journal on Artificial Intelligence* 24(2), 125–145 (2011)
116. Hustadt, U., Motik, B., Sattler, U.: Reducing SHIQ-description logic to disjunctive datalog programs. In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR 2004)*, Whistler, Canada, pp. 152–162 (2004)
117. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive datalog. *Journal of Automated Reasoning* 39(3), 351–384 (2007)

118. Leone, N., Gottlob, G., Rosati, R., Eiter, T., Faber, W., Fink, M., Greco, G., Ianni, G., Kalka, E., Lembo, D., Lenzerini, M., Lio, V., Nowicki, B., Ruzzi, M., Staniszczak, W., Terracina, G.: The INFOMIX System for Advanced Integration of Incomplete and Inconsistent Data. In: Proceedings of the 24th ACM SIGMOD International Conference on Management of Data (SIGMOD 2005), Baltimore, Maryland, USA, pp. 915–917. ACM Press (June 2005)
119. Lembo, D., Lenzerini, M., Rosati, R.: Source Inconsistency and Incompleteness in Data Integration. In: Proceedings of the Knowledge Representation meets Databases International Workshop (KRDB 2002), Toulouse France, CEUR Electronic Workshop Proceedings (2002), <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-54/>
120. Lembo, D., Lenzerini, M., Rosati, R.: Integrating Inconsistent and Incomplete Data Sources. In: Proceedings of SEBD 2002, Portoferraio, Isola d'Elba, pp. 299–308 (2002)
121. Soininen, T., Niemelä, I.: Developing a Declarative Rule Language for Applications in Product Configuration. In: Gupta, G. (ed.) PADL 1999. LNCS, vol. 1551, pp. 305–319. Springer, Heidelberg (1999)
122. A Rule-Based Formal Model for Software Configuration. Technical Report A55, Digital Systems Laboratory, Department of Computer Science, Helsinki University of Technology, Espoo, Finland (1999)
123. Ruffolo, M., Leone, N., Manna, M., Saccà, D., Zavatto, A.: Exploiting ASP for Semantic Information Extraction. In: de Vos, M., Proveti, A. (eds.) Proceedings ASP 2005 - Answer Set Programming: Advances in Theory and Implementation, Bath, UK, pp. 248–262 (July 2005)
124. Cumbo, C., Iiritano, S., Rullo, P.: Reasoning-Based Knowledge Extraction for Text Classification. In: Proceedings of Discovery Science, 7th International Conference, Padova, Italy, pp. 380–387 (October 2004)
125. Curia, R., Ettorre, M., Gallucci, L., Iiritano, S., Rullo, P.: Textual Document Pre-Processing and Feature Extraction in OLEX. In: Proceedings of Data Mining 2005, Skiathos, Greece (2005)
126. Aiello, L.C., Massacci, F.: Verifying security protocols as planning in logic programming. *ACM Transactions on Computational Logic* 2(4), 542–580 (2001)
127. Baral, C., Uyan, C.: Declarative Specification and Solution of Combinatorial Auctions Using Logic Programming. In: Eiter, T., Faber, W., Truszczyński, M. (eds.) LPNMR 2001. LNCS (LNAI), vol. 2173, pp. 186–199. Springer, Heidelberg (2001)
128. Faber, W., Leone, N., Pfeifer, G.: Representing School Timetabling in a Disjunctive Logic Programming Language. In: Egly, U., Tompits, H. (eds.) Proceedings of the 13th Workshop on Logic Programming (WLP 1998), Vienna, Austria, pp. 43–52 (October 1998)
129. Bertino, E., Mileo, A., Proveti, A.: User Preferences VS Minimality in PDDL. In: Buccafurri, F. (ed.): Proceedings of the Joint Conference on Declarative Programming APPIA-GULP-PRODE 2003, pp. 110–122 (September 2003)
130. Greco, G., Guzzo, A., Saccà, D.: A Logic Programming Approach for Planning Workflows Evolutions. In: Buccafurri, F. (ed.): Proceedings of the Joint Conference on Declarative Programming APPIA-GULP-PRODE 2003, pp. 75–85 (September 2003)
131. Greco, G., Greco, S., Zumpano, E.: A Logical Framework for Querying and Repairing Inconsistent Databases. *IEEE Transactions on Knowledge and Data Engineering* 15(6), 1389–1408 (2003)
132. Erdem, E., Lifschitz, V., Nakhleh, L., Ringe, D.: Reconstructing the Evolutionary History of Indo-European Languages Using Answer Set Programming. In: Dahl, V. (ed.) PADL 2003. LNCS, vol. 2562, pp. 160–176. Springer, Heidelberg (2002)
133. Buccafurri, F., Caminiti, G.: A Social Semantics for Multi-agent Systems. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 317–329. Springer, Heidelberg (2005)

134. Costantini, S., Tocchio, A.: The DALI logic programming agent-oriented language. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 685–688. Springer, Heidelberg (2004)
135. Garro, A., Palopoli, L., Ricca, F.: Exploiting agents in e-learning and skills management context. *AI Communications – The European Journal on Artificial Intelligence* 19(2), 137–154 (2006)
136. De Vos, M., Schaub, T. (eds.): SEA 2007: Software Engineering for Answer Set Programming, vol. 281. CEUR (2007), <http://CEUR-WS.org/Vol-281/>
137. Brain, M., De Vos, M.: Debugging Logic Programs under the Answer Set Semantics. In: de Vos, M., Proveti, A. (eds.) Proceedings ASP 2005 - Answer Set Programming: Advances in Theory and Implementation, Bath, UK (July 2005)
138. El-Khatib, O., Pontelli, E., Son, T.C.: Justification and debugging of answer set programs in ASP. In: Jeffery, C., Choi, J.D., Lencevicius, R. (eds.) Proceedings of the Sixth International Workshop on Automated Debugging, California, USA, ACM (September 2005)
139. Ricca, F.: The DLV Java Wrapper. In: de Vos, M., Proveti, A. (eds.) Proceedings ASP 2003 - Answer Set Programming: Advances in Theory and Implementation, Messina, Italy, pp. 305–316 (September 2003), <http://CEUR-WS.org/Vol-78/>
140. McCarthy, J.: Formalization of Common Sense, papers by John McCarthy, edited by V. Lifschitz. Ablex (1990)
141. Minker, J. (ed.): Foundations of Deductive Databases and Logic Programming. Morgan Kaufmann Publishers, Inc., Washington, DC (1988)
142. Dix, J., Gottlob, G., Marek, V.W.: Causal Models for Disjunctive Logic Programs. In: Van Hentenryck, P. (ed.) Proceedings of the 11th International Conference on Logic Programming (ICLP 1994), Santa Margherita Ligure, Italy, pp. 290–302. MIT Press (June 1994)

Ontology-Based Data Access with Databases: A Short Course

Roman Kontchakov¹, Mariano Rodríguez-Muro², and Michael Zakharyashev¹

¹ Department of Computer Science and Information Systems,
Birkbeck, University of London, U.K.

² Faculty of Computer Science, Free University of Bozen-Bolzano, Italy

Abstract. Ontology-based data access (OBDA) is regarded as a key ingredient of the new generation of information systems. In the OBDA paradigm, an ontology defines a high-level global schema of (already existing) data sources and provides a vocabulary for user queries. An OBDA system rewrites such queries and ontologies into the vocabulary of the data sources and then delegates the actual query evaluation to a suitable query answering system such as a relational database management system or a datalog engine. In this chapter, we mainly focus on OBDA with the ontology language *OWL 2 QL*, one of the three profiles of the W3C standard Web Ontology Language *OWL 2*, and relational databases, although other possible languages will also be discussed. We consider different types of conjunctive query rewriting and their succinctness, different architectures of OBDA systems, and give an overview of the OBDA system *Ontop*.

1 Introduction

Do you like movies? Do you want to know more about stars, directors, writers, casts, etc.? Then you should probably query IMDb, the Internet Movie Database available at www.imdb.com. You do not know how to use databases. But you have already taken the ‘Introduction to Description Logics’ course. Then what you need is ontology-based data access (OBDA, for short). There is a simple movie ontology MO at www.movieontology.org describing the application domain in terms of concepts (classes), such as *mo:Movie* and *mo:Person*, and roles and attributes (object and datatype properties), such as *mo:cast* and *mo:year*:

$$\begin{aligned} mo:Movie &\equiv \exists mo:title, & mo:Movie &\sqsubseteq \exists mo:year, \\ mo:Movie &\equiv \exists mo:cast, & \exists mo:cast^- &\sqsubseteq mo:Person, \quad \text{etc.} \end{aligned}$$

And you can query the IMDb data in terms of concepts and roles of the MO ontology; for example,

$$q(t, y) = \exists m (mo:Movie(m) \wedge mo:title(m, t) \wedge mo:year(m, y) \wedge (y > 2010))$$

is a conjunctive query asking for the titles (the variable t) of recent movies with their production year (the variable y). An OBDA system such as *Ontop*

available at ontop.inf.unibz.it will automatically rewrite your query into the language of IMDb, optimise the rewriting and use a conventional relational database management system (RDBMS) to find the answers. (We will return to the IMDb example in Section 6.)

The idea of OBDA was explicitly formulated in 2008 [16,26,54], though query answering over description logic knowledge bases has been investigated since at least 2005. Nowadays, OBDA is often deemed to be an important ingredient of the new generation of information systems as it (i) gives a high-level conceptual view of the data, (ii) provides the user with a convenient vocabulary for queries, (iii) allows the system to enrich incomplete data with background knowledge, and (iv) supports queries to multiple and possibly heterogeneous data sources.

One can distinguish between several types of OBDA depending on the expressive power of description logics (DLs).

OBDA with databases: some DLs, such as the logics of the *DL-Lite* family (and *OWL 2 QL*), allow a reduction of conjunctive queries over ontologies to first-order queries over standard relational databases [11,54,4,36];

OBDA with datalog engines: other DLs encompassing logics in the \mathcal{EL} family (*OWL 2 EL*), Horn-*SHIQ* and Horn-*SROIQ*, support a datalog reduction and can be used with datalog engines [63,44,50,17];

OBDA with expressive DLs such as *ALC* or *SHIQ* require some special techniques for answering conjunctive queries; see [27,45,49,21,18,13,33] and references therein for details.

In this chapter, we give a brief and easy introduction to the theory and practice of OBDA with relational databases, assuming that the reader has some basic knowledge of description logic. Our plan is as follows. In Section 2, we introduce and discuss the DLs supporting first-order rewritability of conjunctive queries. Then, in Section 3, we show how to compute first-order and nonrecursive datalog rewritings of conjunctive queries over *OWL 2 QL* ontologies. The size of rewritings is discussed in Section 4. In Section 5, we introduce the basics of the combined approach to OBDA. Finally, in Section 6, we present the OBDA system *Ontop*, which is available as a plugin for the Protégé 4 ontology editor as well as OWLAPI and Sesame libraries and a SPARQL end-point.

2 Description Logics for OBDA with Databases

The key notion of OBDA with databases is query rewriting. The user formulates a query q in the vocabulary of a given ontology \mathcal{T} . (Such a pair (\mathcal{T}, q) is sometimes called an *ontology-mediated query*.) The task of an OBDA system is to ‘rewrite’ q and \mathcal{T} into a new query q' in the vocabulary of the data such that, for any possible data \mathcal{A} (in this vocabulary), the answers to q over $(\mathcal{T}, \mathcal{A})$ are precisely the same as the answers to q' over \mathcal{A} . Thus, the problem of querying data \mathcal{A} (the structure of which is not known to the user) in terms of the ontology \mathcal{T} (accessible to the user) is reduced to the problem of querying \mathcal{A} directly. As witnessed by the 40 years history of relational databases, RDBMSs are usually

very efficient in query evaluation. Other query answering systems, for example datalog engines can also be employed. In this section, we consider the ontology languages supporting query rewriting. To be more focused, we concentrate on description logics (DLs) as ontology formalisms and only provide the reader with references to languages of other types. We also assume in this section that \mathcal{A} is simply a DL ABox stored in a relational database (proper databases will be considered in Section 6).

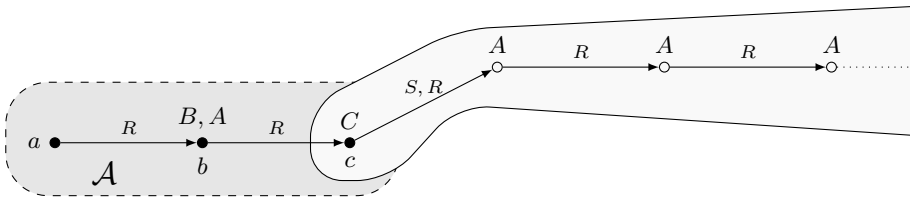
Example 1. Consider the query $q(x) = \exists y (R(x, y) \wedge A(y))$ that asks for the individuals x in the ABox such that $R(x, y)$ and $A(y)$, for some y (which does not have to be an ABox individual). Suppose also that we are given the DL ontology (or TBox)

$$\mathcal{T} = \{ B \sqsubseteq A, C \sqsubseteq \exists S, \exists S^- \sqsubseteq A, \exists R^- \sqsubseteq \exists R, S \sqsubseteq R \},$$

where A and B are concept names and R and S are role names (thus, the second axiom says that C is a subset of the domain of S , and the third that the range of S is a subset of A). It takes a moment's thought to see that we obtain $R(x, y)$, for some y , if we have one of $R(x, y)$, $S(x, y)$ or $C(x)$ (in the last case y may not even exist among the ABox individuals). It follows from \mathcal{T} that, in the second case, $A(y)$ also holds, and, in the third case, there exists some y such that $R(x, y)$ and $A(y)$. Similarly, we obtain $A(y)$ if we have one of $A(y)$, $B(y)$ or $S(z, y)$, for some z . These observations give the following first-order rewriting of q and \mathcal{T} :

$$q'(x) = \exists y [R(x, y) \wedge (A(y) \vee B(y) \vee \exists z S(z, y))] \vee \exists y S(x, y) \vee C(x).$$

Now, suppose $\mathcal{A} = \{ R(a, b), B(b), R(b, c), C(c) \}$. It is easy to compute the answers to $q'(x)$ over \mathcal{A} : these are $x = a$ and $x = c$. What about the answers to q over $(\mathcal{T}, \mathcal{A})$? We can compute them by first applying the axioms of \mathcal{T} to \mathcal{A} , always creating *fresh* witnesses for the existential quantifiers $\exists R$ and $\exists S$ if they do not already exist, and then evaluating $q(x)$ over the resulting (possibly infinite) structure known as the *canonical model* of $(\mathcal{T}, \mathcal{A})$. The canonical model is illustrated in the picture below (where the fresh witnesses are shown as \circ). Taking into consideration that we only need answers from the data (the individuals from \mathcal{A}), we see again that they are $x = a$ and $x = c$.



Formulas such as $q(x)$ in Example 1 are called *conjunctive queries* (CQs, for short). CQs are the most basic type of database queries, also known as SELECT-PROJECT-JOIN SQL queries. More precisely, in the context of OBDA over DL ontologies, a CQ $q(x)$ is a first-order formula $\exists y \varphi(x, y)$, where φ is a conjunction of atoms of the form $A(t_1)$ or $P(t_1, t_2)$, and each t_i is a *term* (an individual or

a variable in \mathbf{x} or \mathbf{y}). The variables in \mathbf{x} are called *answer variables* and those in \mathbf{y} *existentially quantified variables*. Formulas such as $\mathbf{q}'(x)$ in Example 1 can also use disjunctions and are called *positive existential queries* (PE-queries). If all Boolean connectives (conjunction, disjunction and negation) as well as both quantifiers, \forall and \exists , are allowed then we call the queries *first-order* (FO-queries). FO-queries (more precisely, domain-independent FO-queries) roughly correspond to the class of queries expressible in SQL. A query $\mathbf{q}(\mathbf{x})$ is called *Boolean* if $\mathbf{x} = \emptyset$.

A tuple \mathbf{a} of individuals in \mathcal{A} (of the same length as \mathbf{x}) is a *certain answer* to $\mathbf{q}(\mathbf{x})$ over $(\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \mathbf{q}(\mathbf{a})$ for all models \mathcal{I} of $(\mathcal{T}, \mathcal{A})$; in this case we write $(\mathcal{T}, \mathcal{A}) \models \mathbf{q}(\mathbf{a})$. In other words, \mathbf{a} is a certain answer to $\mathbf{q}(\mathbf{x})$ over $(\mathcal{T}, \mathcal{A})$ if $\mathbf{q}(\mathbf{a})$ follows logically from \mathcal{A} and \mathcal{T} . For a Boolean \mathbf{q} , the certain answer is ‘yes’ if \mathbf{q} holds in all models of $(\mathcal{T}, \mathcal{A})$, and ‘no’ otherwise. Finally, an *FO-rewriting* of $\mathbf{q}(\mathbf{x})$ and \mathcal{T} is an FO-query $\mathbf{q}'(\mathbf{x})$ such that $(\mathcal{T}, \mathcal{A}) \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{A} \models \mathbf{q}'(\mathbf{a})$, for any ABox \mathcal{A} and any tuple $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, where $\text{ind}(\mathcal{A})$ is the set of individuals in \mathcal{A} .

For the purposes of OBDA we are naturally interested in maximal ontology languages that ensure FO-rewritability of CQs. To distinguish between DLs with and without FO-rewritability, we require a few facts from the computational complexity theory. For details, the reader is referred to [3,28,38,42]. The hierarchy of the complexity classes we use in this chapter is shown below:

$$\text{AC}^0 \subsetneq \text{NLOGSPACE} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}.$$

It is also known that $\text{P} \subsetneq \text{EXPTIME}$ and $\text{NLOGSPACE} \subsetneq \text{PSPACE}$ (whether the remaining inclusions are proper is a major open problem in computer science). Consider, for example, the problem of evaluating a (Boolean) FO-query over relational databases: given an ABox \mathcal{A} and a query \mathbf{q} , decide whether $\mathcal{A} \models \mathbf{q}$. If both \mathcal{A} and \mathbf{q} are taken as an input, then the problem is PSPACE-complete for FO-queries (due to alternation of quantifiers), and NP-complete for CQs and PE-queries. In this case, we refer to the *combined complexity* of query answering. If only the ABox is an input (and the query is fixed), the problem is in AC^0 in *data complexity* (this is regarded as an appropriate measure in databases because the database instance is much smaller than the query).

Returning to FO-rewritability of CQs \mathbf{q} and DL TBoxes \mathcal{T} , we observe that if the problem ‘ $(\mathcal{T}, \mathcal{A}) \models \mathbf{q}$?’ is at least NLOGSPACE-hard for data complexity (that is, with fixed \mathcal{T} and \mathbf{q}), then \mathbf{q} and \mathcal{T} cannot be FO-rewritable. Indeed, if there was an FO-rewriting then this problem could be solved in AC^0 . This observation allows us to delimit the DL constructs that ruin FO-rewritability.

Example 2. A typical example of an NLOGSPACE-complete problem is the *reachability problem* for directed graphs: given a directed graph $G = (V, R)$ with vertices V and arcs R and two distinguished vertices $s, t \in V$, decide whether there is a directed path from s to t in G . We represent the input by means of the ABox

$$\mathcal{A}_{G,s,t} = \{R(v_1, v_2) \mid (v_1, v_2) \in R\} \cup \{A(s), B(t)\}.$$

Consider now the following TBox and Boolean CQ

$$\mathcal{T} = \{ \exists R.B \sqsubseteq B \}, \quad \mathbf{q} = \exists y (A(y) \wedge B(y)).$$

It is readily seen that $(\mathcal{T}, \mathcal{A}_{G,s,t}) \models \mathbf{q}$ iff there is a path from s to t in G . As \mathcal{T} and \mathbf{q} do not depend on G , s and t , the problem ‘ $(\mathcal{T}, \mathcal{A}_{G,s,t}) \models \mathbf{q}$?’ is NLOGSPACE-hard for data complexity, and so \mathbf{q} and \mathcal{T} cannot be FO-rewritable.

In other words, TBoxes capable of computing the transitive closure of some relations in ABoxes do not allow FO-rewritability.

Example 3. The *path system accessibility* problem is an example of a P-complete problem [20]: given a finite set V of vertices and a relation $E \subseteq V \times V \times V$ with a set of source vertices $S \subseteq V$ and a terminal vertex $t \in V$, decide whether t is accessible, where all $v \in S$ are accessible, and if $(v_1, v_2, v) \in E$, for accessible inputs v_1 and v_2 , then v is also accessible. The path system can be encoded by an ABox \mathcal{A} in the following way:

$$\{ A(v) \mid v \in S \} \cup \{ P_1(e, v_1), P_2(e, v_2), R(v, e) \mid e = (v_1, v_2, v) \in E \}.$$

Consider now the TBox \mathcal{T} and Boolean CQ \mathbf{q} given by

$$\mathcal{T} = \{ \exists P_1.A \sqcap \exists P_2.A \sqsubseteq B, \exists R.B \sqsubseteq A \}, \quad \mathbf{q} = A(t).$$

It should be clear that $(\mathcal{T}, \mathcal{A}) \models A(v)$ iff v is accessible (and that $(\mathcal{T}, \mathcal{A}) \models B(e)$ iff *both* inputs of e are accessible, that is, both belong to A). Therefore, the answer to \mathbf{q} is ‘yes’ iff t is accessible. Thus, the problem ‘ $(\mathcal{T}, \mathcal{A}) \models \mathbf{q}$ ’ is P-hard for data complexity, and so \mathbf{q} and \mathcal{T} cannot be FO-rewritable.

Note that the TBox \mathcal{T} in Example 3 is formulated in the DL \mathcal{EL} .

Example 4. A good example of an NP-complete problem is *graph 3-colouring*: given an (undirected) graph $G = (V, E)$, decide whether each of its vertices can be painted in one of three given colours in such a way that no adjacent vertices have the same colour. We represent the input graph G by means of the ABox

$$\mathcal{A}_G = \{ R(v_1, v_2) \mid \{v_1, v_2\} \in E \}.$$

Consider now the Boolean CQ $\mathbf{q} = \exists y N(y)$ and the following TBox

$$\mathcal{T} = \{ \top \sqsubseteq C_1 \sqcup C_2 \sqcup C_3 \} \cup \{ C_i \sqcap C_j \sqsubseteq N \mid 1 \leq i < j \leq 3 \} \cup \{ C_i \sqcap \exists R.C_i \sqsubseteq N \mid 1 \leq i \leq 3 \},$$

where the C_i are concept names representing the given three colours. It is not hard to see that the answer to \mathbf{q} over $(\mathcal{T}, \mathcal{A}_G)$ is ‘no’ iff G is 3-colourable. It follows that the problem ‘ $(\mathcal{T}, \mathcal{A}_G) \models \mathbf{q}$?’ is coNP-hard for data complexity, and so \mathbf{q} and \mathcal{T} are not FO-rewritable.

The moral of this example is that to allow FO-rewritability, TBoxes should not contain axioms with disjunctive information (which can be satisfied in essentially different ways when applied to ABoxes). The TBox in Example 4 is formulated in the DL \mathcal{ALC} .

The data complexity of answering CQs over ontologies formulated in various DLs has been intensively investigated since 2005; see, e.g., [12,39,11,49,4]. Thus, answering CQs over \mathcal{EL} ontologies is P-complete for data complexity, while for \mathcal{ALC} it is coNP-hard. One of the results of this research was the inclusion in the current W3C standard Web Ontology Language *OWL 2* of a special sublanguage (or profile) that is suitable for OBDA with databases and called *OWL 2 QL*. The DLs underlying *OWL 2 QL* belong to the so-called *DL-Lite* family [11,4]. Below, we present *OWL 2 QL* in the DL parlance rather than the *OWL 2* syntax.

The language of *OWL 2 QL* contains *individual names* a_i , *concept names* A_i , and *role names* P_i ($i = 1, 2, \dots$). *Roles* R , *basic concepts* B and *concepts* C are defined by the grammar:

$$\begin{aligned} R & ::= P_i \mid P_i^-, \\ B & ::= \perp \mid A_i \mid \exists R, \\ C & ::= B \mid \exists R.B \end{aligned}$$

(here P_i^- is the inverse of P_i and $\exists R$ is regarded as an abbreviation for $\exists R.\top$). An *OWL 2 QL TBox*, \mathcal{T} , is a finite set of *concept* and *role inclusions* of the form

$$B \sqsubseteq C, \quad R_1 \sqsubseteq R_2$$

and *concept* and *role disjointness constraints* of the form

$$B_1 \sqcap B_2 \sqsubseteq \perp, \quad R_1 \sqcap R_2 \sqsubseteq \perp.$$

Apart from this, \mathcal{T} may contain assertions stating that certain roles P_i are reflexive and irreflexive. Note that symmetry and asymmetry of a role R can be expressed in *OWL 2 QL* as, respectively,

$$R \sqsubseteq R^- \quad \text{and} \quad R \sqcap R^- \sqsubseteq \perp.$$

An *OWL 2 QL ABox*, \mathcal{A} , is a finite set of *assertions* of the form $A_k(a_i)$ and $P_k(a_i, a_j)$ and *inequality constraints* $a_i \neq a_j$ for $i \neq j$. \mathcal{T} and \mathcal{A} together constitute the *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.

It is to be noted that concepts of the form $\exists R.B$ can only occur in the right-hand side of concept inclusions in *OWL 2 QL*. An inclusion $B' \sqsubseteq \exists R.B$ can be regarded as an abbreviation for three inclusions:

$$B' \sqsubseteq \exists R_B, \quad \exists R_B^- \sqsubseteq B \quad \text{and} \quad R_B \sqsubseteq R,$$

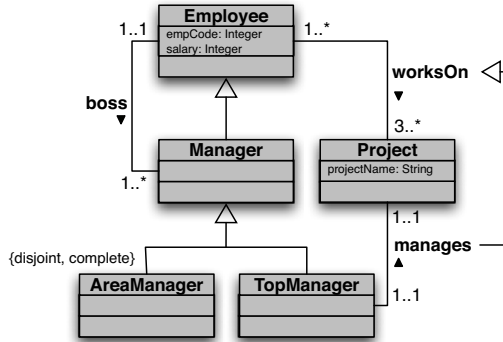
where R_B is a fresh role name. Thus, inclusions of the form $B' \sqsubseteq \exists R.B$ are just convenient syntactic sugar. To simplify presentation, in the remainder of this chapter we consider the sugar-free *OWL 2 QL*, assuming that every concept inclusion is of the form $B_1 \sqsubseteq B_2$, where both B_1 and B_2 are basic concepts.

Unlike standard DLs, *OWL 2* does not adopt the *unique name assumption* (UNA, for short) according to which, for any interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and any distinct individual names a_i and a_j , we must have $a_i^{\mathcal{I}} \neq a_j^{\mathcal{I}}$. That is why *OWL 2 QL* has inequality constraints $a_i \neq a_j$ in its syntax.

Theorem 1 ([11,4]). (i) *The satisfiability problem for OWL 2 QL knowledge bases is NLOGSPACE-complete for combined complexity and in AC⁰ for data complexity.*

(ii) *CQs and OWL 2 QL TBoxes are FO-rewritable, and so CQ answering over OWL 2 QL TBoxes is in AC⁰ for data complexity.*

We will explain how one can compute FO-rewritings for CQs over *OWL 2 QL* TBoxes in the next section. Meanwhile, we are going to illustrate the expressive power of the language and discuss whether it can be extended without losing FO-rewritability. The following example shows what can and what cannot be represented in *OWL 2 QL* TBoxes. As the primary aim of *OWL 2 QL* is to facilitate OBDA with relational databases, our example is from the area of conceptual data modelling.



Example 5. Consider the UML class diagram in the picture above representing (part of) a company information system. According to the diagram, all managers are employees and are partitioned into area managers and top managers. In *OWL 2 QL*, we can write

$$\begin{aligned}
 & \text{Manager} \sqsubseteq \text{Employee}, \\
 & \text{AreaManager} \sqsubseteq \text{Manager}, \quad \text{TopManager} \sqsubseteq \text{Manager}, \\
 & \text{AreaManager} \sqcap \text{TopManager} \sqsubseteq \perp.
 \end{aligned}$$

However, the covering constraint $\text{Manager} \sqsubseteq \text{AreaManager} \sqcup \text{TopManager}$ uses union \sqcup , which is not allowed in *OWL 2 QL*. Each employee has two attributes, *empCode* and *salary*, with integer values. Unlike OWL, here we do not distinguish between abstract objects and data values (there are, however, approaches based on concrete domains [5,64]). Hence we model a datatype, such as *Integer*, by a concept, and an attribute, such as employee’s salary, by a role:

$$\text{Employee} \sqsubseteq \exists \text{salary}, \quad \exists \text{salary}^- \sqsubseteq \text{Integer}.$$

However, the constraint $\geq 2 \textit{salary} \sqsubseteq \perp$ (saying that the attribute *salary* is functional) is not allowed. The attribute *empCode* with values in *Integer* is represented in the same way. The binary relation *worksOn* has *Employee* as its domain and *Project* as its range:

$$\exists \textit{worksOn} \sqsubseteq \textit{Employee}, \quad \exists \textit{worksOn}^- \sqsubseteq \textit{Project}.$$

The relation *boss* with domain *Employee* and range *Manager* is treated similarly. Of the constraints that each employee works on a project and has exactly one boss, and a project must involve at least three employees, we can only capture the following:

$$\textit{Employee} \sqsubseteq \exists \textit{worksOn}, \quad \textit{Employee} \sqsubseteq \exists \textit{boss}.$$

Both cardinality constraints $\geq 2 \textit{boss} \sqsubseteq \perp$ and $\textit{Project} \sqsubseteq \geq 3 \textit{worksOn}^-$ require a more powerful language. Finally, we have to say that a top manager manages exactly one project and also works on that project, while a project is managed by exactly one top manager. In *OWL 2 QL*, we can only write:

$$\begin{aligned} \exists \textit{manages} \sqsubseteq \textit{TopManager}, & \quad \exists \textit{manages}^- \sqsubseteq \textit{Project}, \\ \textit{TopManager} \sqsubseteq \exists \textit{manages}, & \quad \textit{Project} \sqsubseteq \exists \textit{manages}^-, \\ \textit{manages} \sqsubseteq \textit{worksOn}, & \end{aligned}$$

but not $\geq 2 \textit{manages} \sqsubseteq \perp$ and $\geq 2 \textit{manages}^- \sqsubseteq \perp$. We cannot, obviously, represent constraints such as $\textit{CEO} \sqcap (\geq 5 \textit{worksOn}) \sqcap \exists \textit{manages} \sqsubseteq \perp$ (no CEO may work on five projects and be a manager of one of them) either.

As we saw in the example above, some constructs that are important for conceptual modelling are not available in *OWL 2 QL*. Can we add these constructs to the language without destroying FO-rewritability?

Let us recall from Example 2 that we cannot extend *OWL 2 QL* with the construct $\exists R.B$ in the left-hand side of concept inclusions or with transitivity constraints (stating that certain roles are interpreted by transitive relations). Example 4 shows that \sqcup in the right-hand side can be dangerous. On the other hand, we can safely use concept and role inclusions with \sqcap in the left-hand side:

$$B_1 \sqcap \dots \sqcap B_n \sqsubseteq B, \quad R_1 \sqcap \dots \sqcap R_m \sqsubseteq R, \quad \text{for } m, n \geq 1.$$

Unqualified number restrictions $\geq k R$, for $k \geq 2$, are a bit trickier. As *OWL 2 QL* does not adopt the UNA, an axiom such as $(\geq 3 R \sqsubseteq \perp)$ over an ABox containing the atoms $R(a, b_i)$, for $i \geq 3$, means that some of the b_i must coincide, and there are various ways to make it so by identifying some of the b_i . In fact, unqualified number restrictions added to *OWL 2 QL* make it coNP-hard for data complexity; and even role functionality makes it P-hard for data complexity [4].

One can argue, however, that in the context of OBDA it is more natural and important to adopt the UNA. Indeed, after all, databases do respect the UNA. It turns out that if we stipulate that the UNA is respected in *OWL 2 QL*, then

unqualified number restrictions are ‘harmless’ provided that we do not use both axioms ($\geq k R \sqsubseteq B$), for $k \geq 2$, and $R' \sqsubseteq R$ in the same TBox (consult [4] for a more precise condition).

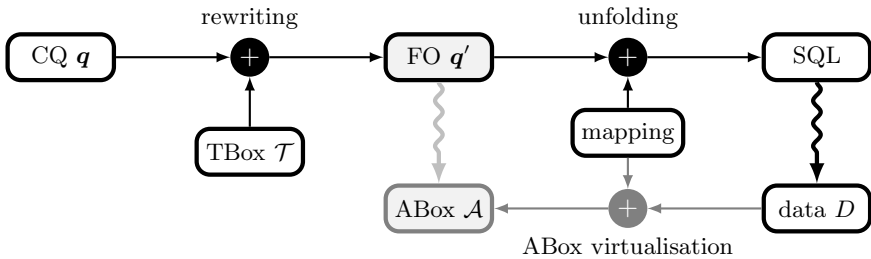
The results discussed so far guarantee that if our ontology is formulated in a DL with the help of such and such constructs then it can safely be used for OBDA with databases. A different approach to understanding the phenomenon of FO-rewritability has recently been suggested [47]: it attempts to classify all TBoxes \mathcal{T} in some master DL, say \mathcal{ALCT} , according to the complexity of answering CQs over \mathcal{T} .

Finally, we note that another family of ontology languages suitable for OBDA with databases has been designed by the datalog community. We refer the reader to the recent papers [10,9] for a survey. Connections of query answering via DL ontologies with disjunctive datalog and constraint satisfaction problems have been established in [7].

In the next section, we shall see how one can construct FO-rewritings of CQs and $OWL\ 2\ QL$ TBoxes.

3 Tree-Witness Rewriting

A standard architecture of an OBDA system over relational data sources can be represented as follows:



The user is given an $OWL\ 2\ QL$ TBox \mathcal{T} and can formulate CQs $q(\mathbf{x})$ in the signature of \mathcal{T} . The system rewrites $q(\mathbf{x})$ and \mathcal{T} into an FO-query $q'(\mathbf{x})$ such that $(\mathcal{T}, \mathcal{A}) \models q(\mathbf{a})$ iff $\mathcal{A} \models q'(\mathbf{a})$, for any ABox \mathcal{A} and any tuple \mathbf{a} of individuals in \mathcal{A} . The rewriting q' is called a *PE-rewriting* if it is a PE-query and an *NDL-rewriting* if it is an NDL-query.

The rewriting $q'(\mathbf{x})$ is formulated in the signature of \mathcal{T} and has to be further transformed into the vocabulary of the data source D before being evaluated. For instance, $q'(\mathbf{x})$ can be unfolded into an SQL query by means of a mapping \mathcal{M} relating the signature of \mathcal{T} to the vocabulary of D . We consider unfolding in Section 6, but before that we assume the data to be given as an ABox (say, as a universal table in a database or as a triple store) with a trivial mapping.

A number of different rewriting techniques have been proposed and implemented for $OWL\ 2\ QL$ (PerfectRef [54], Presto/Prexto [62,61], Rapid [15]) and its extensions ([35], Nyaya [22], Requiem/Blackout [52,53], Clipper [17]). In this section, we discuss the tree-witness rewriting [32].

3.1 Canonical Model

All types of FO-rewritings are based on the fact that, if an *OWL 2 QL* KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent, then there is a special model of \mathcal{K} , called a *canonical model* and denoted $\mathcal{C}_{\mathcal{K}}$, such that $\mathcal{K} \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{C}_{\mathcal{K}} \models \mathbf{q}(\mathbf{a})$, for any CQ $\mathbf{q}(\mathbf{x})$ and any $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$. We have already seen one canonical model in Example 1. Intuitively, the construction of $\mathcal{C}_{\mathcal{K}}$ is pretty straightforward: we start with \mathcal{A} and then apply to it recursively the axioms of \mathcal{T} wherever possible. The only nontrivial case is when we apply an axiom of the form $B \sqsubseteq \exists R$ to some $w \in B$. Then either we already have an R -arrow starting from w , in which case we do not have to do anything, or such an R -arrow does not exist, and then we create a *fresh* individual, say w_R , in the model under construction and draw an R -arrow from w to w_R .

Formally, let

$$[R] = \{ S \mid \mathcal{T} \models R \sqsubseteq S \text{ and } \mathcal{T} \models S \sqsubseteq R \}.$$

We write $[R] \leq_{\mathcal{T}} [S]$ if $\mathcal{T} \models R \sqsubseteq S$; thus, $\leq_{\mathcal{T}}$ is a partial order on the set of equivalence classes $[R]$, for roles R in \mathcal{T} . For each $[R]$, we introduce a *witness* $w_{[R]}$ and define a *generating relation* $\rightsquigarrow_{\mathcal{K}}$ on the set of these witnesses together with $\text{ind}(\mathcal{A})$ by taking:

$a \rightsquigarrow_{\mathcal{K}} w_{[R]}$ if $a \in \text{ind}(\mathcal{A})$ and $[R]$ is $\leq_{\mathcal{T}}$ -minimal such that $\mathcal{K} \models \exists R(a)$ and $\mathcal{K} \not\models R(a, b)$ for any $b \in \text{ind}(\mathcal{A})$;

$w_{[S]} \rightsquigarrow_{\mathcal{K}} w_{[R]}$ if $[R]$ is $\leq_{\mathcal{T}}$ -minimal with $\mathcal{T} \models \exists S^- \sqsubseteq \exists R$ and $[S^-] \neq [R]$.

A \mathcal{K} -*path* is a finite sequence $aw_{[R_1]} \cdots w_{[R_n]}$, $n \geq 0$, such that $a \in \text{ind}(\mathcal{A})$ and, if $n > 0$, then $a \rightsquigarrow_{\mathcal{K}} w_{[R_1]}$ and $w_{[R_i]} \rightsquigarrow_{\mathcal{K}} w_{[R_{i+1}]}$, for $i < n$. Denote by $\text{tail}(\sigma)$ the last element in the path σ . The canonical model $\mathcal{C}_{\mathcal{K}} = (\Delta^{\mathcal{C}_{\mathcal{K}}}, \cdot^{\mathcal{C}_{\mathcal{K}}})$ is defined by taking $\Delta^{\mathcal{C}_{\mathcal{K}}}$ to be the set of all \mathcal{K} -paths and setting:

$$a^{\mathcal{C}_{\mathcal{K}}} = a, \text{ for all } a \in \text{ind}(\mathcal{A}),$$

$$A^{\mathcal{C}_{\mathcal{K}}} = \{ a \in \text{ind}(\mathcal{A}) \mid \mathcal{K} \models A(a) \} \cup$$

$$\{ \sigma \cdot w_{[R]} \mid \mathcal{T} \models \exists R^- \sqsubseteq A \}, \text{ for all concept names } A,$$

$$P^{\mathcal{C}_{\mathcal{K}}} = \{ (a, b) \in \text{ind}(\mathcal{A}) \times \text{ind}(\mathcal{A}) \mid R(a, b) \in \mathcal{A} \text{ with } [R] \leq_{\mathcal{T}} [P] \} \cup$$

$$\{ (\sigma, \sigma \cdot w_{[R]}) \mid \text{tail}(\sigma) \rightsquigarrow_{\mathcal{K}} w_{[R]}, [R] \leq_{\mathcal{T}} [P] \} \cup$$

$$\{ (\sigma \cdot w_{[R]}, \sigma) \mid \text{tail}(\sigma) \rightsquigarrow_{\mathcal{K}} w_{[R]}, [R] \leq_{\mathcal{T}} [P^-] \}, \text{ for all role names } P.$$

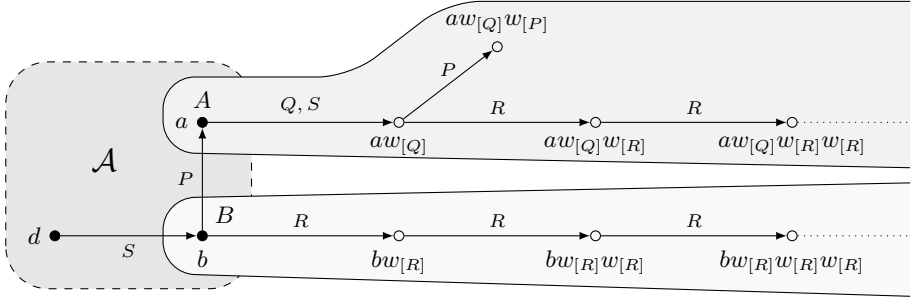
We call $\text{ind}(\mathcal{A})$ the *ABox part* of $\mathcal{C}_{\mathcal{K}}$, and $\Delta^{\mathcal{C}_{\mathcal{K}}} \setminus \text{ind}(\mathcal{A})$ the *anonymous part*.

Example 6. Consider the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where

$$\mathcal{T} = \{ A \sqsubseteq \exists Q, \exists Q^- \sqsubseteq B, Q \sqsubseteq S, \exists S^- \sqsubseteq \exists R, \exists R^- \sqsubseteq \exists R, B \sqsubseteq \exists P \},$$

$$\mathcal{A} = \{ A(a), B(b), P(b, a), S(d, b) \}.$$

The canonical model $\mathcal{C}_{\mathcal{K}}$ for \mathcal{K} is depicted below:



Theorem 2. *For any consistent OWL 2 QL KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, any CQ $\mathbf{q}(\mathbf{x})$ and any $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $\mathcal{K} \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{C}_{\mathcal{K}} \models \mathbf{q}(\mathbf{a})$.*

Thus, to compute certain answers to $\mathbf{q}(\mathbf{x})$ over $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, it is enough to find answers to $\mathbf{q}(\mathbf{x})$ in the canonical model $\mathcal{C}_{\mathcal{K}}$. To do this, we have to check all possible homomorphisms from \mathbf{q} to $\mathcal{C}_{\mathcal{K}}$ that map the answer variables \mathbf{x} to the ABox part of $\mathcal{C}_{\mathcal{K}}$. More precisely, let us regard $\mathbf{q}(\mathbf{x}) = \exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$ as simply the set of atoms in φ , so we can write $A(x) \in \mathbf{q}$, $P(x, y) \in \mathbf{q}$, etc. If $P(x, y) \in \mathbf{q}$ then we also write $P^-(y, x) \in \mathbf{q}$. By a *homomorphism* (or a *match*) from $\mathbf{q}(\mathbf{x})$ to $\mathcal{C}_{\mathcal{K}}$ we understand any map $h: \mathbf{x} \cup \mathbf{y} \cup \text{ind}(\mathcal{A}) \rightarrow \Delta^{\mathcal{C}_{\mathcal{K}}}$ such that the following conditions hold:

- $h(a) = a^{\mathcal{C}_{\mathcal{K}}}$, for every $a \in \text{ind}(\mathcal{A})$,
- $h(x) \in \text{ind}(\mathcal{A})$, for every x in \mathbf{x} ,
- if $A(z) \in \mathbf{q}$ then $h(z) \in A^{\mathcal{C}_{\mathcal{K}}}$,
- if $P(z, z') \in \mathbf{q}$ then $(h(z), h(z')) \in P^{\mathcal{C}_{\mathcal{K}}}$.

In this case we write $h: \mathbf{q} \rightarrow \mathcal{C}_{\mathcal{K}}$ (a homomorphism is easily extended from terms to the set of atoms of \mathbf{q}). It is readily seen that $\mathcal{C}_{\mathcal{K}} \models \mathbf{q}(\mathbf{a})$ iff there is a homomorphism $h: \mathbf{q} \rightarrow \mathcal{C}_{\mathcal{K}}$ such that $h(\mathbf{x}) = \mathbf{a}$. We are now fully equipped to introduce the tree-witness rewriting of a CQ $\mathbf{q}(\mathbf{x})$ and an OWL 2 QL TBox \mathcal{T} .

3.2 PE-Rewritings

Following the divide and conquer strategy, we first define our rewriting in two simplified cases.

Flat TBoxes. To begin with, let us assume that the given TBox \mathcal{T} is *flat* in the sense that it contains no axioms of the form $B \sqsubseteq \exists R$. This means that the anonymous part of the canonical model $\mathcal{C}_{\mathcal{K}}$, for any $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, is *empty*, and so we have to look for homomorphisms into to the ABox part of the canonical model. Recall that the canonical model $\mathcal{C}_{\mathcal{K}}$ is constructed by extending \mathcal{A} with all the consequences of \mathcal{A} with respect to \mathcal{T} . For example, if $P(a, b) \in \mathcal{A}$ and \mathcal{T} contains $\exists P \sqsubseteq A$, then \mathcal{A} is extended with the atom $A(a)$. So, to obtain an FO-rewriting of \mathbf{q} and \mathcal{T} , it is enough to replace every atom α in \mathbf{q} with a

disjunction of all atoms that imply α over \mathcal{T} . More precisely, for any concept name A and role name P , we take the formulas:

$$\text{ext}_A(u) = \bigvee_{\mathcal{T} \models A' \sqsubseteq A} A'(u) \vee \bigvee_{\mathcal{T} \models \exists R \sqsubseteq A} \exists v R(u, v), \quad (1)$$

$$\text{ext}_P(u, v) = \bigvee_{\mathcal{T} \models R \sqsubseteq P} R(u, v). \quad (2)$$

We define a PE-query $\mathbf{q}_{\text{ext}}(\mathbf{x})$ as the result of replacing every atom $A(u)$ in \mathbf{q} with $\text{ext}_A(u)$ and every atom $P(u, v)$ in \mathbf{q} with $\text{ext}_P(u, v)$. It is not hard to see that, for any ABox \mathcal{A} and any $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $\mathcal{C}_{\mathcal{K}} \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{A} \models \mathbf{q}_{\text{ext}}(\mathbf{a})$. Thus, we arrive at the following:

Proposition 1. *For any CQ $\mathbf{q}(\mathbf{x})$ and any flat OWL 2 QL TBox \mathcal{T} , $\mathbf{q}_{\text{ext}}(\mathbf{x})$ is a PE-rewriting of \mathbf{q} and \mathcal{T} .*

Thus, in the flat case, it is really easy to compute PE-rewritings.

Example 7. Consider the CQ $\mathbf{q}(\mathbf{x}) = \exists y (A(x) \wedge P(x, y))$ and the flat TBox \mathcal{T} with the axioms

$$\begin{array}{lll} A' \sqsubseteq A, & \exists P \sqsubseteq A, & \exists R' \sqsubseteq A', \\ R^- \sqsubseteq P, & R' \sqsubseteq P, & S \sqsubseteq R. \end{array}$$

Then

$$\begin{aligned} \text{ext}_A(x) &= A(x) \vee A'(x) \vee \exists z P(x, z) \vee \exists z R'(x, z) \vee \exists z R(z, x) \vee \exists z S(z, x), \\ \text{ext}_P(x, y) &= P(x, y) \vee R(y, x) \vee R'(x, y) \vee S(y, x). \end{aligned}$$

Therefore, the PE-rewriting is as follows:

$$\mathbf{q}_{\text{ext}}(x) = \exists y [(A(x) \vee A'(x) \vee \exists z P(x, z) \vee \exists z R'(x, z) \vee \exists z R(z, x) \vee \exists z S(z, x)) \wedge (P(x, y) \vee R(y, x) \vee R'(x, y) \vee S(y, x))].$$

H-complete ABoxes. In the second simplified case, we assume that all the ABoxes for which we have to construct an FO-rewriting of $\mathbf{q}(\mathbf{x})$ and (not necessarily flat) \mathcal{T} are *H-complete with respect to \mathcal{T}* in the sense that

$$\begin{array}{lll} A(a) \in \mathcal{A} & \text{if} & A'(a) \in \mathcal{A} \text{ and } \mathcal{T} \models A' \sqsubseteq A, \\ A(a) \in \mathcal{A} & \text{if} & R(a, b) \in \mathcal{A} \text{ and } \mathcal{T} \models \exists R \sqsubseteq A, \\ P(a, b) \in \mathcal{A} & \text{if} & R(a, b) \in \mathcal{A} \text{ and } \mathcal{T} \models R \sqsubseteq P, \end{array}$$

for all concept names A, A' , roles R and role names P . An FO-query $\mathbf{q}'(\mathbf{x})$ is an *FO-rewriting of \mathbf{q} and \mathcal{T} over H-complete ABoxes* if, for any H-complete (with respect to \mathcal{T}) ABox \mathcal{A} and any $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{A} \models \mathbf{q}'(\mathbf{a})$. Note that if \mathcal{T} is flat then \mathbf{q} itself can clearly serve as a rewriting of \mathbf{q} and \mathcal{T} over H-complete ABoxes. The following example illustrates the notions we need in order to introduce the tree-witness rewriting over H-complete ABoxes.

Example 8. Consider an ontology \mathcal{T} with the axioms

$$RA \sqsubseteq \exists \text{worksOn}.Project, \tag{3}$$

$$Project \sqsubseteq \exists \text{isManagedBy}.Prof, \tag{4}$$

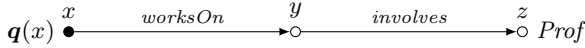
$$\text{worksOn}^- \sqsubseteq \text{involves}, \tag{5}$$

$$\text{isManagedBy} \sqsubseteq \text{involves} \tag{6}$$

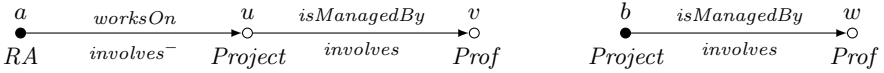
and the CQ asking to find those who work with professors:

$$\mathbf{q}(x) = \exists y, z (\text{worksOn}(x, y) \wedge \text{involves}(y, z) \wedge \text{Prof}(z)),$$

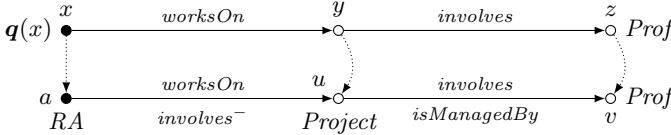
or graphically:



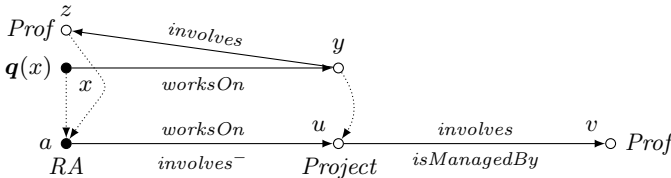
Observe that if the canonical model $\mathcal{C}_{\mathcal{K}}$ of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, for some ABox \mathcal{A} , contains individuals $a \in RA^{\mathcal{C}_{\mathcal{K}}}$ and $b \in Project^{\mathcal{C}_{\mathcal{K}}}$, then $\mathcal{C}_{\mathcal{K}}$ must also contain the following fragments:



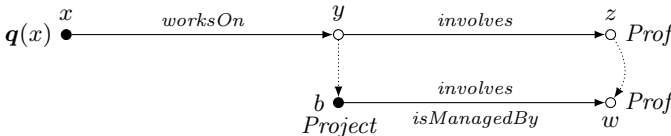
Here the vertices \circ are either named individuals from the ABox or anonymous witnesses for the existential quantifiers (generated by the axioms (3) and (4)). It follows then that a is an answer to $\mathbf{q}(x)$ if $a \in RA^{\mathcal{C}_{\mathcal{K}}}$ because we have the following homomorphism from \mathbf{q} to $\mathcal{C}_{\mathcal{K}}$:



Alternatively, if a is in both $RA^{\mathcal{C}_{\mathcal{K}}}$ and $Prof^{\mathcal{C}_{\mathcal{K}}}$, then we obtain the following homomorphism:



Another option is to map x and y to ABox individuals, a and b , and if b is in $Project^{\mathcal{C}_{\mathcal{K}}}$, then the last two atoms of \mathbf{q} can be mapped to the anonymous part:

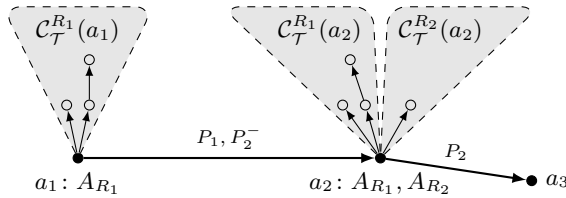


Finally, all the atoms of \mathbf{q} can be mapped to ABox individuals. The possible ways of mapping parts of \mathbf{q} to the anonymous part of the canonical model are called *tree witnesses*. The tree-witnesses for \mathbf{q} found above give the following tree-witness rewriting \mathbf{q}_{tw} of \mathbf{q} and \mathcal{T} over H-complete ABoxes:

$$\mathbf{q}_{\text{tw}}(x) = \exists y, z \left[(\text{worksOn}(x, y) \wedge \text{involves}(y, z) \wedge \text{Prof}(z)) \vee \right. \\ \left. RA(x) \vee (RA(x) \wedge \text{Prof}(x)) \vee (\text{worksOn}(x, y) \wedge \text{Project}(y)) \right].$$

We now give a general definition of tree-witness rewriting over H-complete ABoxes. For every role name R in \mathcal{T} , we take two fresh concept names A_R , A_{R^-} and add to \mathcal{T} the axioms $A_R \equiv \exists R$ and $A_{R^-} \equiv \exists R^-$. We say that the resulting TBox is in *normal form* and assume, without loss of generality, that every TBox in this section is in normal form.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Every individual $a \in \text{ind}(\mathcal{A})$ with $\mathcal{K} \models A_R(a)$ is a root of a (possibly infinite) subtree $\mathcal{C}_{\mathcal{T}}^R(a)$ of $\mathcal{C}_{\mathcal{K}}$, which may intersect another such tree only on their common root a . Every $\mathcal{C}_{\mathcal{T}}^R(a)$ is isomorphic to the canonical model of $(\mathcal{T}, \{A_R(a)\})$.



Suppose now that there is a homomorphism $h: \mathbf{q} \rightarrow \mathcal{C}_{\mathcal{K}}$. Then h splits \mathbf{q} into the subquery mapped by h to the ABox part and the subquery mapped to the anonymous part of $\mathcal{C}_{\mathcal{K}}$, which is a union of the trees $\mathcal{C}_{\mathcal{T}}^R(a)$. We can think of a rewriting of \mathbf{q} and \mathcal{T} as listing possible splits of \mathbf{q} into such subqueries.

Suppose $\mathbf{q}' \subseteq \mathbf{q}$ and there is a homomorphism $h: \mathbf{q}' \rightarrow \mathcal{C}_{\mathcal{T}}^R(a)$, for some a , such that h maps all answer variables in \mathbf{q}' to a . Let $\mathbf{t}_r = h^{-1}(a)$ and let \mathbf{t}_i be the remaining set of (existentially quantified) variables in \mathbf{q}' . Suppose $\mathbf{t}_i \neq \emptyset$. We call the pair $\mathbf{t} = (\mathbf{t}_r, \mathbf{t}_i)$ a *tree witness for \mathbf{q} and \mathcal{T} generated by R* if the query \mathbf{q}' is a *minimal* subset of \mathbf{q} such that, for any $y \in \mathbf{t}_i$, every atom in \mathbf{q} containing y belongs to \mathbf{q}' . In this case, we denote \mathbf{q}' by $\mathbf{q}_{\mathbf{t}}$. By definition, we have

$$\mathbf{q}_{\mathbf{t}} = \{S(z) \in \mathbf{q} \mid z \subseteq \mathbf{t}_r \cup \mathbf{t}_i \text{ and } z \not\subseteq \mathbf{t}_r\}.$$

Note that the same tree witness $\mathbf{t} = (\mathbf{t}_r, \mathbf{t}_i)$ can be generated by different roles R . We denote the set of all such roles by $\Omega_{\mathbf{t}}$ and define the formula

$$\text{tw}_{\mathbf{t}} = \bigvee_{R \in \Omega_{\mathbf{t}}} \exists z (A_R(z) \wedge \bigwedge_{x \in \mathbf{t}_i} (x = z)). \quad (7)$$

(From a practical point of view, it is enough to take only A_R for $\leq_{\mathcal{T}}$ -maximal roles R .)

Tree witnesses \mathbf{t} and \mathbf{t}' are called *consistent* if $\mathbf{q}_{\mathbf{t}} \cap \mathbf{q}_{\mathbf{t}'} = \emptyset$. Each *consistent set* Θ of tree witnesses (in which any pair of distinct tree witnesses is consistent)

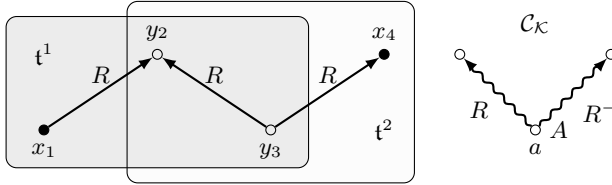
determines a subquery \mathbf{q}_Θ of \mathbf{q} that comprises all atoms of the \mathbf{q}_t , for $t \in \Theta$. The subquery \mathbf{q}_Θ is to be mapped to the $\mathcal{C}_T^R(a)$, whereas the remainder, $\mathbf{q} \setminus \mathbf{q}_\Theta$, obtained by removing the atoms of \mathbf{q}_Θ from \mathbf{q} , is mapped to $\text{ind}(\mathcal{A})$. The following PE-query \mathbf{q}_{tw} is called the *tree-witness rewriting of \mathbf{q} and \mathcal{T} over H-complete ABoxes*:

$$\mathbf{q}_{\text{tw}}(\mathbf{x}) = \bigvee_{\Theta \text{ consistent}} \exists \mathbf{y} \left((\mathbf{q} \setminus \mathbf{q}_\Theta) \wedge \bigwedge_{t \in \Theta} \text{tw}_t \right). \quad (8)$$

Example 9. Consider the KB $\mathcal{K} = (\mathcal{T}, \{A(a)\})$, where

$$\mathcal{T} = \{A \sqsubseteq \exists R, A \sqsubseteq \exists R^-, A_R \equiv \exists R, A_{R^-} \equiv \exists R^-\},$$

and the CQ $\mathbf{q}(x_1, x_4) = \{R(x_1, y_2), R(y_3, y_2), R(y_3, x_4)\}$ shown in the picture below alongside the canonical model $\mathcal{C}_\mathcal{K}$ (with A_R and A_{R^-} omitted).



There are two tree witnesses for \mathbf{q} and \mathcal{T} : $t^1 = (t_r^1, t_l^1)$ generated by R , and $t^2 = (t_r^2, t_l^2)$ generated by R^- , with

$$\begin{aligned} t_r^1 &= \{x_1, y_3\}, & t_l^1 &= \{y_2\}, & \text{tw}_{t^1} &= \exists z (A_R(z) \wedge (x_1 = z) \wedge (y_3 = z)), \\ t_r^2 &= \{y_2, x_4\}, & t_l^2 &= \{y_3\}, & \text{tw}_{t^2} &= \exists z (A_{R^-}(z) \wedge (x_4 = z) \wedge (y_2 = z)). \end{aligned}$$

We have $\mathbf{q}_{t^1} = \{R(x_1, y_2), R(y_3, y_2)\}$ and $\mathbf{q}_{t^2} = \{R(y_3, y_2), R(y_3, x_4)\}$, so t^1 and t^2 are inconsistent. Thus, we obtain the following tree-witness rewriting over H-complete ABoxes:

$$\mathbf{q}_{\text{tw}}(x_1, x_4) = \exists y_2, y_3 [(R(x_1, y_2) \wedge R(y_3, y_2) \wedge R(y_3, x_4)) \vee (R(y_3, x_4) \wedge \text{tw}_{t^1}) \vee (R(x_1, y_2) \wedge \text{tw}_{t^2})].$$

Theorem 3 ([32]). *For any ABox \mathcal{A} that is H-complete with respect to \mathcal{T} and any $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $\mathcal{C}_{(\mathcal{T}, \mathcal{A})} \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{A} \models \mathbf{q}_{\text{tw}}(\mathbf{a})$.*

Tree-witness rewriting. Finally, to obtain an FO-rewriting of $\mathbf{q}(\mathbf{x})$ and \mathcal{T} over arbitrary ABoxes, it is enough to take the tree-witness rewriting \mathbf{q}_{tw} over H-complete ABoxes and replace every atom $\alpha(\mathbf{u})$ in it with $\text{ext}_\alpha(\mathbf{u})$.

3.3 NDL-Rewritings

Next, we show how the tree-witness rewriting can be represented as an NDL-query. We remind the reader that a *datalog program*, Π , is a finite set of Horn clauses (or rules) of the form

$$\forall \mathbf{x} (\gamma_1 \wedge \cdots \wedge \gamma_m \rightarrow \gamma_0),$$

where each γ_i is an atom $P(x_1, \dots, x_l)$ with $x_i \in \mathbf{x}$ (see e.g., [1]). The atom γ_0 is called the *head* of the clause, and $\gamma_1, \dots, \gamma_m$ its *body*. In the datalog literature, a standard agreement is to omit the universal quantifiers, replace \wedge with a comma, and put the head before the body; thus, the rule above is written as

$$\gamma_0 \leftarrow \gamma_1, \dots, \gamma_m.$$

All variables occurring in the head must also occur in the body. We will also assume that the heads do not contain constant symbols. A predicate P *depends* on a predicate Q in Π if Π contains a clause whose head is P and whose body contains Q . Π is called *nonrecursive* if this dependence relation for Π is acyclic. For example, we can define the ext predicates (1) and (2) by a nonrecursive datalog program with the following rules:

$$\text{ext}_A(x) \leftarrow A'(x), \quad \text{for a concept name } A \text{ with } \mathcal{T} \models A' \sqsubseteq A, \quad (9)$$

$$\text{ext}_A(x) \leftarrow R(x, y), \quad \text{for a concept name } A \text{ with } \mathcal{T} \models \exists R \sqsubseteq A, \quad (10)$$

$$\text{ext}_P(x, y) \leftarrow R(x, y), \quad \text{for a role name } P \text{ with } \mathcal{T} \models R \sqsubseteq P. \quad (11)$$

(Note that $\forall x, y (R(x, y) \rightarrow \text{ext}_A(x))$ is equivalent to $\forall x (\exists y R(x, y) \rightarrow \text{ext}_A(x))$.)

Let $\mathbf{q}(\mathbf{x})$ be a CQ and \mathcal{T} an OWL2QL TBox. For a nonrecursive datalog program Π and an atom $\mathbf{q}'(\mathbf{x})$, we say that (Π, \mathbf{q}') is an *NDL-rewriting of $\mathbf{q}(\mathbf{x})$* and \mathcal{T} (over H-complete ABoxes) in case $(\mathcal{T}, \mathcal{A}) \models \mathbf{q}(\mathbf{a})$ iff $\Pi, \mathcal{A} \models \mathbf{q}'(\mathbf{a})$, for any (H-complete) ABox \mathcal{A} and any $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$. An NDL-rewriting over arbitrary ABoxes can clearly be obtained from an NDL-rewriting over H-complete ABoxes by plugging in the ext rules above (at the price of a polynomial blowup).

Let us see how the tree-witness PE-rewriting (8) will look like if we represent it as an NDL-rewriting over H-complete ABoxes. Suppose $\mathbf{t} = (\mathbf{t}_r, \mathbf{t}_i)$ is a tree witness for \mathbf{q} and \mathcal{T} with $\mathbf{t}_r = \{t_1, \dots, t_k\}$, $k \geq 0$. We associate with \mathbf{t} a k -ary predicate $\text{tw}_{\mathbf{t}}$ defined by the following set of rules:

$$\text{tw}_{\mathbf{t}}(x, \dots, x) \leftarrow A_R(x), \quad \text{for } R \in \Omega_{\mathbf{t}}. \quad (12)$$

If $\mathbf{t}_r \neq \emptyset$ then (12) makes all the arguments of $\text{tw}_{\mathbf{t}}$ equal; otherwise, $\mathbf{t}_r = \emptyset$ and $\text{tw}_{\mathbf{t}}$ is a propositional variable, with x being existentially quantified in the body of (12). As the arguments of $\text{tw}_{\mathbf{t}}$ play identical roles, we can write $\text{tw}_{\mathbf{t}}(\mathbf{t}_r)$ without specifying any order on the set \mathbf{t}_r . We obtain an NDL-rewriting $(\Pi, \mathbf{q}_{\text{tw}}(\mathbf{x}))$ of $\mathbf{q}(\mathbf{x})$ and \mathcal{T} over H-complete ABoxes by taking Π to be the nonrecursive datalog program containing the rules of the form (12) together with the rules

$$\mathbf{q}_{\text{tw}}(\mathbf{x}) \leftarrow (\mathbf{q} \setminus \mathbf{q}_{\Theta}), \text{tw}_{\mathbf{t}^1}(\mathbf{t}_r^1), \dots, \text{tw}_{\mathbf{t}^k}(\mathbf{t}_r^k), \quad \text{for consistent } \Theta = \{\mathbf{t}_r^1, \dots, \mathbf{t}_r^k\}. \quad (13)$$

Example 10. Let \mathbf{q} and \mathcal{T} be the same as in Example 9. Denote by Π the datalog program given below:

$$\begin{aligned} \mathbf{q}_{\text{tw}}(x_1, x_4) &\leftarrow R(x_1, y_2), R(y_3, y_2), R(y_3, x_4), \\ \mathbf{q}_{\text{tw}}(x_1, x_4) &\leftarrow R(y_3, x_4), \text{tw}_{\mathbf{t}^1}(x_1, y_3), \\ \mathbf{q}_{\text{tw}}(x_1, x_4) &\leftarrow R(x_1, y_2), \text{tw}_{\mathbf{t}^2}(y_2, x_4), \\ \text{tw}_{\mathbf{t}^1}(x, x) &\leftarrow A_R(x), \\ \text{tw}_{\mathbf{t}^2}(x, x) &\leftarrow A_{R^-}(x). \end{aligned}$$

Then $(\Pi, \mathbf{q}_{\text{tw}}(x_1, x_4))$ is an NDL-rewriting of \mathbf{q} and \mathcal{T} over H-complete ABoxes. To obtain an NDL-rewriting over arbitrary ABoxes, we replace all predicates in the rules above with their ext counterparts and add the appropriate set of rules (9)–(11).

4 Long Rewritings, Short Rewritings

The attractive idea of OBDA with databases relies upon the empirical fact that answering CQs in RDBMSs is usually very efficient in practice. A complexity-theoretic justification for this fact is as follows. In general, to evaluate a Boolean CQ \mathbf{q} with existentially quantified variables \mathbf{y} over a database instance D , we require—in the worst case—time $O(|\mathbf{q}| \cdot |D|^{|\mathbf{y}|})$, where $|\mathbf{q}|$ is the number of atoms in \mathbf{q} . The problem of CQ evaluation is $W[1]$ -complete [51], and so can be really hard for RDBMSs if the input queries are large. However, if \mathbf{q} is tree-shaped (of bounded treewidth, to be more precise)—which is often the case in practice—it can be evaluated in polynomial time, $\text{poly}(|\mathbf{q}|, |D|)$ in symbols [25].

In the context of OBDA, an RDBMS is evaluating not the original CQ \mathbf{q} but an FO-rewriting of \mathbf{q} and the given *OWL 2 QL* TBox \mathcal{T} . All standard FO-rewritings are of size $O((|\mathbf{q}| \cdot |\mathcal{T}|)^{|\mathbf{q}|})$ in the worst case. For example, the size of the tree-witness rewriting (8) is $|\mathbf{q}_{\text{tw}}| = O(|\Xi| \cdot |\mathbf{q}| \cdot |\mathcal{T}|)$, where Ξ is the collection of all consistent sets of tree witnesses for \mathbf{q} and \mathcal{T} (the $|\mathcal{T}|$ factor comes from the tw_t -formulas and multiple roles that may generate a tree witness). Thus, in principle, if there are many consistent sets of tree witnesses for \mathbf{q} and \mathcal{T} , the rewriting \mathbf{q}_{tw} may become prohibitively large for RDBMSs. Recall also that \mathbf{q}_{tw} is a rewriting over *H-complete* ABoxes; to obtain a rewriting over arbitrary ABoxes, we have to replace each atom α in \mathbf{q}_{tw} with the respective ext_α . This will add another factor $|\mathcal{T}|$ to the size of \mathbf{q}_{tw} . RDBMSs are known to be most efficient for evaluating CQs and unions of CQs (UCQs for short) of reasonable size. But transforming a PE-rewriting to the UCQ form can cause an exponential blowup in the size of the query (consider, for example, the PE-rewriting $\mathbf{q}_{\text{ext}}(x)$ in Example 7 and imagine that the original CQ \mathbf{q} contains many atoms). These observations put forward the followings questions:

- What is the overhead of answering CQs via *OWL 2 QL* ontologies compared to standard database query answering in the worst case?
- What is the size of FO-rewritings of CQs and *OWL 2 QL* ontologies in the worst case?
- Can rewritings of one type (say, FO) be substantially shorter than rewritings of another type (say, PE)?
- Are there interesting and useful sufficient conditions on CQs and ontologies under which rewritings are short?

In this section, we give an overview of the answers to the above questions obtained so far [31,34,32,23,30].

In general, succinctness problems such as the second and third questions above can be very hard to solve. Perhaps one of the most interesting and important

examples is succinctness of various formalisms for computing Boolean functions such as propositional formulas, branching programs and circuits, which has been investigated since Shannon's seminal work of 1949 [65], where he suggested the size of the smallest circuit computing a Boolean function as a measure of the complexity of that function.

We remind the reader (see, e.g., [3,29] for more details) that an n -ary Boolean function, for $n \geq 1$, is a function $f: \{0, 1\}^n \rightarrow \{0, 1\}$. An n -input Boolean circuit, \mathbf{C} , for $n \geq 1$, is a directed acyclic graph with n sources (inputs) and one sink (output). Every non-source vertex of \mathbf{C} is called a gate; it is labelled with either \wedge or \vee , in which case it has two incoming edges, or with \neg , in which case there is one incoming edge. The number of vertices in \mathbf{C} will be denoted by $|\mathbf{C}|$. We think of a Boolean formula as a circuit in which every gate has at most one outgoing edge. If $\mathbf{x} \in \{0, 1\}^n$, then $\mathbf{C}(\mathbf{x})$ is the output of \mathbf{C} on input \mathbf{x} . We say that \mathbf{C} computes a Boolean function f if $\mathbf{C}(\mathbf{x}) = f(\mathbf{x})$, for every $\mathbf{x} \in \{0, 1\}^n$.

In the circuit complexity theory, we are interested in families of Boolean functions, that is, sequences f^1, f^2, \dots , where each f^n is an n -ary Boolean function. For example, we can consider the family $\text{CLIQUE}_{n,k}(e)$ of Boolean functions of $n(n-1)/2$ variables $e_{jj'}$, $1 \leq j < j' \leq n$, that return 1 iff the graph with vertices $\{1, \dots, n\}$ and edges $\{\{j, j'\} \mid e_{jj'} = 1\}$ contains a k -clique, for some fixed k .

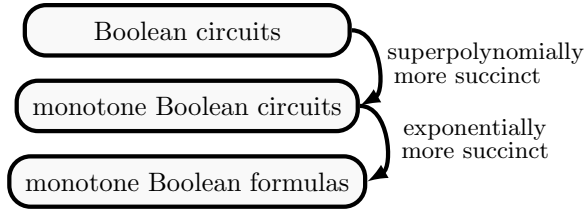
Given a function $T: \mathbb{N} \rightarrow \mathbb{N}$, by a T -size family of circuits we mean a sequence $\mathbf{C}^1, \mathbf{C}^2, \dots$, where each \mathbf{C}^n is an n -input Boolean circuits of size $|\mathbf{C}^n| \leq T(n)$. Every family f^n of Boolean functions can clearly be computed by circuits of size $n \cdot 2^n$ (take disjunctive normal forms representing the f^n). The class of languages that are decidable by families of polynomial-size circuits is denoted by P/poly. It is known that $\text{P} \not\subseteq \text{P/poly}$. Thus, one might hope to prove that $\text{P} \neq \text{NP}$ by showing that $\text{NP} \not\subseteq \text{P/poly}$. In other words, to crack one of the most important problems in computer science and mathematics,¹ it is enough to find a family of Boolean functions in NP that cannot be computed by a polynomial-size family of circuits. This does not look like a particularly hard problem! After all, it has been known since 1949 that the majority of Boolean functions can only be computed by exponential-size circuits. Yet, so far no one has managed to find a concrete family of functions in NP that need circuits with more than $4.5n - o(n)$ gates [40].

Investigating restricted classes of Boolean circuits has proved to be much more successful. The class that is relevant in the context of PE-rewritings consists of monotone Boolean functions, that is, those computable by monotone circuits with only \wedge - and \vee -gates. For example, the function $\text{CLIQUE}_{n,k}(e)$ is monotone. As $\text{CLIQUE}_{n,k}$ is NP-complete, the question whether $\text{CLIQUE}_{n,k}$ can be computed by polynomial-size circuits is equivalent to the open $\text{NP} \subseteq \text{P/poly}$ problem. A series of papers, started by Razborov's [57], gave an exponential lower bound for the size of monotone circuits computing $\text{CLIQUE}_{n,k}$: $2^{\Omega(\sqrt{k})}$ for $k \leq \frac{1}{4}(n/\log n)^{2/3}$ [2]. For monotone formulas, an even better lower bound was obtained: $2^{\Omega(k)}$ for $k = 2n/3$ [56]. It follows that, if we assume $\text{NP} \not\subseteq \text{P/poly}$,

¹ It is actually one of the seven Millennium Prize Problems worth of \$1 000 000 each; consult www.claymath.org/millennium.

then no polynomial-size family of (not necessarily monotone) circuits can compute $\text{CLIQUE}_{n,k}$.

A few other interesting examples of monotone functions have also been found. Thus, [55] gave a family of monotone Boolean functions that can be computed by polynomial-size monotone circuits, but any monotone *formulas* computing this family are of size 2^{n^ϵ} , for some $\epsilon > 0$. Or there is a family of monotone functions [56,8] showing that non-monotone Boolean circuits are in general superpolynomially more succinct than monotone circuits.



Let us now return to rewritings of CQs and *OWL2QL* TBoxes. As we shall see later on in this section, there is a close correspondence between (arbitrary) Boolean formulas and FO-rewritings, monotone Boolean formulas and PE-rewritings, and between monotone Boolean circuits and NDL-rewritings:

Boolean formulas	\approx	FO-rewritings
monotone Boolean circuits	\approx	NDL-rewritings
monotone Boolean formulas	\approx	PE-rewritings

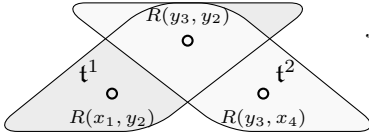
To begin with, we associate with the tree-witness (PE- or NDL-) rewritings \mathbf{q}_{tw} certain monotone Boolean functions that will be called hypergraph functions.

A *hypergraph* $H = (V, E)$ is given by its *vertices* $v \in V$ and *hyperedges* $e \in E$, where $E \subseteq 2^V$. We call a subset $X \subseteq E$ *independent* if $e \cap e' = \emptyset$, for any distinct $e, e' \in X$. The set of vertices that occur in the hyperedges of X is denoted by V_X . With each vertex $v \in V$ and each hyperedge $e \in E$ we associate propositional variables p_v and p_e , respectively. The *hypergraph function* f_H for a hypergraph H is computed by the Boolean formula

$$f_H = \bigvee_{X \text{ independent}} \left(\bigwedge_{v \in V \setminus V_X} p_v \wedge \bigwedge_{e \in X} p_e \right). \tag{14}$$

This definition is clearly inspired by (8). The PE-rewriting \mathbf{q}_{tw} of \mathbf{q} and \mathcal{T} defines a hypergraph whose vertices are the atoms of \mathbf{q} and hyperedges are the sets \mathbf{q}_t , for tree witnesses t for \mathbf{q} and \mathcal{T} . We denote this hypergraph by $H_{\mathcal{T}}^{\mathbf{q}}$. The formula (14) defining $f_{H_{\mathcal{T}}^{\mathbf{q}}}$ is basically the same as the rewriting (8) with the atoms $S(\mathbf{z})$ in \mathbf{q} and tree witness formulas tw_t treated as propositional variables. We denote these variables by $p_{S(\mathbf{z})}$ and p_t (rather than p_v and p_e), respectively.

Example 11. Consider again the CQ \mathbf{q} and TBox \mathcal{T} from Example 9. The hypergraph $H_{\mathcal{T}}^{\mathbf{q}}$ and its hypergraph function $f_{H_{\mathcal{T}}^{\mathbf{q}}}$ are shown below:



$$f_{H_{\mathcal{T}}^{\mathbf{q}}} = (p_{R(x_1, y_2)} \wedge p_{R(y_3, y_2)} \wedge p_{R(y_3, x_4)}) \vee (p_{R(y_3, x_4)} \wedge p_{t^1}) \vee (p_{R(x_1, y_2)} \wedge p_{t^2}).$$

Suppose now that the hypergraph function $f_{H_{\mathcal{T}}^{\mathbf{q}}}$ is computed by some Boolean formula $\chi_{H_{\mathcal{T}}^{\mathbf{q}}}$. Consider the FO-formula $\widehat{\chi}_{H_{\mathcal{T}}^{\mathbf{q}}}$ obtained by replacing each $p_{S(\mathbf{z})}$ in $\chi_{H_{\mathcal{T}}^{\mathbf{q}}}$ with $S(\mathbf{z})$, each p_t with tw_t , and adding the prefix $\exists \mathbf{y}$. By comparing (14) and (8), we see that the resulting FO-formula is a rewriting of \mathbf{q} and \mathcal{T} over H-complete ABoxes. A monotone circuit computing $f_{H_{\mathcal{T}}^{\mathbf{q}}}$ can be converted to an NDL-rewriting of \mathbf{q} and \mathcal{T} over H-complete ABoxes. This gives us the following:

Theorem 4. (i) *If the function $f_{H_{\mathcal{T}}^{\mathbf{q}}}$ is computed by a Boolean formula $\chi_{H_{\mathcal{T}}^{\mathbf{q}}}$, then $\widehat{\chi}_{H_{\mathcal{T}}^{\mathbf{q}}}$ is an FO-rewriting of \mathbf{q} and \mathcal{T} over H-complete ABoxes.*

(ii) *If $f_{H_{\mathcal{T}}^{\mathbf{q}}}$ is computed by a monotone Boolean circuit \mathbf{C} , then there is an NDL-rewriting of \mathbf{q} and \mathcal{T} over H-complete ABoxes of size $O(|\mathbf{C}| \cdot (|\mathbf{q}| + |\mathcal{T}|))$.*

Thus, the problem of constructing short rewritings is reducible to the problem of finding short Boolean formulas or circuits computing the hypergraph functions. We call a hypergraph H *representable* if there are a CQ \mathbf{q} and an OWL2QL TBox \mathcal{T} such that H is isomorphic to $H_{\mathcal{T}}^{\mathbf{q}}$.

Let us consider first hypergraphs of *degree* ≤ 2 , in which every vertex belongs to *at most two* hyperedges. There is a striking correspondence between such hypergraphs and OWL2QL TBoxes \mathcal{T} of *depth one*, which *cannot* have chains of *more than 1* point in the anonymous part of their canonical models (more precisely, whenever $aw_{[R_1]} \cdots w_{[R_n]}$ is an element of some canonical model for \mathcal{T} then $n \leq 1$). As observed above, PE-rewritings of CQs and flat TBoxes (that is, TBoxes of depth 0) can always be made of polynomial size.

Theorem 5 ([30]). (i) *If \mathbf{q} is a CQ and \mathcal{T} a TBox of depth one, then the hypergraph $H_{\mathcal{T}}^{\mathbf{q}}$ is of degree ≤ 2 .*

(ii) *The number of distinct tree witnesses for \mathbf{q} and \mathcal{T} does not exceed the number of variables in \mathbf{q} .*

(iii) *Any hypergraph H of degree ≤ 2 is representable by means of some CQ and TBox of depth one.*

Here we only consider (iii). Suppose that $H = (V, E)$ is a hypergraph of degree ≤ 2 . To simplify notation, we assume that any vertex in H belongs to exactly 2 hyperedges, so H comes with two fixed maps $i_1, i_2: V \rightarrow E$ such that $i_1(v) \neq i_2(v)$, $v \in i_1(v)$ and $v \in i_2(v)$, for any $v \in V$. For every $e \in E$, we take an individual variable z_e and denote by \mathbf{z} the vector of all such variables. For every $v \in V$, we take a role name R_v and define a Boolean CQ \mathbf{q}_H by taking:

$$\mathbf{q}_H = \exists \mathbf{z} \bigwedge_{v \in V} R_v(z_{i_1(v)}, z_{i_2(v)}).$$

For every hyperedge $e \in E$, let A_e be a concept name and S_e a role name. Consider the TBox \mathcal{T}_H with the following inclusions, for $e \in E$:

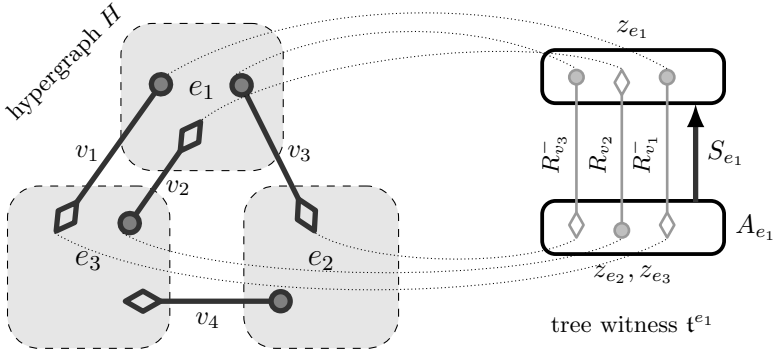
$$\begin{aligned} A_e &\equiv \exists S_e, \\ S_e &\sqsubseteq R_v^-, && \text{for } v \in V \text{ with } i_1(v) = e, \\ S_e &\sqsubseteq R_v, && \text{for } v \in V \text{ with } i_2(v) = e. \end{aligned}$$

We claim that the hypergraph H is isomorphic to $H_{\mathcal{T}_H}^{q_H}$ and illustrate the proof by an example.

Example 12. Let $H = (V, E)$ with $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{e_1, e_2, e_3\}$, where $e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_3, v_4\}$, $e_3 = \{v_1, v_2, v_4\}$. Suppose also that

$$\begin{aligned} i_1: v_1 \mapsto e_1, \quad v_2 \mapsto e_3, \quad v_3 \mapsto e_1, \quad v_4 \mapsto e_2, \\ i_2: v_1 \mapsto e_3, \quad v_2 \mapsto e_1, \quad v_3 \mapsto e_2, \quad v_4 \mapsto e_3. \end{aligned}$$

The hypergraph H is shown below, where each vertex v_i is represented by an edge, $i_1(v_i)$ is indicated by the circle-shaped end of the edge and $i_2(v_i)$ by the diamond-shaped one; the hyperedges e_j are shown as large grey squares:



Then

$$\mathbf{q}_H = \exists z_{e_1} z_{e_2} z_{e_3} (R_{v_1}(z_{e_1}, z_{e_3}) \wedge R_{v_2}(z_{e_3}, z_{e_1}) \wedge R_{v_3}(z_{e_1}, z_{e_2}) \wedge R_{v_4}(z_{e_2}, z_{e_3}))$$

and the TBox \mathcal{T}_H contains the following inclusions:

$$\begin{aligned} A_{e_1} &\equiv \exists S_{e_1}, & S_{e_1} &\sqsubseteq R_{v_1}^-, & S_{e_1} &\sqsubseteq R_{v_2}, & S_{e_1} &\sqsubseteq R_{v_3}^-, \\ A_{e_2} &\equiv \exists S_{e_2}, & S_{e_2} &\sqsubseteq R_{v_3}, & S_{e_2} &\sqsubseteq R_{v_4}^-, \\ A_{e_3} &\equiv \exists S_{e_3}, & S_{e_3} &\sqsubseteq R_{v_1}, & S_{e_3} &\sqsubseteq R_{v_2}^-, & S_{e_3} &\sqsubseteq R_{v_4}. \end{aligned}$$

The canonical model $\mathcal{C}_{\mathcal{T}_H}^{S_{e_1}}(a)$ is shown on the right-hand side of the picture above. We observe now that each variable z_e uniquely determines the tree witness t^e with $\mathbf{q}_{t^e} = \{R_v(z_{i_1(v)}, z_{i_2(v)}) \mid v \in e\}$; \mathbf{q}_{t^e} and $\mathbf{q}_{t^{e'}}$ are consistent iff $e \cap e' \neq \emptyset$. It follows that H is isomorphic to $H_{\mathcal{T}_H}^{q_H}$.

It turns out that answering the CQ q_H over \mathcal{T}_H and certain single-individual ABoxes amounts to computing the Boolean function f_H . Let $H = (V, E)$ be a hypergraph of degree 2 with $V = \{v_1, \dots, v_n\}$ and $E = \{e_1, \dots, e_m\}$. We denote by $\alpha(v_i)$ the i -th component of $\alpha \in \{0, 1\}^n$ and by $\beta(e_j)$ the j -th component of $\beta \in \{0, 1\}^m$. Define an ABox $\mathcal{A}_{\alpha, \beta}$ with a single individual a by taking

$$\mathcal{A}_{\alpha, \beta} = \{R_{v_i}(a, a) \mid \alpha(v_i) = 1\} \cup \{A_{e_j}(a) \mid \beta(e_j) = 1\}.$$

Theorem 6 ([30]). *Let $H = (V, E)$ be a hypergraph of degree 2. Then, for any tuples $\alpha \in \{0, 1\}^{|V|}$ and $\beta \in \{0, 1\}^{|E|}$,*

$$(\mathcal{T}_H, \mathcal{A}_{\alpha, \beta}) \models q_H \quad \text{iff} \quad f_H(\alpha, \beta) = 1.$$

What is so special about hypergraphs of degree ≤ 2 ? First, one can show that all hypergraph functions of degree ≤ 2 can be computed by polynomial-size monotone Boolean circuits—in fact, by polynomial-size monotone nondeterministic branching programs (NBPs), also known as switching-and-rectifier networks [29]. Moreover, the converse also holds: if a family of Boolean functions is computable by polynomial-size NBPs then it can be represented by a family of polynomial-size hypergraphs of degree ≤ 2 . As a consequence of this fact, we obtain a positive result on the size of NDL-rewritings:

Theorem 7 ([30]). *For any CQ q and any TBox \mathcal{T} of depth one, there is an NDL-rewriting of q and \mathcal{T} of polynomial size.*

On the ‘negative’ side, there are families of Boolean functions f_n that are computable by polynomial-size monotone NBPs, but any monotone Boolean formulas computing f_n are of superpolynomial size, at least $2^{\Omega(\log^2 n)}$, to be more precise. For example, Grigni and Sipser [24] consider functions that take the adjacency matrix of a directed graph of n vertices with a distinguished vertex s as input and return 1 iff there is a directed path from s to some vertex of outdegree at least two. Can we use this lower bound to establish a corresponding superpolynomial lower bound for the size of PE-rewritings? The answer naturally depends on what syntactical means we can use in our rewritings.

Let us assume first that the FO- and NDL-rewritings of CQs q and OWL 2 QL TBoxes \mathcal{T} can contain equality ($=$), any non-predifined predicates and only those constant symbols that occur in q . Thus, we do not allow new constants and various built-in predicates in our rewritings. Such rewritings are sometimes called *pure*.

As any NBP corresponds to a polynomial-size hypergraph of degree ≤ 2 , we obtain a sequence H_n of polynomial hypergraphs of degree 2 such that $f_n = f_{H_n}$. We take the sequence of CQs q_n and TBoxes \mathcal{T}_n associated with the hypergraphs of degree ≤ 2 for the sequence f_n of Boolean functions chosen above. We show that any pure PE-rewriting q'_n of q_n and \mathcal{T}_n can be transformed into a monotone Boolean formula χ_n computing f_n and having size $\leq |q'_n|$.

To define χ_n , we eliminate the quantifiers in q'_n in the following way: take a constant a and replace every subformula of the form $\exists x \psi(x)$ in q'_n with $\psi(a)$,

repeating this operation as long as possible. The resulting formula \mathbf{q}''_n is built from atoms of the form $A_e(a)$, $R_v(a, a)$ and $S_e(a, a)$ using \wedge and \vee . For every ABox \mathcal{A} with a single individual a , we have $(\mathcal{T}_n, \mathcal{A}) \models \mathbf{q}_n$ iff $\mathcal{A} \models \mathbf{q}''_n$. Let χ_n be the result of replacing $S_e(a, a)$ in \mathbf{q}''_n with \perp , $A_e(a)$ with p_e and $R_v(a, a)$ with p_v . Clearly, $|\chi_n| \leq |\mathbf{q}''_n|$.

By the definition of $\mathcal{A}_{\alpha, \beta}$ and Theorem 6, we then obtain:

$$\chi_n(\alpha, \beta) = 1 \quad \text{iff} \quad \mathcal{A}_{\alpha, \beta} \models \mathbf{q}''_n \quad \text{iff} \quad \mathcal{A}_{\alpha, \beta} \models \mathbf{q}'_n \quad \text{iff} \\ (\mathcal{T}_n, \mathcal{A}_{\alpha, \beta}) \models \mathbf{q}_n \quad \text{iff} \quad f_{H_n}(\alpha, \beta).$$

It remains to recall that $|\mathbf{q}'_n| \geq |\chi_n| \geq 2^{\Omega(\log^2 n)}$. This gives us the first super-polynomial lower bound for the size of pure PE-rewritings.

Theorem 8 ([30]). *There is a sequence of CQs \mathbf{q}_n and TBoxes \mathcal{T}_n of depth one such that any pure PE-rewriting of \mathbf{q}_n and \mathcal{T}_n (over H-complete ABoxes) is of size $\geq 2^{\Omega(\log^2 n)}$.*

Why are the PE-rewritings in Theorem 8 so large? Let us have another look at the tree-witness rewriting (8) of \mathbf{q} and \mathcal{T} . Its size depends on the number of distinct consistent sets of tree witnesses for \mathbf{q} and \mathcal{T} . In view of Theorem 5 (ii), \mathbf{q}_n and \mathcal{T}_n in Theorem 8 have a polynomial number of tree witnesses. However, many of them are not consistent with each other (see Example 12), which makes it impossible to simplify (8) to a polynomial-size PE-rewriting.

For TBoxes of depth two, we can obtain an even stronger ‘negative’ result:

Theorem 9 ([30]). *There exists a sequence of CQs \mathbf{q}_n and TBoxes \mathcal{T}_n of depth two, such that any pure PE- and NDL-rewriting of \mathbf{q}_n and \mathcal{T}_n is of exponential size, while any FO-rewriting of \mathbf{q}_n and \mathcal{T}_n is of superpolynomial size (unless $\text{NP} \subseteq \text{P/poly}$).*

This theorem can be proved by showing that the hypergraphs $H_{n,k}$ computing $\text{CLIQUE}_{n,k}$ are representable as subgraphs of $H_{\mathcal{T}_n, k}^{\mathbf{q}_n}$ for suitable \mathbf{q}_n and \mathcal{T}_n , and then applying quantifier elimination over single-individual ABoxes as above.

Using TBoxes of unbounded depth, one can show that pure FO-rewritings can be superpolynomially more succinct than pure PE-rewritings:

Theorem 10 ([34]). *There is a sequence of CQs \mathbf{q}_n of size $O(n)$ and TBoxes \mathcal{T}_n of size $O(n)$ that has polynomial-size FO-rewritings, but its pure PE-rewritings are of size $\geq 2^{\Omega(2^{\log^{1/2} n})}$.*

We can also use the machinery developed above to understand the overhead of answering CQs via OWL2QL ontologies compared to standard database query answering:

Theorem 11 ([31]). *There is a sequence of CQs \mathbf{q}_n and OWL2QL TBoxes \mathcal{T}_n such that the problem ‘ $\mathcal{A} \models \mathbf{q}_n$?’ is in P, while the problem ‘ $(\mathcal{T}_n, \mathcal{A}) \models \mathbf{q}_n$?’ is NP-hard (for combined complexity).*

On the other hand, answering tree-shaped CQs (or CQs of bounded treewidth) over TBoxes without role inclusions can be done in polynomial time [6]. On the other hand, a sufficient condition on CQs and *OWL 2 QL* TBoxes that guarantees the existence of polynomial-size pure PE-rewritings can be found in [32].

We conclude this section with a few remarks on ‘impure’ rewritings that can use new constant symbols not occurring in the given CQ. To prove the superpolynomial and exponential lower bounds on the size of pure rewritings above, we established a connection between monotone circuits for Boolean functions and rewritings for certain CQs and *OWL 2 QL* TBoxes. In fact, this connection also suggests a way of making rewritings substantially shorter. Indeed, recall that although no monotone Boolean circuit of polynomial size can compute $\text{CLIQUE}_{n,k}$, we can construct such a circuit if we are allowed to use advice inputs. Indeed, for any family f^1, f^2, \dots of Boolean functions in NP, there exist polynomials p and q and, for each $n \geq 1$, a Boolean circuit C^n with $n + p(n)$ inputs such that $|C^n| \leq q(n)$ and, for any $\alpha \in \{0, 1\}^n$, we have

$$f^n(\alpha) = 1 \quad \text{iff} \quad C^n(\alpha, \beta) = 1, \quad \text{for some } \beta \in \{0, 1\}^{p(n)}.$$

The additional $p(n)$ inputs for β in the C^n are called *advice inputs*. These advice inputs make Boolean circuits *nondeterministic* (in the sense that β is a guess) and, as a result, exponentially more succinct—in the same way as nondeterministic automata are exponentially more succinct than deterministic ones [48]. To introduce corresponding nondeterministic guesses into query rewritings, we can use additional existentially quantified variables—provided that the domain of quantification contains at least two elements. For this purpose, we can assume that all relevant ABoxes contain two fixed individuals (among others).

Theorem 12 ([23,34]). *For any CQ q and OWL 2 QL TBox \mathcal{T} , there are impure polynomial-size PE- and NDL-rewritings of q and \mathcal{T} over ABoxes containing two fixed constants; the rewritings use $O(|q|)$ additional existential quantifiers.*

This polynomial-size impure rewriting hides the superpolynomial and exponential blowups in the case of pure rewritings behind the additional existential quantification over the newly introduced constants.

5 The Combined Approach

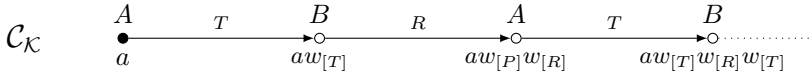
A different approach to OBDA has been suggested in [44,36,37]. It is known as the *combined approach* and aims at scenarios where one can manipulate the database. Suppose we are given a CQ q and a TBox \mathcal{T} , and we want to query an ABox \mathcal{A} . So far, we have first constructed an FO-rewriting q' of q and \mathcal{T} , independently of \mathcal{A} , and then used an RDBMS to evaluate q' over \mathcal{A} . The advantage of this ‘classical’ approach is that it works even when we cannot modify the data in the data source. However, as we saw above, the rewriting q' can be too large for RDBMSs to cope.

But let us assume that we can manipulate the given ABox \mathcal{A} . In this case, a natural question would be: why cannot we first apply \mathcal{T} to \mathcal{A} and then evaluate \mathbf{q} over the resulting canonical model? However, many *OWL 2 QL* TBoxes are of infinite depth, and so the canonical model of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ can be infinite; even if \mathcal{T} is of bounded depth, the canonical model of \mathcal{K} can be of exponential size.

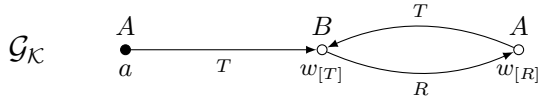
Example 13. Consider the *OWL 2 QL* TBox

$$\mathcal{T} = \{ A \sqsubseteq \exists T, \exists T^- \sqsubseteq B, B \sqsubseteq \exists R, \exists R^- \sqsubseteq A \}.$$

The infinite canonical model of $\mathcal{C}_{\mathcal{K}}$, for $\mathcal{K} = (\mathcal{T}, \{A(a)\})$, looks as follows:



But what if we fold the infinite chain of alternating *R*- and *T*-arrows in a simple cycle such as in the picture below?



Note that $\mathcal{G}_{\mathcal{K}}$ is still a model of \mathcal{K} . Now, consider the CQ

$$\mathbf{q}(x) = \exists y, z (T(x, y) \wedge R(y, z) \wedge T(z, y)).$$

Clearly, we have $\mathcal{G}_{\mathcal{K}} \models \mathbf{q}(a)$, but $\mathcal{C}_{\mathcal{K}} \not\models \mathbf{q}(a)$. Observe, however, that to get rid of this spurious answer a , it is enough to slightly modify \mathbf{q} by adding to it the ‘filtering conditions’ ($z \neq w_{[R]}$), ($y \neq w_{[R^-]}$), ($y \neq w_{[T]}$) and ($z \neq w_{[T^-]}$) as conjuncts (in the scope of $\exists y, z$). Indeed, it is not hard to see that \mathbf{q} and \mathcal{T} have no tree witnesses, and so the variables y and z cannot be mapped anywhere in the anonymous part of the canonical model.

It turns out that this idea works perfectly well at least for the case when *OWL 2 QL* TBoxes *do not contain role inclusions*. Suppose \mathcal{T} is such a TBox and \mathcal{A} an arbitrary ABox. We define the *generating model* $\mathcal{G}_{\mathcal{K}}$ of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ by taking:

$$\begin{aligned} \Delta^{\mathcal{G}_{\mathcal{K}}} &= \{\text{tail}(\sigma) \mid \sigma \in \Delta^{\mathcal{C}_{\mathcal{K}}}\}, \\ a^{\mathcal{G}_{\mathcal{K}}} &= a, \quad \text{for all } a \in \text{ind}(\mathcal{A}), \\ A^{\mathcal{G}_{\mathcal{K}}} &= \{\text{tail}(\sigma) \mid \sigma \in A^{\mathcal{C}_{\mathcal{K}}}\}, \quad \text{for all concept names } A, \\ P^{\mathcal{G}_{\mathcal{K}}} &= \{(\text{tail}(\sigma), \text{tail}(\sigma')) \mid (\sigma, \sigma') \in P^{\mathcal{C}_{\mathcal{K}}}\}, \quad \text{for all role names } P, \end{aligned}$$

where $\mathcal{C}_{\mathcal{K}}$ is the standard canonical model of \mathcal{K} . It is not hard to see that the map $\text{tail}: \Delta^{\mathcal{C}_{\mathcal{K}}} \rightarrow \Delta^{\mathcal{G}_{\mathcal{K}}}$ is a homomorphism from $\mathcal{C}_{\mathcal{K}}$ onto $\mathcal{G}_{\mathcal{K}}$, and $\mathcal{C}_{\mathcal{K}}$ can be viewed as the ‘unraveling’ of $\mathcal{G}_{\mathcal{K}}$. The generating model $\mathcal{G}_{\mathcal{K}}$ can be constructed in polynomial time in $|\mathcal{K}|$. As positive existential queries are preserved under homomorphisms, we obtain:

Theorem 13 ([36]). *For any consistent OWL 2 QL KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, which does not contain role inclusion axioms, any CQ $\mathbf{q}(\mathbf{x})$ and any tuple $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, if $\mathcal{C}_{\mathcal{K}} \models \mathbf{q}(\mathbf{a})$ then $\mathcal{G}_{\mathcal{K}} \models \mathbf{q}(\mathbf{a})$ (but not the other way round).*

Suppose now that we are given a CQ $\mathbf{q}(\mathbf{x}) = \exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$. Our aim is to rewrite \mathbf{q} to an FO-query $\mathbf{q}^\dagger(\mathbf{x})$ in such a way that $\mathcal{C}_{\mathcal{K}} \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{G}_{\mathcal{K}} \models \mathbf{q}^\dagger(\mathbf{a})$, for any tuple $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, and the size of \mathbf{q}^\dagger is polynomial in the size of \mathbf{q} and \mathcal{T} . (Note that the rewriting given in [36,37] does not depend on \mathcal{T} because of a different definition of tree witness.) We define the rewriting \mathbf{q}^\dagger as a conjunction

$$\mathbf{q}^\dagger(\mathbf{x}) = \exists \mathbf{y} (\varphi \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3),$$

where φ_1, φ_2 and φ_3 are Boolean combinations of equalities $t_1 = t_2$, and each of the t_i is either a term in \mathbf{q} or a constant of the form $w_{[R]}$. The purpose of the conjunct

$$\varphi_1 = \bigwedge_{x \in \mathbf{x}} \bigwedge_{R \text{ a role in } \mathcal{T}} (x \neq w_{[R]})$$

is to ensure that the answer variables \mathbf{x} receive their values from $\text{ind}(\mathcal{A})$ only.

The conjunct φ_2 implements the matching dictated by the tree witnesses. Suppose $\mathbf{t} = (t_r, t_i)$ is a tree witness for \mathbf{q} and \mathcal{T} generated by R . If $R(t, t') \in \mathbf{q}$ and t' is mapped to $w_{[R]}$, then all the variables in t_r are to be mapped to the same point as t :

$$\varphi_2 = \bigwedge_{\substack{R(t,t') \in \mathbf{q} \\ \text{there is a tree witness } \mathbf{t} \text{ with } R \in \Omega_{t_i}}} ((t' = w_{[R]}) \rightarrow \bigwedge_{s \in t_r} (s = t)).$$

If there is no tree witness \mathbf{t} generated by R and such that $R(t, t') \in \mathbf{q}_t$, then t' cannot be mapped to the witness $w_{[R]}$ at all. This is ensured by the conjunct

$$\varphi_3 = \bigwedge_{\substack{R(t,t') \in \mathbf{q} \\ \text{no tree witness } \mathbf{t} \text{ with } R \in \Omega_{t_i} \text{ exists}}} (t' \neq w_{[R]}).$$

Theorem 14 ([36]). *For any consistent OWL 2 QL KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ containing no role inclusion axioms, any CQ $\mathbf{q}(\mathbf{x})$ and any tuple $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, we have $\mathcal{C}_{\mathcal{K}} \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{G}_{\mathcal{K}} \models \mathbf{q}^\dagger(\mathbf{a})$.*

The rewriting \mathbf{q}^\dagger can be computed in polynomial time in \mathbf{q} and \mathcal{T} and is of size $O(|\mathbf{q}| \cdot |\mathcal{T}|)$.

A slightly different idea was proposed in [46] to extend the combined approach to unrestricted OWL 2 QL TBoxes. As before, given a CQ $\mathbf{q}(\mathbf{x})$, an OWL 2 QL TBox \mathcal{T} and an ABox \mathcal{A} , we first construct a polynomial-size interpretation $\mathcal{G}'_{\mathcal{K}}$ representing the (possibly infinite) canonical model $\mathcal{C}_{\mathcal{K}}$ (in fact, $\mathcal{G}'_{\mathcal{K}}$ is more involved than $\mathcal{G}_{\mathcal{K}}$). Then we use an RDBMS to compute the answers to the original CQ $\mathbf{q}(\mathbf{x})$ over $\mathcal{G}'_{\mathcal{K}}$ stored in the RDBMS. As we saw above, some of the answers can be spurious. To eliminate them, instead of φ_1, φ_2 and φ_3 , one can use a ‘filtering procedure’ that is installed as a user-defined function in the

RDBMS. This procedure takes as input a match of the query in $\mathcal{G}'_{\mathcal{K}}$ and returns ‘false’ if this match is spurious (that is, cannot be realised in the real canonical model $\mathcal{C}_{\mathcal{K}}$) and ‘true’ otherwise. The filtering procedure runs in polynomial time, although it may have to run exponentially many times (to check exponentially many matches for the same answer tuple) in the worst case.

6 OBDA with *Ontop*

In this final section, we consider the architecture of the OBDA system *Ontop* (`ontop.inf.unibz.it`) implemented at the Free University of Bozen-Bolzano and available as a plugin for the ontology editor Protégé 4, a SPARQL endpoint and OWLAPI and Sesame libraries.

In OBDA with databases, the data comes from a relational database rather than an ABox. From a logical point of view, a *database schema* [1] contains predicate symbols (with their arity) for both stored database relations (also known as tables) and views (with their definitions in terms of stored relations) as well as a set Σ of *integrity constraints* (in the form of functional and inclusion dependencies; for example, primary and foreign keys). Any instance \mathbf{I} of the database schema must satisfy its integrity constraints Σ .

The vocabularies of the database schema and the given TBox are linked together by means of mappings produced by a domain expert or extracted (semi)automatically. There are different known types of mappings: LAV (local-as-views), GAV (global-as-views), GLAV, etc.; consult, e.g., [41] for an overview. Here we concentrate on GAV mappings because they guarantee low complexity of query answering (in what follows we call them simply mappings). A *mapping*, \mathcal{M} , is a set of rules of the form

$$S(\mathbf{x}) \leftarrow \varphi(\mathbf{x}, \mathbf{z}),$$

where S is a concept or role name in the ontology and $\varphi(\mathbf{x}, \mathbf{z})$ a conjunction of atoms with database relations (both stored and views) and a *filter*, that is, a Boolean combination of built-in predicates such as = and <. (Note that, by including views in the schema, we can express any SQL query in mappings, which is important from the practical point of view.)

Given a mapping \mathcal{M} from a database schema to an OWL 2 QL TBox \mathcal{T} and an instance \mathbf{I} of this schema, the ground atoms

$$S(\mathbf{a}), \quad \text{for } S(\mathbf{x}) \leftarrow \varphi(\mathbf{x}, \mathbf{z}) \text{ in } \mathcal{M} \text{ and } \mathbf{I} \models \exists \mathbf{z} \varphi(\mathbf{a}, \mathbf{z}),$$

comprise the ABox, $\mathcal{A}_{\mathbf{I}, \mathcal{M}}$, which is called the *virtual ABox* for \mathcal{M} over \mathbf{I} [59] (this ABox is just a convenient presentational tool and does not have to be materialised by the system). We can now define *certain answers* to a CQ \mathbf{q} over \mathcal{T} linked by \mathcal{M} to \mathbf{I} as certain answers to \mathbf{q} over $(\mathcal{T}, \mathcal{A}_{\mathbf{I}, \mathcal{M}})$.

As an illustration, we consider a (simplified) database IMDb (`www.imdb.com/interfaces`), whose schema contains relations *title*[m, t, y] with information about movies (ID, title, production year), and *castinfo*[p, m, r] with information

about movie casts (person ID, movie ID, person role). Thus, a data instance I of this schema may contain the tables

<i>title</i>		
<i>m</i>	<i>t</i>	<i>y</i>
728	‘Django Unchained’	2012

<i>castinfo</i>		
<i>p</i>	<i>m</i>	<i>r</i>
n37	728	1
n38	728	1

The users are not supposed to know the structure of the database. Instead, they are given an ontology, say MO (www.movieontology.org), describing the application domain in terms of, for example, concepts $mo:Movie$ and $mo:Person$, and roles $mo:cast$ and $mo:year$:

$$\begin{aligned}
 mo:Movie &\equiv \exists mo:title, & mo:Movie &\sqsubseteq \exists mo:year, \\
 mo:Movie &\equiv \exists mo:cast, & \exists mo:cast^- &\sqsubseteq mo:Person.
 \end{aligned}$$

A mapping \mathcal{M} that relates the ontology terms to the database schema contains, for example, the following rules:

$$mo:Movie(m) \leftarrow title(m, t, y), \quad (15)$$

$$mo:title(m, t) \leftarrow title(m, t, y), \quad (16)$$

$$mo:year(m, y) \leftarrow title(m, t, y), \quad (17)$$

$$mo:cast(m, p) \leftarrow castinfo(p, m, r), \quad (18)$$

$$mo:Person(p) \leftarrow castinfo(p, m, r). \quad (19)$$

Then the virtual ABox $\mathcal{A}_{I, \mathcal{M}}$ for \mathcal{M} over I consists of the ground atoms

$$\begin{aligned}
 &mo:Movie(728), \quad mo:title(728, \text{‘Django Unchained’}), \quad mo:year(728, 2012), \\
 &mo:Person(n37), \quad mo:cast(728, n37), \\
 &mo:Person(n38), \quad mo:cast(728, n38).
 \end{aligned}$$

Given a CQ q and an ontology \mathcal{T} , one could first construct a rewriting q' of q and \mathcal{T} over *arbitrary* ABoxes. The rewriting q' could then be *unfolded* into an SQL query by using *partial evaluation* [43], which exhaustively applies SLD-resolution to q' and the mapping \mathcal{M} and returns those rules whose bodies contain only database atoms. Consider the simple CQ $q(x) = mo:Movie(x)$. An obvious rewiring of q and the TBox above (over arbitrary ABoxes) contains the following three rules:

$$q'(x) \leftarrow mo:Movie(x), \quad (20)$$

$$q'(x) \leftarrow mo:title(x, y), \quad (21)$$

$$q'(x) \leftarrow mo:cast(x, y). \quad (22)$$

The unfolding applies the SLD-resolution procedure to these three rules and the mapping \mathcal{M} and produces the rules:

$$q'(x) \leftarrow title(x, t, y), \quad (20+15)$$

$$q'(x) \leftarrow title(x, t, y), \quad (21+16)$$

$$q'(x) \leftarrow castinfo(p, x, r). \quad (22+18)$$

The resulting union of SELECT-PROJECT-JOIN queries could then be forwarded for execution to an RDBMS.

One can achieve the same result by using the tree-witness rewriting q_{tw} of q and \mathcal{T} over H-complete ABoxes introduced in Section 3. An obvious way to construct H-complete ABoxes is to take the composition of \mathcal{M} and the inclusions in \mathcal{T} , that is, a mapping $\mathcal{M}^{\mathcal{T}}$ given by

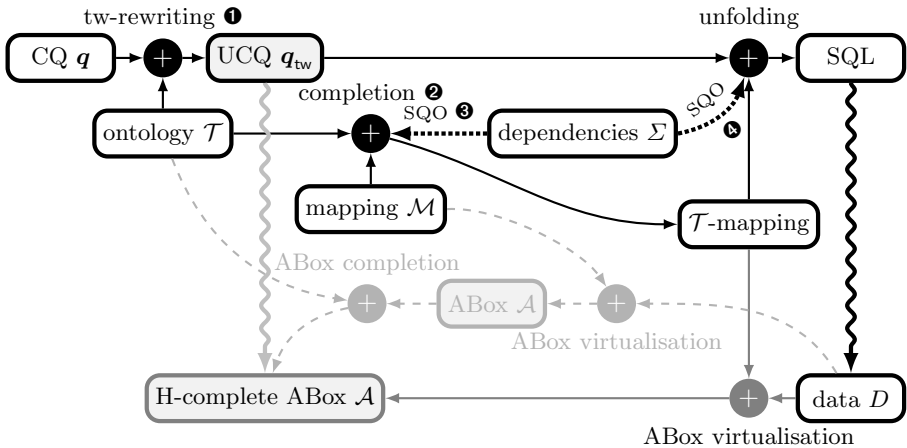
$$\begin{aligned} A(x) \leftarrow \varphi(x, z), & \quad \text{if } A'(x) \leftarrow \varphi(x, z) \in \mathcal{M} \text{ and } \mathcal{T} \models A' \sqsubseteq A, \\ A(x) \leftarrow \varphi(x, y, z), & \quad \text{if } R(x, y) \leftarrow \varphi(x, y, z) \in \mathcal{M} \text{ and } \mathcal{T} \models \exists R \sqsubseteq A, \\ P(x, y) \leftarrow \varphi(x, y, z), & \quad \text{if } R(x, y) \leftarrow \varphi(x, y, z) \in \mathcal{M} \text{ and } \mathcal{T} \models R \sqsubseteq P. \end{aligned}$$

(Recall that we do not distinguish between $P^-(y, x)$ and $P(x, y)$.) Thus, for any I and any tuple \mathbf{a} of individuals in $\mathcal{A}_{I, \mathcal{M}}$, we have:

$$(\mathcal{T}, \mathcal{A}_{I, \mathcal{M}}) \models \mathbf{q}(\mathbf{a}) \quad \text{iff} \quad \mathcal{A}_{I, \mathcal{M}^{\mathcal{T}}} \models \mathbf{q}_{\text{tw}}(\mathbf{a}). \quad (23)$$

So, to compute the answers to q over \mathcal{T} linked by \mathcal{M} to I , one can unfold the tree-witness rewriting q_{tw} over H-complete ABoxes with the help of the composition $\mathcal{M}^{\mathcal{T}}$. However, the resulting query will produce duplicating answers if the ontology axioms express the same properties of the application domain as the integrity constraints of the database [58]. For example, the IMDb schema contains a foreign key: movie ID in *castinfo* references movie ID in *title*, and therefore the unfolded rewriting above will return the same movie many times—once from *title* and once for each of the cast members of the movie in *castinfo*. Such a duplication is clearly an undesirable feature of this straightforward approach.

For this reason, before applying $\mathcal{M}^{\mathcal{T}}$ to unfold the tree-witness rewriting, *Ontop* optimises the mapping using the database integrity constraints Σ . This allows us to (a) reduce redundancy in answers, and (b) substantially shorten the SQL queries, which makes the OBDA system more efficient. The process of query rewriting and unfolding in *Ontop* with all optimisations is shown in the picture below (the dashed lines illustrate processes that do *not* take place):



The key ingredients of the architecture of *Ontop* are as follows:

- ❶ the tree-witness rewriting q_{tw} assumes the virtual ABoxes to be H-complete; it separates the topology of q from the taxonomy defined by \mathcal{T} , is fast in practice and produces short UCQs as demonstrated for real-world ontologies and queries [60];
- ❷ the \mathcal{T} -mapping combines the system mapping \mathcal{M} with the taxonomy of \mathcal{T} to ensure H-completeness of virtual ABoxes;
- ❸ the \mathcal{T} -mapping is simplified using the Semantic Query Optimisation (SQO) technique and SQL features; the \mathcal{T} -mapping is constructed and optimised for the given \mathcal{T} and Σ only once, and is used for unfolding all rewritings q_{tw} ;
- ❹ the unfolding algorithm uses SQO to produce small and efficient SQL queries.

We illustrate the last three items in the remainder of this section.

6.1 \mathcal{T} -Mappings

We say that a mapping \mathcal{M} is a *\mathcal{T} -mapping over dependencies Σ* if the ABox $\mathcal{A}_{I,\mathcal{M}}$ is H-complete with respect to \mathcal{T} , for any data instance I satisfying Σ . The composition $\mathcal{M}^{\mathcal{T}}$ defined above is trivially a \mathcal{T} -mapping over any Σ . *Ontop* starts with $\mathcal{M}^{\mathcal{T}}$ and then applies a series of optimisations to construct a simpler \mathcal{T} -mapping.

Inclusion Dependencies. Suppose $\mathcal{M} \cup \{S(\mathbf{x}) \leftarrow \psi_1(\mathbf{x}, \mathbf{z})\}$ is a \mathcal{T} -mapping over Σ . If there is a more specific rule than $S(\mathbf{x}) \leftarrow \psi_1(\mathbf{x}, \mathbf{z})$ in \mathcal{M} , then \mathcal{M} itself will also be a \mathcal{T} -mapping. To discover such ‘more specific’ rules, we run the standard query containment check (see, e.g., [1]) taking account of the inclusion dependencies. For example, since $\mathcal{T} \models \exists mo:cast \sqsubseteq mo:Movie$ in our running example, the composition \mathcal{M}^{MO} of the mapping \mathcal{M} given above and MO contains the following rules for *mo:Movie*:

$$\begin{aligned} mo:Movie(m) &\leftarrow title(m, t, y), \\ mo:Movie(m) &\leftarrow castinfo(p, m, r). \end{aligned}$$

The latter rule is redundant since IMDb contains the foreign key (an inclusion dependency)

$$\forall m (\exists p, r \text{ castinfo}(p, m, r) \rightarrow \exists t, y \text{ title}(m, t, y)).$$

Disjunctions in SQL. Another way to reduce the size of a \mathcal{T} -mapping is to identify pairs of rules whose bodies are equivalent up to *filters with respect to constant values*. This optimisation deals with the rules introduced due to the so-called type (discriminating) attributes [19] in database schemas. For example, the mapping \mathcal{M} for IMDb and MO contains six rules for sub-concepts of *mo:Person*:

$$\begin{aligned} mo:Actor(p) &\leftarrow castinfo(c, p, m, r), (r = 1), \\ &\dots \\ mo:Editor(p) &\leftarrow castinfo(c, p, m, r), (r = 6). \end{aligned}$$

Then the composition \mathcal{M}^{MO} contains six rules for $mo:Person$ that differ only in the last condition ($r = k$), $1 \leq k \leq 6$. These can be reduced to a single rule:

$$mo:Person(p) \leftarrow castinfo(c, p, m, r), (r = 1) \vee \dots \vee (r = 6).$$

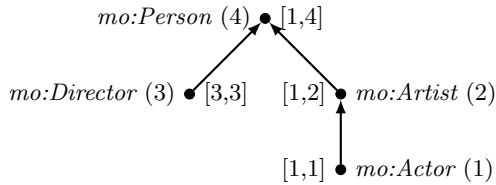
Note that such disjunctions lend themselves to efficient evaluation by RDBMSs.

Materialised ABoxes and Semantic Index. In addition to working with proper relational data sources, *Ontop* also supports ABox storage in the form of structureless *universal tables*: a binary relation $CA[id, concept-id]$ and a ternary relation $RA[id_1, id_2, role-id]$ represent concept and role assertions. The universal tables give rise to trivial mappings, and *Ontop* implements a technique, the *semantic index* [59], that takes advantage of SQL features in \mathcal{T} -mappings for this scenario. The key observation is that, since the IDs in the universal tables CA and RA can be chosen by the system, each concept and role in the TBox \mathcal{T} can be assigned a numeric *index* and a set of numerical *intervals* in such a way that the resulting \mathcal{T} -mapping contains simple SQL queries with interval filter conditions. For example, in IMDb, we have

$$\begin{aligned} mo:Actor &\sqsubseteq mo:Artist, \\ mo:Artist &\sqsubseteq mo:Person, \\ mo:Director &\sqsubseteq mo:Person, \end{aligned}$$

so we can choose indexes and intervals for these concepts as in the table below:

concept	index	interval
$mo:Actor$	1	[1,1]
$mo:Artist$	2	[1,2]
$mo:Director$	3	[3,3]
$mo:Person$	4	[1,4]



It can be seen that these intervals respect the concept inclusions of the TBox: e.g., [1,1] for $mo:Actor$ is a subset of [1,2] for $mo:Artist$. This will generate a \mathcal{T} -mapping with

$$\begin{aligned} mo:Actor(p) &\leftarrow CA(p, concept-id), (concept-id = 1), \\ mo:Artist(p) &\leftarrow CA(p, concept-id), (1 \leq concept-id \leq 2), \\ mo:Director(p) &\leftarrow CA(p, concept-id), (concept-id = 3), \\ mo:Person(p) &\leftarrow CA(p, concept-id), (1 \leq concept-id \leq 4). \end{aligned}$$

Thus, by choosing appropriate concept and role IDs, we effectively construct H-complete ABoxes *without* the expensive forward chaining procedure (and the need to store large amounts of derived assertions). On the other hand, the semantic index \mathcal{T} -mappings are based on range expressions that can be evaluated efficiently by RDBMSs using standard B-tree indexes [19].

6.2 Unfolding with Semantic Query Optimisation

Ontop applies the Semantic Query Optimisation (SQO) [14] to rules obtained at the intermediate steps of unfolding. In particular, this eliminates redundant JOIN operations caused by reification of database relations by means of concepts and roles. Consider, for example, the CQ

$$\mathbf{q}(t, y) \leftarrow mo:Movie(m), mo:title(m, t), mo:year(m, y), (y > 2010).$$

It has no tree witnesses, and so $\mathbf{q}_{tw} = \mathbf{q}$. By straightforwardly applying the unfolding to \mathbf{q}_{tw} and the \mathcal{T} -mapping \mathcal{M} above, we obtain the query

$$\mathbf{q}'_{tw}(t, y) \leftarrow title(m, t_0, y_0), title(m, t, y_1), title(m, t_2, y), (y > 2010),$$

which requires two (potentially) expensive JOIN operations. However, by using the primary key m of *title*:

$$\begin{aligned} &\forall m \forall t_1 \forall t_2 (\exists y title(m, t_1, y) \wedge \exists y title(m, t_2, y) \rightarrow (t_1 = t_2)), \\ &\forall m \forall y_1 \forall y_2 (\exists t title(m, t, y_1) \wedge \exists t title(m, t, y_2) \rightarrow (y_1 = y_2)) \end{aligned}$$

(a functional dependency with determinant m), we reduce two JOIN operations in the first three atoms of \mathbf{q}'_{tw} to a single atom $title(m, t, y)$:

$$\mathbf{q}''_{tw}(t, y) \leftarrow title(m, t, y), (y > 2010).$$

Note that these two JOIN operations were introduced to reconstruct the ternary relation from its reification by means of the roles *mo:title* and *mo:year*.

The role of SQO in OBDA systems appears to be much more prominent than in conventional RDBMSs, where it was initially proposed to optimise SQL queries. While some of SQO techniques reached industrial RDBMSs, it never had a strong impact on the database community because it is costly compared to statistics- and heuristics-based methods, and because most SQL queries are written by highly-skilled experts (and so are nearly optimal anyway). In OBDA scenarios, in contrast, SQL queries are generated automatically, and so SQO becomes the only tool to avoid redundant and expensive JOIN operations.

6.3 Why Does It Work?

The techniques above prove to be very efficient in practice. Moreover, they often automatically produce queries that are similar to those written by human experts. To understand why, we briefly review the process of designing database applications [19]. It starts with conceptual modelling which describes the application domain in such formalisms as ER, UML or ORM. The conceptual model gives the vocabulary of the database and defines its semantics by means of hierarchies, cardinality restrictions, etc. The conceptual model is turned into a relational database by applying a series of *standard* procedures that *encode* the semantics of the model into a relational schema. These procedures include:

- amalgamating many-to-one and one-to-one attributes of an entity to a single n -ary relation with a primary key identifying the entity (e.g., *title* with *mo:title* and *mo:year*);
- using foreign keys over attribute columns when a column refers to the entity (e.g., *title* and *castinfo*);
- using type (discriminating) attributes to encode hierarchical information (e.g., *castinfo*).

As this process is universal, the \mathcal{T} -mappings created for the resulting databases are dramatically simplified by the *Ontop* optimisations, and the resulting UCQs are usually of acceptable size and can be executed efficiently by RDBMSs.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Alon, N., Boppana, R.: The monotone circuit complexity of Boolean functions. *Combinatorica* 7(1), 1–22 (1987)
3. Arora, S., Barak, B.: Computational Complexity: A Modern Approach, 1st edn. Cambridge University Press, New York (2009)
4. Artale, A., Calvanese, D., Kontchakov, R., Zakharyashev, M.: The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR)* 36, 1–69 (2009)
5. Artale, A., Ryzhikov, V., Kontchakov, R.: DL-Lite with attributes and datatypes. In: Proc. of the 20th European Conf. on Artificial Intelligence (ECAI 2012). *Frontiers in Artificial Intelligence and Applications*, vol. 242, pp. 61–66. IOS Press (2012)
6. Bienvenu, M., Ortiz, M., Šimkus, M., Xiao, G.: Tractable queries for lightweight description logics. In: Proc. of the 23 Int. Joint Conf. on Artificial Intelligence (IJCAI 2013) (2013)
7. Bienvenu, M., ten Cate, B., Lutz, C., Wolter, F.: Ontology-based data access: A study through disjunctive Datalog, CSP and MMSNP. In: Proc. of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2013). ACM (2013)
8. Borodin, A., von zur Gathen, J., Hopcroft, J.: Fast parallel matrix and gcd computations. In: Proc. of the 23rd Annual Symposium on Foundations of Computer Science (FOCS 1982), pp. 65–71 (1982)
9. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantics* 14, 57–83 (2012)
10. Calì, A., Gottlob, G., Pieris, A.: Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence* 193, 87–128 (2012)
11. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *Journal of Automated Reasoning* 39(3), 385–429 (2007)
12. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), pp. 260–270 (2006)
13. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: View-based query answering in description logics: Semantics and complexity. *Journal of Computer and System Sciences* 78(1), 26–46 (2012)

14. Chakravarthy, U.S., Fishman, D.H., Minker, J.: Semantic query optimization in expert systems and database systems. Benjamin-Cummings Publishing Co., Inc. (1986)
15. Chortaras, A., Trivela, D., Stamou, G.: Optimized query rewriting for OWL 2 QL. In: Björner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS, vol. 6803, pp. 192–206. Springer, Heidelberg (2011)
16. Dolby, J., Fokoue, A., Kalyanpur, A., Ma, L., Schonberg, E., Srinivas, K., Sun, X.: Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 403–418. Springer, Heidelberg (2008)
17. Eiter, T., Ortiz, M., Šimkus, M., Tran, T.-K., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI 2012). AAAI Press (2012)
18. Eiter, T., Ortiz, M., Šimkus, M.: Conjunctive query answering in the description logic SH using knots. *Journal of Computer and System Sciences* 78(1), 47–85 (2012)
19. Elmasri, R., Navathe, S.: *Fundamentals of Database Systems*, 6th edn. Addison-Wesley (2010)
20. Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York (1979)
21. Glimm, B., Lutz, C., Horrocks, I., Sattler, U.: Conjunctive query answering for the description logic SHIQ. *Journal of Artificial Intelligence Research (JAIR)* 31, 157–204 (2008)
22. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: Proc. of the 27th Int. Conf. on Data Engineering (ICDE 2011), pp. 2–13. IEEE Computer Society (2011)
23. Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive datalog programs. In: Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012). AAAI Press (2012)
24. Grigni, M., Sipser, M.: Monotone separation of logarithmic space from logarithmic depth. *Journal of Computer and System Sciences* 50(3), 433–437 (1995)
25. Grohe, M., Schwentick, T., Segoufin, L.: When is the evaluation of conjunctive queries tractable? In: Proc. of the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001), pp. 657–666. ACM (2001)
26. Heymans, S., Ma, L., Anicic, D., Ma, Z., Steinmetz, N., Pan, Y., Mei, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Feier, C., Hench, G., Wetzstein, B., Keller, U.: Ontology reasoning with large data repositories. In: *Ontology Management, Semantic Web, Semantic Web Services, and Business Applications. Semantic Web And Beyond Computing for Human Experience*, vol. 7, pp. 89–128. Springer (2008)
27. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive datalog. *Journal of Automated Reasoning* 39(3), 351–384 (2007)
28. Immerman, N.: *Descriptive Complexity*. Springer (1999)
29. Jukna, S.: *Boolean Function Complexity — Advances and Frontiers. Algorithms and combinatorics*, vol. 27. Springer (2012)
30. Kikot, S., Kontchakov, R., Podolskii, V., Zakharyashev, M.: Query rewriting over shallow ontologies. In: Proc. of the 2013 International Workshop on Description Logics (DL 2013) (2013)
31. Kikot, S., Kontchakov, R., Zakharyashev, M.: On (In)Tractability of OBDA with OWL 2 QL. In: Proc. of the 2011 International Workshop on Description Logics (DL 2011). CEUR-WS, vol. 745 (2011)

32. Kikot, S., Kontchakov, R., Zakharyashev, M.: Conjunctive query answering with OWL 2 QL. In: Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012). AAAI Press (2012)
33. Kikot, S., Tsarkov, D., Zakharyashev, M., Zolin, E.: Query answering via modal definability with FaCT++: First blood. In: Proc. of the 2013 International Workshop on Description Logics (DL 2013) (2013)
34. Kikot, S., Kontchakov, R., Podolskii, V., Zakharyashev, M.: Exponential lower bounds and separation for query rewriting. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part II. LNCS, vol. 7392, pp. 263–274. Springer, Heidelberg (2012)
35. König, M., Leclère, M., Mugnier, M.-L., Thomazo, M.: A sound and complete backward chaining algorithm for existential rules. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 122–138. Springer, Heidelberg (2012)
36. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyashev, M.: The combined approach to query answering in DL-Lite. In: Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2010). AAAI Press (2010)
37. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyashev, M.: The combined approach to ontology-based data access. In: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2011), pp. 2656–2661. AAAI Press (2011)
38. Kozen, D.: Theory of Computation. Springer (2006)
39. Krisnadhi, A., Lutz, C.: Data complexity in the EL family of description logics. In: Proc. of the 2007 International Workshop on Description Logics (DL 2007). CEUR-WS, vol. 250, pp. 88–99 (2007)
40. Lachish, O., Raz, R.: Explicit lower bound of $4.5n - o(n)$ for Boolean circuits. In: Proc. of the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001), pp. 399–408. ACM (2001)
41. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2002), pp. 233–246. ACM (2002)
42. Libkin, L.: Elements of Finite Model Theory. Springer (2004)
43. Lloyd, J.W., Shepherdson, J.C.: Partial Evaluation in Logic Programming. The Journal of Logic Programming 11(3-4), 217–242 (1991)
44. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic EL using a relational database system. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009), pp. 2070–2075 (2009)
45. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 179–193. Springer, Heidelberg (2008)
46. Lutz, C., Seylan, I., Toman, D., Wolter, F.: The combined approach to OBDA: taming role hierarchies using filters. In: Proc. of the Joint Workshop on Scalable and High-Performance Semantic Web Systems (SSWS+HPCSW 2012). CEUR-WS, vol. 943, pp. 16–31 (2012)
47. Lutz, C., Wolter, F.: Non-uniform data complexity of query answering in description logics. In: Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012). AAAI Press (2012)
48. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: Proc. of the 12th Annual Symposium on Switching and Automata Theory (SWAT 1971), pp. 188–191 (1971)

49. Ortiz, M., Calvanese, D., Eiter, T.: Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning* 41(1), 61–98 (2008)
50. Ortiz, M., Rudolph, S., Simkus, M.: Query answering in the Horn fragments of the description logics SHOIQ and SROIQ. In: *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2011)*, pp. 1039–1044. AAAI Press (2011)
51. Papadimitriou, C.H., Yannakakis, M.: On the complexity of database queries. In: *Proc. of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 1997)*, pp. 12–19. ACM Press (1997)
52. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for DL-Lite. In: *Proc. of the 2009 International Workshop on Description Logics (DL 2007)*. CEUR-WS, vol. 477 (2009)
53. Pérez-Urbina, H., Rodríguez-Díaz, E., Grove, M., Konstantinidis, G., Sirin, E.: Evaluation of query rewriting approaches for OWL 2. In: *Proc. of the Joint Workshop on Scalable and High-Performance Semantic Web Systems (SSWS+HPCSW 2012)*. CEUR-WS, vol. 943, pp. 32–44 (2012)
54. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. In: Spaccapietra, S. (ed.) *Journal on Data Semantics X*. LNCS, vol. 4900, pp. 133–173. Springer, Heidelberg (2008)
55. Raz, R., McKenzie, P.: Separation of the monotone NC hierarchy. In: *Proc. of the 38th Annual Symposium on Foundations of Computer Science (FOCS 1997)*, pp. 234–243 (1997)
56. Raz, R., Wigderson, A.: Monotone circuits for matching require linear depth. *Journal of the ACM* 39(3), 736–744 (1992)
57. Razborov, A.: Lower bounds for the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR* 281(4), 798–801 (1985)
58. Rodríguez-Muro, M.: Tools and Techniques for Ontology Based Data Access in Lightweight Description Logics. PhD thesis, KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano (2010)
59. Rodríguez-Muro, M., Calvanese, D.: Dependencies: Making ontology based data access work. In: *Proc. of the 5th A. Mendelzon Int. Workshop on Foundations of Data Management (AMW 2011)*. CEUR-WS, vol. 749 (2011)
60. Rodríguez-Muro, M., Kontchakov, R., Zakharyashev, M.: Ontop at work. In: *Proc. of the 10th OWL: Experiences and Directions Workshop (OWLED 2013)* (2013)
61. Rosati, R.: Prexto: Query rewriting under extensional constraints in DL-Lite. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) *ESWC 2012*. LNCS, vol. 7295, pp. 360–374. Springer, Heidelberg (2012)
62. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: *Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2010)*. AAAI Press (2010)
63. Rosati, R.: On conjunctive query answering in EL. In: *Proc. of the 2007 International Workshop on Description Logics (DL 2007)*. CEUR-WS, vol. 250 (2011)
64. Savkovic, O., Calvanese, D.: Introducing datatypes in DL-Lite. In: *Proc. of the 20th European Conf. on Artificial Intelligence (ECAI 2012)*. *Frontiers in Artificial Intelligence and Applications*, vol. 242, pp. 720–725. IOS Press (2012)
65. Shannon, C.E.: The synthesis of two-terminal switching circuits. *Bell Systems Technical Journal* 28, 59–98 (1949)

A Geo-semantics Flyby

Krzysztof Janowicz¹, Simon Scheider², and Benjamin Adams³

¹ Department of Geography, University of California, Santa Barbara, CA, USA

² Institute for Geoinformatics, University of Münster, Germany

³ National Center for Ecological Analysis and Synthesis (NCEAS)
University of California, Santa Barbara, CA, USA

Abstract. Geospatial semantics as a research field studies how to publish, retrieve, reuse, and integrate geo-data, how to describe geo-data by conceptual models, and how to develop formal specifications on top of data structures to reduce the risk of incompatibilities. Geo-data is highly heterogeneous and ranges from qualitative interviews and thematic maps to satellite imagery and complex simulations. This makes ontologies, semantic annotations, and reasoning support essential ingredients towards a Geospatial Semantic Web. In this paper, we present an overview of major research questions, recent findings, and core literature.

1 Introduction and Motivation

A flyby is a flight maneuver to celebrate an important event, to demonstrate aircraft, or to showcase flying skills. Independent of the particular purpose, the audience will see the aircraft approaching from afar, catch a more detailed glimpse of certain parts when the aircraft passes by, and then see the tail disappear in the sky. Using the flyby as a metaphor, we will give a broader overview of the general field of geo-semantics first, later highlight some selected topics in more detail, and also touch upon a few topics of emerging interest. The selection of these topics is biased, and, as with the flyby, depends on the viewer's vantage point. We will assume that the reader is familiar with the core concepts of the Semantic Web, but not with geo-semantics, the broader Geosciences, or Geographic Information Science. Consequently, we will focus on intuitive examples that do not require domain knowledge but nonetheless illustrate the research challenges. For those interested in a detailed introduction to the Semantic Web and related core technologies, we refer the reader to the recent textbook by Hitzler et al. [42].

Before we dive into the discussion, it is worth clarifying some of the terminology we will use. Much like Bioinformatics combines Computer and Information Science with Biology to improve handling and analyzing biological data, Geoinformatics is an interdisciplinary research field concerned with geo-data in their broadest definition. *Geo*, in this context, refers to the Earth Sciences, Geography, Ecology, and related research fields. Geographic Information Science (GIScience) puts more emphasis on geographic aspects and qualitative as well as quantitative data. It is closely related to geographic information systems (GIS), which are software and services to manage and analyze geographic data. These systems are used, for example, to reason about crime densities, optimize location

choices, visualize land use dynamics, and so forth.¹ GIS and spatial statistics play an important role in many domains such as economics, health research, and archaeology. In fact, it is often claimed that most data has some sort of spatial reference. Sometimes the name Spatial Informatics is used to emphasize the integrative role and omnipresence of spatial aspects in many datasets and to broaden the research beyond the geo-realm. The term *geospatial* is used to explicitly restrict the scale to between 10^{-2} - 10^7 meters. That is, everything smaller than a grain of sand and larger than the Earth is usually not considered to be part of the geo-domain. Finally, as the Geosciences are largely concerned with processes, the temporal dimension is implicit, and thus geo-data is typically *spatiotemporal*.

In order to illustrate some of the complexities and interesting challenges of geo-semantics we would like to start with what on the surface might appear to be a trivial task. That is, we want to describe the semantics of the term *mountain*. *Mountain* is one of many geographic feature types, including *river*, *lake*, *forest*, and *city*, that we are accustomed to using in everyday language and for which we have developed internal notions through our social and physical experiences. This knowledge, which has been described as *Naive Geography* by Egenhofer and Mark [25], is used by people every day to reason about the *surrounding geographic world*. Given our familiarity with these terms, we are often quick to jump to conclusions and assume a shared understanding, while, in fact, most of these terms have dozens, domain-specific, and often incompatible meanings. The fact that we want to formalize the semantics of terms for which almost everyone has common-sense understandings makes this a challenging task.

One method to formalize the notion of *mountain* is to define a minimum height for the mountain as a necessary property. This technique has been adopted in the United Kingdom and was used to humorous effect in the movie *The Englishman Who Went Up a Hill But Came Down a Mountain*. In that movie, the members of a fictional Welsh community are dismayed to learn that their local mountain is not, in fact, a *mountain* at all but rather a *hill*, because it is a couple feet under the threshold of 1000 feet. They, however, successfully get the mountain classified as a *mountain* by adding a small pile of earth on the top. This story illustrates an important point about the semantics of geographic features. The definitions of feature types are a product of human perception, cognition, current state of knowledge, and social agreement. There is no human-independent *true* definition of *mountain* (or *forest*, *river*, *city*, etc.). Consequently, the definitions will vary between places and cultures, and it is important to represent local definitions appropriately. The importance of the local geographic context to understand the meaning of geographic terms is further illustrated by the map shown in Figure 1. This schematic map from 1846 was designed to represent the *principal* mountains and rivers of the Earth. The set of principal rivers and mountains are determined by length and height, respectively, but within context of the part of the Earth in which the feature is found. As a result, the principal mountains

¹ The distinction between Geoinformatics and GIScience is not crisp and mainly an artifact of their parallel evolution in Europe and United States.



Fig. 1. Schematic map (1846) showing *principal* mountains and rivers of the world [73]

of England, such as Scafell Pike, the tallest mountain in England at a relative height of 912 meters, would barely be hills in the Himalayas.

Apart from the notion of context based on local jurisdictions and communities, since geospatial terms like *mountain* are based on perception, their meanings can be highly situation-dependent. For example, imagine a geo-semantics informed location-based service that is designed to give wayfinding instructions. A human that gives a route description might say, “take a right and walk toward the mountain,” where the mountain in question is a clearly identifiable higher-elevation landmark feature. The meaning of *mountain* in this case is not based on any canonical definition but entirely on the situated context of the given route [13,6].

How does this impact searching and integrating geo-data in the Linked Data Cloud and the Semantic Web? Let us assume you are interested in studying the role of the forest industry in rural economics. For instance, you may be interested in migration and depopulation, government policies, or the changing role of forestry in the context of ecological and amenity services. While we will use a simplified example here, this use case is real and was, for instance, addressed by Elands and Wiersum [26]. Suppose terms such as *forest*, *town*, *farm*, and *countryside* are used

without making their intended meaning explicit. Suppose further that you would like to query for *towns near forests* such as in the SPARQL query shown below, and you plan to use the retrieved towns to conduct your analysis.²

```
[...]
SELECT distinct ?town ?forest
WHERE {
    ?town
        geo-pos:lat ?lat ;
        geo-pos:long ?long ;
        a dbp-ont:Town .
    ?forest
        omgeo:nearby(?lat ?long "25mi") ;
        a dbp-ont:Forest .
}[...]
```

No matter what the query will return and how you will process and analyze the data from those thousands of towns, your results will be misleading at best. Most likely you will have overlooked that among all those small populated places, your dataset will also contain Los Angeles, Stuttgart,³ and other metropolises. The reason for this apparently odd result is that the class *city* and *town* are defined to be equivalent by Californian law. In fact, most of US states have their own legal definition of these terms. While some rely on maximum population as a criterion, others do not [47]. The situation for forests is even more complicated. Lund [66], for instance, lists over 900 different definitions for *forest*, *afforestation*, and related terms. These definitions are not without consequences but often legally binding. In the past, loop holes in these definitions have been used for massive deforestation.⁴ Finally, the alert reader may be wondering why a radius of 25 miles is used in the example above to define *nearby*. First, as with many other terms, the semantics of *nearby* is context-dependent. Second, unfortunately, most of today's Linked Data represents geographic features by their centroids (geometric center points) instead of polygons. Thus, even if a GIS would represent a particular town and forest by two adjacent polygons, their centroids may still be miles apart; see [8] for more details on spatial queries over Linked Data.

As these examples show, understanding what the authors of a scientific study or data providers in general mean by apparently obvious terms is a difficult task. Without better geo-ontologies, semantically annotated (meta) data, and more complex ontology alignment services that can map between local ontologies,

² This query will fail as the class *Forest* (or similar classes) are not defined in DBpedia. However, querying for Mountains, for instance, would work.

³ Stuttgart is described as the 'sixth-largest city in Germany' in DBpedia but classified as a town via `dbpedia:Stuttgart rdf:type dbpedia-owl:Town`; see <http://live.dbpedia.org/page/Stuttgart>.

⁴ Readers interested in deforestation and in combining SPARQL with spatial statistics in R may want to check the new Linked Brazilian Amazon Rainforest Dataset[52].

reusing and integrating data from heterogeneous sources is merely a distant dream.

Geospatial terms are often taken for granted, but they pose interesting challenges once we want to formally describe their semantics in information systems and share them in an environment such as the Linked Data. Interestingly, the problem is not that machines cannot communicate, but that humans misunderstand each other when communicating via machines [86]. It is worth noting that geo-semantics research is therefore not interested in *overcoming* or *resolving* semantic heterogeneity. Local definitions exist for good reasons. If one could just standardize meaning globally, there would be no need for Semantic Web research. Instead, geo-semantics tries to restrict the interpretation of domain-vocabulary towards their intended meanings, map and translate between local conceptualizations, and try to reduce the risk of combining incompatible data [60,44].

2 Using Geospatial Referents on the Semantic Web

Geospatial and spatiotemporal phenomena, such as places (*Florida*), geographic objects (*the Eiffel tower*), and events (*Hurricane Katrina*), serve as central referents on the Semantic Web. Spatial relations between these phenomena, like *above*, *below*, *in front*, *north of*, *overlaps*, *contains*, serve to localize them relative to each other. Geospatial information maps these phenomena to points or regions in spatial reference systems and temporal reference systems to ensure their interpretation. This makes them amenable not only for cartographic mapping, but also for locational querying and comparison. It is therefore not surprising that many linked datasets either contain spatiotemporal identifiers themselves or link out to such datasets, making them central hubs of the Linked Data cloud.

Figure 2 depicts point-like features classified as *places* extracted from a representative fraction of the Linked Data cloud using SPARQL endpoints and their geindexing capabilities. It is remarkable that the figure does not contain a base map, but is entirely composed of millions of extracted points. In other words, the coverage of *places* on the Linked Data cloud is very high. On the downside, the map also shows massive (and often systematic) errors; e.g., the huge cross in the middle of the map. For lack of space we do not discuss these errors here.⁵

Prominent examples of geo-specific, yet general purpose Linked Data include Geonames.org as well as the Linked Geo Data project, which provides a RDF serialization of Points Of Interest from Open Street Map [91]. Besides such voluntary geographic information (VGI), governments and governmental agencies started to develop geo-ontologies and publish *Linked Spatiotemporal Data* [33,48]; see Table 1. Examples include the US Geological Survey [95] and the UK Ordnance Survey [35]. Furthermore, many other Linked Data sources contain location-based references as well. To give a concrete example, data from the digital humanities may interlink information about particular exhibits to places and their historic names [69]. By following these outgoing links,

⁵ See http://stko.geog.ucsb.edu/location_linked_data for more details.



Fig. 2. A representative fraction of places in Linked Data (EPSG:4326, Plate Carrée)

researchers can explore those places and learn about events which took occurred there. The historic events data may in turn link to repositories about objects and actors that were involved in the described events [36].

Table 1. Selected Linked Data repositories and ontologies for geo-data

Linked Data repositories	
LinkedGeoData.org	http://linkedgeo.org/About
GeoLinkedData.es	http://geo.linkeddata.es/web/
Geo.Data.gov	http://geo.data.gov/geoportal/catalog/main/home.page
Ordnance Survey	http://data.ordnancesurvey.co.uk/.html
TaxonConcept	http://lsd.taxonconcept.org/sparql
USGS The National Map	http://cegis.usgs.gov/ontology.html
Linked Amazon Rainforest Data	http://linkedsience.org/data/linked-brazilian-amazon-rainforest/
Ontologies	
The Geonames Ontology	http://www.geonames.org/ontology/documentation.html
GAZ Ontology	http://gensc.org/gc_wiki/index.php/GAZ_Project
W3C Geospatial Ontologies	http://www.w3.org/2005/Incubator/geo/XGR-geo-ont-20071023/
TaxonConcept Ontologies	http://www.taxonconcept.org/ontologies/
GeoLinkedData.es Ontologies	http://geo.linkeddata.es/web/guest/modelos
Darwin Core	http://rs.tdwg.org/dwc/
GeoSPARQL Schemas	http://www.opengeospatial.org/standards/geosparql
U.S. Geological Survey Ontologies	http://cegis.usgs.gov/ontology.html
European INSPIRE Models	http://inspire.ec.europa.eu/index.cfm/pageid/2/list/datamodels

Arriving at such a network of interlinked repositories, however, is not trivial as geospatial phenomena come with a large degree of *semantic ambiguity*. This makes them challenging to use as semantic referents. Consider the problem of retrieving objects on a sports ground, such as the one depicted on the areal photograph in Figure 3, for the purpose of noise abatement planning [39]. For example, the goal may be to find objects that serve as referents for localizing sources of noise.

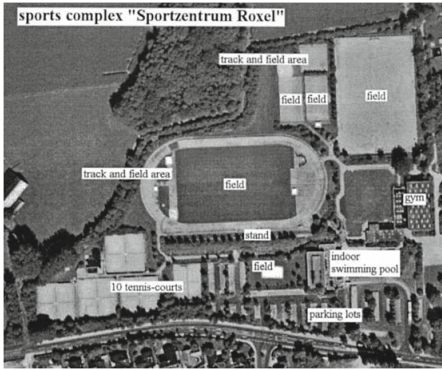


Fig. 3. Sportsground Roxel near Münster in an aerial photograph; taken from [39]



Fig. 4. The sportsground on Google maps

There are different kinds of maps (and datasets used to render them) of the same sports ground, showing different kinds of objects (Figures 5 and 6). Starting with the *cadastral* map in Figure 5, one can see that the most prominent feature, the track and soccer field area, is not depicted on the map. Similarly, the tennis courts are left out. This can be explained if we recall that a cadastral map depicts land parcels and ownerships and that the distinction between a soccer field and its surrounding area is not one of ownership. Thus, the German cadastre does not store this information, and, consequently, they are not rendered on the map. Google maps shares a similar but slightly different perspective as shown in Figure 4 and often reduces areal features to points.



Fig. 5. Cadastral map (ALK) of the Sportsground Roxel [39]

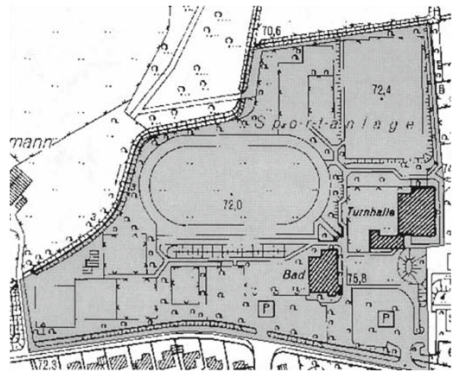


Fig. 6. Topographic map (DGK) of the Sportsground Roxel [39]

In contrast, the sports ground as such is represented in the cadastral map, since it can be distinguished from its surroundings precisely based on ownership.

The *topographic map* in Figure 6 shows the track area and the tennis courts but leaves out the soccer field. This can be explained if we recall that a topographic map is a map of ground surface features. There is a distinction in surface texture between the elliptic track area or the tennis courts on one hand and the lawn on the other hand. However, since our goal is to identify sources of noise, we are interested in identifying the soccer field as well. Surface texture does not allow to distinguish it from its embedding lawn. Soccer fields are not just physical objects, they are actual dispositions to play a game, indicated by linear signs on the lawn.⁶ Consequently, their identification requires yet a different perspective on the sports ground, and, thus, an additional data source. There are many different sources that could be used in addition such as yellow pages, human activity surveys, geo-social check-ins from location-based services and their semantic trajectories [98], just to name a few. Establishing identity between features from all those sources, i.e., declaring that the sports ground in dataset A is the same entity in the physical world as the sports ground in dataset B, is a major research challenges [37,38]. Once this relation is established and made explicit, e.g., by using *owl:sameAs*, the attribute spaces of the involved data sources can be conflated.

As this example illustrates, there is large variability in mapping a single area, and many geospatial concepts have an intrinsic multi-perspectival nature. They are situated, i.e., their interpretation depends on space and time, and sometimes on a group of researchers [13]. This frequently causes interoperability problems. However, if we make the inherent geospatial semantic perspectives explicit, we are able to distinguish them, understand their advantages and limitations, combine them, and to a certain degree also translate between them. Geo-semantics develops methods and tools for disambiguating and describing these spatial perspectives, and, thus, provides reliable referents for the Semantic Web, together with useful spatial relations between them. Both can be used to improve retrieval, re-usage, and interlinkage of data.

In this section, we discussed how geo-spatial phenomena are important referents that enable semantic linking between datasets and addressed complexities in representing these referents and establishing identities of geographic features due to semantic ambiguity. In the next section, we give an overview on the geo-semantics research field before discussing how it serves to address the aforementioned challenges.

3 Geo-semantics from 30.000 Feet

The research field of geo-semantics builds on GIScience/geoinformatics, spatial databases, cognitive science, Artificial Intelligence (AI), the Semantic Web, and other related research areas [59]. It focuses on the meaning of digital referents at a geographic scale, such as places, locations, events, and geographic objects in digital maps, geodatabases, and earth models. Geo-semantics applies and investigates a variety of methods ranging from top-down knowledge engineering

⁶ Similar to traffic locomotion infrastructure [85].

and logical deduction to bottom-up data mining and induction. It combines knowledge engineering with methods specific to GIScience, e.g., spatial reference systems, spatial reasoning, and geographic information analysis. Geo-semantics also extends work which originated in related disciplines. For example, it uses semantic similarity and analogy reasoning, which have a long research tradition in cognitive science, to enable semantics-based geographic information retrieval [45], and it combines geo-ontologies with spatial statistics, e.g., to study land cover change [4].

What are major research challenges that are addressed by the field of geo-semantics? One early challenge was to describe the semantics of Web services that provide measurements from spatially distributed sensors. For example, in order to simulate the spread of a potentially toxic gas plume, two services may be queried for wind direction observations. Both services may seem compatible as they return a string called *wind_direction* as output together with an integer ranging from 0 - 360°. However, they can have contradicting interpretations of what the returned values actually mean: *wind blows to* or *wind blows from*. Thus, sending observation values from both services to an evacuation simulation running on a Web Processing Service (WPS) will yield misleading results [78]. Besides such challenges that arise from integrating *heterogeneous data* and *combining services* [29], an important future task is to semantically describe *spatial prediction models* in order to enable data-model inter-comparison [75]. For example, spatial statistics and simulation models are an essential part of geospatial information technology, posing their own set of interoperability challenges. Another challenge for geo-semantics is *concept drift*. Most geospatial concepts are not static, they evolve over time and may even change abruptly. This leads to research challenges such as how to handle semantic aging [87]; i.e., how to preservation and maintenance of geo-data and ontologies over long periods of time to make them reusable for future generations. To give a concrete example, geo-referenced data about the distribution of species, temperature, and other ecological variables collected in the 1960s are used in recent studies as base line to study species turnover [14]. The interpretation of these datasets is difficult and time consuming as different spatial and semantic reference systems have been used and scientific workflows have evolved over the years.

Two major strands of scientific thought in geo-semantics can be differentiated, by analogy with Kuhn's [62] distinction between modeling vs. encoding on the Semantic Web. One is studying the design task of semantic modeling. How should geo-data be modeled in an information ontology? Which classes and relations are required to describe the meaning of spatiotemporal phenomena and to discover, capture, and query geospatial referents? Examples include work on geo-ontology engineering [30,54,12] and the formalization of spatial reasoning [19]. These spatial relations support localizing complex geometrical objects, such as *cities* or *forests*, relative to other referents, such as *roads* and *countries* [49]. These queries need to deal with indeterminate boundaries of geographic objects [15], a scientific strand which goes back to a tradition of research on spatial

representations and operations in Geographic Information Systems [17] and on integrity constraints for spatial databases.

Another strand is interested in the retrieval, re-usage, integration, and interoperation of geo-referenced information. How can geographic referents be semantically linked to other kinds of information with related meaning? Due to the broad thematic coverage of geo-data spanning fields such as human and cognitive Geography, Transportation Research, Economics, Ecology, Climatology, Geology, and Oceanography, data integration and sharing require methods that reduce the risk of semantic incompatibilities [39]. Recent work has developed technology for enabling *spatial queries* on Linked Data. This work includes GeoSPARQL as a common query language for the Geospatial Semantic Web as well as RDF triple stores which can effectively store and index Linked Spatiotemporal Data [8]. Similar work addressed the role of semantic similarity for spatial scene queries [76,64]. Current research challenges investigate how geo-data can be represented on different *levels of abstraction, scale, and granularity* [28], and how to semantically account for its *uncertainty* [11]. Another challenge is that geospatial referents, such as places and events, are often only implicitly contained in a data set, and thus need to be *automatically discovered* in data repositories which are not yet linked or geo-referenced.

In this section we gave an overview of the field of geo-semantics and current research challenges, and highlighted two main threads of geo-semantics research: 1) representing geographic data models in ontologies and 2) semantically linking geographic data with information coming from other domains.

4 Research Questions and Major Findings

In the following, we will use seven research questions to introduce major areas of research and discuss findings used to provide reliable geospatial referents for the Semantic Web.

What Kinds of Geospatial Classes Should Be Distinguished?

Even though geographic referents are rooted in diverse domains, they share certain common characteristics and principles that can be exploited in the design of geo-ontologies. Kuhn [61] proposed the core concepts *location, neighborhood, field, object, network, event* as well as the information concepts of *granularity* and *accuracy* as a common core that can be used to *spatialize* information. Geo-ontologies need to support access to phenomena on flexible resolution levels and scales in order to allow systems to query and reason on scale dependent representations [7]. For this purpose, scale dependency of representations needs to be formally expressed, as recently shown by Carral Martínez et al. [16]. Geo-ontologies also have to deal with the various natures of spatial boundaries, as distinguished in [90]. Examples for top-level geo-ontologies that incorporate the principle of spatial granularity include the work of Bittner et al. [12]. Usually, such foundational ontologies are extended by domain ontologies, such as the SWEET ontology for Earth and environmental science [80].

As we argued in Section 2, geospatial concepts are situated and context-dependent [13] and can be described from different, equally valid points of view [47,86]. This makes standard comprehensive approaches towards ontology engineering unrealistic. Semantic engineering, however, can be slightly redefined, namely as a method of communicating possible interpretations of terms by constraining them towards the intended ones [60,47], without prescribing a huge amount of abstract *ontological commitments*. Ontology design patterns can provide reusable building blocks or strategies to support knowledge engineers and scholars in defining local, data-centric, and purpose-driven ontologies [32]. Vague terms may be grounded multiple interpretations [10]. Ontologies may also be built up in a layered fashion Frank2003,Couclelis.2010. In such cases, one can start with observation procedures on the bottom level and then arrive at more abstract but reproducible ontological categories by deductive and inductive methods [2,47,21].

How to Refer to Geospatial Phenomena?

Geographic information technology relies, to large extent, on the availability of *reference systems* for the precise semantic interpretation of its spatial, temporal, and thematic components [17]. Spatial reference systems provide the formal vocabulary to calculate with precise locations, e.g., in the form of coordinates on a mathematical ellipsoid, and to perform a multitude of operations such as distance measurement. Geodetic datums, i.e., standard directions and positions of the ellipsoid, enable the interpretation of locations as results of repeatable measurements on the earth's surface. Both are required to understand spatial data. Temporal reference systems, e.g., calendars, manage the representation of time, and allow one to translate between different calendars. The thematic (sometimes also called attributive) component of geo-information requires reference systems as well [17,58]. Examples are measurement scales for qualities such as temperature or air pressure. As a consequence, Kuhn introduced the generalized notion of Semantic Reference Systems (SRS) [58]. They are supposed to enable a precise interpretation of all components of geospatial data in terms of semantic datums, which provide for their grounding in terms of measurement scales or observation procedures [86]. For example, attribute values such as the wind directions discussed before can be interpreted in terms of reference systems for cardinal wind directions and anemometers. Establishing these SRS, their standard operations, as well as their formal vocabularies, is an ongoing research area [60]. It has been mentioned among the most pressing and challenging projects of GIScience/Geoinformatics [70]. Recent research results include methodologies and formalisms for grounding reference systems [86,79], as well as technologies for translation of attribute values based on reference systems [84].

How to Perform Geo-reasoning and Querying over the Semantic Web?

There are several research traditions of geospatial reasoning, ranging from more computational to more conceptual approaches. One is based on *geometric*

operators in a spatial database, i.e., on *explicitly represented spatial geometry*. These include point-in-polygon tests, R-tree search algorithms, quadtree compression, and geometric and set-theoretic operators for vector data. Another form is based on graph-based computational methods, which, for instance, allow reasoning about road networks [17]. *Spatial reasoning*, i.e., reasoning with qualitative spatial relations, includes topological reasoning, such as about *overlap*, *meet*, and *disjoint* relations, and reasoning with directions. Prominent spatial calculi are mereotopological calculi, Frank's cardinal direction calculus, Freksa's double cross calculus, Egenhofer and Franzosa's 4- and 9-intersection calculi, Ligozat's flip-flop calculus, Cohn's region connection calculi (RCC), and the Oriented Point Relation Algebra [82]. The latter kind of reasoning is based on deductive inference in first-order predicate logic [19], as well as on finite composition tables and constraint reasoning, in which all possible relations are enumerated exhaustively [82].

In comparison, most Semantic Web reasoning is rather narrowly defined. It is concerned with particular decidable subsets of first-order predicate logic, namely description logics and Horn rules, which often lack the expressivity needed to reason with spatial relations [92]. Furthermore, other forms of geospatial reasoning, such as geometrical computation or approximate reasoning, are only rudimentary supported by the Semantic Web [41]. The integration of such reasoning paradigms into the Semantic Web requires further consideration of their RDF representation and computability, as well as a broadening of the existing reasoning paradigm itself. It has been argued that in the past, the reasoning paradigm of the Semantic Web might have been too narrowly occupied by soundness, completeness, and decidability constraints. Thus, in the context of geo-semantics, it might be useful to loosen soundness and completeness demands of proof procedures in order to allow for scalable approximate reasoning [41].

How can geospatial reasoning be integrated with Semantic Web technologies in a tractable way? Many spatial qualitative decision problems are NP-hard, however, tractable subsets can be identified [82]. There are several recent efforts to integrate qualitative spatial reasoning into RDF reasoners, such as Racer [97] and Pellet [92]. A promising direction of research is to combine qualitative reasoners with geometric computation. In the Semantic Web, this may be realized in terms of spatial extensions to RDF and SPARQL, such as stSPARQL or GeoSPARQL [57,8].

How to Discover Events and How to Account for Geographic Change?

Geographic assertions, such as partonomic relations between administrative regions, trajectories of moving objects and their relations to the places they cross, and membership in organizations, are valid only over a certain period of time [53]. Consequently, research investigates how this temporal dimension can be accounted for. There is a multitude of research on spatiotemporal modeling, temporal GIS [18], and simple temporal gazetteer models [40]. There is also related ontological work on the formal relationship between objects, processes,

and events [31]. Research also addressed event ontology design patterns [36]. However, a particular challenge remains the automated detection of events from observation data on a geographic scale [9], such as blizzards, rainstorms, or floods [23]. Examples of research on the detection of geographic event as well as identification algorithms include the work by Agouris and Stefanidis [3]. Nevertheless, many questions regarding general formal and computational procedures of geographic event detection remain unsolved. This is especially the case for work concerning the tight coupling of geospatial ontologies with detected events, as well as the triggering of data and ontology updates by automatically detected events [65].

How to Handle Places and Moving Object Trajectories?

Humans communicate using *places* in order to refer to space. These references to places go well beyond geographic coordinates. Locations as simple coordinates are point-like, ubiquitous, and precise. Contrarily, places are not point-like and have fuzzy boundaries determined by physical, cultural, and cognitive processes [94,74]. Additionally, places, e.g., *downtown*, can change their locations over time, just like physical objects do [53]. Therefore, mere positioning data insufficiently captures the identity and meaning of places.

GIScience and geo-semantics has generated useful results in handling places in a number of different ways, for example, by the specification of place data models [40] and place ontologies [1], which can be used to improve geographic information retrieval [68,50,45]. A interesting direction for future work along these lines are affordance-based approaches toward place [51] as they allow one to associate places with the activities that can be performed at them. Another important development are technologies that can be used to handle place by automated discovery, in order to enrich data with geo-references. A typical direction of work is *geoparsing*, the discovery of places in texts by natural language processing techniques. Such research can also be applied which to identify place-related activities [5]. Recently, research has started to address the discovery of places and user activities by mining (semantic) trajectories [98]. Researchers also investigated how to reconstruct spatial footprints of places based on *geotags* in social media, e.g., Flickr [43]. Future research may address the design of place-based information systems [34], in which traditional operations of GIS have to be redesigned to handle places as referents. Geo-ontologies are considered a central part of this vision.

How to Compare, Align, and Translate Geospatial Classes?

Comparing geographic feature types across heterogeneous resources requires methods to compute conceptual similarity. Geo-semantics provides unique approaches to do this based on reference systems or conceptual spaces. Thus, over the past years, semantic translation [39,59,24,77], semantic similarity measurement [83,81,64,88,76,45], and geo-ontology alignment [22] have been major research directions. While semantic translation maps between geo-ontologies and

Table 2. Seven exemplary research questions in the field of geo-semantics

What Kinds of Geospatial Classes Should be Distinguished?
How to Refer to Geospatial Phenomena?
How to Perform Geo-Reasoning & Querying over the Semantic Web?
How to Discover Events and how to Account for Geographic Change?
How to Handle Places and Moving Object Trajectories?
How to Compare, Align, and Translate Geospatial Classes?
How to Process, Publish and Retrieve Geodata?

can be thought of as the analogy to datum transformation, similarity measures the proximity between classes in a semantic space as an analogy to distance in space (and time). Geo-ontology alignment is concerned with the combination of multiple ontologies to foster data reuse and integration. The fact that most types of geographic information analysis, e.g. kernel methods, interpolation, or point pattern analysis, are based on spatial auto-correlation and distance in space, shows why semantic similarity is regarded crucial for making ontologies and geo-semantics first class citizens of geographic information systems. The notion of similarity also plays a key role in many cognitive approaches. Semantic similarity and analogy reasoning also enable new types of interaction paradigms and user interfaces which may ease browsing and navigating through (unfamiliar) geo-data and ontologies [45]. On the downside, similarity is highly sensitive to context. Therefore, researchers have analyzed the influence of contextual information and proposed different techniques to account for such effects [55].

How to Process, Publish and Retrieve Geodata?

Standardized means for publishing, querying, retrieving, and accessing geodata via Web services are provided by Spatial Data Infrastructures (SDI) as part of the framework developed by the Open Geospatial Consortium (OGC). These SDIs also support a variety of notification and processing services and, thereby, go beyond simple data stores. Data and processing services can be combined to model complex scientific workflows and be integrated as core elements in cyberinfrastructures. To ensure a meaningful combination of services, however, relies on formal specifications of the service inputs, outputs, side effects, and parameters. Therefore, semantic markups for Web services have been actively researched for years [71,27,93]. Examples of SDI specific proposals include the work of Lemmens et al. [63], Vaccari et al. [96], and Lutz [67].

SDI services use their own markup languages (e.g., the Geographic Markup language GML) and protocols, which differ considerably from the Semantic Web technology stack. This prevents interoperability and makes a combination of the Semantic Web and the Geo-Web challenging. Consequently, researchers have proposed and implemented different approaches for a semantic enablement of the Geo-Web. Janowicz et al. [46], for instance, specified transparent and bi-directional proxies which enable users of both infrastructures to share data and

combine services. Semantic annotations have been proposed to lift existing geo-data to a semantic level [56,72]. In context of digital humanities research, annotations have been applied to create Linked Spatiotemporal Data, e.g., to enrich old maps with interlinked information from the global graph [89]. With respect to OGC's family of Sensor Web Enablement standards (SWE), researchers have developed sensor and observation ontologies, semantically-enabled versions of OGC services such as the Sensor Observation Service (SOS), or RESTful transparent proxies that serve Linked Sensor Data [20].

In this section, we identified and described seven important research questions being asked in the field of geo-semantics (listed in Table 2).

5 Conclusion

We have argued that the relevance of geospatial information lies in the usefulness of geospatial referents, such as places, events, and geographic objects, and their spatiotemporal relations, which allow systems to indirectly localize and interlink numerous other resources in the Semantic Web. In part, this importance of geospatial information is reflected by the fact that the few already existing geo-data repositories, e.g., Geonames, have become central hubs on the Web of Linked Data. In addition, other key repositories, such as DBpedia, Freebase, and so forth, contain substantial collections of geo-data. However, we have also illustrated that even though geospatial information has reference systems, which allow one to precisely map and index the extent of geospatial referents, conceptualizing and formalizing these referents is an unsolved challenge. This challenge is mainly due to the situated and multi-perspectival nature of geospatial phenomena. It calls for semantic strategies that allow highlighting, distinguishing, and linking of the different perspectives to the localities that are inherent in geo-data.

Geospatial semantics addresses this need with semantic modeling of geospatial classes as well as semantic technology for access, comparison, and interlinking of geo-data. Specific semantic modeling challenges include the notions of resolution and scale in geo-ontologies, ontological perspectivity, semantic reference systems, place reference, trajectories, event discovery, the formalization of spatial relations, and the computation of spatial reasoning. Semantic technology for access and retrieval include semantically enabled spatial data infrastructures and Linked Spatiotemporal Data, as well as cognitively plausible similarity measures, analogy-based reasoning, and translation tools for geo-ontologies.

References

1. Abdelmoty, A.I., Smart, P., Jones, C.B.: Building place ontologies for the semantic web: Issues and approaches. In: Proceedings of the 4th ACM Workshop on Geographical Information Retrieval, GIR 2007, pp. 7–12. ACM, New York (2007)

2. Adams, B., Janowicz, K.: Constructing geo-ontologies by reification of observation data. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2011, pp. 309–318. ACM, New York (2011)
3. Agouris, P., Stefanidis, A.: Efficient summarization of spatiotemporal events. *Commun. ACM* 46(1), 65–66 (2003)
4. Ahlqvist, O., Shortridge, A.: Characterizing land cover structure with semantic variograms. In: Riedl, A., Kainz, W., Elmes, G. (eds.) *Progress in Spatial Data Handling -12th International Symposium on Spatial Data Handling*, pp. 401–415 (Springer 2006)
5. Alazzawi, A., Abdelmoty, A., Jones, C.: What can I do there? Towards the automatic discovery of place-related services and activities. *International Journal of Geographical Information Science* 26(2), 345–364 (2012)
6. Barsalou, L.: Situated simulation in the human conceptual system. *Language and Cognitive Processes* 5(6), 513–562 (2003)
7. Bateman, J.: Towards a generic foundation for spatial ontology. In: Varzi, A., Vieu, L. (eds.) *Proceedings of the 3rd International Conference on Formal Ontology in Information Systems (FOIS 2004)*, pp. 237–248. IOS Press, Amsterdam (2004)
8. Battle, R., Kolas, D.: Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL. *Semantic Web Journal* 3(4) (2012)
9. Beard, K.: Modeling change in space and time: An event based approach. In: Billen, R., Joao, E., Forrest, D. (eds.) *Dynamic and Mobile GIS: Investigating Changes in Space and Time*, pp. 55–74. CRC Press (2007)
10. Bennett, B., Mallenby, D., Third, A.: An ontology for grounding vague geographic terms. In: Eschenbach, C., Gruninger, M. (eds.) *Proc. 5th Intern. Conf. on Formal Ontology in Information Systems*, pp. 280–293. IOS-Press [u.a.], Saarbrücken (2008)
11. Bennett, B.: Spatial vagueness. In: Jeansoulin, R., Papini, O., Prade, H., Schockaert, S. (eds.) *Methods for Handling Imperfect Spatial Information. STUDFUZZ*, vol. 256, pp. 15–47. Springer, Heidelberg (2010)
12. Bittner, T., Donnelly, M., Smith, B.: A spatio-temporal ontology for geographic information integration. *International Journal of Geographical Information Science* 23(6), 765–798 (2009)
13. Brodaric, B., Gahegan, M.: Experiments to examine the situated nature of geoscientific concepts. *Spatial Cognition & Computation* 7(1), 61–95 (2007)
14. Buckley, L.B., Jetz, W.: Linking global turnover of species and environments. *Proceedings of the National Academy of Sciences* 105(46), 17836–17841 (2008)
15. Burrough, P., Frank, A.U. (eds.): *Geographic Objects with Indeterminate Boundaries*. Taylor and Francis (1996)
16. Carral, D., Scheider, S., Janowicz, K., Vardeman, C., Krisnadhi, A.A., Hitzler, P.: An ontology design pattern for cartographic map scaling. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) *ESWC 2013. LNCS*, vol. 7882, pp. 76–93. Springer, Heidelberg (2013)
17. Chrisman, N.: *Exploring geographic information systems*. Wiley (2001)
18. Christakos, G., Bogaert, P., Serre, M.: *Temporal GIS: Advanced Functions for Field-Based Applications*. Springer (2002)
19. Cohn, A.G., Hazarika, S.M.: Qualitative spatial representation and reasoning: an overview. *Fundamenta Informaticae* 46, 1–29 (2001)

20. Compton, M., Barnaghi, P.M., Bermudez, L., Garcia-Castro, R., Corcho, Ó., Cox, S., Graybeal, J., Hauswirth, M., Henson, C.A., Herzog, A., Huang, V.A., Janowicz, K., Kelsey, W.D., Phuoc, D.L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K.R., Passant, A., Sheth, A.P., Taylor, K.: The ssn ontology of the w3c semantic sensor network incubator group. *Journal of Web Semantics* 17, 25–32 (2012)
21. Couclelis, H.: Ontologies of geographic information. *International Journal of Geographical Information Science* 24(12), 1785–1809 (2010)
22. Cruz, I., Sunna, W.: Structural alignment methods with applications to geospatial ontologies. *Transactions in GIS, special issue on Semantic Similarity Measurement and Geospatial Applications* 12(6), 683–711 (2008)
23. Devaraju, A.: Representing and reasoning about geographic occurrences in the sensor web. Ph.D. thesis, University of Münster (2012)
24. Dou, D., McDermott, D.V., Qi, P.: Ontology translation on the semantic web. *Journal of Data Semantics* 2, 35–57 (2005)
25. Egenhofer, M.J., Mark, D.M.: Naive geography. In: Kuhn, W., Frank, A.U. (eds.) *COSIT 1995. LNCS*, vol. 988, pp. 1–15. Springer, Heidelberg (1995)
26. Elands, B., Wiersum, K.F.: Forestry and rural development in europe: an exploration of socio-political discourses. *Forest Policy and Economics* 3(1), 5–16 (2001)
27. Fensel, D., Bussler, C.: The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications* 1(2), 113–137 (2002)
28. Fonseca, F., Egenhofer, M., Davis, C., Câmara, G.: Semantic granularity in ontology-driven geographic information systems. *Annals of Mathematics and Artificial Intelligence* 36(1-2), 121–151 (2002)
29. Fox, P., McGuinness, D., Raskin, R., Sinha, K.: A volcano erupts: semantically mediated integration of heterogeneous volcanic and atmospheric data. In: *Proceedings of the ACM First Workshop on CyberInfrastructure: Information Management in eScience, CIMS 2007*, pp. 1–6. ACM (2007)
30. Frank, A.U.: Chapter 2: Ontology for spatio-temporal databases. In: Sellis, T.K., et al. (eds.) *Spatio-Temporal Databases. LNCS*, vol. 2520, pp. 9–77. Springer, Heidelberg (2003)
31. Galton, A., Mizoguchi, R.: The water falls but the waterfall does not fall: New perspectives on objects, processes and events. *Applied Ontology* 4(2), 71–107 (2009)
32. Gangemi, A., Presutti, V.: Towards a pattern science for the semantic web. *Semantic Web* 1(1-2), 61–68 (2010)
33. Goodchild, M.: Citizens as sensors: the world of volunteered geography. *GeoJournal* 69(4), 211–221 (2007)
34. Goodchild, M.: Formalizing place in geographic information systems. In: Burton, L., et al. (eds.) *Communities, Neighborhoods, and Health: Expanding the Boundaries of Place*, vol. 1, pp. 21–34. Springer, Berlin (2011)
35. Goodwin, J., Dolbear, C., Hart, G.: Geographical linked data: The administrative geography of great britain on the semantic web. *Transactions in GIS* 12, 19–30 (2008)
36. van Hage, W., Malaise, V., Segers, R., Hollink, L., Schreiber, G.: Design and use of the Simple Event Model (SEM). *Web Semantics: Science, Services and Agents on the World Wide Web* 9(2) (2011)
37. Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: When owl: sameas isn't the same: An analysis of identity in linked data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part I. LNCS*, vol. 6496, pp. 305–320. Springer, Heidelberg (2010)

38. Halpin, H., Presutti, V.: The identity of resources on the web: An ontology for web architecture. *Applied Ontology* 6(3), 263–293 (2011)
39. Harvey, F., Kuhn, W., Pundt, H., Bishr, Y., Riedemann, C.: Semantic interoperability: A central issue for sharing geographic information. *The Annals of Regional Science* 2(33), 213–232 (1999)
40. Hill, L.: Georeferencing. *The geographic associations of information*. MIT Press, Cambridge (2006)
41. Hitzler, P., van Harmelen, F.: A reasonable semantic web. *Semantic Web* 1(1-2), 39–44 (2010)
42. Hitzler, P., Krötzsch, M., Rudolph, S.: *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC (2009)
43. Hollenstein, L., Purves, R.: Exploring place through user-generated content: Using flickr tags to describe city cores. *J. Spatial Information Science*, 21–48 (2010)
44. Janowicz, K., Hitzler, P.: The digital earth as knowledge engine. *Semantic Web Journal* 3(3) (2012)
45. Janowicz, K., Raubal, M., Kuhn, W.: The semantics of similarity in geographic information retrieval. *Journal of Spatial Information Science* (2), 29–57 (2011)
46. Janowicz, K., Schade, S., Bröring, A., Keßler, C., Maue, P., Stasch, C.: Semantic enablement for spatial data infrastructures. *Transactions in GIS* 14(2), 111–129 (2010)
47. Janowicz, K.: Observation-driven geo-ontology engineering. *Transaction in GIS* 16(3), 351–374 (2012)
48. Janowicz, K., Scheider, S., Pehle, T., Hart, G.: Geospatial semantics and linked spatiotemporal data - past, present, and future. *Semantic Web* 3(4), 321–332 (2012)
49. Jones, C.B., Abdelmoty, A.I., Finch, D., Fu, G., Vaid, S.: The spirit spatial search engine: Architecture, ontologies and spatial indexing. In: Egenhofer, M., Freksa, C., Miller, H.J. (eds.) *GIScience 2004*. LNCS, vol. 3234, pp. 125–139. Springer, Heidelberg (2004)
50. Jones, C.B., Alani, H., Tudhope, D.: Geographical information retrieval with ontologies of place. In: Montello, D.R. (ed.) *COSIT 2001*. LNCS, vol. 2205, pp. 322–335. Springer, Heidelberg (2001)
51. Jordan, T., Raubal, M., Gartrell, B., Egenhofer, M.J.: An Affordance-Based Model of Place in GIS. In: Poiker, T., Chrisman, N. (eds.) *8th Int. Symposium on Spatial Data Handling (SDH 1998)*, pp. 98–109. IUG, Vancouver (1998)
52. Kauppinen, T., de Espindola, G.M., Jones, J., Sánchez, A., Gräler, B., Bartoschek, T.: Linked brazilian amazon rainforest data. *Semantic Web Journal* (forthcoming, 2013)
53. Kauppinen, T., Hyvönen, E.: *Modeling and Reasoning about Changes in Ontology Time Series*, pp. 319–338. Springer, New York (2007)
54. Kavouras, M., Kokla, M.: *Theories of Geographic Concepts: Ontological Approaches to Semantic Integration*. Taylor & Francis (2007)
55. Keßler, C.: What is the difference? A cognitive dissimilarity measure for information retrieval result sets. *Knowl. Inf. Syst.* 30(2), 319–340 (2012)
56. Klien, E.: A rule-based strategy for the semantic annotation of geodata. *Transactions in GIS* 11(3), 437–452 (2007)
57. Koubarakis, M., Kyzirakos, K., Karpathiotakis, M., Nikolaou, C., Sioutis, M., Vassos, S., Michail, D., Herekakis, T., Kontoes, C., Papoutsis, I.: Challenges for Qualitative Spatial Reasoning in Linked Geospatial Data. In: *IJCAI 2011 Workshop on Benchmarks and Applications of Spatial Reasoning (BASR 2011)*, pp. 33–38 (2011)

58. Kuhn, W.: Semantic reference systems (guest editorial). *International Journal of Geographical Information Science* 17(5), 405–409 (2003)
59. Kuhn, W.: Geospatial semantics: Why, of what, and how? In: Spaccapietra, S., Zimányi, E. (eds.) *Journal on Data Semantics III*. LNCS, vol. 3534, pp. 1–24. Springer, Heidelberg (2005)
60. Kuhn, W.: Semantic engineering. In: Navratil, G. (ed.) *Research Trends in Geographic Information Science*. Lecture Notes in Geoinformation and Cartography, vol. 12, pp. 63–76. Springer, Berlin (2009)
61. Kuhn, W.: Core concepts of spatial information for transdisciplinary research. *International Journal of Geographical Information Science* 26(12), 2267–2276 (2012)
62. Kuhn, W.: Modeling vs encoding for the semantic web. *Semantic Web Journal* 1(1-2) (2010)
63. Lemmens, R., Wytzisk, A., de By, R., Granell, C., Gould, M., van Oosterom, P.: Integrating semantic and syntactic descriptions to chain geographic services. *IEEE Internet Computing* 10(5), 42–52 (2006)
64. Li, B., Fonseca, F.: TDD - A Comprehensive Model for Qualitative Spatial Similarity Assessment. *Spatial Cognition and Computation* 6(1), 31–62 (2006)
65. Llaves, A., Michels, H., Maue, P., Roth, M.: Semantic event processing in envision. In: *Proceedings of the International Conference on Web Intelligence, Mining and Semantics (WIMS 2012)*, pp. 29:1–29:2. ACM, New York (2012) (forthcoming)
66. Lund, G.: Definitions of forest, deforestation, afforestation, and reforestation. Tech. rep. (2012), gainesville, va: Forest information services, available from the world wide web: <http://home.comcast.net/~gyde/DEFpaper.htm>
67. Lutz, M.: Ontology-based descriptions for semantic discovery and composition of geoprocessing services. *GeoInformatica* 11, 1–36 (2007)
68. Lutz, M., Klien, E.: Ontology-based retrieval of geographic information. *International Journal of Geographical Information Science* 20(3), 233–260 (2006)
69. Mäkelä, E., Hyvönen, E., Ruotsalo, T.: How to deal with massively heterogeneous cultural heritage data - lessons learned in culturesampo. *Semantic Web* 3(1), 85–109 (2012)
70. Mark, D.M., Smith, B., Egenhofer, M., Stephen Hirtle, S.C.: Ontological foundations for geographic information science. In: McMaster, R., Uery, L. (eds.) *Research Challenges in Geographic Information Science*, pp. 335–350. CRC Press (2004)
71. Martin, D., Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: Owl-s: Semantic markup for web services (November 2004), <http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/>
72. Maue, P., Michels, H., Roth, M.: Injecting semantic annotations into (geospatial) web service descriptions. *Semantic Web Journal* 3(4) (2012)
73. Mitchell, S.A.: Heights of the principal mountains in the world (1846), <http://www.davidrumsey.com>
74. Montello, D., Goodchild, M., Gottsegen, J., Fohl, P.: Where’s downtown? Behavioral methods for determining referents of vague spatial queries. *Spatial Cognition & Computation* 2(3), 185–204 (2003)
75. NASA: A.40 computational modeling algorithms and cyberinfrastructure (December 19, 2011). Tech. rep., National Aeronautics and Space Administration (NASA) (2012)
76. Nedas, K., Egenhofer, M.: Spatial-scene similarity queries. *Transactions in GIS* 12(6), 661–681 (2008)

77. Noy, N.: Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec.* 33, 65–70 (2004)
78. Probst, F., Lutz, M.: Giving Meaning to GI Web Service Descriptions. In: *International Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI 2004)* (2004)
79. Probst, F.: Observations, measurements and semantic reference spaces. *Appl. Ontol.* 3(1-2), 63–89 (2008)
80. Raskin, R., Pan, M.: Knowledge representation in the semantic web for earth and environmental terminology (sweet). *Computers & Geosciences* 31(9), 1119–1125 (2005)
81. Raubal, M.: Formalizing conceptual spaces. In: Varzi, A., Vieu, L. (eds.) *Proceedings of the Third International Conference on Formal Ontology in Information Systems (FOIS 2004)*. *Frontiers in Artificial Intelligence and Applications*, vol. 114, pp. 153–164. IOS Press, Torino (2004)
82. Renz, J., Nebel, B.: Qualitative spatial reasoning using constraint calculi. In: Aiello, M., Pratt-Hartmann, I., van Benthem, J. (eds.) *Handbook of Spatial Logics*, pp. 161–215. Springer (2007)
83. Rodríguez, A., Egenhofer, M.: Comparing geospatial entity classes: an asymmetric and context-dependent similarity measure. *International Journal of Geographical Information Science* 18(3), 229–256 (2004)
84. Schade, S.: *Ontology-driven translation of geospatial data*. Ph.D. thesis (2010)
85. Scheider, S., Kuhn, W.: Affordance-based categorization of road network data using a grounded theory of channel networks. *International Journal of Geographical Information Science* 24(8), 1249–1267 (2010)
86. Scheider, S.: *Grounding Geographic Information in Perceptual Operations*. *Frontiers in Artificial Intelligence and Applications*, vol. 244. IOS Press (2012)
87. Schlieder, C.: Digital heritage: Semantic challenges of long-term preservation. *Semantic Web* 1(1-2), 143–147 (2010)
88. Schwering, A., Raubal, M.: Measuring semantic similarity between geospatial conceptual regions. In: Rodríguez, M.A., Cruz, I., Levashkin, S., Egenhofer, M.J. (eds.) *GeoS 2005*. LNCS, vol. 3799, pp. 90–106. Springer, Heidelberg (2005)
89. Simon, R., Sadilek, C., Korb, J., Baldauf, M., Haslhofer, B.: Tag clouds and old maps: Annotations as linked spatiotemporal data in the cultural heritage domain. In: *Workshop On Linked Spatiotemporal Data 2010*, held in conjunction with the 6th International Conference on Geographic Information Science (GIScience 2010), vol. 691. CEUR-WS, Zurich, Switzerland (2010)
90. Smith, B., Mark, D.M.: *Ontology and geographic kinds*. In: *Proceedings International Symposium on Spatial Data Handling*, Vancouver, Canada, July 12–15 (1998)
91. Stadler, C., Lehmann, J., Höffner, K.: *LinkedGeoData: A core for a web of spatial open data*
92. Stocker, M., Sirin, E.: PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine. In: *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED)* (2009)
93. Szekely, P., Knoblock, C.A., Gupta, S., Taheriyan, M., Wu, B.: Exploiting semantics of web services for geospatial data fusion. In: *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Spatial Semantics and Ontologies, SSO 2011*, pp. 32–39. ACM (2011)
94. Tuan, Y.: *Space and place: humanistic perspective*. Theory and decision library. D. Reidel Pub. Co. (1979)

95. Usery, E.L., Varanka, D.: Design and development of linked data from the national map. *Semantic Web Journal* 3(4) (2012)
96. Vaccari, L., Shvaiko, P., Marchese, M.: A geo-service semantic integration in Spatial Data Infrastructures. *International Journal of Spatial Data Infrastructures Research* 4 (2009), <http://ijmdir.jrc.ec.europa.eu/index.php/ijmdir/article/viewFile/107/113>
97. Wessel, M., Möller, R.: Flexible software architectures for ontology-based information systems. *Journal of Applied Logic* 7(1), 75–99 (2009)
98. Ying, J.J.C., Lee, W.C., Weng, T.C., Tseng, V.S.: Semantic trajectory mining for location prediction. In: Cruz, I.F., Agrawal, D., Jensen, C.S., Ofek, E., Tanin, E. (eds.) *Proceedings of the 19th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2011*, pp. 34–43. ACM (2011)

Statistical Relational Data Integration for Information Extraction*

Mathias Niepert

Department of Computer Science & Engineering
University of Washington, Seattle, WA, USA
`mniepert@cs.washington.edu`

Abstract. These lecture notes provide a brief overview of some state of the art large scale information extraction projects. Consequently, these projects are related to current research activities in the semantic web community. The majority of the learning algorithms developed for these information extraction projects are based on the lexical and syntactical processing of Wikipedia and large web corpora. Due to the size of the processed data and the resulting intractability of the associated inference problems existing knowledge representation formalism are often inadequate for the task. We will present recent advances in combining tractable logical and probabilistic models that bring statistical language processing and rule-based approaches closer together. With these lecture notes we hope to convince the attendees that there are numerous synergies and research agendas that can arise when uncertainty-based data-driven research meets rule-based schema-driven research. We also describe certain theoretical and practical advances in making probabilistic inference scale to very large problems.

1 Introduction

Historically, semantic web research has focused on problems concerned with the logical form of the schema, that is, the meta-level descriptions of classes and roles that comprise the structure of the knowledge base. It comes at no surprise, therefore, that the highly popular research areas of ontology learning, ontology matching, and knowledge engineering have mostly concentrated on the terminological structure, that is, the set of axioms involving class and role descriptions. While meaningful progress has been made and the logical, computational, and empirical understanding of these problems is deeper than ever before, this has come at the cost of largely ignoring the *data*, that is, *assertions* of the aforementioned classes and roles. Instead of building knowledge representations around existing data, more often than not, ontologies were designed and constructed in a data vacuum. It is probably not far-fetched to assume that this is the main reason for the skepticism (if not outright rejection) other research communities have demonstrated towards the semantic web endeavor.

* These lecture notes are based on several previous publications of the author and his colleagues in conference proceedings such as AAAI, UAI, IJCAI, and ESWC.

This skepticism manifests itself in a recent surge of information extraction projects such as the open information extraction [23] (OIE) and the never ending language learning [13] (NELL) projects. Indeed, the OIE project *explicitly* defines itself as *open*, meaning that it does not leverage ontologies or relational schemas. The major argument supporting this position is that a relational schema or ontology unnecessarily constrains what can be extracted from large web corpora. The NELL project leverages a type system and a fixed set of relations, even though recent work has moved towards (semi-)automatically extending the set of relations. However, insight and expertise accumulated in the semantic web community over the last 10 years is largely ignored. For instance, the project does not employ canonical labels for its entities ('Argentina' refers to both, the national soccer team and the country itself) and makes no use of existing knowledge representation formalisms even though it actually uses notions such as range and domain restrictions implicitly. While this could be explained with the specific applications the creators have in mind (improved keyword search and natural language question answering, for instance) there are some reasonable arguments in favor of not completely ignoring the existing body of work and experience of the semantic web community. Other information extraction projects such as DBpedia [4,59] and YAGO [87,36] are more in line with semantic web technologies as they use unique canonical identifiers for entities (derived from the URIs of the corresponding Wikipedia articles) and notions such as range and domain restrictions that closely resemble the RDF standard. The advantage of using these standardized RDF formalisms is that they enable the creation of links across heterogeneous data sets and a unifying syntactic and semantic framework for knowledge bases. DBpedia, for instance, has established itself as a linking hub for the linked open data cloud. The existence of a relational schema or ontology also facilitates relational query processing and the use of statistical relational approaches such as Markov logic [80].

The present lecture notes provide a brief overview of existing information extraction projects ranging from those with a predetermined ontology, that is, a relational schema, high precision extractions, and limited coverage, to those without any kind of schema, low precision extractions, and broader coverage. We do not take sides and instead focus on possible synergies that arise when we consider each of the projects as disparate and heterogeneous knowledge bases whose integration would not only broaden the amount of extracted knowledge but also increase the extraction quality and provide relational schemas for facts that were previously schema-less. We provide an overview of the problem areas ontology matching and object reconciliation from a semantic web perspective. We then show how both the relational schema *and* the data can be jointly modeled with statistical relational formalisms.

Ontology matching, or ontology alignment, is the problem of determining correspondences between concepts, properties, and individuals of two or more different formal ontologies [26]. The alignment of ontologies allows semantic applications to exchange and enrich the data expressed in the respective ontologies. An important results of the yearly ontology alignment evaluation initiative

(OAEI) [25,27] is that there is no single best approach to all existing matching problems. The factors influencing the quality of alignments range from differences in lexical similarity measures to variations in alignment extraction approaches. This insight provides justification not only for the OAEI itself but also for the development of a framework that facilitates the comparison of different strategies with a flexible and declarative formalism. We argue that Markov logic [80] provides an excellent framework for ontology matching. Markov logic (ML) offers several advantages over existing matching approaches. Its main strength is rooted in the ability to combine *soft* and *hard* first-order formulas. This allows the inclusion of both *known* logical and *uncertain* statements modeling potential correspondences and structural properties of the ontologies. For instance, hard formulas can help to reduce incoherence during the alignment process while *soft* formulas can factor in lexical similarity values computed for each correspondence. An additional advantage of ML is joint inference, that is, the inference of two or more interdependent hidden predicates. Several results show that joint inference is superior in accuracy when applied to a wide range of problems such as ontology refinement [92] and multilingual semantic role labeling [60].

Identifying different representations of the same data item is called object reconciliation. The problem of object reconciliation has been a topic of research for more than 50 years. It is also known as record linkage [29], entity resolution [7], and instance matching [30]. While the majority of the existing methods were developed for the task of matching database records, modern approaches focus mostly on graph-based data representations such as the resource description framework (RDF). Using the proposed Markov logic based framework for data integration, we leverage schema information to exclude logically inconsistent correspondences between objects improving the overall accuracy of instance alignments. In particular, we use logical reasoning and linear optimization techniques to compute the overlap of derivable types of objects. This information is combined with the classical similarity-based approach, resulting in a novel approach to object reconciliation that is more flexible than state-of-the-art alignment systems. We demonstrate how description logic axioms are modeled within the framework and show that alignment problems can be posed as linear optimization problems. These problems can be efficiently solved with integer linear programming methods also leveraging recent meta-algorithms such as cutting plane inference [81] and delayed column generation [64] first proposed in the context of Markov logic.

The chapter is organized as follows. First, we briefly introduce some basic formalism such as description logics and Markov logic. Second, we define ontology matching and object reconciliation and introduce detailed running examples that we use throughout the chapter to facilitate a deeper understanding of the ideas and methods. We also introduce the syntax and semantics of the ML framework and show that it can represent numerous different matching scenarios. We describe probabilistic reasoning in the framework of Markov logic and show that a solution to a given matching problem can be obtained by solving the maximum a-posteriori (MAP) problem of a ground Markov logic network using

integer linear programming. Finally, we discuss some recent advances in the development of efficient algorithms for probabilistic inference.

2 Information Extraction – The State of the Art

There are numerous information extraction projects each with foci on particular subproblems of information extraction and knowledge base construction. We selected several representative projects without making a claim of completeness. Other IE projects we are aware of and that we are not able to cover here due to space considerations are Freebase [10] and DeepDive [72].

The following descriptions of the information extraction projects demonstrate that all use a combination of statistical and logical formalism to extract facts and to improve the quality of the derived knowledge. Hence, information extraction projects are prime examples where statistical relational learning and joint inference proves tremendously useful and is naturally applicable. It is also interesting to observe that many of these projects have strong commonalities despite their different objectives and premises. The main motivation for presenting the various approaches to knowledge base extraction is to demonstrate the importance of methods that combine probability and logic and to excite the reader with a semantic web background about the data that these projects continuously aggregate. There are numerous research directions for young researchers to pursue.

2.1 YAGO

YAGO was introduced with the publication [87]. Each entity in YAGO corresponds to an article in Wikipedia. Whenever Wikipedia's volunteer editors deem an entity worthy of a Wikipedia article, YAGO will create the corresponding entity in its knowledge base. The taxonomic backbone of YAGO is based on a hierarchy of user-created Wikipedia categories. YAGO establishes links between Wikipedia categories and synsets in WordNet [28].

YAGO has roughly 100 manually defined relations, such as `locatedIn` and `hasPopulation`. YAGO extracts instances of these relations from Wikipedia infoboxes (meta-data boxes). These instances are commonly denoted as facts: triples of an entity (the subject), a relation (the predicate), and another entity (the object). YAGO utilizes a set of manually created patterns that map categories and infobox attributes to fact templates. YAGO contains more than 80 million facts involving more than 9 million entities [36].

The YAGO knowledge base also utilizes a set of deterministic and probabilistic rules. These declarative rules are used to ensure that facts do not contradict each other in certain ways. For instance, some of these declarative rules specify the domains and ranges of the relations and the definition of the classes of the YAGO concept hierarchy. The rules can, for instance, be used to enforce that instances

of relations adhere to the domain and range restrictions, essentially filtering out incorrect triples.

So-called implication rules are used to infer novel facts from a set of existing ones. For instance, knowing that Baden-Wuerttemberg is locatedIn Germany and Heidelberg is locatedIn Baden-Wuerttemberg one could infer that Heidelberg is also locatedIn Germany.

There are several other classes of rules such as replacement rules for interpreting microformats, cleaning up HTML tags, and normalizing numbers and extraction rules which detect particular patterns in the Wikipedia infoboxes to extract facts.

With the second generation of YAGO, spatial and temporal information was included in the knowledge base [36]. For instance, geographical locations are stored with their geographical coordinates and relation instances with a duration if they have a known start and/or end date.

2.2 DBpedia

DBpedia [4,59] is a project that is in many ways similar to the YAGO project. Both projects operate on Wikipedia pages and use page URIs as canonical labels of entities. Both maintain a set of linkage and extraction rules for Wikipedia infoboxes. While YAGO's rules are maintained by its creators, DBpedia's rules are created and maintained with the help of a sizable user community. In addition, the two projects can be distinguished based on their ontology. Both projects have a fixed set of classes and relations. However, the DBpedia ontology was manually created and not learned based on the Wikipedia categories as in the case of YAGO. Moreover, the DBpedia ontology is with more than 1000 different relations much broader than the YAGO ontology with respect to the relationships it currently models. The developers of the DBpedia knowledge base also emphasize and support it as a multi-lingual knowledge base [59] in the original spirit of Wikipedia as an encyclopedia in numerous different languages.

Dbpedia represent its data in accordance with the best-practices of publishing linked open data. The term *linked data* describes an assortment of best practices for publishing, sharing, and connecting structured data and knowledge over the web [8]. These standards include the assignment of URIs to each datum, the use of the HTTP protocol, RDF data model (Resource Description Framework), and hyperlinks to other URIs [6]. Whereas the traditional World Wide Web is a collection of documents and hyperlinks between these documents, the data web extends this to a collection of arbitrary objects (resources) and their properties and relations. DBpedias relations are modeled using the resource description framework (RDF)[52], a generic graph-based data model for describing objects and their relationships with each other. There are many different representation formats, known as serializations, for RDF data. Examples of RDF serializations include RDF/XML, Notation-3 (N3), Turtle, N-Triples, RDFa, and RDF/JSON. For numerous of DBpedia's relations a domain and range restriction is specified.

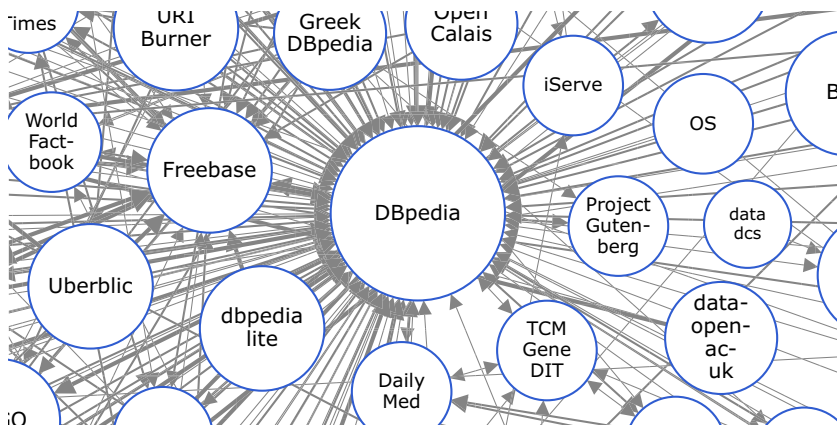


Fig. 1. A small fragment of the web of data. DBpedia is a de-facto hub of the linked open data cloud.

2.3 NELL

The never ending language learning [62,13] (NELL) project's objective is the creation and maintenance of a large-scale machine learning system that *learns* to extract structured information from unstructured web pages. NELL distinguishes itself from YAGO and DBpedia in that its extraction algorithms operate on a large corpus of more than 500 million web pages¹ and not solely on the set of Wikipedia articles. The NELL system was bootstrapped with a small set of classes and relations and, for each of those, 10-15 positive and negative instances. The guiding principle of NELL is to build several semi-supervised machine learning [14] components that accumulate instances of the classes and relations, re-train the machine learning algorithms with these instances as training data, and re-apply the machine learning algorithms to extract novel instances. This process is repeated indefinitely with each re-training and extraction phase called an iteration. Since numerous extraction components work in parallel, extracting facts with different degrees of confidence in their correctness, one of the most important aspects of NELL is its ability to combine these different extraction algorithms into one coherent model. This is also accomplished with relatively simple linear machine learning algorithms that weigh the different components based on their past accuracy.

NELL's algorithm have been running since 2010, initially fully automated and without any human supervision. Since it has experienced concepts drift for some of its relations and classes, that is, a increasingly worse extraction performance over time NELL now is given some corrections by humans to avoid this long-term behavior.

¹ <http://lemurproject.org/clueweb09/>

Interestingly, even though NELL does not adhere to any of the semantic web standards such as RDF or description logics numerous of the same notions are implicitly used. For instance, NELL types the domain and range of its relations.

2.4 Open Information Extraction

Open information extraction [23,24] (OEI) is an umbrella term coined by researchers at the University of Washington to denote information extraction approaches that are not constrained to a predefined set of relations and human-annotated data. On the schema expressivity spectrum of information extraction projects it is one the opposite end of DBpedia and YAGO as it does not leverage or learn an explicit relational schema. In contrast to DBpedia and YAGO which leverage human-created, supervised rules to build their knowledge base and NELL which follows a semi-supervised machine learning approach to information extraction, the open information extraction paradigm is unsupervised and operates solely on large text collections. To avoid low quality extractions, a problem that open IE projects without relational schema are more prone to, several of the open IE extractors apply syntactical and lexical meta-constraints to the extractions. For instance, ReVerb [24] uses a syntactic constraint that requires extracted relation phrases to match a particular part of speech pattern. This pattern limits relation phrases to be either a simple verb phrase, a verb phrase followed immediately by a preposition or particle (e.g., located in), or a verb phrase followed by a simple noun phrase and ending in a preposition or particle (e.g., has atomic weight of) [24]. A lexical constraint enforces a particular relation phrase to occur several times with different arguments (i.e., subjects and objects) in a large text corpus. This is a method that works well in reducing erroneous extractions.

More recent advances within open information extraction is the learning of logical rules of inference from web text [83]. One of the insights coming out of this line of research is that extracting and using inference rules leads to three times more extracted facts than approaches that merely extract explicitly stated facts from text. Again, even within the framework of open information extraction the combination of logical rules and statistical processing of large text corpora emerges as an approach that it has in common with the more structured, schema-driven approaches.

3 Statistical Relational Data Integration

The integration of distributed information sources is a key challenge in data and knowledge management applications. Instances of this problem range from mapping schemas of heterogeneous databases to object reconciliation in linked open data repositories. Both problems are crucial to the integration of heterogeneous knowledge bases learned through information extraction algorithms. In the following, we discuss two instances of the data integration problem: ontology matching and object reconciliation. Both problems have been in the focus of the

semantic web community in recent years. We investigate and assess the applicability and performance of our probabilistic-logical approach to data integration using these two prominent problems. In order to make the article comprehensive, however, we first briefly cover description logics and ontologies as these logical concepts are needed in later parts of the document.

3.1 Ontologies and Description Logics

An Ontology usually groups objects of the world that have certain properties in common (e.g. cities or countries) into concepts. A specification of the shared properties that characterize a set of objects is called a concept definition. Concepts can be arranged into a subclass–superclass relation in order to further discriminate objects into subgroups (e.g. capitals or European countries). Concepts can be defined in two ways, by enumeration of its members or by a concept expression. The specific logical operators that can be used to formulate concept expressions can vary between ontology languages.

Description logics are decidable fragments of first order logic that are designed to describe concepts in terms of complex logical expressions² The basic modeling elements in description logics are concepts (classes of objects), roles (binary relations between objects) and individuals (named objects). Based on these modeling elements, description logics contain operators for specifying so-called concept expressions that can be used to specify necessary and sufficient conditions for membership in the concept they describe. These modeling elements are provided with a formal semantics in terms of an abstract domain interpretation mapping \mathcal{I} mapping each instance onto an element of an abstract domain $\Delta^{\mathcal{I}}$. Instances can be connected by binary relations defined as subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Concepts are interpreted as a subset of the abstract domain Δ . Intuitively, a concept is a set of instances that share certain properties. These properties are defined in terms of concept expressions. Typical operators are the Boolean operators as well as universal and existential quantification over relations to instances in other concepts.

A description logic knowledge base consists of two parts. The A-Box contains information about objects, their type and relations between them, the so-called T-Box consists of a set of axioms about concepts (potentially defined in terms of complex concept expressions and relations. The first type of axioms can be used to describe instances. In particular, axioms can be used to state that an instance belongs to a concept or that two instances are in a certain relation. It is easy to see, that these axioms can be used to capture case descriptions as labeled graphs. The other types of axioms describe relations between concepts and instances. It can be stated that one concept is a subconcept of the other (all its instances are also instances of this other concept). Further, we can define a relation to be a subrelation or the inverse of another relation. The formal semantics of concepts and relations as defined by the interpretation into the abstract domain $\Delta^{\mathcal{I}}$ can

² Details about the relation between description logics and first-order logic can be found in [11] and [88].

Table 1. Axiom patterns for representing description logic ontologies

DL Axiom	Semantics	Intuition
A-Box		
$C(x)$	$x^{\mathcal{I}} \in C^{\mathcal{I}}$	x is of type C
$r(x, y)$	$(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}}$	x is related to y by r
T-Box		
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	C is more specific than D
$C \sqcap D \sqsubseteq \perp$	$C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$	C and D are disjoint
$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$	r is more specific than s
$r \equiv s^{-}$	$r^{\mathcal{I}} = \{(x, y) (y, x) \in s^{\mathcal{I}}\}$	r is the inverse of s
$\exists r. \top \sqsubseteq C$	$(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}} \Rightarrow x^{\mathcal{I}} \in C^{\mathcal{I}}$	the domain of r is restricted to C
$\exists r^{-1}. \top \sqsubseteq C$	$(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}} \Rightarrow y^{\mathcal{I}} \in C^{\mathcal{I}}$	the range of r is restricted to C

be used to automatically infer new axioms from existing definitions. Table 1 lists a few examples of DL axioms, their semantics, and the intuition behind them.

Encoding ontologies in description logics is beneficial, because it enables inference engines to reason about ontological definitions. In this context, deciding subsumption between two concept expressions, i.e. deciding whether one expression is more general than the other one is one of the most important reasoning tasks as it has been used to support various tasks.

3.2 Ontology Matching

Ontology matching is the process of detecting links between entities in heterogeneous ontologies. Based on a definition by Euzenat and Shvaiko [26], we formally introduce the notion of *correspondence* and *alignment* to refer to these links.

Definition 1 (Correspondence and Alignment). *Given ontologies \mathcal{O}_1 and \mathcal{O}_2 , let q be a function that defines sets of matchable entities $q(\mathcal{O}_1)$ and $q(\mathcal{O}_2)$. A correspondence between \mathcal{O}_1 and \mathcal{O}_2 is a triple $\langle 3, e_1, e_2 \rangle r$ such that $e_1 \in q(\mathcal{O}_1)$, $e_2 \in q(\mathcal{O}_2)$, and r is a semantic relation. An alignment between \mathcal{O}_1 and \mathcal{O}_2 is a set of correspondences between \mathcal{O}_1 and \mathcal{O}_2 .*

The generic form of Definition 1 captures a wide range of correspondences by varying what is admissible as matchable element and semantic relation. In the context of ontology matching, we are only interested in equivalence correspondences between concepts and properties. In the first step of the alignment process most matching systems compute a-priori similarities between matching candidates. These values are typically refined in later phases of the matching process. The underlying assumption is that the degree of similarity is indicative of the likelihood that two entities are equivalent. Given two matchable entities e_1 and e_2 we write $\sigma(e_1, e_2)$ to refer to this kind of a-priori similarity. Before presenting the formal matching framework, we motivate the approach by a simple instance of an ontology matching problem which we use as a running example.

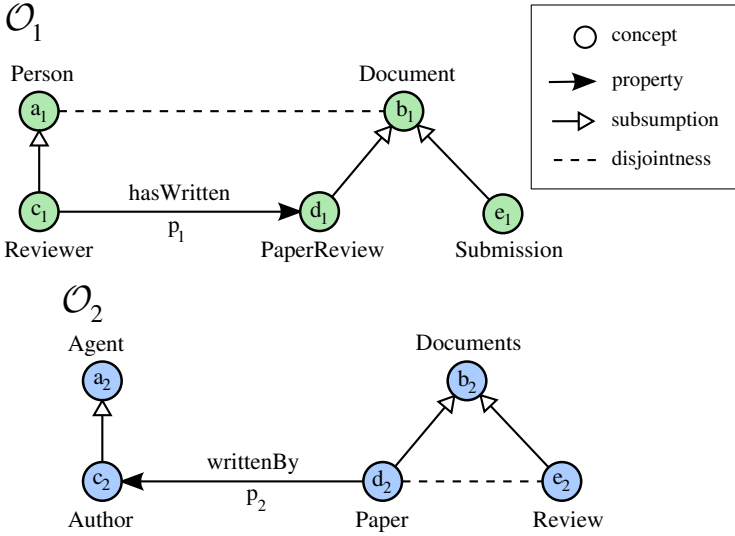


Fig. 2. Example ontology fragments

Example 1. Figure 2 depicts fragments of two ontologies describing the domain of scientific conferences. The following axioms are part of ontology \mathcal{O}_1 and \mathcal{O}_2 , respectively. If we apply a similarity measure σ based on the Levenshtein distance [50] there are four pairs of entities such that $\sigma(e_1, e_2) > 0.5$.

$$\sigma(\text{Document}, \text{Documents}) = 0.88 \quad (1)$$

$$\sigma(\text{Reviewer}, \text{Review}) = 0.75 \quad (2)$$

$$\sigma(\text{hasWritten}, \text{writtenBy}) = 0.7 \quad (3)$$

$$\sigma(\text{PaperReview}, \text{Review}) = 0.54 \quad (4)$$

The alignment consisting of these four correspondences contains two correct (1 & 4) and two incorrect (2 & 3) correspondences resulting in a precision of 50%.

3.3 Object Reconciliation

The problem of object reconciliation has been a topic of research for more than 50 years. It is also known as the problem of record linkage [29], entity resolution [7], and instance matching [30]. While the majority of the existing methods were developed for the task of matching database records, modern approaches focus mostly on graph-based data representations extended by additional schema information. We discuss the problem of object reconciliation using the notion of instance matching. This allows us to describe it within the well-established ontology matching framework [26]. Ontology matching is the process of detecting links between entities in different ontologies. These links are annotated by a confidence value and a label describing the type of link. Such a link is referred

Table 2. Discription logics axioms in the ontology of Figure 2

Ontology \mathcal{O}_1		Ontology \mathcal{O}_2
$\exists hasWritten \sqsubseteq Reviewer$		$\exists writtenBy \sqsubseteq Paper$
$PaperReview \sqsubseteq Document$		$Review \sqsubseteq Documents$
$Reviewer \sqsubseteq Person$		$Paper \sqsubseteq Documents$
$Submission \sqsubseteq Document$		$Author \sqsubseteq Agent$
$Document \sqsubseteq \neg Person$		$Paper \sqsubseteq \neg Review$

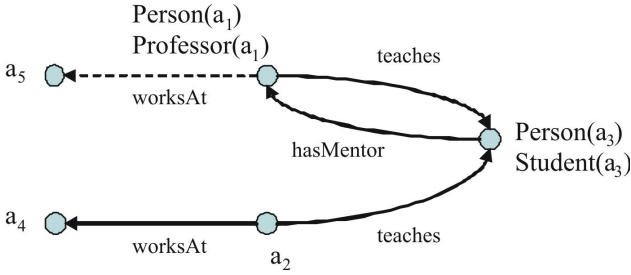
to as a *correspondence* and a set of such correspondences is referred to as an *alignment*.

In the following we refer to an alignment that contains correspondences between concepts and properties as *terminological alignment* and to an alignment that contains correspondences between individuals as *instance alignment*. Since instance matching is the task of detecting pairs of instances that refer to the same real world object [30], the semantic relation expressed by an instance correspondence is that of identity. The confidence value of a correspondence quantifies the degree of trust in the correctness of the statement. If a correspondence is automatically generated by a matching system this value will be computed by aggregating scores from multiple sources of evidence.

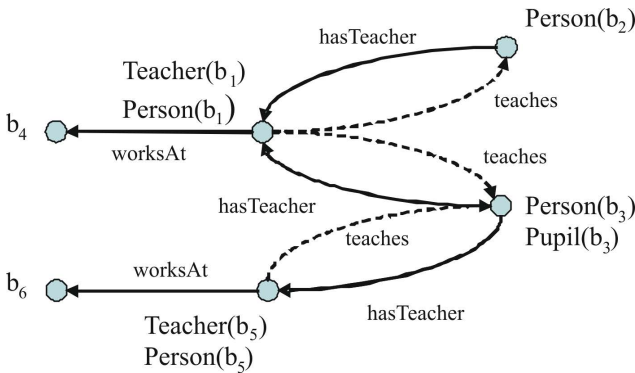
Example 2. An A-Box is a set of membership statements of the following form: $C(a), P(a, b)$ where a, b are individuals, C is a concept name and P is a property name. Further, we extend the notion of an A-Box by also allowing membership statements of the form $\neg C(a)$ and $\neg P(a, b)$ stating that object a is not a member of concept C and that the objects a and b are not an instance of the object property R , respectively. We illustrate the problem of object reconciliation using the following example A-Boxes and their corresponding graphs. A-Boxes can be regarded as labeled directed multi-graphs, where individuals are represented by nodes and object properties are represented by links labeled with the name of the corresponding object property. Object reconciliation is the task of finding the 'right' mapping between the nodes in different A-Box graphs. The basis for finding the right mapping between different objects is typically based on a measure of similarity between the nodes that is determined on the local or global structures in the corresponding graph. Typical features for determining the similarity of two objects are:

- the similarity of their labels
- the similarity of the classes the objects belong to
- the similarity of object properties and related objects

Based on these features, we would generate a priori similarities. For the example depicted in Figure 2 we would compute high a-priori similarities $\sigma(a_5, b_4)$, $\sigma(a_1, b_1)$, $\sigma(a_3, b_3)$, $\sigma(a_3, b_2)$, $\sigma(a_2, b_5)$ and $\sigma(a_4, b_6)$. Besides the similarity between objects, in the case where the A-Box is based on an ontology, the logical



(a) Graph for A-Box \mathcal{A}_1



(b) Graph for A-Box \mathcal{A}_2

Fig. 3. Examples of two different A-Boxes representing triples in RDF. Dashed arrows indicate object properties that occur in both A-Boxes.

constraints from the ontologies should be taken into account in the matching process. In particular, objects should not be maps on each other if they have incompatible types. In the example this means that assuming the underlying ontology contains a statement $student \perp pupil$ declaring the classes 'student' and 'pupil' as disjoint, the objects a_3 and b_3 should not be mapped on each other, despite the high a priori similarity.

4 Statistical Relational Learning

Data integration for heterogeneous knowledge bases typically involves both purely logical and uncertain data. For instance, the description logic axioms of the ontologies are known to be true and, therefore, should be modeled as logical rules – the alignment system should not alter the logical structure of the input ontologies. Conversely, matching systems usually rely on degrees of confidence that have been derived through the application of lexical similarity, data

mining, and machine learning algorithms. The presence of both known logical rules and degrees of uncertainty requires formalism that allow the representation of both deterministic and uncertain aspects of the problem. In the following, we introduce such a probabilistic-logical framework based on Markov logic and show how description logic ontologies are represented in the language. Moreover, we describe the application of an efficient probabilistic inference algorithm that uses integer linear programming. The main reason for focusing on Markov logic is its declarative and easy to understand syntax as well as its rather expressive nature – numerous other statistical relational formalism can be represented with Markov logic networks. However, depending on the application and the intended users, other statistical relational learning (SRL) formalisms might be more appropriate and we make no claim of Markov logic being superior to other languages.

4.1 Markov Logic

A Markov logic network (MLN) is a set of first-order formulas with weights. Intuitively, the more evidence we have that a formula is true the higher the weight of this formula. To simplify the presentation of the technical parts we do *not* include functions. Moreover, Markov logic makes several assumptions such as (a) different constants refer to different objects (the unique names assumption) and (b) the only objects in the domain are those representable using the constants (the closed domain assumption) [80]. In addition, we assume that all (ground) formulas of a Markov logic network are in clausal form and use the terms *formula* and *clause* interchangeably.

Syntax. A signature is a triple $S = (\mathbf{O}, \mathbf{H}, \mathbf{C})$ with \mathbf{O} a finite set of observable predicate symbols, \mathbf{H} a finite set of hidden predicate symbols, and \mathbf{C} a finite set of constants. A Markov logic network (MLN) is a set of pairs $\{(F_i, w_i)\}$ with each F_i being a function-free first-order formula built using predicates from $\mathbf{O} \cup \mathbf{H}$ and each $w_i \in \mathbb{R}$ a real-valued weight associated with formula F_i . For most application of Markov logic, one also wants to include so-called *hard formulas*. Hard formulas are constraints on the possible world. Intuitively speaking, whenever a possible world violates a constraint its the probability of said world is either zero or close to zero. There are different to define Markov logic networks depending on whether one wants to model hard formulas with large weights (resulting in a probability close to zero of a possible world violating it) or with constraint (resulting in a zero probability of possible worlds violating it). For the sake of simplicity, we will use the standard definition with weights only. Later, when we cover some inference methods, we will also introduce the alternative definition.

Semantics. Let $M = \{(F_i, w_i)\}$ be a Markov logic network with signature $S = (\mathbf{O}, \mathbf{H}, \mathbf{C})$. A *grounding* of a first-order formula F is generated by substituting each occurrence of every variable in F with constants in \mathbf{C} . Existentially quantified formulas are substituted by the disjunctions of their groundings over the finite set of constants. A formula that does not contain any variables is

ground. A formula that consists of a single predicate is an *atom*. Note again that Markov logic makes several assumptions such as (a) different constants refer to different objects and (b) the only objects in the domain are those representable using the constants [80]. A set of ground atoms is a *possible world*. We say that a possible world W *satisfies* a formula F , and write $W \models F$, if F is true in W . Let $\mathcal{G}_F^{\mathbf{C}}$ be the set of all possible groundings of formula F with respect to \mathbf{C} . We say that W satisfies $\mathcal{G}_F^{\mathbf{C}}$, and write $W \models \mathcal{G}_F^{\mathbf{C}}$, if F satisfies every formula in $\mathcal{G}_F^{\mathbf{C}}$. Let \mathcal{W} be the set of all possible worlds with respect to S . Then, the probability of a possible world W is given by

$$p(W) = \frac{1}{Z} \exp \left(\sum_{(F_i, w_i)} \sum_{G \in \mathcal{G}_{F_i}^{\mathbf{C}}: W \models G} w_i \right).$$

Here, Z is a normalization constant. The *score* s_W of a possible world W is the sum of the weights of the ground formulas implied by W

$$s_W = \sum_{(F_i, w_i)} \sum_{G \in \mathcal{G}_{F_i}^{\mathbf{C}}: W \models G} w_i. \quad (5)$$

We will see later that, in the data integration context, possible worlds correspond to possible alignments. Hence, the problem of deriving the most probably alignment given the evidence can be interpreted as finding the possible world W with highest score.

4.2 Representing Ontologies and Alignments in Markov Logic

Our approach for data integration based on logics and probability is now based on the idea of representing description logic ontologies as Markov logic networks and utilizing the weights to incorporate similarity scores into the integration process [67,73,69]. The most obvious way to represent a description logic ontology in Markov logic would be to directly use the first-order translation of the ontology. For instance, the axiom $C \sqsubseteq D$ would be written as $\forall x C(x) \Rightarrow D(x)$. In other words, the representation would simply map between concepts and unary predicates and roles and binary predicates. However, we take a *different* approach by mapping axioms to predicates and use constants to represent the classes and relations in the ontology. Some typical axioms with their respective predicates are the following:

$$\begin{aligned} C \sqsubseteq D & \mapsto \text{sub}(c, d) \\ C \sqcap D \sqsubseteq \perp & \mapsto \text{dis}(c, d) \\ \exists r.T \sqsubseteq C & \mapsto \text{dom}(r, c) \\ \exists r^{-1}.T \sqsubseteq C & \mapsto \text{range}(r, c) \end{aligned}$$

This way of representing description logic ontologies has the advantage that we can model some basic inference rules and directly use them in the probabilistic reasoning process. For example, we can model the transitivity of the subsumption relation as

$$\text{sub}(x, y) \wedge \text{sub}(y, z) \Rightarrow \text{sub}(x, z)$$

Table 3. The description logic \mathcal{EL}^{++} without nominals and concrete domains

Name	Syntax	Semantics
top	\top	$\Delta^{\mathcal{I}}$
bottom	\perp	\emptyset
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
RI	$r_1 \circ \dots \circ r_k \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \dots \circ r_k^{\mathcal{I}} \subseteq r^{\mathcal{I}}$

and the intuition that two classes that subsume each other should not be disjoint at the same time³

$$\neg sub(x, y) \vee \neg dis(x, y)$$

While the use of such axioms in a Markov logic network does not guarantee consistency and coherence of the results, they often cover the vast majority of conflicts that can exist in an ontology, especially in cases where the ontology is rather simple and does not contain a complex axiomatization.

For certain description logics, it is possible to completely capture the model using the kind of translation described above. In particular, if an ontology can be reduced to a normal form with a limited number of axiom types, we can provide a complete translation based on this normal form. An example for such a description logic is \mathcal{EL}^{++} , a light weight description logic that supports polynomial time reasoning. Table 3 shows the types of axioms an \mathcal{EL}^{++} Model can be reduced to.

We can completely translation any \mathcal{EL}^{++} model into a Markov Logic representation using the following translation rules:

$$\begin{aligned}
 C_1 \sqsubseteq D &\mapsto sub(c_1, d) \\
 C_1 \sqcap C_2 \sqsubseteq D &\mapsto int(c_1, c_2, d) \\
 C_1 \sqsubseteq \exists r.C_2 &\mapsto rsup(c_1, r, c_2) \\
 \exists r.C_1 \sqsubseteq D &\mapsto rsub(c_1, r, d) \\
 r \sqsubseteq s &\mapsto psub(r, s) \\
 r_1 \circ r_2 \sqsubseteq r_3 &\mapsto pcom(r_1, r_2, r_3)
 \end{aligned}$$

In principle, such a complete translation is possible whenever there is a normal form representation of a description logic that reduces the original model to a finite number of axiom types that can be captured by a respective predicate in the Markov logic network.

³ Please note that two classes can be disjoint and subsume each other whenever one of the classes is empty. However, these are exactly the situation we want to avoid when we integrate two or more ontologies.

Finally, being interested in data integration, we often treat correspondences between elements from different models separately although in principle they could be represented by ordinary DL axioms. In particular, we often use the following translation of correspondences to weighted ground predicates of the Markov logic network

$$(e_1, e_2, R, c) \mapsto \langle \text{map}_R(e_1, e_2), c \rangle$$

where c is a a-priori confidence values.

5 Markov Logic and Ontology Matching

We provide a formalization of the ontology matching problem within the probabilistic-logical framework. The presented approach has several advantages over existing methods such as ease of experimentation, incoherence mitigation during the alignment process, and the incorporation of a-priori confidence values. We show empirically that the approach is efficient and more accurate than existing matchers on an established ontology alignment benchmark dataset.

5.1 Problem Representation

Given two ontologies \mathcal{O}_1 and \mathcal{O}_2 and an initial a-priori similarity σ we apply the following formalization. First, we introduce observable predicates O to model the structure of \mathcal{O}_1 and \mathcal{O}_2 with respect to both concepts and properties. For the sake of simplicity we use uppercase letters D, E, R to refer to individual concepts and properties in the ontologies and lowercase letters d, e, r to refer to the corresponding constants in \mathcal{C} . In particular, we add ground atoms of observable predicates to the set of hard formulas for $i \in \{1, 2\}$ according to the following rules:

$$\begin{array}{lll}
 \mathcal{O}_i \models D \sqsubseteq E & \mapsto & \text{sub}_i(d, e) \\
 \mathcal{O}_i \models D \sqsubseteq \neg E & \mapsto & \text{dis}_i(d, e) \\
 \mathcal{O}_i \models \exists R. \top \sqsubseteq D & \mapsto & \text{sub}_i^d(r, d) \\
 \mathcal{O}_i \models \exists R^{-1}. \top \sqsubseteq D & \mapsto & \text{sub}_i^r(r, d) \\
 \mathcal{O}_i \models \exists R. \top \sqsupseteq D & \mapsto & \text{sup}_i^d(r, d) \\
 \mathcal{O}_i \models \exists R^{-1}. \top \sqsupseteq D & \mapsto & \text{sup}_i^r(r, d) \\
 \mathcal{O}_i \models \exists R. \top \sqsubseteq \neg D & \mapsto & \text{dis}_i^d(r, d) \\
 \mathcal{O}_i \models \exists R^{-1}. \top \sqsubseteq \neg D & \mapsto & \text{dis}_i^r(r, d)
 \end{array}$$

The knowledge encoded in the ontologies is assumed to be true. Hence, the ground atoms of observable predicates are added to the set of hard formulas, making them hold in every computed alignment. The hidden predicates map_c

and map_p , on the other hand, model the sought-after concept and property correspondences, respectively. Given the state of the observable predicates, we are interested in determining the state of the hidden predicates that maximize the a-posteriori probability of the corresponding possible world. The ground atoms of these hidden predicates are assigned the weights specified by the a-priori similarity σ . The higher this value for a correspondence the more likely the correspondence is correct *a-priori*. Hence, the following ground formulas are added to the set of formulas:

$$\begin{array}{ll} (map_c(c, d), \sigma(C, D)) & \text{if C and D are concepts} \\ (map_p(p, r), \sigma(P, R)) & \text{if P and R are properties} \end{array}$$

Notice that the distinction between m_c and m_p is required since we use typed predicates and distinguish between the *concept* and *property* type.

Cardinality Constraints. A method often applied in real-world scenarios is the selection of a functional one-to-one alignment [17]. Within the ML framework, we can include a set of hard cardinality constraints, restricting the alignment to be functional and one-to-one. In the following we write x, y, z to refer to variables ranging over the appropriately typed constants and omit the universal quantifiers.

$$\begin{array}{l} map_c(x, y) \wedge map_c(x, z) \Rightarrow y = z \\ map_c(x, y) \wedge map_c(z, y) \Rightarrow x = z \end{array}$$

Analogously, the same formulas can be included with hidden predicates map_p , restricting the property alignment to be one-to-one and functional.

Coherence Constraints. Incoherence occurs when axioms in ontologies lead to logical contradictions. Clearly, it is desirable to avoid incoherence during the alignment process. Some methods of incoherence removal for ontology alignments were introduced in [57]. All existing approaches, however, remove correspondences *after* the computation of the alignment. Within the ML framework we can incorporate incoherence reducing constraints *during* the alignment process for the first time. This is accomplished by adding formulas of the following type to set of hard formulas.

$$\begin{array}{l} dis_1(x, x') \wedge sub_2(x, x') \Rightarrow \neg(map_c(x, y) \wedge map_c(x', y')) \\ dis_1^d(x, x') \wedge sub_2^d(y, y') \Rightarrow \neg(map_p(x, y) \wedge map_c(x', y')) \end{array}$$

The second formula, for example, has the following purpose. Given properties X, Y and concepts X', Y' . Suppose that $\mathcal{O}_1 \models \exists X. \top \sqsubseteq \neg X'$ and $\mathcal{O}_2 \models \exists Y. \top \sqsubseteq Y'$. Now, if $\langle X, Y, \equiv \rangle$ and $\langle X', Y', \equiv \rangle$ were both part of an alignment the merged ontology would entail both $\exists X. \top \sqsubseteq X'$ and $\exists X. \top \sqsubseteq \neg X'$ and, therefore, $\exists X. \top \sqsubseteq \perp$. The specified formula prevents this type of incoherence. It is known that such constraints, if carefully chosen, can avoid a majority of possible incoherences [56].

Stability Constraints. Several existing approaches to schema and ontology matching propagate alignment evidence derived from structural relationships between concepts and properties. These methods leverage the fact that existing evidence for the equivalence of concepts C and D also makes it more likely that, for example, child concepts of C and child concepts of D are equivalent. One such approach to evidence propagation is *similarity flooding* [58]. As a reciprocal idea, the general notion of stability was introduced, expressing that an alignment should not introduce new structural knowledge [55]. The *soft* formula below, for instance, decreases the probability of alignments that map concepts X to Y and X' to Y' if X' subsumes X but Y' does *not* subsume Y .

$$\begin{aligned} \langle sub_1(x, x') \wedge \neg sub_2(y, y') \Rightarrow map_c(x, y) \wedge map_c(x', y'), w_1 \rangle \\ \langle sub_1^d(x, x') \wedge \neg sub_2^d(y, y') \Rightarrow map_p(x, y) \wedge map_c(x', y'), w_2 \rangle \end{aligned}$$

Here, w_1 and w_2 are *negative* real-valued weights, rendering alignments that satisfy the formulas possible but less likely.

The presented list of cardinality, coherence, and stability constraints is by no means meant to be exhaustive. Other constraints could, for example, model known correct correspondences or generalize the one-to-one alignment to m-to-n alignments. Moreover, a novel hidden predicate could be added modeling correspondences between instances of the ontologies. To keep the discussion of the approach simple, however, we leave these considerations to future research.

Example 3. We apply the previous formalization to Example 1. To keep it simple, we only use a-priori values, cardinality, and coherence constraints. Given the two ontologies \mathcal{O}_1 and \mathcal{O}_2 in Figure 2, and the matching hypotheses (1) to (4) from Example 1, the ground MLN would include the following relevant ground formulas. We use the concept and property labels from Figure 2 and omit ground atoms of observable predicates.

A-priori similarity:

$$\langle map_c(b_1, b_2), 0.88 \rangle, \langle map_c(c_1, e_2), 0.75 \rangle, \langle map_p(p_1, p_2), 0.7 \rangle, \langle map_c(d_1, e_2), 0.54 \rangle$$

Cardinality constraints:

$$map_c(c_1, e_2) \wedge map_c(d_1, e_2) \Rightarrow c_1 = d_1 \quad (6)$$

Coherence constraints:

$$dis_1^d(p_1, b_1) \wedge sub_2^d(p_2, b_2) \Rightarrow \neg(map_p(p_1, p_2) \wedge map_c(b_1, b_2)) \quad (7)$$

$$dis_1(b_1, c_1) \wedge sub_2(b_2, e_2) \Rightarrow \neg(map_c(b_1, b_2) \wedge map_c(c_1, e_2)) \quad (8)$$

$$sub_1^d(p_1, c_1) \wedge dis_2^d(p_2, e_2) \Rightarrow \neg(map_p(p_1, p_2) \wedge map_c(c_1, e_2)) \quad (9)$$

Let the binary ILP variables x_1, x_2, x_3 , and x_4 model the ground atoms $map_c(b_1, b_2)$, $map_c(c_1, e_2)$, $map_p(p_1, p_2)$, and $map_c(d_1, e_2)$, respectively. The set of ground formulas is then encoded in the following integer linear program:

Maximize: $0.88x_1 + 0.75x_2 + 0.7x_3 + 0.54x_4$

Subject to:

$$x_2 + x_4 \leq 1 \tag{10}$$

$$x_1 + x_3 \leq 1 \tag{11}$$

$$x_1 + x_2 \leq 1 \tag{12}$$

$$x_2 + x_3 \leq 1 \tag{13}$$

The a-priori confidence values of the potential correspondences are factored in as coefficients of the objective function. Here, the ILP constraint (9) corresponds to ground formula (5), and ILP constraints (10),(11), and (12) correspond to the coherence ground formulas (6), (7), and (8), respectively. An optimal solution to the ILP consists of the variables x_1 and x_4 corresponding to the correct alignment $\{m_c(b_1, b_2), m_c(d_1, e_2)\}$. Compare this with the alignment $\{map_c(b_1, b_2), map_c(c_1, e_2), map_p(p_1, p_2)\}$ which would be the outcome without coherence constraints.

6 Markov Logic and Object Reconciliation

We are primarily concerned with the scenario where both A-Boxes are described in terms of the same T-Box. This is a reasonable assumption if we want to integrate information extraction projects with a limited set of classes and relations. In this case, it is often straight-forward to align the ontologies of the involved knowledge bases and to exploit these links for improving the alignment between individuals. However, for open information extraction projects, where the number of relations is not bounded and essentially every surface form could correspond to a particular relation, this approach is not feasible and has to be substituted with one that aligns relations and individuals jointly. We will not discuss this scenario here.

Instead, we present an approach that does not rely on specific types of axioms or a set of predefined rules but computes the alignment by maximizing the similarity of the two knowledge bases given the alignment subject to a set of constraints. Our method factors in a-priori confidence values that quantify the degree of trust one has in the correctness of the object correspondences based on lexical properties. The resulting similarity measure is used to determine an instance alignment that induces the highest agreement of object assertions in \mathcal{A}_1 and \mathcal{A}_2 with respect to \mathcal{T} .

6.1 Problem Representation

The current instance matching configuration leverages terminological structure and combines it with lexical similarity measures. The approach is presented in more detail in [75]. The alignment system uses one T-Box \mathcal{T} but two different A-Boxes $\mathcal{A}_1 \in \mathcal{O}_1$ and $\mathcal{A}_2 \in \mathcal{O}_2$. In cases with two different T-Boxes the T-Box

matching approach is applied as a preprocessing step to merge the two aligned T-Boxes first. The approach offers complete conflict elimination meaning that the resulting alignment is always consistent for OWL DL ontologies. To enforce consistency, we need to add constraints to model conflicts, that is, we have to prevent an equivalence correspondence between two individuals if there exists a positive class assertion for the first individual and a negative for the second for the same class. These constraints are incorporated for both property and concept assertions. Analogous to the concept and property alignment before, we introduce the hidden predicate map_i representing instance correspondences. Let C be a concept and P be a property of T-Box \mathcal{T} . Further, let $A \in \mathcal{A}_1$ and $B \in \mathcal{A}_2$ be individuals in the respective A-Boxes. Then, using a reasoner such as Pellet, ground atoms are added to the set of *hard* constraints according to the following rules:

$$\begin{array}{lll}
 \mathcal{T} \cup \mathcal{A}_1 \models C(A) & \wedge \mathcal{T} \cup \mathcal{A}_2 \models \neg C(B) & \mapsto \neg map_i(a, b) \\
 \mathcal{T} \cup \mathcal{A}_1 \models \neg C(A) & \wedge \mathcal{T} \cup \mathcal{A}_2 \models C(B) & \mapsto \neg map_i(a, b) \\
 \mathcal{T} \cup \mathcal{A}_1 \models P(A, A') & \wedge \mathcal{T} \cup \mathcal{A}_2 \models \neg P(B, B') & \mapsto \neg map_i(a, b) \vee \neg map_i(a', b') \\
 \mathcal{T} \cup \mathcal{A}_1 \models \neg P(A, A') & \wedge \mathcal{T} \cup \mathcal{A}_2 \models P(B, B') & \mapsto \neg map_i(a, b) \vee \neg map_i(a', b')
 \end{array}$$

In addition to these formulas we included cardinality constraints analogous to those used in the previous concept and property alignment problem. In the instance matching formulation, the a-priori similarity σ_c and σ_p measures the *normalized overlap* of concept and property assertions, respectively. For more details on these measures, we refer the reader to [75]. The following formulas are added to the set of formulas:

$$\begin{array}{ll}
 \langle map_i(a, b), \sigma_c(A, B) \rangle & \text{if A and B are instances} \\
 \langle map_i(a, b) \wedge map_i(c, d), \sigma_p(A, B, C, D) \rangle & \text{if A, B, C, and D are instances}
 \end{array}$$

7 Efficient Probabilistic Inference

Many successful applications of artificial intelligence research are based on large probabilistic models. Examples include Markov logic networks [80], conditional random fields [48] and, more recently, deep learning architectures [35,5,79]. The statistical relational models resulting from common data integration problems are usually very large and in many cases intractable. Indeed, probabilistic inference in general graphical models is known to be NP-hard [46]. Hence, the problem of probabilistic inference in large statistical relational models seems daunting considering the size and complexity of the resulting probabilistic graphical models. For numerous of these models, however, scalable approximate and, to a lesser extend, exact inference algorithms do exist. Most notably, there has been a strong focus on lifted inference algorithms, that is, algorithms that group indistinguishable variables and features during inference. For an overview we refer the reader to [44]. Lifted algorithms facilitate efficient inference in numerous large probabilistic models for which inference is NP-hard in principle. We refer

the reader to the related work section for some lifted inference algorithms related to our work.

The computation of a maximum a-posterior (MAP) state in statistical relational models for data integration corresponds to computing the most probable alignment between relations, classes, and individuals, respectively. Hence, we more closely describe the state-of-the-art MAP inference engine ROCKIT[74].

7.1 Cutting Plane Aggregation

Each MAP query corresponds to an optimization problem with linear constraints and a linear objective function and, hence, we can formulate the problem as an instance of integer linear programming. The novel cutting plane aggregation approach is tightly integrated with cutting plane inference (CPI) a meta-algorithm operating between the grounding algorithm and the ILP solver [81]. Instead of immediately adding one constraint for each ground formula to the ILP formulation, the ILP is initially formulated so as to enforce the given evidence to hold in any solution. Based on the solution of this more compact ILP one determines the violated constraints, adds these to the ILP, and resolves. This process is repeated until no constraints are violated by an intermediate solution.

We begin by introducing a novel ILP formulation of MAP queries for Markov logic networks. In contrast to existing approaches [81,39], the formulation requires only one linear constraint per ground clause irrespective of the ground clause being weighted or unweighted. Moreover, we introduce the notion of *context-specific exchangeability* and describe the novel *cutting plane aggregation* (CPA) algorithm that exploits this type of local symmetry. Contrary to most symmetry-aware and lifted inference algorithms that assume no or only a limited amount of evidence, the presented approach specifically exploits model symmetries induced by the given evidence.

7.2 General ILP Formulation

In order to transform the MAP problem to an ILP we have to first ground, that is, instantiate, the first-order theory specified by the Markov logic network. Since we are employing cutting plane inference, ROCKIT runs in each iteration several join queries in a relational database system to retrieve the ground clauses violated by the current solution. Hence, in each iteration of the algorithm, ROCKIT maintains a set of ground clauses \mathcal{G} that have to be translated to an ILP instance.

Given such a set of ground clauses \mathcal{G} , we associate one binary ILP variable x_ℓ with each ground atom ℓ occurring in some $g \in \mathcal{G}$. For the sake of simplicity, we will often denote ground atoms and ILP variables with identical names. For a ground clause $g \in \mathcal{G}$ let $L^+(g)$ be the set of ground atoms occurring unnegated in g and $L^-(g)$ be the set of ground atoms occurring negated in g . Now, we encode the given evidence by introducing linear constraints of the form $x_\ell \leq 0$ or $x_\ell \geq 1$ depending on whether the evidence sets the corresponding ground atom ℓ to

Table 4. An example of the ILP formulation

weight	ground clause		max $1.1z_1 - 0.5z_2$
			subject to
1.1	$x_1 \vee \neg x_2 \vee x_3$	\rightsquigarrow	$x_1 + (1 - x_2) + x_3 \geq z_1$
-0.5	$\neg x_1 \vee x_2$		$(1 - x_1) + x_2 \leq 2 \cdot z_2$
∞	$\neg x_1 \vee x_2$		$(1 - x_1) + x_2 \geq 1$

false or true. For every ground clause $g \in \mathcal{G}$ with weight $w > 0$, $w \in \mathbb{R}$, we add a novel binary variable z_g and the following constraint to the ILP:

$$\sum_{\ell \in L^+(g)} x_\ell + \sum_{\ell \in L^-(g)} (1 - x_\ell) \geq z_g.$$

Please note that if any of the ground atoms ℓ in the ground clause is set to false (true) by the given evidence, we do not include it in the linear constraint.

For every g with weight $w_g < 0$, $w \in \mathbb{R}$, we add a novel binary variable z_g and the following constraint to the ILP:

$$\sum_{\ell \in L^+(g)} x_\ell + \sum_{\ell \in L^-(g)} (1 - x_\ell) \leq (|L^+(g)| + |L^-(g)|)z_g.$$

For every g with weight $w_g = \infty$, that is, a hard clause, we add the following linear constraint to the ILP:

$$\sum_{\ell \in L^+(g)} x_\ell + \sum_{\ell \in L^-(g)} (1 - x_\ell) \geq 1$$

If a ground clause has zero weight we do not have to add the corresponding constraint.

Finally, the objective of the ILP is:

$$\max \sum_{g \in \mathcal{G}} w_g z_g,$$

where we sum over weighted ground clauses only, w_g is the weight of g , and $z_g \in \{0, 1\}$ is the binary variable previously associated with ground clause g . We compute a MAP state by solving the ILP whose solution corresponds one-to-one to a MAP state \mathbf{x} where $x_i = \mathbf{true}$ if the corresponding ILP variable is 1 and $x_i = \mathbf{false}$ otherwise.

For example, Table 4 depicts three clauses with $w > 0$, $w < 0$, and $w = \infty$, and the respective ILP formulations.

7.3 Constraint Aggregation

In this section we optimize the compilation of *sets* of weighted ground clauses to *sets* of linear constraints. More concretely, we introduce a novel approach

Table 5. A set of ground clauses that can be aggregated

g	ℓ_i	c	w
g_1	$x_1 \vee$	$\neg y_1 \vee y_2$	1.0
g_2	$x_2 \vee$	$\neg y_1 \vee y_2$	1.0
g_3	$\neg x_3 \vee$	$\neg y_1 \vee y_2$	1.0
g_4	$\neg x_4 \vee$	$\neg y_1 \vee y_3$	1.0
g_5	$x_5 \vee$	$\neg y_1$	0.5

 \rightsquigarrow

ℓ_i	c	w
$x_1 \vee$		
$x_2 \vee$	$\neg y_1 \vee y_2$	1.0
$\neg x_3 \vee$		
$\neg x_4 \vee$	$\neg y_1 \vee y_3$	1.0
$x_5 \vee$	$\neg y_1$	0.5

that aggregates sets of ground clauses so as to make the resulting ILP have (a) fewer variables (b) fewer constraints and (c) its context-specific symmetries more exposed to the ILP solver’s symmetry detection heuristics.

We first demonstrate that evidence often introduces symmetries in the resulting sets of ground clauses and, therefore, at the level of ILP constraints. The proposed approach aggregates ground clauses, resulting in smaller constraint matrices and aiding symmetry detection algorithms of the ILP solvers. The solvers apply heuristics to test whether the ILP’s constraint matrix exhibits symmetries in form of permutations of its columns and rows. For a comprehensive overview of existing principles and algorithms for detecting and exploiting symmetries in integer linear programs we refer the reader to [54,53,76,9]. We describe cutting plane aggregation in two steps. First, we explain the aggregation of ground formulas and, second, we describe the compilation of aggregated formulas to ILP constraints.

Definition 2. Let $G \subseteq \mathcal{G}$ be a set of n weighted ground clauses and let c be a ground clause. We say that G can be aggregated with respect to c if (a) all ground clauses in G have the same weight and (b) for every $g_i \in G, 1 \leq i \leq |G|$, we have that $g_i = \ell_i \vee c$ where ℓ_i is a (unnegated or negated) literal for each $i, 1 \leq i \leq |G|$.

Example 4. Table 5 lists a set of 5 ground clauses. The set of clauses $\{g_1, g_2, g_3\}$ can be aggregated with respect to $\neg y_1 \vee y_2$ since we can write each of these ground clauses as $\ell_i \vee \neg y_1 \vee y_2$ with $\ell_1 := x_1, \ell_2 := x_2$, and $\ell_3 := \neg x_3$.

Before we describe the advantages of determining ground clauses that can be aggregated and the corresponding ILP formulation encoding these sets of clauses, we provide a typical instance of a Markov logic network resulting in a large number of clauses that can be aggregated.

Example 5. Let us consider the clause $\neg \text{smokes}(x) \vee \text{cancer}(x)$ and let us assume that there are 100 constants C_1, \dots, C_{100} for which we have evidence $\text{smokes}(C_i), 1 \leq i \leq 100$. For $1 \leq i \leq 100$, let y_i be the ILP variable corresponding to the ground atom $\text{cancer}(C_i)$. The naive formulation would contain 100 constraints $y_i \geq z_i$ and the objective of the ILP with respect to these clauses would be $\max 1.5z_1 + \dots + 1.5z_{100}$. Instead, we can aggregate the ground clauses $\text{cancer}(C_i), 1 \leq i \leq 100$, for $c = \text{false}$ and $\ell_i = \text{cancer}(C_i), 1 \leq i \leq 100$.

Let $G \subseteq \mathcal{G}$ be a set of ground clauses with weight w and let c be a ground clause. Moreover, let us assume that G can be aggregated with respect to c , that is, that

Table 6. Illustration of the constraint aggregation formulation. For the sake of simplicity, we denote the ground atoms and ILP variables with identical names.

ℓ_i	c	w	
$x_1 \vee$			max $0.5z_1 - 1.5z_2$
$x_2 \vee$	$\neg y_1$	0.5	subject to
$x_3 \vee$			$x_1 + x_2 + x_3 + 3(1 - y_1) \geq z_1$
$\neg x_1 \vee$			$z_1 \leq 3$
$x_2 \vee$	$y_1 \vee \neg y_2$	-1.5	$(1 - x_1) + x_2 + (1 - x_3) \leq z_2$
$\neg x_3 \vee$			$3 \cdot y_1 \leq z_2$
			$3 \cdot (1 - y_2) \leq z_2$

each $g \in G$ can be written as $\ell_i \vee c$. The *aggregated feature* f^G for the clauses G with weight w maps each interpretation I to an integer value as follows

$$f^G(I) = \left\{ \begin{array}{ll} |G| & \text{if } I \models c \\ |\{\ell_i \vee c \in G \mid I \models \ell_i\}| & \text{otherwise} \end{array} \right\}.$$

The feature resulting from the aggregation, therefore, counts the number of literals ℓ_i that are satisfied if the ground clause c is not satisfied and returns the number of aggregated clauses otherwise. Please note that an encoding of this feature in a factor graph would require space exponential in the number of ground atoms even though the feature only has a linear number of possible values. The feature, therefore, is highly symmetric – each assignment to the random variables corresponding to the unnegated (negated) literals that has the same Hamming weight results in the same feature weight contribution. This constitutes a feature-specific local form of finite exchangeability [31,20] of random variables induced by the evidence. Therefore, we denote this form of finite exchangeability as *context-specific exchangeability*. Please note that the concept is related to counting formulas used in some lifted inference algorithms [61]. While standard models such as factor graphs cannot represent such symmetric features compactly, one can encode these *counting features* directly with a *constant* number of ILP constraints. We now describe this translation in more detail.

As before, for any ground clause c , let $L^+(c)$ ($L^-(c)$) be the set of ground atoms occurring unnegated (negated) in c . We first show the formulation for clauses with positive weights. Let $G \subseteq \mathcal{G}$ be a set of n ground clauses with weight $w > 0$ that can be aggregated with respect to c , that is, for each $g \in G$ we have that $g = x_i \vee c$ or $g = \neg x_i \vee c$ for some ground atom x_i and a fixed clause c . We now add the following two linear constraints to the ILP:

$$\sum_{(x_i \vee c) \in G} x_i + \sum_{(\neg x_i \vee c) \in G} (1 - x_i) + \sum_{\ell \in L^+(c)} nx_\ell + \sum_{\ell \in L^-(c)} n(1 - x_\ell) \geq z_g \quad (14)$$

and

$$z_g \leq n \quad (15)$$

Linear constraint (2) introduces the novel *integer* variable z_g for each aggregation. Whenever a solution satisfies the ground clause c this variable has the value

n and otherwise it is equal to the number of literals ℓ_i satisfied by the solution. Since constraint (2) alone might lead to values of z_g that are greater than n , the linear constraint (3) ensures that the value of z_g is at most n . However, linear constraint (3) only needs to be added if clause c is not the constant **false**.

We describe the aggregation of clauses with negative weight. Let $G \subseteq \mathcal{G}$ be a set of n ground clauses with weight $w < 0$ that can be aggregated with respect to c , that is, for each $g \in G$ we have that $g = x_i \vee c$ or $g = \neg x_i \vee c$ for a ground atom x_i and a fixed clause c . We now add the following linear constraints to the ILP:

$$\sum_{(x_i \vee c) \in G} x_i + \sum_{(\neg x_i \vee c) \in G} (1 - x_i) \leq z_g \quad (16)$$

and

$$nx_\ell \leq z_g \text{ for every } \ell \in L^+(c) \quad (17)$$

and

$$n(1 - x_\ell) \leq z_g \text{ for every } \ell \in L^-(c). \quad (18)$$

Linear constraint (4) introduces an integer variable z_g that counts the number of ground clauses in G that are satisfied. For each of the integer variables z_g representing an aggregated set of clauses we add the term $w_g z_g$ to the objective function where w_g is the weight of each of the aggregated clauses. Table 6 shows a set of aggregated ground clauses and the corresponding ILP formulation. It is not difficult to verify that each solution of the novel formulation corresponds to a MAP state of the MLN it was constructed from.

Example 6. In Example 5 we aggregate the ground clauses $\text{cancer}(C_i), 1 \leq i \leq 100$, for $c = \text{false}$ and $\ell_i = \text{cancer}(C_i), 1 \leq i \leq 100$. Now, instead of 100 linear constraints and 100 summands in the objective function the aggregated formulation only adds the linear constraint $y_1 + \dots + y_{100} \leq z_g$ and the term $1.5z_g$ to the objective.

In addition to the project code, ROCKIT is made available as a web-service where users can upload MLNs. Furthermore, programmers can integrate the MLN query engine in their own applications via a REST interface.

8 Related Work

There have been a number of approaches for extending description logics with probabilistic information in the earlier days of description logics. Heinsohn [34] was one of the first to propose a probabilistic notion of subsumption for the logic ALC. Jaeger [40] investigated some general problems connected with the extension of T-Boxes and ABoxes with objective and subjective probabilities and proposed a general method for reasoning with probabilistic information in terms of probability intervals attached to description logic axioms. Recently, Giugno and Lukasiewicz proposed a probabilistic extension of the logic SHOQ along the lines sketched by Jaeger [32]. A major advantage of this approach is the integrated treatment of probabilistic information about Conceptual and Instance

knowledge based on the use of nominals in terminological axioms that can be used to model uncertain information about instances and relations. An alternative way of combining description logics with probabilistic information has been proposed by Koller et al. [47]. In contrast to the approaches mentioned above, the P-CLASSIC approach is not based on probability intervals. Instead it uses a complete specification of the probability distribution in terms of a Bayesian network which nodes correspond to concept expressions in the CLASSIC description logic. Bayesian networks have also been used in connection with less expressive logics such as TDL [94]. The approaches for encoding probabilities in concept hierarchies using Bayesian networks described in the section preliminaries and background can be seen as a simple special case of these approaches.

More recently proposals for combining the web ontology language OWL with probabilistic information have been proposed. The first kind of approach implements a loose coupling of the underlying semantics of OWL and probabilistic models. In particular these approaches use OWL as a language for talking about probabilistic models. An example of this approach is the work of Yang and Calmet that propose a minimal OWL ontology for representing random variables and dependencies between random variables with the corresponding conditional probabilities [93]. This allows the user to write down probabilistic models that correspond to Bayesian networks as instances of the OntoBayes Ontology. The encoding of the model in OWL makes it possible to explicitly link random variables to elements of an OWL ontology, a tighter integration on the formal level, however, is missing. A similar approach is proposed by Costa and Laskey. They propose the PR-OWL model which is an OWL ontology for describing first order probabilistic models [15]. More specifically, the corresponding ontology models Multi-Entity Bayesian networks [49] that define probability distributions over first-order theories in a modular way. Similar to OntoBayes, there is no formal integration of the two representation paradigms as OWL is used for encoding the general structure of Multi-entity Bayesian networks on the meta-level. The second kind of approaches actually aims at enriching OWL ontologies with probabilistic information to support uncertain reasoning inside OWL ontologies. These approaches are comparable with the work on probabilistic extensions of description logics also presented in this section. A survey of the existing work reveals, however, that approaches that directly address OWL as an ontology language are less ambitious with respect to combining logical and probabilistic semantics than the work in the DL area. An example is the work of Holi and Hyvonen [37] that describe a framework for representing uncertainty in simple classification hierarchies using Bayesian networks. A slightly more expressive approach called BayesOWL is proposed by Ding and others [21]. They also consider Boolean operators as well as disjointness and equivalence of OWL classes and present an approach for constructing a Bayesian network from class expressions over these constructs. An interesting feature of BayesOWL is some existing work on learning and representing uncertain alignments between different BayesOWL ontologies reported in [77]. An additional family of probabilistic logics are

log-linear description logics [70] which integrate lightweight description logics and probabilistic log-linear models.

Probabilistic approaches to ontology matching based on undirected probabilistic graphical models have recently produced competitive matching results [1]. There are numerous other non-probabilistic approaches to ontology matching and to mention all of them would be beyond the scope of this chapter. We refer the reader to the systems participating in the OAEI [27] which are described in the respective papers. More prominent systems with a long history of OAEI participation are Falcon [38], Aroma [18], ASMOV [41], and AgreementMaker [16]. In the linked open data context, the Silk framework [91] is a suite of tools for finding and creating mappings between entities within different data sources. Silk facilitates a declarative language for specifying which types of RDF links are to be found between data sets. The OAEI [27] also organizes a track for instance matching and we refer the reader to the track's report for an overview of existing linked open data integration algorithms.

The commonly applied methods for object reconciliation include structure-based strategies as well as strategies to compute and aggregate value similarities. Under the notion of instance matching, similarities between instance labels and datatype properties are mostly used to compute confidence values for instance correspondences. Examples of this are realized in the systems RiMOM [95] and OKKAM [86]. Both systems participated in the instance matching track of the Ontology Alignment Evaluation in 2009. Additional refinements are related to a distinction between different types of properties. The developers of RiMOM manually distinguish between *necessary* and *sufficient* datatype properties. The FBEM algorithm of the OKKAM project assigns higher weights to certain properties like names and IDs. In both cases, the employed methods focus on appropriate techniques to interpret and aggregate similarity scores based on a comparison of datatype property values. Another important source of evidence is the knowledge encoded in the T-Box. RiMOM, for example, first generates a terminological alignment between the T-Boxes \mathcal{T}_1 and \mathcal{T}_2 describing the A-Boxes \mathcal{A}_1 and \mathcal{A}_2 , respectively. This alignment is then used as a filter and only correspondences that link instances of equivalent concepts are considered valid [95]. An object reconciliation method applicable to our setting was proposed in [82] where the authors combine logical with numerical methods. For logical reasons it is in some cases possible to preclude that two instances refer to the same object while in other cases the acceptance of one correspondence directly entails the acceptance of another. The authors extend this approach by modeling some of these dependencies into a similarity propagation framework. However, their approach requires a rich schema and assumes that properties are defined to be functional and/or inverse functional. Hence, the approach cannot be used effectively to exploit type information based on a concept hierarchy and is therefore not applicable in many web of data scenarios.

MaxWalkSAT (MWS), a random walk algorithm for solving weighted SAT problems [42], is the standard inference algorithm for MAP queries in the Markov logic engine ALCHEMY [22]. The system TUFFY [71] employs relational database

management systems to ground Markov logic networks more efficiently. TUFFY also runs MWS on the ground model which it initially attempts to partition into disconnected components.

MAP queries in SRL models can be formulated as integer linear programs (ILPs). In this context, cutting plane inference (CPI) solving multiple smaller ILPs in several iterations has shown remarkable performance [81]. In each CPI iteration, only the ground formulas violated by the current intermediate solution are added to the ILP formulation until no violated ground formulas remain. Since CPI *ignores* ground formulas satisfied by the evidence, it can be seen as a generalization of pre-processing approaches that count the formulas satisfied by the evidence [84]. In the context of max-margin weight learning for MLNs [39] the MAP query was formulated as a linear relaxation of an ILP and a rounding procedure was applied to extract an approximate MAP state.

There is a large class of symmetry-aware algorithms for SRL models. Examples of such *lifted inference* algorithms include first-order variable elimination (FOVE) [78] and some of its extensions [61,45] making use of counting and aggregation parfactors. FOVE has also been adapted to solve MAP problems (FOVE-P) [19]. [2] introduced an approach for MAP inference that takes advantage of uniform assignments which are groups of random variables that have identical assignments in some MAP solution. Automorphism groups of graphical models were used to lift variational approximations of MAP inference [12]. [63] computed solutions to linear programs by reducing the LP problem to a pairwise MRF over Gaussians and applying lifted Gaussian belief propagation. Similar to the approach of [12] lifted linear programming can be used to compress LP relaxations [3] of the MAP ILP.

There are several lifted *marginal* inference approaches such as lifted message passing [85,43], variants of lifted knowledge compilation and theorem proving [89,33], and lifted MCMC and related symmetry-aware sampling approaches [65,68,90,66]. There are some generic parallel machine learning architectures such as GRAPHLAB [51] which could in principle be used for parallel MAP inference.

References

1. Albagli, S., Ben-Eliyahu-Zohary, R., Shimony, S.E.: Markov network based ontology matching. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1884–1889 (2009)
2. Apsel, U., Brafman, R.: Exploiting uniform assignments in first-order mpe. In: Proceedings of UAI, pp. 74–83 (2012)
3. Asano, T.: An improved analysis of goemans and williamson’s lp-relaxation for max sat. *Theoretical Computer Science* 354(3), 339–353 (2006)
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A nucleus for a web of open data. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
5. Bengio, Y., LeCun, Y.: Scaling learning algorithms towards AI. In: *Large Scale Kernel Machines*. MIT Press (2007)

6. Berners-Lee, T.: Linked data – design issues (2006), <http://www.w3.org/DesignIssues/LinkedData.html>
7. Bhattacharya, I., Getoor, L.: Entity resolution in graphs. In: Mining Graph Data. Wiley & Sons (2006)
8. Bizer, C., Heath, T., Berners-Lee, T.: Linked data – the story so far. *International Journal on Semantic Web and Information Systems* (2012)
9. Bödi, R., Herr, K., Joswig, M.: Algorithms for highly symmetric linear and integer programs. *Mathematical Programming* 137(1-2), 65–90 (2013)
10. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250 (2008)
11. Borgida, A.: On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence* 82(1-2), 353–367 (1996)
12. Bui, H.H., Huynh, T.N., Riedel, S.: Automorphism groups of graphical models and lifted variational inference. *CoRR*, abs/1207.4814 (2012)
13. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010), pp. 1306–1313 (2010)
14. Chapelle, O., Schölkopf, B., Zien, A. (eds.): *Semi-Supervised Learning*. MIT Press, Cambridge (2006)
15. Costa, P.C.G., Laskey, K.B.: Pr-owl: A framework for probabilistic ontologies. In: Bennett, B., Fellbaum, C. (eds.) *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS)*. *Frontiers in Artificial Intelligence and Applications*, pp. 237–249. IOS Press (2006)
16. Cruz, I.F., Stroe, C., Caci, M., Caimi, F., Palmonari, M., Antonelli, F.P., Keles, U.C.: Using AgreementMaker to Align Ontologies for OAEI 2010. In: *Proceedings of the 5th Workshop on Ontology Matching* (2010)
17. Cruz, I., Antonelli, F.P., Stroe, C.: Efficient selection of mappings and automatic quality-driven combination of matching methods. In: *Proceedings of the ISWC 2009 Workshop on Ontology Matching* (2009)
18. David, J., Guillet, F., Briand, H.: Matching directories and OWL ontologies with AROMA. In: *Proceedings of the 15th Conference on Information and Knowledge Management* (2006)
19. de Salvo Braz, R., Amir, E., Roth, D.: MPE and partial inversion in lifted probabilistic variable elimination. In: *Proceedings of AAAI*, pp. 1123–1130 (2006)
20. Diaconis, P.: Finite forms of de finetti’s theorem on exchangeability. *Synthese* 36(2), 271–281 (1977)
21. Ding, L., Kolari, P., Ding, Z., Avancha, S.: Bayesowl: Uncertainty modeling in semantic web ontologies. In: Ma, Z. (ed.) *Soft Computing in Ontologies and Semantic Web*. Springer (2006)
22. Domingos, P., Jain, D., Kok, S., Lowd, D., Poon, H., Richardson, M.: *Alchemy website* (2012), <http://alchemy.cs.washington.edu/> (last visit: November 22, 2012)
23. Etzioni, O., Banko, M., Soderland, S., Weld, D.S.: Open information extraction from the web. *Communications of the ACM* 51(12), 68–74 (2008)
24. Etzioni, O., Fader, A., Christensen, J., Soderland, S., Mausam, M.: Open information extraction: the second generation. In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pp. 3–10 (2011)

25. Euzenat, J., Hollink, A.F.L., Joslyn, C., Malaisé, V., Meilicke, C., Pane, A.N.J., Scharffe, F., Shvaiko, P., Spiliopoulos, V., Stuckenschmidt, H., Sváb-Zamazal, O., Svátek, V., dos Santos, C.T., Vouros, G.: Results of the ontology alignment evaluation initiative 2009. In: Proceedings of the ISWC 2009 workshop on Ontology Matching (2009)
26. Euzenat, J., Shvaiko, P.: *Ontology matching*. Springer (2007)
27. Euzenat, J., et al.: First Results of the Ontology Alignment Evaluation Initiative 2010. In: Proceedings of the 5th Workshop on Ontology Matching (2010)
28. Fellbaum, C.: *WordNet*. Springer (2010)
29. Fellegi, I., Sunter, A.: A theory for record linkage. *Journal of the American Statistical Association* 64(328), 1183–1210 (1969)
30. Ferrara, A., Lorusso, D., Montanelli, S., Varese, G.: Towards a Benchmark for Instance Matching. In: The 7th International Semantic Web Conference (2008)
31. Finetti, B.D.: *Probability, induction and statistics: the art of guessing*. Probability and mathematical statistics. Wiley (1972)
32. Giugno, R., Lukasiewicz, T.: P-shoq(d): A probabilistic extension of shoq(d) for probabilistic ontologies in the semantic web. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) *JELIA 2002*. LNCS (LNAI), vol. 2424, pp. 86–97. Springer, Heidelberg (2002)
33. Gogate, V., Domingos, P.: Probabilistic theorem proving. In: Proceedings of UAI, pp. 256–265 (2011)
34. Heinsohn, J.: A hybrid approach for modeling uncertainty in terminological logics. In: Kruse, R., Siegel, P. (eds.) *ECSQAU 1991 and ECSQARU 1991*. LNCS, vol. 548, pp. 198–205. Springer, Heidelberg (1991)
35. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. *Neural Computation* 18(7), 1527–1554 (2006)
36. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence* 194, 28–61 (2013)
37. Holí, M., Hyvönen, E.: Modeling uncertainty in semantic web taxonomies. In: Ma, Z. (ed.) *Soft Computing in Ontologies and Semantic Web*. Springer (2006)
38. Hu, W., Chen, J., Cheng, G., Qu, Y.: ObjectCoref & Falcon-AO: Results for OAEI 2010. In: Proceedings of the 5th International Ontology Matching Workshop (2010)
39. Huynh, T.N., Mooney, R.J.: Max-margin weight learning for markov logic networks. In: Proceedings of EMCL PKDD, pp. 564–579 (2009)
40. Jaeger, M.: Probabilistic reasoning in terminological logics. In: Doyle, J., Sandewall, E., Torasso, P. (eds.) *Proceedings of the 4th international Conference on Principles of Knowledge Representation and Reasoning*, pp. 305–316. Morgan Kaufmann (1994)
41. Jean-Marya, Y.R., Patrick Shironoshitaa, E., Kabuka, M.R.: Ontology matching with semantic verification. *Web Semantics* 7(3) (2009)
42. Kautz, H., Selman, B., Jiang, Y.: A general stochastic approach to solving problems with hard and soft constraints. *Satisfiability Problem: Theory and Applications* 17 (1997)
43. Kersting, K., Ahmadi, B., Natarajan, S.: Counting belief propagation. In: Proceedings of UAI, pp. 277–284 (2009)
44. Kersting, K.: Lifted probabilistic inference. In: Proceedings of the 20th European Conference on Artificial Intelligence, pp. 33–38 (2012)
45. Kisynski, J., Poole, D.: Lifted aggregation in directed first-order probabilistic models. In: Proceedings of IJCAI, pp. 1922–1929 (2009)

46. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
47. Koller, D., Levy, A., Pfeffer, A.: P-classic: A tractable probabilistic description logic. In: Proceedings of the 14th AAAI Conference on Artificial Intelligence (AAAI 1997), pp. 390–397 (1997)
48. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 282–289 (2001)
49. Laskey, K.B., Costa, P.C.G.: Of klingons and starships: Bayesian logic for the 23rd century. In: Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, pp. 346–353. AUAI Press (2005)
50. Levenshtein, V.I.: Binary codes capable of correcting deletions and insertions and reversals. In: Doklady Akademii Nauk SSSR, pp. 845–848 (1965)
51. Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., Hellerstein, J.M.: Graphlab: A new framework for parallel machine learning. In: Proceedings of UAI, pp. 340–349 (2010)
52. Manola, F., Miller, E.: RDF primer. Technical report, WWW Consortium (February 2004), <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>
53. Margot, F.: Exploiting orbits in symmetric ilp. Math. Program. 98(1-3), 3–21 (2003)
54. Margot, F.: Symmetry in integer linear programming. In: 50 Years of Integer Programming 1958–2008, pp. 647–686. Springer, Heidelberg (2010)
55. Meilicke, C., Stuckenschmidt, H.: Analyzing mapping extraction approaches. In: Proceedings of the Workshop on Ontology Matching, Busan, Korea (2007)
56. Meilicke, C., Stuckenschmidt, H.: An efficient method for computing alignment diagnoses. In: Polleres, A., Swift, T. (eds.) RR 2009. LNCS, vol. 5837, pp. 182–196. Springer, Heidelberg (2009)
57. Meilicke, C., Tamilin, A., Stuckenschmidt, H.: Repairing ontology mappings. In: Proceedings of the Conference on Artificial Intelligence, Vancouver, Canada, pp. 1408–1413 (2007)
58. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: Proceedings of ICDE, pp. 117–128 (2002)
59. Mendes, P.N., Jakob, M., Bizer, C.: Dbpedia: A multilingual cross-domain knowledge base. In: Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC), pp. 1813–1817 (2012)
60. Meza-Ruiz, I., Riedel, S.: Multilingual semantic role labelling with markov logic. In: Proceedings of the Conference on Computational Natural Language Learning, pp. 85–90 (2009)
61. Milch, B., Zettlemoyer, L.S., Kersting, K., Haimes, M., Kaelbling, L.P.: Lifted probabilistic inference with counting formulas. In: Proceedings of AAAI, pp. 1062–1068 (2008)
62. Mitchell, T.M., Betteridge, J., Carlson, A., Hruschka, E., Wang, R.: Populating the semantic web by macro-reading internet text. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 998–1002. Springer, Heidelberg (2009)
63. Mladenov, M., Ahmadi, B., Kersting, K.: Lifted linear programming. Journal of Machine Learning Research 22, 788–797 (2012)
64. Niepert, M.: A Delayed Column Generation Strategy for Exact k-Bounded MAP Inference in Markov Logic Networks. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (2010)

65. Niepert, M.: Markov chains on orbits of permutation groups. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pp. 624–633 (2012)
66. Niepert, M.: Symmetry-aware marginal density estimation. In: Proceedings of the Conference on Artificial Intelligence (AAAI) (2013)
67. Niepert, M., Meilicke, C., Stuckenschmidt, H.: A Probabilistic-Logical Framework for Ontology Matching. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence (2010)
68. Niepert, M., Meilicke, C., Stuckenschmidt, H.: Towards distributed mcmc inference in probabilistic knowledge bases. In: Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction, pp. 1–6 (2012)
69. Niepert, M., Noessner, J., Meilicke, C., Stuckenschmidt, H.: Probabilistic-logical web data integration. In: Polleres, A., d’Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 504–533. Springer, Heidelberg (2011)
70. Niepert, M., Noessner, J., Stuckenschmidt, H.: Log-Linear Description Logics. In: Proceedings of the International Joint Conference on Artificial Intelligence (2011)
71. Niu, F., Ré, C., Doan, A.H., Shavlik, J.: Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. Proceedings of the VLDB Endowment 4(6), 373–384 (2011)
72. Niu, F., Zhang, C., Ré, C., Shavlik, J.: Deepdive: Web-scale knowledge-base construction using statistical learning and inference. In: Second Int.l Workshop on Searching and Integrating New Web Data Sources (2012)
73. Noessner, J., Niepert, M., Stuckenschmidt, H.: Coherent top-k ontology alignment for OWL EL. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 415–427. Springer, Heidelberg (2011)
74. Noessner, J., Niepert, M., Stuckenschmidt, H.: RockIt: Exploiting Parallelism and Symmetry for MAP Inference in Statistical Relational Models. In: Proceedings of the Conference on Artificial Intelligence (AAAI) (2013)
75. Noessner, J., Niepert, M., Meilicke, C., Stuckenschmidt, H.: Leveraging Terminological Structure for Object Reconciliation. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part II. LNCS, vol. 6089, pp. 334–348. Springer, Heidelberg (2010)
76. Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Orbital branching. *Math. Program.* 126(1), 147–178 (2011)
77. Pan, R., Ding, Z., Yu, Y., Peng, Y.: A bayesian network approach to ontology mapping. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 563–577. Springer, Heidelberg (2005)
78. Poole, D.: First-order probabilistic inference. In: Proceedings of IJCAI, pp. 985–991 (2003)
79. Poon, H., Domingos, P.: Sum-product networks: A new deep architecture. In: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, pp. 337–346 (2011)
80. Richardson, M., Domingos, P.: Markov logic networks. *Machine Learning* 62(1-2) (2006)
81. Riedel, S.: Improving the accuracy and efficiency of map inference for markov logic. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (2008)
82. Saïs, F., Pernelle, N., Rousset, M.-C.: Combining a logical and a numerical method for data reconciliation. In: Spaccapietra, S. (ed.) *Journal on Data Semantics XII*. LNCS, vol. 5480, pp. 66–94. Springer, Heidelberg (2009)

83. Schoenmackers, S., Etzioni, O., Weld, D.S., Davis, J.: Learning first-order horn clauses from web text. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pp. 1088–1098 (2010)
84. Shavlik, J., Natarajan, S.: Speeding up inference in markov logic networks by preprocessing to reduce the size of the resulting grounded network. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence, pp. 1951–1956 (2009)
85. Singla, P., Domingos, P.: Lifted first-order belief propagation. In: Proceedings of AAAI, pp. 1094–1099 (2008)
86. Stoermer, H., Rassadko, N.: Results of OKKAM feature based entity matching algorithm for instance matching contest of OAEI 2009. In: Proceedings of the ISWC 2009 Workshop on Ontology Matching (2009)
87. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, pp. 697–706 (2007)
88. Tsarkov, D., Riazanov, A., Bechhofer, S., Horrocks, I.: Using vampire to reason with OWL. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 471–485. Springer, Heidelberg (2004)
89. Van den Broeck, G.: On the completeness of first-order knowledge compilation for lifted probabilistic inference. In: Proceedings of NIPS, pp. 1386–1394 (2011)
90. Venugopal, D., Gogate, V.: On lifting the gibbs sampling algorithm. In: Proceedings of Neural Information Processing Systems (NIPS), pp. 1664–1672 (2012)
91. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Silk - a link discovery framework for the web of data. In: Proceedings of the WWW 2009 Workshop on Linked Data on the Web (LDOW) (2009)
92. Wu, F., Weld, D.S.: Automatically refining the wikipedia infobox ontology. In: Proceeding of the International World Wide Web Conference, pp. 635–644 (2008)
93. Yang, Y., Calmet, J.: Ontobayes: An ontology-driven uncertainty model. In: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC 2005), pp. 457–463 (2005)
94. Yelland, P.M.: An alternative combination of bayesian networks and description logics. In: Cohn, A., Giunchiglia, F., Selman, B. (eds.) Proceedings of of the 7th International Conference on Knowledge Representation (KR 2000), pp. 225–234. Morgan Kaufman (2000)
95. Zhang, X., Zhong, Q., Shi, F., Li, J., Tang, J.: RiMOM results for OAEI 2009. In: Proceedings of the ISWC 2009 Workshop on Ontology Matching (2009)

Author Index

Adams, Benjamin	230	Ngonga Ngomo, Axel-Cyrille	1
Auer, Sören	1	Niepert, Mathias	251
Delbru, Renaud	91	Polleres, Axel	91
Faber, Wolfgang	162	Rodríguez-Muro, Mariano	194
Hogan, Aidan	91	Scheider, Simon	230
Janowicz, Krzysztof	230	Turhan, Anni-Yasmin	150
Kontchakov, Roman	194	Umbrich, Jürgen	91
Lehmann, Jens	1	Zakharyashev, Michael	194
		Zaveri, Amrapali	1