Managing and Using Provenance in the Semantic Web

Renata Queiroz Dividino

Institute for Web Science and Technologies
University of Koblenz-Landau

June 2017

Vom Promotionsausschuss des Fachbereichs 4: Informatik der Universität Koblenz-Landau zur Verleihung des akademischen Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation.

Datum der wissenschaftlichen Aussprache:	14 Dezember 2016
Vorsitz des Promotionsausschusses:	Prof. Dr. Hannes Frey
Gutachter:	Prof. Dr. Steffen Staab
Gutachter:	Prof. Dr. York Sure-Vetter
Gutachter:	Prof. Dr. Luc Moreau

Abstract

The Web contains some extremely valuable information; however, often poor quality, inaccurate, irrelevant or fraudulent information can also be found. With the increasing amount of data available, it is becoming more and more difficult to distinguish truth from speculation on the Web. One of the most, if not the most, important criterion used to evaluate data credibility is the information source, i.e., the data origin. Trust in the information source is a valuable currency users have to evaluate such data. Data popularity, recency (or the time of validity), reliability, or vagueness ascribed to the data may also help users to judge the validity and appropriateness of information sources. We call this knowledge derived from the data the *provenance* of the data.

Provenance is an important aspect of the Web. It is essential in identifying the suitability, veracity, and reliability of information, and in deciding whether information is to be trusted, reused, or even integrated with other information sources. Therefore, models and frameworks for representing, managing, and using provenance in the realm of Semantic Web technologies and applications are critically required.

This thesis highlights the benefits of the use of provenance in different Web applications and scenarios. In particular, it presents management frameworks for querying and reasoning in the Semantic Web with provenance, and presents a collection of Semantic Web tools that explore provenance information when ranking and updating caches of Web data.

To begin, this thesis discusses a highly flexible and generic approach to the treatment of provenance when querying RDF datasets. The approach re-uses existing RDF modeling possibilities in order to represent provenance. It extends SPARQL query processing in such a way that given a SPARQL query for data, one may request provenance without modifying it. The use of provenance within SPARQL queries helps users to understand how RDF facts are derived, i.e., it describes the data and the operations used to produce the derived facts.

Turning to more expressive Semantic Web data models, an optimized algorithm for reasoning and debugging OWL ontologies with provenance is presented. Typical reasoning tasks over an expressive Description Logic (e.g., using tableau methods to perform consistency checking, instance checking, satisfiability checking, and so on) are in the worst case doubly-exponential, and in practice are often likewise very expensive. With the algorithm described in this thesis, however, one can efficiently reason in OWL ontologies with provenance, i.e., provenance is efficiently combined and propagated within the reasoning process. Users can use the derived provenance information to judge the reliability of inferences and to find errors in the ontology.

Next, this thesis tackles the problem of providing to Web users the right content at the right time. The challenge is to efficiently rank a stream of messages based on user preferences. Provenance is used to represent preferences, i.e., the user defines his preferences over the messages' popularity, recency, etc. This information is then aggregated to obtain a joint ranking. The aggregation problem is related to the problem of preference aggregation in Social Choice Theory. The traditional problem formulation of preference aggregation assumes a

fixed set of preference orders and a fixed set of domain elements (e.g. messages). This work, however, investigates how an aggregated preference order has to be updated when the domain is dynamic, i.e., the aggregation approach ranks messages 'on the fly' as the message passes through the system. Consequently, this thesis presents computational approaches for online preference aggregation that handle the dynamic setting more efficiently than standard ones.

Lastly, this thesis addresses the scenario of caching data from the Linked Open Data (LOD) cloud. Data on the LOD cloud changes frequently and applications relying on that data – by pre-fetching data from the Web and storing local copies of it in a cache – need to continually update their caches. In order to make best use of the resources (e.g., network bandwidth for fetching data, and computation time) available, it is vital to choose a good strategy to know when to fetch data from which data source.

A strategy to cope with data changes is to check for provenance. Provenance information delivered by LOD sources can denote when the resource on the Web has been changed last. Linked Data applications can benefit from this piece of information since simply checking on it may help users decide which sources need to be updated. For this purpose, this work describes an investigation of the availability and reliability of provenance information in the Linked Data sources.

Another strategy for capturing data changes is to exploit provenance in a time-dependent function. Such a function should measure the frequency of the changes of LOD sources. This work describes, therefore, an approach to the analysis of data dynamics, i.e., the analysis of the change behavior of Linked Data sources over time, followed by the investigation of different scheduling update strategies to keep local LOD caches up-to-date.

This thesis aims to prove the importance and benefits of the use of provenance in different Web applications and scenarios. The flexibility of the approaches presented, combined with their high scalability, make this thesis a possible building block for the Semantic Web proof layer cake - the layer of provenance knowledge.

Zusammenfassung

Das Internet enthalt äußerst nutzbringende Informationen, allerdings stößt der Nutzer häufig auch auf unwichtige Daten oder solche, die ungenau, von geringer Qualität oder sogar mit betrügerischer Absicht verfälscht worden sind. Mit der stetig wachsenden Menge an Daten, die im Internet verfügbar ist, wird es zunehmend schwieriger zwischen Wahrheit und Spekulation zu unterscheiden. Eines der wichtigsten, wenn nicht das wichtigste Kriterium zur Analyse der Glaubwürdigkeit der Daten ist die Informationsquelle oder in anderen Worten, die Herkunft der Daten. Das Vertrauen in die Informationsquelle ist für den Nutzer entscheidend zur Beurteilung der Information. Darüber hinaus können ihm die Popularität, Aktualität, die Zuverlässigkeit der Daten als Anhaltspunkte für die Eignung und Aussagekraft der Informationsquelle dienen. Die oben genannten Eigenschaften einer Datenquelle werden als "Provenance" bezeichnet.

Die Betrachtung der "Provenance" der Information gewinnt zunehmend an Bedeutung im Semantic Web. "Provenance" ist entscheidend für die Bestimmung der Eignung, des Wahrheitsgehaltes und der Zuverlässigkeit der Information und damit für die Beurteilung, ob der Information vertraut werden, sie weiterverbreitet oder sogar mit anderen Informationsquellen verknüpft werden kann. Demzufolge ist die Entwicklung von Modellen und Strukturen, die computergestützte Repräsentation, Verwaltung und Nutzung der "Provenance" ermöglichen, sehr wichtig.

Diese Doktorarbeit untersucht und hebt die Vorteile der Nutzung der "Provenance" der Daten in verschiedenen Webanwendungen und Szenarien hervor. Insbesondere werden Rahmenkonzepte zum Abfragen und zum Schlussfolgern mit "Provenance" im Semantic Web und eine Reihe von Semantic Webtools präsentiert, die die "Provenance" von Daten während der Klassifizierung und der Aktualisierung des Webdatencaches untersucht.

Zunächst wird ein flexibler Ansatz zur Verarbeitung von "Provenance" bei dem Abfragen von RDF Datensätzen diskutiert. Dieser Ansatz nutzt existierende RDF Modellierungsmöglichkeiten um "Provenance" darzustellen. Er erweitert die SPARQL Abfrageverarbeitung derart, dass es möglich ist "Provenance" zu erfragen ohne die Abfrage zu verändern. Die Nutzung von "Provenance" in SPARQL Abfragen hilft dem Nutzer zu verstehen, wie RDF Fakten hergeleitet wurden, d.h. es beschreibt die Daten und Verarbeitungsschritte, die benutzt wurden die Fakten herzuleiten.

Im weiteren Verlauf beschäftigt sich diese Arbeit mit einer ausdrucksstärkeren Modellierungssprache und präsentiert einen optimierten Algorithmus für das Schlussfolgern und Fehlersuchen mit Hilfe von "Provenance" in OWL Ontologien. Typische Schlussfolgerungsaufgaben mit einer ausdrucksstarken Beschreibungslogik (z.B. Tableau Methoden nutzend um Konsistenzprüfungen durchzuführen, Instanzenprüfung, Erfüllbarkeitsprüfung, etc.) sind im ungünstigsten Fall doppelt exponentiell und in der Praxis oft außerordentlich teuer. Der in dieser Dissertation erarbeitete und beschriebene Algorithmus erlaubt effiziente Schlussfolgerungsunterstützung mit "Provenance" in OWL Ontologien, d.h. "Provenance" wird im Schlussfolgerungsprozess effizient kombiniert und verbreitet. Der Nutzer kann die hergeleiteten "Pro-

venance" Informationen verwenden um die Zuverlässigkeit der Inferenzen zu beurteilen und Fehler in der Ontologie zu finden.

In einem weiteren Schritt wird das Problem den Internetnutzer mit dem richtigen Inhalt zum richtigen Zeitpunkt zu versorgen untersucht. Die Herausforderung ist hierbei, den Nutzer laufend und effizient mit den wichtigsten Nachrichten im Einklang mit dessen Vorlieben zu versorgen. "Provenance" wird genutzt um Vorlieben darzustellen, d.h. der Nutzer definiert seine Vorlieben über die Popularität und Neuigkeit von Nachrichten. Diese Daten werden dann aggregiert und klassifiziert. Diese Aggregationsproblemstellung basiert auf Präferenzaggregation in der "Social Choice Theory". Infolgedessen präsentiert diese Dissertation Ansätze für online Präferenzaggregation, die das dynamische Setting effizienter als die Standardansätze handhabt.

Schließlich beschäftigt sich diese Arbeit mit dem Szenario des Cachens von Daten aus der Linked Open Data (LOD) Cloud. Die Daten in der LOD Cloud ändern sich häufig und Anwendungen, die von diesen Daten abhängen (beim vorherigen Abrufen von Daten aus dem Internet und dem lokalen Speichern von Kopien im Cache), müssen laufend ihre Caches aktualisieren. Eine gute Strategie, die bestimmt, wann Daten von welchen Datenquellen vorher abgerufen werden, zu wählen, ist wesentlich um die Ressourcen (z.B. Netzwerkbandbreite und Berechnungskapazitäten) bestmöglich zu nutzen. Eine Strategie ist die Überprüfung der "Provenance" Information, die LOD Quellen zur Verfügung stellen. "Provenance" kann anzeigen, wann die Ressource im Internet zuletzt geändert worden ist. Linked Data Anwendungen können von dieser Information profitieren, da eine einfache Überprüfung der "Provenance" die Entscheidungsfindung des Nutzers, welche Datenquellen aktualisiert werden müssen, unterstützen kann. Zu diesem Zweck enthält diese Dissertation eine Untersuchung der Verfügbarkeit und Zuverlässigkeit von "Provenance" Informationen in den Linked Data Quellen. Eine weitere erarbeitete Strategie ist, die Datenveränderungen mit zeitabhängigem Maß, das die Frequenz der Änderung der LOD Quellen erfasst, zu messen. Deshalb beschreibt diese Arbeit einen Ansatz der Analyse von Datendynamik, d.h. der Analyse des Änderungsverhaltens von Linked Data Quellen über der Zeit. Auf dieser Analyse baut die Untersuchung von unterschiedlichen Strategien zur Aktualisierungsplanung, um den lokalen LOD Cache auf einem aktuellen Stand zu halten, auf.

Diese Dissertation hat das Ziel die Wichtigkeit und den Vorteil der Nutzung von "Provenance" in unterschiedlichen Webanwendungen und Szenarien zu belegen. Die Flexibilität der erarbeiteten Ansätze in Kombination mit ihrer hohen Skalierbarkeit macht diese Dissertation zu einem möglichen Baustein für eine Semantic Web Nachweisebene – die Ebene des "Provenance"-Wissens.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Professor Steffen Staab, whose love of science has been an inspiration to me. Professor Staab's expertise, understanding, academic guidance, and support made it possible for me to work on a topic that is of great interest to me.

I would like to thank York Sure-Vetter and Daniel Oberle who woke my interest in science even before I had started my PhD program.

To my co-authors in various scientific publications, Sergej Sizov, Simon Schenck, Gerd Gröner, Matthias Thimm, Stefan Scheglmann, Thomas Gottron, Ansgar Scherp and Steffen Staab: you have provided me with great insights into the research topics covered in this dissertation and I am grateful for all the long, productive and friendly discussions we had. Without your cooperation and valuable feedback, this dissertation would not have been possible.

To Thomas Gottron, Thomas Franz, Christoph Ringelstein, Julia Perl, and Gerd Gröner who encouraged me throughout my work. Thanks for your support in the different stages of my research.

I thank Olaf Görlitz, Fernando Silva Parreiras, Antje Schultz, Cristina Sarasua, Tina Walter, Jerome Küniges, Christoph Kling, Kerstin Falkowski, Sabine Hülstrunk and Silke Werger. You are incredible colleagues, and I am grateful for the great moments we have spent together.

My gratitude goes to all those who talked and listened to me over the years at conferences, workshops and seminars.

Lastly, I'd like to thank my parents, my husband, and my children for supporting and encouraging me over many years. This dissertation is dedicated to you.

Contents

List of Figures

List of Tables

Part I.

Fundamentals

1. Introduction

1.1. Motivation

The Web is a global information space and has become in recent years one of the most important communication platforms of our daily life. Today's digital economy is developing rapidly and the information being spread on the Web has a great impact on the way we live and do business.

Indeed, the Web contains some extremely valuable information; however, often poor quality, inaccurate, irrelevant, or fraudulent information can be found. This is due to its liberal nature; anyone is allowed — and without any control — to generate and spread information.

In particular, popular online social sites such as Twitter and Facebook have become important news and marketing channels as people and entities are willing to take part in opinion-building processes. In such channels, misinformation is rapidly spread, intentionally or unintentionally, which can lead to undesirable effects such as discrediting and conflicting information or supporting false conclusions.

In 2014, Farida Viz expressed her concerns on misinformation being spread on Web in the `theguardian.com`[1] and described two real cases of it: (1) near to the anniversary of the 2013 Boston Marathon bombings, information posted on Reddit led to *the New York Post* printing images of two suspects on its front page, who ultimately had nothing to do with the bombings. (2) Just following the disappearance of Malaysia Airlines flight MH370 in March 2014, NBC news highlighted various false reports having been spread on social media alleging that the plane had made a safe landing.

Due to these and many other cases where inaccurate information has been spread over the Web, information seekers are experiencing difficulties in filtering correct information from contradictory and/or questionable sources. With the increasing amount of data available on the Web, it is becoming more and more difficult to distinguish truth from speculations.

One of the most valuable tools in verifying data authenticity and credibility is knowledge of its information source, that is, the data origin. Trust in the information source is a valuable currency users have to assess the data [Adler and de Alfaro, 2007a, Flöck and Acosta, 2014]. Furthermore, data popularity, the recency (or the time of validity), the credibility, or the vagueness ascribed to the data may also help users to judge the validity and appropriateness of information sources. We call this knowledge derived from the data the *provenance* of the data.

Provenance is an important piece of information used to identify the validity of information, and to decide whether information is to be trusted, reused or even integrated with other sources. Provenance information has been used for many years in database and scientific workflow management systems [Buneman et al., 2006, Simmhan et al., 2005], and more recently to access the value of the data on the Web of Data [Cappiello, 2015, Moreau, 2010].

[1] Article: To tackle the spread of misinformation online we must first understand it `http://www.theguardian.com/commentisfree/2014/apr/24/tackle-spread-misinformation-online`

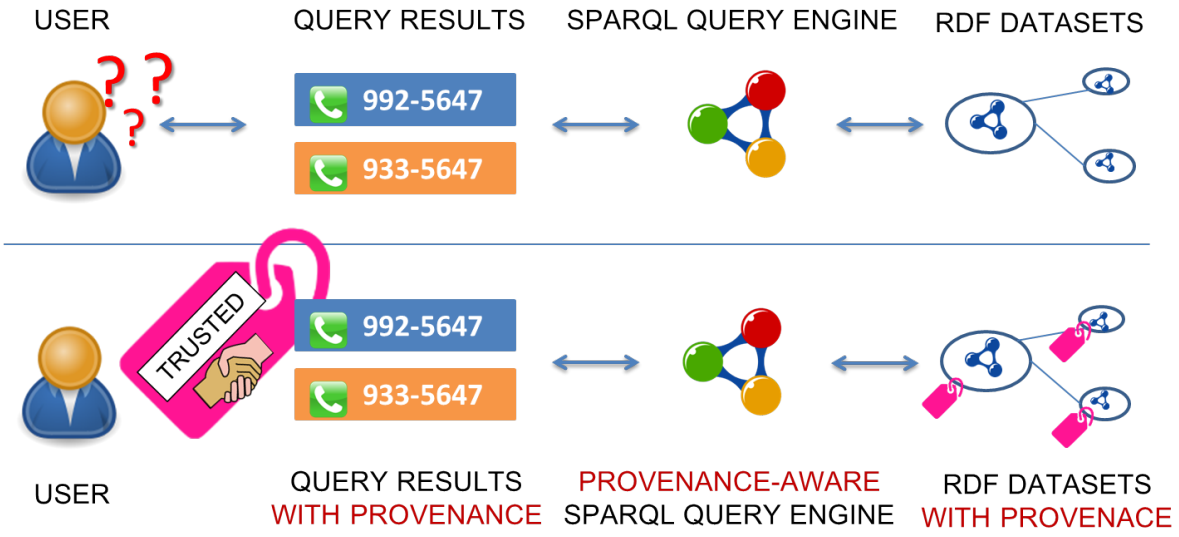


Figure 1.1.: When querying the Web of data we are confronted with a large number of diverse data sources of varying quality. Provenance helps in identifying, selecting, and reasoning with high quality data.

The Web of Data, as an interlinked data space, provides an environment to exploit the Web as a platform for data and information integration as well as for searching, querying, and reasoning. Semantic Web technologies promote common data formats and exchange protocols on the Web such as the Resource Description Framework (RDF), the SPARQL Query Language (SPARQL) and Web Ontology Language (OWL). Typically, many data sources in the Web provide points of access for RDF data through data portals (i.e. SPARQL query endpoints) that allow for selecting and re-using data (see Figure 1.1). RDF plays the role of a common model, as data can easily be integrated and combined with other data published on the Web, which is basically done by establishing links between Web resources.

In such an open and interlinked data space where querying goes beyond simple keyword searches and reasoning beyond data mining, provenance seems to be not only an important but an essential piece of information to establish trust between data providers and data consumers. Provenance is an important aspect of the Web, and therefore models and frameworks for representing, managing, and using provenance in the realm of Semantic Web technologies and applications are critically required.

This thesis will examine the benefits of using provenance in different Web applications and scenarios. In particular, it presents management frameworks for querying and reasoning with provenance in the Semantic Web, and presents a collection of Semantic Web tools that explore provenance information when ranking and updating caches of Web data.

Provenance Management in the Semantic Web:

When querying and reasoning with data on the Web, we are faced with a highly variable quality of information. With the increasing amount of data available on the Web and more sophisticated processing through query and reasoning engines, users encounter challenging questions linked to provenance about the data such as:

- *Where* is this data from?
- *Who* provided the data?
- *When* was this data provided?
- Was the provider certain about the *truth* of this data?
- Was the data *believed by others*?

Therefore, when querying and reasoning in the Semantic Web, techniques to extract the relevant information from Web data should include ways to investigate the value of information. *Provenance* provides knowledge that can be used to quantify this value. For instance, the use of provenance within SPARQL queries may help users to understand how an RDF fact has been derived, that is, provenance describes the data and the operations used to produce the derived fact as illustrated in Figure 1.1. Analogously, the use of provenance within OWL reasoning may help users to understand how an OWL axiom is entailed.

Furthermore, provenance may be used to tackle the problem of debugging OWL ontologies as it allows users to use the derived provenance information to judge not only the reliability of inferences but also to find errors in the ontology.

Querying for and reasoning with data and provenance on the Semantic Web requires a highly flexible and generic approach to the treatment of provenance that is able to adapt to its many dimensions and adjust to new dimensions when the need arises. These approaches will be addressed in this thesis.

Using Provenance in Semantic Web Applications

Nowadays, the Web is an almost continuous flow of information. Data is constantly being produced, shared, and consumed by a diversity of stakeholders. With so much data on the Web, users want to be assure not to miss anything of interest. Methods and algorithms dealing with Web data require an understanding of the data dynamicity.

For instance, social media sites rely on news feed ranking algorithms. The goal of such algorithms is to deliver the right content to the user at the right time, i.e., they determine which messages are important for the user (based on their preferences), and from those, they decide which one should be shown first. Given the assumption that messages come 'on the fly', the challenge is to efficiently process the user's preferences to provide the most relevant messages first. Such "dynamic" preference aggregation algorithm is addressed in this work, where users define their preference over provenance, i. e., the messages' popularity, over their recency, etc.

Further, quite often, Web applications pre-fetch data from the Web and store local copies of it in a cache for faster access at runtime. As data evolves over time, local copies of such data need to be updated from time to time to ensure the quality (or freshness) of such copies.

In order to grab the changes of the Web sources and make best use of the resources available, it is vital to choose a good scheduling strategy to know when to fetch data from a given data source. So far, there is no work addressing strategies to efficiently keep local copies of Web sources up-to-date. Nevertheless, the effectiveness of well-known update scheduling strategies for maintaining indices of web documents, such as the PageRank

algorithm [Page et al., 1999], could be evaluated for updating local copies of Web sources. Provenance may be used within such strategies to cope with cache updates since a simple check on it may support their decision process of determining which sources need to be updated. A novel and different strategy to perform cache updates could be done via change behavior analysis. A metric which captures the change behavior of a Web source over time, i. e., the intensity of how the data evolved in this period, is then necessary and therefore addressed in this work.

1.2. Research Questions

In order to show the benefits of the use of provenance information for different Semantic Web applications and scenarios, the following research questions are to be addressed. These research questions are individually tackled in the chapters of this thesis as follows.

PART II: Provenance Management in the Semantic Web

Provenance of data can be represented in existing standard Semantic Web data modelling and query languages such as RDF, OWL and SPARQL. However, we must distinguish the notation of such languages with only *implicit* notation of provenance, but no semantic consequences specifically due to this provenance, from a formally extended model of RDF, OWL, and SPARQL with *explicit* notation of provenance. Such provenance model should ideally retain upward compatibility with existing usage of the language and corresponding tools and methods. In the following, we tackle the research challenges we faced with the development of a framework for querying for data and provenance, as well as for reasoning with data and provenance in the Semantic Web.

Chapter 3: Querying RDF Datasets with Provenance

Querying for RDF data and provenance with SPARQL requires a more sophisticated query processing that adapts to the treatment of provenance. RDF plays the role of a common model, as data can easily be integrated and combined with other data published on the Web, which is basically done by establishing links between Web resources. Provenance should then be embedded in RDF in such a way that it retains upward compatibility with existing language usage and corresponding tools and methods, being a major concern for Semantic Web approaches. These requirements lead us to the first research question:

RQ 1.I *Can RDF be used to represent provenance in its different dimensions e. g., source, certainty, and timestamp?*

Further, the SPARQL query language enables querying in interlinked data space that goes beyond simple keyword searches. SPARQL mechanisms, however, do not include techniques to find the relevant information out of the Web data. Therefore query engines require mechanisms to track provenance in its many dimensions when exploring data, guiding us to our next research question:

RQ 1.II *Does the treatment of provenance within the SPARQL query language allow for changing the existing SPARQL semantics (it is thus not supported by existing query engines)?*

Moreover, in general, provenance information can grow to be larger than the data it describes if the data is fine-grained and provenance information rich. Hence, the manner in which the provenance is propagated along the computation is crucial to its scalability, which leads to the following research question:

RQ 1.III *Does the exploitation of provenance lead to computation overhead?*

Chapter 4: Reasoning and Debugging Evolving OWL Ontologies with Provenance

Provenance can be used within reasoning to help users to judge the reliability of inferences and to find errors in the ontologies. OWL is, however, heavily based on Description Logic (DL), i.e. its model-theoretic semantics is compatible with the semantics of Description Logics. Typical reasoning tasks over an expressive DL (e. g. using tableau methods to perform consistency checking, instance checking, satisfiability checking, etc. [Baader et al., 2003, Rudolph, 2011]) are in the worst case doubly-exponential, and in practice often likewise very expensive.

In [Schenk et al., 2011] Schenk et al. proposes a generic formalization of provenance (in its multiple dimensions) and a debugging framework for provenance in OWL based on *pinpointing*. Pinpointing means identifying relevant axioms where an axiom is defined as relevant if an incoherent ontology becomes coherent once this axiom is removed, or if a previously unsatisfiable concept turns satisfiable.

Pinpoiting as well as standard algorithms for debugging ontologies poorly support the user in answering provenance questions and also require expensive reasoning. For these reasons, even though the exploitation of provenance helps users to find undesired inferences and inconsistencies in evolving ontologies, such algorithms are not applicable for expressive and large-scale real-world ontologies. These challenges lead to the following research question:

RQ 1.IV *Does the exploitation of provenance when reasoning and debugging OWL ontologies lead to computation overhead?*

PART III: Using Provenance in Semantic Web Applications

Dealing with the data dynamicity on the Web is crucial as it has a great impact on the way people live and do business. Methods and algorithms dealing with Web data require an understanding of the data changes over time. Provenance information and models as studied and proposed in this thesis may be used within such algorithms, methods and strategies to support the process of data quality and change behavior assessment of dynamic information sources. In the following, we discuss the research challenges we experiencing when using provenance information to filter relevant information from the Web to Web consumers, and to verify change behavior and metadata conformance of such dynamic information sources.

Chapter 5: An Efficient Provenance-Aware News Feed Ranking Algorithm via Preference Aggregation

Due to the continuous flow of information produced and available on the Web, the challenge is to efficiently process the user's preferences to provide the most relevant and correct information first.

The traditional problem of preference aggregation assumes a fixed set of preference orders and a fixed set of domain elements (static setting). Using the standard aggregation algorithms, whenever a new element is added to the domain, a new aggregation has to be re-built. This operation requires expensive computation and cannot be adopted in real use case scenarios. Therefore, it is essential to investigate how preference aggregation methods can be modified in order to assure efficiency in a dynamic setting as pointed out in the next research question:

RQ 2.I *Can the preference aggregation problem be efficiently solved in a dynamic setting?*

Chapter 6: Managing Data Changes in the Linked Open Data Sources

As Web data evolves over time, that is, as RDF Graphs change when information is added and removed, local copies of Web sources need to be updated from time to time to ensure the quality of such copies.

Mainly, the naïve approach for detecting changes of a resource in a LOD source consists of downloading two (arbitrarily-sized) RDF descriptions of that resource and further comparing them [Käfer et al., 2013, Völkel and Groza, 2006, Zeginis et al., 2011]. As these descriptions can be of significant size, Linked Data applications would benefit if Linked Data servers provided provenance information. A simple check on this provenance could support their decision process of determining which sources need to be updated.

However, it is crucial that Linked Data servers provide correct and valid provenance values; otherwise, they are of no use in any practical application:

RQ 2.II *Can the provenance delivered by Linked Data servers support applications for detecting changes of the resources in a LOD source?*

Further, when considering the change analysis of a dataset over a period of time, state-of-the-art metrics can be used. State-of-the-art metrics for change detection of RDF datasets [Ding and Finin, 2006, Dividino et al., 2013, Käfer et al., 2013] mainly quantify changes between any two datasets. Nevertheless, such metrics do not include a mechanism to exploit the dynamics of a dataset , i.e. to consider a time interval described by more than two points in time. A time-dependent function could capture the frequency degree and regularity of the changes of the data, which leads to the following research question:

RQ 2.III *Does the consideration of changes within a time interval improve change analysis?*

Often applications relying on that dynamic data (by pre-fetching data from the Web and storing local copies of it in a cache) need to continually update their caches. Instead of continually visiting all of the LOD sources at brief intervals, a good scheduling strategy is essential to know when to fetch data of which data source in order to grab most of the changes.

To the best of our knowledge, there is no work addressing strategies to efficiently keep local copies of LOD source up-to-date. In the literature, however, many update scheduling strategies for maintaining indices of web documents, such as the

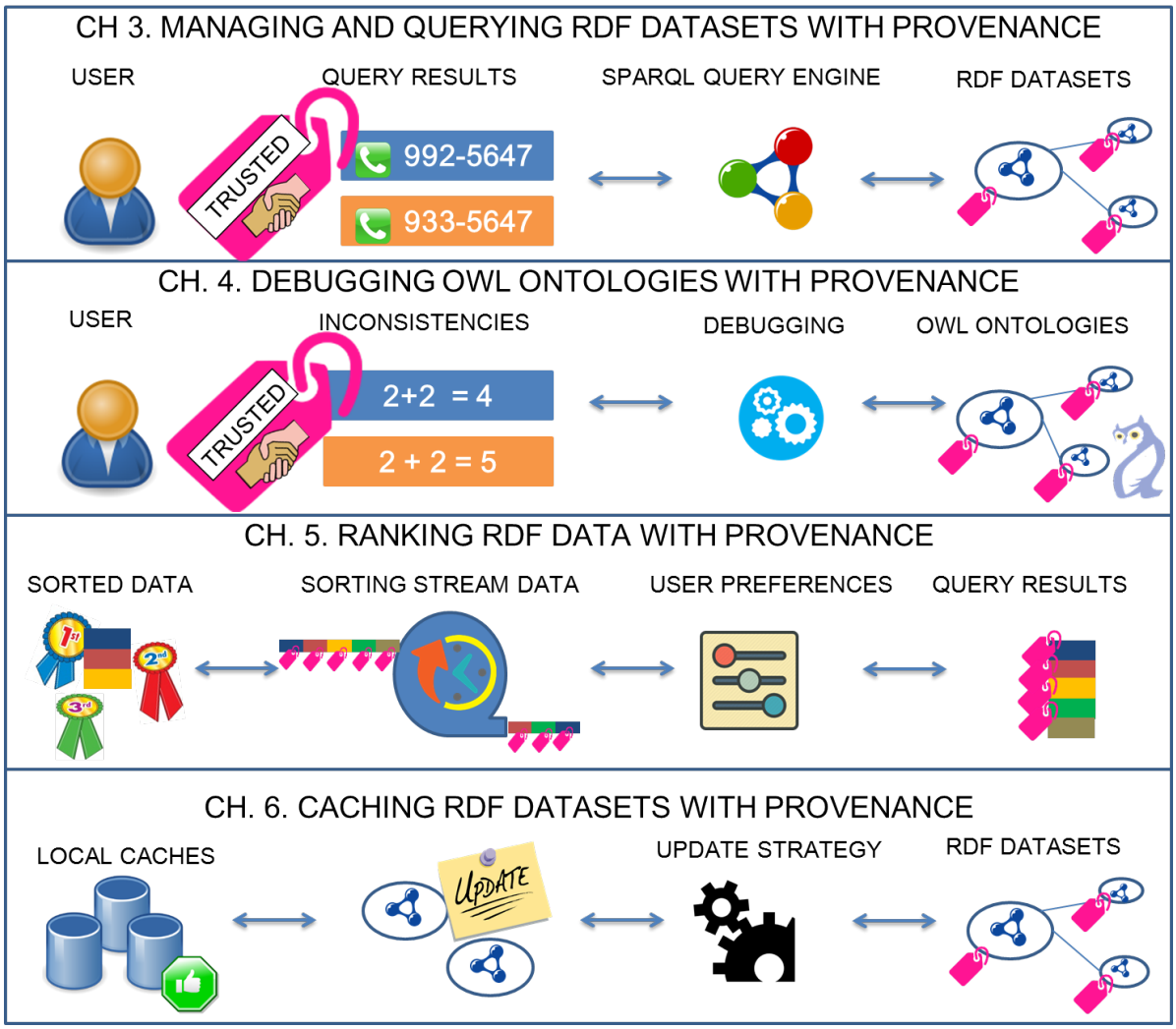


Figure 1.2.: This thesis shows the benefits of using provenance information in different Semantic Web applications and scenarios. This figure illustrates the Semantic Web tools presented and the chapters where they are discussed.

PageRank algorithm [Page et al., 1999], and metrics for quantifying the changes of LOD sources have been proposed. To determine their adequacy and effectiveness in the context of updating local copies of LOD sources, as pointed out in the next research question, an extensive evaluation is required.

RQ 2.IV *Which is the most adequate update scheduling strategy to manage caching copies of the LOD sources?*

1.3. Thesis Outline

The thesis consists of three parts. The first part describes the motivation and foundations of data provenance and the Semantic Web. The second presents a framework for provenance management in the Semantic Web. The last part presents different Semantic Web tools and scenarios that benefit from the use of provenance information. An outline of this thesis is described in Figure 1.2.

PART I: Foundations

Chapter 1: Introduction This chapter describes the motivation for using provenance information when querying, filtering, debugging or repairing, and updating caches of Web data. It presents the thesis' structure and lists the research publications.

Chapter 2: Foundations This chapter presents the fundamental concepts of Provenance and the Semantic Web technologies and analyzes state-of-the art approaches.

PART II: Provenance Management in the Semantic Web

Chapter 3: Querying RDF Datasets with Provenance

Querying for data and provenance in the Semantic Web requires a highly flexible and generic approach to the treatment of provenance that is able to adapt to its many dimensions and is open to accommodate new dimensions when the need arises. Such a principled, original framework is worked out in this chapter. This approach re-uses existing RDF modeling possibilities in order to represent provenance.

Based on the modeling proposed by Schüler et al. [Schueler et al., 2008], provenance is modeled in existing RDF structures by embedding a slightly more expressive language, which is called RDF$^+$, into RDF. This embedding implies that the different provenance dimensions, e. g. source, certainty, and timestamp, may be defined using RDF snippets in their literal sense, and further their intended semantics in order to facilitate query processing with complex expressions and pattern combinations. (see RQ 1.I - *Can RDF can be used to represent provenance on its different dimensions, e. g. source, certainty, and timestamp?*)

Further, SPARQL query processing is extended in such a way that given a SPARQL query for data, one may request provenance without modifying the query properly. This extension allows the user to expand a given conventional SPARQL query by a keyword for provenance, triggering the construction of provenance by the query processor. The evaluation of SPARQL with provenance provides the user additional access to valuable provenance that can be used for relevance ranking, conflict resolution, or other applications in connection with retrieved knowledge. Nevertheless, standard evaluation of SPARQL queries is still fully supported. This is an important advantage for compatibility with existing applications and interfaces (see RQ 1.II *Does the treatment of provenance within the SPARQL query language afford changing the existing SPARQL semantics, leaving it unsupported by existing query engines?*).

This thesis outlines how the construction of annotations influences the complexity of the decision problem related to SPARQL. Hence, it shows the time complexity

and space complexity analysis of evaluating SPARQL with provenance (see RQ 1.III *Does the exploitation of provenance lead to computation overhead?*).

Chapter 4: Reasoning and Debugging Evolving OWL Ontologies with Provenance

This chapter presents an algorithm for reasoning and tracking undesired inferences and inconsistencies using provenance when answering queries upon evolving ontologies in the Semantic Web. Debugging inconsistent ontologies both diagnoses and repairs them. When provenance information is attached to the diagnosis, it can help users to understand why inconsistencies exist, emphasizing who created them, from what location, and when they were created. Provenance information can thus guide repairs.

With this algorithm one can efficiently reason in OWL ontologies with provenance. This debugging approach with provenance supports users in coping with the complexity and dynamics of evolving ontologies and to understand why inconsistencies exist (or have been created) as well as how to fix them.

The work presented in this chapter is the result of the author's joint work with Simon Schenk [Schenk et al., 2011]. Simon Schenk's main contribution consists of the formalization of provenance. His formalization allows for the computation of provenance of inferred knowledge in description logics and includes reasoning with conflicting and incomplete provenance. My main contribution consists of an optimized algorithm for computing provenance and its evaluation.

Correspondingly, Schenk proposed a debugging framework for provenance based on *pinpointing*. Pinpointing means identifying relevant axioms where an axiom is defined as relevant if an incoherent ontology becomes coherent once this axiom is removed, or if a previously unsatisfiable concept turns satisfiable. As pinpointing summarizes explanations for axioms in a single Boolean formula, it then can be evaluated using a provenance algebra as described in Chapter 3. The algorithm presented in Chapter 4 computes explanations of the answer and uses pinpointing to compute provenance in an algebraic way.

However, the computation of all pinpoints may become very expensive and inapplicable if users need to interact with dynamically changing knowledge in real time. The optimized algorithm proposed in this work enables the use of provenance for debugging in real time even for very large and expressive ontologies.

Lastly, this chapter presents a discussion on the algorithm complexity (see RQ 1.IV *Does the exploitation of provenance lead to computation overhead?*).

PART III: Using Provenance in Semantic Web Applications

Chapter 5: An Efficient Provenance-Aware News Feed Ranking Algorithm via Preference Aggregation

This chapter tackles the problem of providing the most relevant messages according to the user's preferences. Users state their preferences on different aspects of the data (e.g., on information source, recency, reliability, location, etc.), and this information is then aggregated to obtain a joint ranking. The aggregation problem is related to the problem of *preference aggregation* in Social Choice Theory [Kelly, 1988]. The

traditional problem formulation of preference aggregation assumes a fixed set of preference orders and a fixed set of domain elements. This thesis investigates how an aggregated preference order has to be updated when the domain is dynamic, i.e., the aggregation approach ranks messages 'on the fly' as the message passes through the system. In order to efficiently handle the dynamic setting, Chapter 5 presents a computational approach for online preference aggregation (see RQ 2.I *Does the preference aggregation problem can be efficiently solved in a dynamic setting?*).

Chapter 6: Managing Data Changes in Linked Open Data Sources

This chapter presents two techniques for dealing with data dynamics in the LOD sources using provenance information. The first one describes an investigation of availability and conformance of provenance information for detecting changes of LOD sources. More precisely, it describes an investigation on the availability, conformance and reliability of the HTTP Header *Last-Modified* field, which indicates the date and time at which the origin server interprets the resource was last modified. The HTTP header is a textual part of the HTTP response message which is sent by any Linked Data server when dereferencing a resource.

Naïve approaches for detecting changes in a resource in a LOD source mainly download two arbitrarily-sized descriptions of an RDF resource and further compare them. The main idea here is to use available provenance information to speed up change detection and avoid such comparisons (see RQ 2.II *Can provenance delivered by Linked Data servers support applications for detecting changes of the resources in a LOD source?*). Linked Data applications benefit from this information since simply checking them may help users to decide which caches or sources need to be updated.

However, it is crucial that *Last-Modified* field provides correct values, otherwise it has no use in any practical application. To this end, this chapter presents an analysis of a large-scale dataset obtained from the LOD cloud by weekly crawls over almost two years. In these weekly snapshots, the author checked to see if the HTTP header field *Last-Modified* was available and if the date provided for the last modification aligned with the observed changes in the data itself.

The second part of this chapter formalizes and evaluates a strategy for capturing data changes based on a time-dependence measure that captures the frequency, degree, and regularity of the changes of the LOD sources. While state-of-the-art metrics [Ding and Finin, 2006, Dividino et al., 2013, Käfer et al., 2013] quantify changes between two sets, e.g. two versions of the same dataset captured at two different points in time, the dynamics function proposed in this work involves the analysis of the dataset's development over a period of time, i.e. a time interval beginning at an initial point in time up to a final one. (see RQ 2.III *Does the consideration of changes within a time interval instead of between two points in time improve change analysis?*). The proposed dynamics function is defined as an aggregation of changes over time, built on top of contemporary change metrics and may be extended to incorporate the use of different decay functions for stressing or weakening periods within a time interval. This notion intends to establish a method for capturing the behaviour of the LOD sources and evolution over a certain period

of time.

Lastly, this chapter discusses the effectiveness of the dynamics function for the purpose of caching maintenance. In order to identify which is/are the most adequate update scheduling strategy(ies) to manage caching copies of the LOD sources (see RQ 2.IV *Which is the most adequate update scheduling strategy to manage caching copies of the LOD sources?*), an evaluation on different scheduling strategies is conducted. The evaluation is done on a large-scale LOD dataset that is obtained from the LOD cloud by weekly crawls over the course of three years. The evaluation is divided in two different setups: (i) in the single step setup, the quality of update strategies for a single and isolated update of a local data cache is evaluated, while (ii) in the iterative progression setup, the quality of the local data cache is measured when considering iterative updates over a longer period of time. The evaluation indicates the effectiveness of each strategy for updating local copies of LOD sources, i.e. it demonstrates, for given limitations of bandwidth, the strategies' performance in terms of data accuracy and freshness.

1.4. Publications and Exploitation

The research discussed in this thesis has been presented in conference papers, journal papers, conference posters, and workshop papers. The following is a list of scientific publications where portions of chapters may be found.

Chapter 3: Querying RDF Datasets with Provenance

[Dividino et al., 2009b] Renata Queiroz Dividino, Sergej Sizov, Steffen Staab, Bernhard Schueler: Querying for provenance, trust, uncertainty and other meta knowledge in RDF. J. Web Sem. 7(3): 204-219 (2009)

This journal paper has been written in collaboration with Bernhard Schüler, Sergej Sivoz, and Steffen Staab on a framework for querying with provenance in RDF. It is a completely revised and extended version of the work proposed by Schüler et al. [Schueler et al., 2008]. My contributions include proofs showing that the provenance evaluation of SPARQL queries is equivalent to the standard SPARQL evaluation, a discussion of soundness and completeness, as well as an extended empirical evaluation section.

[Dividino et al., 2009a] Renata Queiroz Dividino, Simon Schenk, Sergej Sizov, Steffen Staab: Provenance, Trust, Explanations– and all that other Meta Knowledge. KI 23(2): 24-30 (2009) Querying for data and provenance in the Semantic Web

This journal paper summarizes different approaches for querying with provenance in the Semantic Web. My contribution was to summarize the work done in [Dividino et al., 2009b].

Chapter 4: Reasoning and Debugging Evolving OWL Ontologies with Provenance

[Schenk et al., 2009] Simon Schenk, Renata Queiroz Dividino, Steffen Staab: Reasoning With Provenance, Trust and all that other Meta Knowlegde in OWL. SWPM 2009

This workshop paper was written in collaboration with Simon Schenk and Steffen Staab. It describes our initial work towards reasoning with provenance in description logics.

My contributions include providing a detailed grammar for provenance annotations in OWL-2 and comparisons to other approaches.

[Schenk et al., 2011] Simon Schenk, Renata Queiroz Dividino, Steffen Staab: Using provenance to debug changing ontologies. J. Web Sem. 9(3): 284-298 (2011)

This journal paper has been written in collaboration with Simon Schenk and Steffen Staab. Simon Schenk's main contribution consists of the formalization of provenance which allows for the computation of provenance of inferred knowledge in description logics (and includes reasoning with conflicting and incomplete provenance). My contribution consists of an optimized algorithm for computing provenance and a comprehensive evaluation.

Chapter 5: An Efficient Provenance-Aware News Feed Ranking Algorithm via Preference Aggregation

[Dividino et al., 2012] Renata Queiroz Dividino, Gerd Gröner, Stefan Scheglmann, Matthias Thimm: Ranking RDF with Provenance via Preference Aggregation. EKAW 2012: 154-163

This conference paper has been written in collaboration with Gerd Gröner, Stefan Scheglmann, and Matthias Thimm. My main contribution was to provide the algorithms and complexity analysis for the Borda and sequential pairwise aggregators, as well as a comprehensive evaluation for all aggregators proposed.

Chapter 6: Managing Data Changes in Linked Open Data Sources

[Dividino et al., 2014b] Renata Queiroz Dividino, Andre Kramer, Thomas Gottron: An Investigation of HTTP Header Information for Detecting Changes of Linked Open Data Sources. ESWC (Satellite Events) 2014: 199-203

This workshop paper has been written in collaboration with Andre Kramer, and Thomas Gottron. My contributions include the evaluation and analysis of HTTP header data provided by [Käfer et al., 2013].

[Dividino et al., 2013] Renata Queiroz Dividino, Ansgar Scherp, Gerd Gröner, Thomas Gottron: Change-a-LOD: Does the Schema on the Linked Data Cloud Change or Not? COLD 2013

This workshop paper has been written in collaboration with Ansgar Scherp, Gerd Gröner, and Thomas Gottron. This work was inspired by Ansgar Scherp' and Thomas Gottron's previous work [Konrath et al., 2012] on an approach and tool for a stream-based indexing and schema extraction of Linked Open Data (LOD) at web-scale. Using their formalization of *Abstract Schema* e.g., the combinations of sets of properties and sets of types to describe the resources in a specific domain, my contributions include the evaluation of the use of the vocabularies in the LOD cloud, and the analysis how the use of vocabularies on the LOD cloud changes over time.

[Dividino et al., 2014b] Renata Queiroz Dividino, Thomas Gottron, Ansgar Scherp, Gerd Gröner: From Changes to Dynamics: Dynamics Analysis of Linked Open Data Sources. PROFILES@ESWC 2014

This workshop paper has been written in collaboration with Ansgar Scherp, Gerd Gröner, and Thomas Gottron. Together with Thomas Gottron, I have proposed the dynamic

function which can compute the dynamics or evolution of any RDF dataset over a period of time. Additionally, I applied our dynamics function to real world LOD data sources in order to illustrate how the proposed function works and related its results to temporal change patterns.

[Dividino et al., 2015] Renata Queiroz Dividino, Thomas Gottron, Ansgar Scherp: Strategies for Efficiently Keeping Local Linked Open Data Caches Up-To-Date. International Semantic Web Conference (2) 2015: 356-373

This conference paper has been written in collaboration with Ansgar Scherp, and Thomas Gottron. My contributions include the research and implementation of update strategies proposed in the literature, and the analysis of their effectiveness for updating local copies of the LOD sources, i.e. which strategy performs better in terms of data accuracy and freshness for given restrictions of bandwidth.

2. Foundations

2.1. Provenance Foundations

According to the Oxford English Dictionary, provenance means:

1. the fact of coming from some particular source or quarter; origin, derivation
2. the history or pedigree of a work of art manuscript, rare book, etc.; concr., a record of the ultimate derivation and passage of an item through its various owners.

In the past, database and data repositories were mainly under centralized control to serve all of the organization's needs. Such databases and repositories were assumed to be reliable and trustworthy sources of information.

Today, data is often stored in different repositories (such as on the Web) with no centralized control. In an open and collaborative environment, external agents are involved in the process of publishing and sharing data – data is created, copied, moved around, and combined indiscriminately.

With the increasing amount of data available on the Web, data quality assessment becomes more and more important as it allows the user to ascertain the veracity, credibility and validity of the information. Information about *Provenance* is essential as it provides context which can help end users judge the validity of the knowledge derived from such information sources, such as

- *Where* the data is from,
- *Who* has provided the data,
- *When* was the data provided,
- Proof from the provider about the truth of this data,
- External validation of the credibility of the data.

The provenance of the information can help data consumers make judgments and seek for validation as it describes the data's origins and its derivation in its life cycle. Provenance information (also called lineage) is a record that describes the recentness, reputation, and degree of confidence in the piece of data as well as the people, institutions, entities, and activities involved in the data transformation process. Provenance has been used in many data management tasks for checking the integrity of data in databases or repositories.

2.1.1. Workflow and Data Provenance

Provenance has been extensively studied both in database [Buneman et al., 2012, Cheney et al., 2009, Kostylev and Buneman, 2012, Boulakia and Tan, 2009, Buneman et al., 2006, Cheney et al., 2014, Buneman, 2013] and workflow [Bose and Frew, 2005, Simmhan et al., 2005, Davidson and Freire, 2008] management.

Database provenance, also called fine-grained provenance [Tan, 2007], has been designed for relational data and is modeled as annotations on data. Provenance models define mechanisms for evaluating queries over annotated relations with a focus on understanding the transport of annotations from the input relation to the output data. Particularly, provenance in databases has been successfully used in a wide range of applications such as in confidence computation [Archer et al., 2013, Agrawal et al., 2006], view maintenance and update [Buneman et al., 2002], annotation and propagation [Buneman et al., 2008, Bhagwat et al., 2005, Wang and Madnick, 1990], and viualization [Hoekstra and Groth, 2014, Deutch et al., 2015a].

Workflow provenance – or coarse provenance – aims to capture a complete history of the workflow execution that resulted in a data item. Workflow provenance captures a notion of a causal graph, explaining how the output data was derived in an execution. This may involve tracking the interaction of programs, external devices as well as human interaction with the process when such components of the system are treated as black boxes. Workflow provenance is crucial to verification and validation in many scientific applications such as workflow management [Bao et al., 2012, Huang et al., 2015, Murta et al., 2014], upgrades and evolution [Koop et al., 2010, Koop and Freire, 2014, Simmhan et al., 2008], privacy [Davidson et al., 2011b, Davidson et al., 2011c, Davidson et al., 2011a] and a number of tools for capturing provenance have been developed such as myGrid/Taverna [Oinn et al., 2004], Kepler [Bowers and Ludäscher, 2005] and VisTrails [Callahan et al., 2006].

New approaches work towards marrying database-style and workflow-style provenance [Deutch et al., 2015c, Chirigati and Freire, 2012, Amann et al., 2013, Amsterdamer et al., 2011a]. They enable different levels of granularity in provenance querying, i.e., they support tracking and querying fine-grained workflow provenance.

This dissertation focuses on provenance within databases (i.e. data provenance); therefore, in the next sections we introduce the most common notions of provenance for database queries.

2.1.2. Why, Where, and How Provenance

Provenance was initially studied as an extension for relational databases (i.e., data management systems based on relational algebra), probabilistic databases [Fuhr and Rölleke, 1997], and was later adopted for RDF knowledge bases (i.e., for semantics of SPARQL query language).

In the area of database systems, provenance is often represented using an extension of the relational data model, coined *annotated relations*. Its purpose is primarily the description of data origins (provenance) and the process by which it arrived as a query answer [Cui and Widom, 2000, Buneman et al., 2000, Buneman et al., 2001, Ding et al., 2005]. Basically they define custom (possibly different) interpretations for algebraic operations of Boolean formulas (built on tuple identifiers as Boolean variables) for particular dimensions of provenance (e.g., agent, timestamp, source, certainty) to obtain the m-dimensional record (i.e., query result). Indeed, a Boolean expression of the result set built from tuple identifiers does not only carry the information *which* triples have contributed to a variable assignment (why-provenance) but

Movie Theater	Movie	Provenance
Cinemark Palace I	*Star Wars: Episode VII*	b1
Cinemark Palace I	*The Revenant*	b2
Cinemark Palace I	*Man on a Ledge*	b3
Prado Cafe & Cinema	*Start Wars: Episode VII*	b4
Prado Cafe & Cinema	*Mistress America*	b5
Cineplex	*Star Wars: Episode VII*	b6
Cineplex	*The Revenant*	b7

Table 2.1.: Our example (annotated) database M: Movie Theaters and Movies

Movie	Genre	Provenance
Star Wars:Episode VII	Action	c1
Mistress America	Comedy	c2
Man on a Ledge	Action	c3
The Revenant	Action	c4

Table 2.2.: Our example (annotated) database G: Movies and Movie Genres

also *how* they contributed (how-provenance). This draws a distinction between the three main notions of provenance:

- Where-provenance [Wang and Madnick, 1990, Buneman et al., 2001]: where the given pieces of data are physically serialized in database tuples,
- Why-provenance [Cui et al., 2000, Buneman et al., 2001]: which subset of database tuples contributed to the result), and
- How-provenance[Green et al., 2007]: how particular tuples were used for constructing the result.

Here we provide some examples of these three notions. For more details, please refer to [Cheney et al., 2009]. Cheney et al. [Cheney et al., 2009] provide a high-level overview of why, how, and where provenance, and describe the relationships among these three main notions of database provenance.

Example 2.1.1 *As a matter of example, Table 2.1 and Table 2.2 show two relations with annotated tuples respectively. The first relation describes movie theaters and the movies they display. The second relation describes the movie genres.*

The notion of why-provenance was formally introduced by Buneman et al. [Buneman et al., 2001]. Suppose an annotated relation $V = Q(D)$ is constructed by a query Q applied to source annotation relations D, why-provenance of an output tuple t in V is defined as the witness basis of t, i. e. which are all the combinations of the source tuples which justify the existence of t. Buneman et al. [Buneman et al., 2001] also introduce the notion of where-provenance. While why-provenance describes the source tuples that had some influence on the existence of

Result of Q_1	Why-Provenance
Cinemark Palace I	$\{\{b1, c1\}, \{b2, c4\}, \{b3, c3)\}$
Prado Cafe & Cinema	$\{\{b4, c1\}\}$
Cineplex	$\{\{b6, c1\}, \{b7, c4\}\}$

Table 2.3.: Example of Why-Provenance

Result of Q_1
Cinemark Palace I^{b1,b2,b3}
Prado Cafe & Cinema^{b4}
Cineplex^{b6,b7}

Table 2.4.: Example of Where-Provenance

the output tuples, where-provenance describes locations in the source databases from which the data was extracted. In the relational setting, location is a column of a tuple in a relation.

Example 2.1.2 *We construct a SQL Query to retrieve all movie theaters that play action movies.*

```
SELECT m.MovieTheater
FROM m M, g G
WHERE m.Movie = g.Movie
AND g.Genre = 'Action'
```

And in relational algebra notation:

$\pi_{\text{'Movie Theater'}}(Movie \bowtie \sigma_{genre=\text{'Action'}}(Genre))$

Why- and where- provenance leave out some information about how an output tuple is derived according to the query. How-provenance fills this gap by describing the source data and the operations used to produce the derived data. In the next section, we formally describe the notion of how-provenance which has been introduced by Green et al. [Green et al., 2007] for relational algebra in terms of an appropriate provenance semirings.

Example 2.1.3 *Using the relations presented in Table 2.1 and Table 2.2 in which tuples are tagged with their own id. Applying to it the query from Example 2.1.2, where- and why-provenance is shown in Table 2.4 and Table 2.3 respectively.*

The why-provenance of "Cinemark Palace I" is the set $\{\{b1, c1\}, \{b2, c4\}, \{b3, c3\}\}$*. This tells us that the output source is witnessed by the sources tuples in three different ways: the first uses the tuples* $b1$ *and* $c1$*, the second* $b2$ *and* $c4$*, and the last* $b3$ *and* $c3$*.*

The where-provenance of "Cinemark Palace I" is the location ($b1$*, Movie Theater), (*$b2$*, Movie Theater), and (*$b3$*, Movie Theater) since "Cinemark Palace I" was copied from the Movie Theater attribute of the tuples* $b1$*,* $b2$*,and* $b3$ *in the relation* M*, according to the query.*

2.1.3. Provenance Semirings

The notion of *how-provenance* for positive relational algebra (RA^+) was introduced by Green et al. [Green et al., 2007]. Suppose an annotated relation $V = Q(D)$ is constructed by a query Q applied to source annotation relations D, how-provenance models how each output tuple in V was actually derived as a result of applying a relational query Q from sources tuples in D. Green et al. show that many of the mechanisms for evaluating queries over such annotated relations can be unified in a general framework based on K-relations, which are relations whose tuples are annotated with elements from a commutative semiring K.

They used the perspective [Fuhr and Rölleke, 1997] of the relational model in which tuples $t : U \mapsto D$ with U a finite set of attributes and D a domain of values, e. g. it considers relations in which the tuples are annotated (tagged) with tuples ids. Formally, let K be a set containing a distinguished element 0. A K-relation models a relation as a function R on all possible tuples, where R maps tuples in the relation to nonzero elements of K, and tuples that are not in the relation to the special element 0. Here, U-Tuples denotes the set of all tuples with attributes U. A K-relation $R : U\text{-Tuple} \mapsto K$ corresponds to a finite relation whose elements are tagged with elements of K.

Definition 2.1.1 *Let K be a set containing a distinguished element 0. A K-relation over a finite set of attributes U is a function $R : U\text{-Tuples} \mapsto K$ such that its support defined by $supp(R) = \{t | R(t) = 0\}$ is finite.*

Basically, the semiring framework distinguishes two basic operations that may be applied to the source tuples in D by Q to derive the output tuples in V: source tuples can be either joined together as an effect of a join, or merged together via union or projection. The basic relational algebra operators are mapped into operations on an algebraic structure $(K, 0, 1, \wedge, \vee)$ as follows.

Definition 2.1.2 *Let $(K, 0, 1, \wedge, \vee)$ be an algebraic structure with two binary operations $\wedge$ and $\vee$ and two distinguished elements 0 and 1. The operations of the positive K-relational algebra are defined as follows.*

- Empty relation. *For any set of attributes U, there exists $\varnothing : U\text{-Tuples} \mapsto K$ such that $\varnothing(t) = 0$.*

- Selection. *Let $R : U\text{-Tuples} \mapsto K$ and P be a selection predicate that maps each U-Tuple to either 0 or 1. Then $\sigma_P(R) : U\text{-Tuples} \mapsto K$ is defined by*

$$(\sigma_P(R))(t) = R(t)P(t).$$

  That is, $(\sigma_P(R))(t)$ is $R(t)$ if P holds on t and 0 otherwise.

- *Projection. Let $R : U\text{-Tuples} \mapsto K$ and $J \subseteq U$. Then $\pi_J(R) : V\text{-Tuples} \mapsto K$ is defined by:*

$$(\pi_J(R))(t) = \sum_{t=t'[J] \ and R(t') \neq 0} R(t).$$

	Result of Q_1
Cinemark Palace I	$(b1 \wedge c1) \vee (b2 \wedge c4) \vee (b3 \wedge c3)$
Prado Cafe & Cinema	$(b4 \wedge c1)$
Cineplex	$(b6 \wedge c1) \vee (b7 \wedge c4)$

Table 2.5.: Provenance Polynomials

- *Union. Let* $R_1, R_2 : U\text{-}Tuples \mapsto K$. *Then* $R_1 \cup R_2 : U\text{-}Tuples \mapsto K$ *is defined by:*

$$(R_1 \cup R_2)(t) = R_1(t) \wedge R_2(t).$$

- *Natural Join. Let* $R_1 : U_1\text{-}Tuples \mapsto K$ *and* $R_2 : U_2\text{-}Tuples \mapsto K$. *Then* $R_1 \bowtie R_2 : U_1 \cup U_2\text{-}Tuples \mapsto K$ *is defined by:*

$$(R_1 \bowtie R_2)(t) = R_1(t_1) \vee R_2(t_2), \text{ where } t_1 = t[U_1] \text{ and } t_2 = t[U_2]$$

If we assume that K-relational semantics satisfies the same equivalence laws as RA^+ operators over bags (i. e. union ($\wedge$) is associative, commutative and has identity $\varnothing$, join ($\vee$) is associative, commutative and distributive over union, and projection and selection commute with each other, as well as with union and join), Green et al. conclude that (K,0,1,$\wedge$,$\vee$) must be a commutative semiring. An algebraic structure (K,0,1,$\wedge$,$\vee$) is a commutative semiring if (K,0,$\wedge$) and (K,1,$\vee$) are commutative monoids, $\vee$ distributes over $\wedge$ and $0 \vee a = a \vee 0 = 0$, $\forall a \in K$.

Example 2.1.4 *Using again the relations in Table 2.1 and Table 2.2 in which tuples are tagged with their own id. Applying to it the query from Example 2.1.2 and evaluating in the provenance semiring, we obtain the relation shown in Table 2.5.*

The provenance of "Cinemark Palace I" is $(b1 \wedge c1) \vee (b2 \wedge c4) \vee (b3 \wedge c3)$ *which can be understood as follows: "Cinemark Palace I" is derived by joining the input tuples* $b1$ *and* $c1$ *or* $b2$ *and* $c4$*, or* $b3$ *and* $c3$*.*

Green et al. propose polynomials $(\mathbb{N}[X], +, *, 0, 1)$, the most general commutative semiring, as a suitable provenance model for RA+. If each source tuple in a database D is annotated with a distinct tuple id, the semiring gives us the how-provenance for each output tuple in the form of a polynomial with coefficients from the set $\mathbb{N}$ of natural numbers and indeterminate (or variables) from the set of source tuple ids.

The classical semiring algebraic structure is used in devising a general framework for uniformly treating various extensions to relational algebra such as handling bag semantics or incomplete and probabilistic databases.

Examples of commutative semirings:

- $(B$, false,true, $\vee, \wedge)$
- $(P(\Omega), \varnothing, \Omega, \cup, \cap)$, used for event tables which is also an example of distributive lattice.
- $(\mathbb{N}, 0, 1, +, *)$, the natural numbers

	Result of Q_1
Cinemark Palace I	$(1 * 1) + (1 * 1) + (1 * 1) = 3$
Prado Cafe & Cinema	$(1 * 1) = 1$
Cineplex	$(1 * 1) + (1 * 1) = 2$

Table 2.6.: Bag semantics example: We evaluating the provenance polynomials given in Table 2.5 in $(\mathbb{N}, 0, 1, +, *)$, with $b1, \ldots, b7 = 1, c1, \ldots, c4 = 1$ which represent the multiplicity of the tuples in the multiset.

- $(\mathbb{N}^{\infty}, \infty, 0, min, +)$, the tropical semiring [Kuich, 1997]
- $([0, 1], 0, 1, max, min)$ is related to fuzzy sets [Zadeh, 1965], called the fuzzy semiring.
- The semiring of confidentiality policies [Foster et al., 2008] $(C, P, 0, min, max)$, where the total order $C = P < C < S < T < 0$ describes levels of security clearance: P public, C confidential, S secret, and T top-secret.

Example 2.1.5 *To illustrate instances of communitative semirings, consider the provenance polynomials in Table 2.5. Suppose the annotations are natural numbers which represent the multiplicity of the tuples in the multiset. (A tuple not listed in the table has multiplicity 0.) Evaluating the provenance formula from Example 2.1.4 of "Cinemark Palace I" in* $(\mathbb{N}, 0, 1, +, *)$ *for* $b1, \ldots, b7 = 1, c1, \ldots, c4 = 1$*, we get 3 which is indeed the multiplicity of "Cinemark Palace I" in Table 2.6.*

The concept of provenance semirings has been extended [Kostylev and Buneman, 2012, Amsterdamer et al., 2012, Amsterdamer et al., 2011c, Amsterdamer et al., 2011b, Kostylev and Buneman, 2012] to deal with update, aggregation, and negation, as well as has been successfully used in a wide range of database applications such as in confidence computation, view maintenance and update, debugging, and annotation propagation, and summarization.

In Chapter 3 we propose a framework for querying RDF with provenance. Our methodology follows the same idea and adopts the notion of provenance semirings which was introduced here and also allows the same distinction between where-, how-, and why-provenance.

2.2. Semantic Web Foundations

The traditional World Wide Web is a global information space comprising linked Web documents. The Semantic Web can be seen as an extension of the traditional Web which goes from linked documents to linked data.

The Semantic Web provides a common framework that makes Web data available in a well-structured and machine-readable format such that it can be published, reused, combined, and automatically integrated with other machine-readable data. Essentially, it is a paradigm that facilitates discovery and interoperability of structured Web data spread in the many different Web sources. There is no doubt that one of the key benefits of Semantic Web technology is the better support of decentralized, self-organizing knowledge exchange between users.

The remaining sections outline the core Semantic Web standards, starting with the Resource Description Framework (RDF) data-model and related ones such the SPARQL query

language for querying RDF, and representing ontologies with the Web Ontology Language (OWL). Lastly, we briefly introduce Linked Data which aims to provide a set of principles by which the Semantic Web standards can be effectively deployed on the Web.

2.2.1. Resource Description Framework

We introduce the necessary notions about RDF. For more information please refer to [Manola et al., 2014, Klyne et al., 2014, Gutierrez et al., 2011].

The RDF standard provides the basis for a core data model for data interchange on the Semantic Web. RDF is a graph-based knowledge representation language that provides a flexible way to describe resources (i.e., entities in the world) and the relationships between them. The nodes in a graph are represented by Internationalized Resource Identifier (IRIs) [Durst and Suignard, 2005], blank nodes (a kind of existentially quantified variables) or literals. IRIs serve as global identifiers that can be used to identify any resource. A literal is used to identify values such as numbers and dates by means of a lexical representation. Arcs between the nodes, labeled with IRIs, represent their relationships. In the following definitions, we simplify the RDF graph model in order to come up with a more concise formal characterization like [Arenas et al., 2009].

Definition 2.2.1 (RDF Terms, Triple and Variables) *Let I be the set of IRIs, L the set of RDF Literals and B the set of Blank Nodes as defined in [Patrick J. Hayes and Peter F. Patel-Schneider, 2014]. I, L and B are pairwise disjoint. A* statement *is an RDF triple in $(I \cup B) \times I \times (I \cup L \cup B)$. If $S = (s, p, o)$ is a statement, s is called the subject, p the predicate and o the object of S. We denote the union $I \cup L \cup B$ by T (RDF terms). Assume additionally the existence of an infinite set V of variables disjoints from sets aboves.*

Example 2.2.1 *Table 2.7 presents some of the RDF data about movies from our toy scenario. In this example there are five prefix declarations and four triples (# denotes a comment line in Turtle). Each triple is comprised of a subject, a predicate, and an object. The subject describes the resources (identified by IRIs or blank nodes) which of our scenario (e. g. ,* `dbr:Star_Wars:_The_Force_Awakens` *and* `dbr:Man_on_a_Ledge`*). The predicate describes the relation or attributes of the resources (e. g. , foaf:name, rdf:types) using an IRI. Lastly, the object represents the value for that relation (e. g. , 6120,* `dbo:Film`*) and is represented by an IRI or literal (and potentially a blank node).*

Definition 2.2.2 (RDF Graph) *An RDF* graph *G is a set of statements. If G is an RDF graph, then* $\text{term}(G)$ *is the set of elements of T appearing in the triples of G, and* $\text{blank}(G)$ *is the set of blank nodes appearing in G* ($\text{blank}(G) = \text{term}(G) \cap B$)*. For every two RDF graphs G_1 and G_2 the sets of blank nodes used in G_1 and in G_2 are disjoint, i.,e.* $\text{blank}(G_1) \cap \text{blank}(G_2) = \emptyset$.

Figure 2.1 renders a diagram of the RDF graph presented in Table 2.7 as a directed labeled graph.

Named graphs [Carroll et al., 2005, Carroll and Stickler, 2004] offer a means to group a set of statements in a graph and to refer to this graph using a IRI. This way, information about the graph can be expressed in RDF using its name as subject or object:

# PREFIX DECLARATIONS		
@prefix	dbo:	<http://dbpedia.org/ontology/> .
@prefix	dbr:	<http://dbpedia.org/resource/> .
@prefix	dbp:	<http://dbpedia.org/property/> .
@prefix	rdf:	<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix	foaf:	<http://xmlns.com/foaf/0.1/> .
#RDF TRIPLES		
`dbr:Star_Wars:_The_Force_Awakens`	`rdf:type`	`dbo:Film`
`dbr:Star_Wars:_The_Force_Awakens`	`foaf:name`	Star Wars: The Force Awakens (en).
`dbr:Man_on_a_Ledge`	`rdf:type`	`dbo:Film`
`dbr:Man_on_a_Ledge`	`dbp:runtime`	6120.

Table 2.7.: The following example presents some of the RDF data about movies from our toy scenario (# denotes a comment line in Turtle):

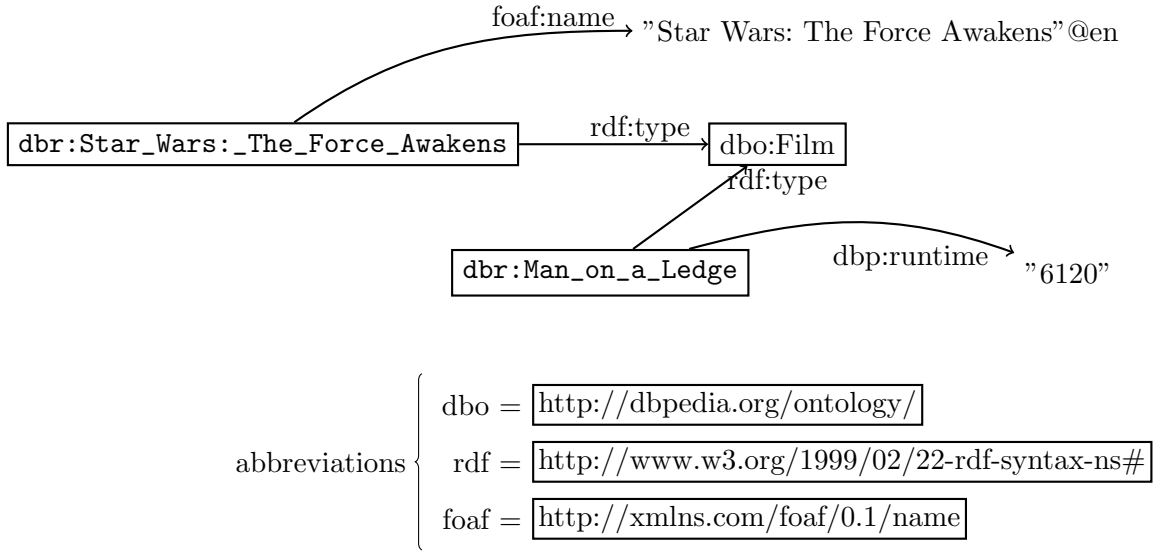


Figure 2.1.: Diagram of the RDF graph presented in Example 2.2.1 as a directed labeled graph.

Definition 2.2.3 (RDF Named Graph) *A named graph is a pair* (i, G) *of a IRI* i*, called name, and an RDF graph* G*, called the extension.*

Example 2.2.2 *Following up with our toy scenario, we extended the RDF graph presented in Table 2.8 to a RDF Named Graph by adding a name in the form of a IRI reference (e. g. ,* `ex:FilmsISaw`*). The example is serialized in TriG [Bizer and Cyganiak, 2014] which extends Turtle by introducing curly brackets to group triples into multiple graphs, and to precede each by the name of that graph.*

Finally, RDF datasets can be defined as followed:

Definition 2.2.4 (RDF Dataset) *An RDF dataset* $\mathcal{D}$ *is a set of RDF graphs in which every graph is identified by an IRI, except for a distinguished graph in the set called the default*

# PREFIX DECLARATIONS		
@prefix	dbo:	<http://dbpedia.org/ontology/> .
@prefix	dbr:	<http://dbpedia.org/resource/> .
@prefix	dbp:	<http://dbpedia.org/property/> .
@prefix	rdf:	<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix	foaf:	<http://xmlns.com/foaf/0.1/> .
@prefix	ex:	<http://www.mythesis.de/ToyScenario/> .
#RDF TRIPLES		
ex:FilmsISaw{		
dbr:Star_Wars:_The_Force_Awakens	rdf:type	dbo:Film
dbr:Star_Wars:_The_Force_Awakens	foaf:name	Star Wars: The Force Awakens (en).
}		
ex:FilmsIWantToSee{		
dbr:Man_on_a_Ledge	rdf:type	dbo:Film
dbr:Man_on_a_Ledge	dbp:runtime	6120.
}		

Table 2.8.: Example of a RDF Named Graph containing some of the RDF data about movies from our toy scenario:

graph. Formally, $\mathcal{D} = \{(G_0), (u_1, G_1), ..., (u_n, G_n)\}$*, where each* $G_0, \ldots, G_n$ *are RDF graphs and* $u_0, \ldots, u_n$ *distinct IRI,* $n \geq 0$*.* G_0 *is called the default graph of* $\mathcal{D}$*, and each pair* (u_i, G_i) *is a named graph, with* u_i *being the name of* G_i*; defined* $gr(u_i) = G_i$ *if* $\langle G_i, u_i \rangle \in \mathcal{D}$*,* $gr(u_i) = 0$ *otherwise. e. Additionally,* $\text{name}(\mathcal{D})$ *stands for the set of IRIs that are names of graphs in* $\mathcal{D}$*, and* $\text{term}(\mathcal{D})$ *and* $\text{blank}(\mathcal{D})$ *stand for the set of terms and blank nodes appearing in the graphs of* $\mathcal{D}$*, respectively. For every two RDF graphs* G_1 *and* G_2 *in* $\mathcal{D}$ *the sets of blank nodes used in* G_1 *and in* G_2 *are disjoint, i.,e.* $\text{blank}(G_1) \cap \text{blank}(G_2) = 0$*.*

Figure 2.2 renders a diagram of the RDF graph presented in Table 2.8 as a directed labeled graph.

The semantics of RDF is defined in as a model theory [Patrick J. Hayes and Peter F. Patel-Schneider, 2014]. Model theory assumes that the language refers to a *world*, and describes the minimal conditions that a world must satisfy in order to assign an appropriate meaning for every expression in the language. Worlds serve as interpretations of RDF data. Basically, the idea is to provide an abstract and formal foundation for stating what the IRIs used in the RDF data identify in the world, what things in the world are related through which properties, and so forth.

Definition 2.2.5 *Let* $\mathfrak{V}$ *be a vocabulary containing all names (IRIs and literals) occurring in RDF triples. An RDF interpretation* $\mathcal{I}$ *of* $\mathfrak{V}$ *consists of:*

- $\mathcal{IR}$*: a non-empty set of resources (domain or universe) of* $\mathcal{I}$
- $\mathcal{IP}$*: a set of properties of* $\mathcal{I}$
- $\mathcal{I}_S : \mathfrak{V} \mapsto (\mathcal{IR} \cup \mathcal{IP})$*: a function mapping resources (with each symbol from* $\mathfrak{V}$ *a resource is associated).*

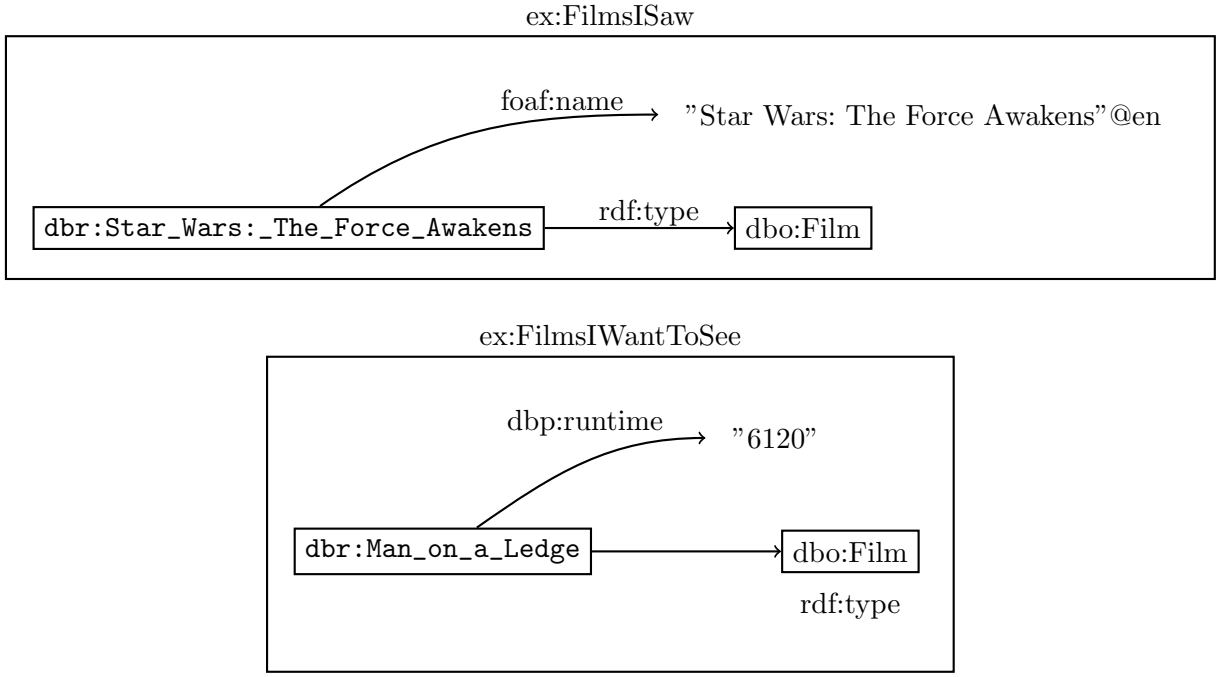


Figure 2.2.: Diagram of the RDF dataset presented in Table 2.8 as a directed labeled graph.

- $\mathcal{I}_{\mathcal{EXT}} : \mathcal{IP} \mapsto 2^{\mathcal{IR}\times\mathcal{IR}}$*: a function mapping properties (each property is a binary relation).*
- $\mathcal{I}_{\mathcal{L}}$ *: a partial mapping from literals into* $\mathcal{IR}$

Then:

- *An literal* l *is true in* $\mathcal{I}$ *if and only if* $l \in \mathfrak{V}$, $\mathcal{I}_{\mathcal{L}}(l) \in IR$.
- *A IRI* i *is true in* $\mathcal{I}$ *if and only if* $i \in \mathfrak{V}$, $\mathcal{I}_{\mathcal{S}}(i) \in IR$.
- *An RDF triple* $\langle s, p, o\rangle$ *is true in* $\mathcal{I}$ *if and only if* $s, p, o \in \mathfrak{V}$, $\mathcal{I}_{\mathcal{S}}(p) \in IP$ *and* $(\mathcal{I}_{\mathcal{S}}(s), \mathcal{I}_{\mathcal{S}}(o)) \in \mathcal{I}_{\mathcal{EXT}}(\mathcal{I}_{\mathcal{S}}(p))$.
- *An RDF triple is false in* $\mathcal{I}$ *if it is not true in* $\mathcal{I}$.
- *An RDF graph is true in* $\mathcal{I}$ *if and only if every triple of it is true in* $\mathcal{I}$*. In particular, it is false in* $\mathcal{I}$ *if some triple is not true in* $\mathcal{I}$.

The RDF Schema (RDFS) provides a data-modeling vocabulary for RDF data. RDF Schema is an extension of the basic RDF vocabulary and allows for attaching semantics to user-defined classes and properties. For details, please refer to [Dan Brickley and R.V. Guha, 2014].

2.2.2. Querying RDF: The SPARQL Query Language

SPARQL is a query language for RDF based on graph pattern matching. Its syntax and semantics is specified in [Prud'hommeaux and Seaborne, 2008]. Essentially, SPARQL is a graph-matching query language. Query results in SPARQL are given by partial substitutions (mapping) of the query variables by RDF terms, i.e., given a RDF dataset D, a query consists of a pattern which may contain a variable instead of an RDF term in the subject, predicate or object positions. The query is matched against D, and the RDF terms obtained from this matching are processed to give the results.

Syntax of SPARQL

On a high level, a SPARQL query can consist of up to five main parts:

- Prefix Declarations: define URI prefixes (in a similar fashion to Turtle).
- Dataset Clause: specifies the dataset over which the query should be executed. A SPARQL query is evaluated against a dataset consisting of a set of named graphs (declared using FROM NAMED clauses) and a default graph, which is the union of one or more named graphs (declared using FROM clauses).
- Query Forms: specifies what type of SPARQL query is being executed. In this work we are only interested in SPARQL SELECT and CONSTRUCT query forms. which allow us to specify how resulting variable bindings or RDF graphs, respectively, are formed based on the solutions from graph pattern matching [Prud'hommeaux and Seaborne, 2008].
- Query Clause: specifies the query patterns that are matched against the data and used to generate variable bindings.
- Solution Modifiers: used for projection, ordering, slicing and paginating the results.

In the following definitions, we lay out the core features of a query clause, and describe the existing foundations of SPARQL introduced in [Pérez et al., 2009, Arenas et al., 2009].

A SPARQL graph pattern expression is defined recursively as follows:

1. A tuple from $(I \cup L \cup V) \times (I \cup V) \times (I \cup L \cup V)$ is a graph pattern (a triple pattern).
2. If $P1$ and $P2$ are graph patterns, then expressions $(P1$ AND $P2)$, $(P1$ OPT $P2)$, and $(P1$ UNION $P2)$ are graph patterns.
3. If P is a graph pattern and R is a SPARQL built-in condition, then the expression $(P$ FILTER $R)$ is a graph pattern.

A SPARQL built-in condition is constructed using elements of the set $V \cup T$ and constants, logical connectives $(\neg, \wedge, \vee)$, inequality symbols $(<, \leq, \geq, >)$, the equality symbol $(=)$, unary predicates like bound, isBlank, and isIRI, plus other features (refer to [Prud'hommeaux and Seaborne, 2008] for a complete list).

In the rest of this chapter, we use $var(P)$ to denote the set of variables occurring in the graph pattern P. In particular, if P is a basic graph pattern, then $var(P)$ denotes the set of variables occurring in the triple patterns that form P. Additionally, for a built-in condition R, we use $var(R)$ to denote the set of variables occurring in R, and the condition $var(R) \subseteq var(P)$ holds. We conclude the definition of the algebraic framework by describing the formal syntax of the SELECT query result form. A SELECT SPARQL query is simply a tuple (W, P), where P is a SPARQL graph pattern expressions and W is a set of variables such that $W \subseteq var(P)$.

Example 2.2.3 *Here is a brief example of a SPARQL query. Comment lines are prefixed with #. The query first defines prefixes that can be re-used later in a similar fashion to that allowed by Turtle. The # DATASET CLAUSE selects the dataset over which the query should be run. The # RESULT CLAUSE states what kind of results should be returned for the query: in this case, a unique (i. e. DISTINCT) set of pairs of RDF terms matching the ?film and ?duration variables respectively. Next, the # QUERY CLAUSE states the graph patterns that the query should match against: we are looking for the title and runtime of all entities of rdf:type dbo:Film. Finally, the # SOLUTION MODIFIER section allows for putting a limit on the number of results returned, to order results, or to paginate results: in this case, a maximum (i. e. LIMIT) of two results is requested from the query*

```
# PREFIX DECLARATIONS
@prefix dbo: <http://dbpedia.org/ontology/>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix dbp: <http://dbpedia.org/property/>
# DATASET CLAUSE
FROM <http://www.mythesis.de/ToyScenario/FilmsIWantToSee.rdf>
# RESULT CLAUSE
SELECT DISTINCT ?film ?duration
# QUERY CLAUSE
WHERE { ?film rdf:type dbo:Film ;
       dbp:runtime ?duration.}
# SOLUTION MODIFIER
LIMIT 2
```

If we assume that the triples presented in Table 2.8, we expect a result like:

?film	?duration
dbr:Man_on_a_Ledge	6120

SPARQL Semantics

Definition 2.2.6 (Triple and Basic Graph Patterns) *A tuple t in $(I \cup L \cup V) \times (I \cup V) \times (I \cup L \cup V)$ is a triple pattern. A Basic Graph Pattern (BGP) is a finite set of triple patterns. Given a triple pattern t, $var(t)$ is a set of variables occurring in t. Similarly, given a basic graph pattern P, $var(P) = \bigcup_{t \in P} var(t)$, i. e. $var(P)$ is the set of variables occurring in P.*

In triple patterns, blank nodes are considered as existential variables, i.e., as query variables whose bindings cannot be used outside of the query clause (and thus that cannot be returned in results). Therefore, we do not consider blank nodes in SPARQL query patterns since they can be treated analogously to non-distinguished query variables: variables that cannot be used elsewhere outside of the query-clause scope.

Thereby, a variable assignment is defined by a mapping:

Definition 2.2.7 (Mapping) *A mapping μ from V to T is a partial function $\mu : V \to T$. The domain of μ, $dom(\mu)$, is a subset of V where μ is defined. The empty mapping $\mu_\emptyset$ is a mapping such that $dom(\mu_\emptyset) = \emptyset$ (i.e. $\mu_\emptyset = \emptyset$). Two mappings μ_1 and μ_2 are compatible when for all $x \in dom(\mu_1) \cap dom(\mu_2)$, it is the case that $\mu_1(x) = \mu_2(x)$, then $\mu_1 \cup \mu_2$ is also a mapping.*

Let Ω_1 and Ω_2 be set of mappings; join, union, the difference, and left outer-join between Ω_1 and Ω_2 are defined as:

- $\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2 \text{ are compatible mappings}\}$,
- $\Omega_1 \cup \Omega_2 = \{\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2\}$,
- $\Omega_1 \setminus \Omega_2 = \{\mu \in \Omega_1 \mid \text{for all } \mu' \in \Omega_2 \text{ , } \mu \text{ and } \mu' \text{ are not compatible}\}$,
- $\Omega_1 = \bowtie \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2)$.

The operation $\Omega_1 \bowtie \Omega_2$ is the set of mappings that result from extending mappings in Ω_1 with their compatible mappings in Ω_2. $\Omega_1 \cup \Omega_2$ is the usual set theoretical union. $\Omega_1 \setminus \Omega_2$ is the set of mappings in Ω_1 that cannot be extended with any mapping in Ω_2. A mapping μ is in $\Omega_1 \bowtie \Omega_2$ if it is the extension of a mapping of Ω_1 with a compatible mapping of Ω_2, or if it belongs to Ω_1 and cannot be extended with any mapping of Ω_2. These operations resemble the relational algebra operations but over sets of mappings (partial functions).

A set of mappings can be represented by a relation Ω over the domain $T^{|V|}$, where the variables V are the attributes and assignments are the tuples of this relation.

Definition 2.2.8 (Basic Graph Pattern and Mappings) *Given a basic graph pattern P and a mapping μ such that $var(P) \subseteq dom(\mu)$, we have that $\mu(P) = \bigcup_{t \in P} \{\mu(t)\}$, i. e. $\mu(P)$ is the set of triples obtained by replacing the variables in the triples of P according to μ.*

We can now define the semantics for basic graph patterns as a function $[\![P]\!]_G^D$ that given a basic graph pattern P returns a set of mappings.

Definition 2.2.9 (Basic Graph Pattern Matching) *Let G be an RDF graph over a RDF dataset D, and P a basic graph pattern. The evaluation of P over G, denoted by $[\![P]\!]_G^D$ is defined as the set of mappings:*

$$[\![P]\!]_G^D = \{\mu : V \to T \mid dom(\mu) = var(P) \text{ and } \mu(P) \subseteq G\} \tag{2.1}$$

If $\mu \in [\![P]\!]_G^D$, we say that μ is a solution for P in G.

The evaluation of complex graph patterns is defined recursively:

Definition 2.2.10 (Complex Graph Pattern Matching) *Let G be an RDF graph over a RDF dataset D, and P, P_1 and P_2 basic graph patterns. The evaluation of P, P_1, and P_2 over G is defined recursively as follow:*

- $[\![P]\!]_G^D$ *is given by definition 2.2.9,*
- $[\![P_1 \text{ AND } P_2]\!]_G^D = [\![P_1]\!]_G^D \bowtie [\![P_2]\!]_G^D$,
- $[\![P_1 \text{ OPT } P_2]\!]_G^D = [\![P_1]\!]_G^D = \bowtie [\![P_2]\!]_G^D$,
- $[\![P_1 \text{ UNION } P_2]\!]_G^D = [\![P_1]\!]_G^D \cup [\![P_2]\!]_G^D$,
- $[\![P_1 \text{ FILTER } C]\!]_G^D = \{[\![P_1]\!]_G^D | \mu \vDash C\}$,
- $[\![u \text{ GRAPH } P]\!]_G^D = [\![P]\!]_{gr(u)_D}^D$
- $[\![?X \text{ GRAPH } P]\!]_G^D = \bigcup_{v \in names(D)} ([\![P]\!]_{gr(u)_D}^D = \bowtie \mu_{X \to v})$ *where $\mu_{X \to v}$ is a mapping such that $dom(\mu) = X$ and $\mu(X) = v$.*

Finally, [Arenas et al., 2009] defines the notion of equivalence for graph patterns.

Definition 2.2.11 (Equivalence of Graph Patterns) *Two graph pattern expressions $P1$ and $P2$ are equivalent, denoted by $P1 \equiv P2$, if $[\![P1]\!]_G^D = [\![P2]\!]_G^D$ for every graph G and RDF dataset D.*

Proposition 2.2.1 *Let $P1$, $P2$ and $P3$ be graph pattern expressions and R a built-in condition then:*

- *AND and UNION are associative and commutative.*
- *(P1 AND (P2 UNION P3)) ≡ ((P1 AND P2) UNION (P1 AND P3))*
- *(P1 OPT (P2 UNION P3)) ≡ ((P1 OPT P2) UNION (P1 OPT P3)).*
- *((P1 UNION P2) OPT P3) ≡ ((P1 OPT P3) UNION (P2 OPT P3)).*
- *((P1 UNION P2) FILTER R) ≡ ((P1 FILTER R) UNION (P2 FILTER R)).*

2.2.3. The Ontology Web Language

The Web Ontology Language (OWL) is a W3C Recommendation ontology language for the Semantic Web with more expressive semantics than RDFS [Dan Brickley and R.V. Guha, 2014] and a formally defined meaning [Deborah L. McGuinness and Frank van Harmelen, 2004, Christine Golbreich and Evan K. Wallace and Peter F. Patel-Schneider, 2012]. OWL can also be serialized as RDF triples. In fact, OWL re-uses the core RDFS vocabulary, but adds new vocabulary for defining classes, properties, individuals, and data values. In the next table we summarize a small subset of DL description that can be formed with OWL (2). The first column gives OWL abstract syntax for the construction, while the second column gives the equivalent Description Logic syntax. Note that the letters A and R represent, respectively, names for classes (concepts) and object properties (abstract roles); C, possibly subscripted, represents an arbitrary class description.

DL Syntax	Abstract Syntax
A	A
$\top$	`owl:Thing`
$\bot$	`owl:Nothing`
$C_1 \sqcap \ldots \sqcap C_n$	intersectionOf(C_1...C_n)
$C \sqcup \ldots \sqcup C_n$	unionOf(C_1...C_n)
$\neg C$	complementOf(C)
$\forall R.C$	restriction(R allValuesFrom(C))
$\exists R.C$	restriction(R someValuesFrom(C))
$\geq nR$	restriction(R maxCardinality(n))
$\leq nR$	restriction(R minCardinality(n))

Later in this thesis, we present an approach for reasoning with provenance information (see Chapter 4). Even though it is generic enough to be applicable beyond OWL, we focus on an extension of the description logic DL, called $\mathcal{SRIQ}(\mathcal{D})$, underlying OWL lite and OWL DL. Hence, we briefly revisit the definition of $\mathcal{SRIQ}(\mathcal{D})$, and the fundamental reasoning problems related to DLs. For a complete definition of SRIQ, refer to Horrocks et al. [Horrocks et al., 2005, Horrocks et al., 2006], and for Description Logics refer to [Baader et al., 2003].

Definition 2.2.12 (Vocabulary) *A vocabulary* $V = (N_C, N_R, N_I)$ *is a triple where*

- N_C *is a set URIs used to denote classes,*
- N_R *is a set URIs used to denote roles and*
- N_I *is a set URIs used to denote individuals.*

N_C, N_R, N_I *need not be disjoint.*

An interpretation grounds the vocabulary in objects from an object domain.

Definition 2.2.13 (Interpretation) *Given a vocabulary* V*, an* interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}_C}, \cdot^{\mathcal{I}_R}, \cdot^{\mathcal{I}_I})$ *is a quadruple where*

- $\Delta^{\mathcal{I}}$ *is a nonempty set called the object domain;*
- $\cdot^{\mathcal{I}_C}$ *is the class interpretation function, which assigns to each class a subset of the object domain* $\cdot^{\mathcal{I}_C} : N_C \to 2^{\Delta^{\mathcal{I}}}$
- $\cdot^{\mathcal{I}_R}$ *is the role interpretation function, which assigns to each role a set of tuples over the object domain* $\cdot^{\mathcal{I}_P} : N_P \to 2^{(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})}$*;*
- $\cdot^{\mathcal{I}_i}$ *is the individual interpretation function, which assigns to each individual* $a \in N_I$ *an element* $a^{\mathcal{I}_I}$ *from* $\Delta^{\mathcal{I}}$*.*

Let $C, D \in N_C$*, let* $R, R_i, S \in N_R$ *and* $a, a_i, b \in N_I$*. We extend the role interpretation function* $\cdot^{\mathcal{I}_R}$ *to role expressions:*

$$(R^-)^{\mathcal{I}_R} = \{(\langle x, y\rangle) | (\langle y, x\rangle) \in R^{\mathcal{I}_R}\}$$

We extend the class interpretation function $\cdot^{\mathcal{I}_C}$ *to class descriptions:*

top	$\top^{\mathcal{I}_C}$	$\equiv \Delta^{\mathcal{I}}$
bottom	$\bot^{\mathcal{I}_C}$	$\equiv \varnothing$
disjuction	$(C \sqcap D)^{\mathcal{I}_C}$	$\equiv C^{\mathcal{I}_C} \cap D^{\mathcal{I}_C}$
conjuction	$(C \sqcup D)^{\mathcal{I}_C}$	$\equiv C^{\mathcal{I}_C} \cup D^{\mathcal{I}_C}$
negation	$(\neg C)^{\mathcal{I}_C}$	$\equiv \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}_C}$
value restriction	$(\forall R.C)^{\mathcal{I}_C}$	$\equiv \{x \in \Delta^{\mathcal{I}} \mid \langle x, y\rangle \in R^{\mathcal{I}_R} \rightarrow y \in C^{\mathcal{I}_C}\}$
exist restriction	$(\exists R.C)^{\mathcal{I}_C}$	$\equiv \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : \langle x, y\rangle \in R^{\mathcal{I}_R}\}$
self restriction	$(\exists R.Self)^{\mathcal{I}_C}$	$\equiv \{x \in \Delta^{\mathcal{I}} \mid \langle a, a\rangle \in R^{\mathcal{I}}\}$
atleast restriction	$(\geq nR)^{\mathcal{I}_C}$	$\equiv \{x \in \Delta^{\mathcal{I}} \mid \exists y_1, ..., y_m \in \Delta^{\mathcal{I}} :$ $\langle x, y_1\rangle, ..., \langle x, y_m\rangle \in R^{\mathcal{I}_R} \wedge m \geq n\}$
atmost restriction	$(\leq nR)^{\mathcal{I}_C}$	$\equiv \{x \in \Delta^{\mathcal{I}} \mid \nexists y_1, ..., y_m \in \Delta^{\mathcal{I}} :$ $\langle x, y_1\rangle, ..., \langle x, y_m\rangle \in R^{\mathcal{I}_R} \wedge m > n\}$

Class expressions are used in axioms.

Definition 2.2.14 (Axiom) *An axiom is one of the following:*

- *A general concept inclusion of the form $C \sqsubseteq D$ for concepts C and D;*
- *An individual assertion of one of the forms $a : C$, $(a, b) : R$, $(a, b) : \neg R$, $a = b$ or $a \neq b$ for individuals a, b and a role R.*
- *A role assertion of one of the forms $R \sqsubseteq S$, $R_1 \circ ... \circ R_n \sqsubseteq S$, $Asy(R)$, $Ref(R)$, $Irr(R)$, $Dis(R, S)$ for roles R, R_i, S.*

The following table shows a small subset of the features provided by OWL (2) associated with the equivalent Description Logic syntax.

DL Syntax	Abstract Syntax
$R_1 \sqsubseteq R_2$	SubPropertyOf($R_1 R_2$)
$C_1 \sqsubseteq C_2$	SubClassOf(C_1 C_2)
$o \in C$	ClassAssertion(C o)
$o_1 \equiv \ldots \equiv o_n$	SameIndividual(o_1 $\ldots$ o_n)

Example 2.2.4 *Continuing with our running example about movies, here we present a simple exhibition of some features of OWL. The first statement defines that the entity named* `:Star_Wars:_The_Force_Awakens` *is a film. The second one stated that the entity* `:Harrison_Ford` *is an actor. With the following statements, we specify relationships between the entities by indicating that* `:Harrison_Ford` *has acted in the* `Star_Wars:_The_Force_Awakens` *movie.*

ClassAssertion(`:Film :Star_Wars:_The_Force_Awakens`*)*
ClassAssertion(`:Actor :Harrison_Ford`*)*
ObjectPropertyDomain(`:act :Film`*)*
ObjectPropertyRange(`:act :Actor`*)*
ObjectPropertyAssertion(`:act :Star_Wars:_The_Force_Awakens :Harrison_Ford`*)*

We now define satisfaction of axioms.

Definition 2.2.15 (Satisfaction of Axioms)
Satisfaction of axioms in an interpretation $\mathcal{I}$ is defined as follows. With $\circ$ we denote the composition of binary relations.

$$
\begin{aligned}
(R \sqsubseteq S)^{\mathcal{I}} &\equiv \langle x, y\rangle \in R^{\mathcal{I}} \rightarrow \langle x, y\rangle \in S^{\mathcal{I}} \\
(R_1 \circ ... \circ R_n \sqsubseteq S)^{\mathcal{I}} &\equiv \forall \langle x, y_1\rangle \in R_1^{\mathcal{I}}, \langle y_1, y_2\rangle \in R_2^{\mathcal{I}}, ..., \\
&\quad \langle y_{n-1}, z\rangle \in R_n^{\mathcal{I}} : \langle x, z\rangle \in S^{\mathcal{I}} \\
(Asy(R))^{\mathcal{I}} &\equiv \exists \langle x, y\rangle \in R^{\mathcal{I}} : \langle y, x\rangle \notin R^{\mathcal{I}} \\
(Ref(R))^{\mathcal{I}} &\equiv \forall x \in \Delta^{\mathcal{I}} : \langle x, x\rangle \in R^{\mathcal{I}} \\
(Dis(R, S))^{\mathcal{I}} &\equiv R^{\mathcal{I}} \cap S^{\mathcal{I}} = \varnothing \\
(Irr(R))^{\mathcal{I}} &\equiv \forall x \in \Delta^{\mathcal{I}} : \langle x, x\rangle \notin R^{\mathcal{I}} \\
(C \sqsubseteq D)^{\mathcal{I}} &\equiv x \in C^{\mathcal{I}} \rightarrow x \in D^{\mathcal{I}} \\
(a : C)^{\mathcal{I}} &\equiv a^{\mathcal{I}} \in C^{\mathcal{I}} \\
((a, b) : R)^{\mathcal{I}} &\equiv \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}} \\
((a, b) : \neg R)^{\mathcal{I}} &\equiv \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \notin R^{\mathcal{I}} \\
a = b &\equiv a^{\mathcal{I}} = b^{\mathcal{I}} \\
a \neq b &\equiv a^{\mathcal{I}} \neq b^{\mathcal{I}}.
\end{aligned}
$$

An ontology is comprised of a set of axioms.

Definition 2.2.16 (Ontology) *A $\mathcal{SRIQ}(\mathcal{D})$ ontology O is a set of axioms as defined in definition 2.2.14.*

In this work we make use of two basic reasoning mechanisms for $\mathcal{SRIQ}(\mathcal{D})$, i.e. consistency checking and axiom entailment.

Definition 2.2.17 (Reasoning with $\mathcal{SRIQ}(\mathcal{D})$) *We say an interpretation $\mathcal{I}$ of an ontology O is a model of O ($\mathcal{I} \vDash O$), if all axioms in O are satisfied in I. We say an ontology O is consistent if there exists a model of O. We say an axiom A is entailed by an ontology O ($O \vDash A$), if A is satisfiable in all models of O.*

Obviously, a subontology again is an ontology. Moreover, if $O' \subseteq O$ and $O' \vDash A$, then also $O \vDash A$.

Reasoning is important for using and working with ontologies. The reasoning task of computing the individuals that belong to a given (set of) class(es) is called instance retrieval. Instance checking is used when one wants to find out whether one particular individual belongs to the given class, and subsumption checking to check whether one particular individual is subsumed by the given class — computing all subclass relationships between a set of classes is called classification, and checking a particular subclass relationship is called subsumption checking.

Another important reasoning task is consistency checking which is used to check whether an ontology is logically consistent or contradictory. An inconsistent ontology entails every axiom, and is therefore of no practical use. In the following we show an inconsistency detection example.

Example 2.2.5 *Suppose we start out with the ontology from our Example 2.2.4. We extend this initial example extended with the axiom:*

$$\mathit{DisjointClasses}(\texttt{:Actor :FilmProducer})$$

i. e. Film producers and Actors are disjoint. This ontology is logically consistent. Now we add the axiom:

$$\mathit{ClassAssertion}(\texttt{:FilmProducer :Harrison\_Ford})$$

stating that the entity `:Harisson_Ford` *is a film producer. Obviously, this results in an inconsistent ontology, since we have stated that actors and film producers are disjoint.*

2.2.4. Linked Data: Principles and Best Practices

Linked Data, the publication of data on the Web, refers to a set of principles and best practices by which the Semantic Web standards (RDF, OWL, SPARQL, etc) can be effectively deployed on the Web. Linked Data aims to facilitate access to structure data on the Web and the easy reuse of this data. We will briefly introduce some concepts related to Linked Data; for more information please refer to [Hogan, 2014, Bizer et al., 2009]

Linked Data relies on two technologies that are fundamental to the Web: Uniform Resource Identifiers (URIs) [T. Berners-Lee, 2005] and the Hypertext Transfer Protocol (HTTP) [R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, T. Berners-Lee, 1999]. The HTTP protocol is used to access a resource identified by an Uniform Resource Identifier (URI) (simply by dereferencing the URI over the HTTP protocol) and to retrieve structure data (RDF) about this resource. The structure data about that resource can provide links to other resources enabling further discovery, thus forming the Web of Data.

Example 2.2.6 *The data used in our toy example so far has been taken from DBpedia:*

```
# http://dbpedia.org/data/Man_on_a_Ledge.xml
dbr:Man_on_a_Ledge                          rdfs:label   "Man_on_a_Ledge"@en ;
dbp:runtime                                 6120;
rdf:type                                    dbo:Film;
...
```

The first document, obtained from the Linked Data source DBpedia looking up the URI `http://dbpedia.org/data/Man_on_a_Ledge.xml` *using dereferencing, is about the movie* `dbr:Man_on_a_Ledge`*. This means the resource* `dbr:Man_on_a_Ledge` *is of type* `dbo:Film`*. Information about the resource* `dbo:Film` *is available in a different document displayed below. Thus, the* `rdf:type` *describes a link between these documents. By looking up the URI using dereferencing, we can retrieve this document about* `dbo:Film`*.*

```
# http://dbpedia.org/data/Film.xml
dbo:Film                                    rdfs:label      "Film"@en;
owl:sameAs                                  de_dbo:Filme    ;
...
```

Berners-Lee introduced the Linked Data principles in the initial W3C Design Issues document [Berners-Lee, 2006]. The four Linked Data principles are as follows:

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

In summary, the Linked Data best practices lay the foundations for extending the Web with a global data space based on the same architectural principles as the classic document Web and connecting data from diverse domains such as people, companies, books, scientific publications, films, music, television and radio programmers; genes, proteins, drugs and clinical trials; online communities, statistical and scientific data.

2.3. The Use of Provenance in Semantic Web Applications

On the Semantic Web, a large amount of work on provenance has already been carried out and has been quite diverse. In a 2007 publication, Sergej Sizov pointed out in [Sizov, 2007] the importance of provenance information in the Semantic Web as a tool to assess the veracity of knowledge. Sizov [Sizov, 2007] argued that provenance information contributes to the Semantic Web layer cake's proof layer as it delivers explanations of where the presented knowledge comes from and how the resulting conclusions have been constructed (see Figure 2.3). Many surveys on provenance have been published since then which highlight the foundations, challenges and opportunities of managing provenance in the Semantic Web [Moreau, 2010, Freire, 2009, Lakshmanan et al., 2011, Bienvenu et al., 2012]. In the next section, we present a literature review of the work on provenance on the Semantic Web.

2.3.1. Recent Work on Provenance in the Semantic Web

Representation and Querying: Recent work on provenance in the Semantic Web domain includes representational mechanisms for provenance and ways of using it. For example, in [Carroll et al., 2005] the authors propose an application of so-called Named RDF Graphs for publishing information on the Semantic Web. Information providers publish information together with provenance about its intended assertional status. An information consumer may accept some of these claims and reject others. In [Ding et al., 2005] the authors describe the use of provenance and trust information in the context of homeland security analysis. Their work concentrates on the description of an inference framework addressing the use of provenance to evaluate the trustworthiness of derived facts, and to prune search on the Semantic Web.

Provenance mechanisms for RDF datasets have been investigated by Udrea et al. [Udrea et al., 2010] where the authors provide a semantics for RDF augmented with a partially ordered set and algorithms for query processing and view maintenance. The literature describes several approaches to extract data provenance/annotated information from

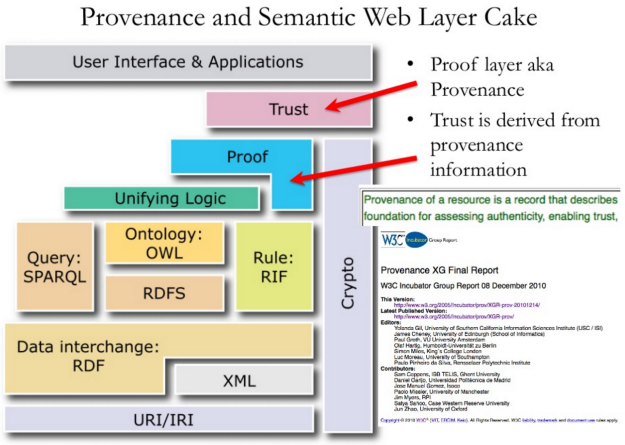


Figure 2.3.: Provenance and the Semantic Web Layer Cake. Taken from the presentation "Provenance Analysis and RDF Query Processing: W3C PROV for Data Quality and Trust" from Satya Sahoo and Praveen Rao, October 2015 .

RDF(S) data [Straccia et al., 2010, Zimmermann et al., 2012] supporting RDFS entailment. Lopes et al. [Lopes et al., 2010] go behond supporting RDFS entailment and provide a query language extending many of the SPARQL features in order to deal with annotated data, exposing annotations at query level via annotation variables, and including aggregates and subqueries.

Kwasnikowska et al. [Kwasnikowska et al., 2015] specify a formal account of the open provenance model (OPM) [Moreau et al., 2011]. The OPM is a data model for provenance that is designed to facilitate the meaningful interchange of provenance information between systems. OPM graphs are directed graphs where nodes represent data products and processes involved in past computations and edges represent dependencies between them. Kwasnikowska et al. propose the formal semantics for the OPM, by viewing OPM graphs as temporal theories on the temporal events represented in the graph. Hartig [Hartig, 2009a] proposes a specialization of the OPM, referred to as provenance vocabulary, to describe the provenance of Linked Data over the Web. His model accounts for the creation and access of RDF data. Fabrizio Orlandi and Alexandre Passant [Orlandi and Passant, 2011] use the OPM to represent provenance (in RDF) about DBpedia statements. Aldeco-Perez et al. [Aldeco-Pérez and Moreau, 2010] proposes a provenance-based compliance framework, based on the OPM. The framework provides a processing view (represented as a provenance graph for a specific execution time) and usage policy definition (UPD). It uses the UPD to validate against the processing view for compliance. The framework lacks the integration with the commercial applications and policy standard such as XACML.

The PROV data model (PROV-DM) has recently become a recommendation of the World Wide Web Consortium to represent provenance information. The PROV data model will be covered in more detail later in this chapter.

Relational Provenance Models Extensions for SPARQL Queries: In [Theoharis et al., 2011,

Geerts et al., 2013, Karvounarakis et al., 2013] Theoharis et al. and Karvounarakis et al. discuss the need for extending the relational provenance models to be leveraged for SPARQL queries over RDF. In particular, they discuss that the use of semiring models for SPARQL has been shown inadequate to handle the OPTIONAL construct, and advocate the need for a new abstract provenance model capturing the full expressiveness of SPARQL. In addition, Geerts et al. [Geerts et al., 2016] identify SPARQL fragments for which provenance models for positive relational queries can be leveraged, despite the subtle differences between the semantics of SPARQL and relational algebra operators. In [Amsterdamer et al., 2011c], the authors show particular semirings for which an extension for supporting difference is impossible. The consideration of extended provenance models for capturing the semantics of both explicit and scoped weak negation is also considered in [Analyti et al., 2014]. In [Damásio et al., 2012], Damasio et al. proposed a provenance model for a significant fragment of SPARQL 1.1, based on the relational annotated provenance models including for non-monotonic constructs under multiset semantics.

Amsterdamer et al. [Amsterdamer et al., 2012, Amsterdamer et al., 2011b] propose the notion of core provenance that is defined in a very general way (using the semiring provenance model), and its representation is minimal and compact. They study algorithms that,given a query, compute an equivalent query that realizes the core provenance for all tuples in its result. In [Deutch et al., 2015b], Deutch et al. describe an approach for web scale provenance tracking. Their approach allows selective tracking of how-provenance, where the selection criteria are partly based on the metadata itself (and significantly reducing provenance size).

Provenance Dimensions : Hartig [Hartig, 2009b] presents a trust-aware extension to SPARQL, the tSPARQL. Hartig first proposes a trust model that associates RDF statements with trust values and extends the SPARQL semantics to access these trust values in tSPARQL. In [Bonatti et al., 2011], Bonatti et al. formalize a logical framework for determining trust to perform robust reasoning. In [Schenk, 2008], Schenk models multiple levels of trust on Web data to resolve inconsistencies arising from connecting multiple data sources.

The version control aspect of provenance such as temporal RDF [Gutiérrez et al., 2005] and querying time in RDF and OWL [Motik, 2012] has been tackled by attaching temporal annotations to RDF facts and OWL axioms.

Summarization : Luc Moreau et al. [Moreau, 2015] present a summarization technique for provenance graphs. Provenance summaries can be considered as "compact users views" and have a good potential to detect anomalies and outliers. Summaries can be displayed using a minimally adapted version of the PROV Working Group diagrams [PRO, 2012]. Richardson et al. [Richardson and Moreau, 2016] have shown how the use of more sophisticated architectures for natural language generation can be applied to the task of explaining provenance graphs.

Evolving RDF datasets: Some work such as that of Flouris et al. [Flouris et al., 2009], Avgoustaki et al. [Avgoustaki et al., 2016], and Halpin et al. [Halpin and Cheney, 2014] consider evolving RDF datasets. Particularly Flouris et al. [Flouris et al., 2009] consider the problem of how to maintain provenance information for RDFS inferences when tuples

are inserted or deleted, using coherence semantics. Avgoustaki et al. [Avgoustaki et al., 2016] introduce a provenance model that borrows from both where and how provenance models and is suitable for encoding both triple and attribute level provenance of quadruples obtained via INSERT updates, and allows the reconstructability of such updates from their provenance. Halpin et al. [Halpin and Cheney, 2014] consider a provenance model for SPARQL queries and updates to data stores involving named graphs, whose purpose is to provide a record of how the raw data in a dataset has changed over time, and then tackle the problem of recording and providing (queryable) access to the detailed change history of an RDF graph that is updated over time via SPARQL updates.

Crowdsourcing : In order to handle Linked Data quality problems, different approaches have been proposed in the literature. See [Dragan et al., 2015, Keshavarz et al., 2014, Zhao and Hartig, 2012, Huynh et al., 2013]. Basically, they make use of metadata which are captured by provenance frameworks (using specific provenance vocabulary) or by crowdsourcing. Such kinds of metadata are often provided by agents (applications and/or users) which manually analyze the data on the basis of given criteria.

Storage : In [Wylot et al., 2014, Wylot et al., 2015a], Wylot et al. present the TripleProv, a native RDF store that allows tracking and querying provenance over Web Data. In later work [Wylot et al., 2015b], Wylot et al. use the TripleProv store to investigate the effectiveness of different query execution strategies for provenance-enabled queries.

Provenance in workflow systems : Eckert et al. [Eckert et al., 2014] present a framework which integrates all phases of a typical workflow lifecycle, including the specification of services, their composition to workflows as well as the execution of the workflows. With the specified ontology, all resources in this lifecycle are described. Provenance of all resources can be tracked and to expose the provenance using W3C PROV-DM [Belhajjame et al., 2013a]. The RDFProv [Chebotko et al., 2010] system focused on managing and enabling querying over provenance that results from scientific workflows. Biton et al. [Biton et al., 2007] showed how user views can be used to reduce the amount of information returned by provenance queries in a workflow system. In [Hoekstra and Groth, 2015], Hoekstra et al. propose Sankey diagrams that are used to represent provenance in a process-centric way.

2.3.2. An Introduction to PROV standard

PROV, the provenance standard, is a family of specifications which has recently been released by the Provenance Working Group. PROV aims to define a generic data model for provenance that can be extended in a principled way to suit many application areas. Here we provide a short overview of PROV. For more details, please refer to [Moreau and Groth, 2013, Moreau et al., 2015].

The PROV family of documents is a set of specifications allowing provenance to be modeled, serialized, exchanged, accessed, merged, translated, and reasoned over, i.e., they define a data model and an OWL ontology, along with a number of serializations for representing aspects of provenance. The term provenance, as understood in these specifications, refers to information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness (PROVOverview [Groth and Moreau, 2013]).

Recommendation documents include (i) the main PROV data model specification (PROV-DM [Belhajjame et al., 2013a]), with an associated set of constraints and inference rules (PROV-CONSTRAINTS [Nies, 2013]); (ii) an OWL ontology that allows a mapping of the data model to RDF (PROV-O [Belhajjame et al., 2013b]), and (iii) a notation for PROV with a relational-like syntax, aimed at human consumption (PROV-N [Cheney and Soiland-Reyes, 2013]). All other documents are Notes documents. These include PROV-XML, which defines a XSD schema for XML serialization [Moreau, 2013]. PROV-AQ, the Provenance Access and Query document [Moreau et al., 2013], which defines a Web-compliant mechanism to associate a dataset to its provenance; PROV-DICTIONARY [Missier et al., 2013], for expressing the provenance of data collections defined as sets of key-entity pairs; and PROV-DC [Miles et al., 2013], which provides a mapping between PROV-O and Dublin Core Terms.

PROV-DM provides an operational definition of provenance for the community to use and is organized into six components respectively, dealing with: (1) entities and activities, and the time at which they were created, used, or ended; (2) derivations of entities from entities; (3) agents bearing responsibility for entities that were generated and activities that happened; (4) a notion of bundle, a mechanism to support provenance of provenance; and, (5) properties to link entities that refer to the same thing; (6) collections forming a logical structure for its members.

The purpose of standardization of PROV was to specify a comprehensive and minimum set of constructs that could easily address many areas of application. PROV is designed to be extensible; therefore, it is expected that with more implementations of, and application using, the PROV standards, more concepts (such as dictionary, mention, representation and mappings) may appear. For instance, Trung Dong Huynh et al. [Huynh et al., 2016] propose a new representation for PROV based on JSON-LD. Given its compatibility with both JSON and the RDF data model, in addition to being amenable to stream processing, PROV-JSONLD has a vast range of useful applications. PROV-JSONLD enables the adoption of PROV as the provenance standard of choice in Linked Data and Web applications alike. Furthermore, in previous work, Trung Dong Huynh et al. [Huynh and Moreau, 2014] presented the ProvStore - a public provenance repository. ProvStore is an online public provenance repository supporting the PROV standards. It allows users and applications to store and (optionally) publish the provenance of their data on the Web.

In this thesis we do not use the PROV data model and ontology for representing and exchanging provenance. This work focuses on providing a semantics for querying, reasoning and efficiently handling provenance data within RDF database systems as well as dynamics ontologies. Indeed, we provide representational mechanisms for provenance, nevertheless our work could incorporate (parts of) the PROV standards.

Part II.

Provenance Management in the Semantic Web

3. Querying RDF Datasets with Provenance

Overview

The Semantic Web is based on accessing and reusing RDF data from many different sources, which one may assign different levels of authority and credibility. Existing Semantic Web query languages, like SPARQL, have targeted the retrieval, combination, and reuse of facts, but have so far ignored all aspects of *provenance*, such as origins, authorship, recentness or certainty of data. In this chapter, we present an original, generic, formalized and implemented approach for managing many dimensions of provenance, like source, authorship, certainty, among others. The approach re-uses existing RDF modeling possibilities in order to represent provenance. Then, it extends SPARQL query processing in such a way that given a SPARQL query for data, one may request provenance without modifying the query proper. Thus, our approach achieves highly flexible and automatically coordinated querying for data and provenance, while completely separating the two areas of concern.

Structure

The structure of this chapter is as follows: Section 3.2 presents a motivation scenario that emphasizes the need for provenance within the query processing. In Section 3.3, we start to explain our approach with a discussion of important design choices. We model provenance in existing RDF structures by embedding a slightly more expressive language, which we call RDF$^+$, into RDF. Later, we define the abstract syntax of RDF$^+$, its semantics and its embedding in RDF.

In Section 3.4, we present our approach to the treatment of provenance when querying Semantic Web data by extending the SPARQL syntax and semantics to work on data and provenance of RDF$^+$. The extension allows the user to extend a given conventional SPARQL query by a keyword for provenance, triggering the construction of provenance by the query processor. We demonstrate in Section 3.5 the equivalences holding between the standard evaluation of SPARQL queries and the evaluation with provenance. Section 3.6 summarizes the previously discussed steps of provenance representation and utilization for the sample scenario that was introduced in Section 3.2.

We report on initial promising results for provenance processing from a theoretic point of view in Section 3.7 and provide pointers to the prototype implementation of the system and initial experimental testing in Section 3.8. Section 3.9 reviews existing research on querying and ranking RDF data and the use of provenance in the Semantic Web. Section 3.10 concludes this chapter.

3.1. Introduction

Integrating and re-using Semantic Web data is proving to be a more and more fruitful method of answering questions and delivering results. Typically, engines like Glimmer[1], Swoogle[2], and Datahub [3] provide points of access for RDF data, and query languages like SPARQL with their corresponding query engines allow for selecting and re-using data in the appropriate format. When querying the Web, however, we are faced with a highly variable quality of information. However, with the increasing data available on the Web and more sophisticated processing through query and reasoning engines, one now encounters challenging questions linked to provenance about the data like:

- *Where* is this data from?
- *Who* provided the data?
- *When* was this data provided?
- Was the provider certain about the *truth* of this data?
- Has the data been verified by others?

Provenance provides knowledge that can be used to quantify this value, and may come in different forms, e.g., trustworthiness of a source, time of validity, certainty and absence of vagueness. Therefore, techniques to find the relevant information out of the Web data should include ways to investigate the value of information.

For instance, when querying the Semantic Web with the help of SPARQL for the occupation of a person named 'James Cameron', one finds (at least) two answers: 'Film Producer' and 'Truck Driver'. Without further indication as to **where**, **by whom**, **when**, etc. such information was given, it is impossible to decide which of the two occupations is still valid.

The problem might be remedied in several ways. First, an idiosyncratic solution by the search engine, such as returning the corresponding RDF files or links to sources of knowledge extraction (say `http://dbpedia.org/data/James_Cameron.xml` and `http://www.myspace.com/deprecated.doc`), might help in this special case. However, an idiosyncratic solution may not be appropriate in a second case in which the **when** was more relevant than the **where** or even in a third case where such a piece of information had to be aggregated from several resources and multiples **where**, **who**, and **when** have been retrieved.

Second, the person or system requesting the provenance information might manually extend the SPARQL query formalizing the request for the occupation in order to return the **where**, the **who** and the **when**. Such a modification will, however, be very tedious, as it will include a number of additional optional statements, and expressing it manually will be error prone. Also, it will not help in delivering provenance that arises from joining several statements, e. g. provenance that was extracted from several statements with different provenance values.

Therefore, querying Semantic Web data requires a principled, generic approach to the treatment of provenance that is able to adapt to its many dimensions and that is open to

[1] YAHOO!Glimmer:`http://glimmer.research.yahoo.com/`
[2] Swoogle Semantic Web Search: `http://swoogle.umbc.edu/`
[3] Datahub: `https://datahub.io/dataset`

accommodate itself to new dimensions when the need arises. Such a principled, original framework is given in this chapter. The approach re-uses existing RDF modeling possibilities in order to represent provenance. Then, it extends SPARQL query processing in such a way that given a SPARQL query for data, one may request provenance without modifying the query proper. Thus, our approach achieves highly flexible and automatically coordinated querying for data and provenance, while completely separating the two areas of concern.

Research Questions

This chapter addresses the following research questions:

RQ 1.I *Can RDF be used to represent provenance on its different dimensions e. g., source, certainty, and timestamp?*

RDF plays the role of a common model, as data can easily be integrated and combined with other data published on the Web, which is basically done by establishing links between Web resources. Provenance should then be embedded in RDF in such a way that it retains upward compatibility with existing usage of the language and corresponding tools and methods, which is a major concern for Semantic Web approaches.

RQ 1.II *Does the treatment of provenance within the SPARQL query language allow for changing the existing SPARQL semantics (thus leaving it unsupported by existing query engines)?*

The SPARQL query language enables querying in interlinked data space (RDF graphs) that goes beyond simple keyword searches. SPARQL mechanisms, however, do not include techniques to find the relevant information out of the Web data. Therefore, query engines need mechanisms to track provenance in its many dimensions when exploring data.

RQ 1.III *Does the exploitation of provenance lead to computation overhead?*

In general, provenance information can grow to be larger than the data it describes if the data is fine-grained and provenance information rich. So the manner in which the provenance is propagated along the computation is crucial to its scalability.

3.2. Motivation Scenario

In our scenario, the sample user aims to explore the knowledge and provenance information by querying knowledge extracted from the Web. Here, we assume that he aims to find movies and movie directors. Example 3.2.1 shows the relevant facts that may have been obtained from different websites.

Example 3.2.1 *Relevant facts obtained from different websites*

Site A	*JamesCameron's occupation is Film Producer.* *JamesCameron produces* Avatar.
Site B	*JamesCameron's occupation is Truck Driver.* *JamesCameron produces* Battle Beyond the Stars. *Martin Scorsese's occupation Film Producer.* *Martin Scorsese produces Taxi Driver.*

In addition, the user wants to exploit provenance information for obtaining results with best certainty and for analyzing contradicting answers (e. g. different occupations for the same person 'James Cameron' in Example 3.2.1). An example of provenance associated with the extracted facts is presented in Example 3.2.2 and shows that the facts have been extracted from different data sources, at different timepoints, and with different degrees of extraction confidence.

Example 3.2.2 *Associated provenance of facts presented in Example 3.2.1*

Site A	*source = dbpedia.org/data/JamesCameron.xml* *certainty degree = 0.9* *timestamp = 5/5/2014*
Site B	*source = myspace.com/deprecated.doc* *certainty degree = 0.6* *timestamp = 6/6/1980*

3.3. Syntax and Semantics for RDF with Provenance

In order to adapt our sample application scenario to our framework, we formalize it using the notion of Named RDF Graphs [Carroll et al., 2005, Carroll and Stickler, 2004]. We assume that the user utilizes knowledge which has been initially extracted from different Web pages and stored in form of RDF triples locally. Example 3.3.1 shows the relevant facts already presented in Example 3.2.1 in RDF triple language TriG [Bizer and Cyganiak, 2014] with Named Graphs in a simplified form that abstracts from (default) namespaces.

Example 3.3.1 *Facts of Example 3.2.1 represented in TriG with Named Graphs*

G1	{*JamesCameron occupation FilmProducer.* *JamesCameron produces Avatar*}
G2	{ *JamesCameron occupation TruckDriver.* *JamesCameron produces BattleBeyondtheStars.* *Martin Scorsese occupation FilmProducer.* *Martin Scorsese produces TaxiDriver*}

Likewise, the provenance associated to the extracted knowledge presented in Example 3.2.2 is stored into the same RDF repository using the notion of Named RDF Graphs.

Example 3.3.2 *Provenance of Example 3.2.2 represented in TriG with Named Graphs*

G3	{ *G1 prov:source* <dbpedia.org/data/James_Cameron.xml>. *G1 prov:certainty '0.9'.* *G1 prov:timestamp "5/5/2014"* }
G4	*G2 prov:source* <www.myspace.com/deprecated.doc>. *G2 prov:certainty '0.6'.* *G2 prov:timestamp "6/6/1980"* }

In the examples above we have used existing RDF modeling possibilities in order to represent provenance. However we must distinguish the notation of RDF with only *implicit* notation of provenance, but no semantic consequences specifically due to this provenance, from a formally extended model of RDF with *explicit* notation of provenance.

In the course of representing and reasoning with provenance we embed a language with provenance reasoning, i.e., RDF$^+$, into a language without such specific facilities, i. e., in RDF. This embedding implies that we may consider an RDF snippet in its literal sense *and* we may possibly interpret it as making a provenance statement. Embedding provenance in RDF is not the most expressive means to deal with all the needs of provenance processing, but it retains upward compatibility with existing usage of the language and corresponding tools and methods, which is a major concern for Semantic Web approaches. The following definition of RDF$^+$helps us to draw this line very clearly and concisely. The abstract syntax for this embedded language, RDF$^+$, is given in Section 3.3.2 and its semantics in Section 3.3.3. Then we show how to embed RDF$^+$in RDF with named graphs.

3.3.1. RDF Design Choices

This section summarizes and shortly motivates the design choices for our provenance framework.

Reification

Establishing relationships between knowledge and provenance requires appropriate reification mechanisms for supporting statements about statements. Our general objective is to execute queries on original data (i. e. without provenance) directly without complex transformations. For compliance with existing applications that access the repository in a common way (e. g. using SPARQL queries), we do not modify existing user data. This requirement does not allow us to use mechanisms like RDF reification, which decompose existing triples and fully change the representational model. In our framework described in section 3.3, we adopt the notion of Named RDF Graphs for provenance representation [Carroll et al., 2005, Carroll and Stickler, 2004].

Storage Mechanisms

Following the overall philosophy of RDF, we do not separate provenance from user knowledge in the repository. Following this paradigm, a user or developer has unlimited access to all contents of the triple store and can manipulate provenance directly. In other words, the user can directly access provenance (e. g. using suitable SPARQL queries). Beyond explicitly designed queries for provenance access, in Section 3.4 we describe the extension of SPARQL that allows us to access provenance about the result set automatically without user intervention.

Provenance Dimensions

An important point for the application design is the definition of relevant provenance dimensions and their suitable interpretation for arbitrarily complex query patterns. In general, these dimensions are application dependent and must be carefully chosen by the system administrator. In our scenario (Section 3.2 and Section 3.6) we discuss common and widely used

dimensions, such as timestamp, source, and (un)certainty, and show ways of defining and utilizing them in our framework.

Syntax Extensions

Seamlessly integrated access to provenance requires corresponding extensions of existing querying mechanisms. These can be realized at different levels, for instance at the level of query languages (e. g. SPARQL) or at the level of application-specific interfaces (e. g. Sesame API). In Section 3.4 we describe our SPARQL extension for constructing query results with associated provenance. It is system-independent and not related to some particular implementation of the RDF repository. Furthermore, it fully supports the existing SPARQL syntax and semantics. Compliance with existing established standards makes the integration with existing applications and interfaces substantially easier.

3.3.2. Syntax for RDF with Provenance

The abstract syntax of RDF$^+$is based on the same building blocks as RDF:

- U are Uniform Resource Identifiers (URIs).
- L are all RDF literals.
- $G \subseteq U$ is the set of graph names.
- $P \subseteq U$ is the set of properties.

In addition, we must be able to refer to statements directly without use of reification. For this purpose, we exploit the notion of (internal) unique statement identifiers:

- Θ is a set of statement identifiers, which is disjoint from U and L.

Now, we may define RDF$^+$literal statements that are placed in named graphs and have, in addition to RDF, a globally unique statement identity.

Definition 3.3.1 (RDF$^+$ Literal Statements) *The set of all RDF$^+$ literal statements, $\mathfrak{S}$, is defined as quintuples by: $\mathfrak{S} := \{(g, s, p, o, \theta) \mid g \in G,\ s \in U,\ p \in P,\ o \in U \cup L,\ \theta \in \Theta\}$. Thereby, θ and (g, s, p, o) are keys such that there exists a bijection f_1 with $f_1(g, s, p, o) = \theta \wedge f^{-1}(\theta) = (g, s, p, o)$. Moreover, we define the overloaded function f_5 to return the complete quintuple given either θ or (g, s, p, o), i. e. $f_5(\theta) := (g, s, p, o, \theta) =: f_5(g, s, p, o)$, when $f_1(g, s, p, o) = \theta$.*

The reader may note that we assume that f_1 is fixed and given before any statement is defined. Furthermore, this definition of literal statements and the rest of this work abstracts from RDF blank nodes in order to keep the formalization more concise. However, there is no conceptual problem in extending our treatment to blank nodes, too. The two statements of Graph G1 of Example 3.3.1 may now be represented in RDF$^+$in the following way:

Example 3.3.3 *Knowledge statements in RDF$^+$*

$\mathfrak{S} \supseteq K \supseteq \{$
(G1, JamesCameron, occupation, FilmProducer, θ_1),
(G1, JamesCameron, produces, Avatar, θ_2)}

Thereby, the exact form of statement identifiers in Θ is up to the implementation, as they are only used for internal processing.

Having represented the literal interpretation of RDF statements in RDF$^+$, we may now address the representation of selected RDF statements as RDF$^+$provenance. This is done using a structure of RDF$^+$provenance statements, $\mathfrak{M}$, that is separate from the set of RDF$^+$literal statements:

Definition 3.3.2 (RDF$^+$ Provenance Statements) *Let $\Gamma \subseteq P$ be the set of provenance dimensions (with designated semantics that is defined later). Let Ω_γ, with $\gamma \in \Gamma$, be sets providing possible value ranges for the provenance dimension $\gamma \in \Gamma$. Then, the set of all RDF$^+$ provenance statements, $\mathfrak{M}$, is defined by: $\mathfrak{M} := \{(\theta, \gamma, \omega) \mid \theta \in \Theta, \gamma \in \Gamma, \omega \in \Omega_\gamma\}$.*

The following example illustrates the target representation of the first two provenance statements of graph *G3* from Example 3.3.3.

Example 3.3.4 *Provenance statements in RDF$^+$*

$\mathfrak{M} \supseteq M \supseteq \{$
(θ_1, prov:source, <dbpedia.org/data/James_Cameron.xml>).
(θ_1, prov:certainty, '0.9')}

Together we may now define a RDF$^+$dataset.

Definition 3.3.3 (RDF$^+$Dataset) *A RDF$^+$dataset D^+ of literal statements and associated provenance statements is a pair $D^+ = (K, M)$ referring to a set of literal statements $K \subseteq \mathfrak{S}$ and a set of provenance statements $M \subseteq \mathfrak{M}$.*

A (partial) example for such a dataset given by the pair (K, M) with definitions for $K \subseteq \mathfrak{S}$ and $M \subseteq \mathfrak{M}$ has been given in Example 3.3.3 and Example 3.3.4, respectively. Finally, we define a RDF$^+$dataset.

3.3.3. Semantics for RDF with Provenance

We now have an abstract syntax for representing RDF triples like *"JamesCameron occupation FilmProducer"* as part of *G1* and provenance statements like *"the source of the statement that James Cameron's occupation is Film Producer is found in the document* `dbpedia.org/data/James_Cameron.xml` *"*.

However, such an abstract syntax may remain remarkably ambiguous if it cannot be linked to a formal semantics. Assume two provenance statements:

(θ_1, prov:source, <dbpedia.org/data/James_Cameron.xml>)
(θ_1, prov:source, <http://www.imdb.com/name/nm0000116.xml>)

For the same literal statement identified by θ_1, the question may arise whether this means a disjunction, i. e. one of the two documents has provided the fact, or a conjunction, i. e. both documents have provided the fact, or a collective reading, i. e. the two documents together gave rise to the fact, or whether this situation constitutes invalid provenance. In order to prevent such ambiguities we introduce a generic semantic framework for provenance in RDF$^+$. However, the framework must also be able to reproduce the literal interpretations found in RDF. For the latter purpose, we first define a 'standard' model for a RDF$^+$dataset.

Definition 3.3.4 (Standard Interpretation and Model) *A standard interpretation* $I_s : \mathfrak{S} \to \{\top, \bot\}$ *for a dataset* $D^+ = (K, M)$ *assigns truth values to all statements*[4] *in* K*. A standard interpretation for* K *is a standard model for* K *if and only if it makes all statements in* K *become true. This is denoted by* $I_s \models_s (K, M)$*.*

For instance, any standard model I_s for (K, M) in Example 3.3.3 would include

(G1, JamesCameron, occupation, FilmProducer, θ_1*)*

in its set of literal statements evaluating to $\top$. In order to address the level of provenance we foresee an additional model layer that provides a different interpretation to each provenance dimension.

Definition 3.3.5 (Γ-Interpretation and Model) *A* Γ*-interpretation* $I_\gamma : \mathfrak{S} \rightharpoonup \Omega_\gamma$ *for a dimension* $\gamma \in \Gamma$ *is a partial function mapping statements into the allowed value range of* γ*.*

A Γ*-interpretation* I_γ *is a* Γ*-model for* (K, M) *if and only if for all provenance statements* $(\theta, \gamma, \omega) \in M$ *where* $f_1(\theta) = (g, s, p, o)$ *the value of the interpretation equals to* ω*, i. e.* $I_\gamma((g, s, p, o, \theta)) = \omega$*. This is denoted by* $I_\gamma \models_\gamma (K, M)$*.*

As an example, consider the literal statement *(G1, JamesCameron, occupation, FilmProducer,* θ_1*)* from Example 3.3.3, and the provenance statement *(*θ_1*, certainty, 0.9)* from Example 3.3.4. A Γ-interpretation $I_{certainty}$, that is a Γ-model, would map the literal statement onto *0.9*.

The 'standard' interpretation and Γ-interpretation may now be combined to define what an overall, unambiguous model is:

Definition 3.3.6 (Provenance Interpretation) *A provenance interpretation* $\mathfrak{I}$ *is a set including a standard interpretation* I_s *and the* Γ*-interpretations* I_γ *for all provenance dimensions* $\gamma \in \Gamma$*.*

Definition 3.3.7 (Provenance Model) *A provenance interpretation* $\mathfrak{I}$ *is a model for a RDF*$^+$ *dataset* $D^+ = (K, M)$ *if and only if all its interpretations* $I \in \mathfrak{I}$ *are a standard model or* Γ*-models for* (K, M)*. This is denoted by* $\mathfrak{I} \models (K, M)$*.*

3.3.4. Mapping between RDF and RDF+

The mapping between RDF and RDF$^+$needs to be defined in two directions. First, one must be able to map from RDF, as given in the examples from Section 3.2, to RDF$^+$. Second, one must be able to map from RDF$^+$to RDF. Because RDF$^+$is more fine-grained than RDF the first direction will be easy. For the second a compromise on the granularity of the representation has to be made.

[4]Note that because f_1 is fixed there are no two tuples $(g, s, p, o, \theta_1), (g, s, p, o, \theta_2)$, where $\theta_1 \neq \theta_2$. This implies that the standard interpretation is independent of the identifiers θ_1, θ_2.

From RDF to RDF$^+$

The examples of Section 3.2 reify groups of statements, i. e. the ones found in *G1* and *G2*, in order to associate provenance, such as given in *G3* and *G4*. In order to allow for an interpretation of the provenance as defined in the preceding section, we map RDF into RDF$^+$. For all RDF statements, including statements in graphs *G1* and *G2* of Example 3.3.1, the mapping performed is close to an identity mapping. One only needs to add statement identifiers. The result for *G1* in RDF$^+$is:

Example 3.3.5 *Representing graph* G1 *in RDF$^+$*

$K \supseteq \{$
(G1, JamesCameron, occupation, FilmProducer, θ_1*),*
(G1, JamesCameron, produces, Avatar, θ_2*)}*
with
$\theta_1 :=$ *f_1(G1, JamesCameron, occupation, FilmProducer)*
and
$\theta_2 :=$ *f_1(G1, JamesCameron, produces, Avatar)*

The same mapping – close to the identity mapping – is performed for provenance statements like statements of graph *G3*, resulting in their representation as literal statements:

Example 3.3.6 *Representing graph* G3 *in RDF$^+$*

$K \supseteq \{$
(G3, G1, prov:source, <dbpedia.org/data/James_Cameron.xml>, θ_3*),*
(G3, G1, prov:certainty, '0.9', θ_4*),*
... $\}$

Note that this step is necessary in order to achieve upward and limited downward compatibility between RDF$^+$ and RDF. The interpretation of statements, like the ones found in *G3*, also require an interpretation as provenance. This is achieved by mapping RDF statements with designated dimensions from Γ like *prov:source* and *prov:certainty* to the additional provenance layer:

Example 3.3.7 *Provenance representation of statements in* G3

$M \supseteq \{$
(θ_1*, prov:certainty, '0.9'),*
(θ_1*, prov:source, <dbpedia.org/data/James_Cameron.xml>),*
... $\}$

The mapping of predicates of these provenance statements from RDF to RDF$^+$is obvious, they are mapped to itself. Objects are mapped to the corresponding elements of the value ranges Ω_γ. For the subjects, however, there arise modeling choices. For instance, if *prov:certainty* were interpreted using probability theory, one might assign a distributive or a collective reading. In the distributive reading, each fact in *G1* receives the probability value of 0.9 and, eventually, the distributive reading will assign a joint probability of close to 0 for a large number of n stochastically independent facts, i. e. the joint probability 0.9^n. In the

collective reading, the collection of facts in *G1* as a whole will receive the probability value 0.9. Therefore, the collective reading will assign an individual certainty close to 1 for each individual fact, when the number of facts is high and each fact is independent from the others, i. e. the individual probability would be $\sqrt[n]{0.9}$. *A priori*, none of the two (and more) modeling choices is better than the other, but they constitute different modeling targets.

The mapping from RDF to RDF$^+$ for the *distributive reading* of a provenance γ is easy to achieve.

Definition 3.3.8 (Distributive Embedding) *Given an RDF statement "$G\ \{s\ p\ o\}$" and an RDF provenance statement "$H\ \{G\ \gamma\ \omega\}$", a distributive embedding of RDF$^+$ in RDF adds the provenance statement*

$$\{(\theta, \gamma, \omega) \mid \theta = f_1(G, s, p, o) \wedge f_5(\theta) \in K\}$$

to M.

This means that such a provenance statement is applied individually to all statements in the graph to which it refers in RDF, as indicated in the example above. For certain γ there might be several RDF provenance statements $H\ \{G\ \gamma\ \omega_i\}$ which attach different values ω_i to a graph G via a single provenance dimension γ. In that case set-valued ranges have to be elements of Ω_γ in order to be consistent with Definition 3.3.5.

From RDF$^+$ to RDF

The serialization of RDF$^+$ data in an RDF knowledge base is straightforward. Each quintuple (g, s, p, o, θ) is realized as a corresponding triple in a named graph and the tuple identifier θ is discarded.

Example 3.3.8 *Serialization of some RDF$^+$ statemtents in* G5 *in RDF*

(G5, JamesCameron, occupation, FilmProducer, θ)

-is mapped to-

G5 {JamesCameron occupation FilmProducer}

For provenance statements the situation is more challenging, because literal statements with different statement identifiers may belong to only one named graph. Their corresponding provenance statements may differ, but the realization of the provenance statements in RDF does not allow for retaining these fine-grained distinctions – unless one chooses to change the modeling approach drastically, e.g.. by assigning each literal statement to a named graph of its own, which is undesirable (cf. discussion in Section 3.3.1).

We have preferred to pursue a more conventional modeling strategy for RDF with named graphs. Therefore, we weaken the association between provenance statements and their corresponding literal statements when mapping to RDF, that is, we group sets of provenance dimension values into one complex value.

Definition 3.3.9 (Grouped Provenance) *Given an RDF$^+$ dataset (K, M), RDF provenance is generated by grouping RDF$^+$ provenance statements as follows:*

Add the triple $(g\ \gamma\ \omega')$ *to the RDF graph* $g' := hashGraph(g)$ *for each*

$$\omega' := \omega_1 \vee_\gamma \ldots \vee_\gamma \omega_n\,,$$

where $(\theta, \gamma, \omega_i) \in M \wedge (g, S, P, O, \theta) \in K$*. Further,* hashGraph *is a function mapping existing graph names onto graph names suitable for associating provenance and* $\vee_\gamma$ *is an operation defined on* Ω_γ*.*

If ω' is set-valued then a set of triples is added to g' in order to represent ω'. The suitability of *hashGraph* may be application specific. A general strategy may map graph names g to graph names prefixed by `http://provenance.semanticweb.org` in a deterministic manner. Operations on provenance dimensions are discussed in Section 3.4.2. In the following example the grouping of provenance values is illustrated.

Example 3.3.9 *Group of provenance values*

K:= {
(G5, JamesCameron, occupation, FilmProducer, θ_1*).*
(G5, JamesCameron, produces, Avatar, θ_2*)}*
M:= {
(θ_1*, prov:source, <http://www.imdb.com/name/nm0000116.xml>).*
(θ_2*, prov:source, <dbpedia.org/data/James_Cameron.xml>)}*

-is mapped to-

G5 { JamesCameron occupation FilmProducer,
JamesCameron produces Avatar}
G6 { G5 prov:source <http://www.imdb.com/name/nm0000116.xml>,
<dbpedia.org/data/James_Cameron.xml>}

In Example 3.3.9, the resulting grouped value is the set consisting of the two documents

`http://www.imdb.com/name/nm0000116.xml` and `dbpedia.org/data/James_Cameron.xml`

which is represented by two triples. For specific provenance dimensions, an additional function may be necessary to provide a mechanism for representing grouped values in an appropriate RDF data structure.

3.4. Syntax and Semantics for SPARQL for Querying RDF Datasets with Provenance

For querying RDF repositories with provenance awareness, we exploit the capabilities of the SPARQL query language. In this section we first introduce a small extension to standard SPARQL syntax [Prud'hommeaux and Seaborne, 2008] [5] and then define how SPARQL can be applied to an RDF$^+$knowledge base. The objective of our considerations is the derivation of provenance about query results.

[5] In this thesis, we work with the version of SPARQL query language defined in [Prud'hommeaux and Seaborne, 2008]. The SPARQL Working Group has, however, produced a W3C Recommendation in 2013 for a new version of SPARQL which adds features to this 2008 version.

3.4.1. SPARQL Syntax Revisited

In our scenario, we assume that the user aims to explore the knowledge and provenance using the RDF query language SPARQL. The following SPARQL query shown in Example 3.4.1 enables the user to find out about movies and movie producers on the RDF dataset presented in Example 3.3.1.

Example 3.4.1 *SPARQL query to be evaluated on the RDF knowledge base in Example 3.3.1*

```
CONSTRUCT {?x occupation ?z}
FROM NAMED G1
FROM NAMED G2
WHERE { GRAPH ?g {?x produces ?z .
                  ?x occupation 'FilmProducer'}}
```

When using SPARQL to query RDF$^+$ we introduce one additional expression "WITH META *MetaList*". This expression includes the named graphs specified in *MetaList* for treatment as provenance. This statement is optional. When it is present the SPARQL processor may digest the RDF$^+$ provenance statements derivable from the RDF named graphs appearing in the *MetaList*. The SPARQL processor will then use this meta knowledge to compute and output all the provenance statements derivable by successful matches of RDF$^+$ literal statements with the WHERE pattern.

According to SPARQL semantics, FROM g expressions replicate all RDF triples of g into the default triple space of the query. When the same triple appears in multiple source graphs (say (s, p, o) appears in both $G1$ and $G2$, and the query contains a clause FROM NAMED G1 FROM NAMED G2), its provenance may become ambiguous. As an intermediate step of our query processing, this provenance is aggregated using RDF$^+$interpretations as introduced in Section 3.3.3. We note that this intermediate step is not necessary for queries with WITH NAMED g clauses, which are also fully compatible with our framework.

Thus, SPARQL queries on RDF$^+$ have one of the two following overall forms:

Definition 3.4.1 (SPARQL SELECT Query) *The structure of a SPARQL SELECT query has the following form:*

SELECT *SelectExpression*
(WITH META *MetaList*)?
(FROM *GraphName*)+
WHERE *P*

Definition 3.4.2 (SPARQL CONSTRUCT Query) *The structure of a SPARQL CONSTRUCT query has the following form:*

CONSTRUCT *ConstructExpression*
(WITH META *MetaList*)?
(FROM *GraphName*)+
WHERE *P*

In these definitions, P refers to a graph pattern that explains how RDF$^+$ literal statements from named graphs specified using FROM statements are matched. Matches bind variables that are used for providing results according to the *SelectExpression* or the *ConstructExpression*.

3.4.2. SPARQL Semantics Revisited

In this subsection we define the semantics of SPARQL queries evaluated on an RDF$^+$ theory. For our definitions we use two building blocks: algebraic semantics of SPARQL [Pérez et al., 2009, Arenas et al., 2009] and the *how-provenance* calculated via annotated relations (cf. [Green et al., 2007]).

The algebraic semantics of SPARQL queries are given based on set-theoretic operations for sets of variable assignments. Such a set of assignments may be assigned information about the so-called *how-provenance* [Green et al., 2007], i.e., the assignments may be annotated with formulae describing the individual derivation tree used to assign the variables. The how-provenance annotation may be represented by a function $\Phi : (U \cup L)^{|V|} \to \mathfrak{F}$, where $(U \cup L)^{|V|}$ is the set of all tuples of the length $|V|$ over the domain $U \cup L$ and $\mathfrak{F}$ is the set of formulae annotating variable assignments. The set of formulae $\mathfrak{F}$ is given by all Boolean formulas constructed over the set of literal statements $\mathfrak{S}$ and including a bottom element $\bot$ and a top element $\top$. The formulae constitute an algebra $(\mathfrak{F}, \wedge, \vee, \bot, \top)$. The special element $\bot$ is used as an annotation of variable assignments which are not in the result set. The special element $\top$ may be omitted, but it allows for the simplification of complex formulas.

The following definition shows how a set of variable bindings is generalized to SPARQL queries of arbitrary complexity by a recursive definition of simultaneous query evaluation and computation of the annotations. The first step in evaluating a graph pattern is to find matches for the triple pattern contained in the query. Because the RDF$^+$ dataset D^+ consists of quintuples, we need to adapt the SPARQL evaluation procedures. The statement identifiers do not need to be matched, as they depend functionally on graph name, subject, predicate and object. Therefore, we consider that the matching of triple patterns $P = (\alpha, \beta, \delta)$, given a graph name $\lambda \in U$, is always defined by the following SPARQL grammar (GRAPH λ P). As a simplification of our formalization we assume that the keyword GRAPH together with a URI or a graph variable is used in any given SPARQL query. If it is not used, we may expand a given SPARQL query to include it.

Definition 3.4.3 (Basic Graph Pattern Matching) *Given a basic graph pattern $P = (\alpha\, \beta\, \delta)$, a graph name $\lambda \in U$ and a mapping μ such that $var(P) \subseteq dom(\mu)$. Let D^+ be an RDF$^+$ dataset, $G = gr(\lambda)$ a RDF graph in D^+, and $\mu(P)$ is the set of triples obtained by replacing the variables in the triples of P according to μ. The (meta) evaluation of P over G, denoted by $[\![[P]\!]]_G^{D^+}$ is defined as the set of annotated mappings Φ, $\mu \subseteq \mathrm{dom}(\Phi)$: $[\![[P]\!]]_G^{D^+} = \{\mu \circ (\Phi(\mu)) \mid \mu : V \to U \cup L \ \wedge \Phi : (U \cup L)^{|V|} \to \mathfrak{F} \ \wedge dom(\mu) = var(P) \wedge \exists\ \theta\ (name(G) \circ \mu(P) \circ (\theta) \in K_G)\ \}$*

where $K_G := \sigma_{\#1=name(G)}(K)$, i. e. the selection of statements in K belonging to graph G and $\circ$ denotes vector concatenation. If $\mu \circ (\Phi(\mu)) \in [\![[P]\!]]_G^{D^+}$, we say that $\mu \circ (\Phi(\mu))$ is a solution for P in G.

$$\Phi(\mu) = \begin{cases} \theta & \text{if } \mu(P) = (g,s,p,o) \ \wedge \\ & \quad (g,s,p,o,\theta) \in K_G \ \wedge \\ & \quad f_1(g,s,p,o) = \theta, \\ \bot & \text{else} \end{cases}$$

Assume, for the next SPARQL queries example, the following RDF$^+$knowledge base D^+:

Example 3.4.2 *RDF$^+$ dataset $D^+ = (K, M)$*

$K = \{$
(G1, JamesCameron, occupation, FilmProducer, θ_1),
(G1, JamesCameron, produces, Avatar, θ_2),
(G2, JamesCameron, occupation, TruckDriver, θ_3),
(G2, JamesCameron, produces, BattleBeyondtheStars, θ_4),
(G2, MartinScorsese, occupation, Film Producer, θ_5),
(G2, MartinScorsese, produces, TaxiDriver , θ_6)}

For corresponding M, see Example 3.4.14

Let the SPARQL query in Example 3.4.3 be evaluated on the dataset D^+:

Example 3.4.3 *SPARQL query to be evaluated on the knowledge base D^+*

```
SELECT ?g ?x ?y
WITH META G3, G4
FROM NAMED G1
FROM NAMED G2
WHERE {
        GRAPH ?g {?x occupation ?y} }
```

For the query of Example 3.4.3, we may find the following variable assignments in Example 3.4.4 using standard SPARQL processing and we may indicate, which atomic formulae, i. e. RDF$^+$ quintuples in this simple example, led to these variable assignments. This indication is given by the statement identifiers representing their statements.

Example 3.4.4 *Variable assignments for query of Example 3.4.3*

$\Phi =$

?g	*?x*	*?y*	$\mathfrak{F}$
G1	*JamesCameron*	*FilmProducer*	θ_1
G2	*JamesCameron*	*TruckDriver*	θ_3
G2	*MartinScorsese*	*FilmProducer*	θ_5

Basic pattern matching is not directly applicable, if an expression 'GRAPH λ' appears outside a complex triple pattern. In such a case, we first need to distribute the expression 'GRAPH λ' appropriately to atomic triple patterns in order to prescribe atomic SPARQL expressions accessible by basic pattern matching. Because named graphs cannot be nested, this distribution is always possible and unambiguous.

In the following we use the function *quads*(P) to denote the query resulting from this transformation. In example 3.4.5 this transformation is demonstrated on a conjunction of two triple patterns.

Example 3.4.5 *Distributing the expression 'GRAPH λ' in a complex triple pattern with the function quads(P)*

P_1 =
```
GRAPH ?src {
       { ?x occupation ?y .}
       { ?x produces ?z .}}
```

$quads(P_1)$ =
```
GRAPH ?src { ?x occupation ?y .}
GRAPH ?src { ?x produces ?z .}
```

Now we define the evaluation of complex graph patterns by operations on sets of variable assignments similar to [Pérez et al., 2009, Arenas et al., 2009]. Note that we restrict to SPARQL fragments for which provenance models will be used for positive relational queries. For more details about the limitations of relational provenance models for capturing the semantics of the SPARQL OPTIONAL operator, see [Geerts et al., 2013].

Definition 3.4.4 (Complex Graph Pattern Matching) *Given the basic triple graph patterns P, P_1, P_2 and a graph name $\lambda \in U$. Let D^+ be an RDF$^+$ dataset, $G = gr(\lambda)$ a RDF graph in D^+. The meta evaluation of P, P_1, and P_2 over G, denoted by $[\![[.]\!]_G^{D^+}$, is defined recursively as follow:*

- $[\![[P]\!]_G^{D^+}$ *is given by definition 3.4.3,*
- $[\![[P_1 \text{ AND } P_2]\!]_G^{D^+} = [\![[P_1]\!]_G^{D^+} \bowtie [\![[P_2]\!]_G^{D^+}$,
- $[\![[P_1 \text{ UNION } P_2]\!]_G^{D^+} = [\![[P_1]\!]_G^{D^+} \cup [\![[P_2]\!]_G^{D^+}$,
- $[\![[P_1 \text{ FILTER } C]\!]_G^{D^+} = \sigma_c([\![[P_1]\!]_G^{D^+})$.

The definition uses the operation AND. In standard SPARQL the operation AND is denoted by the absence of an operator. Like [Pérez et al., 2009, Arenas et al., 2009] we still use the explicit term AND in order to facilitate referencing to this operator.

The recursion in the SPARQL query evaluation defined in Definition 3.4.3 is indeed identical to [Pérez et al., 2009, Arenas et al., 2009] (the work of [Pérez et al., 2009, Arenas et al., 2009] on the SPARQL semantics is briefly summarized in Chapter 3.4). Only the basic pattern matching has been changed slightly. Basic pattern matching now considers the basic pattern triple with named graph matching, and it annotates variable assignments $\Phi(\mu)$ from basic matches with atomic statements from $\mathfrak{S}$ and variable assignments from complex matches with Boolean formulae $F \in \mathfrak{F}$ over these statements.

The following definition specifies how complex formulae from $\mathfrak{F}$, which are used to construct formulas representing the how-provenance and serve as annotations for results of matching complex graph pattern, are derived using algebraic operations on sets of annotated variables bindings.

Definition 3.4.5 (Algebra of Annotated Relations) *Let Φ, Φ_1 and Φ_2 be sets of annotated variable assignments. We define $\bowtie$, $\cup$, σ, and ψ via operations on the annotations of the assignments (the two binary operations $\wedge$ and $\vee$) as following:*

- $(\Phi_1 \bowtie \Phi_2)(\mu) = \Phi_1(\mu_1) \wedge \Phi_2(\mu_2)$, *where* $\forall x \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2) : \mu_1(x) = \mu_2(x)$ *and where μ_1 and μ_2 are compatible and $\mu = \mu_1 \cup \mu_2$*[6],

[6] Two mappings μ_1 and μ_2 are *compatible* when for all $x \in dom(\mu_1) \cap dom(\mu_2)$, it is the case that $\mu_1(x) = \mu_2(x)$, then $\mu_1 \cup \mu_2$ is also a mapping.

- $(\Phi_1 \cup \Phi_2)(\mu) = \Phi_1(\mu) \vee \Phi_2(\mu)$, $\mu \in \Phi_1$ *or* $\mu \in \Phi_2$
- $(\sigma_c(\Phi(\mu))) = \Phi(\mu) \wedge f_c(\mu)$, *where* $f_c(\mu)$ *denotes a function mapping* μ *to either* $\top$ *or* $\bot$ *according to the condition* c.
- $(\psi_X(\Phi(\mu))) = \bigvee_{\mu \subseteq \nu, \mathrm{dom}(\mu)=X, \Phi(\nu)\neq\bot} \Phi(\nu)$.

Furthermore, following the algebraic definition of how-provenance formula, we present the set of laws for annotated relations.

Definition 3.4.6 (Laws of Annotated Relations) *Let* Φ_1, Φ_2 *and* Φ_3 *be sets of annotated variable assignments. We define the laws for annotation relations via operations on the annotations of the assignments as follows:*

- *Idempotency:*
  $\Phi_1(\mu_1) \wedge \Phi_1(\mu_1) = \Phi_1(\mu_1)$
  $\Phi_1(\mu_1) \vee \Phi_1(\mu_1) = \Phi_1(\mu_1)$
- *Commutativity:*
  $\Phi_1(\mu_1) \wedge \Phi_2(\mu_2) = \Phi_2(\mu_2) \wedge \Phi_1(\mu_1)$
  $\Phi_1(\mu_1) \vee \Phi_2(\mu_2) = \Phi_2(\mu_2) \vee \Phi_1(\mu_1)$
- *Associativity:*
  $\Phi_1(\mu_1) \wedge (\Phi_2(\mu_2) \wedge \Phi_3(\mu_3)) = (\Phi_1(\mu_1) \wedge \Phi_2(\mu_2)) \wedge \Phi_3(\mu_3)$
  $\Phi_1(\mu_1) \vee (\Phi_2(\mu_2) \vee \Phi_3(\mu_3)) = (\Phi_1(\mu_1) \vee \Phi_2(\mu_2)) \vee \Phi_3(\mu_3)$
- *Distributivity:*
  $\Phi_1(\mu_1) \wedge (\Phi_2(\mu_2) \vee \Phi_3(\mu_3)) = (\Phi_1(\mu_1) \vee \Phi_2(\mu_2)) \wedge (\Phi_1(\mu_1) \vee \Phi_3(\mu_3))$
  $\Phi_1(\mu_1) \vee (\Phi_2(\mu_2) \wedge \Phi_3(\mu_3)) = (\Phi_1(\mu_1) \wedge \Phi_2(\mu_2)) \vee (\Phi_1(\mu_1) \wedge \Phi_3(\mu_3))$
- *Absorption:*
  $\Phi_1(\mu_1) \wedge (\Phi_1(\mu_1) \vee \Phi_2(\mu_2)) = \Phi_1(\mu_1)$
  $\Phi_1(\mu_1) \vee (\Phi_1(\mu_1) \wedge \Phi_2(\mu_2)) = \Phi_1(\mu_1)$

In order to show the evaluation of a SPARQL query, we consider the query from Example 3.4.6 to be evaluated on the knowledge base D^+.

Example 3.4.6 *SPARQL query to be evaluated on the knowledge base* D^+

```
SELECT ?h1 ?h2 ?x ?y
WITH META G3, G4
FROM NAMED G1
FROM NAMED G2
WHERE {
          {GRAPH ?h1 {?x produces ?y}} AND
          {GRAPH ?h2 {?x occupation 'FilmProducer'}}
            FILTER {?x='JamesCameron'}}
```

Let P be the graph pattern contained in the WHERE clause of the query. Then the evaluation of P is defined by an algebraic expression:

$$\begin{aligned}
[\![[\![P]\!]]\!]_G^{D^+} &= [\![[\![\{P_1 \text{ AND } P_2\} \\
&\quad \text{FILTER } \{?\text{x} = \text{JamesCameron}\}]\!]]\!]_G^{D^+} \\
&= \sigma_{?\text{x=JamesCameron}}([\![[\![P_1 \text{ AND } P_2]\!]]\!]_G^{D^+}) \\
&= \sigma_{?\text{x=JamesCameron}}([\![[\![P_1]\!]]\!]_G^{D^+} \bowtie [\![[\![P_2]\!]]\!]_G^{D^+}) \\
&= \sigma_{?\text{x=JamesCameron}}(\Phi_1 \bowtie \Phi_2)
\end{aligned}$$

where Φ_1 and Φ_2 are relations representing variable assignments and their annotations. In this example and in the preceding definition we have used algebraic operations on sets of annotated bindings. In order to evaluate the expression $\sigma_{?\text{x=JamesCameron}}(\Phi_1 \bowtie \Phi_2)$ we need to determine Φ_1 and Φ_2 using Definition 3.4.3. The intermediate result is shown in Example 3.4.7.

Example 3.4.7 *Determining Φ_1 and Φ_2 - intermediate result of the expression $\sigma_{?\text{x=JamesCameron}}(\Phi_1 \bowtie \Phi_2)$*

$\Phi_1 =$

?h1	?x	?y	$\mathfrak{S}_1$
G1	JamesCameron	Avatar	θ_2
G2	JamesCameron	BattleBeyondtheStars	θ_4
G2	MartinScorsese	TaxiDriver	θ_6

$\Phi_2 =$

?h2	?x	$\mathfrak{S}_2$
G1	JamesCameron	θ_1
G2	MartinScorsese	θ_5

To evaluate the conjunction of two patterns, the operation $\bowtie$ is applied, and the result is shown in Example 3.4.8. The annotation $\theta_1 \wedge \theta_2$ of the first row represents that this assignment has been derived from the conjunction of the two literal statements θ_1 and θ_2 (see Example 3.4.2).

Example 3.4.8 *Determining $\Phi_1 \bowtie \Phi_2$ - intermediate result of the expression $\sigma_{?\text{x=JamesCameron}}(\Phi_1 \bowtie \Phi_2)$*

$\Phi_1 \bowtie \Phi_2 =$

?h1	?h2	?x	?y	$\mathfrak{S}_3$
G1	G1	JamesCameron	Avatar	$\theta_1 \wedge \theta_2$
G1	G2	JamesCameron	BattleBeyondtheStars	$\theta_1 \wedge \theta_4$
G2	G2	MartinScorsese	TaxiDriver	$\theta_5 \wedge \theta_6$

Application of the σ-operation to the intermediate results gives the annotated relation shown in Example 3.4.9.

Example 3.4.9 *Result of $\sigma_{?\text{x=JamesCameron}}(\Phi_1 \bowtie \Phi_2)$*

$\sigma_{?\text{x=JamesCameron}}(\Phi_1 \bowtie \Phi_2) =$

?h1	?h2	?x	?y	$\mathfrak{S}_4$
G1	G1	JamesCameron	Avatar	$(\theta_1 \wedge \theta_2) \wedge \top$
G1	G2	JamesCameron	BattleBeyondtheStars	$(\theta_1 \wedge \theta_4) \wedge \top$
G2	G2	MartinScorsese	TaxiDriver	$\bot$

The annotations $\Phi(\mu)$ can now be used to assign truth values for each tuple binding μ. I_s (cf. Definition 3.3.4) assigns truth values to all atomic statements $s_i \in K \subseteq \mathfrak{S}$. We extend the interpretation I_s to capture all the Boolean formulae over statements $\mathfrak{S}$.

Definition 3.4.7 (Standard Interpretation of Formulae) *Let $F, F_1, F_2 \in \mathfrak{F}$ be Boolean formulae over $\mathfrak{S}$, let $F_a \in \mathfrak{S}$ be an atomic formula. We define the standard interpretation of formulae I_s^f as follows:*

- $I_s^f(F_a) := I_s(F_a)$;
- $I_s^f(F_1 \wedge F_2)$ *is* $\top$ *if* $I_s^f(F_1) = I_s^f(F_2) = \top$, *otherwise* $\bot$
- $I_s^f(F_1 \vee F_2)$ *is* $\top$ *if* $I_s^f(F_1) = \top$ *or* $I_s^f(F_2) = \top$, *otherwise* $\bot$.

For instance, I_s^f returns $\top$ for the assignment shown in the first row of $\Phi_1 \bowtie \Phi_2$ from Example 3.4.8, because the statements θ_1 and θ_2 are in the knowledge base.

Basically, the standard interpretation of formulae I_s^f assigns truth values for all variable bindings that are in the knowledge base, as well as captures all the Boolean formulae over these statements bindings. For producing the result set, we dismiss all variable bindings with $\bot$ value assigned.

Analogously to I_s^f, we can extend a Γ-interpretation I_γ over RDF$^+$ statements to a Γ-interpretation I_γ^f over formulae. Similar to the definition of the standard interpretation of formulae, the Γ-interpretation assigns values which represent provenance dimensions for all variable bindings which are not in a set of bindings annotated with $\bot$. Thus we do not consider provenance interpretations of such bindings over formulae. Remember that provenance interpretations allow for only one ω per $\theta \in \Theta$ and $\gamma \in \Gamma$ (cf. Definition 3.3.5). In order to make use of the how-provenance represented by the annotations we require that for each provenance dimension γ an algebra $(\Omega_\gamma, \wedge_\gamma, \vee_\gamma, \top_\gamma, \bot_\gamma)$ with two operations $\wedge_\gamma$ and $\vee_\gamma$, and two special elements $\top_\gamma, \bot_\gamma \in \Omega_\gamma$ is defined. The definition of the algebras can be supplied by a modeler according to the intended semantics of the different provenance dimensions.

Definition 3.4.8 (Γ-Interpretation of Formulae)
Let $F, F_1, F_2 \in \mathfrak{F}$ be Boolean formulae over $\mathfrak{S}$, let $F_a \in \mathfrak{S}$ be an atomic formula. We define the interpretation I_γ^f as follows:

- $I_\gamma^f(F_a) := I_\gamma(F_a)$;
- $I_\gamma^f(F_1 \wedge F_2)$ *is* $I_\gamma^f(F_1) \wedge_\gamma I_\gamma^f(F_2)$;
- $I_\gamma^f(F_1 \vee F_2)$ *is* $I_\gamma^f(F_1) \vee_\gamma I_\gamma^f(F_2)$;

The definition of knowledge dimensions can be supplied by the administrator of the knowledge base, according to the intended semantics of particular provenance dimensions. For illustration we consider in Example 3.4.10 the definition of fuzzy logic operations to calculate

a possibility measure on variable assignments, operations defined on timestamps which calculate the time of the last modification, and set operations defined for source documents that construct the combined 'how-provenance'.

Example 3.4.10 *Fuzzy logic operations for provenance dimensions*

$\Omega_{certainty} = [0,1]$
$I^f_{certainty}(x_1 \wedge x_2) = \min(I^f_{certainty}(x_1), I^f_{certainty}(x_2))$
$I^f_{certainty}(x_1 \vee x_2) = \max(I^f_{certainty}(x_1), I^f_{certainty}(x_2))$
$\Omega_{time} = [0,\infty)$
$I^f_{time}(x_1 \wedge x_2) = \max(I^f_{time}(x_1), I^f_{time}(x_2))$
$I^f_{time}(x_1 \vee x_2) = \min(I^f_{time}(x_1), I^f_{time}(x_2))$

$\Omega_{source} = 2^D$, D the set of document URIs
$I^f_{source}(x_1 \wedge x_2) = I^f_{source}(x_1) \cup I^f_{source}(x_2)$
$I^f_{source}(x_1 \vee x_2) = I^f_{source}(x_1) \cup I^f_{source}(x_2)$

As discussed earlier, the graph pattern queries evaluation with provenance algorithm proposed here follows a compositional semantics, i. e. for each P' sub-pattern of P the result of evaluating P' can be used to evaluate P. In order to solve this problem algorithmically, we have implemented, analogously to [Pérez et al., 2009, Arenas et al., 2009], a depth-first strategy algorithm for evaluating SPARQL query patterns with provenance. This algorithm, denoted by $Eval^+$, has been implemented so that it emulates exactly the recursive definition of $[\![[.]]\!]$ for well-designed pattern as defined in [Polleres, 2007]. Formally, given an RDF$^+$dataset D^+, we define the evaluation of pattern P with the set of mappings Ω, denoted by $Eval^+_{D^+}(P,\Omega)$ as follows:

Algorithm 1 Evaluation Algorithm: $Eval^+_{D^+}(P, \Omega)$

1: **if** $\Omega = \varnothing$ **then**
2: **return** $(\varnothing)$
3: **end if**
4: **if** P is a triple pattern t **then**
5: **return** $(\Omega \bowtie [\![[t]]\!]_{D^+})$
6: **end if**
7: **if** $P = (P1$ AND $P2)$ **then**
8: **return** $Eval^+_{D^+}(P2, Eval^+_{D^+}(P1,\ \Omega))$
9: **end if**
10: **if** $P = (P1$ UNION $P2)$ **then**
11: **return** $Eval^+_{D^+}(P1,\ \Omega)\ \cup Eval^+_{D^+}(P2,\ \Omega))$
12: **end if**
13: **if** $P = (P1$ FILTER $R)$ **then**
14: **return** $\{\ \Phi(\mu) \in Eval^+_{D^+}(P1,\ \Omega)\ \wedge\ f_R(\mu)\ \}$ where $f_R(\mu)$ denotes a function mapping μ to either $\top$ or $\bot$ according to the condition R
15: **end if**

Then, the evaluation of P against a dataset D^+, which we denote simply by $Eval^+_{D^+}(P)$, is $Eval^+_{D^+}(P,\ \Phi(\mu)_\varnothing)$, where $\Phi(\mu)_\varnothing$ is the mapping with empty domain.

3.4.3. SPARQL Query forms

As mentioned before, in this work we are only interested in standard SPARQL SELECT and CONSTRUCT query forms. The evaluation of SPARQL queries on RDF$^+$data differs from the standard evaluation in that provenance is attached to the results.

The evaluation of SELECT queries on an RDF$^+$dataset is based on $\psi_X([\![[P]\!]]_G^{D^+})$ (see Definition 3.4.5), where X denotes the set of variables specified in the *SelectExpression* (see Definition 3.4.1). If X forms a proper subset of the variables used in the graph pattern then the annotations of all bindings ν are grouped. This grouping is analogous to the generation of grouped provenance described in Definition 3.3.9. As an example consider the query shown in Example 3.4.11, which is a slight modification of the query from Example 3.4.6, applied to the data shown in Example 3.4.2. For the result see Example 3.4.12. In contrast to Example 3.4.8 there is only one row for *JamesCameron*.

Example 3.4.11 *SPARQL query to be evaluated on knowledge base D^+*

```
SELECT ?x
WITH META G3, G4
FROM NAMED G1
FROM NAMED G2
WHERE {
        {GRAPH ?h1 {?x produces ?y}} AND
        {GRAPH ?h2 {?x occupation 'FilmProducer'}}}
```

The result of a SELECT query is a set of extended bindings. Such an extended binding contains values for the specified variables and values for each provenance dimension $\gamma \in \Gamma$ which can be regarded as additional variables.

Example 3.4.12 *Determining $\psi_{\{?x\}}(\Phi_1 \bowtie \Phi_2)$ - intermediate result of the expression defined in Example 3.4.11*

$\psi_{\{?x\}}(\Phi_1 \bowtie \Phi_2) =$

?x	$\mathfrak{S}_5$
JamesCameron	$(\theta_1 \wedge \theta_2) \vee (\theta_1 \wedge \theta_4)$
MartinScorsese	$\theta_5 \wedge \theta_6$

For each binding μ of query from the Example 3.4.11, the variables γ are bound to $I_\gamma^f(\psi_X([\![[P]\!]]_G^{D^+})(\mu_i))$ (see Example 3.4.13). For this result the provenance from Example 3.4.14 has been used. For instance $I^f_{certainty}((\theta_1 \wedge \theta_2) \vee (\theta_1 \wedge \theta_4)) = 0.9$. If no provenance statement (θ, γ, ω) exists for a particular RDF$^+$ literal statement $f_5(\theta)$ and a particular provenance dimension γ then $\perp_\gamma$ serves as default value. For the result of a SELECT query all bindings from $\psi_X([\![[P]\!]]_G^{D^+})$ are extended in this way.

Example 3.4.13 *Variable bindings for $I_\gamma^f(\psi_X([\![[P]\!]]_G^{D^+})(\mu_i))$ - results of the expression defined in Example 3.4.11*

$I_\gamma^f(\psi_X([\![[P]\!]]_G^{D^+})(\mu_i)) =$

?x	*certainty*	*time*
JamesCameron	'0.9'	"5/5/20014"
MartinScorsese	'0.6'	"6/6/1980"

Example 3.4.14 *Provenance statements,* $D^+ = (K, M)$

M = {
(θ_1, prov:certainty, '0.9')
(θ_1, prov:time, "5/5/2014")
(θ_2, prov:certainty, '0.9')
(θ_2, prov:time, "5/5/2014")
(θ_3, prov:certainty, '0.6')
(θ_3, prov:time, "6/6/1980")
(θ_4, prov:certainty, '0.6')
(θ_4, prov:time, "6/6/1980")
(θ_5, prov:certainty, '0.6')
(θ_5, prov:time, "6/6/1980")
(θ_6, prov:certainty, '0.6')
(θ_6, prov:time, "6/6/1980")}

Analogously to standard evaluation, the evaluation of a CONSTRUCT query on an RDF$^+$dataset results in a single RDF$^+$ graph which is built using the graph template specified in the *ConstructExpression* (see Definition 3.4.2). This is in line with the fact that the graph template consists of a conjunction of triple patterns and a named graph thus quadruple patterns cannot be stated.[7]

Similar to the evaluation of SELECT queries, the evaluation of CONSTRUCT queries is based on $\psi_X([\![[\![P]\!]]\!]_G^{D^+})$, where X denotes the set of variables specified in the *ConstructExpression*. The RDF$^+$ graph is constructed as described in the following:

Let t_j be the triple pattern j specified in the *ConstructExpression*, P denote the graph pattern specified in the WHERE-clause, $(s_{i,j}, p_{i,j}, o_{i,j})$ denote the triple obtained by replacing the variables in t_j according to a mapping μ_i and $\hat{g}$ denote a new graph name. Then, for each binding $\mu_i \in \psi_X([\![[\![P]\!]]\!]_G^{D^+})$ and for each t_j the quintuple $(\hat{g}, s_{i,j}, p_{i,j}, o_{i,j}, \theta_{i,j})$ is added to $\mathfrak{S}$, where $\theta_{i,j}$ is the statement identifier $f_1(\hat{g}, s_{i,j}, p_{i,j}, o_{i,j})$. Further $(\theta_{i,j}, \gamma, \omega_{i,j})$ is added to M, where $\omega_{i,j} = I_\gamma^f(\psi_X([\![[\![P]\!]]\!]_G^{D^+})(\mu_i))$.

Each new quintuple inherits the provenance dimensions γ associated with the binding which has been used to create that quintuple. The value of $\omega_{i,j}$ is determined by applying I_γ^f to the formula which annotates the binding. Note that since $\psi_X([\![[\![P]\!]]\!]_G^{D^+})$ and the interpretations I_γ^f are functions and further the graph template in *ConstructExpression* is a set of triples the provenance dimensions $(\theta_{i,j}, \gamma, \omega_{i,j})$ are unique for a given $\theta_{i,j}$.

As an example for a CONSTRUCT statement consider Example 3.4.15. Provenance for some of the RDF$^+$ statements presented in Example 3.4.2 is specified in Example 3.4.14. For graph pattern P contained in this query the result of $\psi_X([\![[\![P]\!]]\!]_G^{D^+})$ is identical to the annotated relation shown in Example 3.4.8 except for the first two columns. Based on the single triple pattern (`?x occupation ?y`) contained in the graph template and the two bindings contained in $\psi_X([\![[\![P]\!]]\!]_G^{D^+})$ two quintuples are constructed and added to the RDF$^+$ literal statements K_{res} as shown in Example 3.4.16. M_{res} contains the corresponding provenance statements resulting from $I_\gamma^f(\psi_X([\![[\![P]\!]]\!]_G^{D^+})(\mu_i))$.

Example 3.4.15 *SPARQL query CONSTRUCT*

[7] Standard SPARQL does not allow for giving this graph a name. In order to associate provenance, multiple named graphs as outputs are convenient. In order to remain standard compliant, the SPARQL engine may however also return data and provenance in two different batches distinguished by some implementation-specific mechanism.

```
CONSTRUCT {?x occupation ?y}
WITH META G3, G4
FROM NAMED G1
FROM NAMED G2
WHERE {
          {GRAPH ?h1 {?x produces ?y}} AND
          {GRAPH ?h2 {?x occupation 'FilmProducer'}}}
```

Example 3.4.16 *Variable binding for query of the Example 3.4.15*

K_{res} = {
(G_{new}, JamesCameron, occupation, FilmProducer, θ_{new1})
(G_{new}, JamesCameron, occupation, TruckDriver, θ_{new2})
(G_{new}, MartinScorsese, occupation, FilmProducer, θ_{new3})}
M_{res} = {
(θ_{new1}, prov:certainty, '0.9')
(θ_{new1}, prov:time, "5/5/2014")
(θ_{new2}, prov:certainty, '0.6')
(θ_{new2}, prov:time, "6/6/1980")
(θ_{new3}, prov:certainty, '0.6')
(θ_{new3}, prov:time, "6/6/1980")}

3.4.4. Advanced Query Forms

The SPARQL query forms presented so far can be seen as a straightforward adaptation of querying capabilities to a knowledge base with associated provenance. Beyond this, the presented framework can potentially be extended for more complex query forms with provenance awareness.

Queries with conditions on provenance

A possible interesting extension of our framework is the support for additional selection and/or ranking conditions on associated provenance attributes. From the conceptual point of view, this functionality can be seen as an extension of the WHERE clause. In our example introduced in Section 3.2, the query about affiliations of researchers (Example 3.4.1) may also require that only certain facts (say with associated certainty values greater 0.8) shall be used. This restriction would result in exclusion of triples from graph $G2$ (with insufficient certainty of 0.6) from evaluation. Consequently, the result set would only contain the statement <JamesCameron occupation FilmProducer> which is completely constructed using triples from remaining graph $G1$, with its high certainty of 0.9. This idea is reflected in [Carroll et al., 2005], which gives examples of querying RDF through SPARQL queries with provenance constraints, using the notion of Named Graphs. Such functionality can be realized in our framework in a straightforward manner, using *nested* queries (or querying with provenance from *views*).

We note that integration of nested queries with our approach would result in a substantially more flexible and powerful solution than the approach from [Carroll et al., 2005]. In particular, our framework potentially allows to express filtering and/or ranking conditions on provenance of *query results*. Conceptually, this functionality can be seen as an extension of the HAVING

clause (in the common SQL-like sense). For instance, for our sample query from Example 3.4.1 we may require the minimum overall certainty of each result to be greater than 0.8. The resulting filter condition is then applied to query results from Example 3.4.16, which contain complex interpretations of meta knowledge for possible variable bindings.

Beyond this, the exploitation of SPARQL on RDF$^+$would allow for specifying conditions on *aggregated* provenance of the query result in a natural way. Conceptually, this functionality can be established in a quite straightforward manner using nested queries (or querying with provenance from views). The necessary formalization for built-in functions and aggregates by means of external predicates can be found in [Polleres et al., 2007].

Nested provenance

The *nested* provenance (i.e. provenance about provenance) can potentially be expressed through RDF capabilities. Following our example from Section 3.2, in the nested setting we would be able to specify conditions regarding recentness or reliability of associated provenance. For instance, we may require that only query results with high certainty of their extraction sources shall be returned. These kinds of queries were not yet a concern in our framework presented so far. However, the required adaptation is not too difficult. As discussed in Section 3.3.1, there is no conceptual separation between knowledge and provenance in our repository. For this reason, complex queries may treat provenance as knowledge and obtain its meta annotation (i.e., nested provenance) in a quite straightforward manner. Likewise, advanced querying mechanisms with nested provenance support can easily be supported. As a consequence of the preceding discussion, we note that advanced conditions with nested provenance can also be specified both for repository contents (i.e., kind of advanced WHERE clause) and for results of complex queries (i.e., kind of advanced HAVING clause).

3.5. Equivalences for SPARQL expressions with Provenance

The main goal of this section is to show the equivalence holding between the standard evaluation of SPARQL queries, denoted by $[\![.]\!]$ (see Section 3.4), and the evaluation with provenance, presented in this work denoted by $[\![\![.]\!]\!]$. For this purpose, we first define the provenance projection operator Ψ as following:

Definition 3.5.1 *(Provenance Projection)*
Let Φ be an annotated relation representing the set of tuples of variable assignments and their annotations, then

$$\Psi(\Phi) = \Pi_{var(P)}(\sigma_{(I_s \to \top)}(\Phi))$$

where $var(P)$ is the set of variables occurring in P, and $I_s \to \top$ denotes the standard interpretation I_s of an annotation formula evaluated to $\top$.

Basically, the provenance projection operator selects all the statements corresponding to the variables bindings of the evaluation and ignores the provenance dimensions assigned to the statements in the knowledge base. As discussed earlier, the standard interpretation I_s of an annotation formula evaluates to $\top$ if and only if the statements in the knowledge base are

elements of the variable assignments obtained via evaluation, i. e. the corresponding variable assignment is in the set of variable assignments obtained via standard SPARQL evaluation extended with identifiers (see Definition 3.4.3). Based on this, the next proposition defines the equivalence holding between the evaluation with provenance and the standard SPARQL evaluation.

Proposition 3.5.1 *Given a graph pattern expression P, we say that the meta projection on the results of evaluation with provenance of the graph pattern expression P over a RDF$^+$ dataset D^+ is equivalent to the standard evaluation of P over the RDF dataset D, denoted by* $\Psi[\![[\![P]\!]]\!]^{D^+} = [\![P]\!]^D$.

Proof: (sketch) For sets of variable assignments obtained via basic pattern, this follows immediately from Definition 3.4.3 and the application of operator Ψ over Φ, so that for each substitution $\Phi(\mu)$ obtained by evaluation over a dataset D^+, $[\![[\![P]\!]]\!]^{D^+}$, can be reduced to a substitution μ obtained from the evaluation P over a RDF dataset D, $[\![P]\!]^D$, defined in [Pérez et al., 2009, Arenas et al., 2009] by selecting all μ where I_s of an annotation formula evaluates to $\top$ and dropping all remaining substitutions of θ. For complex graph patterns, the evaluation defined (see Definition 3.4.3) is indeed identical to [Pérez et al., 2009, Arenas et al., 2009].■

For relations annotated with Boolean formulae and standard relational algebra a proof for Proposition 3.5.1 can be found in [Fuhr and Rölleke, 1997].

Now, we can define the equivalence holding between equivalence graph pattern expressions with respect to the evaluation with provenance, i.e. show that the evaluation of equivalent queries are annotated with equivalent provenance dimensions. By equivalence we mean that the interpretation of formulae used to annotate bindings contained in the results of the equivalent queries are equivalent.

We use for the algebra of annotation formulae as syntactic objects $(\mathfrak{F}, \wedge, \vee, \bot, \top)$ (see Section 3.4). The elements of $\mathfrak{F}$ are Boolean formulae built over the set of all literal RDF$^+$statements $\mathfrak{S}$. Instead of the statements themselves we use their identifiers. In the tagged tuple model which we adopt here a relation is represented by a function mapping all tuples of a domain to an annotation. Since we use $\mathfrak{F}$ only as a placeholder we could just as well directly annotate variable bindings with elements of different provenance algebras. Therefore, first we need to define restrictions on the algebra for provenance interpretation for each provenance dimension γ, $(\Omega_\gamma, \wedge_\gamma, \vee_\gamma, \top_\gamma, \bot_\gamma)$ as defined in Section 3.4, so that the equivalence holds. As a motivation for the reader, before presenting the restrictions for the provenance algebra, in the next example we show one case where the equivalence holds and another where the equivalence does not hold when the provenance dimensions are defined using the fuzzy logic operations as presented in Example 3.4.10. Let P = (P_1 AND P_2) and P' = (P_2 AND P_1) be equivalent complex graph pattern expressions. The annotation results for P and P' consist of $(F_1 \wedge F_2)$, and $(F_2 \wedge F_1)$, respectively, where F_1 and F_2 denote annotation Boolean formulae. We are going to evaluate P and P' with the provenance dimension *time*. The Γ–interpretation for I_{time} for the $(F_1 \wedge F_2)$ formulae is $I^f_{time}(F_1 \wedge F_2) = max(I^f_{time}(F_1) \wedge_{time} I^f_{time}(F_2)) = max(I^f_{time}(F_1), I^f_{time}(F_2))$. Likewise, the Γ–interpretation for I_{time} for the $(F_2 \wedge F_1)$ formulae is $I^f_{time}(F_2 \wedge F_1) = max(I^f_{time}(F_2), I^f_{time}(F_1))$. As shown in this example the provenance evaluation for P and P' leads to the same result; thus it is equivalent.

In the following we summarize some of the restrictions on the provenance algebra. Let $P1$, $P2$ and $P3$ be graph pattern expressions and the built-in condition P:

1. $[\![[\![P1 \text{ AND } P2]\!]]\!]_G^{D^+} = [\![[\![P2 \text{ AND } P1]\!]]\!]_G^{D^+}$, and $[\![[\![P1 \text{ UNION } P2]\!]]\!]_G^{D^+} = [\![[\![P2 \text{ UNION } P1]\!]]\!]_G^{D^+}$

   Since annotations for results of graph pattern of the form P_1 AND P_2 are of the form $F_1 \wedge F_2$, where F_1 and F_2 denote annotation formulae, and annotations for results of graph pattern of the form P_1 UNION P_2 are of the form $F_1 \vee F_2$ we require that:

   $$\wedge \textit{ and } \vee \textit{ are associative and commutative.}$$

   for annotation formulae in order to guarantee that results of equivalent queries are associated with equivalent provenance.

2. $[\![[\![(P1 \text{ AND } (P2 \text{ UNION } P3))]\!]]\!]_G^{D^+} = [\![[\![((P1 \text{ AND } P2) \text{ UNION } (P1 \text{ AND } P3))]\!]]\!]_G^{D^+}$

   In order to satisfy this equivalence, the annotation formulae and algebras used to interpret these formulae need to satisfy the equivalence:

   $$I_\gamma^f(F_1 \wedge (F_2 \vee F_3)) \equiv I_\gamma^f((F_1 \wedge F_2) \vee (F_1 \wedge F_3))$$

3. $[\![[\![((P1 \text{ UNION } P2) \text{ FILTER } R)]\!]]\!]_G^{D^+} = [\![[\![((P1 \text{ FILTER } R) \text{ UNION } (P2 \text{ FILTER } R))]\!]]\!]_G^{D^+}$, and
   $[\![[\![((((P1 \text{ FILTER } R1) \text{ FILTER } R2)))]\!]]\!]_G^{D^+} = [\![[\![(((((P1 \text{ FILTER } (R1 \wedge R2))))]\!]]\!]_G^{D^+}$.

   Now we look at equivalences in the context of FILTER expressions. Equivalence requires:

   $$I_\gamma^f((F_1 \vee F_2) \wedge r) \equiv I_\gamma^f((F_1 \wedge r)) \vee I_\gamma^f((F_2 \wedge r))$$

   which follows from (i) and (ii).
   Analogous:
   $$I_\gamma^f((F_1 \wedge r_1) \wedge r_2) \equiv I_\gamma^f(F_1 \wedge r_3)$$

   where $r_3 = r_1 \wedge r_2$.

4. $[\![[\![(P1 \text{ AND } P1)]\!]]\!]_G^{D^+} = [\![[\![P1]\!]]\!]_G^{D^+}$
   Equivalence requires idempotency:

   $$I_\gamma^f(F \wedge F) \equiv I_\gamma^f(F)$$

Based on the restrictions presented above, we see evidence for requiring the provenance algebra to form such a commutative semiring [Green et al., 2007] since the laws of commutative semirings are forced by certain expected identities in the provenance algebra. For this purpose, we define in the next theorem that the provenance algebra has to take the form of a commutative ring algebraic structure in order to hold the equivalences presented above for provenance interpretations. With the commutative ring approach, the extendability of our approach increases due to the low restriction level on the provenance algebra required for the equivalences.

Theorem 3.5.1 *Let P1 and P2 be equivalent graph pattern expressions using AND, OR, and FILTER operator and D^+ a RDF$^+$ dataset. We have that $[\![[\![P1]\!]]\!]^{D^+} = [\![[\![P2]\!]]\!]^{D^+}$ if and only if all provenance dimensions form a commutative semiring.*

Proof:(sketch) This proof is an immediate consequence of the laws of commutative semirings defined in [Green et al., 2007]. Suppose that the provenance algebra $(\Omega_\gamma, \wedge_\gamma, \vee_\gamma, \top_\gamma, \bot_\gamma)$ is not a commutative semiring with the following identities (see Definition 3.4.5, and Definition 3.4.6): *union* is associative, commutative and has identity; *and* is associative, commutative and distributive over *union*; *projections* and *selections* commute with each other as well as with *unions* and *joins*. However, according to [Green et al., 2007] these identities hold for provenance algebra if and only if $(\Omega_\gamma, \wedge_\gamma, \vee_\gamma, \top_\gamma, \bot_\gamma)$ is a commutative semiring, which contradicts our assumption. Hence the provenance algebra is a commutative semiring, i. e. algebraic structure $(\Omega_\gamma, \wedge_\gamma, \vee_\gamma, \top_\gamma, \bot_\gamma)$ such that $(\Omega_\gamma, \wedge_\gamma, \top_\gamma)$ and $(\Omega_\gamma, \vee_\gamma, \bot_\gamma))$ are commutative monoids, $\wedge$ is distributive over $\vee$ and $\forall a, 0,\ a \wedge 0 = a \wedge 0 = 0$■

To illustrate an instance of this theorem, consider the provenance formula $F_1 \wedge F_2$ and $F_2 \wedge F_1$ for the graph pattern expressions $P = (P_1 \text{ AND } P_2)$ and $P' = (P_2 \text{ AND } P_1)$. Evaluating them in any of the provenance dimensions, we indeed get the same value from the interpretation.

3.6. Tasks and Benefits

This section summarizes the discussed steps of provenance representation and utilization for the sample scenario that was introduced in Section 3.2.

Tasks for Administrators

In order to represent and utilize provenance, the system administrator has to make some design choices. In particular, the application-specific provenance dimensions must be defined. In our sample scenario, we consider three provenance dimensions: source, certainty, and timestamp. In the next step, the administrator defines the intended semantics of these dimensions in order to facilitate query processing with complex expressions and pattern combinations. For evaluating graph pattern expressions, we use the definition from Section 3.4.2, and we assume that the corresponding definitions for provenance dimensions are defined, e.g., the fuzzy logic operations presented in Example 3.4.10.

Finally, data and available associated provenance are represented in RDF using named graphs [Carroll et al., 2005, Carroll and Stickler, 2004], and are imported into our RDF$^+$-based repository.

Processing Performed by the System

We assume that the system imports the small sample dataset introduced in Section 3.2. The knowledge base is transformed into the RDF$^+$ quintuples shown in Example 3.4.2 as discussed in Section 3.3. Associated provenance is transformed into further RDF$^+$ literal statements and RDF$^+$ provenance statements. For the dimensions *prov:time* and *prov:certainty* an example is presented in Example 3.4.14.

Following our sample scenario, the query from Example 3.4.1 can be reformulated as the query from Example 3.4.15 which retrieves names of movie experts together with their occupations and the associated provenance.

Internally, the query processor evaluates this query using graph patterns as discussed in Section 3.4.2. If P denotes the graph pattern from this query then all matches for all variables

in P are given by the evaluation $[[[P]]]$. The resulting set of annotated variable assignments is shown in Example 3.4.8. It contains possible variable assignments, and the how-provenance ($\mathfrak{S}_3$) that explains how these source statements have been used.

By combining this information with definitions for provenance dimensions and available provenance statements, the query processor constructs the result shown in Example 3.4.16. This result is then serialized in RDF.

Benefits for Users and Developers

The user or application developer can access the knowledge stored in the RDF$^+$-based repository in different ways. On one hand, the repository does not change the existing SPARQL semantics and thus fully supports common SPARQL queries. This is an important advantage for compatibility with existing applications and interfaces. On the other hand the repository supports the SPARQL with provenance (Section 3.4.2). Thus, the user obtains additional access to valuable provenance that can be used for relevance ranking, conflict resolution, or other applications in connection with retrieved knowledge.

In our application scenario, the user may realize that the query answer is potentially contradictory (James Cameron is a film producer and a truck driver). By inspecting the associated provenance, he would realize that the second fact was generated by mistake. In fact, it is based on outdated information (knowledge from the document *myspace.com/deprecated.doc* with timestamp *6/6/1980* that was wrongly combined with knowledge from a more recent source (namely document *dbpedia.org/data/JamesCameron.xml* with timestamp *5/5/2014*. It turns out that the occupation of James Cameron has actually changed from truck driver to film producer, and the erroneous tuple can be safely excluded from further processing.

3.7. Complexity

In this section we analyze how the construction of the annotations influences the complexity of the decision problem related to SPARQL. The decision problem associated with the standard evaluation of a SPARQL query can be stated as following [Pérez et al., 2009]: *Given an RDF dataset D, a graph pattern P and a mapping μ, determine whether μ is in the result of P applied to D.* For this decision problem, which we denote by $Eval_D(P)$, an analysis of the complexity is presented in [Pérez et al., 2009, Arenas et al., 2009]. In the context of RDF$^+$ datasets and annotated variable assignments we have a slightly different decision problem: *Given an RDF$^+$ dataset D^+, an RDF$^+$ graph pattern P, a variable assignment μ and an annotation α determine whether α is the correct annotation of μ.* We denote this problem by $Eval^+{}_{D^+}(P)$. An annotation is correct if the formula is *equivalent* (in the logical sense) to the formula obtained by evaluation as defined in Section 3.4.

With the following theorems we show the time complexity and space complexity of $Eval^+$. Note that the RDF$^+$dataset D^+ is built over a set of RDF literal statements and provenance statements of the RDF dataset D (see Section 3.3.4). For $D^+ = (K, M)$, we assume that the size of $|D^+| = |K| \cdot |M| \leq 1/2\ |D|^2$, and consequently the time complexity of building D^+ is the order of $O(|D|^2)$.

The RDF counterparts of both theorems have been established by [Pérez et al., 2009, Arenas et al., 2009]. Likewise we restrict to graph patterns which do not contain blank nodes. In the first theorem we consider graph patterns which use only AND and FILTER operations.

Bindings obtained by such patterns are annotated with formulae which do not contain any other operator besides $\wedge$ according to Definition 3.4.4.

Theorem 3.7.1 *$Eval^+$ can be solved in time $O(|P| \cdot |D^+|)$ for graph pattern expressions constructed by using only AND and FILTER operators and for annotation formulas using only the operation $\wedge$.*

Proof: The proof consists of two parts: In the first part we construct a correct annotation $\hat{\alpha}$ for μ and in the second we check whether α and $\hat{\alpha}$ are equivalent. In the following let p_i denote pattern i from P and $\mu(p_i)$ denote the triple obtained by replacing the variables in the pattern p_i according to μ.

In order to construct $\hat{\alpha}$ we start by evaluating $[\![[\![p_i]\!]]\!]^{D^+}$ for all pattern using Definition 3.4.3. In order to achieve this D^+ needs to be searched for all $\mu(p_i)$. This can be performed in $O(|P| \cdot |D^+|)$. Then, we construct the algebraic expression Φ by evaluating $[\![[\![P]\!]]\!]^{D^+}$ using definition 3.4.4. This can be performed by traversing P. The correct annotation $\hat{\alpha}$ of μ in the result of evaluating on D^+ is defined as $\Phi(\mu)$ (see Definition 3.4.5). We can construct $\hat{\alpha} = \Phi(\mu)$ in a depth-first traversal of Φ as following:

Let Φ_1, Φ_2 be algebraic expressions part of Φ. For each join operation in Φ the annotation of $(\Phi_1 \bowtie \Phi_2)(\mu)$ is given by $\Phi_1(\mu) \wedge \Phi_2(\mu)$. For each select operation $\sigma_c(\Phi)(\mu)$ is given by $\Phi(\mu)$ if condition c is fulfilled otherwise it is given by $\bot$. Since traversing and Φ each has time complexity $O(|P|)$ the evaluation of $\hat{\alpha} = \Phi(\mu)$ remains in $O(|P| \cdot |D^+|)$.

Now we determine whether for a given annotation α holds $\alpha \equiv \hat{\alpha}$. We transform both formulas into a normal form in $O(|P| \cdot \log|P|)$ using associativity, commutativity and idempotency of $\wedge$. First, we remove all brackets then establish an order among atomic formulas (identifiers and $\top, \bot$) and finally remove duplicates. For α and $\hat{\alpha}$ normalized this way syntactic equality bi-implies logical equivalence. Assuming $|P| < |D^+|$ the overall complexity remains in $O(|P| \cdot |D^+|)$. ■

Theorem 3.7.2 *$Eval^+$ is NP-complete for graph pattern expressions constructed by using only AND, FILTER and UNION operators.*

Proof: The proof consists of two parts: In the first part we show that $Eval^+$ is contained in NP and in the second we establish NP-hardness of $Eval^+$. As above, let p_i denote triple pattern i from P and $\mu(p_i)$ denote the triple obtained by replacing the variables in the pattern p_i according to μ.

The first part consists of two steps as well: first we construct a correct annotation $\hat{\alpha}$ for μ and then we check whether α and $\hat{\alpha}$ are equivalent. In order to construct $\hat{\alpha}$ we start by evaluating $[\![[\![p_i]\!]]\!]^{D^+}$ for all patterns using definition 3.4.3. In order to achieve this D^+ needs to be searched for all $\mu(p_i)$. Then, we construct the algebraic expression Φ by evaluating $[\![[\![P]\!]]\!]^{D^+}$ using Definition 3.4.4. This can be performed by traversing P. The correct annotation $\hat{\alpha}$ of μ in the result of evaluating P on D^+ is defined as $\Phi(\mu)$, see definition 3.4.5. We can construct $\hat{\alpha} = \Phi(\mu)$ in a depth-first traversal of Φ as following:

Let Φ_1, Φ_2 be algebraic expressions part of Φ. For each join operation $(\Phi_1 \bowtie \Phi_2)(\mu)$ is given by $\Phi_1(\mu) \wedge \Phi_2(\mu)$. For each union operation $(\Phi_1 \cup \Phi_2)(\mu)$ is given by $\Phi_1(\mu) \vee \Phi_2(\mu)$. For each select operation $\sigma_c(\Phi)(\mu)$ is given by $\Phi(\mu)$ if condition c is fulfilled otherwise it is given by $\bot$.

Time complexity for evaluating $\Phi(\mu)$ is $O(|P| \cdot |D^+|)$. The evaluation of $\alpha \equiv \hat{\alpha}$ is more difficult if UNION operations are contained in P. But it is subsumed by checking equivalence

of Boolean formulae which is a NP-complete problem. Thus, the decision problem $Eval^+$ is contained in NP.

We can deduce that $Eval^+$ is an NP-complete problem if $Eval$, which has been shown to be NP-complete [Pérez et al., 2009, Arenas et al., 2009], can be reduced to it. $Eval$ can be reduced to $Eval^+$ if the evaluation defined in Section 3.4 results in an annotation $\alpha \equiv \bot$ exactly for such bindings which are not in the result of standard SPARQL evaluation. ■

3.8. Evaluation

The framework *metaK* described in this work has been implemented in Java language, using libraries from the Sesame Project[8].

From a practical point of view, the framework can easily be customized for new provenance aspects. Dimension-specific interpretations of provenance dimensions are implemented separately as small Java classes. For defining a new provenance aspect, the corresponding interpretation function must be implemented with respect to framework interfaces.

In order to evaluate the overhead produced by the evaluation of provenance dimensions for results of SPARQL queries, we carried out two experiments based on the well-known LUBM benchmark [Guo et al., 2005]. Our main aim was to find out whether the evaluation of SPARQL queries remained feasible if provenance was provided within the query results, i.e., conduct a family of experiments to validate the theoretical results of Section 3.7.

A key question is how to separate the additional effort for the evaluation of provenance and provenance from standard SPARQL processing. Triples describing provenance receive an *additional* provenance interpretation according to section 3.3. At the same time they are also treated as ordinary RDF triples (and thus can be queried using standard SPARQL). Thus, adding provenance to a knowledge base increases its overall size which also increases the workload for standard query processing. In order to account for this we compare query evaluations performed on the same knowledge base which includes provenance. Our implementation is built on top of Sesame[9] 2.0 (beta 6) using query rewriting. The triple store is used to store both, knowledge and provenance. We expect that a native implementation of the provenance framework can achieve an increased performance.

3.8.1. Methodology

For the evaluation of SPARQL on RDF$^+$we defined the results of SELECT queries to be set-valued (see Section 3.4.2). For standard SPARQL [Prud'hommeaux and Seaborne, 2008], however, a SELECT query returns a *solution sequence* which may contain duplicate elements. All 14 queries of the LUBM benchmark are SELECT queries. To allow for a consistent comparison of extended evaluation on one side and standard evaluation on the other we added the keyword DISTINCT to the queries for standard evaluation. This tells the query processor to eliminate duplicates. Since the evaluation of individual provenance dimensions can be arbitrarily complex we compare the following three kinds of query evaluation:

- Standard evaluation with additional duplicate elimination (SD), as performed by Sesame,

[8]Sesame Project: <http://www.openrdf.org>

[9]Sesame: open source framework for storage, inferencing and querying of RDF data (`www.openrdf.org`)

- Evaluation of provenance formulae for each query result (PF) and
- Evaluation of four basic provenance dimensions (M4), namely *agent*, *confidence*, *creation time* and *source*, see Example 3.4.10 (we evaluate *agent* analogously to *source*).

Only the last type of query processing actually makes use of the additional triples.

3.8.2. Data

We added artificial provenance to the LUBM data. Amount and granularity of the additional provenance are key dimensions of the resulting dataset.

We created two datasets containing a different percentage of provenance triples. The first dataset (MK29) was created based on LUBM OWL data for ten universities. We added the provenance by putting groups of ten consecutive original triples into one named graph and associating random values of the provenance dimensions *agent*, *confidence*, *creation time* and *source* with this graph name, see Example 3.8.1. As a consequence the dataset contains 29 percent of provenance which might be reasonable in a real-world scenario. The resulting dataset consists of 1.8 million triples of which 1.3 million triples were created by the LUBM generator to which we added 0.5 million provenance triples. It contains 0.3 million (additional) graph URIs.

Example 3.8.1 *Original triples of dataset MK29 in one named graph associated with provenance dimensions*

```
<graph>
  <uri>http://www.x-media-project.org/
       ontologies/someGraph#0-0_30</uri>
  <triple>
    <uri>http://www.Department0.University0.
                  edu/FullProfessor1</uri>
    <uri>http://www.lehigh.edu/%7Ezhp2/2004/
    0401/univ-bench.owl#doctoralDegreeFrom
    </uri>
    <uri>http://www.University882.edu</uri>
  </triple>
  <triple>
    <uri>http://www.University882.edu</uri>
    <uri>http://www.w3.org/1999/02/
                  22-rdf-syntax-ns#type</uri>
    <uri>http://www.lehigh.edu/%7Ezhp2/2004/
    0401/univ-bench.owl#University</uri>
  </triple>
...
<graph>
  <uri>http://www.provenance.semanticweb.
                                org0-0_30</uri>
  <triple>
    <uri>http://www.x-media-project.org/
```

```
        ontologies/someGraph#0-0_30</uri>
    <uri>http://www.x-media.org/ontologies/
                            metaknow#source</uri>
    <uri>http://www.x-media-project.com/
                              ex#source30</uri>
</triple>
<triple>
    <uri>http://www.x-media-project.org/
            ontologies/someGraph#0-0_30</uri>
    <uri>http://www.x-media.org/ontologies/
             metaknow#confidence_degree</uri>
    <typedLiteral datatype='http://www.w3.org/
      2001/XMLSchema#double'>0.7</typedLiteral>
</triple>
    ...
```

The dataset MK400 was created based on LUBM OWL data for three universities and we assigned each triple to a different graph and the four provenance dimensions are associated with it. This way the knowledge base contains four times as many provenance triples as knowledge triples. The resulting dataset consists of 1.7 million triples of which 0.35 million are original LUBM data and 1.4 million are additional metadata. Note that the overall sizes of the two datasets are comparable. As indicated above, the overall size of the knowledge base influences the workload for query processing. In fact, main memory consumption appeared to be a key factor. By choosing similar sizes for the two datasets we reduce the influence of factors related to the size of the knowledge base and concentrate on how the different evaluations influence the processing time.

	SD	PF	M4
MK29	313	894	1382
MK400	196	104	425

Table 3.1.: Random query sequence experiments: average processing time (ms)

3.8.3. Discussion and Results

Random Query Sequence.

We conducted two experiments with each of the two datasets. One experiment aims at simulating behavior of a query engine in a real- life scenario: first a dataset is loaded and then a sequence of queries is submitted to the query engine. We measured the average processing time of these queries. The query sequence was created based on 8 of the 14 queries from the LUBM benchmark. Query 2 was not included since we were not able to obtain results for this query with this version of Sesame, this dataset and an average machine. Five more queries were discarded since they require OWL inferencing or hierarchy information to obtain complete results and Sesame 2 was not able to obtain bindings using plain SPARQL processing.

Since no provenance needs to be calculated if the result set is empty, using these queries would bias the evaluation in favor of the provenance processing. The authors of the benchmark identified three main characteristics of queries with respect to plain SPARQL processing: *input size*, *selectivity* and *complexity*. The remaining 8 queries still cover different settings for these features. Experiments with single queries will be presented below.

The query sequence consisted of a random shuffle of 20 copies of each of the remaining queries. For each query in the sequence we measured the time which elapsed during issuing the query, obtaining the result and traversing the result sequentially. This measure is similar to the *query response time* defined in [Guo et al., 2005]. The only difference is that in [Guo et al., 2005] each query was performed ten times after the knowledge base had been loaded to measure the caching performance of the query engine. For each run we determined the average of the execution times of the queries in the sequence in question. The results are summarized in Table 3.1.

Evaluation of provenance formulae (PF) given dataset MK29 almost tripled the average query execution time. On average, the evaluation took about half a second longer than standard evaluation (SD). The evaluation of four provenance dimensions (M4) adds again half a second to the evaluation of provenance. The overall overhead to obtain provenance was about a second, given a non-trivial dataset. We consider this to be an indication for the feasibility of our approach. Since these numbers are average values the question remains whether reasonable processing times may be achieved for all individual queries as well. This question will be analyzed below.

For the dataset MK400 the computations were faster for all three kinds of evaluations. This can be explained by the fact that this dataset contains a smaller number of knowledge triples and therefore the result sets for some of the queries contain fewer bindings as we will see below. Surprisingly, evaluation of provenance formulas needed less time than standard evaluation. A possible explanation is optimization performed by the query processor of Sesame 2. Since our implementation uses query rewriting, different queries are evaluated for the different kinds of evaluations. Optimization techniques might be easier to apply to some of them. Why does the evaluation of MK29 not show similar characteristics? Here a possible explanation is the larger number of bindings involved in the evaluation. Main memory consumption was quite large in both cases. Possibly there was no space left for caching of (intermediary) results given dataset MK29. The calculation of the four provenance dimensions did result in an increase in processing time, as expected.

Single LUBM Queries.

We also measured processing times for single queries. As stated above, we measured the time which elapsed during issuing the query, obtaining the result and traversing the result sequentially. In contrast to the previous experiment and to the definition of the *query response time* from [Guo et al., 2005] the application was restarted before each single query execution. That way each query was evaluated against a newly loaded knowledge base since we wanted to measure the effort it takes to evaluate the queries and not the caching strategy of the query engine. This procedure was repeated ten times for each query and each method of query evaluation. The average values from these runs are summarized in tables 3.2 and 3.3. The standard deviations estimated from these ten runs are less than or equal to 10 percent for all queries and methods evaluated on the two datasets.

Query		*Q1*	*Q4*	*Q5*	*Q6*	*Q7*	*Q8*	*Q10*	*Q14*	*Av.*
# bindings		5	10	411	24019	3	1874	4	75547	
processing time (ms)	SD	127	163	211	357	135	1619	127	977	465
	PF	324	347	386	995	337	691	325	2320	716
	M4	340	375	426	2187	346	878	326	10501	1922

Table 3.2.: Processing time of query evaluation with and without provenance and provenance for individual queries and the MK29 dataset. Processing times are average values of 10 runs each.

Query		*Q1*	*Q4*	*Q5*	*Q6*	*Q7*	*Q8*	*Q10*	*Q14*	*Av.*
# bindings		5	10	411	6390	3	1874	4	19868	
processing time (ms)	SD	137	189	204	217	131	1596	124	379	372
	PF	367	350	367	517	316	676	300	769	458
	M4	346	359	440	859	321	1083	306	1953	708

Table 3.3.: Processing time of query evaluation with and without provenance and provenance for individual queries and the MK400 dataset. Processing times are average values of 10 runs each.

On average the calculation of provenance formulas (PF) increases processing time by factor 1.5. In absolute numbers the average increase is about 0.2 seconds. The largest increase (1.3 seconds for the MK29 dataset) can be observed for query 14 which also gives the largest result set. We attribute this to the main memory consumption of our implementation. For query 8 there is even a decrease in processing time. A random query sequence evaluated on MK400 may possibly be explained by optimizations of the query processor. The additional querying for provenance might guide the optimization of the query execution.

Calculation of the four basic provenance dimensions (MD4) causes an average increase of factor 4.1 (1.5 seconds) for the MK29 dataset and factor 1.9 (0.3 seconds) for the MK400 dataset compared to standard evaluation. We ascribe the larger increase for the runs based on dataset MK29 to the larger number of results for queries 6 and 14 and the non-optimized memory consumption of our implementation. These results are in line with the results from the experiments using a random query sequence shown in table 3.1. For query 8 the processing time decreases for query evaluation including provenance which might be explained by query optimization of the underlying triple store as stated above.

Discussion.

At first we consider the experiments with individual queries. From the estimated standard deviations of the 10 runs for each query, kind of evaluation, and dataset we conclude that the measurements are reliable enough to draw two general conclusions: On the one hand we can observe a noticeable increase of processing time using our implementation and on the other hand the amount of this increase can be described by a small linear factor.

The results of the experiments using a random sequence of queries indicate that similar results also hold for real-world scenarios. Here caching can be applied by a query processor.

Several evaluations of the same query were repeated directly one after another in the sequence. The resulting average processing times were smaller but still comparable to the average values for the evaluations of single queries. A key insight is that the overall processing times remain feasible for evaluations on a dataset of up to 1.8 million triples and results of up to 75,000 bindings.

If we compare the processing times obtained for the two different datasets we might expect that for dataset MK400 processing times would increase by a larger factor if provenance were involved. MK400 contains a larger number of provenance triples which need to be processed in order to evaluate provenance – especially compared to the number of knowledge triples. However, at least with our implementation, the dominant characteristic of query evaluation appears to be the size of the result set.

3.9. Related Work

Better understanding the ways by which results came about is fundamental to many Semantic Web applications and scenarios. The specification of the Semantic Web proof layer was discussed in [Murdock et al., 2006, da Silva et al., 2006, McGuinness and da Silva, 2004].

Provenance in Data Management Systems based on Relational Algebra

Provenance has been initially studied as an extension for relational databases (i. e. data management systems based on relational algebra), probabilistic databases [Fuhr and Rölleke, 1997], and later adopted for RDF knowledge bases (i. e. for semantics of SPARQL query language). In the area of database systems, provenance is often represented using an extension of the relational data model, coined *annotated relations*. Their purpose is primarily the description of data origins (provenance) and the process by which it arrived as a query answer [Cui and Widom, 2000, Buneman et al., 2000, Buneman et al., 2001, Ding et al., 2005]. The authors define custom (possibly different) interpretations for algebraic operations of Boolean formulas (built on tuple identifiers as Boolean variables) for particular dimensions of provenance (e.g., agent, timestamp, source, certainty) to obtain the m-dimensional record (i.e., query result). Indeed, a Boolean expression of the result set build from tuple identifier not only tells us *which* triples have contributed to a variable assignment (why-provenance) but also *how* they contributed (how-provenance). This information draw a distinction between where-provenance (where the given pieces of data are physically serialized in database tuples), why-provenance (which subset of database tuples contributed to the result), and how-provenance (how particular tuples were used for constructing the result).

Basically, our methodology follows the same idea and adopts the notion of provenance semirings which was introduced as a generalization framework in [Green et al., 2007] and also allows the same distinction between where-, how-, and why-provenance. In contrast to other approaches discussed in [Cui and Widom, 2000, Buneman et al., 2000, Buneman et al., 2001, Ding et al., 2005] (based on the notion of annotated tuples in a relational schema), our solution is customized for RDF graphs (i. e. annotated RDF quadruples). The same holds for the corresponding query language (basically, SPARQL vs. SQL) and its semantics. An important conceptual difference to the relational model is the natural ability of RDF/SPARQL repositories for result serialization and thus seamless exchanging and utilization of knowledge

and provenance from our framework across multiple Semantic Web nodes without additional schema integration efforts.

Provenance and the Semantic Web

In the Semantic Web field, provenance has been recently considered in applications for assessing the trustworthiness of information [Carroll et al., 2005, Ding et al., 2005]. Hartig [Hartig, 2009b] presents a trust-aware extension to SPARQL, tSPARQL. Hartig first proposes a trust model that associates RDF statements with trust values and allows us to extend the SPARQL semantics to access these trust values in tSPARQL. In [Bonatti et al., 2011], Bonatti et al. formalise a logical framework for determining trust to perform robust reasoning. In [Schenk, 2008], Schenk models multiple levels of trust on Web data to resolve inconsistencies arising from connecting multiple data sources. The version control aspect of provenance such as temporal RDF [Gutiérrez et al., 2005] and querying time in RDF [Tappolet and Bernstein, 2009, Gutierrez et al., 2007] and OWL [Motik, 2012] has been tackled by attaching temporal annotations to RDF facts and OWL axioms.

Our approach is focused on an RDF language model and provides fine-grained provenance management for retrieval queries with SPARQL that is not directly comparable with proof traces for OWL reasoning.

The use of Named Graphs for provenance management and querying RDF with provenance constraints is introduced in [Carroll et al., 2005]. As discussed in Section 3.4.4, this functionality can be realized in our framework in a straightforward manner by the means of *nested* queries (or querying from *views*), which are expected features of the upcoming SPARQL2 query language. Moreover, our framework potentially allows us to express filtering and/or ranking conditions on provenance of *query results* and thus is more expressive and flexibe than the solution presented in [Carroll et al., 2005].

Provenance mechanisms for RDF datasets have been investigated by Udrea et al. [Udrea et al., 2010] where they provide a semantics for RDF augmented with a partially ordered set and algorithms for query processing and view maintenance. As our approach does not consider the OPTIONAL SPARQL constructor, follow-up approaches to ours [Theoharis et al., 2011, Karvounarakis and Green, 2012, Geerts et al., 2013, Karvounarakis et al., 2013], discuss the need for extending the relational provenance models to be leveraged for SPARQL queries over RDF. In particular, they state that the use of semiring models for SPARQL has been shown inadequate to handle the OPTIONAL construct, and advocated the need for a new abstract provenance model capturing the full expresiveness of SPARQL. Geerts et al. [Geerts et al., 2016] identify SPARQL fragments for which provenance models for positive relational queries can be leveraged, despite the subtle differences between the semantics of SPARQL and relational algebra operators. In [Amsterdamer et al., 2011c], Amsterdamer et al. show particular semirings for which an extension for supporting difference is impossible. The consideration of extended provenance models for capturing the semantics of both explicit and scoped weak negation is also considered in [Analyti et al., 2014]. In [Damásio et al., 2012], Damasio et al. proposed a provenance model for a significant fragment of SPARQL 1.1, based on the relational annotated provenance models including for non-monotonic constructs under multiset semantics.

3.10. Findings and Research Contribution

In this chapter, we have presented an original, generic, formalized and implemented approach for the management of many dimensions of provenance, like source, authorship, certainty, and others, for RDF repositories. Our method re-uses existing RDF modeling possibilities in order to represent provenance. Then, it extends SPARQL query processing in such a way that given a SPARQL query for data, one may request provenance without modifying the query proper. We achieve highly flexible and automatically coordinated querying for data and provenance, while completely separating the two areas of concern. Our approach remains compatible to existing standards and query languages and can be easily integrated with existing applications and interfaces.

4. Reasoning and Debugging Evolving OWL Ontologies with Provenance

Overview

For many tasks, such as the integration of knowledge bases in the Semantic Web, one must not only handle the knowledge itself, but also characterizations of this knowledge, e.g.: (i) where did a knowledge item come from? (ii) what level of trust can be assigned to a knowledge item? or (iii) what degree of certainty can be associated with it? We refer to all such kinds of characterizations as *provenance* of the information. Approaches exist for providing provenance for query answers in relational databases and RDF repositories, based on algebraic operations. As query answering in description logics in general does not boil down to algebraic evaluation of tree-shaped query models, these formalizations do not easily carry over. In this chapter, based on a formalization of provenance proposed in [Schenk et al., 2011], which is still algebraic, but allows for the computation of provenance of inferred knowledge in description logics (and includes reasoning with conflicting and incomplete provenance), we present an optimized algorithm for computing provenance. Our black-box algorithm for reasoning with provenance, which uses pinpointing to come up with provenance formulas for description logics, enables the use of provenance for debugging in real time even for very large and expressive ontologies, such as those used in biomedical portals.

Structure

The remainder of this chapter is structured as follows: In Section 4.2 we formalize a use-case scenario to motivate the use of provenance for tracking changes in ontologies. Section 4.3 introduces the foundations for provenance computation: first we briefly introduce an extension of the description logic DL, called $\mathcal{SRIQ}(\mathcal{D})$, underlying OWL lite and OWL DL, followed by the formalization of pinpointing and existing algorithms for pinpoint computations, and the formalization of provenance orders.

In Section 4.4 we define the semantics of provenance, including the composition of provenance dimensions (to model complex provenance) and merging of conflicting provenance. In Section 4.5 we define the computation of provenance for a query answer based on pinpointing and propose an optimized algorithm for debugging with provenance. We discuss the issue of complexity in Section 4.6. Section 4.7 presents the evaluation results. Our evaluation shows that this algorithm performs orders of magnitude better than a naïve implementation. A review of related work and a comparison of this work with our approach is presented in Section 4.8.

4.1. Introduction

Ontologies often evolve in open, multi-user ontology editing environments. Examples of open, evolving ontologies are the Open Biomedical Ontologies repository of the US National Cancer Institutes Center for Bioinformatics [Smith et al., 2007] or the Ontology for Biomedical Investigations (`http://obi-ontology.org`), which is an integrated ontology for the description of life-science and clinical investigations. Many similar projects exist in varying domains. They are characterized by an open community with contributions at various points in time and from various sources, which might not be equally reliable.

In such open settings conflicting changes can occur that may be contributed by various users and knowledge sources at different points in time, and it is desirable to track two unwanted situations:

1. Undesired inferences and

2. Inconsistencies

In order to judge the reliability of inferences and to find errors in the ontology when debugging ontologies, it is hence necessary to ask questions such as "When has this inconsistency been introduced and who is responsible for this change?" as well as "Can I trust this inference?". Provenance information is used to answer these questions.

Provenance can be tracked in many dimensions: knowledge source, last recently modified dates, degrees of trustworthiness, or the experience level of the editor. In all these cases, we have provenance labels, which are attached to axioms (e.g. timestamps) and orders over these labels, such as the ascending or descending order of timestamps or a partial order of trust among sources. This provenance information must be combined and propagated to a conclusion in order to answer the above-mentioned questions.

Standard algorithms for debugging ontologies poorly support the user in answering provenance questions and require expensive reasoning. For these reasons they are not applicable for expressive and large-scale real-world ontologies. With the approach presented in this paper we will show how to represent provenance and efficiently reason in OWL with provenance. Our approach supports the user in coping with the complexity and dynamics of evolving ontologies.

Various approaches to the problem of debugging with provenance have been proposed. They can be grouped into three categories: (a) Extensions of given logical formalisms that deal with a particular type of provenance. Examples include extensions for debugging with uncertainty, such as fuzzy and probabilistic [Lukasiewicz and Straccia, 2008] or possibilistic [Qi et al., 2011] description logics. In [Schenk, 2008] generalize such works to consider partial trust orderings. (b) Flexible extensions for systems allowing for algebraic query evaluation (e.g. as relational databases and SPARQL engines), as discussed in [Dividino et al., 2009b], [Buneman et al., 2001] and [Lopes et al., 2010], allow for many kinds of provenance, but are limited to lower expressiveness of the underlying logical formalism. (c) [Tran et al., 2008] provides a two-step evaluation for provenance, which is very expressive, but which does not assign a uniform semantics to the definition and composition of provenance in the knowledge base.

Expressive descriptions of provenance combined with less expressive base languages (such as SPARQL and SQL) [Dividino et al., 2009b, Buneman et al., 2001, Lopes et al., 2010] make use of the fact that the base languages can be evaluated bottom-up using relatively simple

algebraic expressions. However, debugging frameworks frequently have non-tree-based derivations used for consistency checking and querying. In order to be able to debug with algebraic provenance on top of such expressive base languages, Schenk et al. [Schenk et al., 2011] propose a debugging framework for provenance based on *pinpointing*. Pinpointing summarizes explanations for axioms in a single Boolean formula, which then can be evaluated using a provenance algebra. Consequently, provenance for query answers can be computed by computing the explanations of the answer and using the pinpointing formula to compute provenance in an algebraic way.

Unfortunately, the computation of pinpoints may become very expensive and inapplicable if users need to interact with dynamically changing knowledge in real time. Therefore, we provide an optimized black-box algorithm, which does not need to compute *all* pinpoints [1]. Our evaluation shows that our algorithm performs significantly better than the naïve algorithm, based on both real-world and synthetic datasets. We restrict ourselves to a fragment of description logics and of OWL-2, called $\mathcal{SRIQ}(\mathcal{D})$, and use OWL-2 as a base language and consistency checks as debugging task.

Finally, we will show an evaluation of our approach using real-world data from evolving ontologies. For the sake of example, we present our framework restricted to ontology diagnosis scenarios. This work, however, introduces an approach for provenance querying under a variety of scenarios such as restrictions of access rights [Baader et al., 2009], knowledge validity when the truth of knowledge changes with time [Motik, 2012], and inferring trust value [Hartig, 2009b].

Research Questions

This chapter addresses the following research questions.

RQ 1.IV *Does the exploitation of provenance lead to computation overhead?*

OWL is heavily based on Description Logic (DL), i. e. its model-theoretic semantics is compatible with the semantics of Description Logics. Typical reasoning tasks over an expressive DL (e. g. using tableau methods to perform consistency checking, instance checking, satisfiability checking, etc. [Baader et al., 2003, Rudolph, 2011]) are in the worst case doubly-exponential, and in practice are often likewise very expensive.

Standard algorithms for debugging ontologies poorly support the user in answering provenance questions and also require expensive reasoning. For these reasons, even though the exploitation of provenance helps users to find undesired inferences and inconsistencies in evolving ontologies, such algorithms are not applicable for expressive and large-scale real-world ontologies.

4.2. Motivation Scenario

We use as toy-example a scenario based on a serious faux pas, which actually happened to the German Press Agency DPA.

[1] Initial work towards the approach presented in this chapter has been published in [Schenk et al., 2009, Schenk et al., 2011]. We extend this work with an optimized algorithm for computing provenance and a comprehensive evaluation.

On September 10^{th} 2009, a person called the German Press Agency (DPA) and notified them that a terrorist attack had just taken place in Bluewater, CA. In order to check the accuracy of this information, DPA did a short Internet research and found a website for Bluewater's local TV station (VPKTV) and for the town itself. Additionally, they found Wikipedia entries for both the town and the local TV station. DPA announced the attack as breaking news.

Unfortunately for DPA, the information was fake. No terrorist attack had happened in Bluewater; in fact, the town Bluewater, CA, does not even exist. The breaking news had been spread and the fake background information had been set up by the marketing agency Neverest to support guerrilla marketing for the new movie *Shortcut to Hollywood*.

Even though DPA relied on the information found on web pages and Wikipedia entries to make the decision on announcing as breaking news a supposed terrorist attack, DPA did not take into account the dynamics and the provenance of the information retrieved when checking the plausibility of the news. The new Wikipedia article had been created by the same author and modified multiple times shortly before the call. Moreover, all related websites as well as the Wikipedia articles had been set up by the same marketing agency at roughly the same time.

This situation is common, as Wikipedia is used nowadays as an information source by many people every day. However, since Wikipedia entries can be edited by anyone, people have begun to question their quality. The validity and quality of Wikipedia entries depend on the community knowledge, and unfortunately, Wikipedia cannot assure the reliability of its sources.

When exploiting knowledge found in external sources, such as the Semantic Web, the user must be able to judge the appropriateness of such knowledge, i.e. he must discriminate knowledge and its validity based on where it comes from, how much he trusts the knowledge sources, and the degree of (un)certainty in such knowledge. We refer to all such kinds of characterizations as provenance.

To illustrate our method in a concise fashion, Table 4.1 shows the instantiation of our scenario. We suppose that the webpages of Bluewater and VPKTV are described by axioms of ontologies from a single source, data from Wikipedia corresponds to an open, Wiki-like ontology editing system such as Freebase[2], and the provenance consists of modification dates and respective degrees of trustworthiness based on the information source.

The axioms of Table 4.1 describe that a *RealCity* is a *City* with at least one *Broadcaster* (# 1) and a *Broadcaster* has its headquarters in a *City* (# 2). Additionally it defines that for any *Broadcaster* that has its headquarters in a *City* then this *City hasCompany* which is the *Broadcaster* (#3), that *bluewater* is a *City* (#4), *vpktv* is a *Broadcaster* (#5), and *vpktv* has its headquarters in the city *bluewater* (#6). Furthermore the columns provenance and *asserted* of Table 4.1 show provenance associated with each axiom. For instance, *DPA*, the German Press Agency, created axiom #1, *Nev*, the marketing agency Neverest, created axiom #4 and NevWP, their Wikipedia user, created axiom #5. The rightmost column indicates, when axioms have been asserted, for instance, the axiom #1 has been last asserted on September 10^{th} 2009.

DPA needs to verify that Bluewater indeed is a city in CA. Hence, they need to answer the query "*bluewater: RealCity*?" and to obtain provenance for the answer. In the rest of this

[2] `http://freebase.org`

ID	axiom	source	date
#1	$RealCity \sqsubseteq City \sqcap \exists hasCompany.Broadcaster$	DPA	2009-09-10
#2	$Broadcaster \sqsubseteq \exists hqIn.City$	DPA	2009-09-09
#3	$inverseProperty(hasCompany, hqIn)$	DPA	2009-09-10
#4	$bluewater : City$	Neverest	2009-09-09
#5	$vpktv : Broadcaster$	NeverestWikiP	2009-09-10
#6	$hqIn(vpktv, bluewater)$	NeverestWikiP	2009-09-09

Table 4.1.: Ontology axioms describing our scenario and their provenance.

chapter we will explain how this is done.

To come up with a flexible mechanism, which at the same time supports expressive logics and multiple kinds of provenance, a suitable formalization of provenance in a semantically precise manner is needed. Moreover, such a mechanism must be supported with a suitable operationalization. The DPA example above illustrates that varying provenance such as authorship, timestamp, and trust in individual sources must be exploited and combined to arrive at an accurate assessment of information value. In the following sections we introduce such a mechanism.

4.3. Foundations: Pinpointing

The term "pinpointing" has been coined for the process of finding explanations for concluded axioms or for a discovered inconsistency. A pinpoint is a minimal subset of an ontology, which makes the concluded axiom true or the theory inconsistent, respectively. Such an explanation is called a pinpoint. While there may be multiple ways to establish the truth or falsity of an axiom, a pinpoint describes exactly one such way.

Definition 4.3.1 (Pinpoint) *A pinpoint P for an axiom A wrt. an ontology O is a set of axioms, such that $P \subseteq O$, $P \vDash A$, and $\forall B \in P : P \setminus \{B\} \nvDash A$.*

Analogously, we can define a justification for a refuted axiom ($O \vDash \neg A$) as $P \subseteq O$, $P \vDash \neg A$, and $\forall B \in P : P \setminus \{B\} \nvDash \neg A$. Hence, finding pinpoints for a refuted axiom corresponds to finding the Minimum Unsatisfiable Subontologies (MUPS) for this axiom [Kalyanpur et al., 2006]. In this work we will focus on entailed axioms ($O \vDash A$). However, all definitions and algorithms can be modified to justifications as well.

Pinpointing is the computation of all pinpoints for a given axiom A and ontology O. The *pinpointing formula* [Baader and Peñaloza, 2010] describes, which axioms need to be present for O to entail A.

Definition 4.3.2 (Pinpointing Formula) *Let A be an axiom, O an ontology and $P_1, ..., P_n$ with $P_i = A_{i,1}, ..., A_{i,m_i}\}$ the pinpoints of A wrt. O. Let* id *be a function assigning a unique identifier to an axiom. Then $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} id(A_{i,j})$ is a pinpointing formula of A wrt. O.*

A pinpointing formula of an axiom A describes, which (combinations of) axioms need to be true in order to make A true (or inconsistent respectively).

Algorithms for finding pinpoints can be grouped into three groups:

Finding one Pinpoint Algorithms for finding one pinpoint can either derive a pinpoint by tracking the reasoning process of a tableaux reasoner, or use an existing reasoner as a black box. In the latter case, a pinpoint is searched for by subsequently growing (or shrinking) a subontology, until it starts (or stops) entailing the axiom under a searched-for pinpoint. Based on the derived smaller ontology the process is refined, until a pinpoint has been found. The advantage of blackbox algorithms is that they can support any DL, for which a reasoner is available [Kalyanpur et al., 2006].

Finding all Pinpoints using a Tableaux Reasoner Baader and Peñaloza have shown how tableaux reasoners for DLs such as OWL can be extended to find pinpointing formulas [Baader and Peñaloza, 2010]. In their approach a tableaux reasoner is extended to find not only one, but all pinpoints. Special care needs to be taken in order to ensure termination of the tableaux algorithm. As an advantage, the overhead for pinpointing is lower compared to a black-box algorithm. Moreover, this approach can derive a compact representation of the pinpointing formula, which however might still have worst-case exponential size in normal form.

Finding all Pinpoints using Blackbox Algorithms The most performant black-box algorithms extract a relevant module from the overall ontology, ensuring that this module yields the same inferences with respect to the axiom in interest. Then, starting from a single pinpoint, Reiter's Hitting Set Tree algorithm [Reiter, 1987] is used to compute all pinpoints by iteratively removing one axiom from the pinpoint at hand and growing it to a full pinpoint again [Kalyanpur et al., 2007, Ji et al., 2009].

For both tableaux-based and black-box algorithms, the worst case complexity of finding all pinpoints is rather high, as there can be exponentially many pinpoints for any given ontology. Our approach is based on this third group. Basically, we need to find all pinpoints to derive the provenance for a (refuted) piece of knowledge. However, since finding all pinpoints is a very expensive operation, we present an optimized algorithm for computing pinpointing formula and deriving provenance.

4.4. Syntax and Semantics for OWL with Provenance

The following section presents the formalization of provenance that allows for the computation of provenance of inferred knowledge in description logics (and includes reasoning with conflicting and incomplete provenance). The formalization of provenance has mainly been developed by Simon Schenk. To aid in understanding, the formalization of provenance will be repeated in this dissertation. The results of our joint work have been published in [Schenk et al., 2011].

4.4.1. Syntax for OWL with Provenance

Provenance in OWL-2 can be expressed as annotations on axioms. Annotations are of importance since the management of ontologies as annotations may be used to support analysis during collaborative engineering.

Basically, an axiom annotation assigns an annotation object to an axiom. For instance, in our scenario the axiom "$RealCity \sqsubseteq City \sqcap \exists hasCompany.Broadcaster$" is assigned the information source annotation object *DPA*.

A provenance annotation consists of an annotation URI and a provenance object specifying the value of the annotation. In our case, the provenance object is a constant-value representing who asserted the axiom, when the axiom was last asserted, the degree of trustworthiness of the axiom, or a combination thereof. We provide a detailed grammar for provenance annotations in [Schenk et al., 2009]. The grammar for provenance annotations as an extension of OWL-2 annotations[3] is as follows. For the sake of clarity we use the prefix provenance for extensions to the OWL-2 grammar, which uses prefix *OWL*.

OWLAnnotation := **ProvenanceAxiomAnnotation**
ProvenanceAxiomAnnotation := 'ProvenanceAxiomAnnotation' '('**IRI** **ProvenanceAnnotation**$^+$')'
ProvenanceAnnotation := **ProvenanceCertaintyAnnotation** | **ProvenanceDateAnnotation** | **ProvenanceSourceAnnotation**
ProvenanceCertaintyAnnotation := 'ProvenanceCertaintyAnnotation' '('**Value**')'
ProvenanceSourceAnnotation := 'ProvenanceSourceAnnotation' '('**Value**')'
ProvenanceDateAnnotation := 'ProvenanceDateAnnotation' '('**Value**')'

An example of how provenance is represented and associated with OWL axioms is presented below. We consider the axioms #4 and #5 from our scenario:

```
OWLAxiomAnnotation(ClassAssertion(bluewater City)
 ProvenanceAxiomAnnotation(annot1 ProvenanceSourceAnnotation(Nev)))

OWLAxiomAnnotation(ObjectPropertyAssertion(hqIn vpktv bluewater)
 ProvenanceAxiomAnnotation(annot2 ProvenanceDateAnnotation(09.09.2009)))
```

4.4.2. Semantics of Provenance

Provenance assignments are syntactically expressed in OWL-2 using axiom annotations. Annotations, however, have no semantic meaning in OWL-2. All annotations are ignored by the reasoner, and they may not themselves be structured by further axioms.

Furthermore, such an abstract syntax may remain remarkably ambiguous if it cannot be linked to a formal semantics. Assume that the following provenance axioms are part of our scenario:

ID	axiom	trust
#1	$RealCity \sqsubseteq City \sqcap \exists hasCompany.Broadcaster$	∞
#1	$RealCity \sqsubseteq City \sqcap \exists hasCompany.Broadcaster$	Nev

Table 4.2.: Example of multiple provenance assignments to the same axiom.

[3]OWL 2 Web Ontology Language: Spec. and Func.-Style Syntax: http://www.w3.org/TR/2008/WD-owl2-syntax-20081202

For the same axiom identified by #1 presented in Table 4.4.2, the question may arise whether this means a disjunction, i.e. one of the two sources has provided the fact, or a conjunction, i.e. both sources have provided the fact, or a collective reading, i.e. the two sources together gave rise to the fact, or whether this situation constitutes invalid provenance. In order to prevent such ambiguities, we introduce a generic semantic framework for provenance.

Provenance Order

Provenance may be established through various complex dimensions such as knowledge source, editors, modification data, and degrees of trustworthiness. Provenance dimensions must be exploited and combined to arrive at an accurate assessment of information value [Halpin, 2009]. Provenance dimensions are defined in detail in the next section.

First we take a closer look at the two specific dimensions of provenance used in examples in this work: time and source. Provenance is expressed using *provenance labels*. An example for a provenance label is a timestamp or a source name.

The label alone is not sufficient for the tracking of provenance when debugging an ontology. We need a *provenance order* on the provenance labels. For example, suppose that an user has introduced an inconsistency during his last change. We might be interested in when the oldest axiom leading to an inconsistency has been added, or when the youngest has been added. The oldest axiom tells us when this particular topic was first addressed and the newest one tells us when the inconsistency was introduced. We use an ascending or descending order of timestamps.

Not all types of provenance labels have a natural order. There are, for example, many ways in which *degrees of trustworthiness* can be computed. Often simplifications are used, such as assuming trust to be measured on a scale from 1 to 10. Such simplifications usually are not easily justifiable [Halpin, 2009] when trust should be established using provenance labels such as the knowledge source.

In particular, trust (and provenance in general) can not always be measured on a total order, but there may be agents which are incomparable. Please note that even though we use trust in our running example, the same principle applies, e.g., when modeling access rights, roles in an editing workflow or comparing world views of users participating in an ontology editing platform. Access right systems are a good analogy, which usually introduce some kind of ordering of users and groups. This ordering always is artificial and usually also partial (not in every pair of users or groups one is more powerful than the other) [Baader et al., 2009].

We provide a generic formalization of provenance orders here, which subsumes others such as [Golbeck and Hendler, 2004] and enables us to use any kind of order over provenance labels in the following. The following formalization has been used in similar form in [Schenk, 2008].

Definition 4.4.1 (Provenance Order) *A provenance order $\mathcal{T}$ is a lattice over a finite set of* provenance labels*. ∞ is the label used for the maximal element of the lattice.*

If two provenance labels a and b are not comparable, we introduce virtual provenance label inf_{ab} and sup_{ab}, such that:

- $inf_{ab} < a < sup_{ab}$ and $inf_{ab} < b < sup_{ab}$;
- $\forall_{c<a,c<b} : c < inf_{ab}$ and

source	trust
DPA	∞
Neverest	Nev
NeverestWP	NevWP

Table 4.3.: Correspondences between source and degrees of trustworthiness.

- $\forall_{d>a,d>b} : d > sup_{ab}$

To understand the importance of the last two steps, assume that $c > a > d$ and $c > b > d$ and a, b are incomparable. Then the provenance label of $a \wedge b$ would be the provenance label of c, as c is the supremum in the provenance order. Obviously this escaping to a higher provenance label is not desirable. In our trust example, it would mean escaping to a higher trust level. Considering roles as a workflow, we might end up with the wrong role. Instead, the virtual provenance labels represent that we need to pick at least one, a or b. For convenience, we will write $\{a, b\}$ for $inf_{a,b}$ in the following.

Note that in practice, these values need not be pre-computed, so exponential blowup is not an issue. If they are needed, they can be encoded as lists of atomic labels.

Provenance orders subsume strict orders (such as [0..1], `xsd:datetime` and numeric trust degrees computed for Wikipedia [Adler and de Alfaro, 2007b]). A provenance order also allows for incomparable provenance labels, which are common on the Web due to its sheer size and usually incomplete knowledge.

We assign degrees of trustworthiness to axioms based on the expertise of the user by whom they have been modified. Hence, the *knowledge source* of a piece of information is used to establish *trust*. In our running example, we use the following correspondences between source and degrees of trustworthiness presented in Table 4.3.

Figure 4.1 illustrates a trust order for the sources shown in Table 4.1 from the perspective of DPA. ∞ is assigned to the most trustworthy source which is DPA and *Nev, NevWP* are incomparable.

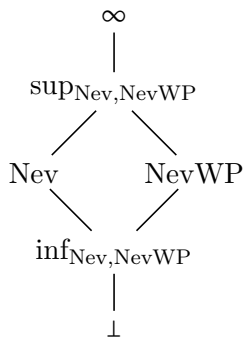


Figure 4.1.: Provenance order of the knowledge sources.

Provenance Dimension

Provenance can have atomic and complex dimensions such as knowledge source, modification date or a combination thereof. We assume that these (and possible further) dimensions are independent of each other. In the next section, we generalize from this assumption.

Definition 4.4.2 (Provenance Dimension) *A provenance dimension D is an algebraic structure $(B_D, \dot{\vee}_D, \dot{\wedge}_D)$, such that $(B_D, \dot{\vee}_D)$ and $(B_D, \dot{\wedge}_D)$ are complete semilattices. We denote the minimal element of D by $\bot_D$.*

B_D represents the labels the provenance can take, e.g. all valid timestamps for the modification date. As $(B_D, \dot{\vee}_D)$ and $(B_D, \dot{\wedge}_D)$ are *complete* semilattices, they are, in fact, also lattices. Hence, a finite set of provenance labels always has a join (supremum, least upper bound) and a meet (infimum, greatest lower bound) with respect to the corresponding order.

In contrast to provenance orders defined in Section 4.4.2 the join and meet operators in a provenance dimension need not be dual as they can come from two different lattices, which share the same values but have different orders. An example of a provenance dimension where this may become important is *where provenance* [Buneman et al., 2001]: which only tracks who contributed in any way to a certain inference. In this case, join and meet would coincide and would both be set unions of information sources. In all dimensions discussed in this chapter, join and meet are dual. In this case, provenance dimension and provenance order directly correlate.

Example 4.4.1 *To illustrate the meaning of $\dot{\wedge}$ and $\dot{\vee}$, let I be the provenance interpretation that is a partial function mapping axioms into the label range of trust, and A and B be axioms of an ontology such that $A \neq B$. When combining two provenance labels from D, which are assigned to A and B, the intuitive meaning of $\dot{\vee}$ is "I need to trust one of A and B" (corresponding to a logical "or"). The intuitive meaning of $\dot{\wedge}$ is "I need to trust both A and B" (corresponding to a logical "and")). Hence, trust can be modeled as:*

- $I_{trust}(A \dot{\vee} B) = sup(I_{trust}(A), I_{trust}(B))$
- $I_{trust}(A \dot{\wedge} B) = inf(I_{trust}(A), I_{trust}(B)$

Likewise, the modification date as well as the creation date can be modeled as:

- $I_{date}(A \dot{\vee} B) = min(I_{date}(A), I_{date}(B))$
- $I_{date}(A \dot{\wedge} B) = max(I_{date}(A), I_{date}(B))$

As we have seen in the Example 4.4.1, provenance is assigned to ontology axioms. Within a single assignment, the provenance must be uniquely defined.

Definition 4.4.3 (Provenance Assignment) *A provenance assignment M is a set $\{(D_1, d_1 \in D_1), ..., (D_n, d_n \in D_n)\}$ of pairs of a provenance dimension and a corresponding provenance label, such that $D_i = D_j \Rightarrow d_i = d_j$. As default label for a provenance assignment in dimension D_j we choose the minimal element $\bot_{D_j}$. By $ProvAss(A)$ we denote the set of provenance assignments for an axiom A.*

Example 4.4.2 *As an example, for the axiom* "bluewater: City" *of our running example, we have the following provenance assignment:*

$$ProvAss(bluewater: City) = \{(trust, Nev), (date, 2009\text{-}09\text{-}09)\}$$

Without loss of generality we assume a fixed number of provenance dimensions.

Next, we formalize how provenance assignments are composed. To obtain a logical formula which express how provenance assignments are composed, called *provenance formula*, we make use of pinpointing formulas (discussed in Section 4.3) and of the *how- provenance* strategy. *How provenance* [Green et al., 2007] is a strategy, which describes how an axiom A can be inferred from a set of axioms $\{A_1, ..., A_n\}$, i.e. it is a Boolean formula connecting the A_i. As pinpointing summarizes explanations for axioms in a single Boolean formula, and thus it provides *how-provenance*, we use it to come up with a provenance formula.

Example 4.4.3 *Consider the query "for each city, find all companies located in that city":*

$$x{:}City \wedge hasCompany(x,y).$$

The result of this query and the corresponding pinpointing formula are based on the example data from Table 4.1:

x	y	*pinpointing formula*
bluewater	*vpktv*	#3∧#4∧#6

The associated provenance formula for this query result is:

provenance formula
ProvAss*(#3)* $\dot{\wedge}$ ProvAss*(#4)* $\dot{\wedge}$ ProvAss*(#6)*

To evaluate the corresponding provenance formula, we need to define the operators for provenance dimensions.

Definition 4.4.4 (Provenance Operations) *Let A, B be axioms. Let $ProvAss(A) = \{(D_1, x_1), ..., (D_n, x_n)\}$ and $ProvAss(B) = \{(D_1, y_1), ..., (D_n, y_n)\}$ be provenance assignments. Then the provenance operations $\dot{\vee}$ and $\dot{\wedge}$ are defined as follows:*

$ProvAss(A)\dot{\vee}ProvAss(B) = \{(D, x\dot{\vee}_D y) | (D, x) \in ProvAss(A)$ *and* $(D, y) \in ProvAss(B)\}$.
$ProvAss(A)\dot{\wedge}ProvAss(B) = \{(D, x\dot{\wedge}_D y) | (D, x) \in ProvAss(A)$ *and* $(D, y) \in ProvAss(B)\}$.

Example 4.4.4 *To illustrate how provenance is derived, for instance, the provenance assignment for the axiom* "bluewater: City" *is* {(trust, Nev), (date, 2009-09-09)}*, and the provenance assignment for the axiom* "hqIn (vpktv,bluewater)" *it is* {(trust, NeverestWikiP), 2009-09-09)}*. We assume the provenance order described in Table 4.3. The provenance for* bluewater: City ∨ hqIn(vpktv, bluewater) *is determined as follows:*

$$ProvAss(bluewater : City)\dot{\vee}ProvAss(hqIn(vpktv, bluewater)) = \{(trust, Nev), (date, 2009\text{-}09\text{-}09)\}\dot{\vee}\{(trust, NevWP), (date, 2009\text{-}09\text{-}09)\}.$$

Note that, due to the defaults introduced in Definition 4.4.3, the operations on provenance assignments are defined even in the presence of incomplete provenance from a domain. In our framework, just as $\dot{\vee}$ and $\dot{\wedge}$ correspond to a logical "or" and "and", default corresponds to a default truth value of "unknown" in a default logic. With a default assignment, we provide a uniform treatment of axioms in case of absence of provenance, and thus we are able to combine arbitrarily provenance assignments. In any case, as illustrated in Example 4.4.1, the interpretation of provenance and defaults assignments as well as the interpretation of the provenance operations are domain application dependent. Such considerations are orthogonal to our framework.

While axioms in the underlying description logic may contain negation, these negations are not visible and not needed at the level of the provenance. Hence, the provenance algebra does not need to contain a negation operator.

Finally, we define how to retrieve the provenance assigned to an axiom A within a provenance dimension. The provenance of an axiom A within a provenance dimension is obtained by evaluating the corresponding provenance formula in the dimension under consideration.

Definition 4.4.5 (Provenance Evaluation) *Let prov(A) be a function mapping from an axiom A to a provenance assignment in dimension D. The provenance of an axiom A wrt. O in D is obtained by computing a pinpointing formula ϕ of A wrt. O and obtaining ψ by replacing each axiom in ϕ with its provenance assignment in D and the logical operators $\dot{\vee}$ and $\dot{\wedge}$ with their corresponding operators in D. Then prov(A) is computed by evaluating ψ.*

Example 4.4.5 *In our running example, for the query "for each city find all companies",* x:City $\wedge$ hasCompany(x,y)*, we have the result* bluewater, vpktv *and the following provenance formula:*

$$ProvAss(\#3)\dot{\wedge}ProvAss(\#4)\dot{\wedge}ProvAss(\#6)$$

The corresponding provenance evaluation for the dimension trust is:

$$(trust, \infty)\dot{\wedge}(trust, Nev)\dot{\wedge}(trust, NevWP) = (trust, \infty\dot{\wedge}Nev\dot{\wedge}NevWP) = (trust, inf_{Nev,NevWP}).$$

Complex Provenance Dimensions

In the previous section we have described how provenance can be computed in a single dimension. This focus on a single dimension is useful if independence of dimensions can be assumed. Sometimes, however, this is not the case. For example, when a group of users collaboratively edits an ontology, the time an axiom was asserted and the user responsible for the modification will often correlate. In this case, two knowledge dimensions can be composed into a complex provenance dimension:

Definition 4.4.6 (Composition of Dimensions) *Let* $D_1 = (B_{D_1}, \dot{\vee}_{D_1}, \dot{\wedge}_{D_1})$ *and* $D_2 = (B_{D_2}, \dot{\vee}_{D_2}, \dot{\wedge}_{D_2})$ *be provenance dimensions. Then* $D = (B_D, \dot{\vee}_D, \dot{\wedge}_D)$ *is a composed provenance dimension, such that*

- $B_D = B_{D_1} \times B_{D_2}$,
- $(B_D, \dot{\vee}_D) = \{((x,y)(v,w)) | x, v \in B_{D_1} \text{ and } y, w \in B_{D_2} \text{ and } x \leq_{\vee_{D_1}} v \text{ and } y \leq_{\vee_{D_2}} w\}$, *and*
- $(B_D, \dot{\wedge}_D) = \{((x,y),(v,w)) | x, v \in B_{D_1} \text{ and } y, w \in B_{D_2} \text{ and } x \leq_{\wedge_{D_1}} v \text{ and } y \leq_{\wedge_{D_2}} w\}$

We will refer to the elements of B_D *as* complex provenance labels *and to the elements of* B_{D_1} *and* B_{D_2} *as* atomic provenance labels.

Example 4.4.6 *As an example of composition of dimensions, we compose the dimensions trust and modification date and compute the provenance for* x:City ∧ hqIn(y,x). *We have the provenance formula:*

$$ProvAss(\#4) \dot{\wedge} ProvAss(\#6)$$

If we treat the trust and modification date dimensions separately, the results are

$$\{(trust, Nev))\} \dot{\wedge} \{(trust, NevWP)\} = \{(trust, inf_{Nev,NevWP})\}$$

$$\{(date, 2009\text{-}09\text{-}09)\} \dot{\wedge} \{(date, 2009\text{-}09\text{-}10)\} = \{(date, 2009\text{-}09\text{-}10)\}$$

If we combine them into one dimension as shown in Figure 4.2, however, the result is

$$\{(trust, Nev), (date, 2009\text{-}09\text{-}09)\} \dot{\wedge} \{(trust, NevWP), (date, 2009\text{-}09\text{-}10)\} = \{(trust, inf_{Nev,NevWP}), (date, 2009\text{-}09\text{-}10)\}$$

Having composed the interdependent dimensions into one, one may use the composed provenance dimension just as an atomic one. Definition 4.4.5 applies exactly as for simple provenance dimensions.

Semantics for Conflicting Provenance

In the following section we extend our model to support conflicting provenance which can arise from conflicting changes or provenance assignments by multiple users at different times. A new operator is needed for this merging, as the merge operator need not coincide with one of $\dot{\vee}$ and $\dot{\wedge}$.

Definition 4.4.7 (Conflict Tolerant Dimension) *A conflict tolerant provenance dimension* D *is an algebraic structure* $(B_D, \dot{\vee}_D, \dot{\wedge}_D, \oplus_D)$, *such that* $(B_D, \dot{\vee}_D)$, $(B_D, \dot{\wedge}_D)$ *and* $(B_D, \oplus_D)$ *are complete semilattices. The minimum of* $(B_D, \oplus_D)$ *is* $\perp_D$.

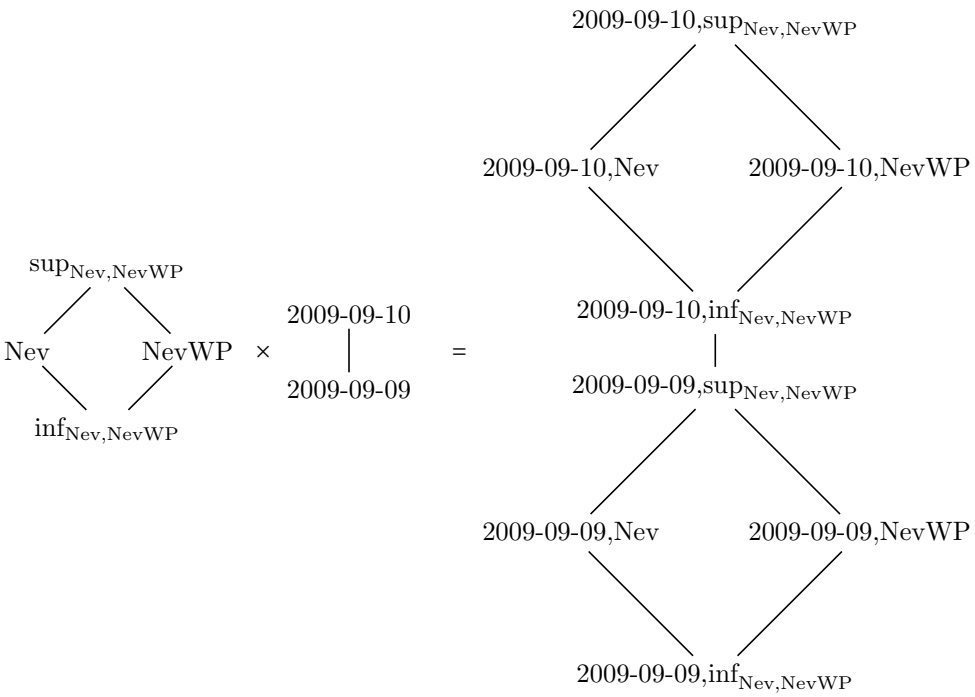


Figure 4.2.: Provenance order of the knowledge sources.

Example 4.4.7 *To illustrate how to support conflicting provenance by multiple provenance assignments, let I be a provenance interpretation and let A be an axiom of an ontology, for which there exist multiple provenance assignments. We use A' and A'' to represent the axiom A with different provenance assignments. Let us compare the already known provenance dimension* modification date *with the* creation date*. For* modification date*, $\oplus$ needs to be the* maximum *(we are interested in the last assertion), while for* creation date*, $\oplus$ needs to be the* minimum *(we are interested in the first assertion). In contrast to example 4.4.1 the operators for creation date and modification date do not coincide. The* modification date *could be modeled as:*

$$I_{date}(A' \oplus A'') = max(I_{date}(A'), I_{date}(A''))$$

The creation date *could be modeled as:*

$$I_{cre}(A' \oplus A'') = min(I_{cre}(A'), I_{cre}(A''))$$

Likewise, the trust *could be modeled as:*

$$I_{trust}(A' \oplus A'') = sup(I_{trust}(A'), I_{trust}(A''))$$

To show how the support to conflicting provenance by multiple provenance assignments can be applied to our scenario, we slightly extend our running example in Example 4.4.8:

Example 4.4.8 *We assume that axiom (#1), $RealCity \sqsubseteq City \sqcap \exists hasCompany.Broadcaster$, has been modified by two sources at different times:*

ID	*trust*	*date*
#1	∞	*2009-09-10*
#1	*Nev*	*2009-09-09*

Then the provenance assignment for axiom $\#_1$ *is*

$$\{(trust, \infty), (date, 2009\text{-}09\text{-}10)\} \oplus \{(trust, Nev), (date, 2009\text{-}09\text{-}09)\} = \{(trust, \infty \oplus Nev), (date, 2009\text{-}09\text{-}09 \oplus 2009\text{-}09\text{-}10)\} = \{(trust, \infty), (date, 2009\text{-}09\text{-}10)\}.$$

In contrast, if first Neverest and then DPA assert the same axiom, we would have

$$\{(trust, \infty), (date, 2009\text{-}09\text{-}09)\} \oplus \{(trust, Nev), (date, 2009\text{-}09\text{-}10)\}$$

As $\infty >_{trust} Nev$*, but* $2009\text{-}09\text{-}09 <_{date} 2009\text{-}09\text{-}10$*, these labels are incomparable and can not be further simplified.*

In order to accommodate such potentially conflicting provenance assignments about ontology axioms, we extend the semantics of provenance, which we have introduced in Section 4.4.2. For this purpose, we redefine the *prov* function of Definition 4.4.5, such that it uses $\oplus$ to merge provenance assignments in a preprocessing step. Afterwards, we have a unique provenance assignment again and apply Definition 4.4.5 as before.

Definition 4.4.8 (Provenance Extended) *Let allprov: axioms* $\rightarrow 2^{B_D}$ *be a function mapping from an axiom to all provenance assignments to that axiom in a provenance dimension* D*.*

Let prov *be a function mapping from an axiom to a provenance assignment in dimension* D*. The provenance of an axiom* A *wrt.* O *in* D *is obtained by computing a pinpointing formula* ϕ *of* A *wrt.* O *and obtaining* ψ *by replacing each axiom* A *in* ϕ *with* $\bigoplus(allprov(A))$ *and the logical operators* $\dot{\vee}$ *and* $\dot{\wedge}$ *with their corresponding operators in* D*. Then* prov*(*A*) is computed by evaluating* ψ*.*

Example 4.4.9 *Consider the provenance assignment of axiom* #1 *with conflicting provenance presented below and the provenance formula:*

$$ProvAss(\#1) \dot{\wedge} ProvAss(\#2)$$

the derived provenance is:

$$\{(trust, \infty), (date, 2009\text{-}09\text{-}10)\} \oplus \{(trust, Nev), (date, 2009\text{-}09\text{-}09)\} \dot{\wedge} \{(trust, \infty), (date, 2009\text{-}09\text{-}09)\} = \{(trust, \infty \oplus Nev) \dot{\wedge} (trust, \infty), (date, 2009\text{-}09\text{-}10 \oplus 2009\text{-}09\text{-}10) \dot{\wedge} (date, 2009\text{-}09\text{-}09)\} = \{(trust, \infty \dot{\wedge} \infty), (date, 2009\text{-}09\text{-}10 \dot{\wedge} 2009\text{-}09\text{-}10)\} = \{(trust, \infty), (date, 2009\text{-}09\text{-}10)\}.$$

This definition of *prov* not only allows to aggregate provenance from multiple sources, but also to gracefully handle unknown provenance, i.e. situations where a knowledge source does

not provide a label for some provenance dimension, in which case $\bot_D$ is assumed as a default, as introduced in Definition 4.4.3

Example 4.4.10 shows how our approach can be applied to our use case scenario.

Example 4.4.10 *Back to our scenario, DPA needed to verify that Bluewater indeed is a city in CA. Hence, they need to answer the query "bluewater: RealCity?" and to obtain provenance for the answer. The query results in the following provenance formula:*

$$ProvAss(\#3) \dot{\wedge} ProvAss(\#4) \dot{\wedge} ProvAss(\#6)$$

The resulting provenance label is:

$$\{(trust, \infty), (date, 2009\text{-}09\text{-}10)\} \dot{\wedge} \{(trust, Nev), (date, 2009\text{-}09\text{-}09)\} \dot{\wedge} \\ (trust, NevWP), (date, 2009\text{-}09\text{-}09)\} = \{(trust, inf_{Nev,NevWP}), (date, 2009\text{-}09\text{-}10)\}.$$

Note that in Example 4.4.10 the labels of the modification date dimension are comparable, while the labels of the trust dimension are not. Hence, the resulting provenance label is a tuple of the infimum of Nev and NevWP in the trust dimension and the maximum of the dates in the assertion date component.

4.5. Using Provenance to Debug Changing Ontologies

We now use our definitions of provenance in order to track down what recent addition to the knowledge base led to a desired or undesired effect. In the case of our collaborative ontology editing scenario, we may want to identify who was the least authoritative source that contributed most recently to an inconsistency.

The algorithms discussed in this section fulfill this purpose, i.e. given a query (an axiom), an ontology and a provenance dimension, they extract a subontology as explanation and compute the provenance label of the query.

As we can see above, Definition 4.4.5 relies on a pinpointing formula for the computation of provenance. Hence, we need to find all pinpoints to compute a pinpointing formula. Then we can immediately derive the provenance label.

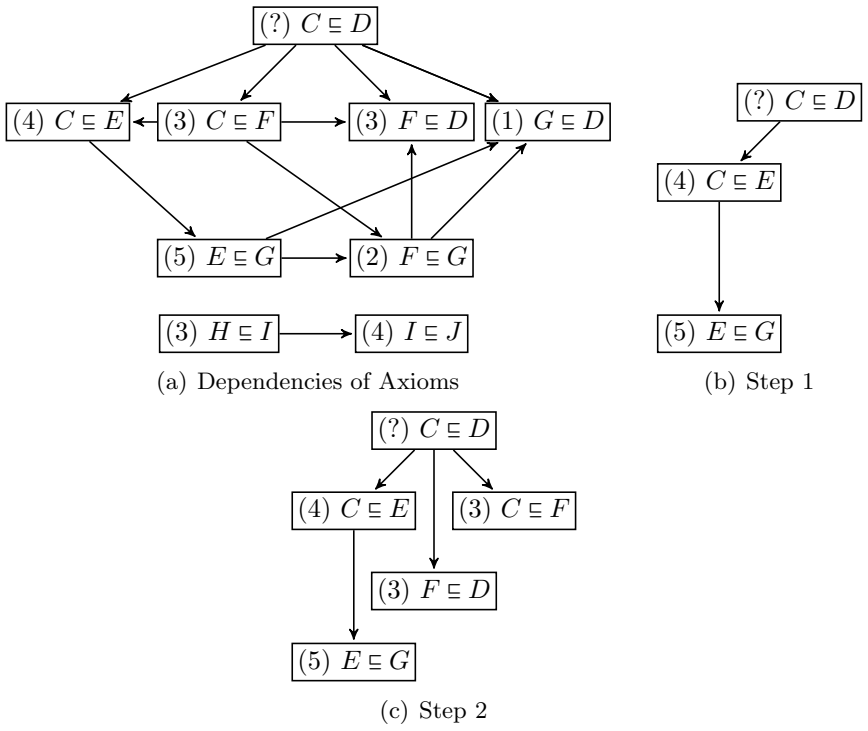
4.5.1. Naïve Approach

A naïve evaluation of provenance for an axiom might compute all pinpoints and then evaluate the pinpointing formula. This strategy is illustrated in Algorithm `ProvNaive`. `ProvNaive` takes as parameters an ontology O, an axiom A and a provenance dimension D. It returns the provenance label of A wrt. O and a subontology containing all pinpoints for $O \models A$.

However, finding all pinpoints is a very expensive operation and this approach is not appropriate in real-use cases. Finding a pinpoint using a black-box approach may need an exponential number of consistency checks in the underlying DL. Moreover, there can be exponentially many pinpoints.

Algorithm 2 Evaluation Algorithm: ProvNaive(O, A, D)

1: pinpoints ← GetAllPinpoints(A, O)
2: labels ← $\{l | \exists p \in \text{pinpoints}: l = \dot{\bigwedge}_D prov(p)\}$
3: labels ← $\{\dot{\bigvee}_D \text{ lab} | \text{ lab} \in \text{labels}\}$
4: M ← $\bigcup_{p \in \text{pinpoints}} p$
5: **return** M, lab



(a) Dependencies of Axioms (b) Step 1

(c) Step 2

Figure 4.3.: Selection of Axioms by Optimized Algorithm.

4.5.2. Debugging with Provenance - An Optimized Algorithm

We present an optimized algorithm for deriving provenance which performs significantly better in the average case than naïve implementations. It does not need to compute all pinpoints, nor even one precise pinpoint. Instead, we compute an approximation which is sufficient for deriving provenance.

The optimization is based on the assumption that the provenance dimension is a lattice $(B_D, \dot{\vee}_D, \dot{\wedge}_D)$ such that $a \dot{\vee}_D b = \sup(a, b)$ and $a \dot{\wedge}_D b = \inf(a, b)$ for $a, b \in B_D$ (or vice versa). Note that in the general case the interpretation of $\dot{\vee}_D$ and $\dot{\wedge}_D$ is independent as shown in Definition 4.4.2, i.e., they do not need to be dual. However, for optimization issues we assume that $\dot{\vee}_D$ and $\dot{\wedge}_D$ are no longer independent. This assumption is true for all provenance dimensions discussed above, such as modification date, degree of trustworthiness, and for all total orders. In this case, the pinpointing formula has the structure of a supremum of infima when expressed in disjunctive normal form.

Considering this assumption, we make the evaluation more efficient since we can exploit

monotonicity properties of our provenance dimensions where applicable.

For instance, for a provenance dimension with a total order, once we find the pinpoint with the highest provenance label, we have also found the overall maximal provenance label. Thus, in many cases, we do not need to compute all pinpoints and we may restrict the computation of the pinpointing formula to those parts of an ontology that are relevant for the provenance computation given its particular lattice structure.

If the provenance dimension is a partial order, several pinpoints with incomparable provenance labels may be found. Thus, we need to find all pinpoints with maximal (or minimal) provenance labels and merge the corresponding provenance labels to determine the resulting provenance label.

Without loss of generality, we only consider for our optimized algorithm the case that the corresponding pinpointing formulas have the structure of a supremum of infima. In this case the provenance label of a single pinpoint is the infimum of the labels of the axioms it contains, and the overall provenance label is the supremum of the labels of all such pinpoints. As a result, we do not need to take into account any axiom, which has a provenance label less than the infimum of the labels of the greatest pinpoints. For the case of a infimum of suprema, we simply need to invert all comparisons and replace min by max in Listings 3 and 4.

Likewise, we only consider consistent ontologies. The proposed algorithms handle inconsistent ontologies equally well, if the entailment check in line 10 of Listing 3 is negated.

The algorithm starts with a query of the form "$O \vDash A$?", as we focus on entailment checking here. In order to find those axioms that are relevant for the provenance computation given the query, the algorithm iteratively grows a subontology around A based on the *syntactic relevance* selection function [Ji et al., 2009] (see Definition 4.5.1). Intuitively, an axiom is syntactically relevant for another axiom, if it contributes to the definition of one of the concepts or properties in the other axiom. In Figure 4.3, a small example ontology is shown. The arrows represent syntactic relevance relationships between axioms. Assume the query is $C \sqsubseteq D$?. Then the two axioms at the bottom can not be relevant to the answer, because they are neither directly nor indirectly relevant to the query. The rest of the ontology contains three justifications for $C \sqsubseteq D$, namely $\{C \sqsubseteq E, E \sqsubseteq G, G \sqsubseteq D\}, \{C \sqsubseteq F, F \sqsubseteq G, G \sqsubseteq D\}$ and $\{C \sqsubseteq F, F \sqsubseteq D\}$

Definition 4.5.1 (Syntactic Relevance [Ji et al., 2009]) *An axiom B is directly syntactically relevant for an axiom A, if their clusters overlap, i.e. if they share a concept, role or individual. B is syntactically relevant for A, if it is directly syntactically relevant for A, or if B is syntactically relevant for C, which is syntactically relevant for A.*

We define a convenience function σ: Given A and an ontology O, $\sigma(A, O) = \{B \in O | B$ is directly syntactically relevant for $A\}$. The definition carries over to sets of axioms M: $\sigma(M, O) = \{B \in O | \exists A \in M : B$ is directly syntactically relevant for $A\}$

Using the syntactic relevance selection function, the algorithm `Prov(O, A, D)` computes the provenance label of $O \vDash A$ in dimension D, if $O \vDash A$. We start by determining the set of syntactically relevant axioms for $A = C \sqsubseteq D$ in line 3 ($C \sqsubseteq E$, $C \sqsubseteq F$, $F \sqsubseteq D$ and $G \sqsubseteq D$). Then we add the ones with the greatest provenance label to the module in line 4 ($C \sqsubseteq E$ in step1; $C \sqsubseteq F$ in step 2). In the inner loop we recursively add all syntactically relevant axioms for the new module which do not decrease the provenance degree of the overall solution ($E \sqsubseteq G$ in step 2). Note that this is just an optimization step to avoid unnecessary entailment checks. It can be omitted to compute a more precise approximation of the pinpoint. The trade-off is a possibly higher number of iterations and entailment checks.

In each iteration we add axioms until we have found a module which contains a pinpoint in line 10 (this is the case after step 2 of the example). The provenance label for A is the smallest provenance degree of the axioms in the module (hence, 3). We work with a set for *labels* to account for the fact that in the presence of partial orders, minimum and maximum are not unique. Hence, the *min* function used in line 11 is the set version, which returns the set of smallest elements of a set. For total orders, *label* always is a singleton.

Note that our optimization strategy is strongly related to two issues: (a) the syntactic relationships among the pinpointing axioms, and (b) the size of pinpoints. The closer the pinpointing axioms are syntactically correlated and the smaller the pinpoints are, the faster the module can be built and the provenance label be derived. By growing only a small module of the ontology and avoiding the use of instantiated ontology as a whole for approximating pinpoints, we save lots of entailment checks which are very expensive.

Algorithm 3 Evaluation Algorithm: Prov(O, A, D)

```
1:  M ← {A}
2:  repeat
3:      syn ← σ(M, O)
4:      add ← {B1 ∈ syn|∄B2 ∈ syn : prov(B1) <D prov(B2)}
5:      repeat
6:          M ← M ∪ add
7:          syn ← σ(M, O)
8:          add ← {B1 ∈ syn|∃B2 ∈ M : prov(B1) ≥D prov(B2)}
9:      until add ⊆ M
10: until M − A ⊨ A;
11: labels := minD({prov(B)|B ∈ M})
12: return M, labels
```

Theorem 4.5.1 *Let O be an ontology, A an axiom such that $O \models A$ and D a provenance dimension, which is a total order.* `Prov(O, A, D)` *computes the provenance degree of $O \models A$ in dimension D.*

Proof: First we show that `Prov` terminates. The inner loop terminates when *M-A* $\models A$. At each iteration *add* is assigned to a subset of O, which is syntactically relevant to *module* and has the greatest provenance degree. Consequently, *module* is a subset of O. As O is finite and $O \models A$, the loop must eventually terminate. In this case *M-A* indeed $\models A$ and *labels* = $min_D(\{prov(B)|B \in M\})$.

It remains to show that if `Prov` terminates, it returns the correct provenance degree of $O \models A$.

Remember that we are looking for the pinpoint with the highest provenance label and that the provenance label of the pinpoint is the infimum of the provenance labels of its axioms.

M contains only axioms which are syntactically relevant for A. When the outer loop terminates, M contains a (superset of a) greatest pinpoint of A wrt. O. Assume there is a pinpoint P, which has not been found and has a greatest provenance label. Then it must already be part of *module*: All axioms in a pinpoint of A must be syntactically relevant for A. Moreover, $\forall B \in P : \text{prov}(B) >_D labels$. M contains all such axioms which are directly syntactically relevant for A, or which are indirectly syntactically relevant via axioms with provenance labels $\geq_D$ *labels*. Hence, M must already contain P. Refutation.

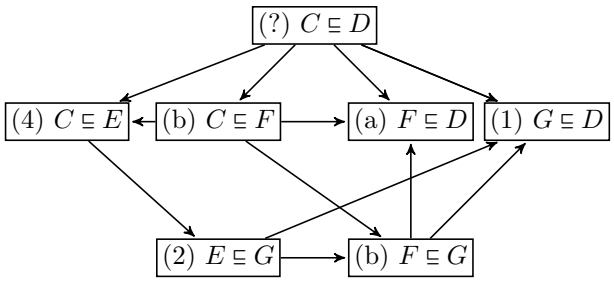


Figure 4.4.: Justifications with incomparable provenance.

For partial orders the resulting *label* may contain multiple elements which need to be merged. Figure 4.4 illustrates a case where we have a complex dimension composed of numbers and letters and their natural orders and multiple relevant pinpoints which together result in a provenance label of $\inf_{1,a}$. `ProvPartial` computes accurate labels for partial orders. It uses `Prov` to compute an approximation first. Although there might be multiple pinpoints with incomparable provenance labels, `Prov` stops after the first pinpoint is contained in the approximation. Therefore, we need to further extend the approximation, such that all possibly relevant axioms are included. If the label returned by `Prov` is a singleton, there is a unique maximal pinpoint (lines 2 and 3). Otherwise, we compute the minimum of all elements of *lab* (line 4), which is the lower bound D we need to consider. For example, as numbers and letters are incomparable, the lower bound for a and 2 is $\inf_{1,a}$. This means we could ignore any axiom which does not have a provenance label (and hence defaults to $\bot$), but need to consider $G \sqsubseteq D$. Every pinpoint with a provenance label equal to or less than *min* is also less than the label of the pinpoint we have already found in `Prov`, and hence we can ignore it. We now extend our approximation with all axioms which are syntactically relevant to A and, if they are indirectly relevant, are connected to A only through axioms which have a provenance label greater than *min*. The loop in lines 5 to 9 works analogously to lines 5 to 9 in `Prov`. Finally, we compute the correct label using `ProvNaive`.

Algorithm 4 Evaluation Algorithm: ProvPartial(O, A, D)

1: $(M, labels) \leftarrow Prov(O, A, D)$
2: **if** $(|labels| = 1)$ **then return** $(M, labels)$
3: **end if**
4: $min \leftarrow \dot{\bigwedge}_{lab \in labels} lab$;
5: **repeat**
6: $N := M$
7: $syn := \sigma(M, O)$
8: $M := M \cup \{B \in syn | prov(B) >_D min\}$
9: **until** $M = N$
10: **return** ProvNaive(M, A, D)

Theorem 4.5.2 *Let O be an ontology, A an axiom such that $O \models A$ and D a provenance dimension, which is a partial order.* `ProvPartial(O, A, D)` *computes the provenance degree of $O \models A$ in dimension D.*

In line 9 we use the naïve algorithm. Therefore we only need to show two things: (a) If `Prov` returns a singleton, it is indeed correct also for partial orders and (b) lines 4 to 9 indeed compute a module, which contains all axioms relevant for computing the correct solution using the naïve algorithm.

(a) In line 3 and 4 of `Prov` exactly those axioms are added to M, which are minimal in the current syntactic relevance set `syn`. Hence, if incomparable provenance degrees occur, all of them are selected. They are also preserved in line 11, as there is no unique minimum in this case. Thus, if a unique minimum is returned, then it must in fact be equal or greater than the provenance labels of all other pinpoints.

(b) Assume there is an axiom B, which is contained in some pinpoint P, which is relevant for `labels` and missing in M. We consider two cases:

(b1) $\dot{\bigwedge}_{p \in P} \text{prov}(p) \leq$ `min`. That means some prov(p) is less or equal `min`. Then P is not relevant. Refutation.

(b2) Hence, *prov(p)* must be greater than `min` for all $p \in P$. Then all p and hence also B have been selected in lines 5 to 9. Refutation.

Figure 4.7 presents a diagram showing the relations between the set of axioms of the ontology (set O), the set of axioms of the relevant pinpoints (set P), and the set of axioms of the module retrieved by our algorithm (set R). Note that the algorithm retrieves some of the pinpoint axioms. Axioms that are part of the pinpointing formula and are retrieved by the algorithm (the overlap of the sets P and R) correspond to those that are relevant and necessary for deriving provenance.

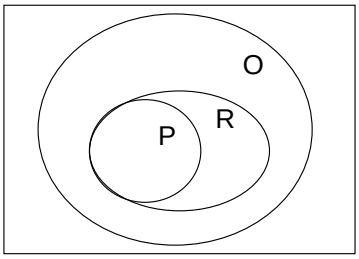


Figure 4.5.: Algorithm with optimization

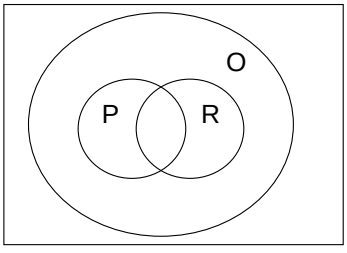


Figure 4.6.: Algorithm without optimization

Figure 4.7.: Relations between the set of axioms of the ontology (O), the axioms in all pinpoints (P) and the axioms retrieved by the algorithm (R).

4.6. Complexity

The worst case complexity of both the naïve and the optimized approach for computing provenance is equivalent to the computation of all pinpoints in the underlying logic as the pinpointing formula can be evaluated in polynomial time. If it is expressed in normal form, however, the size of the formula can blow up exponentially.

Approaches for computing pinpoints like [Baader and Peñaloza, 2010] which, rather than representing pinpoints formula in a normal form, derive a compact representation of the pinpoints formula, benefit the computation of provenance since they avoid exponential blow-up.

While in the worst case the complexity of computing provenance is the same as the complexity of finding all pinpoints and hence quite high, in the average case we can do much better using the optimized algorithm as shown in Section 4.7.

4.7. Evaluation

The prototype of the optimized provenance computation algorithm (`Prov`) has been implemented in Java 1.6 using Pellet 2.0.0[4] and OWL API trunk revision 1310[5].

4.7.1. Data

We have performed our experiments using two groups of ontologies. The first experiment is composed of standard ontologies for debugging ontologies that have already been used for testing the computing time of laconic justifications in [Horridge et al., 2008]. These ontologies have been used in [Horridge et al., 2008] to demonstrate the efficiency of traditional computation of pinpoints. The main goal of this experiment is to show that our optimized algorithm speeds up the performance of provenance computations compared to the standard computation of pinpoints.

The second experiment uses real-world living ontologies from Bioportals containing real change logs with timestamps and editors. The main goal of this experiment is to test the applicability and scalability of our approach for computing provenance in real time. For both experiments we use a virtual machine with 2GB RAM and a single Intel(R) Xeon(TM) CPU core with 3.60GHz.

4.7.2. Methodology

1. We compare our optimized algorithm to the naïve approach (baseline) based on full axiom pinpointing using the ontologies shown in Table 4.4. The ontologies used already have been taken in [Horridge et al., 2008] to demonstrate the efficiency of traditional computation of pinpoints, and offer a range of varying complexities and pinpoint sizes.

2. We evaluate if our approach is fast enough to support user assessments of the reliability of inferences in real time also for real-world, large-scale data. For this experiment, we have used the two real-world ontologies shown in Table 4.6.

[4]Pellet: OWL 2 Reasoner for Java (`http://clarkparsia.com/pellet`)
[5]The OWL API (http://owlapi.sourceforge.net/)

	Ontology	Expressiveness	Total Axioms	No.Unsat. Classes	Av. of Pinpoints per Unsat. Class
1,7	People	$\mathcal{ALCHOIN}$	372	1	1
2,8	MiniTambis	$\mathcal{ALCN}$	400	30	1
3,9	University	$\mathcal{SOIN}$	92	9	1
4,10	Economy	$\mathcal{ALCH}(\mathcal{S})$	2.330	51	1
5,11	Chemical	$\mathcal{ALCHF}$	192	37	11
6,12	Transport	$\mathcal{ALCH}$	2.178	62	2

Table 4.4.: Ontologies used in the experiments.

3. We compute the precision and recall of our approximation using all ontologies described above. Precision is defined here as the number of retrieved axioms that are relevant for deriving provenance divided by the total number of retrieved axioms. Recall is defined as the number of retrieved axioms that are relevant for deriving provenance divided by all relevant axioms in the ontology.

4.7.3. Discussion and Results

Experiment I - Debugging Ontologies

In this experiment, we compare our optimized algorithm to the naïve approach (baseline) based on full axiom pinpointing. For this experiment, we have used the ontologies from Table 4.4.

Since for these ontologies no provenance information is available, we have generated artificial provenance in two ways:

random - ontologies 1-6 We have augmented the original ontologies by randomly assigning timestamp values and degrees of trustworthiness to the axioms.

cluster - ontologies 7-12 We have augmented the original ontologies by assigning similar timestamp values and degrees of trustworthiness to such clusters of axioms which are syntactically relevant to each other. This reflects the fact that a user usually does not do random modifications, but changes a part of the ontology focused around a certain class or property.

As the ontologies 1 to 12 are inconsistent, our implementation focuses on pinpoints for inconsistencies instead of subsumptions. For this reason, the termination condition in line 8 of `Prov` has been changed to a consistency check, i.e. $M - A \models \neg A$.

For each ontology, we have measured the time needed to compute the provenance for an inconsistency in average over all inconsistent classes of an ontology. In order to investigate the influence of composing dimensions, we have performed the experiment for each ontology with a single and with two provenance dimensions (see Section 4.4.2).

We use as a baseline the naïve approach based on full axiom pinpointing the black-box algorithm implementation in the OWL API for computing all pinpoints.

The results of the evaluation are presented in Figure 4.8 and Table 4.5. In Figure 4.8, we have normalized the results to the processing time of the naïve approach and used a logarithmic

	Ontology	Base line	Av.Size Pinpoints Uns.Class	*Date* Dimension	Av.Size of Module Uns.Class	*Date* and *Trust* Dimensions	Av.Size Module Uns.Class
1	People R.	312	4	**141**	15	141	15
2	MiniTambis R.	141	7	**18**	22	22	22
3	University R.	53	4	**8**	18	8	18
4	Economy R.	395	3	**23**	57	27	58
5	Chemical R.	3239	7	**32**	126	38	126
6	Transport R.	1165	5	**23**	70	29	71
7	People C.	141	4	**16**	15	16	15
8	MiniTambis C.	127	7	**11**	22	11	22
9	University C.	52	4	**7**	18	7	18
10	Economy C.	386	3	**4**	19	5	19
11	Chemical C.	3236	7	**32**	126	37	126
12	Transport C.	1121	5	**28**	99	34	102

Table 4.5.: Average time to compute provenance for an inconsistency of an ontology (ms).

scale. The absolute numbers ranged over three orders of magnitude. Our optimized algorithm performs significantly better for all cases and scales very well. Note that in some ontologies we can find some very large pinpoints and some entailments with high numbers of pinpoints. In contrast, some other ontologies contain relatively small and few pinpoints.

As shown in Figure 4.8 ontology 5, which is an annotated version of the Chemical ontology, required the most computation time in the naïve approach as it contains some very large pinpoints and some entailments with high numbers of pinpoints. Especially for such cases, our approach has shown its potential.

As already expected, the optimized cluster ontology group (ontologies 7-12) has performed better than the optimized random ontology group (ontologies 1-6). The reason is that for the optimized cluster group we have assigned similar provenance labels to syntactically related axioms. Since in our approach, axioms that are syntactically relevant to the query are selected first, for the optimized cluster group the module could be built faster than for the random group. This increases the performance for computing provenance since axioms belonging to a common pinpoint are in general syntactically related to each other. As instance, ontology 10 has shown better performance than ontology 4 since the relevant axioms could be selected first, and thus a smaller module could be built.

Furthermore, we observe that debugging with many independent dimensions can have a negative performance impact due to lots of incomparable labels in the composed dimension, but in most cases still performs significantly better repeating the computation for two atomic dimensions.

Experiment II - Living Ontologies

In our second experiment, we evaluate if our approach is fast enough to support user assessments of the reliability of inferences in real time also for real-world, large-scale data. For this experiment, we have used the two real-world ontologies shown in Table 4.6.

The BIO ontology is a mapping ontology for the Open Biomedical Ontologies (BIO) repos-

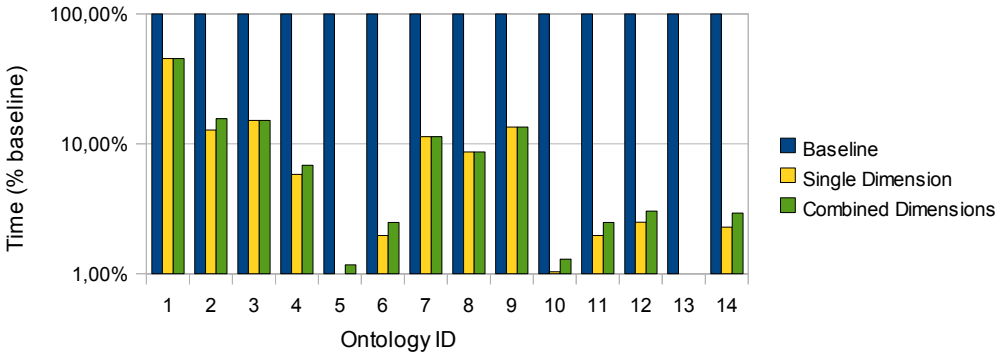


Figure 4.8.: Average time to compute provenance for an inconsistency of an ontology (in logarithmic scale). The computation of provenance for ontology 5, 10 and 13 scales orders of magnitude better than a naïve implementation algorithm and for that reason, they are not displayed on the graph.

	Ontology	Expressivity	Total Axioms	No. Unsat. Classes	Av. of Pinpoints per Unsat. Class
13	BIO	$\mathcal{SHOIN}$(D)	660.915	1	1
14	OBI	$\mathcal{SHOIN}$(D)	16.415	1	1

Table 4.6.: Ontologies used in the experiments.

itory of the US National Cancer Institutes Center for Bioinformatics [Smith et al., 2007], ontology 13. The BIO ontology comes with real change logs containing timestamps and editor information for each axiom, which we use to evaluate our approach.

The OBI ontology is the Ontology for Biomedical Investigations[6], ontology 14. This is an integrated ontology for the description of life-science and clinical investigations. It supports the consistent annotation of biomedical investigations, regardless of the particular field of study. The OBI ontology has been developed in collaboration with groups representing different biological and technological domains involved in Biomedical Investigations. In the OBI ontology, changes are tracked in change logs with timestamps and provenance information.

For both ontologies, we mapped information about editors to degrees of trustworthiness as defined in Section 4.4.2, and extracted the timestamps for the modification dates.

We compared our approach to the naïve approach for computing all pinpoints. For each ontology, we have evaluated the time needed to compute the provenance for an inconsistency in average over all inconsistent classes of an ontology.

Since ontology 13 and 14 are consistent, we have generated random queries as follows: For randomly selected disjoint classes A and B we introduced a new class C such that $A \sqsubseteq C$ and $B \sqsubseteq C$ and used these new axioms as queries. For this reason, we could use exactly the same implementation as for Experiment I. We have performed the experiment for each ontology with a single and with two dimensions to investigate the influence of composing dimensions.

[6]OBI: The Ontology for Biomedical Investigations (http://obi-ontology.org/)

	Ontology	Base line	Av.Size Pinpoints Uns. Class	*Date* Dimension	Av.Size Module Uns. Class	*Date* and *Trust* Dimensions	Av.Size Module Uns. Class
13	BIO	1.273.281	3	7.514	33	**7.306**	33
14	OBI	24.474	3	**660**	150	717	150

Table 4.7.: Absolute times needed for the experiments (ms).

The results of the evaluation are presented in Figure 4.8 and Table 4.7. For ontology 13 and 14 our algorithm is 97-99% faster than the baseline. For the BIO ontology, which contains 660.915 axioms, provenance can be computed in under 8 seconds with our algorithm, which is fast enough for interactive applications. In contrast, the baseline approach takes almost 22 minutes.

The missing bars for ontology 13 in Figure 4.8 are due to the extreme differences in runtime. For the relatively small ontologies used for experiments 1 to 12, existing pinpointing algorithms perform pretty well. However, for the extremely large ontologies 13 and 14, our optimizations show their full potential.

Using our approach, the time needed to compute provenance mainly scales with the size of pinpoints, rather than with the size of the ontology. Compared to the size especially of ontology 13, our machine had a rather small main memory (most of it was already necessary to just classify the ontology using Pellet). We expect even better performance, when memory management is less of an issue.

Debugging with many independent dimensions can have a negative performance impact due to lots of incomparable labels in the composed dimension. Results for the real-world data from the BIO and the OBI ontologies show that in reality this is less of a problem, as modification dates and editors are not independent there. In fact, the lack of independence of the provenance dimensions is an advantage in practice. The results for ontology 13 are even better for the complex dimension than for the atomic dimensions. This shows that our assumption for the clustered random data obviously is correct.

Precision and Recall

In the final part of our experiment, we have analyzed the precision and recall of our approximation. Precision is defined here as the number of retrieved axioms that are relevant for deriving provenance divided by the total number of retrieved axioms. Recall is defined as the number of retrieved axioms that are relevant for deriving provenance divided by all relevant axioms in the ontology.

The recall clearly always is 1.0, as assured by the inner loop in lines 5 to 9 in `Prov` and `ProvPartial`. We retrieve all axioms of the pinpoint(s) that are relevant and necessary for computing the provenance label (see Figure 4.7). However, we have to tolerate a certain percentage of false positives (low precision), since not all retrieved axioms are relevant for deriving provenance.

Table 4.8 shows that for ontologies 1-3 and 7-9 our approach leads to high precision. However, for the ontologies 4-6, 10-12, as well as for both real-world ontologies 13 and 14, our approach achieves low precision. The reason for this discrepancy is that the latter ontologies are

highly axiomatized ontologies with many syntactic dependencies among axioms. Thus, using the syntactic relevance function our algorithm also retrieves many irrelevant ones. As we have mentioned before, we can trade the runtime optimization in the inner loop of the algorithm for higher precision of the module. In other words, we have a trade-off between (low) precision and (high) performance.

Table 4.8 shows the relation between the size of the module retrieved by the algorithm `Prov` presented in Section 4.5.2 and the total number of relevant axioms for deriving provenance. We measure the average module size for computing the provenance and the average number of axioms in all relevant pinpoints for an inconsistency. The numbers are averaged over all inconsistent classes of an ontology.

	Ontology	Average Axioms	Average Module Size	Precision
1	People Random	**4**	**15**	**0.3**
2	MiniTambis Random	**7**	**22**	**0.3**
3	University Random	4	18	0.2
4	Economy Random	3	57	0.05
5	Chemical Random	7	126	0.06
6	Transport Random	5	70	0.07
7	People Cluster	**4**	**15**	**0.3**
8	MiniTambis Cluster	**7**	**22**	**0.3**
9	University Cluster	4	18	0.2
10	Economy Cluster	3	19	0.16
11	Chemical Cluster	7	126	0.06
12	Transport Cluster	5	102	0.05
13	BIO	3	33	0.09
14	OBI	3	150	0.02

Table 4.8.: Average of the relevant axioms for deriving provenance vs the average of the module size.

4.8. Related Work

In [Baader et al., 2009] the authors propose an approach for computing boundaries for reasoning in sub-ontologies to enforce access restrictions. Provenance is used as criterion for precomputing sub-ontologies during the development phase where the consequences still follow for pre-determined axioms. Two optimizations for reasoning are proposed: An extension of the hitting set tree (HST) algorithm for general lattices, which is close to state of the art pinpointing algorithms and binary search for total orders. For large ontologies, [Baader et al., 2009] uses a modularization step, before applying the actual reasoning. The evaluations of [Baader et al., 2009] and our approach are not directly comparable, as [Baader et al., 2009] focuses on either large ontologies with low expressivity ($\mathcal{EL}$+) or small ontologies with high expressivity. Moreover, the modularization step is not included in the evaluation. [Knechtel and Peñaloza, 2010] extends the approach towards explanations and debugging of access rights.

In contrast, our approach is built on modularization at its core without a need for preprocessing and generates explanations while computing provenance. We have shown that our approach scales very well even for large and very expressive ($\mathcal{SHOIN}(D)$) ontologies. Besides a single access rights dimension, we discuss the management of the various provenance dimensions relevant for the tracking of ontology dynamics. The two approaches might be joined in future work by using the HST algorithm from [Baader et al., 2009] to replace the naïve algorithm used in `ProvPartial`. Such a combined algorithm would have desirable features of both approaches.

[Sensoy et al., 2013] propose an approach to resolve conflicts before performing reasoning. They combine DL-Lite with the Dempster-Shafer theory of evidence (DST) to allow scalable reasoning over uncertain semantic knowledge bases. [Dong et al., 2009] propose to resolve conflicts in information from multiple sources by a voting mechanism. Our approach does not requires that inconsistencies or uncertain information should be resolved before the reasoning process. More similar to our approach, in [Golbeck and Halaschek-Wiener, 2009] Golbeck et al. present a belief revision algorithm for ontologies which is based on trust degrees of information sources to remove conflicting statements from a knowledge base. The authors use axiom pinpointing for detecting conflicts. There are two main differences between our approach and Golbeck's : (1) our approach allows for arbitrary dimensions of provenance, (2) we present an optimized algorithm for computing provenance which uses pinpointing to come up with provenance formulas for description logics.

In [Baader et al., 2009], Baader et al. propose a method for access control to inferences from OWL ontologies. Access rights are modeled using a lattice and pinpointing is used to compute access rights in a way similar to the approach proposed here. Our approach allows for arbitrary dimensions of provenance and hence subsumes the special case of access rights. [McGuinness and da Silva, 2004] propose infrastructure for trust on the semantic web such as portable explanations and service registries. While [McGuinness and da Silva, 2004] and our approach can augment each other, ours is significantly different, because we provide a flexible framework for arbitrary provenance dimensions. Furthermore, we do not need to generate accurate proofs to track provenance.

[Fokoue et al., 2010] introduces a trust framework based on Bayesian Description Logics that allows us to compute a degree of inconsistency over a probabilistic knowledge base. They consider pinpoints as possible worlds for an axiom and derive for each possible word a probability measure. The degree of inconsistency of a knowledge base is then computed as the sum of the probabilities associated with possible worlds that are inconsistent. Due to scalability reasons, the proposed trust computation model operates on a random sample of justifications. Our optimized algorithm does not deal with the sum of all justifications, and thus we do not need to compute all justifications to determine the degree of inconsistency for a query. We are interested in deriving the provenance labels of the changes which led to the inconsistency.

In [Shchekotykhin et al., 2011], the authors propose a framework which allows us to select the most probable diagnosis (pinpoint) to repair an inconsistency. Basically, it creates a sequence of queries which is used to reduce the set of diagnosis until it identifies the target one. The target diagnosis is the one which contains all entailments of the target ontology. To select the best queries to be posted next, the algorithm predicts the information gain of a query result, i. e., the result which minimizes the entropy. Our approach has a different goal when using provenance information for reasoning. Since we do not aim to compute the most probable possible world over all possible worlds but we want to select some specific axioms

based on their provenance, we do not need to compute all pinpoints to track provenance.

Further related work can be grouped into the following categories: (i) Extensions of description logics with a particular provenance dimension, especially uncertainty. (ii) General provenance for query answering with algebraic query languages. (iii) Extensions of description logics with general provenance and (iv) Provenance for other logical formalisms.

ad (i) Several multi-valued extensions of description logic have been proposed: [Lukasiewicz and Straccia, 2008] propose fuzzy and probabilistic extensions of the DLs underlying the web ontology language OWL. [Qi et al., 2011] describe an extension towards a possibilistic logic. [Jung and Lutz, 2012] show an approach for querying probabilistic databases in the presence of an OWL2 QL ontology. Each assertion is assumed to be stored in a database and associated with probabilistic events. Answer probabilities to conjunctive queries can be computed. [Riguzzi et al., 2015b] compute the probability of queries from uncertain DL knowledge bases following the DISPONTE semantics presented in [Riguzzi et al., 2015a]. DISPONTE is a prototype of an expressive DL which minimally extends the language to which it is applied and allows both assertional and terminological probabilistic knowledge. Another extension towards multi-valued logic is presented by [Schenk, 2008]. They aim at trust and paraconsistency instead of uncertainty. OWL-2 is extended to reasoning over logical bilattices. Bilattices, which reflect the desired trust orders, are then used for reasoning. The approach allows for paraconsistent reasoning with OWL taking into account trust levels. [Ma et al., 2007] provide an extension to reasoning in OWL with paraconsistency by reducing it to classical DL reasoning.

On the level of RDF, [Flouris et al., 2009] "color" triples to track information sources. All of these approaches have in common that they modify the character of models in the underlying description logic, e.g. to fuzzy or probabilistic models. In our approach in contrast, we reason on a meta level: While the underlying model remains unchanged, we compute consequences of annotations on axioms. This meta level reasoning is not possible in the approaches proposed above. Unlike general provenance, these approaches are tailored to a specific need and most of them do not respect the provenance dimensions needed for tracking ontology dynamics.

In [Lutz et al., 2008, Artale et al., 2007, Artale et al., 2009, Artale et al., 2013, Gutiérrez-Basulto and Klarman, 2012] extend classical DLs by temporal operators, which then occur within the knowledge base. However, most of these logics yield high reasoning complexities, even if the underlying atemporal DL has tractable reasoning problems In [Artale et al., 2007], the authors propose a temporal extension of description logics which is a combination of the epistemic modal logic $S5$ with the standard DL $\mathcal{ALCQI}$. $S5_{\mathcal{ALCQI}}$ that can represent rigid concepts and roles and allows one to state that concept and role memberships change in time (but without discriminating between changes in the past and future). This approach weakens the temporal dimension to the much simpler S5, but can nevertheless show that adding change (i.e., timestamping on entities, relationships and attributes) pushes the complexity of $\mathcal{ALCQI}$ from ExpTime-complete to 2-ExpTime-hard. Likewise in [Artale et al., 2009] the authors propose a temporal extension of description logic $T_{\diamond}DL-Lite_{bool}$ that can additionally capture some form of evolution constraints. Both works aim at analyzing the satisfiability problem (decidable or undecidable) for temporal extensions of DL. [Baader et al., 2013]'s temporal query language is based on conjunctive queries. Since we do not work with any temporal extension of DL, our problem is restrict to the underlying logic. In [Gutierrez et al., 2007], temporal query answering over temporalized RDF triples and OWL axioms [Motik, 2012, Batsakis et al., 2015, Tappolet and Bernstein, 2009] using an extension of the SPARQL query language is considered.

ad (ii) Provenance for algebraic base languages has been proposed by various authors, for example for the Semantic Web Query Language SPARQL [Geerts et al., 2016, Theoharis et al., 2011, Karvounarakis and Green, 2012, Geerts et al., 2013, Karvounarakis et al., 2013, Dividino et al., 2009b, Hartig, 2009b, Lopes et al., 2010] and for relational databases [Cui and Widom, 2000, Buneman et al., 2000, Buneman et al., 2001, Ding et al., 2005]. While the actual provenance formalisms are comparable to ours, the underlying languages are of lower expressivity, typically Datalog. Provenance in these languages can be evaluated directly using the tree shaped algebraic representations of a query, which is not possible in description logics.

ad (iii) [Tran et al., 2008] propose a provenance extension of OWL, which is also based on annotation properties. Even though provenance can be expressed in ways comparable to ours, it has a rather ad-hoc semantics, which may differ from query to query. In our approach, provenance and classical reasoning take place in parallel. Hence, we can answer queries such as "Give me all results with a confidence degree of $\geq x$". In contrast, reasoning on the ontology and meta level is separated in [Tran et al., 2008]. As a result, [Tran et al., 2008] allows for queries such as "Give me all results which are based on axioms with a confidence degree of $\geq x$". Although this difference might seem quite subtle, depending on the provenance dimension these queries may have dramatically different results.

ad (iv) [Bistarelli et al., 2008] propose an extension of Datalog with weights, which are based on c-semirings and can be redefined to reflect various notions of trust and uncertainty. Our provenance dimensions are similar to c-semirings, but additionally allow to handle conflicting provenance using a third operator. C-semirings have been investigated in great detail and have some desirable properties, such as the fact that the cartesian product of two c-semirings again is a c-semiring. Although based on a less strictly defined algebraic structure, our composition of provenance dimensions described in Section 4.4.2 follows a similar idea.

4.9. Findings and Research Contribution

When reasoning with knowledge from different sources on the Semantic Web, applications need to track the provenance of axioms in living ontologies. Users need tool support for judging the usability and trustworthiness of ontological data, as well as engineering tools supporting the collaborative development of living ontologies. In dynamic ontologies ever-changing criteria may be necessary to establish trust or locate bugs.

We propose a black-box algorithm for optimized reasoning with provenance, which our evaluation with real-world ontologies shows performs significantly better than existing general pinpointing algorithms, scales well, and is applicable for interactive applications. The flexibility of our approach combined with the high scalability makes it a possible building block for a semantic web proof and trust layer.

In this chapter we present our framework restricted to ontology diagnosis scenarios. Our work, however, introduces an approach for provenance querying under a variety of scenarios that consider many of the dimensions of provenance such as restrictions of access rights, knowledge validity when the truth of knowledge changes with time, and inferring trust value.

We have shown that provenance information does not only provide value to the end user, it can be further used to significantly speed up debugging processes by rapidly approximating a solution. In future work, we will apply our approach to provenance to other logical formalisms beyond DL $\mathcal{SRIQ}(\mathcal{D})$.

Part III.

Using Provenance in Semantic Web Applications

5. An Efficient Provenance-Aware News Feed Ranking Algorithm via Preference Aggregation

Overview

Popular online social media sites such as Twitter and Facebook have also become important news and marketing channels. These sites rely on news feed ranking algorithms as they deliver the most relevant messages to the users. In this chapter, we tackle the problem of providing the most relevant messages according to the user's preferences. Users state their preferences on different aspects of the data (e.g., on information source, recentness, reliability, location, etc.), and this information is then aggregated to obtain a joint ranking. We relate our aggregation problem to the problem of *preference aggregation* in social choice theory. The traditional problem formulation of preference aggregation assumes a fixed set of preference orders and a fixed set of domain elements. In this work, however, we investigate how an aggregated preference order has to be updated when the domain is dynamic, i.e., our aggregation approach ranks messages 'on the fly' as the message passes through the system. We establish a framework for online preference aggregation. Our analysis shows that, for all aggregation methods proposed, our approach can handle the dynamic setting more efficiently than standard ones.

Structure

This chapter is organized as follows. Following the introduction, in Section 5.2, we provide a motivation for the use of the different dimensions of provenance information when ranking highly dynamic data (e.g., a news feed ranking scenario). Section 5.3 introduces the foundations for aggregation computation: we briefly introduce the basic concepts of social theory and present three different preference aggregation methods, namely, the plurality, Borda count, and sequential pairwise aggregation methods. Section 5.2 describes the formalization of the online aggregators. Section 5.5 presents some general relationships between the original aggregated preference order and the updated aggregated preference order. Section 5.6 shows a framework for computational approaches to online preference aggregation. Concrete online aggregation algorithms and complexity analyses are presented in Section 5.7. We compare our work with related ones in Section 5.9 and conclude the work in Section 5.10.

5.1. Introduction

Nowadays, most information on the Web is an almost continuous flow of information. Data is constantly being produced, shared, and consumed by a diversity of stakeholders. With so much data on the Web, users want to be sure they won't miss anything they want to see. The goal of a news feed algorithm is deliver to the user the right content at the right time, i.e., it determines which stories are important to the user, and from those, it decides which one should appear first.

One common way to identify the most relevant stories is scoring all of them based on some scoring function, for instance, by showing stories in chronological order. However, even though the newest stories will be displayed to the user, this procedure does not assure us that the stories displayed are also the ones the user is interested in. It is essential that such an algorithm considers a set of features where the score of each element is given according to each feature, and the total score for each element is defined as a combination over its partial scores.

Our problem deals with the task of selecting the set of most relevant messages, given the assumption that messages come 'on the fly', i. e. deriving their rankings that are available upon their arrival. The user's preferences are used to specify which messages are the most relevant to the user. Users state their preferences on different aspects of provenance information (information source, recentness, reliability, location, etc.), i.e., "I am primarily interested reading the most recent messages, the ones which are written by friends, and those which describe events happening near to where I live".

There are many different ways in which preferences can be combined. Preferences have been a long studied subject across many fields including philosophy [Hansson, 2002], artificial intelligence [Brafman and Domshlak, 2009], and databases [Chomicki, 2011]. In the realm of the Semantic Web, Querying the Semantic Web with preferences is also considered in [Gueroussova et al., 2013, Siberski et al., 2006]. Both articles have proposed extensions to SPARQL that support the expression of preferred query results. They illustrated the realization of aggregation with respect to skyline and conditional preference queries. In [Pahlevan et al., 2015], Pahlevan et al. investigate the dynamics of web services in a recommendation scenario. They use well-established database techniques, such as scoring functions to determine an ordering over the results and skyline returning a set of undominated tuples.

We relate the problem of ranking stemming from an aggregation of different provenance dimensions to the problem of *preference aggregation* (or *judgement aggregation*) in social choice theory [Kelly, 1988]. Social choice theory studies the problem how to reach collective consent between a group of people. Judgment aggregation investigates how to aggregate individual judgments on logically related propositions to a group judgment on those propositions. We translate the aggregation problem to the problem of aggregating provenance rankings. By adopting methods from preference aggregation, we formulate a general framework for provenance dimension aggregation that is based on solid formal grounds.

We have established a framework for computational approaches to online preference aggregation. Concretely, we propose three algorithms that iteratively update the aggregated ranking when new results are available, namely for the plurality, the Borda count, and the sequential pairwise voting methods. Finally, we discuss the time complexity analysis of the proposed online preference aggregation and the standard methods and conduct a feasibility study of our algorithms.

Message ID	Movie	Friendship	Date	Popularity
o_1	Star Wars: Episode VII	Family	02.01.16	Low
o_2	The Revenant	Workmate	03.01.16	Medium
o_3	Mistress America	Close friend	01.01.16	High
...				

Table 5.1.: List of messages about movies posted in a social media application by friends, workmates and family members of our sample user. This list is used in our scenario presented in Section 5.2

Research Questions

This chapter addresses the following research question.

RQ 2.I *Can the preference aggregation problem be efficiently solved in a dynamic setting?*

The traditional problem of preference aggregation assumes a fixed set of preference orders and a fixed set of domain elements. Using the standard aggregation algorithms, whenever a new element is added to the domain, a new aggregation has to be re-built. This operation requires expensive computation and cannot be adopted in real use case scenarios. Therefore, it is essential to investigate how preference aggregation methods can be modified in order to assure efficiency in a dynamic setting.

5.2. Motivation Scenario

In our scenario, our sample user, Peter, wants to read messages from his friends and other subscribed sources using a social media application. An example of such messages is shown in Table 5.1.

Our user feels like watching a movie. Therefore, he checks for all messages about movies he received lately in his social media network application. He prefers to read the messages sent from his closest friends first since their interests are more related. Recentness is an important consideration to him since Peter has not been in a movie theater for a long time. Still he would rather read messages that have been recently posted than the most popular ones (e.g. popularity is given by how often a message has been liked and/or shared by other users). So he sets up his preferences: he prefers messages from friends which should not be too old, but recentness is more important than the message's degree of popularity.

Usually, ranking algorithms select the most relevant query results according to each ranking criterion separately, e.g., they return the most recent ones, the most popular ones, and the ones provided only by very close friends, displaying all of them. Since ranking criteria separately may lead to different ranking orderings of the answers, ranking in terms of multiple ranking criteria often delivers an even larger amount of information instead of filtering it. For example, in Table 5.1, a ranking algorithm may return to Peter the messages $o_1 o_2, o_3$ in order o_2, o_1, o_3 if it only considers recentness. The ordering o_3, o_1, o_2 if it considers the friendships degree, and o_3, o_2, o_1 if it considers the popularity degree.

Our user, however, wishes to have just one ranking ordering of answers with respect to an aggregation of his preferences (a personalized aggregation). Additionally, as such messages are

permanently being posted, a ranking mechanism should compute the most relevant messages 'on the fly' as the messages pass through the system.

In order to provide Peter with an optimal user experience, we need a ranking algorithm that scales for real data and assures that Peter constantly receives updates with the most relevant messages with respect to his preferences.

5.3. Foundations: Preference Aggregation

In social choice theory [Kelly, 1988], there are various investigations on the aggregation of multiple preference relations in order to come up with a single preference order. The traditional application for these methods lies in the field of *voting theory*. In the following, we describe the foundations of aggregating preferences.

Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of elements. A *preference order* $\preceq \subseteq \mathcal{X} \times \mathcal{X}$ is a total preorder, i. e., $\preceq$ is a relation that satisfies transitivity

$$(x \preceq y \text{ and } y \preceq z \text{ imply } x \preceq z \text{ for all } x, y, z \in \mathcal{X})$$

and is total

$$(x \preceq y \text{ or } y \preceq x \text{ for all } x, y \in \mathcal{X}).$$

If $x \preceq y$ then we say that y is at least as preferred as x for $x, y \in \mathcal{X}$. Let $\mathcal{R}_{\mathcal{X}}$ be the set of all possible preference orders over elements of $\mathcal{X}$. A vector of preference orders $\vec{\preceq} = \langle \preceq_1, \ldots, \preceq_m \rangle \in \mathcal{R}_{\mathcal{X}}^m$ is called a *preference profile*. A *preference aggregator* Θ is a function $\Theta : \mathcal{R}_{\mathcal{X}}^m \to \mathcal{R}_{\mathcal{X}}$, which maps a preference profile $\vec{\preceq}$ to its aggregated preference order $\preceq$. Specific approaches for preference aggregators are, for instance, plurality, Copeland, and single transferable vote and the Borda preference aggregator [Walsh, 2007].

Example 5.3.1 *Let $\mathcal{X} = \{o_1, o_2, o_3\}$ be the messages presented in Table 5.1 and $\vec{\preceq} = \langle \preceq_1, \preceq_2, \preceq_3 \rangle$ be the preference profile, where $\preceq_1$ defines the friendship order (workmate $\preceq_1$ family $\preceq_1$ friend $\preceq_1$ close friend), $\preceq_2$ the recentness order, and $\preceq_3$ the popularity order. Then:*

$$o_3 \preceq_1 o_1 \preceq_1 o_2$$
$$o_2 \preceq_2 o_1 \preceq_2 o_3$$
$$o_3 \preceq_3 o_2 \preceq_3 o_1$$

Preference aggregators can be classified according to the dimensions they satisfy. Let $\vec{\preceq} \in \mathcal{R}_{\mathcal{X}}^m$ be a preference profile and $\preceq = \Theta(\vec{\preceq})$ an aggregated preference order. Some properties that are usually considered as desirable for a preference aggregator Θ are as follows:

Freeness (FR) Θ is surjective.

Non-dictatorship (ND) There is no $i \in \{1, \ldots, m\}$ such that $\preceq_i = \preceq$ for *every* profile.

Independence of irrelevant alternatives (IIA) If for every two profiles $\langle \preceq_1, \ldots, \preceq_m \rangle$ and $\langle \preceq'_1, \ldots, \preceq'_m \rangle$ and every $i = 1, \ldots, m$ it holds $o \preceq_i o'$ implies $o \preceq'_i o'$ then $o \preceq o'$ implies $o \preceq' o'$ (with $\preceq' = \Theta(\preceq'_1, \ldots, \preceq'_m)$).

Monotonicity (MON) If for every two profiles $\langle \preceq_1, \ldots, \preceq_m \rangle$ and $\langle \preceq'_1, \ldots, \preceq'_m \rangle$ and every $i = 1, \ldots, m$ we have that $\forall o, o' \in \mathcal{O}\ o \preceq_i o'$ implies $o \preceq'_i o'$, under the condition that the relative order of other elements remains the same, then $o \preceq o'$ implies $o \preceq' o'$ (with $\preceq = \Theta(\preceq_1, \ldots, \preceq_m)$ and $\preceq' = \Theta(\preceq'_1, \ldots, \preceq'_m)$).

The main intuitions behind these properties are as follows. Freeness (FR) says that an aggregator can produce any relation order. Non-dictatorship (ND) says that there is no single preference relation $\preceq_i$ that alone can determine the outcome of the aggregated preference relation $\preceq$. Independence of irrelevant alternatives (IIA) states that the preference relation between any pair of elements x and x' depends only on their preferences in $\preceq_1, \ldots, \preceq_m$ (the preference relations of any other pairs of elements y, y' are irrelevant). Monotonicity (MON) requires that if an element is the most preferred under a profile $\vec{\preceq}$ and if under a new profile $\vec{\preceq}'$ every preference order ranks that candidate at least as highly as under $\vec{\preceq}$, without changing the other elements, then the candidate should still be the most preferred one. According to Arrow's famous impossibility result [Arrow, 1950], there is no preference aggregator Θ that satisfies all four properties at once.

In this chapter, we focus on the *plurality* aggregator, the *Borda* count aggregator, and *sequential pairwise voting* (see [Walsh, 2007]). Definitions for these aggregators are given for total orders as well as for total preorders. We adapt the definitions for total preorders (see [Pini et al., 2005]).

The plurality preference aggregator Θ_p is defined as follows.

Definition 5.3.1 (Plurality aggregator) *Let $\vec{\preceq} = \langle \preceq_1, \ldots, \preceq_m \rangle \in \mathcal{R}^m_{\mathcal{X}}$ be a preference profile. For each $x, x' \in \mathcal{X}$ define*

$$\delta(x) = |\{i \mid \forall x' \in \mathcal{X} : x' \preceq_i x\}|$$

Then the plurality preference aggregator Θ_p *is defined via $\Theta_p(\preceq_1, \ldots, \preceq_m) = \preceq$ with*

$$x \preceq x' \qquad \textit{iff} \qquad \delta(x) \leq \delta(x')$$

The plurality preference aggregator defines the set of most preferred elements to be the set of elements with the most first place votes.

Example 5.3.2 *Let $\vec{\preceq}$ be the preference profile defined in Example 5.3.1 over $\mathcal{X} = \{o_1, o_2, o_3\}$, the messages presented in Table 5.1. Using the plurality aggregator, we obtain that $\delta(o_1) = 0$, $\delta(o_2) = 1$, and $\delta(o_3) = 2$.*

$\mathcal{X}$	$\preceq_1$	$\preceq_2$	$\preceq_3$	δ
o_1	*0*	*0*	*0*	*0*
o_2	*0*	*1*	*0*	*1*
o_3	*1*	*0*	*1*	*2*

For $\preceq = \Theta_p(\vec{\preceq})$ it follows

$$o_1 \preceq o_2 \preceq o_3$$

The plurality aggregator ranks the messages as follows: The most relevant message is the message o_3 which, even though it is the oldest of them, it has been delivered by a close friend

and has the highest popularity degree. At second place, it ranks the message o_2 which has been recently provided by a family member. At last, it ranks the message o_1, the most recent one, but delivered by a colleague and with a medium popularity degree.

The Borda preference aggregator Θ_b is defined as follows.

Definition 5.3.2 (Borda aggregator) *Let $\vec{\preceq} = \langle \preceq_1, \ldots, \preceq_m \rangle \in \mathcal{R}_{\mathcal{X}}^m$ be a preference profile. For each $x, x' \in \mathcal{X}$ we define $\delta(x)$ via*[1]

$$\delta(x) = \sum_{i=1}^{m} \sum_{x' \in \mathcal{X} \setminus x} [x' \preceq_i x]$$

Then the Borda preference aggregator Θ_b *is defined via $\Theta_b(\preceq_1, \ldots, \preceq_m) = \preceq$ with*

$$x \preceq x' \qquad \textit{iff} \qquad \delta(x) \leq \delta(x')$$

The Borda preference aggregator defines the aggregated preference relation by ordering outcomes by scores. The score of an element is obtained by summing up the positions of the element in each preference relation. More precisely, an element x is at least as preferred as an outcome x' ($x' \preceq x$), if the sum of all positions over all considered preference orders for x' is smaller or equal to the sum for x.

Example 5.3.3 *Let $\vec{\preceq}$ be the preference profile defined in Example 5.3.1. The positions of all outcomes in the preference relations $\preceq_1, \ldots, \preceq_3$ and the values $\delta(o_1), \delta(o_2)$, and $\delta(o_3)$ are given as*

$\mathcal{X}$	$\preceq_1$	$\preceq_2$	$\preceq_3$	δ
o_1	*1*	*1*	*0*	*2*
o_2	*0*	*2*	*1*	*3*
o_3	*2*	*0*	*2*	*4*

For $\preceq = \Theta_b(\vec{\preceq})$ it follows

$$o_1 \preceq o_2 \preceq o_3$$

Just like plurality, the Borda aggregator ranks o_3 at first, followed by o_2 and then o_1.

As a remark, note that both the plurality and the Borda preference aggregator are instances of *scoring rule voting rules* [Walsh, 2007]. Both plurality and Borda preference aggregators satisfy FR, ND, and MON, but fail IIA.

Lastly, we present the *Sequential pairwise voting*. A nice property of sequential pairwise voting is that is satisfies IIA.

Definition 5.3.3 (Sequential pairwise voting) *Let $\vec{\preceq} = \langle \preceq_1, \ldots, \preceq_m \rangle \in \mathcal{R}_{\mathcal{X}}^m$ be a preference profile. For each $x, x' \in \mathcal{X}$ define*

$$\delta(x, x') = \sum_i x' \preceq_i x$$

[1] Note that $[A]$ is the Iverson bracket defined via $[A] = 1$ if A is true and $[A] = 0$ otherwise.

Then the Sequential pairwise voting Θ_p *is defined via* $\Theta_s(\preceq_1, \ldots, \preceq_m) = \preceq$ *with*

$$\Theta_s(x_1, \ldots, x_n) = \begin{cases} \{x_1 \preceq x_2\} \cup \Theta_s(x_2, \ldots, x_n) & \text{if } \delta(x_1, x_2) \leq \delta(x_2, x_1) \\ \{x_2 \preceq x_1\} \cup \Theta_s(x_1, \ldots, x_n) & \text{if } \delta(x_2, x_1) \leq \delta(x_1, x_2) \end{cases}$$

Sequential voting is similar to single elimination procedures. First, it picks the two first elements of the agenda. An agenda is a pre-defined fixed order on the elements for proceeding with the comparisons. If one element is more preferred than another, it wins the current comparison. This element is taken to face the next element on the preordered agenda. The winner of the last pairwise comparison is the winner of the competition.

Example 5.3.4 *Let* $\vec{\preceq}$ *be the preference profile defined in Example 5.3.1, and an agenda* o_1, o_2, o_3*. We start with* o_1 *vs.* o_2*.* o_2 *wins since it is preferred twice, and* o_1 *just once. Then* o_2 *vs* o_3 *and* o_3 *wins. For* $\preceq = \Theta_s(\vec{\preceq})$ *it follows*

$$o_1 \preceq o_2 \preceq o_3$$

5.4. Preference Aggregation in a Dynamic Setting

5.4.1. Online Preference Aggregator

The standard usage of preference aggregation, as introduced in Secttion 5.3, assumes a static setting, i.e., the elements of the set $\mathcal{X}$ (the domain) and the preference profile $\vec{\preceq} = \langle \preceq_1, \ldots, \preceq_m \rangle$ are known when the aggregated relation $\preceq = \Theta(\vec{\preceq})$ is constructed.

In contrast, in a dynamic setting where the set $\mathcal{X}$ is not available at once, we rather receive the messages continuously. Whenever a new message comes,the preference profile $\vec{\preceq} = \langle \preceq_1, \ldots, \preceq_m \rangle$ has to be extended to consider this element, and that may affect the aggregated relation $\preceq = \Theta(\vec{\preceq})$.

In the following discussion, we move towards preference aggregation with domain changes. We start with the formal introduction of the dynamic preference aggregation problem under domain changes, followed by a formal specification of required updates of the aggregated preference relation $\preceq$.

We assume that $\mathcal{X}$ is the universal set of outcomes and we restrict our attention to a subset $\mathcal{O} = \{o_1, \ldots, o_n\} \subseteq \mathcal{X}$. Let $\mathcal{R}_\mathcal{O}$ be the set of all preference orders on $\mathcal{O}$, i. e., $\mathcal{R}_\mathcal{O} = \mathcal{O} \times \mathcal{O} \subseteq \mathcal{R}_\mathcal{X}$. For $\preceq \in \mathcal{R}_\mathcal{X}$, we denote with $\preceq^\mathcal{O}$ the projection of $\preceq$ on $\mathcal{O}$, i. e. $\preceq^\mathcal{O} = \preceq \cap \mathcal{O} \times \mathcal{O}$ and the preference profile on $\mathcal{O}$ with $\vec{\preceq}^\mathcal{O} = \langle \preceq_1^\mathcal{O}, \ldots, \preceq_m^\mathcal{O} \rangle$. Let $e \in \mathcal{X} \setminus \mathcal{O}$ be a new element in the domain of the preference orders and let $\mathcal{O} \cup \{e\}$ be the set of all elements we consider after adding e to the setting.

Example 5.4.1 *Let* $\vec{\preceq}^\mathcal{O}$ *be a preference profile on the elements in* $\mathcal{O}$*, as defined in Example 5.3.1. Assume now a new element* $e \in \mathcal{X} \setminus \mathcal{O}$ *is added to our setting such as illustrated in Table 5.2. The preferences orders* $\preceq_1, \preceq_2, \preceq_3$ *are updated to incorporate* e *within the orders as follows:*

$$o_2 \preceq_1 o_1 \preceq_1 o_3 \preceq_1 e$$
$$o_3 \preceq_2 o_1 \preceq_2 o_2 \preceq_2 e$$
$$o_1 \preceq_3 o_2 \preceq_3 e \preceq_3 o_3$$

Message ID	Movie	Friendship	Date	Popularity
o_1	Star Wars: Episode VII	Family	02.01.16	Low
o_2	The Revenant	Workmate	03.01.16	Medium
o_3	Mistress America	Friend	01.01.16	High
e	Man on a Ledge	Close friend	04.01.16	Medium
...				

Table 5.2.: Extended list of messages about movies that have been posted by our user's friends in a social media application

$$\begin{array}{ccc} \mathcal{O}, \preceq_1^{O}, \ldots, \preceq_m^{O} & \xrightarrow[\Theta]{} & \preceq = \Theta(\preceq_1^{O}, \ldots, \preceq_m^{O}) \\ \Big\downarrow e \in \mathcal{X} \setminus \mathcal{O} & & \Big\downarrow e, \Lambda \\ \mathcal{O} \cup \{e\}, \preceq_1^{\mathcal{O}\cup e}, \ldots, \preceq_m^{\mathcal{O}\cup e} & \xrightarrow{\Theta} & \preceq' = \Theta(\preceq_1^{\mathcal{O}\cup e}, \ldots, \preceq_m^{\mathcal{O}\cup e}) \end{array}$$

Figure 5.1.: Commuting diagram for online preference aggregation

In this work, we aim to investigate the relationship between $\preceq = \Theta(\preceq^{\vec{\mathcal{O}}})$ and $\preceq' = \Theta(\preceq^{\vec{\mathcal{O}\cup\{e\}}})$ for a given preference aggregator Θ. More precisely, we want to investigate whether there exists a mapping Λ from $\preceq$ to $\preceq'$ as depicted in Fig. 5.1.

Definition 5.4.1 (Online Preference Aggregator) *An online preference aggregator Λ is a function $\Lambda : \mathcal{R}_{\mathcal{X}} \times \mathcal{X} \to \mathcal{R}_{\mathcal{X}}$.*

In other words, we want to find a function Λ that extends an aggregated preference order $\preceq \in \mathcal{R}_{\mathcal{O}}$ with an additional element e directly to a new aggregated preference order $\preceq' \in \mathcal{R}_{\mathcal{O}\cup\{e\}}$ such that Λ mimics the behavior of the underlying preference aggregator.

The idea behind the online preference aggregator Λ is as follows: Λ works directly on the new element e and $\preceq$, and it delivers the same result as by first updating the preference orders $\preceq_1, \ldots, \preceq_m$ and then aggregating again to $\preceq'$. This is formalized in the following definition.

Definition 5.4.2 (Θ-compliant representation) *Let Θ be a preference aggregator and Λ be an online preference aggregator. We say that Λ is a Θ-compliant representation if and only if it holds*

$$\Theta(\preceq_1^{\mathcal{O}\cup\{e\}}, \ldots, \preceq_m^{\mathcal{O}\cup\{e\}}) = \Lambda(\Theta(\preceq_1^{\mathcal{O}}, \ldots, \preceq_m^{\mathcal{O}}), e)$$

for all $\mathcal{O} \subseteq \mathcal{X}$ and $e \in \mathcal{X} \setminus \mathcal{O}$.

Figure 5.1 illustrates the relationship between a preference aggregator Θ and its Θ-compliant representation Λ. The definition of a Θ-compliant representation is very restricting as it re-

quires a functional dependency between the old ($\preceq$) and the new aggregated relation ($\preceq'$), i.e., it requires that $\preceq'$ is functional dependent from (and determined by) $\preceq$. In general, this demand is not always satisfiable.

Example 5.4.2 *Let $\mathcal{O} = \{o_1, \ldots, o_3\}$ and $\vec{\preceq}$ be a preference profile as defined in Example 5.3.1. By using plurality aggregation (Θ_p):*

$\mathcal{X}$	$\preceq_1$	$\preceq_2$	$\preceq_3$	δ
o_1	0	0	0	0
o_2	0	1	0	1
o_3	1	0	1	2

We obtain the same result when we aggregate $\Theta_p(\preceq_1, \preceq_2) = \preceq_{1,2}$ with $(o_1 \preceq_{1,2} \{o_2, o_3\})$ and $\Theta_p(\preceq_2, \preceq_3) = \preceq_{2,3}$ with $(o_1 \preceq_{2,3} \{o_2, o_3\})$.

Now consider a new set of elements $\{o_1, o_2, o_3, e\}$ and we extend $\preceq_1, \preceq_2, \preceq_3$ as in Ex 5.4.1. By using plurality aggregation (Θ_p), we obtain:

$\mathcal{X}$	$\preceq_1$	$\preceq_2$	$\preceq_3$	δ
o_1	0	0	0	0
o_2	0	0	0	0
o_3	0	0	1	1
e	1	1	0	2

$\Theta_p(\preceq_1, \preceq_2) = \preceq'_{1,2}$ is defined via $(\{o_1, o_2, o_3\} \preceq'_{1,2} e)$, and $\Theta_p(\preceq_2, \preceq_3) = \preceq'_{2,3}$ is defined via $(\{o_1, o_2\} \preceq'_{2,3} \{o_3, e\})$. In summary, we get $\preceq_{1,2} = \preceq_{2,3}$ but $\preceq'_{1,2} \neq \preceq'_{2,3}$. This shows that $\preceq'$ cannot be uniquely determined by $\preceq$.

However, when we consider the sequential pairwise aggregator Θ_s, we obtain the results when we aggregate $\Theta_p(\preceq_1, \preceq_2) = \preceq_{1,2}$ with $(\{o_1, o_2, o_3\} \preceq_{1,2} \varnothing)$ and $\Theta_p(\preceq_2, \preceq_3) = \preceq_{2,3}$ with $(o_1 \preceq_{2,3} \{o_2, o_3\})$, and considering a new set of elements $\{o_1, o_2, o_3, e\}$ and the extended $\preceq_1, \preceq_2, \preceq_3$ as in Ex 5.4.1, we obtain $(\{o_1, o_2, o_3\} \preceq_{1,2} e)$ and $\Theta_p(\preceq_2, \preceq_3) = \preceq_{2,3}$ with $(\{o_1\} \preceq_{2,3} \{o_2, o_3\} \preceq_{2,3} e)$.

In summary, the inclusion of the new element e does not change the preferences of the elements o_1, o_2, and o_3 in $\preceq'_{1,2}$ and $\preceq'_{2,3}$. This shows that in some cases $\preceq'$ could be determined by $\preceq$.

The above example shows that there is not always a direct functional dependency between $\preceq$ and $\preceq'$. Therefore, in Section 5.5 we analyze the relationships of the properties described in Section 5.3 with the online preference aggregation setting.

5.4.2. State-based Online Preference Aggregator

In order to update the aggregated preference order, we need some additional information to be carried on from one update (the addition of one element) to the next one. Consequently, we explore state descriptions within our online aggregator (see Definition 5.4.3). A state describes the current aggregated relation.

Definition 5.4.3 *Let $\mathcal{S}$ be a set of states. A* state-based online preference aggregator Δ *is a pair $\Delta = \langle \iota, \tilde{\Lambda} \rangle$ such that*

1. *ι is function $\iota : \mathcal{R}_{\mathcal{X}}^m \to \mathcal{S} \times \mathcal{R}_{\mathcal{X}}$ and*

2. $\tilde{\Lambda}$ *is a function* $\tilde{\Lambda} : \mathcal{S} \times \mathcal{R}_{\mathcal{X}} \times \mathcal{X} \to \mathcal{S} \times \mathcal{R}_{\mathcal{X}}$

The function ι is the *initialization function* that delivers $\iota(\vec{\preceq}) = (s, \preceq)$, where $\preceq$ is the aggregated preference order of $\vec{\preceq}$ and $s \in \mathcal{S}$ some initial state. The set of states might be defined differently depending on the actual preference aggregator (in Section 5.6 we propose concrete stated-based online preference aggregators). Given a state $s \in \mathcal{S}$, an element $e \in \mathcal{X} \setminus \mathcal{O}$, $\tilde{\Lambda}(s, \preceq, e) = (s', \preceq')$, then the updates from state s to s' are used for computing the updates from the aggregated preference order $\preceq$ to $\preceq'$ when adding the element e to the set of elements $\mathcal{O}$.

Definition 5.4.4 *Let Θ be a preference aggregator and $\Delta = \langle \iota, \tilde{\Lambda} \rangle$ be a state-based online preference aggregator. We say that Δ is a state-based Θ-compliant representation if and only if for all profiles $\vec{\preceq} = \langle \preceq_1, \ldots, \preceq_m \rangle \in \mathcal{R}_{\mathcal{X}}^m$ and all elements $\{e_1, \ldots, e_{k-1}\} \in \mathcal{X} \setminus \mathcal{O}$, there are states $s_1, \ldots, s_k \in \mathcal{S}$ such that*

$$\iota(\vec{\preceq}^{\mathcal{O}}) = \langle s_1, \Theta(\vec{\preceq}^{\mathcal{O}}) \rangle$$
$$\tilde{\Lambda}(s_j, \Theta(\vec{\preceq}^{\mathcal{O} \cup \{e_1, \ldots, e_{j-1}\}}), e_j) = \langle s_{j+1}, \Theta(\vec{\preceq}^{\mathcal{O} \cup \{e_1, \ldots, e_j\}}) \rangle$$

for $j = 2, \ldots, k$.

Example 5.4.3 *In Example 5.3.2 $\preceq = \Theta_p(\vec{\preceq}^{\mathcal{O}})$ is defined via $o_1 \preceq o_2 \preceq o_3$. A new element $e \in \mathcal{X} \setminus \mathcal{O}$ is added to the set of elements and the preference orders $\preceq_1, \preceq_2, \preceq_3$ are updated to consider e within the orders as in Example 5.4.1. The updated aggregated relation $\preceq' = \Theta_p(\vec{\preceq}^{\mathcal{O} \cup \{e\}})$ is defined via $\{o_1, o_2\} \preceq' o_3 \preceq' e$. The previous and updated aggregated relations are defined based on the δ function. The values of δ for each element describe, for example, its state in the current aggregated relation. The online aggregator for plurality $\tilde{\Lambda}_p$ explores the states of each element to update $\Theta_p(\vec{\preceq}^{\mathcal{O}})$ to $\Theta_p(\vec{\preceq}^{\mathcal{O} \cup \{e\}})$.*

The state of each element in $\mathcal{O}$ for the previous aggregated relation $\preceq$ is defined by $\delta(o_1) = 0$, $\delta(o_2) = 1$, and $\delta(o_3) = 2$. The next state of the elements in $\mathcal{O} \cup \{e\}$ for the updated preference profile is defined by $\delta(o_1) = 0$, $\delta(o_2) = 0$, $\delta(o_3) = 1$ and $\delta(e) = 2$.

5.5. Aggregators Property Analysis

5.5.1. Independence of Irrelevant Alternatives

First, we consider aggregators that satisfy the independence of irrelevant alternatives (IIA) property. Basically, IIA specifies that if x is preferred to x' out of the set of elements $\mathcal{O} = \{x, x'\}$, then adding a new element e to the set of elements $\{x, x', e\}$, must not make x' preferable to x. In other words, preferences for x or x' should not be changed by the insertion of e, i.e., e is irrelevant to the choice between x and x'. If Θ satisfies IIA, any element that is not x or x' is irrelevant to the preference order of any pair x and x'.

Proposition 5.5.1 *Let $\mathcal{O} \subseteq \mathcal{X}$, $e \in \mathcal{X} \setminus \mathcal{O}$, $\preceq = \Theta(\vec{\preceq}^{\mathcal{O}})$ and $\preceq' = \Theta(\vec{\preceq}^{\mathcal{O} \cup \{e\}})$. If Θ satisfies IIA then*

$$\forall x, x' \in \mathcal{O}, x \preceq x' \Leftrightarrow x \preceq' x'$$

Corollary 5.5.1 *Let $\mathcal{O} \subseteq \mathcal{X}$, $e \in \mathcal{X} \setminus \mathcal{O}$, $\preceq = \Theta(\vec{\preceq}^{\mathcal{O}})$ and $\preceq' = \Theta(\vec{\preceq}^{\mathcal{O}\cup\{e\}})$. If Θ satisfies IIA then:*

$$\preceq = \preceq' \cap \mathcal{O} \times \mathcal{O}.$$

The next example shows, for sequential voting, an update from $\preceq$ and $\preceq'$.

Example 5.5.1 *In Example 5.3.4 $\preceq = \Theta_s(\vec{\preceq}^{\mathcal{O}})$ is defined via $o_1 \preceq o_2 \preceq o_3$. A new element $e \in \mathcal{X} \setminus \mathcal{O}$ is added to the set of elements, and the preference orders $\preceq_1, \preceq_2, \preceq_3$ are updated to consider e within the orders as Example 5.4.1 and an agenda $\langle o_1, o_2, o_3, e\rangle$. We start with o_1 vs. o_2 and o_2 wins. Then o_2 vs o_3 and o_3 wins, and finally o_3 vs. e, where e wins. The updated aggregated relation $\preceq' = \Theta_s(\vec{\preceq}^{\mathcal{O}\cup\{e\}})$ is defined via $o_1 \preceq' o_2 \preceq' o_3 \preceq' e$. Note that $\preceq'$ does not change, and thus the relative orders of the elements in $\mathcal{O}$ is maintained.*

Furthermore, since all previous relations between every two elements from $\preceq$ are maintained in $\preceq'$, the number of changes from $\preceq$ to $\preceq'$ are limited to the number of the preference relations that have been created due to the insertion of the element e.

Corollary 5.5.2 *Let $\mathcal{O} \subseteq \mathcal{X}$, $e \in \mathcal{X} \setminus \mathcal{O}$, $\preceq = \Theta(\vec{\preceq}^{\mathcal{O}})$ and $\preceq' = \Theta(\vec{\preceq}^{\mathcal{O}\cup\{e\}})$. If Θ satisfies IIA then*[2]

$$|\preceq' \Delta \preceq| \leq 2|\mathcal{O} \cup \{e\}|$$

Example 5.5.2 *In Examples 5.3.4 and 5.5.1, we have computed the aggregated preference relations wrt. the sequential voting for the sets $\{o_1, o_2, o_3\}$, and $\{o_1, o_2, o_3, e\}$. Thus, it is $|\preceq_p \emptyset_1, \ldots, o_3\} \Delta \preceq_p^{\{o_1,\ldots,o_3,e\}}| = |\{(e,e),(o_1,e),(o_2,e),(o_3,e)| = 4 = |\{o_1, o_2, o_3, e\}|$.*

5.5.2. Monotonicity

We turn to aggregators that satisfy monotonicity (MON). Briefly, if an element x gets more preferred without changing the order of other elements, then the position of x cannot decrease. Thus, a monotone aggregation leaves the ordering of the elements unchanged.

Given a monotone aggregation, we can distinguish between the following cases after the insertion of a new element e.

1. If e is positioned arbitrarily in every preference relations $\preceq_i'$, then a monotone aggregation leaves the ordering unchanged of those elements $\mathcal{O}$ that are more preferred than e in every $\preceq_i'$.
2. If e is positioned arbitrarily in every preference relation $\preceq_i'$, then a monotone aggregation leaves the updated ordering in $\preceq'$ at least as good as in $\preceq$ of those elements in $\mathcal{O}$ that are less preferred than e in every $\preceq_i'$.

Following these observations we state the propositions:

Proposition 5.5.2 *Let Θ satisfy MON and let $\mathcal{O} \subseteq \mathcal{X}$, $\preceq = \Theta(\vec{\preceq}^{\mathcal{O}})$ and $\preceq' = \Theta(\vec{\preceq}^{\mathcal{O}\cup\{e\}})$. Then $(\forall e \in \mathcal{X} \setminus \mathcal{O} : (\forall i : (\exists \mathcal{T} \subseteq \mathcal{O} : (\forall x, x' \in \mathcal{T} : (e \preceq_i' x \wedge e' \preceq_i' x')))) \Longrightarrow (x \preceq x' \rightarrow x \preceq' x' \wedge e \preceq' x \wedge e \preceq' x'))$*

[2]The symmetric set difference Δ is defined as follows: $\preceq' \Delta \preceq = \{(x,x') | (x,x') \in \preceq' \wedge (x,x') \notin \preceq \vee (x,x') \notin \preceq' \wedge (x,x') \in \preceq\}$

Proposition 5.5.3 *Let Θ satisfy MON and let $\mathcal{O} \subseteq \mathcal{X}$, $\preceq = \Theta(\vec{\preceq}^{\mathcal{O}})$ and $\preceq' = \Theta(\vec{\preceq}^{\mathcal{O}\cup\{e\}})$. Then $(\forall e \in \mathcal{X} \setminus \mathcal{O} : (\forall i : (\exists \mathcal{B} \subseteq \mathcal{O} : (\forall x, x' \in \mathcal{B} : (x \preceq'_i e \wedge x' \preceq'_i e)))) \Longrightarrow (x \preceq x' \rightarrow x \preceq' x' \wedge x \preceq' e \wedge x' \preceq' e))$*

Note that both plurality and Borda aggregators satisfy monotonicity. The next example shows an update from $\preceq$ to $\preceq'$ for the plurality aggregator.

Example 5.5.3 *In Example 5.3.2 $\preceq = \Theta_p(\vec{\preceq}^{\mathcal{O}})$ is defined via $o_1 \preceq o_2 \preceq o_3$. A new element $e \in \mathcal{X} \setminus \mathcal{O}$ is added to the set of elements, and the preference orders $\preceq_1, \preceq_2, \preceq_3$ are updated to consider e within the orders as Example. 5.4.1. The updated aggregated relation $\preceq' = \Theta_p(\vec{\preceq}^{\mathcal{O}\cup\{e\}})$ is defined via $\{o_1, o_2\} \preceq' o_3 \preceq' e$. Note that Prop. 5.5.3 holds for element o_1 and o_2.*

These observations state that there are cases in which the aggregated preference order only changes minimally when adding a new element to the domain. Thus, it would seem beneficial to investigate the computational issue of online preference aggregation.

5.6. Online Ranking Algorithms

In the following discussion, we consider how to effectively update preference order in the light of a newly-introduced element.

5.6.1. Naïve Algorithm

The standard approach for ranking over streams starts with an empty top-k message set. As messages arrive, they are added to the list until it is full (k messages). We suppose that the k messages are sorted with respect to the user's preferences. As a new message arrives, a simple (naïve) algorithm, the *naïve online preference aggregator*, just applies the original preference aggregation on every change.

Definition 5.6.1 *Let $\mathcal{O} \subseteq \mathcal{X}$, $e \in \mathcal{X} \setminus \mathcal{O}$, Θ be a preference aggregator and $\mathcal{S}_{naive} = \mathcal{R}^o_{\mathcal{X}}$. The naive online preference aggregator Δ^{naive}_{Θ} for Θ is a pair $\Delta^{naive}_{\Theta} = \langle \iota^{naive}_{\Theta}, \tilde{\Lambda}^{naive}_{\Theta} \rangle$ with $\iota^{naive}_{\Theta} : \mathcal{R}^m_X \rightarrow \mathcal{S}_{naive} \times \mathcal{R}_X$ and $\tilde{\Lambda}^{naive}_{\Theta} : \mathcal{S}_{naive} \times \mathcal{R}_X \times X \rightarrow \mathcal{S}_{naive} \times \mathcal{R}_X$ defined via*

$$\iota^{naive}_{\Theta}(\vec{\preceq}^{\mathcal{O}}) = \langle \vec{\preceq}^{\mathcal{O}}, \Theta(\vec{\preceq}^{\mathcal{O}}) \rangle$$
$$\tilde{\Lambda}^{naive}_{\Theta}(\vec{\preceq}^{\mathcal{O}}, \Theta(\vec{\preceq}^{\mathcal{O}}), e) = \langle \vec{\preceq}^{\mathcal{O}\cup\{e\}}, \Theta(\vec{\preceq}^{\mathcal{O}\cup\{e\}}) \rangle$$

Example 5.6.1 *Let $\vec{\preceq}^{\mathcal{O}}$ be a preference profile on $\mathcal{O}$ as defined in Example 5.3.1. For the plurality aggregator Θ_p we obtain*

$$\iota^{naive}_{\Theta}(\vec{\preceq}^{\mathcal{O}}) = \langle \{\langle o_3 \preceq_1 o_1 \preceq_1 o_2 \rangle, \langle o_2 \preceq_2 o_1 \preceq_2 o_3 \rangle, \langle o_3 \preceq_3 o_2 \preceq_3 o_1 \rangle\}, o_1 \preceq \{o_2, o_3\} \rangle$$

Assume the preferences orders $\preceq_1, \preceq_2, \preceq_3$ are updated to consider a new element $e \in \mathcal{X} \setminus \mathcal{O}$ as described in Algorithm 5. Then $\tilde{\Lambda}^{naive}_{\Theta}$ just applies the original preference aggregation Θ on the profile $\vec{\preceq}^{\mathcal{O}\cup\{e\}}$.

$$\tilde{\Lambda}^{naive}_{\Theta}(\{\langle o_3 \preceq_1 o_1 \preceq_1 o_2 \rangle, \langle o_2 \preceq_2 o_1 \preceq_2 o_3 \rangle, \langle o_3 \preceq_3 o_2 \preceq_3 o_1 \rangle\}, o_1 \preceq \{o_2, o_3\} \rangle, e) =$$
$$\langle \{\langle o_2 \preceq_1 o_1 \preceq_1 o_3 \preceq_1 e \rangle, \langle o_3 \preceq_2 o_1 \preceq_2 o_2 \preceq_2 e \rangle, \langle o_1 \preceq_3 o_2 \preceq_3 e \preceq_3 o_3 \rangle\}, \{o_1, o_2\} \preceq' o_3 \preceq' e \rangle$$

Algorithm 5 Plurality Online Aggregator - Naive Algorithm (see Definition 5.3.1)

```
1: function PluralityAggregationNaive(elements[], preferences[], newElement)
2:     for j = 0 to m do
3:         newPref[j]= computePreference(elements, newElement, preferences[j]);
4:         winner[] = getElement(newPref[j], 0);
5:         for i = 0 to winner.size do
6:             winner = winner[i];
7:             score[winner] = score[winner]+ 1;
8:             elementRank.add(winner, score);
9:         end for
10:    end for
11:    return elementRank;
12: end function
```

Algorithm 6 Borda Online Aggregator - Naive Algorithm (see Definition 5.3.2)

```
1: function BordaAggregationNaive(elements[], preferences[], newElement)
2:     for j = 0 to m do
3:         newPref[j]= computePreference(elements, newElement, preferences[j]);
4:     end for
5:     for j = 0 to m do
6:         for i = 0 to n do
7:             element[] = getElement(newPref[j], i);
8:             for k= 0 to element.size do
9:                 score=elementRank.get(element[k]) +
10:                (elementRank.size() - newPref.getPosition(element[k]);
11:                elementRank.add(element[k], score);
12:            end for
13:        end for
14:    end for
15:    return elementRank;
16: end function
```

Algorithms 5, 6, and 7 present the naïve version of the plurality, Borda count, and sequential pairwise voting aggregators respectively based on the Definitions 5.3.1, 5.3.2, and 5.3.3.

The following proposition and theorem follows by construction of the naïve online preference aggregator.

Proposition 5.6.1 *Let Θ be a preference aggregator. Then Δ_{Θ}^{naive} is a stated-based Θ-compliant representation.*

Theorem 5.6.1 *Let Θ be a preference aggregator of time complexity $O(n*m)$. Then ι_{Θ}^{naive} has time complexity $O(n*m)$ and $\tilde{\Lambda}_{\Theta}^{naive}$ has time complexity $O(n+(n*m))$. The space complexity for storing a state in $\mathcal{S}_{naive}$ is $O(n*m)$.*

Our goal is to investigate if the time complexity of $O(n*m)$ can be improved.

5.6.2. Optimized Algorithm

In the following, we are going to define the online aggregator for two popular voting aggregators, namely plurality (Θ_p) and Borda (Θ_b). For these algorithms we explore the monotocity

Algorithm 7 Sequential Pairwise Online Aggregator - Naive Algorithm (see Definition 5.3.3)

```
1: function SequentialAggregationNaive(elements[], preferences[], newElement, agenda[])
2:     for j = 1 to m do
3:         newPref[j]= computePreference(elements, newElement, preferences[j]);
4:     end for
5:     for k = 0 to agenda.size() -1 do
6:         element1 = agenda[k];
7:         element2 = agenda[k+1];
8:         for j = 1 to m do
9:             score1 = score1 + newPref[j].getPosition(element1);
10:            score2 = score2 + newPref[j].getPosition(element2);
11:        end for
12:        if score1 < score2 then
13:            elementRank.add(element1, k);
14:            elementRank.add(element2, k+1)
15:        end if
16:        if score1 > score2 then
17:            elementRank.add(element2, k);
18:            elementRank.add(element1, k+1)
19:        end if
20:        if score1 = score2 then
21:            elementRank.add(element1, k);
22:            elementRank.add(element2, k);
23:        end if
24:    end for
25:    return elementRank;
26: end function
```

property: if an element x gets more preferred without changing the order of other elements, then the position of x cannot decrease. Thus, a monotone aggregation leaves the ordering of elements unchanged (see Section 5.5).

The Online Plurality Aggregator

For the plurality preference aggregator, the status of an aggregated relation is defined by the function *pos* that maps each element $x \in \mathcal{X}$ to a m-dimensional vector. For each dimension $i \in 1, \ldots, m$, the value is equal to 1 if x is the winner on the ith-preference relation, and 0 otherwise, e.g., $pos(o_3)$ is $\{1, 0, 1\}$ (see Example 5.3.2). The δ-plurality score of an element is given by summing up all values in its vector. Accordingly, we define the plurality online preference aggregator as follows:

Definition 5.6.2 *Let $\mathcal{O} \subseteq \mathcal{X}$, $e \in \mathcal{X} \setminus \mathcal{O}$, Θ_p be the plurality preference aggregator and $\mathcal{S} : \{0,1\}^{m \times |\mathcal{X}|}$ is a function which maps an element to the m-dimensional vector on {0,1}. The online plurality preference aggregator $\Delta^{online}_{\Theta_p}$ for Θ_p is a pair $\Delta^{online}_{\Theta_p} = \langle \iota^{online}_{\Theta_p}, \tilde{\Lambda}^{online}_{\Theta_p} \rangle$ with $\iota^{online}_{\Theta_p} : \mathcal{R}^m_X \to \mathcal{S} \times \mathcal{R}_X$ and $\tilde{\Lambda}^{online}_{\Theta_p} : \mathcal{S} \times \mathcal{R}_X \times X \to \mathcal{S} \times \mathcal{R}_X$ defined via*

$$\iota^{online}_{\Theta_p}(\vec{\preceq}^{\mathcal{O}}) = \langle s, \Theta_p(\vec{\preceq}^{\mathcal{O}}) \rangle$$
$$\tilde{\Lambda}^{online}_{\Theta_p}(s, \Theta_p(\vec{\preceq}^{\mathcal{O}}), e) = \langle s', \Theta_p(\vec{\preceq}^{\mathcal{O} \cup \{e\}}) \rangle$$

Algorithm 8 Online Plurality Aggregator Θ_p - Optimized Algorithm

```
1: function PluralityAggregationOpt(element, preference[])
2:     for j = 1 to m do
3:         winner[] = getElement(preference[j], 0);
4:         comp = compare(preference[j], winner[0], element);
5:         if comp = 0 then
6:             score[element] = score[element]+ 1;
7:             elementRank.add(element, score);
8:         elsecomp = 1
9:             score[element] = score[element]+ 1;
10:            elementRank.add(element, score);
11:            for int i= 0 to winner.size do
12:                winner = winner[i];
13:                score[winner] = score[winner] - 1;
14:                elementRank.add(winner,score);
15:            end for
16:        end if
17:    end for
18:    return elementRank;
19: end function
```

For the plurality aggregation, the elements with the most first places win. Therefore, it is essential to check how often the new element e gets the first place in the preference relations $\preceq_1, \ldots, \preceq_n$. Whenever e gets the first place in $\preceq_i, i = \{1, \ldots, n\}$, we have to update $\preceq_i$ by adding e as the new winner, i.e., the ith-value in its m-dimensional vector to 1. It is not necessary to update the states of any element where e is not the winner, since (1) the previous winner remains the winner in the updated relation and (2) we assume that the state of e is initially set to a vector of 0s.

In case the element e gets the most first places, it will be the new winner in the updated aggregated relation. Following monotonicity, if e gets to be the winner, it does not change the order of the other elements in the aggregated relation i.e., it leaves the ordering of the less preferred elements unchanged; therefore to update the new aggregated relation, we only have to add e in the first place.

In the optimized algorithm, instead of updating each preference relation $\preceq_i$ (comparing all elements with each other), the online plurality algorithm compares only the new element e with the winner in $\preceq_i$. The winner in each $\preceq_i$ is given by the element's state. Note that a preference relation can have multiple winners, but as we assume that the preference relations are total preorders (the preorder is called total if any two elements $x, y \in \mathcal{X}$ can be compared, i.e., either $x \preceq y$, or $y \preceq x$ (or both), see Section 5.3), for every two winners they must be equivalent in $\preceq_i$. Therefore, we only have to check the value of one of the winners. In case the element e is the only winner of the updated preference relation, then we add 1 to its δ-plurality score (δ) and decrease the δ-plurality score of the previous winners in 1. In case the element e is one of the winners, we only add 1 to its plurality score. The optimized online plurality algorithm is presented in Algorithm 8.

Example 5.6.2 *Let $\vec{\preceq}^{\mathcal{O}}$ be preference profile on $\mathcal{O}$ as defined in Example 5.3.1. For the online*

plurality aggregator we obtain:

$$\iota^{online}_{\Theta_p}(\vec{\preceq}^{\mathcal{O}}) = \langle\langle\{0,0,0\},\{0,1,0\},\{1,0,1\}\rangle\rangle,\preceq\rangle$$

and $\forall x, x' \in \mathcal{O}, x \preceq x'$ *if* $\sum_{i=1}^{m} pos(x)[i] \leq \sum_{i=1}^{m} pos(x')[i]$*, therefore* $\preceq$ *is defined via* $o_1 \preceq o_2 \preceq o_3$*. Assume we know the winner of each preference order (by checking the state function), then we compare the winner of each preferences orders* $\preceq_1, \preceq_2, \preceq_3$ *with the element* $e \in \mathcal{X} - \mathcal{O}$*. If* e *is the winner, then we update its plurality score i. e. its state as described in Algorithm 8. Then we have,*

$$\Lambda\iota^{online}_{\Theta_p}(\langle\langle\{0,0,0\},\{0,1,0\},\{1,0,1\}\rangle\rangle,\preceq,e) = \langle\langle\{0,0,0\},\{0,0,0\},\{0,0,1\},\{1,0,1\}\rangle\rangle,\preceq'\rangle$$

we update the states of those elements that were winners in $\preceq$*, but* e *is currently the winner in* $\preceq'$*, therefore* $\preceq'$ *is defined via* $\{o_1, o_2\} \preceq' o_3 \preceq' e$*.*

Proposition 5.6.2 Δ_p *is a state-based* Θ_p*-compliant representation.*

The Online Borda Count Aggregator

Now we turn our discussion to the online Borda count preference aggregator. Similar to the plurality aggregator, we describe a state by the function *pos* that maps each element $x \in \mathcal{X}$ to an m-dimensional vector. For each dimension $i \in 1,\ldots,m$, the value is the score of the Borda rule of element x in the ith-preference relation, e.g., $pos(o_3)$ is $\{2,0,2\}$ (see Example 5.3.3). The δ-Borda score of an element is given by summing up all values in its vector.

Definition 5.6.3 *Let* $\mathcal{O} \subseteq \mathcal{X}$*,* $e \in \mathcal{X} \setminus \mathcal{O}$*,* Θ_b *be the Borda preference aggregator and* $\mathcal{S} : \{0,\ldots,|\mathcal{X}|-1\}^{m\times|\mathcal{X}|}$ *is a function which maps an element to the m-dimensional vector on* $\{0,..,|\mathcal{X}|-1\}$*. The Borda online preference aggregator* $\Delta^{online}_{\Theta_b}$ *for* Θ_b *is a pair* $\Delta^{online}_{\Theta_b} = \langle\iota^{online}_{\Theta_b}, \tilde{\Lambda}^{online}_{\Theta_b}\rangle$ *with* $\iota^{online}_{\Theta_b} : \mathcal{R}^m_X \to \mathcal{S}\times\mathcal{R}_X$ *and* $\tilde{\Lambda}^{online}_{\Theta_b} : \mathcal{S}_\times\mathcal{R}_X \times X \to \mathcal{S}\times\mathcal{R}_X$ *defined via*

$$\iota^{online}_{\Theta_b}(\vec{\preceq}^{\mathcal{O}}) = \langle s, \Theta_b(\vec{\preceq}^{\mathcal{O}})\rangle$$
$$\tilde{\Lambda}^{online}_{\Theta_b}(\langle s, \Theta_b(\vec{\preceq}^{\mathcal{O}}), e) = \langle s', \Theta_b(\vec{\preceq}^{\mathcal{O}\cup\{e\}})\rangle$$

For the Borda count aggregation, only the elements those δ-Borda score is the same or bigger than the current δ-Borda score of e have to be updated (increased by 1). Therefore, we have to check, for each preference relation $\preceq_i, i = \{0,\ldots,n\}$, the position of the new element e. For each $\preceq_i$, whenever e gets the first place, we need to update only its m-dimensional vector of itself, i.e., the i-value changed from 0 to its δ-Borda score (the total number of elements). The vectors of the other elements remain the same. Whenever e gets the last place, we need to update the δ-Borda score of all other elements by adding 1 to their previous scores. Note that the relations between every pair of elements remains also the same in the updated relation $\preceq_i$. Additionally, it is not necessary to update the state of e when it is the least preferred element, since we assume that the state of e is initially set to a vector of 0s.

In case the element e gets the highest δ-Borda score in all preference relations, it turns out to be the new winner in the updated aggregated relation. Again, following monotonicity,

Algorithm 9 Borda Online Aggregator Θ_b - Optimized Algorithm

```
1: function BordaAggregationOpt(element, preference[])
2:     for j = 1 to m do
3:         positionOf(element, preference[j], 0 , preference[j].size);
4:     end for
5:     return elementRank;
6: end function
```

if e gets to be the winner, it leaves the ordering of the less preferred elements unchanged; therefore, to update the new aggregated relation, we only have to add e in the first place.

We want to avoid to make unnecessary comparisons as the naïve algorithm does, i.e., update the preference relations by comparing all elements with each other. However, in order to know the position of the new element e in each of the relations, we need to do more comparisons as we did by plurality (only comparing with the winners), since e can be placed everywhere.

The online Borda count algorithm uses the divide and conquer method to recursively break down the search space. At the first step, it checks in $\preceq_i$ if e is more or less preferred as the element placed in the middle, m. If e is less preferred m ($e \preceq_i m$), we already know that we need to update the δ-Borda score of all the elements equally or more preferred than m by adding 1 to their previous score. The algorithm continues the search considering now only the elements less preferred than m in $\preceq_i$. If e is more preferred than m ($m \preceq_i e$), then none of the states have to be updated and the algorithm continues to search the position of e by considering only the elements more preferred than m in $\preceq_i$. Whenever we find the position of e, we update its δ-Borda score and the one of the remaining elements that are more preferred than e.

Note that any preference relation i can have multiple elements placed in one position, but as we assume that all preference relations are total preorders (see Section 5.3), such elements must be equivalent in $\preceq_i$. Therefore, we only have to check the value of one of the elements in each position. The optimized online Borda count algorithm is presented in Algorithm 9 and the search method is presented in Algorithm 10.

Example 5.6.3 *Let $\vec{\preceq}^{\mathcal{O}}$ be preference profile on $\mathcal{O}$ as defined in Example 5.3.1. For the online Borda count aggregator we obtain:*

$$\iota^{online}_{\Theta_b}(\vec{\preceq}^{\mathcal{O}}) = \langle\langle\{1,1,0\},\{0,2,1\},\{2,0,2\}\rangle,\preceq\rangle$$

and $\forall x, x' \in \mathcal{O}, x \preceq x'$ if $\sum_{i=1}^{m} pos(x)[i] \leq \sum_{i=1}^{m} pos(x')[i]$, therefore $\preceq$ is defined via $o_1 \preceq o_2 \preceq o_3$. Assume now the new position of the element $e \in \mathcal{X} - \mathcal{O}$ is found in each $\preceq_1, \preceq_2, \preceq_3$, and the preferences orders $\preceq_1, \preceq_2, \preceq_3$ are updated to consider the element e as described in Algorithm 9. Then we have,

$$\begin{aligned}&\tilde{\Lambda}^{online}_{\Theta_b}(\langle\{1,1,0\},\{0,2,1\},\{2,0,2\}\rangle,\preceq,e)\\ &= \langle\langle\{1,1,0\},\{0,2,1\},\{2,0,3\},\{3,3,2\}\rangle,\preceq'\rangle\end{aligned}$$

we update the states of those elements that their δ-Borda scores in $\preceq$ is the same or bigger than the current Borda score of e for the same dimension, and therefore $\preceq'$ is defined via $o_1 \preceq' o_2 \preceq' o_3 \preceq' e$.

Algorithm 10 Search for the Position of the New Element - Optimized Algorithm

```
1: function positionOf(element, preference, start, end)
2:     if preference.size = 0 then
3:         score = elementRank.get(element) + (elementRank.size() - start);
4:         elementRank.add(element, score);
5:         return
6:     end if
7:     index = start + (end - start) /2;
8:     elements[] = getElement(preference, index);
9:     comp = compare(element, elements[0]);
10:    if comp = 0 then
11:        for i = start to (index - 1) do
12:            e = preference.getElement(i);
13:            score = elementRank.get(e) + 1;
14:            elementRank.add(e, score);
15:        end for
16:        score = elementRank.get(element) + (elementRank.size() - index);
17:        elementRank.add(element, score);
18:        return ;
19:    end if
20:    if comp ¡ 0 then
21:        for i = start to index do
22:            e = preference.getElement(i);
23:            score = elementRank.get(e) + 1;
24:            elementRank.add(e, score);
25:        end for
26:        return positionOf(element, preference, index + 1 , end);
27:    else
28:        return positionOf(element, preference, start, index - 1);
29:    end if
30: end function
```

Proposition 5.6.3 *Δ_b is a state-based Θ_b-compliant representation.*

The Online Sequential Pairwise Aggregator

At last, we propose an optimized algorithm for the online sequential pairwise aggregator (Θ_s). This aggregator follows the IIA property (see Section 5.5), which means that the preferences between the elements do not change by the insertion of e, i.e., e is irrelevant to the choice between any pair of elements.

Since the insertion of a new element into the element set does not change the preference order between any two other elements, it follows that $\preceq$ is a projection of $\preceq'$ (see Collolary 5.5.1). Consequently, we can avoid re-computing the preference relations (as the naïve algorithm) since it is given that the relation between the pairs of elements cannot change, and thus we can copy them. Nevertheless, we need to compute the preference relation between the new element e with all other elements.

For the sequential pairwise aggregator, we describe a state by the function δ that maps each pair of element $x, x' \in \mathcal{X}$ to a value which described the preference between x, x'. If x is more preferred than x', then it wins the current comparison and $\delta(x, x') = 1$, otherwise $\delta(x, x') = 0$

Algorithm 11 Online Sequential Pairwise Aggregator Θ_s - Optimized Algorithm

```
1:  function SequentialPairwiseAggregationOpt(element, preference[], agenda)
2:      for j = 1 to m do
3:          lastElement = getElement(agenda, agenda.size);
4:          position = preference[j].getPosition(lastElement)
5:          comp = compare(element, lastElement);
6:          if comp = 0 then
7:              elementRank.add(element, position);
8:          end if
9:          if comp > 0 then
10:             elementRank.add(element, position + 1);
11:         else
12:             elementRank.add(element, position - 1);
13:         end if
14:     end for
15:     return elementRank;
16: end function
```

(for an example see Example 5.3.4).

Definition 5.6.4 *Let $\mathcal{O} \subseteq \mathcal{X}$, $e \in \mathcal{X} \setminus \mathcal{O}$, Θ_b be the sequential pairwise preference aggregator and $\mathcal{S} : \mathcal{R}$ is a function which maps two elements to a real number. The online sequential pairwise preference aggregator $\Delta^{online}_{\Theta_s}$ for Θ_s is a pair $\Delta^{online}_{\Theta_s} = \langle \iota^{online}_{\Theta_s}, \tilde{\Lambda}^{online}_{\Theta_s} \rangle$ with $\iota^{online}_{\Theta_s} : \mathcal{R}^m_X \to \mathcal{S} \times \mathcal{R}_X$ and $\tilde{\Lambda}^{online}_{\Theta_s} : \mathcal{S}_\times \mathcal{R}_X \times X \to \mathcal{S} \times \mathcal{R}_X$ defined via*

$$\iota^{online}_{\Theta_s}(\vec{\precsim}^{\mathcal{O}}) = \langle s, \Theta_s(\vec{\precsim}^{\mathcal{O}}) \rangle$$
$$\tilde{\Lambda}^{online}_{\Theta_s}(\langle s, \Theta_s(\vec{\precsim}^{\mathcal{O}}), e) = s\langle s', \Theta_s(\vec{\precsim}^{\mathcal{O}\cup\{e\}}) \rangle$$

An important property of the sequential pairwise aggregator is that it uses a fixed agenda. An agenda is a pre-defined fixed order on the elements for proceeding the comparisons. If an element is more preferred than the other, then it wins the current comparison. This element is taken to face the next element on the preordered agenda. The winner of the last pairwise comparison is the winner of the competition.

As the aggregation results may depend on the order in which the elements are compared, for two different agendas, we may get two different aggregated relations. Consequently, in order to preserve the aggregation ordering (IIA), when adding a new element e in the system, the order between the elements in the agenda cannot be changed. Therefore we are only allowed to add e at the top of the agenda. For instance, in Example 5.3.4, the agenda is defined as $\{o_1, o_2, o_3\}$, in order to assure that o_1 will be compared with o_2, and o_3 with o_3, the new element e cannot be placed at first because if it wins the comparison with o_1, then the comparison o_1 vs. o_2 cannot be taken in place anymore. The same phenomenon occurs when e is placed in between any two elements. This implies that in order to update the aggregated relation, the optimized algorithm needs to conduct the comparison of e with only the last element in the agenda, and updates their δ-sequential pairwise scores. The optimized online plurality algorithm is presented in Algorithm 8.

Example 5.6.4 *Let $\vec{\precsim}^{\mathcal{O}}$ be preference profile on $\mathcal{O}$ as defined in Example 5.3.1. For the online*

sequential pairwise aggregator we obtain:

$$\iota^{online}_{\Theta_s}(\vec{\preceq}^{\mathcal{O}}) = \langle\langle\{0,1\}\rangle, \preceq\rangle$$

and $\forall x_1, x_2 \in \mathcal{O}, x_1 \preceq x_2$ *if* $\delta(x_1, x_2) < \delta(x_2, x_1) \cup \Theta_s(x_2, \ldots, x_n)$ *, therefore* $\preceq$ *is defined via* $o_1 \preceq o_2 \preceq o_3$*. Assume now the preferences orders* $\preceq_1, \preceq_2, \preceq_3$ *are updated to consider the element* $e \in \mathcal{X} - \mathcal{O}$ *as described in Algorithm 11. Then we have,*

$$\begin{aligned}\tilde{\Lambda}^{online}_{\Theta_s}(\langle\{1,0\}\rangle, \preceq, e)\\ = \langle\langle\{1,0,0\}\rangle, \preceq'\rangle\end{aligned}$$

we update the state of the last element of the agenda and the one from e*, and therefore* $\preceq'$ *is defined via* $o_1 \preceq' o_2 \preceq' o_3 \preceq' e$*.*

5.7. Complexity

Our main goal is to find a procedure that updates the current state to the next and thus to update $\preceq$ to $\preceq'$ such that is done more efficiently than recomputing the scores (δ) of the elements in $\mathcal{O}$.

As stated already in Theorem 5.6.1, the time complexity of the naïve algorithm, ι^{naive}_{Θ}, is $O(n * m)$ and $\tilde{\Lambda}^{naive}_{\Theta}$ has time complexity $O(n + (n * m))$. The space complexity for storing a state in $\mathcal{S}_{naive}$ is $O(n * m)$.

We first investigate the time complexity of the online plurality aggregator following the algorithm presented in Algorithm 8.

Theorem 5.7.1 $\iota^{online}_{\Theta_p}$ *has time complexity* $O(m)$*. The space complexity for storing a state in* $\mathcal{S}_p$ *is* $O(m * n)$*.*

Proof:(sketch)
We assume that the position (winner or loser) of each element in each (not updated) preference relation is obtained in one step ($O(1)$), since this value is given in $pos(x)$ ($x \in \mathcal{X}$). Given that the preference relations are total preorders (see Section 5.3), we only have to check the value of one of the winners, and therefore we avoid the conduction of multiple comparisons.

For each preference relation $\preceq_i, i = 1, \ldots, m$, we compare e with the current winner x, i. e. if $x \preceq e$; this step can be obtained in one step ($O(1)$). At each comparison, we update the state values of e if it gets to be the winner. Since this comparison is done m-times (for all preference orders), this operation has time complexity $O(m)$.

Once all m-comparisons are done, the plurality scores of all elements are also updated, and the new aggregated relation is built. Consequently, the time complexity of the online plurality algorithm is $O(m)$.■

We turn our investigation to the online Borda count aggregator.

Theorem 5.7.2 $\iota^{online}_{\Theta_b}$ *has time complexity* $O(m * logn)$*. The space complexity for storing a state in* $\mathcal{S}_p$ *is* $O(m * n)$*.*

Proof:(sketch)
We need to insert the new element e in the right position of the preference relations $\preceq_i, i =$

$\{1, \ldots, m\}$. We assume that the position of each element in each (not updated) preference relation is obtained in one step ($O(1)$), since this value is given in $pos(x)$ ($x \in \mathcal{X}$).

Again, since the preference relations are total preorders (see Section 5.3), whenever we compare the element e with elements placed in the same position in $\preceq_i$, we only have to check the value of one of the elements in this position, and therefore we can avoid conducting multiple comparisons.

In order to find the position of the new element e in a preference relation $\preceq_i$, e is first compared with the element, x, placed in the middle in $\preceq_i$. If e is less preferred than x, $e \preceq x$, it is then compared with the element in the middle from the ones that are least preferred than x in $\preceq_i$. If e is more preferred than x, it is compared with an element in the middle from the ones that are more preferred than x in $\preceq_i$. This process continues until we find the position where the new element should be inserted. The search algorithm can be seen as recurrences of dividing n in half with a comparison. The time complexity of this operation is $O(logn)$.

Since we have to find the position of the new element e in each preference relation $\preceq_i, i = 1, \ldots, m$, the time complexity of the online Borda algorithm is $O(m * logn)$.■

At last, we investigate the time complexity of the online sequential pairwise aggregator.

Theorem 5.7.3 *$\iota^{online}_{\Theta_s}$ has time complexity $O(m)$. The space complexity for storing a state in $\mathcal{S}_p$ is $O(m * n)$.*

Proof: (sketch)
Following the Algorithm 11, the new element e is compared with only the last element of the agenda, x, in each preference relation $\preceq_i, i = 1, \ldots, m$. We assume that the position of each element in each (not updated) preference relation is obtained in one step ($O(1)$), since this value is given in $pos(x)$ ($x \in \mathcal{X}$).

The comparisons are done for all preference orders. After the m-comparisons, the new aggregated relation is built. Therefore, the time complexity of the online sequential pairwise algorithm is $O(m)$.■

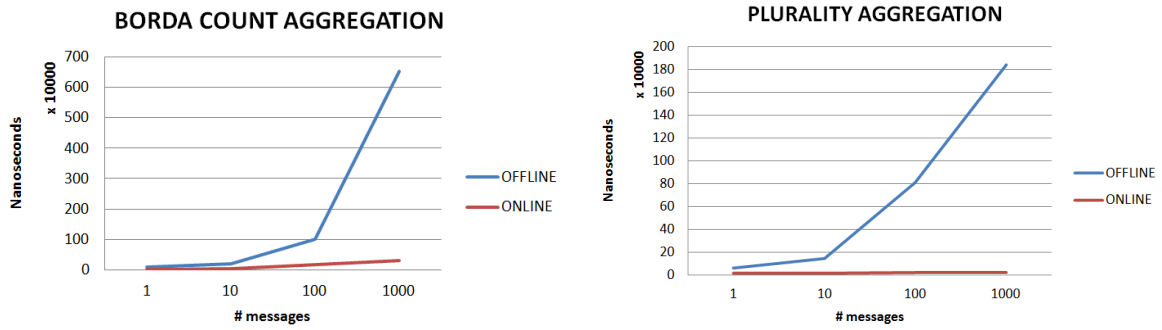
To finish our analysis, and based on the results from Section 5.5, we can state the following proposition:

Proposition 5.7.1 *If Θ satisfies IIA, then $\Lambda^{online}_{\Theta}$ has complexity time $O(m)$ (see Prop 5.5.1).*

Proposition 5.7.2 *If Θ satisfies MON, and the proposition 5.5.2 holds, then $\Lambda^{online}_{\Theta}$ has complexity $O(m)$. If Θ satisfies MON, and proposition 5.5.3 holds, then $\Lambda^{online}_{\Theta}$ has complexity $O(m)$.*

These propositions state that the online preference aggregator only has to compute the state of the new element since e, and thus with time complexity $O(m)$, for the following cases:

1. if Θ satisfies IIA then the ordering of the elements in $\preceq$ remains unchanged in $\preceq'$ after the insertion of e,
2. if Θ satisfies MON and and e is placed in the update profile such that all elements are more preferred than e in every preference relation, then the ordering of the elements in $\preceq$ remains unchanged in $\preceq'$,
3. if Θ satisfies MON and e is placed in the update profile such that all elements are less preferred than e in every preference relation, then the ordering of the elements is at least as good in $\preceq'$ as it was in $\preceq$.



(a) Borda Count Aggregation

(b) Plurality Aggregation

Figure 5.2.: Borda and Plurality Aggregation Methods: Execution time comparison between the standard versus the online approaches

5.8. Evaluation

In this section, we evaluate the plurality and Borda online voting aggregation methods. We conducted a feasibility study to demonstrate the execution time of the proposed algorithms and compare them with the execution time of the standard approaches (baseline). The prototype of the optimized online aggregation algorithms (`Online Voting Systems`) has been implemented in Java 1.7.

5.8.1. Data

We have performed our experiments using a four datasets (of different sizes) which have been created by randomly generating messages and their provenance values. We randomly assigned timestamp values (ranging from anytime between Sep. 1st 2016 to Sept. 3rd 2016), degrees of trustworthiness (ranging from 1 to 10), and degree of popularity (ranging from 1 to 10) to the messages. For this experiment we use a machine with 4GB RAM and a single Intel(R) Core(TM) CPU core with 1.70GHz.

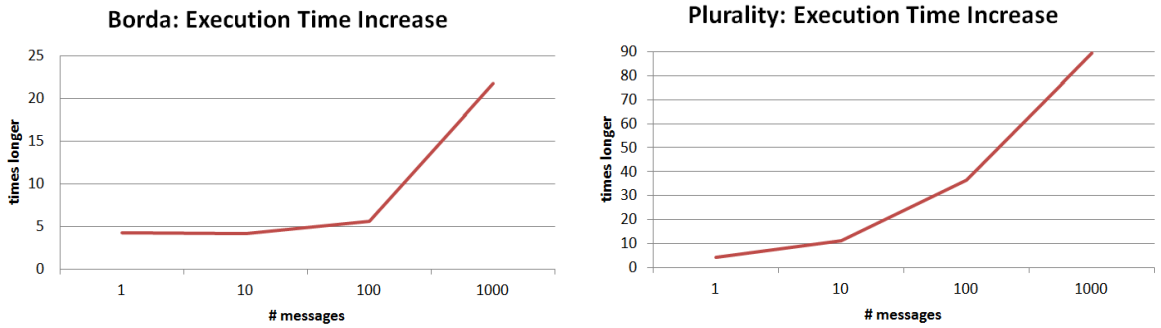
5.8.2. Methodology

We compare our optimized algorithms to the naïve (standard) approaches. We evaluate how effective they behave when preference orders are updated in the light of a newly-introduced element. We want to demonstrate that our approach is fast enough to support real time applications.

5.8.3. Discussion and Results

We consider four different datasets containing 1, 10, 100, 1000 messages respectively. For each setting, we have measured the time needed to re-compute the message rankings based on the provenance dimensions after adding a new message to the system. We conducted 10 runs and we consider only the average runtime.

Figure 5.2(a) and Figure 5.2(b) show the execution time of the Borda (standard vs. online) and plurality (standard vs. online) approaches. The x-axis represents the different messages



(a) Borda Count Aggregation

(b) Plurality Aggregation

Figure 5.3.: The experiment confirms that the execution time of Borda aggregation decreases from polynomial to linearithm time, and the execution time of plurality aggregation is linear.

considered in the system before updating the preference orders, and the algorithm runtime in nanoseconds is placed on the y-axis. As expected, the execution time of the standard approaches increases as the number of messages in the systems increases. As stated already in Proposition 5.6.1, the expected time complexity of the naïve algorithm is $O(n * m)$ while the plurality online approach is $O(m)$ and Borda online approach is $O(m * logn)$. Note that for the online plurality, the execution time almost does not change in all settings. Therefore, the incontestably lower execution time of the online plurality makes it feasible to be used in real use case (streaming) scenarios.

Again, as expected, the experiment confirms that the execution time of Borda aggregation decreases from polynomial to linearithm time. For instance, Figure 5.3(a) and Figure 5.3(b) show the execution time increase order of the Borda (standard vs. online) and plurality (standard vs. online) approaches respectively. Figure 5.3(a) shows that by re-computing the ranking when considering only 11 messages, the time execution of the standard algorithm is about 4 times longer, and with 111 messages, more than 5 times longer.

5.9. Related Work

Preference reasoning and preference aggregation are very active areas within artificial intelligence research, economics, and other fields. Changes on the set of elements, i.e., the successive elimination of candidates, is often used to make manipulation intractable to compute. The work [Davies et al., 2012] studies the relation between eliminating candidates and the computational complexity of manipulation. For many voting rules, such as the elimination version of Borda and Plurality, adding elimination rounds can increase the computational complexity. Although manipulation with successive elimination of candidates also deals with dynamics on the domain, this setting considers flexible preference relations (the preference relation defined by the manipulator), which is not the case for the online setting.

Some work deals with changes on the preference orderings instead of domain changes. In [Thimm, 2013], Thimm considers updates in settings for preference aggregation under preference changes. Thimm investigates how an aggregated preference order has to be updated

when the input orders are dynamic and shows that even for some simple aggregation rules, i.e., for the plurality and the Borda rule, the dynamic setting can be handled more efficiently than recomputing the aggregated preference order from scratch. The work of [Maudet et al., 2012] deals with influence of one agent's preferences to other agents' preferences. Preference relations might be changed through influence. They are mainly interested in computational properties in this framework. The work [Can and Storcken, 2013] explicitly considers updates to input preference orders and their influence to aggregation. The author introduces the property of update monotonicity for (static) preference aggregators as a novel property to assess the quality of an aggregator. An aggregator satisfies update monotonicity if under changes of one input preference order towards the aggregated order, the aggregation does not change. Therefore, [Can and Storcken, 2013] do not consider general updates and computational approaches.

Furthermore, there are approaches in belief revision that deal with dynamics of epistemic states given in form of preorders. For example, the work [Booth et al., 2006] considers iterated belief revision based on enriched preference states. There, a preference state is basically a preference order on possible worlds that is revised upon newly received evidence. The work [Chomicki and Song, 2005] deals with revising a given preference relation with another (partial) one such that the former is modified in a minimal way to incorporate the latter. Although these works also deal with issues related to temporal evolution of preference orders, they do not address the expansion of the set of elements in the setting.

In the realm of the Semantic Web, we compare our work to annotated RDF data in general, and to aggregation principles in particular. Based on semirings [Green et al., 2007], Buneman and Kostylev [Buneman and Kostylev, 2010] and Straccia et al. [Straccia et al., 2010] present an algebra for RDF provenances. Their approaches are for provenances in general, but they do not consider multiple dimensions simultaneously. Zimmermann et al. [Zimmermann et al., 2012] present a combination of multiple provenance dimensions. They combine two dimensions by a composition into one dimension, modeled as a compounded provenance dimension. An aggregation function maps provenance dimensions into a set of pairs of provenances. Kostylev et al. [Kostylev and Buneman, 2012] also consider the problem of combining various forms of provenance that, analogous to the previous discussion, map provenance dimensions into a set of vectors. They introduce restrictions to semirings in order to define containment and equivalence of combined provenance relations. The latter ones are different from our work since we aggregate provenance dimensions considering aggregation functions, which do not rely on the structure of the provenance dimensions and can be generalized to the aggregation of every ordered set.

Querying the Semantic Web with preferences is also considered in [Gueroussova et al., 2013, Siberski et al., 2006]. In [Gueroussova et al., 2013], Gueroussova et al. propose an extension to SPARQL 1.1 that supports the expression of preferred query results. Their language builds on established work on SQL preferences and on an earlier effort by Siberski et al. [Siberski et al., 2006], Gueroussova et al. show that preference queries can be directly expressed in both SPARQL1.0 and SPARQL1.1 using OPTIONAL queries or novel features of SPARQL1.1, such as NOT EXISTS. The authors have not implemented any aggregation but they illustrate such a realization with respect to skyline and conditional preference queries.

More related to the online setting are approaches for preference querying over data stream. Continuous top-k querying [Mouratidis et al., 2006] uses a preference function to determine the best k answers to a (relational) query. Continuous skyline querying approaches [Sarkas et al., 2008] consider the dominance relationship between tuples where tuples are represented

as points in a multi-dimensional space. These works do not focus on the aggregation, but are mainly interested in exploring different data structures to improve performance and decrease memory consumption of preference querying. Furthermore, ranking in streaming data is an emerging problem when querying large data collections as on the Web, and it is therefore considered for RDF querying and reasoning.

For instance, SPARQL extensions allow ranking query results on streams [Barbieri et al., 2010a, Bolles et al., 2008], as well as general reasoning frameworks incorporate the management of streaming data [Barbieri et al., 2010b]. However, ranking with aggregation of multiple provenance dimensions is not studied in streaming RDF data so far.

From the ranking perspective, several approaches consider also the problem of combining several dimensions of ranking criteria [Bruno et al., 2002, Kießling, 2002, Xin et al., 2006, Hristidis et al., 2001]. Preferences are specified in terms of partial orders on attributes and they can be accumulated and aggregated to build complex preference orders. In [Bruno et al., 2002], the general idea is to rank query results when there is no exact match, but some results are similar to the query. They compute the distance of attribute values of the relation with respect to the query attributes. In [Hristidis et al., 2001], linear sums of attributes are used to rank preferences (assigned to attributes). Likewise, Li et al. [Li et al., 2005] present top-k ranking for different dimensions for relational databases. Compared to our work, none of them considers the ranking of semi-structured data like RDF and their focus is not on ranking with respect to provenances of data.

5.10. Findings and Research Contribution

In this chapter, we have proposed three algorithms that provide the most relevant news feeds according to the user's preferences. Our algorithms rank messages 'on the fly' as the message passes through the system.

We have introduced the concept of online preference aggregators and investigated the consequences of adding a new element to the preference orders. We have studied the relationships between the original aggregated preference order and the updated aggregated preference order, and established a framework for computational approaches to online preference aggregation. Concrete online aggregation algorithms and complexity analysis have been presented for the plurality, Borda count, and sequential pairwise voting methods. Our complexity analysis has shown that the online aggregator performs better than the original aggregators after the domain changes. The computation of the original aggregators has time complexity $O(n * m)$, and we have shown that the online plurality aggregator has time complexity $O(m)$, the online Borda count aggregator $O(m * logn)$, and the sequential pairwise aggregator $O(m)$.

We conducted a feasibility study to demonstrate the execution time of the proposed algorithms. Our execution experiments show that our approach runs extremely faster than the standard ones and therefore are more suitable to be used in real world applications.

6. Managing Data Changes in Linked Open Data Sources

Overview

The Linked Open Data (LOD) cloud changes frequently. Quite often, LOD applications prefetch data from the Web and store local copies of it in a cache for faster access at runtime. This chapter presents two techniques for dealing with data dynamics in the LOD sources using provenance information. The first one describes an investigation of availability and conformance of provenance information for detecting changes of LOD sources. Given the HTTP basis recommended in the Linked Data guidelines, the natural way of detecting changes would be to use HTTP header information, such as the *Last-Modified* field as it describes when the resource has been changed last. For LOD applications that operate on local caches of Linked Data, a simple check on this provenance could support the decision process of determining which sources need to be updated. The second part of this chapter formalizes a strategy for capturing data changes also based on a time-dependence measure that captures the frequency, degree, and regularity of the changes of the LOD sources.

Structure

The structure of this chapter is as follows: Section 6.3.1 presents a motivation scenario that emphasizes the importance of knowledge about data changes and especially about the change behavior of a LOD data source over time for the purpose of data caching. Then, this chapter describes two different techniques to capture data changes of the LOD sources: (1) Section 6.3 describes an investigation of availability and conformance of provenance information for detecting changes of LOD sources, (2) Section 6.4 formalizes and evaluates a strategy for capturing data changes based on time-dependence a measure that captures the frequency, degree, and regularity of the changes of the LOD sources. The structure of Section 6.3 and Section 6.4 is described in the subsequent sections.

6.1. Introduction

The Linked Open Data (LOD) cloud is a global information space which structurally represents and connects data items. The LOD principles provide a flexible publishing paradigm to integrate and interlink any kind of data from arbitrary datasets published by various data providers (or data sources) on the Web. From the time the Linked Open Data (LOD) principles have been created until now, the LOD cloud has grown significantly.

Recent investigations [Schmachtenberg et al., 2014, Auer et al., 2012, Hausenblas et al., 2009, Alexander and Hausenblas, 2009, Dividino et al., 2014a, Dividino et al., 2013, Hartig, 2011, Umbrich et al., 2012, Dehghanzadeh et al., 2014] have shown that data published and interlinked on the LOD cloud is subject to frequent changes. As the data in the cloud changes, these caches or indices no longer reflect the current state of the data anymore and need to be updated. Käfer et al. [Käfer et al., 2013] observed a subset of the LOD cloud over a period of 29 weeks and concluded (among other things) that the data of 49.1% of the LOD sources changes. Likewise, Gottron et al. [Gottron and Gottron, 2014] observed LOD data over a period of 77 weeks and described that the accuracy of indices built over the LOD sources drops by 50% after only 10 weeks. These outcomes indicate that almost half of the LOD sources are not appropriate for long-term caching or indexing. Unquestionably, data on the LOD cloud changes and knowledge about these changes, i.e., about the change behavior of a dataset over time, is important as it affects various different LOD applications e., g. data caching [Umbrich et al., 2012], indexing of distributed data sources [Konrath et al., 2012], searching in large graph databases [Gottron et al., 2013b], optimizing the execution of queries [Neumann and Moerkotte, 2011] and recommending appropriate vocabularies to Linked Data engineers [Schaible et al., 2013].

In particularly, the distributed, Web-based nature of the data motivates many applications to keep local copies of the data. Mainly, data is fetched live from the Web only in those cases where the data is missing or known to be highly dynamic [Umbrich et al., 2010, Umbrich et al., 2012]. However, local copies of LOD data sources still need to be updated from time to time. Thus, the question is when to perform such an update.

LOD applications that operate on local caches of Linked Data need to be aware of these changes. In this way they can update their cache to ensure operating on the most recent version of the data. However, due to limitations of the available computational resources (e. g., network bandwidth for fetching data, and computation time) LOD applications may not be able to permanently visit all of the LOD sources at brief intervals in order to check for changes. With the increasing amount of data available on the Cloud, LOD applications encounter challenging questions when conducting updates of local caches of LOD sources.

In this chapter, we exploit two different techniques to capture data changes of the LOD sources:

Provenance Information for Detecting Changes of Linked Open Data Sources The first part of this chapter discusses the use of provenance within LOD resources. Provenance information about the owner, creation date, and modification author can be used to verify (to a certain degree), the quality and the level of trustworthiness of such information. Additionally, modification date can denote when the resource has been changed last. A simple check on this provenance can support the decision process of determining which sources need to be updated.

Given the HTTP basis recommended in the Linked Data guidelines, the most intuitive way of detecting changes would be to to exploit the HTTP header information provided on the Web. According to the Linked Data guidelines, resources should be modeled using dereferenceable HTTP Uniform Resource Identifiers (URIs). Whenever a client application invokes an HTTP request to a server (i. e. an SPARQL endpoint) for a particular URI on the LOD cloud, the server should respond by providing useful information about the resource represented by this URI. Naturally, this response will make use of the HTTP protocol itself. The HTTP header of this response can contain provenance information about the resource (e.g. owner, creation date, etc.) [Fielding et al., 1999]. Provenance information about the owner, creation date, and modification data can be used to verify (to a certain degree), the quality and the level of trustworthiness of such information. Additionally, there is a field which can denote when the resource behind this URI has been changed last. In combination with an HTTP header response, this *Last-Modified* field is intended for probing a resource for whether or not it has been changed since its inclusion in the cache of a Web or Linked Data application.

Nevertheless, even with the existing W3C specifications which define rules and conditions to be followed by the LOD servers, the information contained in the HTTP headers may in practice be inaccurate or wrong [Rula et al., 2012]. Therefore, applications relying on such information are susceptible to drawing wrong conclusions. In this work, we empirically evaluate the conformance of time-related HTTP header metadata information on the LOD cloud. In particularly, we check for the conformance of the *Last-Modified* field. Knowledge about the reliability of this field is important for applications which intend to make use of it.

To this end, in this chapter we analyze a large-scale dataset that is obtained from the LOD cloud by weekly crawls from the period between May 2012 and January 2014. The dataset contains 84 snapshots. For each pair of subsequent snapshots, we check for changes in the data and compare the observations to the information provided by the *Last-Modified* HTTP header field. Using the results of our experiments, we discuss the benefits of the availability and conformance of the HTTP header fields in real-world scenarios.

Capturing the Dynamics of Linked Open Data Sources LOD applications relying on data from the LOD cloud need to cope with constant data updates to be able to guarantee a certain level of quality of service. In an ideal setting, a cache or an index is kept up-to-date by continuously visiting all data sources, fetching the most recent version of the data and synchronizing the local copies with it. However, in real-world scenarios, LOD applications must deal with limitations of the available computational resources (e.g., network bandwidth, computation time) when fetching data from the LOD cloud. These limitations imply the necessity to prioritize which data sources should be first considered for retrieving their data. In order to make best use of the resources available, it is vital to choose a good scheduling strategy for updating local copies of the LOD data sources. While there exists research on the freshness analysis of the cached data for answering SPARQL queries in a hybrid approach such as Umbrich et al [Umbrich et al., 2012], to the best of our knowledge, there is no work addressing strategies to efficiently keep local copies of LOD sources up-to-date.

Intuitively, a strategy dedicated to update data caches built out of data from the LOD cloud would make use of the HTTP protocol. The *Last-Modified* HTTP header field denotes when a LOD source behind this URI has been changed last. However, in a previous chapter (Section 6.3.3), we showed that a very few LOD sources (on average only 8%) provide correct update values. Consequently, applications relying on such information are susceptible to draw the wrong conclusions. Thus, this method is inappropriate for probing a LOD source for whether or not it has been changed since the last retrieval of its data. The only alternative is to actually retrieve the data from the sources and check it for changes.

Scheduling strategies aim for deriving an order for data sources and suggesting when they should be visited. Consequently, the application updates its local copy by fetching data from the data sources following this order. The simplest strategy is to visit the data sources in an arbitrary but fixed order, which guarantees that the local copy of every data source is updated after a constant interval of time. Alternative strategies explore the different features provided by the data sources, e.g., their size, to assign an importance score to each data source, and thus derive an order. An established scheduling strategy for the Web leverages the PageRank algorithm [Page et al., 1999], where a score of importance is given to each data source regarding its centrality in the link network with other data sources.

When considering a set of LOD sources, certainly some of them change more or less often than others [Käfer et al., 2013]. For example, it is not likely that in a short time interval every LOD source changes. Thus, many sources may provide the same information during this entire interval. Therefore, it is not necessary to fetch data from such sources. However, whenever data of a source changes, an update is required. Accordingly, some sources should be fetched at shorter or longer time-intervals. This implies that each LOD source could be given a different update importance, which is based on its change behavior. The change behaviour or dynamics of a dataset involves a notion of how *fluid* a dataset is, i. e. how it behaves and evolves over a certain period of time. In the context of this work, we understand a period of time to be a continuous time interval beginning at an initial point in time up to a final one. Due to this time-dependence a measure for dynamics should capture the frequency, degree, and regularity of the changes of the data. In this chapter, we present a formal notion of dynamics for LOD datasets. We define dynamics as an aggregation of changes, built on top of contemporary change metrics. We further extend this notion to incorporate the use of different decay functions for stressing or weakening periods within a time interval.

We evaluate the effectiveness of the dynamics function for conducting updates local copies of LOD sources, and of different scheduling strategies for maintaining indices of web documents on a large-scale LOD dataset from the Dynamic Linked Data Observatory (DyLDO) [Käfer et al., 2013] that is obtained via 149 weekly crawls in the period from May 2012 until March 2015. We investigate two different setups: (i) in the *single step* setup we evaluate the quality of update strategies for a single and isolated update of a local data cache, while (ii) the *iterative progression* setup involves measuring the quality of the local data cache when considering iterative updates over a longer period of time. Quality is measured in terms of precision and recall with respect to the gold standard, i.e., we check the correctness of data of the (updated) local copy with respect

to the data actually contained in the LOD cloud. We assume that only a certain bandwidth for fetching data from the cloud is available, and we investigate the effectiveness of each strategy for different bandwidths. Therefore, in the first setup, we can observe the relation between strategies and restrictions of bandwidth (i.e., if the strategies show comparatively uniform performance over all restrictions or if better or worse performance depends on a given restriction), and use such findings as parameters for the second setup. The second setup evaluates the behavior of the strategies in a realistic scenario (e.g., a LOD search engine updating its caches). Our evaluation indicates the most effective strategies for updating local copies of LOD sources, i.e., we demonstrate for given restrictions of bandwidth, which strategy performs better in terms of data accuracy and freshness.

6.2. Motivation Scenario

As an example, suppose we are maintaining local copies of data which comes from three different data sources: *dbpedia.org*, *bbc.co.uk*, and *musicbrainz.com*.

Suppose our local copy has been last updated on May 8th, 2015, and we want to again update our local copy on June 10th, 2015. Usually, we would update our cache by visiting all three data sources, fetching the most recent version of the data and synchronizing the local copies with it.

However, in real-world scenarios LOD applications must deal with limitations of the available computational resources (e.g., network bandwidth, and computation time). With such limitations in mind, we suppose our network bandwidth enables only $K = 12,000$ triples to be fetched per time slice.

Under such constraints, suppose we cannot fetch data from all the data sources. These limitations imply the necessity to prioritize which data sources should be first considered for retrieving their data. Ideally, we want to update first the data sources whose data has definitively changed since the last update. Therefore, we need a strategy which enable us to identify these resources, and thus to conduct such updates efficiently.

In the next section we introduce two different techniques for updates strategies coupled with the following-up events of our motivation scenario.

6.3. An Investigation of Provenance Information for Detecting Changes of Linked Open Data Sources

Overview

Data on the Linked Open Data (LOD) cloud changes frequently. LOD applications that operate on local caches of Linked Data need to be aware of these changes. In this way they can update their cache to ensure operating on the most recent version of the data. Given the HTTP basis recommended in the Linked Data guidelines, the natural way of detecting changes would be to use HTTP header information, such as the *Last-Modified* field. The *Last-Modified* field is provenance-like information which is intended for probing a resource to determine whether or not it has been changed since its inclusion in the cache of a Web or Linked Data application. However, it is uncertain to which degree this field is currently supported on the LOD cloud and how reliable the provided information is. In this chapter, we check for the availability and conforming use of the time information in the HTTP header of Linked Data sources. We analyze a large-scale dataset obtained from the LOD cloud by weekly crawls over almost two years. In these weekly snapshots, we check if the HTTP header field *Last-Modified* is available and if the date provided for the last modification aligns with the observed changes in the data itself. Finally, we discuss the benefits of the availability and conformance of the HTTP header fields in a real-world scenario.

Structure

The structure of Section 6.3 is a follows: Section 6.3.1 presents a motivation scenario that emphasizes the need for and use of provenance for the purpose of data caching. Section 6.3.2 introduces provenance information included in the HTTP header of an HTTP response typically obtained when dereferencing or retrieving resources from the LOD cloud. We evaluate the conformance of LOD data source to provide a valid and correct *Last-Modified* HTTP header field, which indicates the date and time at which the resource was last modified in Section 6.3.3. Section 6.3.4 reviews existing works for providing and using provenance in the LOD cloud. Section 6.3.5 then concludes and summarizes the most indicative experiment results.

Research Questions

This chapter addresses the following research question:

RQ 2.II *Can the provenance delivered by Linked Data servers support applications for detecting changes of the resources in a LOD source?*

As Web data evolves over time, that is, RDF Graphs change as information is added and removed, local copies of Web sources need to be updated from time to time to ensure the quality or freshness of such copies.

Mainly, the naïve approach for detecting changes of a resource in a LOD source consists of downloading two arbitrary-size RDF descriptions of that resource and further comparing them [Käfer et al., 2013, Völkel and Groza, 2006, Zeginis et al., 2011]. As these descriptions can be of significant size, Linked Data applications would benefit if Linked Data servers could provide provenance information. A simple check on this provenance could support our decision process of determining which sources need to be updated.

However, it is crucial that Linked Data servers provide correct and valid provenance values; otherwise, it has no use in any practical application.

6.3.1. Motivation Scenario

Following up our motivation scenario described in Section 6.2, as a matter of example, we illustrate a request done by a browser (or the user behind the browser) of the RDF description representing the state of California to the DBpedia server [1]. As presented in Chapter 2.2.4, one of the fundamental LOD principles is that every resource (representing an entity in the real world) is uniquely identified and referenced by Universal Resource Identifiers (URI). A Semantic Web application or a browser uses mechanisms of the HTTP protocol to look up information upon those URI (called dereferencing).

First, the user sends an HTTP request for `http://dbpedia.org/resource/California` to the server. The HTTP request includes a textual header that specifies what format (here, the RDF notation) in which the user wants to receive the description of the resource, and should look like this:

Example 6.3.1 (Scrap of a HTTP request for `http://dbpedia.org/resource/California`)

GET/ resource/California HTTP/1.1 Host:http://dbpedia.org
Accept: text/html;q=0.5, application/rdf+xml
...

The 'Accept' header indicates that the user accepts either HTML or RDF, but prefers RDF. This preference is indicated by the quality value q=0.5 for HTML. This process is called content negotiation.

As it's generally held to be good practice when publishing Linked Data, DBpedia uses different URIs to distinguish between the identity of the resource itself and its descriptions (which can come in different formats). Therefore, in a response to the GET requested as shown in Example 6.3.1 we get a redirect response which tells the user that the description of the requested resource, in the requested format, can be found at other URI `http://dbpedia.org/data/California.rdf`.

Example 6.3.2 (Scrap of a HTTP redirect response to `http://dbpedia.org/data/California`)

HTTP/1.1 303 See Other
Location: http://dbpedia.org/data/California.rdf
Vary: Accept
...

Next, the user has to make a new GET request for looking up the new URI delivered by the server. Afterwards, the server then selects the best match from its file system or generates the desired content on demand and sends it back to the client. The answer for the GET request could be for example:

[1] dbpedia.org is a community effort to extract structured information from periodic Wikipedia dumps and to make this information available on the Web. It is served to the public via a live instance of OpenLink Virtuoso, and also offers Faceted Browsing and Exploration

Example 6.3.3 (Scrap of a HTTP response to `http://dbpedia.org/data/California`)

HTTP/1.1. 200OK Content-Type rdf/xml
Content-Language: en
Content-Location: http://dbpedia.org/data/California.rdf
...

This response header tells the user that the response contains the representation of the information resource in the requested RDF/XML format, and the rest of the message contains its representation in RDF/XML notation. This description can be of significant size, in this particular case it weighs about two megabytes (1,708KB).

The HTTP response header is composed of a set of fields, not all of which were illustrated in this example. Without a doubt, the HTTP response header delivers valuable provenance information about the availability, format, and location of the requested resources. In the following, we explore the header field *Last-Modified* which indicates when the requested resources have been modified last for the purpose of performing updates of local caches of the LOD sources.

6.3.2. Foundations: The HTTP Header

The LOD cloud is composed of various data servers which enable data access via the HTTP protocol. A client application invokes an HTTP request to a server by, for instance, sending a HTTP GET message for a particular URI. The Linked Data server responds via HTTP response message. Mainly, both HTTP request and response messages consist of headers (zero or more header fields), and possibly a message-body. A message-body is used to carry the entity-body, i. e. the description of the represented resource, ideally using RDF as data format.

The first line of a HTTP Response message is the Status-Line, consisting of the protocol version followed by a numeric status code and its associated textual phrase. The Status-Code element is a 3-digit integer result code of the attempt to understand and satisfy the request. The first digit of the Status-Code defines the class of response:

1xx Informational - Request received, continuing process

2xx Success - The action was successfully received, understood, and accepted

3xx Redirection - Further action must be taken in order to complete the request

4xx Client Error - The request contains bad syntax or cannot be fulfilled

5xx Server Error - The server failed to fulfill an apparently valid request

The individual values of the numeric status codes defined for HTTP/1.1, and an example set of corresponding associated textual phrase are: ”202” - Accepted, ”301”- Moved Permanently, ”400” - Bad Request, and ”505” - HTTP Version not supported.

The header fields allow the server to pass additional information about the response which cannot be placed in the Status-Line. These header fields give information about the server, provide further access to the resource identified by the Request-URI, and define metainformation about the entity-body. Some of the headers fields are:

1. *Content-Language* describes the natural language(s) of the intended audience for the enclosed entity, *Content-Length* indicates the size of the entity-body, in decimal number of OCTETs ,
2. *Content-Type* indicates the media type of the resource sent,
3. *Content-MD5* for the purpose of providing an end-to-end message integrity check of the entity-body,
4. *Date* represents the date and time at which the message was originated,
5. *Expires* gives the date/time after which the response is considered stale,
6. *Last-Modified* indicates the date and time at which the origin server believes the variant was last modified, and
7. *Server* contains information about the software used by the origin server to handle the request.

For more detailed information about HTTP messages, please refer to the Hypertext Transfer Protocol – HTTP/1.1 Specification [R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, T. Berners-Lee, 1999].

In this work, we focus on the analysis of the header field *Last-Modified*. This field is intended for a date when the requested objects have been modified last. In the context of Linked Data, this should correspond to the most recent date at which some part of the resources' RDF description has changed. Following the HTTP/1.1 specification [Fielding et al., 1999], the *Last-Modified* value must not be later than the time of the server's response message. In such cases, where the resource's last modification would indicate some time in the future, it is to be considered invalid. Furthermore, the server should obtain the *Last-Modified* value as close as possible to the time that it generates the *Date* value of its response. This allows a recipient to make an accurate assessment of the entity's modification time, especially if the entity changes near the time that the response is generated. Finally, HTTP/1.1 servers should send a *Last-Modified* value whenever feasible.

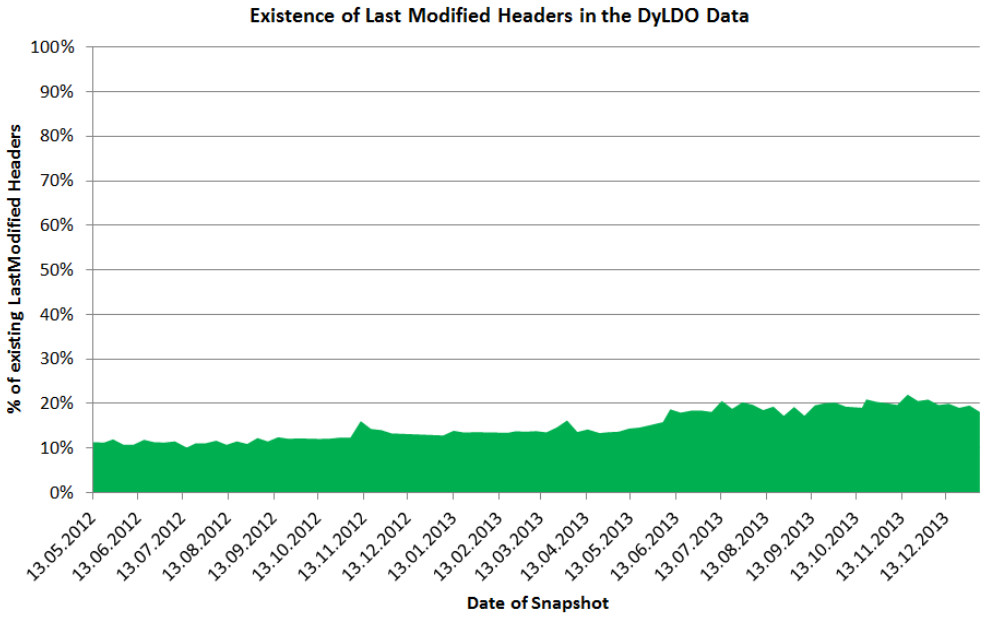
6.3.3. Evaluation of the Conformance of the Last-Modified HTTP Header Field

In this section, we evaluated the conformance of LOD data source to provide a valid and correct *Last-Modified* HTTP header field, which indicates the date and time at which the resource was last modified.

Data

We work with data from the Dynamic Linked Data Observatory (DyLDO) [Käfer et al., 2013]. The DyLDO dataset has been created to monitor a fixed set of Linked Data documents (and their neighborhood) on a weekly basis. For the sake of consistency, we use only the kernel seed documents of DyLDO. Our test dataset is composed of 84 snapshots corresponding to a period of almost two years (from May 2012 until January 2014). Furthermore, the DyLDO dataset contains (parts of) various well-known and large LOD sources, e.g., dbpedia.org, musicbrainz.com, and bbc.co.uk. For more detailed information about the DyLDO dataset, we refer the reader to [Käfer et al., 2013].

Figure 6.1.: Ratio of the Linked Data resources (in percentage) that provide the field *Last-Modified* in their header (in green).



Methodology

The main goal of our experiments is to measure the degree of how often the *Last-Modified* field in HTTP header of LOD resources is available and how often it is used correctly. We consider the use to be correct if this field returns a date and time which does not violate the observations of when the data for a resource has changed last, and therefore, to assure that the *Last-Modified* field from HTTP header could be used by client applications to verify if any change event has occurred for this resource, or even in the LOD cloud.

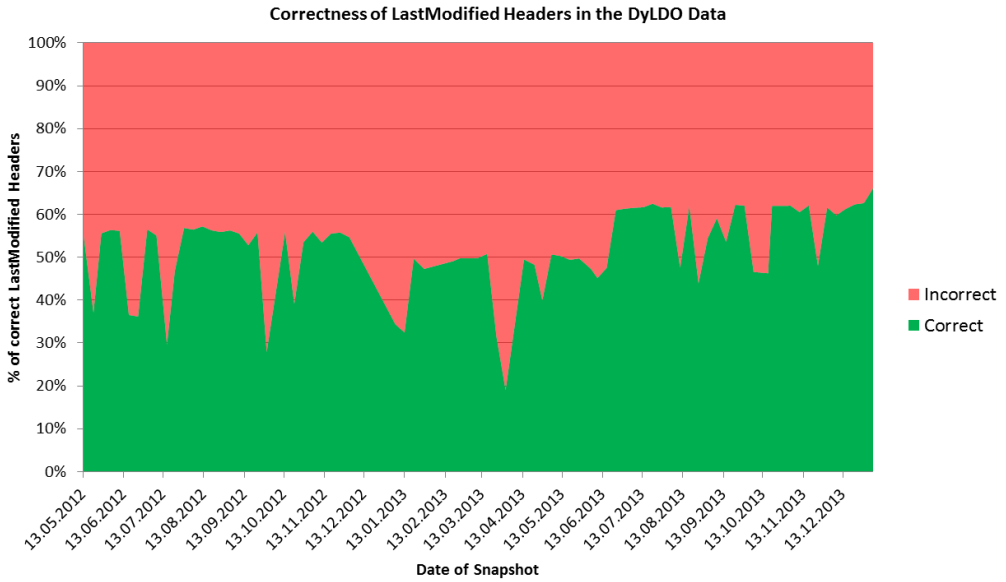
Discussion and Results

Each version of the DyLDO dataset consists of a set of RDF triples retrieved from different LOD sources. Furthermore, the data provides also information about the HTTP headers received when retrieving the data. From the 84 snapshots available in the dataset, we took each pair of subsequent snapshots from the same data source and computed the set difference over their set of triples. If we observe a difference we consider the data to have changed, otherwise we treat it as unchanged. This means if two snapshots have the same set of triples, they are equivalent and no changes occurred. If not, i. e., a deletion or addition of triples has occurred, we consider the data changed. A change should be reflected by a *Last-Modified* date which lies in the time range between the two snapshots of the data.

Figure 6.3.3 illustrates that on average only 15% of the resources actually do provide some value for the *Last-Modified* field in the HTTP header. Towards the end of the time period covered by the data set, we observe that the number of resources providing this value slightly grows.

Subsequently, we checked for those resources which provide a value for the *Last-Modified*

Figure 6.2.: Ratio of the resources (in percentage) providing a *Last-Modified* that is correct (in green), or incorrect or invalid (in red).



field, how many of them return a correct or an incorrect value (see Figure 6.3.3). As mentioned above, correct values are the ones where the last modified data aligns with actual changes in the RDF data. Incorrect values include (1) values that indicate changes but no change has been observed (2) values that indicate no changes but changes have been observed and (3) invalid values. Invalid values are the ones which indicate a time in the future relative to the time of the HTTP response or which indicate a time of last modification which actually precedes the time at which the resource was created. On average, we observe that only 52% of the resources which provide a value for the *Last-Modified* field provide also a correct value for it. The slight growth of both ratios in Figure 6.3.3 towards the end of the time period covered by the dataset is an artifact caused by data sources going offline, i.e. not responding or providing a 400 or 500 status code as response. It seems that more data sources providing no or wrong *Last-Modified* results went offline during the covered time span.

Our experiment shows that overall and on average only 8% of the resources in the datasets provide correct values for this field. This number is far too low to be of use for any practical application. It is, however, not clear why LOD sources do not provide valid information. We conjecture that some default configuration of LOD servers leads to this misbehavior.

6.3.4. Related Work

Similar to our study, Kjernsmo [Kjernsmo, 2015] investigates many of the HTTP caching implementations (i. e. LOD servers) and examine the availability and conformance of the HTTP headers that may allow caching. Even based on different datasets as in our study, the authors conclude that the headers do not reflect the standards compliant freshness lifetimes advertised by servers, and therefore they are not helpful for applications relying on caching.

Besides HTTP headers, there exist several vocabularies, methods, and tools that can be

used to describe or summarize LOD datasets on different levels. Basically, these levels are characterized by (1) information about the dataset (release, updates, access points, licensing), (2) statistical information about the data types and patterns in the dataset, and (3) information about the topic or domain described in the dataset.

Alexander et al. [Alexander and Hausenblas, 2009, Keith Alexander and Richard Cyganiak and Michael Hausenblas and Jun Zhao, 2011] designed the Vocabulary of Interlinked Datasets (VoID). Along with other vocabularies such as Dublin Core, FOAF, etc.), VoID contains a number of useful and recommended properties for providing basic metadata about RDF datasets. VoID is used to formally describe linked RDF datasets, and it allows us to discover datasets over which queries are distributed over. Mainly, the VoID intends to help users judge the appropriateness of the dataset for their purposes. Similar to the HTTP Response header *Last-modified* field, VoID provides the Dublin Core term *dcterms:modified* which specifies the date on which the dataset was changed. There are two reasons why we choose to investigate the conformance of the *Last-modified* field instead of the *dcterms:modified* : (1) there are numerous sources in today's LOD cloud that do not provide any VoID description and, when VoID description is available, it is often incomplete or represents only a part of the entire contents of a dataset [Jentzsch et al., 2011], (2) it is recommended that the *dcterms:modified* date should correspond to the HTTP 1.1 *Last-modified* date [Croome, 2000].

Another vocabulary for describing a dataset is the Data Catalog Vocabulary (DCAT) [Maali and Erickson, 2014]. DCAT is an RDF schema vocabulary designed to facilitate interoperability between datasets published on the Web.

In [Böhm et al., 2011] the authors propose metainformation generation algorithms which automatically generate VoID description for RDF datasets deduced by the resources in the dataset. The authors also introduced new ideas extending the current scope of VoID. make-void[2] is a different tool for generating VoID statistics for RDF files.

RDFStats [Langegger and Wöß, 2009] is a generator for statistics of RDF sources that combines several analyses and provides statistical information for datasets behind SPARQL endpoint and RDF documents. These statistics include different types of histograms, the number of anonymous subjects, the total number of instances for a given class, or a set of classes and methods to obtain the URIs of instances.

LODStats [Auer et al., 2012] provide several statistics about the usage of vocabularies, types and properties of LOD data sets. These statistics include number of triples, triples with blank nodes, labeled subjects, number of owl:sameAs links, class and property usage, class hierarchy depth, and cardinalities, among other things. These statistics are then represented using VoID and Data Cube Vocabulary [Cyganiak and Reynolds, 2014]. Data Cube vocabulary is focused purely on the publication of multi-dimensional data on the web. It is an RDF vocabulary for describing statistical datasets.

Roomba [Assaf et al., 2015b] automatically validats and generats descriptive metadata for RDF datasets. The extracted metadata are grouped into four categories: general (e.g., title, ID), access, ownership, and provenance (temporal and historical information).

Loupe [Mihindukulasooriya et al., 2015] extracts more detailed characteristics of classes and properties used in a dataset as well as common triple patterns. ABSTAT [Palmonari et al., 2015] provides a summary of the most commonly used abstract knowledge patterns.

The ExpLOD [Khatchadourian and Consens, 2010] adopts bisimulation and contraction to

[2]make-void: `https://github.com/cygri/make-void`

find common patterns in data. The tool is based on a mechanism that combines text labels assigned to the RDF graphs and bisimulation contractions. The bisimulation contractions are applied to subgraphs defined via queries, providing for summarization of arbitrary large or small graph neighborhoods. Also, ExpLOD can generate SPARQL queries from a summary.

ProLOD++ [Abedjan et al., 2014] is a web-based tool which includes statistics on frequencies and distributions of distinct subjects, properties, and objects, as well as statistics on data types and value pattern distributions of particular properties. It discovers positive and negative association rules and calculates the keyness measure for each property along the ontology class hierarchy. GraphLOD [Jentzsch et al., 2015], which is a set of new features of ProLOD++, allows exploring the graphical structures of Linked Datasets by visualizing the connected components and the graph patterns mined from them. It identifies graph patterns such as paths, cycles, stars, siamese stars, antennas, caterpillars, and lobsters.

SchemEX [Konrath et al., 2012] extracts schema information for large data sets, by considering the co-occurrence of types and properties, to construct a schema-based index structure. This index allows for answering schema-oriented queries. The SchemEx index structure is further used for a theoretic analysis on the interdependencies between explicit and implicit schema information of RDF data [Gottron et al., 2013a]. Beck et. al. integrate in LinDA the computation of a VoID description [Keith Alexander and Richard Cyganiak and Michael Hausenblas and Jun Zhao, 2011, Görlitz and Staab, 2011], SchemEX index [Konrath et al., 2012], the analysis on schema interdependencies [Gottron et al., 2013a], and the construction of a formal concept lattice of the data [Wille, 2009] in order to derive descriptive meta data.

Datasets' descriptions and statistics are important to make the dataset findable and usable, however, for many LOD datasets there is no good, complete and descriptive meta data available [Jentzsch et al., 2011, Assaf et al., 2015a].

6.3.5. Findings and Research Contributions

We evaluated the conformance of LOD data source to provide a valid and correct *Last-modified* HTTP header field, which indicates the date and time at which the resource was last modified. Our experiment shows that overall and on average only 8% of the resources in the datasets provide correct values for this field. This number is far too low to be of use for any practical application. It is, however, not clear why LOD sources do not provide valid information. We conjecture that some default configuration of LOD servers leads to this misbehavior.

Our analysis is restricted to the *Last-Modified* field, however, it could be easily extended to check the conformance verification of others HTTP header fields.

The reliable provision of provenance information in the context of the established HTTP protocol would be beneficial to the entire Web of Data. Many base technologies such as Linked Data caches and indexes may benefit from this information since simply checking on may help them to decide which sources need to be updated.

We believe that publishing correct the HTTP header information is a step towards quality-oriented data usage in the LOD cloud. Therefore, with this work we point out the dimensions of the problem of erroneous and missing information of the HTTP header for Linked Data. Thereby, we motivate LOD sources to publish correct and valid values to support application needs. For now, since we cannot rely on HTTP header information for change detection, we need ways to recognize and quantify changes in the LOD sources. In the next chapter we dicuss metrics to capture such changes.

6.4. Capturing the Dynamics of Linked Open Data Sources

Overview

Quite often, Linked Open Data (LOD) applications pre-fetch data from the Web and store local copies of it in a cache for faster access at runtime. Yet, recent investigations have shown that data published and interlinked on the LOD cloud is subject to frequent changes. As the data in the cloud changes, local copies of the data need to be updated. However, due to limitations of the available computational resources (e. g., network bandwidth for fetching data, and computation time) LOD applications may not be able to permanently visit all of the LOD sources at brief intervals in order to check for changes. These limitations suggest the need to prioritize which data sources should be considered first for retrieving their data and synchronizing the local copy with the original data. In order to make best use of the resources available, it is vital to choose a good scheduling strategy to know when to fetch data from which data source. In this chapter, we first present a general framework to analyze the dynamics of linked datasets within a given time interval. We propose a function to measure the dynamics of a LOD dataset, which is defined as the aggregation of absolute, infinitesimal changes. i. e. it captures the intensity of how the data evolved in this period. Later, we investigate the effectiveness of the dynamics function for conduction updates of LOD sources, against different strategies proposed in the literature and evaluate them on a large-scale LOD dataset that is obtained from the LOD cloud by weekly crawls over the course of three years. We investigate two different setups: (i) in the single step setup, we evaluate the quality of update strategies for a single and isolated update of a local data cache, while (ii) the iterative progression setup involves measuring the quality of the local data cache when considering iterative updates over a longer period of time. Our evaluation indicates the effectiveness of each strategy for updating local copies of LOD sources, i.e., we demonstrate for given limitations of bandwidth, the strategies' performance in terms of data accuracy and freshness. The evaluation shows that the measures capturing change behavior of LOD sources over time are most suitable for conducting updates.

Structure

The structure of Section 6.4 is a follows: We motivate capturing the change behavior of a fictional LOD source with a hypothetical scenario in Section 6.4.1. Section 6.4.2 describes the foundations of change metrics upon which our notion of dynamics is based. Section 6.4.3 presents a formal notion of dynamics for LOD datasets and its extension using different decay functions.Section 6.4.4 formalizes the need for and of scheduling strategies to efficiently keep LOD caches up-to-dated. Section 6.4.5 discusses the effectiveness of the dynamics function and a set of existing scheduling strategies for data updates strategies. We review some existing works on index/caches updates in Section 6.4.6, and conclude this chapter in Section 6.4.7.

Research Questions

This chapter addresses the following research question:

RQ 2.III *Does the consideration of changes within a time interval instead of between two points in time improve change analysis?*

State-of-the-art metrics for change detection of RDF datasets [Ding and Finin, 2006, Dividino et al., 2013, Käfer et al., 2013] mainly quantify changes between the two datasets.

When considering the change analysis of a dataset over a period of time, such state-of-the-art metrics can be used to compare any two versions of the same dataset each version being fetched at different points in time. Nevertheless, such metrics do not include a mechanism to exploit the dynamics of a dataset i. e. to consider a time interval described by more than two points in time.

RQ 2.IV *Which is the most adequate update scheduling strategy to manage caching copies of the LOD sources?*

Data on the Linked Open Data (LOD) cloud changes frequently and applications relying on that data (by pre-fetching data from the Web and storing local copies of it in a cache) need to continually update their caches. Instead of permanently visiting all of the LOD sources at brief intervals, a good scheduling strategy could help us to know when to fetch data from which data source in order to grab (most of) the changes.

So far, there is no work addressing strategies to efficiently keep local copies of LOD source up-to-date. Nevertheless, the effectiveness of well-known update scheduling strategies for maintaining indices of web documents, such as the PageRank algorithm [Page et al., 1999], and metrics for quantifying the changes of LOD sources could be evaluated in the context of updating local copies of LOD sources.

6.4.1. Motivation Scenario

Following up our motivation scenario described in Section 6.2, we illustrate the differences between change and dynamics analysis on LOD datasets. In this example, we describe three snapshots of a dataset captured at three distinct points in time: t_1, t_2, and t_3.

We are using the dbpedia and the FOAF vocabulary for describing headquarters and their broadcasters in cities located in the state of California. For instance, there are entities like dbr:Los Angeles and dbr:California that are connected via a dbo:isPartOf property. Table 6.1 summarizes the RDF triples published in the first snapshot of the dataset at time t_1.

At time t_2, the same data is visited again. Table 6.2 shows the RDF triples of this new snapshot. We can directly observe changes in the triples between these two snapshots. In the second snapshot, we observe four new RDF triples (see highlighted triples in Table 6.2). Table 6.3 summarizes the RDF triples of the third and last snapshot at time t_3. This snapshot contains the same set of triples as the first one.

Existing metrics proposed in the literature are able to quantify changes for every pair of snapshots of a dataset. For the sake of simplicity, we apply a very simple metric in this example, which only counts the additions, deletions and changes between the set of triples from the first and the second snapshot. In this case, there are four new triples over the total of all triples. The same number of changes is observed when comparing the second and third snapshot. However, since the first and the third snapshot contain the same set of triples, we cannot observe any changes under the considered metric. The direct comparison of the first and third snapshot of the dataset ignores the changes in the second snapshot.

In this work, we argue that the consideration of the changes in the second snapshot is of great importance to the analysis of the dataset dynamics in the time period ranging from t_1 to t_3. Otherwise, when ignoring this evolution the true dynamic character of the dataset is

Table 6.1.: Scenario: Example dataset at time t_1.

@prefix	dbo:	<http://dbpedia.org/ontology/> .
@prefix	dbp:	<http://dbpedia.org/property/> .
@prefix	dbr:	<http://dbpedia.org/resource/> .
@prefix	rdf:	<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix	foaf:	<http://xmlns.com/foaf/0.1/> .
dbr:California	rdf:type	dbo:Location.
dbr:California	foaf:homepage	"http://www.ca.gov/".
dbr:Los Angeles	dbo:isPartOf	dbr:California.
dbr:Los Angeles	dbp:headquarters	dbr:Fox Sports 1.
dbr:Fox Sports 1	dbo:broadcastNetwork	dbr:Fox Sports (United States).

Table 6.2.: Scenario: Example dataset at time t_2.

@prefix	dbo:	<http://dbpedia.org/ontology/> .
@prefix	dbp:	<http://dbpedia.org/property/> .
@prefix	dbr:	<http://dbpedia.org/resource/> .
@prefix	rdf:	<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix	foaf:	<http://xmlns.com/foaf/0.1/> .
dbr:California	rdf:type	dbo:Location.
dbr:California	foaf:homepage	"http://www.ca.gov/".
dbr:Los Angeles	dbo:isPartOf	dbr:California.
dbr:Los Angeles	dbp:headquarters	dbr:Fox Sports 1.
dbr:Fox Sports 1	dbo:broadcastNetwork	dbr:Fox Sports (United States).
dbr:Bluewarter	**dbo:isPartOf**	**dbr:California.**
dbr:Bluewarter	**dbp:headquarters**	**dbr:vpktv.**
dbr:vpktv	**dbo:broadcastNetwork**	**dbr:VPK(United States).**
dbr:VPK	**foaf:homepage**	**"http://www.vpktv.com/".**

Table 6.3.: Scenario: Example dataset at time t_3.

@prefix	dbo:	<http://dbpedia.org/ontology/> .
@prefix	dbp:	<http://dbpedia.org/property/> .
@prefix	dbr:	<http://dbpedia.org/resource/> .
@prefix	rdf:	<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix	foaf:	<http://xmlns.com/foaf/0.1/> .
dbr:California	rdf:type	dbo:Location.
dbr:California	foaf:homepage	"http://www.ca.gov/".
dbr:Los Angeles	dbo:isPartOf	dbr:California.
dbr:Los Angeles	dbp:headquarters	dbr:Fox Sports 1.
dbr:Fox Sports 1	dbo:broadcastNetwork	dbr:Fox Sports (United States).

neglected. In the following sections, we systematically introduce metrics of changes and present a formalization of how to incorporate them into our notion of dataset dynamics.

6.4.2. Foundations: Change Metrics

In this section, we introduce a formal specification of *dynamics* and the *dynamics function* for LOD datasets. In the literature, many change metrics have been proposed for analyzing RDF data of LOD [Käfer et al., 2013, Dividino et al., 2013, Ding and Finin, 2006, Gottron and Gottron, 2014]. These metrics essentially quantify the changes that occurred in a dataset by comparing two snapshots of this dataset. Our goal is to re-use such metrics and to incorporate them as a parameter in our framework for measuring dynamics of a LOD dataset.

We will denote a change metric as a function Δ. Basically, such a Δ-function is a metric that quantifies changes between two datasets, i. e. it is a function that determines the difference (or distance) between two datasets. Without loss of generality, in this chapter, we restrict Δ-metrics to determine the difference between two RDF datasets. For instance, changes between two datasets can be measured by the number of differences between the set of triples of these datasets (such as additions and deletions of RDF triples). Please note that our framework for measuring the datasets dynamics can be parametrized to make use of any existing change metrics that satisfies the formal requirements listed below.

Definition 6.4.1 *Let $\mathcal{S}$ be the set of all possible RDF datasets. A change metric is a function $\Delta : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ that maps two RDF datasets to a real number and satisfies the following conditions (for X_1, X_2 and X_3 being RDF datasets).*

1. *positivity:* $\Delta(X_1, X_2) \geq 0$
2. *symmetry:* $\Delta(X_1, X_2) = \Delta(X_2, X_1)$
3. *identity of indiscernibles:* $\Delta(X_1, X_1) = 0$ *and*
4. *triangle inequality:* $\Delta(X_1, X_3) \leq \Delta(X_1, X_2) + \Delta(X_2, X_3)$

Example 6.4.1 (Jaccard distance as change metric) *The Jaccard Distance $\Delta_{Jaccard}$ between two RDF sets satisfies the requirements of Definition 6.4.1. Let X_1 and X_2 be the two RDF datasets presented in Table 6.1 and Table 6.2, then the Jaccard distance between the set of triples of X_1 and X_2 evaluates to:*

$$\begin{aligned} \Delta_{Jaccard}(X_1, X_2) &= 1 - \frac{|X_1 \cap X_2|}{|X_1 \cup X_2|} \\ &= 1 - (5/9) \\ &= 0.44 \end{aligned} \tag{6.1}$$

6.4.3. Dynamics Analysis of Linked Open Data Sources

Dynamics Function

The dynamics function aims at quantifying the evolution of a dataset over a specific period of time and takes into consideration the changes occurring in this period.

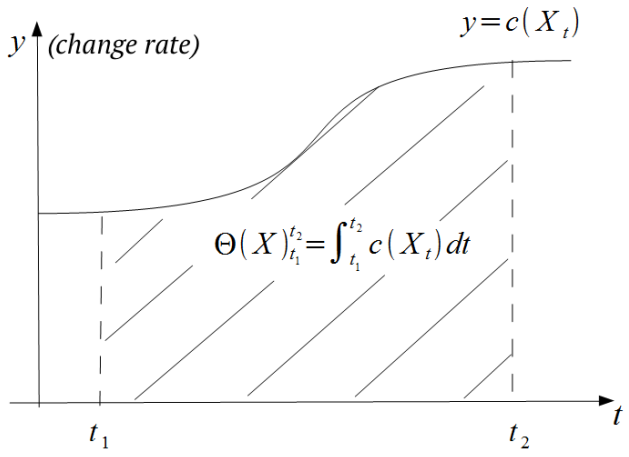


Figure 6.3.: The dynamics of a dataset is obtained by integration over its change rate over time.

For the sake of simplicity, we model time as a real value. We are looking for a function $\Theta : S \times \mathbb{R} \to \mathbb{R}$, which assigns to a dynamic RDF dataset $\mathcal{X}$ consisting of a concrete set of triples $X \in S$ at any point in time t a value which models the quantity of evolution it has undergone[3]. Θ is a monotone, non-negative function. This implies that there cannot be negative evolution. To measure the dynamics as the amount of evolution a dataset exhibited in a given time interval $[t_1, t_2]$, it is sufficient to compute $\Theta(X, t_2) - \Theta(X, t_1) \geq 0$. For ease of notation, we will in the following abbreviate X_t for the dataset X at time t and $\Theta(X, t)$ by $\Theta(X_t)$.

While it is difficult to define the function Θ directly to provide meaningful values, we will define it indirectly. To this end, we assume that the change rate of a dataset X_t at time t is given by a function $c(X_t)$. Then, we define the difference for two values of the function Θ to be obtained by accumulating the dataset change rate function over a time interval. This means, we integrate the change rate function of a dataset over a given period of time. More formally, the dynamics of a dataset is given by:

$$\Theta(X_{t_2}) - \Theta(X_{t_1}) = \int_{t_1}^{t_2} c(X_t) \mathrm{d}t. \tag{6.2}$$

The idea of integrating over a change rate function is depicted in Fig. 6.3 where the area under the curve $c(X_t)$ corresponds to a quantification of a dataset evolution and thus the dynamics of the dataset.

However, also the function c is not explicitly known, and cannot be used for the computation, i. e. it is not possible to determine the change rate of a dataset for a given point in time. Thus, our idea is to use an approximation for $c(X_t)$ based on discrete points in time and the changes between the datasets at these times.

So, we can effectively assume $\mathcal{X} = (X_{t_1}, X_{t_2}, \ldots, X_{t_n})$ to be a set snapshots of the RDF dataset X at points in time t_i, for $i = 1, \ldots n$. For any two snapshots of a dataset, we can measure changes using a Δ-metric, such as the ones presented in Section 6.4.2. Our assumption is that for small time intervals (ideally intervals tending towards zero) the change between

[3]This quantity of evolution is an abstract value but can serve for relative comparisons of datasets.

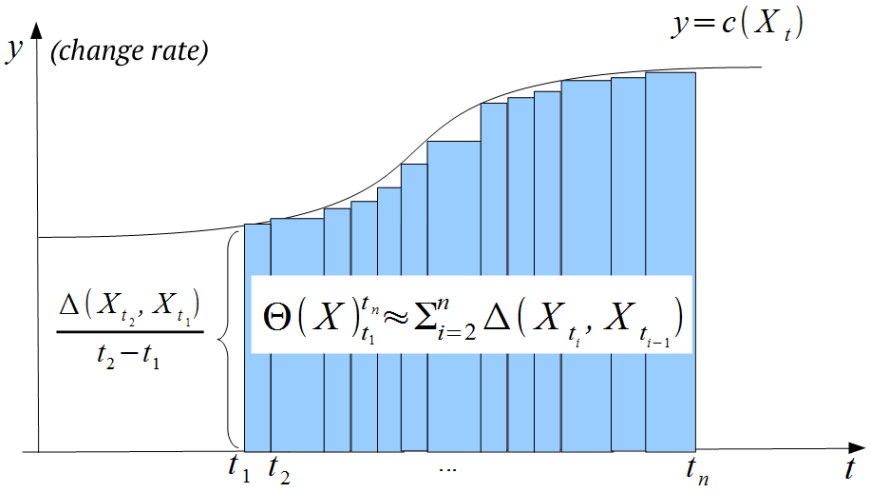


Figure 6.4.: Dataset dynamics defined as aggregation of absolute, infinitesimal Δ-metrics changes.

datasets is a good enough approximation of the change rate. This corresponds to the idea that:

$$\frac{\Delta(X_{t_i}, X_{t_{i-1}})}{t_i - t_{i-1}} \xrightarrow{t_{i-1} \to t_i} c(X_{t_i}) = \frac{\mathrm{d}}{\mathrm{d}t}\Theta(X_{t_i}) \tag{6.3}$$

Therefore, instead of using the change rate function c, we can use its approximation, namely the Δ-metric of (ideally) small time intervals of each pair X_{t_i} and $X_{t_{i-1}} \in \mathcal{X}$. This corresponds to approximating the change rate function using a step function as depicted in Fig. 6.4. Computing the integral given in Eq. 6.2 over this approximated function corresponds to:

$$\Theta(\mathcal{X})_{t_1}^{t_n} \approx \sum_{i=2}^{n} \Delta(X_{t_i}, X_{t_{i-1}}). \tag{6.4}$$

In the following example, we illustrate the computation of the dynamics of the dataset presented in our motivation scenario.

Example 6.4.2 (Computing dynamics based on a Jaccard distance change metric) *Let $\mathcal{X} = (X_{t_1}, X_{t_2}, X_{t_3})$ be a set of snapshots of a RDF dataset X at three distinct points in time t_1, t_2, and t_3 presented in Table 6.1, Table 6.2, and Table 6.3, respectively. Then the Jaccard distance between the set of triples from X_{t_1} and X_{t_2}, and from X_{t_2} and X_{t_3} are given as follows:*

$$\Delta_{Jaccard}(X_{t_2}, X_{t_1}) = 0.44, \quad \Delta_{Jaccard}(X_{t_3}, X_{t_2}) = 0.44 \tag{6.5}$$

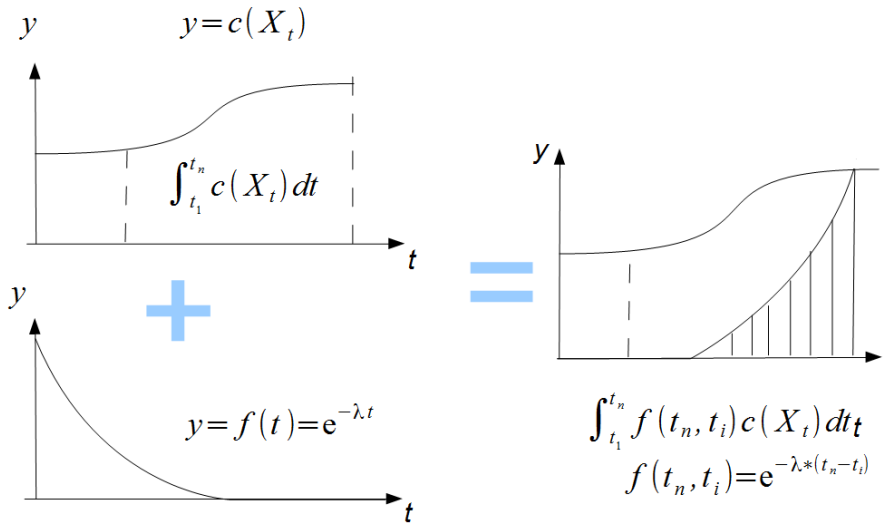


Figure 6.5.: Dynamics function with decay to strengthen the recent changes.

Then the dynamics of X is:

$$\begin{aligned}\Theta(\mathcal{X})_{t_1}^{t_n} &= \sum_{i=2}^{n} \Delta_{Jaccard}(X_{t_i}, X_{t_{i-1}}) \\ &= \Delta_{Jaccard}(X_{t_2}, X_{t_1}) + \Delta_{Jaccard}(X_{t_3}, X_{t_2}) \\ &= 0.88\end{aligned} \tag{6.6}$$

Decay Function

So far, we proposed a general framework in which we can compute the dynamics or evolution of any RDF dataset over a period of time and which incorporates any change metric Δ that follows the requirements mentioned above. Applications such as caching and index maintenance benefit from the analysis of the change history of datasets, since update strategies can incorporate the evolution of a dataset in their computation, instead of only the quantification of changes with respect to the last two snapshots.

However, for such applications changes tend to vary in importance as time passes, e. g. if a dataset used to change much but does not anymore, its index update strategy may need to be adapted (e.g., it should be less aggressive than it used to be). Therefore, it may be important that changes that took place a longer time ago are weakened and that recent ones are strengthened, or the other way around. For this purpose, the dynamics function should be flexible to incorporate such requirements.

Accordingly, we extend the dynamics function with a decay function $f(t)$. Fig 6.5 illustrates the influence of the decay function when combined with the dynamics function. In the upper left side of the figure, the dynamics function is shown. In the lower left side the decay function (in this example, the exponential decay function) is presented. In the right side, the combination of both is depicted. Please note that in this example we want to weaken older changes and strengthen the recent ones.

Based on these considerations, we introduce the following modifications to the definition

of dynamics: Let $\mathcal{X}$ be a dynamic RDF dataset and $c(X_t)$ be a function which measures the change rate of a dataset at time t, and $f(t)$ be a decay function. Then the decayed dynamics function of $\mathcal{X}$ is defined as:

$$\Theta_{decay}(X_{t_2}) - \Theta_{decay}(X_{t_1}) = \int_{t_1}^{t_2} f(t) \cdot c(X_t) \mathrm{d}t. \tag{6.7}$$

Consequently, the discretization is given by:

$$\Theta_{decay}(\mathcal{X})_{t_1}^{t_n} \approx \sum_{i=2}^{n} f(t_i) \Delta(X_{t_i}, X_{t_{i-1}}). \tag{6.8}$$

Example 6.4.3 (Dynamics involving a decay function) *We continue our Example 6.4.2 where we compute the Jaccard distances $\Delta_{Jaccard}(X_{t_2}, X_{t_1})$ and $\Delta_{Jaccard}(X_{t_3}, X_{t_2})$. We want to compute the dynamics of $\mathcal{X}$ using the dynamics with a decay function. In this example, we chose the exponential decay function $f(t_i) = e^{-\lambda \cdot (t_n - t_i)}$ to be our example decay function. For sake of simplicity, we set the parameter λ to 1. Then $\Theta_{decay}(\mathcal{X})$ is:*

$$\begin{aligned}
&\sum_{i=2}^{n} f(t_i) \Delta_{Jaccard}(X_{t_i}, X_{t_{i-1}}) \\
=&e^{-\lambda \cdot (t_3 - t_2)} \cdot \Delta_{Jaccard}(X_{t_2}, X_{t_1}) + e^{-\lambda \cdot (t_3 - t_3)} \cdot \Delta_{Jaccard}(X_{t_3}, X_{t_2}) \\
=&0.38 \cdot 0.44 + 1 \cdot 0.44 \\
=&0.607
\end{aligned} \tag{6.9}$$

6.4.4. Efficiently Keeping Local Linked Open Data Caches Up-To-Date

Update Scheduling Strategies

Linked Data that is crawled from the cloud can be represented in the form of N-Quads[4]. Technically, a quad (s, p, o, c) consists of an RDF triple where s, p, and o correspond to the subject, predicate and object and the context c, i.e., the data source on the Web where this RDF triple was retrieved.

We assume that data from the various LOD sources is retrieved at some fixed point in time t. We consider that an application visits and fetches data of LOD sources at a regular interval (say, once a week). Consequently, a LOD data source is defined by a context c and the data it provides at points in time t, i. e. the set of RDF quads $X_{c,t}$. Furthermore, we denote the size of a data set with $|X_{c,t}|$ to indicate the number of triples contained in the data set at context c at the point in time t.

We rely on local copies of the LOD sources. Such a copy typically covers several data sources. Given a set $C = \{c_1, c_2, \ldots, c_m\}$ of contexts of interest, we can define the overall dataset as:

Definition 6.4.2 *Dataset*

$$X_t = \bigcup_{c \in C} X_{c,t} \tag{6.10}$$

[4]W3C Recommendation `http://www.w3.org/TR/n-quads/`

Example 6.4.4 *As a matter of example, we consider that data comes from three different data sources:* dbpedia.org, bbc.co.uk, *and* musicbrainz.com, *and data is retrieved at two different points in time, on May 8th, 2015 and on June 10th, 2015.*

$$\begin{aligned} X_{2015\text{-}05\text{-}08} = \quad & \{X_{dbpedia.org,2015\text{-}05\text{-}08}, \quad X_{musicbrainz.com,2015\text{-}05\text{-}08}, \quad X_{bbc.co.uk,2015\text{-}05\text{-}08}\}, \\ X_{2015\text{-}06\text{-}10} = \quad & \{X_{dbpedia.org,2015\text{-}06\text{-}10}, \quad X_{musicbrainz.com,2015\text{-}06\text{-}10}, \quad X_{bbc.co.uk,2015\text{-}06\text{-}10}\} \end{aligned}$$

Finally, for distinct points $t_1, t_2, \ldots, t_n$ in time, we define a series of datasets over time:

Definition 6.4.3 *Series of Datasets*

$$\mathcal{X} = (X_{t_1}, X_{t_2}, \ldots, X_{t_n}) \tag{6.11}$$

Example 6.4.5 *Our example dataset is composed by data retrieved at two different points in time (see Example 6.4.4) such that* $\mathcal{X} = (X_{2015\text{-}05\text{-}08}, X_{2015\text{-}06\text{-}10})$.

Due to limitations such as bandwidth restrictions and the frequent data changes in the LOD cloud, LOD applications relying on data from the LOD cloud need to prioritize which data sources should be considered first in order to achieve an optimal accuracy of their local copies under the given constraints. Therefore, applications make use of a scheduling strategy for data updates. A scheduling strategy aims to derive an order of importance for data sources based on a set of data features. In the ideal case, a strategy would derive an order such that the application would visit only the subset of LOD sources which have actually been changed. In this section, we introduce a formal specification of update functions and a set of data features used by update strategies for deriving such an order.

Whenever an application needs to update a local copy covering the data sources in $c \in C$, at time t_{i+1}, it would technically be sufficient to fetch the complete dataset $X_{t_{i+1}}$. However, this would require us to visit all data sources c, retrieve their most recent version of the data $X_{c,t_{i+1}}$ and integrate it into one dataset $X_{t_{i+1}}$. Due to limitations such as network bandwidth capacity for downloading data or computation time, we assume that only a certain fraction of the data can actually be retrieved fresh from the cloud and processed in a certain time interval.

Thus, applications need to apply a scheduling strategy to efficiently manage the accuracy of the data. Based on features extracted from the dataset retrieved at an earlier point in time t_i, a scheduling strategy indicates which data sources c should be visited (i.e., visit the URI c and fetch the latest version of the data made available at this URI) in the time slice between t_i and t_{i+1}. The update strategy can be seen simply as a relation:

Definition 6.4.4 *Update Strategy*

$$U \subset C \times \{1, \ldots, n\} \tag{6.12}$$

A tuple (c, i) in this relation indicates a point in time t_i at which data from a data source c should be updated.

Furthermore, we define the constraint of the bandwidth as a restriction to download at most up to K triples.

Definition 6.4.5 *Constraint of the Bandwidth*

$$\text{For a given } i: \sum_{(c,i)\in U} |X_{c,t_i}| \leq K \tag{6.13}$$

For any given constraint of the bandwidth, it is possible to retrieve data from the sources in their order of preference until the limit has been reached.

Example 6.4.6 *Suppose our dataset has been updated most recently on May 8th, 2015 (see Example 6.4.4), and we want to update our local copy again on June 10th, 2015. However, due to limitations, the constraints of the bandwidth enables only* $K = 12,000$ *triples to be fetched per time slice. For such constraints, we suppose we cannot fetch all the data since* $|X_{dbpedia.org,2015\text{-}05\text{-}08}| + |X_{musicbrainz.com,2015\text{-}05\text{-}08}| + |X_{bbc.co.uk,2015\text{-}05\text{-}08}| \geq 12,000$*. Nevertheless, without violating these restrictions, we can entirely fetch data from the first two data sources:* dbpedia.org *and* musicbrainz.com*.*

We define a *last update* function lu to identify for a specific data source and a given point in time when its data was updated last:

Definition 6.4.6 *Last Update Function*

$$lu(c,i) = \text{argmax}_{j\leq i}\{(c,j) \in U\} \tag{6.14}$$

This function can be used recursively to identify, for instance, the update prior to the last update by $lu(c, lu(c,i) - 1)$.

Example 6.4.7 *In the previous example, we updated our local copy by fetching data from* dbpedia.org *and* musicbrainz.com*, then the* last update *time of the data sources are given as:*

$lu(\text{dbpedia.org}, \textit{2015-06-10}) = lu(\text{musicbrainz.com}, \textit{2015-06-10}) =$ *2015-06-10,*
$lu(\text{bbc.co.uk}, \textit{2015-06-10}) =$ *2015-05-08*

Using the last update function lu at time t_i, we can define the aggregated data set according to an update strategy, i.e., which version of which data source is part of the current local copy. This aggregated data set X'_{t_i} is defined as:

Definition 6.4.7 *Aggregated Data Set*

$$X'_{t_i} = \bigcup_{c\in C} X_{c,t_{lu(c,i)}} \tag{6.15}$$

Example 6.4.8 *Following Example 6.4.7, our updated dataset for June 10th, 2015 is given as:* $X'_{2015\text{-}06\text{-}10} = \{X_{dbpedia.org,2015\text{-}06\text{-}10}, X_{bbc.co.uk,2015\text{-}05\text{-}08}, X_{musicbrainz.com,2015\text{-}06\text{-}10}\}$

Finally, using this notation, we can easily construct the history of a particular data source in the course of execution of an update plan over time up to time t_i:

Definition 6.4.8

$$\mathcal{H}(c,t_i) = \{X_{c,t_j} | (c,j) \in U, t_j \leq t_i\} \tag{6.16}$$

Example 6.4.9 *The history of our sample data source* dbpedia.org *is given as:*

$$\mathcal{H}(\textit{dbpedia.org, 2015-06-10}) = \{X_{dbpedia.org,2015\text{-}06\text{-}10}, X_{dbpedia.org,2015\text{-}05\text{-}08}\}$$

Update Function

As a large number of LOD sources are available but only a limited number of sources can be fetched per run, it is required to determine which sources should be visited first. By using the vector of features of each data source, we define an update function $\rho : f \rightarrow R$, which assigns a preference score to a data source based on the observed features at time t_i.

An update strategy is defined by ranking the data sources according to their preference score in descending or ascending order, and fetching them starting from the top ranked entry to some lower ranked entry. For instance, if we consider f_{Size} to be the feature observed at time t_i for all $c \in C$, ρ could be defined as the rank of the data sources in ascending order (from the smallest to the biggest ones).

Furthermore, the bandwidth defines the amount of data that can be fetched per run. Consequently, at some point in time t_i, data of a set of data sources is updated until the bandwidth constraint has been consumed completely. For the sake of clarity, we discard a data source when its data cannot completely be fetched, i.e., when the last started fetch operation cannot be entirely executed because of the bandwidth limit being violated while reading the data. We consider that the tuple (c, i) is considered to be in the update relation U defined above, if the data from c can be entirely fetched based on the given order and the available bandwidth. Without loss of generality, we assume that the data sources are visited in a sequential order. However, it is left to the implementation to decide whether data from the different data sources should be fetched in sequential or parallel processing.

6.4.5. Evaluation

In this section, we evaluate the dynamics function and different scheduling strategies presented in the literature and analyze their effectiveness for updating local copies of the LOD sources. We evaluate these strategies on a large-scale and real world LOD dataset. Our evaluation goal is to show which of the update strategies produce better updates of the LOD sources, i.e, we demonstrate for given restrictions of bandwidth, which strategy performs better in terms of data accuracy and freshness.

Data

Our evaluation dataset is obtained from the Dynamic Linked Data Observatory (DyLDO). The DyLDO dataset has been created to monitor a fixed set of Linked Data documents (and their neighborhood) on a weekly basis[5]. Our evaluation dataset is composed of 149 weekly crawls (in the following we will refer to a crawl as a snapshot) corresponding to a period over the last three years (from May 2012 to March 2015). Furthermore, the DyLDO dataset contains various well known and large LOD sources, e.g., dbpedia.com, musicbrainz.com, and bbc.co.uk as well as less commonly known ones, e.g., advogato.org, statistics.data.gov.uk, and uefa.status.net. For more detailed information about the DyLDO dataset, we refer the reader to [Käfer et al., 2013]. As we use weekly crawls obtained from the DyLDO dataset, we are only able to grab changes occurring between consecutive weeks (e.g., daily changes are not considered).

To gain a better insight into our evaluation dataset, let us first look at the evolution of the snapshots. The number of data sources per snapshot ranges between 465 and 742. On average,

[5]For sake of consistency, we use only the kernel seeds of LOD documents

a snapshot is composed of 590 data sources. During the period studied, the number of data sources per snapshot slightly decreased, due to data sources going temporarily or permanently offline. Looking at consecutive snapshots, on average 1.05% of the data sources per snapshot are new and previously unseen (data sources birth rate), and 1.36% of the data sources disappear each week (death rate). On average, 99.3% of the data sources remain in existence between consecutive snapshots, and 37.3% of them change on a weekly basis. Taking the first snapshot as reference, only 13.9% of the data sources remain unchanged over the entire interval studied. This overview confirms prior findings [Käfer et al., 2013] indicating that a high portion of the data on the LOD cloud changes.

To provide better insights into how changes are distributed over the data sources, we randomly sampled an arbitrary point in time (June 1st, 2014) and check for the distribution of triples over data sources. We observe that most of the data sources, 78.3%, are small (containing less than 1,000 triples) and they contribute only 0.5% of all triples retrieved at this point in time. The few big data sources (0.6%) that are left (up to 1,000,000 triples), contribute more than 49.2% of all triples. Furthermore, we observe that most of the changes (66.7%) take place in the data sources with more than 1,000,000 triples.

Evaluation Methodology

Ideally, scheduling strategies should prioritize an update of data sources which provide modified data. Note that we do not consider the task of discovering new data sources for inclusion into the data cache. Rather, we want to maintain an as fresh-as-possible local copy of a fixed predefined set of LOD sources. To be able to evaluate different scheduling strategies, we use the following scenarios:

Evaluation Strategies

Single-Step We evaluate the quality of update strategies for a single and isolated update of a local data cache, i.e., starting from a perfectly accurate data cache at time t_i, our goal is to measure which quality can be achieved with different update strategies at time t_{i+1}, for varying settings of bandwidth limitations.

Iterative Progression We evaluate the evolution of the quality of a local data cache when considering iterative updates over a longer period of time, i.e., starting from a perfect data cache at time t_i, our goal is to measure how good is an update strategy in maintaining an accurate local copy at subsequent points in time $t_{i+1}, t_{i+2}, ..., t_{i+n}$ when assuming a fixed bandwidth. In our experiment, we consider four iterations.

Data Features In the following, we present features that will be used in our experiments. Please note that the set of features of a data source can always be extended.

Age provides the time span since the data source has been last visited and updated [Cho and Garcia-Molina, 2000]. It captures the age of the data provided by a data source:

$$f_{Age}(c, X'_{t_i}) = t_i - t_{lu(c,i)} \tag{6.17}$$

PageRank provides the PageRank of a data source in the overall dataset at the (last known) time [Page et al., 1999]:

$$f_{PageRank}(c, X'_{t_i}) = PR(X_{c,t_{lu(c,i)}}) \tag{6.18}$$

Size provides the (last known) number of triples provided by a data source:

$$f_{Size}(c, X'_{t_i}) = |X_{c,t_{lu(c,i)}}| \tag{6.19}$$

ChangeRatio provides the absolute number of changes of the data in a data source between the last two (known) observation points in time [Cho and Ntoulas, 2002].

$$f_{Ratio}(c, X'_{t_i}) = |X_{c,t_{lu(c,i)}} \setminus X_{c,t_{lu(c,lu(c,i)-1)}}| + |X_{c,t_{lu(c,lu(c,i)-1)}} \setminus X_{c,t_{lu(c,i)}}| \tag{6.20}$$

ChangeRate provides the change rate between the observed data in the two (last known) points in time of a data source (see Section 6.4.2).

$$f_{Change}(c, X'_{t_i}) = \Delta(X_{c,t_{lu(c,i)}}, X_{c,t_{lu(c,lu(c,i)-1)}}) \tag{6.21}$$

In this case, the change rate Δ is a function (metric) to measure the change rate between two data sets. We will use two Δ functions:

Jaccard distance:

$$\Delta(X_{c,t_{lu(c,lu(c,i)-1)}}, X_{c,t_{lu(c,i)}}) = 1 - \frac{|(X_{c,t_{lu(c,lu(c,i)-1)}}) \cap (X_{c,t_{lu(c,i)}})|}{|X_{c,t_{lu(c,lu(c,i)-1)}}) \cup (X_{c,t_{lu(c,i)}})|}$$

Dice Coefficient:

$$\Delta(X_{c,t_{lu(c,lu(c,i)-1)}}, X_{c,t_{lu(c,i)}}) = 1 - \frac{2 * |(X_{c,t_{lu(c,lu(c,i)-1)}}) \cap (X_{c,t_{lu(c,i)}})|}{|X_{c,t_{lu(c,lu(c,i)-1)}})| + |(X_{c,t_{lu(c,i)}})|}$$

Dynamics measures the behavior of the data source observed over several points in time (see Section 6.4.3), where the dynamics of a data source is defined as the aggregation of absolute changes, as provided by Δ-metrics.

$$f_{Dynamic}(c, X'_{t_i}) = \sum_{i=0}^{j} \frac{\Delta(X_{c,t_{lu(c,lu(c,i)-1)}}, X_{c,t_{lu(c,i)}})}{t_{lu(c,i)}, t_{lu(c,lu(c,i)-1)}}, j \leq i.$$

We implemented the following update strategies:

1. *Age* updates from the last to the most recently updated data source.
2. *Size-SmallestFirst* updates from the smallest to the biggest data source.
3. *Size-BiggestFirst* updates from the biggest to the smallest data source.
4. *PageRank* updates from the highest to lowest PageRank of a data source.

5. *ChangeRatio* updates from the most to the least changed data source based on set difference applied to the last two retrieved versions of the data.
6. *ChangeRate-J* updates from the most to the least changed data source based on Jaccard distance applied to the last two retrieved versions of the data.
7. *ChangeRate-D* updates from the most to the least changed data source based on Dice Coefficient applied to the last two retrieved versions of the data.
8. *Dynamics-J* updates from the most to the least dynamic data source based on Jaccard distance and previous observed snapshots of the data.
9. *Dynamics-D* updates from the most to the least dynamic data source based on Dice Coefficient and previous observed snapshots of the data.

Please note that we analyze the strategy *Age* only for the *Iterative Progression* scenario. *Age* cannot be used in the *Single Step* scenario. Since we build the follow-up copy from a perfect local copy, the feature *Age* would assign the same value to each data source.

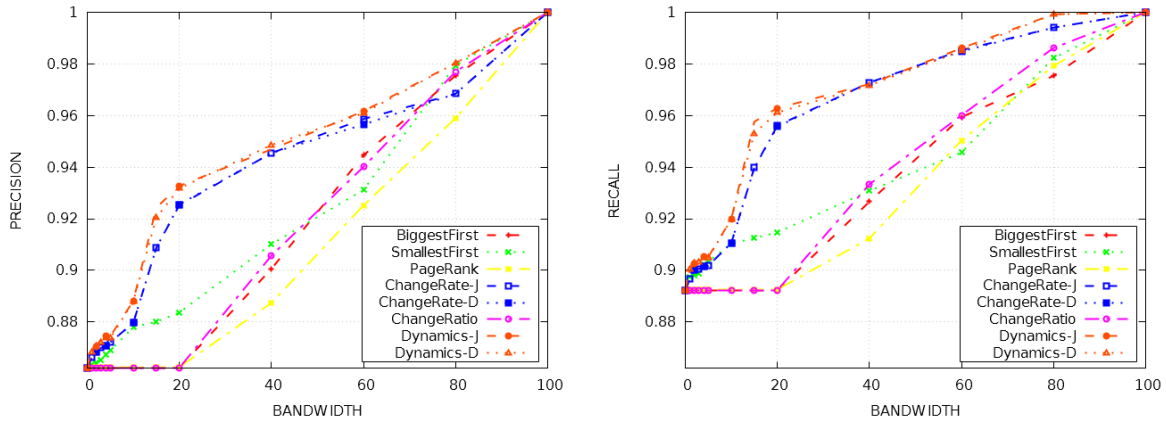
The data features used by the strategies are extracted based on the available history information. In our experiment, the history is composed of the last four updates. In the first setup, the task to be accomplished by the strategies is to compute an update order for all data sources at the point in time t_{i+1}. For the strategies *Size* and *PageRank*, we use information about data retrieved from the last update time t_i. *ChangeRatio* and *ChangeRate* are calculated over the last two updates t_{i-1} and t_i, and *Dynamics* is calculated over the complete history for points in time t_{i-4} to t_i. For the *Iterative Progression* setup, we start with a perfect data cache at t_i. The task is to compute the updates iteratively at the next points in time t_{i+1} to t_{i+4}. In the first step, the history setup is the same as the *single-step* setup and the size of the history increases along with the iterations.

In order to make the results of the different setups comparable, and due to the fact that the iterative setup considers four iterative updates, the snapshots used in the single step evaluation are the same ones which are evaluated in the first place in the iterative evaluation setup (every fifth snapshot of the dataset). Additionally, we simulate network constraints by limiting the relative bandwidth, i.e., that only a certain ratio of triples can be fetched for updating a local copy at a given point in time. In the simulation, we stepwise increase the bandwidth constraint from 0% to 5% in intervals of 1%, from 5% to 20% in intervals of 5%, and from 20% to 100% in intervals of 20% of all available triples.

LOD sources are from time to time unavailable, i.e., some LOD sources cannot be reached by any application at a certain point in time, but may be again reachable at a later point in time. Nevertheless, the implemented strategies do not differentiate whether a source is unavailable for a period of time, or if it is deleted from the cloud. Whenever a LOD source is deleted or unavailable at point in time t_i, no triples are delivered and the empty set is considered for further computations.

Metrics The quality of an update strategy is measured in terms of micro average recall and precision over the gold standard, i.e., the perfect up-to-date local copy:

$$p_{micro}(X'_t, X_t) = \frac{\sum_{c \in C^{X'_t}} |X_{c,t} \cap X'_{c,t}|}{\sum_{c \in C^{X'_t}} |X'_{c,t}|} \tag{6.22}$$



(a) Precision (b) Recall

Figure 6.6.: Quality Outcomes for the Single Step Setup

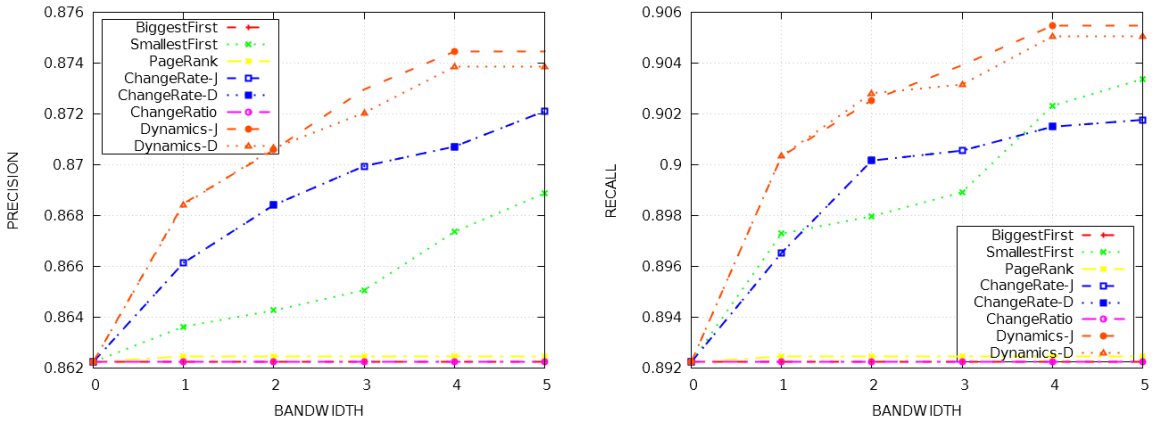
$$r_{micro}(X'_t, X_t) = \frac{\sum_{c \in C^{X'_t}} |X_{c,t} \cap X'_{c,t}|}{\sum_{c \in C^{X'_t}} |X_{c,t}|} \tag{6.23}$$

Discussion and Results

Single-Step Evaluation Figure 6.6(a) and Figure 6.6(b) show the average precision and recall over all snapshots the *single-step* setup. The x-axis represents the different levels of constraints of relative bandwidth (in percent) and the quality in terms of precision and recall is placed on the y-axis. We observe that precision ranges from 0.862 to 1 and recall from 0.892 to 1 for bandwidth from 0% to 100% (see Figure 6.6(a) and Figure 6.6(b)). This implies that if no updates are executed (no bandwidth is available), our dataset is on average 87% correct (F measure) after one update. This value can be interpreted as the probability of getting correct results when issuing an arbitrary query on the data.

Overall, the *Dynamics* strategies outperform all other strategies. First, we look at the precision curve. For very low relative (see Figure 6.7(a)) bandwidth (from 0% to 10%) the *Dynamics* strategies perform best, followed by the *ChangeRate* strategies. With only 3% available bandwidth, precision improvements is from 0.862 to 0.873 for *Dynamics*, and to 0.869 for *ChangeRate*. With 10% bandwidth the improvement rises to 0.888 for *Dynamics* and 0.879 for *ChangeRate* while the third best strategy, *SmallestFirst*, achieves 0.877 while the other strategies do not achieve scores higher than 0.862. For the higher relative bandwidths, the *ChangeRate* and *Dynamics* strategies are comparable and show only small differences in performance. Turning to recall (see Figure 6.7(b)), *ChangeRate* and *Dynamics* perform quite similarly over the entire interval and clearly outperform all strategies and all bandwidth constraints. Even with only 15% bandwidth available, the recall values are above 0.957 while all other strategies achieve at most 0.93.

LOD sources vary in their sizes. As shown, most of the big data sources change frequently and, consequently, they are in the top-ranked entries for strategies such as the *Dynamics* and *ChangeRate*. Also, some of the smaller data sources have a high change frequency. Therefore,



(a) Precision (b) Recall

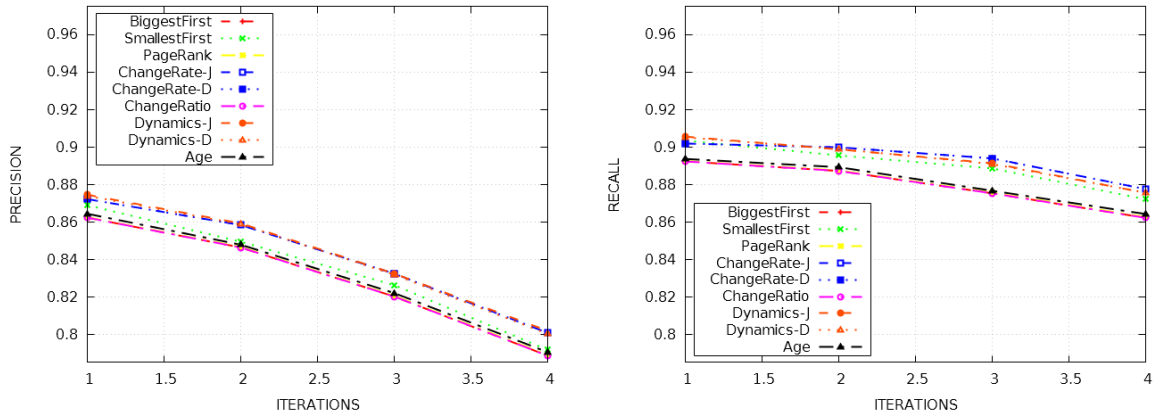
Figure 6.7.: Quality Outcomes for the Single Step Setup Setup at Low Bandwidth Level (5%).

in the ranking list provided by the *Dynamics* and *ChangeRate* strategies, a mix of big and small sources can be found in the top entries. For strategies such *BiggestFirst* and *ChangeRatio* only/most of the biggest sources are top ranked. In contrast, by the strategy *SmallestFirst* only the smallest sources are top ranked. When updating the smallest data sources first, even for a very small bandwidth, a great number of data sources can be fetched and consequently data changes can also be retrieved. This can be observed in the recall curve of the *SmallestFirst* strategy. When only low bandwidth is available, it is not possible to fetch data from big data sources since there is not enough bandwidth. This can be clearly seen for the *BiggestFirst* strategy, where updates are observed only when 20% or more bandwidth is available. The more bandwidth is available, the more changes can be retrieved. Due to the mix of data sources sizes in the ranking list of the *Dynamics* and *ChangeRate* strategies, they are able to retrieve data when only a very small bandwidth is available and overall are able to retrieve more modified data than the other strategies for all bandwidths. The other strategies narrow in quality when more bandwidth is available.

In general, the *single-step* experiments show that update strategies based on dynamics followed by change rate make best use of limited resources in terms of bandwidth. For very low relative bandwidth, the strategies based on data source dynamics tend to provide better results.

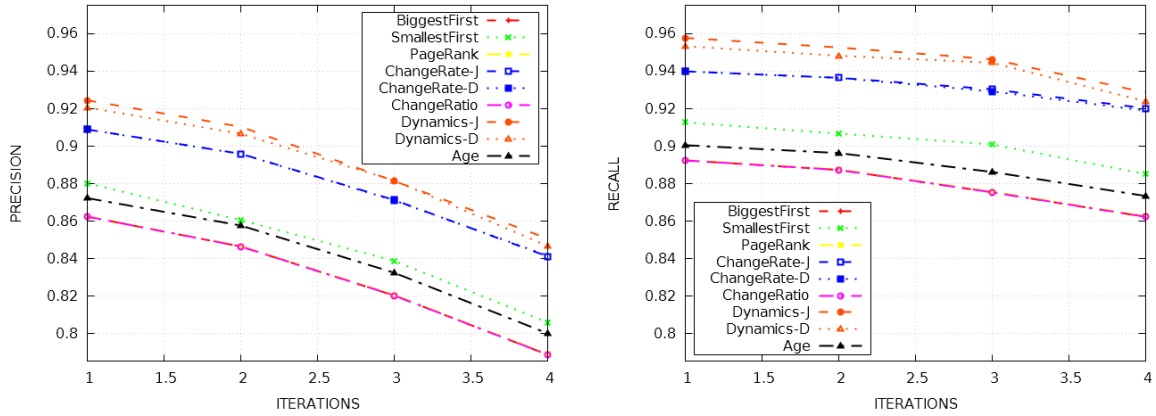
Iterative Progression Evaluation In this evaluation, we look at evolving quality when considering iterative updates. This setup simulates real-use case scenarios such as of a LOD search engine continuously updating its caches. In our experiments, we look at how precision and recall behave over the iterations. First, we fix the bandwidth constraints. We choose a low (5%), mid (15%), and high (40%) bandwidth which provided low, average, and good outcomes based on the previous experiments (*single-step* evaluation).

Figure 6.8(a) and Figure 6.8(b) show precision and recall for bandwidth fixed at 5%. The x-axis represents the iterations (points in time) and the y-axis the quality in terms of precision and recall. Note that quality decreases along the iterations. This is expected, since only at the first iterations the update process starts from a perfect data cache. For low relative bandwidths,



(a) Precision (b) Recall

Figure 6.8.: Quality Outcomes for the Iterative Progression Setup at Low Bandwidth Level (5%).

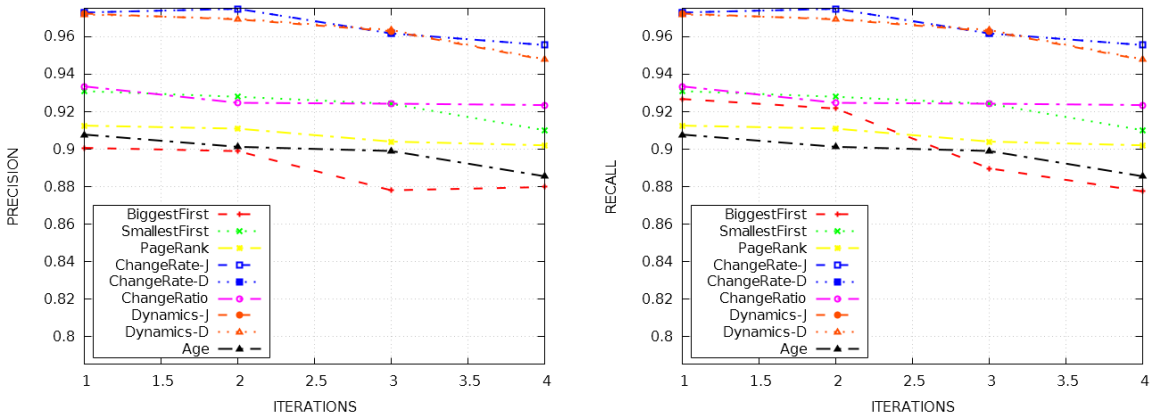


(a) Precision (b) Recall

Figure 6.9.: Quality Outcomes for the Iterative Progression Setup at Mid-Level Bandwidth (15%).

the negative impact on the quality of iterative updates is quite similar for all strategies. Nevertheless, the plot confirms the previous discussion that the *Dynamics* strategies followed by *ChangeRate* are the more appropriate ones, if we need to predict the next best steps and not only the first step anymore. Nevertheless, the strategies show a uniform loss of quality.

A similar output is observed for bandwidth fixed at 15% (see Figure 6.9(a) and Figure 6.9(b)). Here again, fetching data from the source that changes more than others ensure more accurate updates. Even if we can observe that the loss of quality is comparable, the *Dynamics* strategies followed by *ChangeRate* maintain a higher level of quality after the four iterations. *Dynamics* precision and recall decreases from 0.92 to 0.846 and 0.953 to 0.929 and *ChangeRate* from 0.908 to 0.841 and 0.939 to 0.918 after the four iterations, while the quality



(a) Precision (b) Recall

Figure 6.10.: Quality Outcomes for the Iterative Progression Setup at High-Level Bandwidth (40%).

of the other strategies are mostly lower after only one or even no iteration.

Precision and recall under a relative high bandwidth (fixed at 40%) are shown in Figure 6.10(a) and Figure 6.10(b). The recall values of the *Dynamics* strategies and *ChangeRate* hardly change over the iterations (above of 0.947). The precision values decrease with a maximum of 0.889. Over all iterations, these strategies outperform all the others even when only one-step update is applied. Interestingly, at this bandwidth level, the strategies which fetch data from the big data sources first show good performance since—up to that bandwidth level—it is possible to load a big data source entirely. As most changes concentrate in the big data sources, they are also able to fetch most of the changes. For instance, precision and recall of the *ChangeRatio* strategy reaches values of 0.888 and 0.923, respectively, after the iterations.

Overall, the results of this experiment setup confirm the discussion from the previous one, i.e, the strategies based on the dynamics features followed by the ones based on change rate are the more appropriate ones if we need to predict iterative updates. Certainly, the more bandwidth is available, the more changes can be grabbed, and therefore the rate of quality lost over the iterations is lower for all strategies. Still, even after four iterations, *Dynamics* strategies (followed by *ChangeRate*) were able to better maintain an up-to-date local copy for all different bandwidth levels. From our experiments and for low relative bandwidth, these strategies could definitely better support applications to fetch the most changed data, and thus to avoid fetching unchanged data, than the other strategies.

6.4.6. Related Work

Various related work have investigated the characteristics of the LOD cloud. Their goal is to apply these characteristics for the purpose of different applications such as query recommendations and indices updates. Some works conducted structural analysis of the LOD cloud such as [Auer et al., 2012, Hausenblas et al., 2009, Alexander and Hausenblas, 2009] in order to obtain statistical insights into the characteristics of the data. In addition, there is related work on

analyzing the LOD cloud in order to verify its compliance with established guidelines and best practices how to model and publish data as Linked Data [Hogan et al., 2012, Schmachtenberg et al., 2014]. Other works such as Neumann et al. [Neumann and Moerkotte, 2011] analyze LOD in order to obtain statistics like its distribution in the network. The goal is to apply these statistics for the purpose of query recommendation. Although these works provide interesting insights into the characteristics of LOD, they typically do not consider the dynamics of the cloud, and how it changes.

Ding and Finin [Ding and Finin, 2006]. crawled about 300 million triples from different so-called Semantic Web documents (SWDs) in 2006. The authors have conducted different analyses such as extracting the age of the SWDs based on the last-modified time information contained in the HTTP response header of the SWDs. The data exhibits an exponential distribution, which indicates that many new SWDs have been added or that many old ones are actively modified. Overall, their analysis also shows that the volume of the Semantic Web documents available on the web is growing, an observation which is consistent with and well-known from other sources like the LOD cloud web site[6]. However, it remains unknown at which point in time the different snapshots of the SWDs have been captured; the time span starting from the initial to the final snapshot is also unknown.

An analysis of temporal information in LOD is presented in [Rula et al., 2012], i. e. temporal information available in document headers and in triples. The experiments on the BTC 2012 dataset show that only 10% of all triples explicitly provide temporal information. Thus, we have decided to apply our analysis not on this dataset but on the DyLDO dataset that provides weekly snapshots of a selected set of crawled resources. Among those works that are dedicated on the study of the Linked Data dynamics, with a dedicated focus on the update frequencies of LOD search engine indices, Umbrich et al. [Umbrich et al., 2010] compare the dynamics of Linked Data and the dynamics of Linked datasets with HTML documents on the Web. Their change detection uses (i) HTTP metadata monitoring (HTTP headers including timestamps and ETags), (ii) content monitoring, and (iii) active notification of datasets. These three detection mechanisms are compared by several aspects like cost, reliability, and scalability of the mechanism. Similar to our approach, the content monitoring applies a syntactic comparison of the dataset content, i. e. a comparison of RDF triples (but ignoring inference). Change detection is a binary function which is activated whenever changes are found. In our evaluation, we consider more complex change metrics to allow fine-grained ranking.

The Dynamic Linked Data Observatory is a monitoring framework to analyze dynamics of Linked Data [Käfer et al., 2013]. Snapshots of the Web of data are regularly collected and then compared in order to detect and categorize changes. Using these snapshots, the authors study the availability of documents, the links being added to the documents, and the schema signature of documents involving predicates and values for rdf:type to determine their change rate. Only 25% of the documents change frequently and they contain a balance of documents with additions and deletions. Moreover, they showed that the rate of fresh links being added to the documents is very low. Finally, regarding the types of changes occurring on an RDF triple level, the authors conclude that the schema signature of documents involving predicates and values for rdf:type changed very infrequently. Motivated by this work, Dividino et al. [Dividino et al., 2013] analyzed the changes on the usage of the vocabulary terms in the DyLDO dataset. The authors show that the combination of vocabulary terms appearing in the LOD documents

[6] http://www.lod-cloud.net/, last accessed: 23 March, 2013

changes considerably.

In this work we conducted an evaluation of different baseline and state-of-the-art approaches of crawling strategies to improve cache maintenance. Certainly, we could not cover the wide rage of strategies existing in the literature. For instance, the PageRank metric, which is included in our evaluation, has been the subject of extensive research and different variation have been proposed so far [Gyöngyi et al., 2004, Haveliwala, 2003, Jeh and Widom, 2003, Haveliwala, 2003, Wu et al., 2006]. Nevertheless, a lot of research has been done in covering the different aspects of crawling web documents. In this section we describe the work that is the most relevant to ours.

Crawling the Web Wolf et al. [Wolf et al., 2002] discuss re-crawling strategies in order to maintain the freshness of the materialized data or search index. Similar to Pandey and Olston [Pandey and Olston, 2005], they explore page relevance (how much influence a page content has on search queries) to crawling optimization.

Change frequency (how often the content of a page is updated by its owner) is considered in many studies. Cho and Garcia-Molina [Cho and Garcia-Molina, 2003b] estimate change frequencies of Web pages when the complete change histories of the pages is not available. Brewington and Cybenko [Brewington and Cybenko, 2000a, Brewington and Cybenko, 2000b] estimate the distribution of change frequencies based on experimental data. Cho and Garcia-Molina [Cho and Garcia-Molina, 2003a] proposed a crawl policy which focuses only on the probability of change for pages that do not change very frequently. They claim that recrawling pages that change very frequently may dominates crawl resources and actually does not guarantee cache freshness.

Olston and Pandey [Olston and Pandey, 2008] present the concept of information longevity, i. e. the lifetime of the content that appears and disappears from the Web. They show that there is no correlation between information longevity and change frequency, and present a generative model combining the longevity and change profile.

Some works [Fetterly et al., 2004, Cho and Ntoulas, 2002, Tan and Mitra, 2010] exploit the correlation between content change with the top-level domains of web pages. Cho and Ntoulas use this correlation to estimate the change probability of any page from the website. Tan and Mitra designed a crawling algorithm that clusters Web pages based on features that correlate to the change frequencies obtained by examining past history.

Radinsky and Bennett [Radinsky and Bennett, 2013] propose an expert predictive framework for predicting content changes on the Web using past history and based on features, such as relatedness to other pages and similarity in the types of changes.

Calzarossa and Tessera [Calzarossa and Tessera, 2015] studied the temporal patterns of Web content changes as time series whose analysis provides models able to explain their dynamics. Their approach reproduces the dynamics of the empirical change patterns and provides extrapolations into the future to be used for forecasting.

Time series analysis is applied in [Zhang et al., 2009] to develop models for the analysis of Web searchersâ€™ behaviors over time. The predictive models are based on the dynamic patterns of the interactions between users and search engines.

LOD Cloud Many of the works dedicated to crawling the LOD cloud such as LDSpider [Isele et al., 2010]and LOD Laundromat [Beek et al., 2014] focus on different issues, such as

the crawler architecture, topic coverage, crawler parallelization [Lašek et al., 2012], etc.

The importance of caching for efficient querying linked data is analyzed by Hartig et al. [Hartig, 2013]. Query execution is based on traversing RDF links to discover data that might be relevant for a query during the query execution itself. Data is cached and it is used for further queries. Caching show some beneficial impact to improve the completeness of the results.

In [Lampo et al., 2011] the authors investigate which type of SPARQL queries can benefit from caching data during query execution or warming up cache. They demonstrate that caching can have a positive effect on complex SPARQL queries.

Umbrich et al. [Umbrich et al., 2012] propose a hybrid approach for answering SPARQL queries, i. e, deciding which parts of a query are suitable for local or remote execution. They estimate the freshness of cached data using the notion of coherence for triple patterns against the live engine. Only static data is kept locally and they determine which parts of the query should be evaluated remotely and locally. Dehghanzadeh et al. [Dehghanzadeh et al., 2014] extend this approach by extending the statistics of cardinality estimation techniques that are used in the join query processing phase.

Roussakis et al. [Roussakis et al., 2015] proposes an approach that copes with automatic identification of deltas between versions which prescribes (i) the definition of custom, application-specific changes and their management (definition, storage, detection) in a manner that ensures the satisfaction of formal properties, like completeness and unambiguity, (ii) the flexibility and customization of the considered changes, via complex changes that can be defined at run-time, and (iii) the easy configuration of a scalable detection mechanism, via a generic algorithm that builds upon SPARQL queries easily generated from the changes definitions.

6.4.7. Findings and Research Contribution

We proposed and evaluated scheduling strategies for updating on a large-scale LOD dataset that was obtained from the cloud by weekly crawls over the course of three years. In a first setup, where we evaluated the quality of update strategies for a single and isolated update of a local data cache, we observed that update strategies based on dynamics or change rate made best use of limited resources in terms of bandwidth. For very low relative bandwidth, the strategies based on data source dynamics provided better results. With only 15% available bandwidth, we observed improvements of precision and recall for dynamics from 0.862 to 0.924 and from 0.892 to 0.957, respectively.

In a second evaluation setup, we evaluated the behavior of the strategies in a realistic scenario (e.g., a LOD search engine updating its caches) which involves measuring the quality of the local data cache when considering iterative updates over a longer period of time. Overall the results of this experiment setup confirmed the discussion from the previous one: Especially for low relative bandwidth, update strategies based on dynamics or change rate are more appropriate to support applications to fetch the most changed data (and thus avoid fetching unchanged data) than the other strategies.

In future work, we plan to investigate the impact on performance when combining different update strategies as well as to consider further evaluation setups such as the cold start setup, that is, we will measure how good is an update strategy starting from an empty cache and

considering iterative updates over a longer period of time. Furthermore, we plan to extend these strategies to consider the availability of the LOD sources over time, namely, to be able to differentiate whether a source is unavailable for a period of time or has been definitively deleted from the cloud.

7. Conclusion and Further Directions

My work demonstrates by examples the importance and benefits of the use of provenance in different Web applications and scenarios. In particular, this dissertation presents mechanisms to manage and use provenance in its many dimensions when querying, debugging or repairing, ranking and updating caches of Web data. The approaches presented show clearly the value of provenance as it enables new kinds of data analysis and quality assessment on top of the raw data. The flexibility of these approaches combined with their high scalability makes my work a possible building block for a Semantic Web proof and trust layer.

Furthermore, this dissertation can be seen as a motivation to Web sources to publish correct and valid provenance values to support application needs. I believe that publishing correct provenance information is a step towards quality-oriented data usage in the Web.

In the following, the findings and future directions for this research are listed.

7.1. Findings

Provenance Management in the Semantic Web

Querying RDF Datasets with Provenance

This chapter presented an original, generic, formalized and implemented approach for the management of many dimensions of provenance, like source, authorship, certainty, and others, for RDF repositories. This method re-uses existing RDF modeling possibilities in order to represent provenance. Then, it extends SPARQL query processing in such a way that given a SPARQL query for data, one may request provenance without modifying the query proper.

This approach remains compatible to existing standards and query languages and can be easily integrated with existing applications and interfaces. It achieves highly flexible and automatically coordinated querying for data and provenance, while completely separating the two areas of concern.

Reasoning and Debugging Evolving OWL Ontologies with Provenance

This chapter described a black-box algorithm for optimized reasoning with provenance. With the algorithm proposed, one can efficiently reason in OWL ontologies with provenance, i.e., provenance is efficiently combined and propagated within the reasoning process. Therefore, the black-box algorithm for reasoning with provenance enables the use of provenance for real time for very large and expressive ontologies.

The algorithm presented computes the explanations of the answer and using the pinpointing formula to compute provenance in an algebraic way. However, it does not need to compute *all* pinpoints as the computation of pinpoints may become very expensive and inapplicable if users need to interact with dynamically changing knowledge in real time. The optimized algorithm for deriving provenance performs

significantly better in the average case than naïve implementations. It does not need to compute all pinpoints, and in fact does not even need to pinpoint precisely. Instead, it computes an approximation which is sufficient for deriving provenance.

The evaluation has shown that my approach performs significantly better than existing general pinpointing algorithms, scales well, and is applicable for interactive applications. It has demonstrated that the algorithm performs orders of magnitude better than a naïve implementation. Therefore, provenance information does not only provide value to the end user, it can be further used to considerably speed up debugging processes by rapidly approximating a solution.

The framework presented was restricted to ontology diagnosis scenarios. It, however, introduces an approach for provenance querying under a variety of scenarios that consider many of the dimensions of provenance such as restrictions of access rights, knowledge validity when the truth of knowledge changes with time, and inferring trust value.

Using Provenance in Semantic Web Applications

An Efficient Provenance-Aware News Feed Ranking Algorithm via Preference Aggregation

In this chapter, we have proposed three algorithms that provide the most relevant news feeds according to the user's preferences. Provenance is used to represent users preferences. Our algorithms rank messages 'on the fly' as the message passes through the system.

We have introduced the concept of online preference aggregators and investigated the consequences of adding a new element to the preference orders. We have studied the relationships between the original aggregated preference order and the updated aggregated preference order, and established a framework for computational approaches to online preference aggregation. Concrete online aggregation algorithms and complexity analysis have been presented for the plurality, Borda count, and sequential pairwise voting methods. Our complexity analysis has shown that the online aggregator performs better than the original aggregators after the domain changes. The computation of the original aggregators has time complexity $O(n*m)$, and we have shown that the online plurality aggregator has time complexity $O(m)$, the online Borda count aggregator $O(m * logn)$, and the sequential pairwise aggregator $O(m)$.

Managing Data Changes in the Linked Open Data Sources

This chapter presented two techniques for dealing with data dynamics in the LOD sources using provenance information. First, it presented an evaluation of the conformance of LOD data sources to provide a valid and correct *Last-Modified* HTTP header field, which indicates the date and time at which the resource was last modified. The experiment shows that overall and on average only 8% of the resources in the datasets provide correct values for this field. This number is far too low to be of use for any practical application. It is, however, not clear why LOD sources do not provide valid information. We conjecture that some default configuration of LOD servers leads to this misbehavior.

This analysis is restricted to the *Last-Modified* field; however, it could be easily extended to check the conformance verification of other HTTP header fields.

The reliable provision of provenance information in the context of the established HTTP protocol would be beneficial to the entire Web of Data. Many base technologies such as Linked Data caches and indexes may benefit from this information since a simple check on this provenance could support their decision process of determining which sources need to be updated.

We believe that publishing correct HTTP Header information is a step towards quality-oriented data usage in the LOD cloud. Therefore, with this work we point out the dimension of the problem of erroneous and missing information from the HTTP header for Linked Data. My work, therefore, motivates LOD sources to publish correct and valid values to support application needs.

Then, this chapter presented a general and flexible framework for analyzing data dynamics on the LOD cloud. Different from quantifying changes of datasets, the dynamics capture the evolution of a dataset over time. The dynamics function is defined as the aggregation of absolute, infinitesimal changes, where such changes may be quantified by the different existing change metrics in the literature. Furthermore, this method can be parameterized to make use of different decay functions for stressing or weakening changes as time passes.

The dynamics function is evaluated for the purpose of data caching on a large-scale on a large-scale LOD dataset that was obtained from the cloud by weekly crawls over the course of three years. The evaluation includes different scheduling strategies and investigates two different setups: (i) in the single step setup, the quality of update strategies for a single and isolated update of a local data cache is evaluated, while (ii) in the iterative progression setup, the evaluation involves measuring the quality of the local data cache when considering iterative updates over a longer period of time.

Mainly, the evaluation shows that the measures capturing change behavior of LOD sources over time are most suitable for conducting updates. In the first setup, it was observed that the update strategies based on dynamics or change rate make best use of limited resources in terms of bandwidth. For very low relative bandwidth, the strategies based on data source dynamics provide better results. With only 15% available bandwidth, we observed improvements of precision and recall for dynamics from 0.862 to 0.924 and from 0.892 to 0.957, respectively.

In the second evaluation setup, the overall results confirm the discussion from the previous one. The strategies based on the dynamics features followed by the ones based on change rate are the more appropriate ones if one needs to predict (iterative) updates. Certainly, the more bandwidth is available, the more changes can be grabbed; and therefore the rate of quality lost over the iterations is lower for all strategies. Still even after four iterations, *Dynamics* strategies (followed by *ChangeRate*) were able to better maintain an up-to-date local copy for all different bandwidth levels. From the experiments and for low relative bandwidth, these strategies could definitely better support applications to fetch the most changed data (and thus to avoid fetching unchanged data) than the other strategies.

7.2. Future Directions

Querying RDF Datasets with Provenance

Management of provenance incurs costs for its collection and for its storage. In general, provenance information can grow to be larger than the data it describes if the data is fine-grained and provenance information rich. So the manner in which the provenance is propagated along the computation is crucial to its scalability.

Recent works tackle some of these issues. In [Wylot et al., 2014, Wylot et al., 2015a], the authors present the TripleProv, a native RDF store that allows tracking and querying provenance over Web Data. Later in [Wylot et al., 2015b], Wylot et al. use the TripleProv store to investigate the effectivess of different query execution strategy for provenance-enabled queries. In [Deutch et al., 2015b], Deutch et al. describe an approach for web scale provenance tracking. Their approach allows selective tracking of how-provenance, where the selection criteria are partly based on the meta-data itself (thus significantly reducing provenance size). Summarization technique for provenance graphs haven been presented by Luc Moreau et al. [Moreau, 2015]. Provenance summaries can be considered as "compact users views" and have a good potential to detect anomalies and outliers.

In order to extend our approach to fully capture expressiveness of SPARQL (including the OPTIONAL construct), the provenance model has to be extended or changed. In [Theoharis et al., 2011, Geerts et al., 2013, Karvounarakis et al., 2013] Theoharis et al. and Karvounarakis et al. discuss the need for extending the relational provenance models to be leveraged for SPARQL queries over RDF. Particularly, they discuss that the use of semiring models for SPARQL have been shown inadequate to handle the OPTIONAL construct, and have advocated the need for a new abstract provenance model capturing the full expressiveness of SPARQL. In addition, Geerts et al. [Geerts et al., 2016] identify SPARQL fragments for which provenance models for positive relational queries can be leveraged, despite the subtle differences between the semantics of SPARQL and relational algebra operators. In [Amsterdamer et al., 2011c], the writers show particular semirings for which an extension for supporting difference is impossible.

Reasoning and Debbuging Evolving OWL Ontologies with Provenance

This work has introduced an optimized algorithm for tracking undesired inferences and inconsistencies using provenance when answering queries upon evolving ontologies on the Semantic Web. Further optimizations on the provenance selection criteria are possible such as an extension towards a possibilistic logic, e.g. based on [Knechtel and Peñaloza, 2010, Baader et al., 2009] and on integration with a system for tracking ontology changes. Future plans include applying this approach to provenance to other logical formalisms beyond DL $\mathcal{SRIQ}(\mathcal{D})$.

An Efficient Provenance-Aware News Feed Ranking Algorithm via Preference Aggregation

Standard voting methods require in the worst case the complete re-computation of the aggregation after changes in the domain. This chapter has shown, however, that for the plurality, Borda count, and sequential pairwise voting aggregators, the dynamic setting can be better handled.

Future work includes the undertaking of an empirical evaluation to analyze how the proposed algorithms behave in real use case scenarios.

Managing Data Changes in the Linked Open Data Sources

This chapter first presented an analysis of the availability and conformance of the HTTP Header's *Last-Modified* field. This analysis is restricted to the *Last-Modified* field, however, in future work, it could be extended to check the conformance verification of others HTTP header fields.

Then, it presented a function to measure the dynamics of a RDF dataset that capture the evolution of a dataset over time. This research could be further explored to approximate the change rate function based on piecewise linear functions, polynomial interpolation and cubic splines over the observations of changes at discrete points in time. However, the benefit of these more sophisticated approximations needs to be evaluated in different real- world scenarios.

At last, this chapter described an evaluation of different update scheduling strategies in two different setups. Future work includes the investigation of the impact on performance when combining different update strategies as well as to consider further evaluation setups such as the cold start setup in order to measure how good is an update strategy starting from an empty cache and to consider iterative updates over a longer period of time.

Bibliography

[DBL, 2002] (2002). *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*. Morgan Kaufmann.

[PRO, 2012] (2012). Prov graph layout conventions, technical note. Technical report, PROV Working Group - World Wide Web Consortium.

[Abedjan et al., 2014] Abedjan, Z., Grütze, T., Jentzsch, A., and Naumann, F. (2014). Profiling and mining RDF data with prolod++. In Cruz, I. F., Ferrari, E., Tao, Y., Bertino, E., and Trajcevski, G., editors, *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 1198–1201. IEEE.

[Adler and de Alfaro, 2007a] Adler, B. T. and de Alfaro, L. (2007a). A content-driven reputation system for the wikipedia. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 261–270, New York, NY, USA. ACM.

[Adler and de Alfaro, 2007b] Adler, B. T. and de Alfaro, L. (2007b). A content-driven reputation system for the wikipedia. In [Williamson et al., 2007], pages 261–270.

[Agrawal et al., 2006] Agrawal, P., Benjelloun, O., Sarma, A. D., Hayworth, C., Nabar, S. U., Sugihara, T., and Widom, J. (2006). Trio: A system for data, uncertainty, and lineage. In [Dayal et al., 2006], pages 1151–1154.

[Aldeco-Pérez and Moreau, 2010] Aldeco-Pérez, R. and Moreau, L. (2010). *A Provenance-Based Compliance Framework*, pages 128–137. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Alexander and Hausenblas, 2009] Alexander, K. and Hausenblas, M. (2009). Describing linked datasets - on the design and usage of void, the 'vocabulary of interlinked datasets. In *In Linked Data on the Web Workshop (LDOW 09), in conjunction with 18th International World Wide Web Conference (WWW 09*.

[Amann et al., 2013] Amann, B., Constantin, C., Caron, C., and Giroux, P. (2013). Weblab prov: Computing fine-grained provenance links for xml artifacts. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, EDBT '13, pages 298–306, New York, NY, USA. ACM.

[Amsterdamer et al., 2011a] Amsterdamer, Y., Davidson, S. B., Deutch, D., Milo, T., Stoyanovich, J., and Tannen, V. (2011a). Putting lipstick on pig: Enabling database-style workflow provenance. *PVLDB*, 5(4):346–357.

[Amsterdamer et al., 2011b] Amsterdamer, Y., Deutch, D., Milo, T., and Tannen, V. (2011b). On provenance minimization. In Lenzerini, M. and Schwentick, T., editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 141–152. ACM.

[Amsterdamer et al., 2012] Amsterdamer, Y., Deutch, D., Milo, T., and Tannen, V. (2012). On provenance minimization. *ACM Trans. Database Syst.*, 37(4):30.

[Amsterdamer et al., 2011c] Amsterdamer, Y., Deutch, D., and Tannen, V. (2011c). On the limitations of provenance for queries with difference. In Buneman, P. and Freire, J., editors, *3rd Workshop on the Theory and Practice of Provenance, TaPP'11, Heraklion, Crete, Greece, June 20-21, 2011*. USENIX Association.

[Analyti et al., 2014] Analyti, A., Damásio, C. V., Antoniou, G., and Pachoulakis, I. (2014). Why-provenance information for rdf, rules, and negation. *Ann. Math. Artif. Intell.*, 70(3):221–277.

[Archer et al., 2013] Archer, D. W., Delcambre, L. M. L., and Maier, D. (2013). User trust and judgments in a curated database with explicit provenance. In [Tannen et al., 2013], pages 89–111.

[Arenas et al., 2009] Arenas, M., Gutierrez, C., and Pérez, J. (2009). On the semantics of SPARQL. In Virgilio, R. D., Giunchiglia, F., and Tanca, L., editors, *Semantic Web Information Management - A Model-Based Perspective*, pages 281–307. Springer.

[Aroyo et al., 2010] Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., and Tudorache, T., editors (2010). *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I*, volume 6088 of *Lecture Notes in Computer Science*. Springer.

[Aroyo et al., 2009] Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., and Simperl, E. P. B., editors (2009). *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4, 2009, Proceedings*, volume 5554 of *Lecture Notes in Computer Science*. Springer.

[Arrow, 1950] Arrow, K. J. (1950). A Difficulty in the Concept of Social Welfare. *Journal of Political Economy*, 58(4):328–346.

[Artale et al., 2009] Artale, A., Kontchakov, R., Ryzhikov, V., and Zakharyaschev, M. (2009). *DL-Lite* with temporalised concepts, rigid axioms and roles. In Ghilardi, S. and Sebastiani, R., editors, *Frontiers of Combining Systems, 7th International Symposium, FroCoS 2009, Trento, Italy, September 16-18, 2009. Proceedings*, volume 5749 of *Lecture Notes in Computer Science*, pages 133–148. Springer.

[Artale et al., 2013] Artale, A., Kontchakov, R., Wolter, F., and Zakharyaschev, M. (2013). Temporal description logic for ontology-based data access. In Rossi, F., editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. IJCAI/AAAI.

[Artale et al., 2007] Artale, A., Lutz, C., and Toman, D. (2007). A description logic of change. In Veloso, M. M., editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 218–223.

[Assaf et al., 2015a] Assaf, A., Senart, A., and Troncy, R. (2015a). What's up LOD cloud? observing the state of linked open data cloud metadata. In Rula, A., Zaveri, A., Knuth, M., and Kontokostas, D., editors, *Proceedings of the 2nd Workshop on Linked Data Quality co-located with 12th Extended Semantic Web Conference (ESWC 2015), Portorož, Slovenia, June 1, 2015.*, volume 1376 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Assaf et al., 2015b] Assaf, A., Troncy, R., and Senart, A. (2015b). Roomba: An extensible framework to validate and build dataset profiles. In Berendt, B., Dragan, L., Hollink, L., Luczak-Rösch, M., Demidova, E., Dietze, S., Szymanski, J., and Breslin, J. G., editors, *Joint Proceedings of the 5th International Workshop on Using the Web in the Age of Data (USEWOD '15) and the 2nd International Workshop on Dataset PROFIling and fEderated Search for Linked Data (PROFILES '15) co-located with the 12th European Semantic Web Conference (ESWC 2015), Portorož, Slovenia, May 31 - June 1, 2015.*, volume 1362 of *CEUR Workshop Proceedings*, pages 32–46. CEUR-WS.org.

[Auer et al., 2012] Auer, S., Demter, J., Martin, M., and Lehmann, J. (2012). Lodstats - an extensible framework for high-performance dataset analytics. In [ten Teije et al., 2012], pages 353–362.

[Avgoustaki et al., 2016] Avgoustaki, A., Flouris, G., Fundulaki, I., and Plexousakis, D. (2016). Provenance management for evolving RDF datasets. In Sack, H., Blomqvist, E., d'Aquin, M., Ghidini, C., Ponzetto, S. P., and Lange, C., editors, *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, volume 9678 of *Lecture Notes in Computer Science*, pages 575–592. Springer.

[Baader et al., 2013] Baader, F., Borgwardt, S., and Lippmann, M. (2013). Temporalizing ontology-based data access. In *Proceedings of the 24th International Conference on Automated Deduction*, CADE'13, pages 330–344, Berlin, Heidelberg. Springer-Verlag.

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA.

[Baader et al., 2009] Baader, F., Knechtel, M., and Peñaloza, R. (2009). A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In [Bernstein et al., 2009], pages 49–64.

[Baader and Peñaloza, 2010] Baader, F. and Peñaloza, R. (2010). Axiom pinpointing in general tableaux. *J. Log. Comput.*, 20(1):5–34.

[Bao et al., 2012] Bao, Z., Davidson, S. B., and Milo, T. (2012). Labeling workflow views with fine-grained dependencies. *PVLDB*, 5(11):1208–1219.

[Barbieri et al., 2010a] Barbieri, D. F., Braga, D., Ceri, S., Valle, E. D., and Grossniklaus, M. (2010a). C-SPARQL: a continuous query language for RDF data streams. *Int. J. Semantic Computing*, 4(1):3–25.

[Barbieri et al., 2010b] Barbieri, D. F., Braga, D., Ceri, S., Valle, E. D., and Grossniklaus, M. (2010b). Incremental reasoning on streams and rich background knowledge. In [Aroyo et al., 2010], pages 1–15.

[Batsakis et al., 2015] Batsakis, S., Tachmazidis, I., and Antoniou, G. (2015). Representing time for the semantic web. In Bikakis, A. and Zheng, X., editors, *Multi-disciplinary Trends in Artificial Intelligence - 9th International Workshop, MIWAI 2015, Fuzhou, China, November 13-15, 2015, Proceedings*, volume 9426 of *Lecture Notes in Computer Science*, pages 3–15. Springer.

[Beek et al., 2014] Beek, W., Rietveld, L., Bazoobandi, H. R., Wielemaker, J., and Schlobach, S. (2014). LOD laundromat: A uniform way of publishing other people's dirty data. In [Mika et al., 2014], pages 213–228.

[Belhajjame et al., 2013a] Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., Miles, S., James, Sahoo, S., and Tilmes, C. (2013a). Prov-dm: The prov data model. Technical report, W3C Recommendation.

[Belhajjame et al., 2013b] Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., and Zhao, J. (2013b). Prov-o: The prov ontology. Technical report, W3C Recommendation.

[Berners-Lee, 2006] Berners-Lee, T. (2006). Linked Data - W3C Design Issues . Technical report, W3C.

[Bernstein et al., 2009] Bernstein, A., Karger, D. R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., and Thirunarayan, K., editors (2009). *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, volume 5823 of *Lecture Notes in Computer Science*. Springer.

[Bhagwat et al., 2005] Bhagwat, D., Chiticariu, L., Tan, W. C., and Vijayvargiya, G. (2005). An annotation management system for relational databases. *VLDB J.*, 14(4):373–396.

[Bienvenu et al., 2012] Bienvenu, M., Deutch, D., and Suchanek, F. M. (2012). Provenance for web 2.0 data. In Jonker, W. and Petkovic, M., editors, *Secure Data Management - 9th VLDB Workshop, SDM 2012, Istanbul, Turkey, August 27, 2012. Proceedings*, volume 7482 of *Lecture Notes in Computer Science*, pages 148–155. Springer.

[Bistarelli et al., 2008] Bistarelli, S., Martinelli, F., and Santini, F. (2008). A semantic foundation for trust management languages with weights: An application to the rtfamily. In Rong, C., Jaatun, M. G., Sandnes, F. E., Yang, L. T., and Ma, J., editors, *Autonomic and Trusted Computing, 5th International Conference, ATC 2008, Oslo, Norway, June 23-25, 2008, Proceedings*, volume 5060 of *Lecture Notes in Computer Science*, pages 481–495. Springer.

[Biton et al., 2007] Biton, O., Boulakia, S. C., and Davidson, S. B. (2007). Zoom*userviews: Querying relevant provenance in workflow systems. In Koch, C., Gehrke, J., Garofalakis, M. N., Srivastava, D., Aberer, K., Deshpande, A., Florescu, D., Chan, C. Y., Ganti, V., Kanne, C., Klas, W., and Neuhold, E. J., editors, *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, pages 1366–1369. ACM.

[Bizer and Cyganiak, 2014] Bizer, C. and Cyganiak, R. (2014). RDF 1.1 TriG - RDF Dataset Language. . W3c recommendation, W3C.

[Bizer et al., 2009] Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22.

[Böhm et al., 2011] Böhm, C., Lorey, J., and Naumann, F. (2011). Creating void descriptions for web-scale data. *J. Web Sem.*, 9(3):339–345.

[Bolles et al., 2008] Bolles, A., Grawunder, M., and Jacobi, J. (2008). Streaming SPARQL - extending SPARQL to process data streams. In Bechhofer, S., Hauswirth, M., Hoffmann, J., and Koubarakis, M., editors, *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, volume 5021 of *Lecture Notes in Computer Science*, pages 448–462. Springer.

[Bonatti et al., 2011] Bonatti, P. A., Hogan, A., Polleres, A., and Sauro, L. (2011). Robust

and scalable linked data reasoning incorporating provenance and trust annotations. *J. Web Sem.*, 9(2):165–201.

[Booth et al., 2006] Booth, R., Meyer, T. A., and Wong, K. (2006). A bad day surfing is better than a good day working: How to revise a total preorder. In [Doherty et al., 2006], pages 230–238.

[Bose and Frew, 2005] Bose, R. and Frew, J. (2005). Lineage retrieval for scientific data processing: a survey. *ACM Comput. Surv.*, 37(1):1–28.

[Boulakia and Tan, 2009] Boulakia, S. C. and Tan, W. C. (2009). Provenance in scientific databases. In Liu, L. and Özsu, M. T., editors, *Encyclopedia of Database Systems*, pages 2202–2207. Springer US.

[Bowers and Ludäscher, 2005] Bowers, S. and Ludäscher, B. (2005). Actor-oriented design of scientific workflows. In Delcambre, L. M. L., Kop, C., Mayr, H. C., Mylopoulos, J., and Pastor, O., editors, *Conceptual Modeling - ER 2005, 24th International Conference on Conceptual Modeling, Klagenfurt, Austria, October 24-28, 2005, Proceedings*, volume 3716 of *Lecture Notes in Computer Science*, pages 369–384. Springer.

[Brafman and Domshlak, 2009] Brafman, R. I. and Domshlak, C. (2009). Preference handling - an introductory tutorial. *AI Magazine*, 30(1):58–86.

[Brewington and Cybenko, 2000a] Brewington, B. E. and Cybenko, G. (2000a). How dynamic is the web? *Computer Networks*, 33(1-6):257–276.

[Brewington and Cybenko, 2000b] Brewington, B. E. and Cybenko, G. (2000b). Keeping up with the changing web. *IEEE Computer*, 33(5):52–58.

[Bruno et al., 2002] Bruno, N., Chaudhuri, S., and Gravano, L. (2002). Top-k selection queries over relational databases: Mapping strategies and performance evaluation. *ACM Trans. Database Syst.*, 27(2):153–187.

[Buneman, 2013] Buneman, P. (2013). The providence of provenance. In Gottlob, G., Grasso, G., Olteanu, D., and Schallhart, C., editors, *Big Data - 29th British National Conference on Databases, BNCOD 2013, Oxford, UK, July 8-10, 2013. Proceedings*, volume 7968 of *Lecture Notes in Computer Science*, pages 7–12. Springer.

[Buneman et al., 2006] Buneman, P., Chapman, A., and Cheney, J. (2006). Provenance management in curated databases. In [Chaudhuri et al., 2006], pages 539–550.

[Buneman et al., 2012] Buneman, P., Cheney, J., and Kostylev, E. V. (2012). Hierarchical models of provenance. In Acar, U. A. and Green, T. J., editors, *4th Workshop on the Theory and Practice of Provenance, TaPP'12, Boston, MA, USA, June 14-15, 2012*. USENIX Association.

[Buneman et al., 2008] Buneman, P., Cheney, J., and Vansummeren, S. (2008). On the expressiveness of implicit provenance in query and update languages. *ACM Trans. Database Syst.*, 33(4).

[Buneman et al., 2000] Buneman, P., Khanna, S., and Tan, W. C. (2000). Data provenance: Some basic issues. In Kapoor, S. and Prasad, S., editors, *Foundations of Software Technology and Theoretical Computer Science, 20th Conference, FST TCS 2000 New Delhi, India, December 13-15, 2000, Proceedings.*, volume 1974 of *Lecture Notes in Computer Science*, pages 87–93. Springer.

[Buneman et al., 2001] Buneman, P., Khanna, S., and Tan, W. C. (2001). Why and where: A characterization of data provenance. In den Bussche, J. V. and Vianu, V., editors, *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings.*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer.

[Buneman et al., 2002] Buneman, P., Khanna, S., and Tan, W. C. (2002). On propagation of deletions and annotations through views. In Popa, L., Abiteboul, S., and Kolaitis, P. G., editors, *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, pages 150–158. ACM.

[Buneman and Kostylev, 2010] Buneman, P. and Kostylev, E. V. (2010). Annotation algebras for RDFS. In *The Second International Workshop on the role of Semantic Web in Provenance Management (SWPM-10)*. CEUR Workshop Proceedings.

[Callahan et al., 2006] Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., and Vo, H. T. (2006). Managing the evolution of dataflows with vistrails. In Barga, R. S. and Zhou, X., editors, *Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDE 2006, 3-7 April 2006, Atlanta, GA, USA*, page 71. IEEE Computer Society.

[Calzarossa and Tessera, 2015] Calzarossa, M. C. and Tessera, D. (2015). Modeling and predicting temporal patterns of web content changes. *Journal of Network and Computer Applications*, 56:115 – 123.

[Can and Storcken, 2013] Can, B. and Storcken, T. (2013). Update monotone preference rules. *Mathematical Social Sciences*, 65(2):136–149.

[Cappiello, 2015] Cappiello, C. (2015). On the role of data quality in improving web information value. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 1433–1433, New York, NY, USA. ACM.

[Carroll et al., 2005] Carroll, J. J., Bizer, C., Hayes, P. J., and Stickler, P. (2005). Named graphs. *J. Web Sem.*, 3(4):247–267.

[Carroll and Stickler, 2004] Carroll, J. J. and Stickler, P. (2004). TriX: RDF triples in XML. In *Proceedings of the Extreme Markup Languages 2004*, Montreal, Canada.

[Chaudhuri et al., 2006] Chaudhuri, S., Hristidis, V., and Polyzotis, N., editors (2006). *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006*. ACM.

[Chebotko et al., 2010] Chebotko, A., Lu, S., Fei, X., and Fotouhi, F. (2010). Rdfprov: A relational RDF store for querying and managing scientific workflow provenance. *Data Knowl. Eng.*, 69(8):836–865.

[Cheney et al., 2014] Cheney, J., Ahmed, A., and Acar, U. A. (2014). Database queries that explain their work. In *Proceedings of the 16th International Symposium on Principles and Practice of Declarative Programming*, PPDP '14, pages 271–282, New York, NY, USA. ACM.

[Cheney et al., 2009] Cheney, J., Chiticariu, L., and Tan, W. C. (2009). Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474.

[Cheney and Soiland-Reyes, 2013] Cheney, J. and Soiland-Reyes, S. (2013). Prov-n: The provenance notation. Technical report, W3C Recommendation.

[Chirigati and Freire, 2012] Chirigati, F. and Freire, J. (2012). Towards integrating workflow and database provenance. In *Proceedings of the 4th International Conference on Provenance*

and Annotation of Data and Processes, IPAW'12, pages 11–23, Berlin, Heidelberg. Springer-Verlag.

[Cho and Garcia-Molina, 2000] Cho, J. and Garcia-Molina, H. (2000). Synchronizing a database to improve freshness. In Chen, W., Naughton, J. F., and Bernstein, P. A., editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 117–128. ACM.

[Cho and Garcia-Molina, 2003a] Cho, J. and Garcia-Molina, H. (2003a). Effective page refresh policies for web crawlers. *ACM Trans. Database Syst.*, 28(4):390–426.

[Cho and Garcia-Molina, 2003b] Cho, J. and Garcia-Molina, H. (2003b). Estimating frequency of change. *ACM Trans. Internet Techn.*, 3(3):256–290.

[Cho and Ntoulas, 2002] Cho, J. and Ntoulas, A. (2002). Effective change detection using sampling. In [DBL, 2002], pages 514–525.

[Chomicki, 2011] Chomicki, J. (2011). Logical foundations of preference queries. *IEEE Data Eng. Bull.*, 34(2):3–10.

[Chomicki and Song, 2005] Chomicki, J. and Song, J. (2005). Monotonic and nonmonotonic preference revision. *CoRR*, abs/cs/0503092.

[Christine Golbreich and Evan K. Wallace and Peter F. Patel-Schneider, 2012] Christine Golbreich and Evan K. Wallace and Peter F. Patel-Schneider (2012). Owl 2 web ontology language new features and rationale (second edition). Technical report, W3C Recommendation.

[Cimiano et al., 2013] Cimiano, P., Corcho, Ó., Presutti, V., Hollink, L., and Rudolph, S., editors (2013). *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, volume 7882 of *Lecture Notes in Computer Science*. Springer.

[Croome, 2000] Croome, C. (2000). Rdf site summary 1.0 modules: Qualified dublin core.

[Cruz et al., 2006] Cruz, I. F., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., and Aroyo, L., editors (2006). *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*. Springer.

[Cudré-Mauroux et al., 2012] Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J. X., Hendler, J., Schreiber, G., Bernstein, A., and Blomqvist, E., editors (2012). *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I*, volume 7649 of *Lecture Notes in Computer Science*. Springer.

[Cui and Widom, 2000] Cui, Y. and Widom, J. (2000). Practical lineage tracing in data warehouses. In *ICDE*, pages 367–378.

[Cui et al., 2000] Cui, Y., Widom, J., and Wiener, J. L. (2000). Tracing the lineage of view data in a warehousing environment. *ACM Trans. Database Syst.*, 25(2):179–227.

[Cyganiak and Reynolds, 2014] Cyganiak, R. and Reynolds, D. (2014). The rdf data cube vocabulary). W3c recommendation, W3C.

[da Silva et al., 2006] da Silva, P. P., McGuinness, D. L., and Fikes, R. (2006). A proof markup language for semantic web services. *Inf. Syst.*, 31(4-5):381–395.

[Damásio et al., 2012] Damásio, C. V., Analyti, A., and Antoniou, G. (2012). Provenance for SPARQL queries. In [Cudré-Mauroux et al., 2012], pages 625–640.

[Dan Brickley and R.V. Guha, 2014] Dan Brickley and R.V. Guha (2014). Rdf vocabulary description language 1.0: Rdf schema. Technical report, W3C Recommendation.

[Davidson et al., 2011a] Davidson, S. B., Bao, Z., and Roy, S. (2011a). Hiding data and structure in workflow provenance. In Kikuchi, S., Madaan, A., Sachdeva, S., and Bhalla, S., editors, *Databases in Networked Information Systems - 7th International Workshop, DNIS 2011, Aizu-Wakamatsu, Japan, December 12-14, 2011. Proceedings*, volume 7108 of *Lecture Notes in Computer Science*, pages 41–48. Springer.

[Davidson and Freire, 2008] Davidson, S. B. and Freire, J. (2008). Provenance and scientific workflows: challenges and opportunities. In [Wang, 2008], pages 1345–1350.

[Davidson et al., 2011b] Davidson, S. B., Khanna, S., Roy, S., Stoyanovich, J., Tannen, V., and Chen, Y. (2011b). On provenance and privacy. In Milo, T., editor, *Database Theory - ICDT 2011, 14th International Conference, Uppsala, Sweden, March 21-24, 2011, Proceedings*, pages 3–10. ACM.

[Davidson et al., 2011c] Davidson, S. B., Khanna, S., Tannen, V., Roy, S., Chen, Y., Milo, T., and Stoyanovich, J. (2011c). Enabling privacy in provenance-aware workflow systems. In *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings*, pages 215–218. www.cidrdb.org.

[Davies et al., 2012] Davies, J., Narodytska, N., and Walsh, T. (2012). Eliminating the weakest link: Making manipulation intractable? *CoRR*, abs/1204.3918.

[Dayal et al., 2006] Dayal, U., Whang, K., Lomet, D. B., Alonso, G., Lohman, G. M., Kersten, M. L., Cha, S. K., and Kim, Y., editors (2006). *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*. ACM.

[Deborah L. McGuinness and Frank van Harmelen, 2004] Deborah L. McGuinness and Frank van Harmelen (2004). Owl 2 web ontology language. Technical report, W3C Recommendation.

[Dehghanzadeh et al., 2014] Dehghanzadeh, S., Parreira, J. X., Karnstedt, M., Umbrich, J., Hauswirth, M., and Decker, S. (2014). Optimizing SPARQL query processing on dynamic and static data based on query time/freshness requirements using materialization. In Supnithi, T., Yamaguchi, T., Pan, J. Z., Wuwongse, V., and Buranarach, M., editors, *Semantic Technology - 4th Joint International Conference, JIST 2014, Chiang Mai, Thailand, November 9-11, 2014. Revised Selected Papers*, volume 8943 of *Lecture Notes in Computer Science*, pages 257–270. Springer.

[Deutch et al., 2015a] Deutch, D., Gilad, A., and Moskovitch, Y. (2015a). selp: Selective tracking and presentation of data provenance. In [Gehrke et al., 2015], pages 1484–1487.

[Deutch et al., 2015b] Deutch, D., Gilad, A., and Moskovitch, Y. (2015b). Towards web-scale how-provenance. In *31st IEEE International Conference on Data Engineering Workshops, ICDE Workshops 2015, Seoul, South Korea, April 13-17, 2015*, pages 68–70. IEEE Computer Society.

[Deutch et al., 2015c] Deutch, D., Moskovitch, Y., and Tannen, V. (2015c). Provenance-based analysis of data-centric processes. *The VLDB Journal*, 24(4):583–607.

[Ding and Finin, 2006] Ding, L. and Finin, T. (2006). Characterizing the semantic web on the web. In [Cruz et al., 2006], pages 242–257.

[Ding et al., 2005] Ding, L., Kolari, P., Finin, T., Joshi, A., Peng, Y., and Yesha, Y. (2005). On homeland security and the semantic web: A provenance and trust aware inference framework. In *AI Technologies for Homeland Security, Papers from the 2005 AAAI Spring Symposium, Technical Report SS-05-01, Stanford, California, USA, March 21-23, 2005*, pages 157–160. AAAI.

[Dividino et al., 2015] Dividino, R. Q., Gottron, T., and Scherp, A. (2015). Strategies for efficiently keeping local linked open data caches up-to-date. In Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P. T., Dumontier, M., Heflin, J., Thirunarayan, K., and Staab, S., editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, volume 9367 of *Lecture Notes in Computer Science*, pages 356–373. Springer.

[Dividino et al., 2014a] Dividino, R. Q., Gottron, T., Scherp, A., and Gröner, G. (2014a). From changes to dynamics: Dynamics analysis of linked open data sources. In Demidova, E., Dietze, S., Szymanski, J., and Breslin, J. G., editors, *Proceedings of the 1st International Workshop on Dataset PROFIling & fEderated Search for Linked Data co-located with the 11th Extended Semantic Web Conference, PROFILES@ESWC 2014, Anissaras, Crete, Greece, May 26, 2014.*, volume 1151 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Dividino et al., 2012] Dividino, R. Q., Gröner, G., Scheglmann, S., and Thimm, M. (2012). Ranking RDF with provenance via preference aggregation. In [ten Teije et al., 2012], pages 154–163.

[Dividino et al., 2014b] Dividino, R. Q., Kramer, A., and Gottron, T. (2014b). An investigation of HTTP header information for detecting changes of linked open data sources. In Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., and Tordai, A., editors, *The Semantic Web: ESWC 2014 Satellite Events - ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers*, volume 8798 of *Lecture Notes in Computer Science*, pages 199–203. Springer.

[Dividino et al., 2009a] Dividino, R. Q., Schenk, S., Sizov, S., and Staab, S. (2009a). Provenance, trust, explanations - and all that other meta knowledge. *KI*, 23(2):24–30.

[Dividino et al., 2013] Dividino, R. Q., Scherp, A., Gröner, G., and Grotton, T. (2013). Change-a-lod: Does the schema on the linked data cloud change or not? In Hartig, O., Sequeda, J., Hogan, A., and Matsutsuka, T., editors, *Proceedings of the Fourth International Workshop on Consuming Linked Data, COLD 2013, Sydney, Australia, October 22, 2013*, volume 1034 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Dividino et al., 2009b] Dividino, R. Q., Sizov, S., Staab, S., and Schueler, B. (2009b). Querying for provenance, trust, uncertainty and other meta knowledge in RDF. *J. Web Sem.*, 7(3):204–219.

[Doherty et al., 2006] Doherty, P., Mylopoulos, J., and Welty, C. A., editors (2006). *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*. AAAI Press.

[Dong et al., 2009] Dong, X. L., Berti-Equille, L., and Srivastava, D. (2009). Integrating conflicting data: The role of source dependence. *PVLDB*, 2(1):550–561.

[Dragan et al., 2015] Dragan, L., Luczak-Rösch, M., Simperl, E., Packer, H. S., Moreau, L., and Berendt, B. (2015). A-posteriori provenance-enabled linking of publications and datasets via crowdsourcing. *D-Lib Magazine*, 21(1/2).

[Durst and Suignard, 2005] Durst, M. and Suignard, M. (2005). Rfc 3987, internationalized resource identifiers (iris). Technical report, Network Working Group.

[Eckert et al., 2014] Eckert, K., Ritze, D., Baierer, K., and Bizer, C. (2014). Restful open workflows for data provenance and reuse. In Chung, C., Broder, A. Z., Shim, K., and Suel, T., editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014, Companion Volume*, pages 259–260. ACM.

[Fetterly et al., 2004] Fetterly, D., Manasse, M., Najork, M., and Wiener, J. L. (2004). A large-scale study of the evolution of web pages. *Softw., Pract. Exper.*, 34(2):213–237.

[Fielding et al., 1999] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext transfer protocol–http/1.1.

[Flöck and Acosta, 2014] Flöck, F. and Acosta, M. (2014). Wikiwho: Precise and efficient attribution of authorship of revisioned content. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 843–854, New York, NY, USA. ACM.

[Flouris et al., 2009] Flouris, G., Fundulaki, I., Pediaditis, P., Theoharis, Y., and Christophides, V. (2009). Coloring RDF triples to capture provenance. In [Bernstein et al., 2009], pages 196–212.

[Fokoue et al., 2010] Fokoue, A., Srivatsa, M., and Young, R. (2010). Assessing trust in uncertain information. In [Patel-Schneider et al., 2010], pages 209–224.

[Foster et al., 2008] Foster, J. N., Green, T. J., and Tannen, V. (2008). Annotated xml: Queries and provenance. In *Proceedings of the Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '08, pages 271–280, New York, NY, USA. ACM.

[Freire, 2009] Freire, J. (2009). Provenance management: Challenges and opportunities. In Freytag, J. C., Ruf, T., Lehner, W., and Vossen, G., editors, *Datenbanksysteme in Business, Technologie und Web (BTW 2009), 13. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), Proceedings, 2.-6. März 2009, Münster, Germany*, volume 144 of *LNI*, page 4. GI.

[Fuhr and Rölleke, 1997] Fuhr, N. and Rölleke, T. (1997). A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66.

[Geerts et al., 2013] Geerts, F., Karvounarakis, G., Christophides, V., and Fundulaki, I. (2013). Algebraic structures for capturing the provenance of SPARQL queries. In Tan, W., Guerrini, G., Catania, B., and Gounaris, A., editors, *Joint 2013 EDBT/ICDT Conferences, ICDT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, pages 153–164. ACM.

[Geerts et al., 2016] Geerts, F., Unger, T., Karvounarakis, G., Fundulaki, I., and Christophides, V. (2016). Algebraic structures for capturing the provenance of SPARQL queries. *J. ACM*, 63(1):7.

[Gehrke et al., 2015] Gehrke, J., Lehner, W., Shim, K., Cha, S. K., and Lohman, G. M., editors (2015). *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*. IEEE Computer Society.

[Golbeck and Halaschek-Wiener, 2009] Golbeck, J. and Halaschek-Wiener, C. (2009). Trust-based revision for expressive web syndication. *J. Log. Comput.*, 19(5):771–790.

[Golbeck and Hendler, 2004] Golbeck, J. and Hendler, J. A. (2004). Accuracy of metrics for inferring trust and reputation in semantic web-based social networks. In Motta, E., Shadbolt, N., Stutt, A., and Gibbins, N., editors, *Engineering Knowledge in the Age of the Semantic Web, 14th International Conference, EKAW 2004, Whittlebury Hall, UK, October 5-8, 2004, Proceedings*, volume 3257 of *Lecture Notes in Computer Science*, pages 116–131. Springer.

[Görlitz and Staab, 2011] Görlitz, O. and Staab, S. (2011). SPLENDID: SPARQL endpoint federation exploiting VOID descriptions. In Hartig, O., Harth, A., and Sequeda, J., editors, *Proceedings of the Second International Workshop on Consuming Linked Data (COLD2011), Bonn, Germany, October 23, 2011*, volume 782 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Gottron and Gottron, 2014] Gottron, T. and Gottron, C. (2014). Perplexity of index models over evolving linked data. In Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., and Tordai, A., editors, *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, volume 8465 of *Lecture Notes in Computer Science*, pages 161–175. Springer.

[Gottron et al., 2013a] Gottron, T., Knauf, M., Scheglmann, S., and Scherp, A. (2013a). A systematic investigation of explicit and implicit schema information on the linked open data cloud. In [Cimiano et al., 2013], pages 228–242.

[Gottron et al., 2013b] Gottron, T., Scherp, A., Krayer, B., and Peters, A. (2013b). Lodatio: using a schema-level index to support users infinding relevant sources of linked data. In Benjamins, V. R., d'Aquin, M., and Gordon, A., editors, *Proceedings of the 7th International Conference on Knowledge Capture, K-CAP 2013, Banff, Canada, June 23-26, 2013*, pages 105–108. ACM.

[Green et al., 2007] Green, T. J., Karvounarakis, G., and Tannen, V. (2007). Provenance semirings. In Libkin, L., editor, *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 31–40. ACM.

[Groth and Moreau, 2013] Groth, P. and Moreau, L. (2013). Prov-overview. Technical report, W3C Recommendation.

[Gueroussova et al., 2013] Gueroussova, M., Polleres, A., and McIlraith, S. A. (2013). SPARQL with qualitative and quantitative preferences. In Celino, I., Valle, E. D., Krötzsch, M., and Schlobach, S., editors, *Proceedings of the 2nd International Workshop on Ordering and Reasoning, OrdRing 2013, Co-located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 22nd, 2013*, volume 1059 of *CEUR Workshop Proceedings*, pages 2–8. CEUR-WS.org.

[Guo et al., 2005] Guo, Y., Pan, Z., and Heflin, J. (2005). LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3):158–182.

[Gutierrez et al., 2011] Gutierrez, C., Hurtado, C. A., Mendelzon, A. O., and Pérez, J. (2011). Foundations of semantic web databases. *J. Comput. Syst. Sci.*, 77(3):520–541.

[Gutiérrez et al., 2005] Gutiérrez, C., Hurtado, C. A., and Vaisman, A. A. (2005). Temporal RDF. In Gómez-Pérez, A. and Euzenat, J., editors, *The Semantic Web: Research and*

Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings, volume 3532 of *Lecture Notes in Computer Science*, pages 93–107. Springer.

[Gutierrez et al., 2007] Gutierrez, C., Hurtado, C. A., and Vaisman, A. A. (2007). Introducing time into RDF. *IEEE Trans. Knowl. Data Eng.*, 19(2):207–218.

[Gutiérrez-Basulto and Klarman, 2012] Gutiérrez-Basulto, V. and Klarman, S. (2012). Towards a unifying approach to representing and querying temporal data in description logics. In *Proceedings of the 6th International Conference on Web Reasoning and Rule Systems*, RR'12, pages 90–105, Berlin, Heidelberg. Springer-Verlag.

[Gyöngyi et al., 2004] Gyöngyi, Z., Garcia-Molina, H., and Pedersen, J. O. (2004). Combating web spam with trustrank. In Nascimento, M. A., Özsu, M. T., Kossmann, D., Miller, R. J., Blakeley, J. A., and Schiefer, K. B., editors, *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 576–587. Morgan Kaufmann.

[Halpin, 2009] Halpin, H. (2009). Provenance: The Missing Component of the Semantic Web. In *Proceedings of the First Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009)*.

[Halpin and Cheney, 2014] Halpin, H. and Cheney, J. (2014). Dynamic provenance for SPARQL updates. In [Mika et al., 2014], pages 425–440.

[Hansson, 2002] Hansson, S. O. (2002). *Preference Logic*, pages 319–393. Springer Netherlands, Dordrecht.

[Hartig, 2009a] Hartig, O. (2009a). Provenance information in the web of data. In Bizer, C., Heath, T., Berners-Lee, T., and Idehen, K., editors, *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009.*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Hartig, 2009b] Hartig, O. (2009b). Querying trust in RDF data with tsparql. In [Aroyo et al., 2009], pages 5–20.

[Hartig, 2011] Hartig, O. (2011). Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In Antoniou, G., Grobelnik, M., Simperl, E. P. B., Parsia, B., Plexousakis, D., Leenheer, P. D., and Pan, J. Z., editors, *The Semantic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I*, volume 6643 of *Lecture Notes in Computer Science*, pages 154–169. Springer.

[Hartig, 2013] Hartig, O. (2013). SQUIN: a traversal based query execution system for the web of linked data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 1081–1084.

[Hausenblas et al., 2009] Hausenblas, M., Halb, W., Raimond, Y., Feigenbaum, L., and Ayers, D. (2009). SCOVO: using statistics on the web of data. In [Aroyo et al., 2009], pages 708–722.

[Haveliwala, 2003] Haveliwala, T. H. (2003). Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Trans. Knowl. Data Eng.*, 15(4):784–796.

[Hoekstra and Groth, 2015] Hoekstra, R. and Groth, P. (2015). *PROV-O-Viz - Understanding the Role of Activities in Provenance*, pages 215–220. Springer International Publishing, Cham.

[Hoekstra and Groth, 2014] Hoekstra, R. and Groth, P. T. (2014). Prov-o-viz - understanding the role of activities in provenance. In [Ludäscher and Plale, 2015], pages 215–220.

[Hogan, 2014] Hogan, A. (2014). Linked data & the semantic web standards. In Harth, A., Hose, K., and Schenkel, R., editors, *Linked Data Management.*, pages 3–48. Chapman and Hall/CRC.

[Hogan et al., 2012] Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., and Decker, S. (2012). An empirical survey of linked data conformance. *J. Web Sem.*, 14:14–44.

[Horridge et al., 2008] Horridge, M., Parsia, B., and Sattler, U. (2008). Laconic and precise justifications in OWL. In [Sheth et al., 2008], pages 323–338.

[Horrocks et al., 2005] Horrocks, I., Kutz, O., and Sattler, U. (2005). The irresistible $\mathcal{SRIQ}$. In *Proc. of the 1st Int. Workshop on OWL Experiences and Directions (OWLED 2005)*.

[Horrocks et al., 2006] Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible SROIQ. In [Doherty et al., 2006], pages 57–67.

[Hristidis et al., 2001] Hristidis, V., Koudas, N., and Papakonstantinou, Y. (2001). PREFER: A system for the efficient execution of multi-parametric ranked queries. In Mehrotra, S. and Sellis, T. K., editors, *Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001*, pages 259–270. ACM.

[Huai et al., 2008] Huai, J., Chen, R., Hon, H., Liu, Y., Ma, W., Tomkins, A., and Zhang, X., editors (2008). *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*. ACM.

[Huang et al., 2015] Huang, X., Bao, Z., Davidson, S. B., Milo, T., and Yuan, X. (2015). Answering regular path queries on workflow provenance. In [Gehrke et al., 2015], pages 375–386.

[Huynh et al., 2013] Huynh, T. D., Ebden, M., Venanzi, M., Ramchurn, S. D., Roberts, S. J., and Moreau, L. (2013). Interpretation of crowdsourced activities using provenance network analysis. In Hartman, B. and Horvitz, E., editors, *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2013, November 7-9, 2013, Palm Springs, CA, USA*. AAAI.

[Huynh et al., 2016] Huynh, T. D., Michaelides, D. T., and Moreau, L. (2016). PROV-JSONLD: A JSON and linked data representation for provenance. In [Mattoso and Glavic, 2016], pages 173–177.

[Huynh and Moreau, 2014] Huynh, T. D. and Moreau, L. (2014). Provstore: A public provenance repository. In [Ludäscher and Plale, 2015], pages 275–277.

[Isele et al., 2010] Isele, R., Umbrich, J., Bizer, C., and Harth, A. (2010). Ldspider: An open-source crawling framework for the web of linked data. In Polleres, A. and Chen, H., editors, *Proceedings of the ISWC 2010 Posters & Demonstrations Track: Collected Abstracts, Shanghai, China, November 9, 2010*, volume 658 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Jeh and Widom, 2003] Jeh, G. and Widom, J. (2003). Scaling personalized web search. In Hencsey, G., White, B., Chen, Y. R., Kovács, L., and Lawrence, S., editors, *Proceedings of*

the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20-24, 2003, pages 271–279. ACM.

[Jentzsch et al., 2011] Jentzsch, A., Cyganiak, R., and Bizer, C. (2011). State of the LOD Cloud. `http://www4.wiwiss.fu-berlin.de/lodcloud/state/`. [Online; accessed 13-April-2016].

[Jentzsch et al., 2015] Jentzsch, A., Dullweber, C., Troiano, P., and Naumann, F. (2015). Exploring linked data graph structures. In [Villata et al., 2015].

[Ji et al., 2009] Ji, Q., Qi, G., and Haase, P. (2009). A relevance-directed algorithm for finding justifications of DL entailments. In Gómez-Pérez, A., Yu, Y., and Ding, Y., editors, *The Semantic Web, Fourth Asian Conference, ASWC 2009, Shanghai, China, December 6-9, 2009. Proceedings*, volume 5926 of *Lecture Notes in Computer Science*, pages 306–320. Springer.

[Jung and Lutz, 2012] Jung, J. C. and Lutz, C. (2012). Ontology-based access to probabilistic data with OWL QL. In [Cudré-Mauroux et al., 2012], pages 182–197.

[Käfer et al., 2013] Käfer, T., Abdelrahman, A., Umbrich, J., O'Byrne, P., and Hogan, A. (2013). Observing linked data dynamics. In [Cimiano et al., 2013], pages 213–227.

[Kalyanpur et al., 2006] Kalyanpur, A., Parsia, B., and Grau, B. C. (2006). Beyond asserted axioms: Fine-grain justifications for OWL-DL entailments. In Parsia, B., Sattler, U., and Toman, D., editors, *Proceedings of the 2006 International Workshop on Description Logics (DL2006), Windermere, Lake District, UK, May 30 - June 1, 2006*, volume 189 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Kalyanpur et al., 2007] Kalyanpur, A., Parsia, B., Horridge, M., and Sirin, E. (2007). Finding all justifications of OWL DL entailments. In Aberer, K., Choi, K., Noy, N. F., Allemang, D., Lee, K., Nixon, L. J. B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., and Cudré-Mauroux, P., editors, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, volume 4825 of *Lecture Notes in Computer Science*, pages 267–280. Springer.

[Karvounarakis et al., 2013] Karvounarakis, G., Fundulaki, I., and Christophides, V. (2013). Provenance for linked data. In [Tannen et al., 2013], pages 366–381.

[Karvounarakis and Green, 2012] Karvounarakis, G. and Green, T. J. (2012). Semiring-annotated data: queries and provenance? *SIGMOD Record*, 41(3):5–14.

[Keith Alexander and Richard Cyganiak and Michael Hausenblas and Jun Zhao, 2011] Keith Alexander and Richard Cyganiak and Michael Hausenblas and Jun Zhao (2011). Describing linked datasets with the void vocabulary. Technical report, W3C.

[Kelly, 1988] Kelly, J. (1988). *Social choice theory: an introduction*. Springer-Verlag.

[Keshavarz et al., 2014] Keshavarz, A. S., Huynh, T. D., and Moreau, L. (2014). Provenance for online decision making. In [Ludäscher and Plale, 2015], pages 44–55.

[Khatchadourian and Consens, 2010] Khatchadourian, S. and Consens, M. P. (2010). Explod: Summary-based exploration of interlinking and RDF usage in the linked open data cloud. In Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., and Tudorache, T., editors, *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010,*

Proceedings, Part II, volume 6089 of *Lecture Notes in Computer Science*, pages 272–287. Springer.

[Kießling, 2002] Kießling, W. (2002). Foundations of preferences in database systems. In [DBL, 2002], pages 311–322.

[Kjernsmo, 2015] Kjernsmo, K. (2015). A survey of HTTP caching implementations on the open semantic web. In Gandon, F., Sabou, M., Sack, H., d'Amato, C., Cudré-Mauroux, P., and Zimmermann, A., editors, *The Semantic Web. Latest Advances and New Domains - 12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31 - June 4, 2015. Proceedings*, volume 9088 of *Lecture Notes in Computer Science*, pages 286–301. Springer.

[Klyne et al., 2014] Klyne, G., Carroll, J. J., and McBride, B. (2014). Rdf 1.1 concepts and abstract syntax. Technical report, W3C Recommendation.

[Knechtel and Peñaloza, 2010] Knechtel, M. and Peñaloza, R. (2010). A generic approach for correcting access restrictions to a consequence. In [Aroyo et al., 2010], pages 167–182.

[Konrath et al., 2012] Konrath, M., Gottron, T., Staab, S., and Scherp, A. (2012). Schemex - efficient construction of a data catalogue by stream-based indexing of linked data. *J. Web Sem.*, 16:52–58.

[Koop and Freire, 2014] Koop, D. and Freire, J. (2014). Reorganizing workflow evolution provenance. In Chapman, A., Ludäscher, B., and Schreiber, A., editors, *6th Workshop on the Theory and Practice of Provenance, TaPP'14, Cologne, Germany, June 12-13, 2014*. USENIX Association.

[Koop et al., 2010] Koop, D., Scheidegger, C. E., Freire, J., and Silva, C. T. (2010). The provenance of workflow upgrades. In McGuinness, D. L., Michaelis, J., and Moreau, L., editors, *Provenance and Annotation of Data and Processes - Third International Provenance and Annotation Workshop, IPAW 2010, Troy, NY, USA, June 15-16, 2010. Revised Selected Papers*, volume 6378 of *Lecture Notes in Computer Science*, pages 2–16. Springer.

[Kostylev and Buneman, 2012] Kostylev, E. V. and Buneman, P. (2012). Combining dependent annotations for relational algebra. In Deutsch, A., editor, *15th International Conference on Database Theory, ICDT '12, Berlin, Germany, March 26-29, 2012*, pages 196–207. ACM.

[Kuich, 1997] Kuich, W. (1997). Handbook of formal languages, vol. 1. chapter Semirings and Formal Power Series: Their Relevance to Formal Languages and Automata, pages 609–677. Springer-Verlag New York, Inc., New York, NY, USA.

[Kwasnikowska et al., 2015] Kwasnikowska, N., Moreau, L., and den Bussche, J. V. (2015). A formal account of the open provenance model. *TWEB*, 9(2):10.

[Lakshmanan et al., 2011] Lakshmanan, G. T., Curbera, F., Freire, J., and Sheth, A. P. (2011). Guest editors' introduction: Provenance in web applications. *IEEE Internet Computing*, 15(1):17–21.

[Lampo et al., 2011] Lampo, T., Vidal, M.-E., Danilow, J., and Ruckhaus, E. (2011). To cache or not to cache: The effects of warming cache in complex sparql queries. In *Proceedings of the 2011th Confederated International Conference on On the Move to Meaningful Internet Systems - Volume Part II*, OTM'11, pages 716–733, Berlin, Heidelberg. Springer-Verlag.

[Langegger and Wöß, 2009] Langegger, A. and Wöß, W. (2009). Rdfstats - an extensible RDF statistics generator and library. In Tjoa, A. M. and Wagner, R., editors, *Database and*

Expert Systems Applications, DEXA, International Workshops, Linz, Austria, August 31-September 4, 2009, Proceedings, pages 79–83. IEEE Computer Society.

[Lašek et al., 2012] Lašek, I., Klimpera, O., and Vojtáš, P. (2012). Scalable Framework for Semantic Web Crawling. In *WIKT 2012: 7th Workshop on Intelligent and Knowledge Oriented Technologies*, pages 165–168, Bratislava. Slovenská technická univerzita v Bratislave.

[Li et al., 2005] Li, C., Chang, K. C., Ilyas, I. F., and Song, S. (2005). Ranksql: Query algebra and optimization for relational top-k queries. In Özcan, F., editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pages 131–142. ACM.

[Lopes et al., 2010] Lopes, N., Polleres, A., Straccia, U., and Zimmermann, A. (2010). Anql: Sparqling up annotated RDFS. In [Patel-Schneider et al., 2010], pages 518–533.

[Ludäscher and Plale, 2015] Ludäscher, B. and Plale, B., editors (2015). *Provenance and Annotation of Data and Processes - 5th International Provenance and Annotation Workshop, IPAW 2014, Cologne, Germany, June 9-13, 2014. Revised Selected Papers*, volume 8628 of *Lecture Notes in Computer Science*. Springer.

[Lukasiewicz and Straccia, 2008] Lukasiewicz, T. and Straccia, U. (2008). Managing uncertainty and vagueness in description logics for the semantic web. *J. Web Sem.*, 6(4):291–308.

[Lutz et al., 2008] Lutz, C., Wolter, F., and Zakharyaschev, M. (2008). Temporal description logics: A survey. In Demri, S. and Jensen, C. S., editors, *15th International Symposium on Temporal Representation and Reasoning, TIME 2008, Université du Québec à Montréal, Canada, 16-18 June 2008*, pages 3–14. IEEE Computer Society.

[Ma et al., 2007] Ma, Y., Hitzler, P., and Lin, Z. (2007). Algorithms for paraconsistent reasoning with OWL. In Franconi, E., Kifer, M., and May, W., editors, *The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007, Proceedings*, volume 4519 of *Lecture Notes in Computer Science*, pages 399–413. Springer.

[Maali and Erickson, 2014] Maali, F. and Erickson, J. (2014). Data catalog vocabulary (dcat). W3c recommendation, W3C.

[Manola et al., 2014] Manola, F., Miller, E., and McBride, B. (2014). Rdf 1.1 primer. Technical report, W3C Recommendation.

[Mattoso and Glavic, 2016] Mattoso, M. and Glavic, B., editors (2016). *Provenance and Annotation of Data and Processes - 6th International Provenance and Annotation Workshop, IPAW 2016, McLean, VA, USA, June 7-8, 2016, Proceedings*, volume 9672 of *Lecture Notes in Computer Science*. Springer.

[Maudet et al., 2012] Maudet, N., Pini, M. S., Venable, K. B., and Rossi, F. (2012). Influence and aggregation of preferences over combinatorial domains. In van der Hoek, W., Padgham, L., Conitzer, V., and Winikoff, M., editors, *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)*, pages 1313–1314. IFAAMAS.

[McGuinness and da Silva, 2004] McGuinness, D. L. and da Silva, P. P. (2004). Explaining answers from the semantic web: the inference web approach. *J. Web Sem.*, 1(4):397–413.

[Mihindukulasooriya et al., 2015] Mihindukulasooriya, N., Poveda-Villalón, M., García-Castro, R., and Gómez-Pérez, A. (2015). Loupe - an online tool for inspecting datasets in the linked data cloud. In [Villata et al., 2015].

[Mika et al., 2014] Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C. A., Vrandecic, D., Groth, P. T., Noy, N. F., Janowicz, K., and Goble, C. A., editors (2014). *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, volume 8796 of *Lecture Notes in Computer Science*. Springer.

[Miles et al., 2013] Miles, S., Trim, C. M., and Panzer, M. (2013). Dublin core to prov mapping. Technical report, W3C Group Note.

[Missier et al., 2013] Missier, P., Moreau, L., Cheney, J., Lebo, T., and Soiland-Reyes, S. (2013). Prov-dictionary: Modeling provenance for dictionary data structures. Technical report, W3C Group Note.

[Moreau, 2010] Moreau, L. (2010). The foundations for provenance on the web. *Foundations and Trends in Web Science*, 2(2-3):99–241.

[Moreau, 2013] Moreau, L. (2013). Prov-xml: The prov xml schema. Technical report, W3C Group Note.

[Moreau, 2015] Moreau, L. (2015). Aggregation by provenance types: A technique for summarising provenance graphs. In Rensink, A. and Zambon, E., editors, *Proceedings Graphs as Models, GaM 2015, London, UK, 11-12 April 2015.*, volume 181 of *EPTCS*, pages 129–144.

[Moreau et al., 2011] Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P. T., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E. G., and den Bussche, J. V. (2011). The open provenance model core specification (v1.1). *Future Generation Comp. Syst.*, 27(6):743–756.

[Moreau and Groth, 2013] Moreau, L. and Groth, P. T. (2013). *Provenance: An Introduction to PROV*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers.

[Moreau et al., 2015] Moreau, L., Groth, P. T., Cheney, J., Lebo, T., and Miles, S. (2015). The rationale of PROV. *J. Web Sem.*, 35:235–257.

[Moreau et al., 2013] Moreau, L., Hartig, O., Simmhan, Y., Myers, J., Lebo, T., Belhajjame, K., Miles, S., and Soiland-Reyes, S. (2013). Prov-aq: Provenance access and query. Technical report, W3C Group Note.

[Motik, 2012] Motik, B. (2012). Representing and querying validity time in RDF and OWL: A logic-based approach. *J. Web Sem.*, 12:3–21.

[Mouratidis et al., 2006] Mouratidis, K., Bakiras, S., and Papadias, D. (2006). Continuous monitoring of top-k queries over sliding windows. In [Chaudhuri et al., 2006], pages 635–646.

[Murdock et al., 2006] Murdock, J. W., McGuinness, D. L., da Silva, P. P., Welty, C. A., and Ferrucci, D. A. (2006). Explaining conclusions from diverse knowledge sources. In [Cruz et al., 2006], pages 861–872.

[Murta et al., 2014] Murta, L., Braganholo, V., Chirigati, F., Koop, D., and Freire, J. (2014). noworkflow: Capturing and analyzing provenance of scripts. In [Ludäscher and Plale, 2015], pages 71–83.

[Neumann and Moerkotte, 2011] Neumann, T. and Moerkotte, G. (2011). Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In Abiteboul, S., Böhm, K., Koch, C., and Tan, K., editors, *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, pages 984–994. IEEE Computer Society.

[Nies, 2013] Nies, T. D. (2013). Constraints of the prov data model. Technical report, W3C Recommendation.

[Oinn et al., 2004] Oinn, T. M., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, R. M., Carver, T., Glover, K., Pocock, M. R., Wipat, A., and Li, P. (2004). Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054.

[Olston and Pandey, 2008] Olston, C. and Pandey, S. (2008). Recrawl scheduling based on information longevity. In [Huai et al., 2008], pages 437–446.

[Orlandi and Passant, 2011] Orlandi, F. and Passant, A. (2011). Modelling provenance of dbpedia resources using wikipedia contributions. *J. Web Sem.*, 9(2):149–164.

[Page et al., 1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

[Pahlevan et al., 2015] Pahlevan, A., Duprat, J.-L., Thomo, A., and Müller, H. (2015). Dynamis: Effective context-aware web service selection using dynamic attributes. *Future Internet*, 7(2):110.

[Palmonari et al., 2015] Palmonari, M., Rula, A., Porrini, R., Maurino, A., Spahiu, B., and Ferme, V. (2015). ABSTAT: linked data summaries with abstraction and statistics. In Gandon, F., Guéret, C., Villata, S., Breslin, J. G., Faron-Zucker, C., and Zimmermann, A., editors, *The Semantic Web: ESWC 2015 Satellite Events - ESWC 2015 Satellite Events Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers*, volume 9341 of *Lecture Notes in Computer Science*, pages 128–132. Springer.

[Pandey and Olston, 2005] Pandey, S. and Olston, C. (2005). User-centric web crawling. In Ellis, A. and Hagino, T., editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*, pages 401–411. ACM.

[Patel-Schneider et al., 2010] Patel-Schneider, P. F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J. Z., Horrocks, I., and Glimm, B., editors (2010). *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, volume 6496 of *Lecture Notes in Computer Science*. Springer.

[Patrick J. Hayes and Peter F. Patel-Schneider, 2014] Patrick J. Hayes and Peter F. Patel-Schneider (2014). Rdf 1.1 semantics. Technical report, W3C Recommendation.

[Pérez et al., 2009] Pérez, J., Arenas, M., and Gutierrez, C. (2009). Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3).

[Pini et al., 2005] Pini, M. S., Rossi, F., Venable, K. B., and Walsh, T. (2005). Aggregating partially ordered preferences: impossibility and possibility results. In van der Meyden, R., editor, *Proceedings of the 10th Conference on Theoretical Aspects of Rationality and*

Knowledge (TARK-2005), Singapore, June 10-12, 2005, pages 193–206. National University of Singapore.

[Polleres, 2007] Polleres, A. (2007). From SPARQL to rules (and back). In [Williamson et al., 2007], pages 787–796.

[Polleres et al., 2007] Polleres, A., Scharffe, F., and Schindlauer, R. (2007). SPARQL++ for mapping between RDF vocabularies. In Meersman, R. and Tari, Z., editors, *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part I*, volume 4803 of *Lecture Notes in Computer Science*, pages 878–896. Springer.

[Prud'hommeaux and Seaborne, 2008] Prud'hommeaux, E. and Seaborne, A. (2008). Sparql query language for rdf. W3c recommendation, W3C.

[Qi et al., 2011] Qi, G., Ji, Q., Pan, J. Z., and Du, J. (2011). Extending description logics with uncertainty reasoning in possibilistic logic. *Int. J. Intell. Syst.*, 26(4):353–381.

[R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, T. Berners-Lee, 1999] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, T. Berners-Lee (1999). Hypertext transfer protocol – http/1.1. Technical report, W3C Recommendation.

[Radinsky and Bennett, 2013] Radinsky, K. and Bennett, P. N. (2013). Predicting content change on the web. In Leonardi, S., Panconesi, A., Ferragina, P., and Gionis, A., editors, *Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013*, pages 415–424. ACM.

[Reiter, 1987] Reiter, R. (1987). A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95.

[Richardson and Moreau, 2016] Richardson, D. P. and Moreau, L. (2016). Towards the domain agnostic generation of natural language explanations from provenance graphs for casual users. In [Mattoso and Glavic, 2016], pages 95–106.

[Riguzzi et al., 2015a] Riguzzi, F., Bellodi, E., Lamma, E., and Zese, R. (2015a). Probabilistic description logics under the distribution semantics. *Semantic Web*, 6(5):477–501.

[Riguzzi et al., 2015b] Riguzzi, F., Bellodi, E., Lamma, E., and Zese, R. (2015b). Reasoning with probabilistic ontologies. In Yang, Q. and Wooldridge, M., editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 4310–4316. AAAI Press.

[Roussakis et al., 2015] Roussakis, Y., Chrysakis, I., Stefanidis, K., Flouris, G., and Stavrakas, Y. (2015). A flexible framework for understanding the dynamics of evolving RDF datasets. In Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P. T., Dumontier, M., Heflin, J., Thirunarayan, K., and Staab, S., editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, volume 9366 of *Lecture Notes in Computer Science*, pages 495–512. Springer.

[Rudolph, 2011] Rudolph, S. (2011). Foundations of description logics. In Axel Polleres, Claudia d'Amato, M. A. S. H. P. K. S. O. P. F. P.-S., editor, *Reasoning Web. Semantic Technologies for the Web of Data - 7th International Summer School 2011*, volume 6848 of *LNCS*, pages 76–136. Springer.

[Rula et al., 2012] Rula, A., Palmonari, M., Harth, A., Stadtmüller, S., and Maurino, A. (2012). On the diversity and availability of temporal information in linked open data. In [Cudré-Mauroux et al., 2012], pages 492–507.

[Sarkas et al., 2008] Sarkas, N., Das, G., Koudas, N., and Tung, A. K. H. (2008). Categorical skylines for streaming data. In [Wang, 2008], pages 239–250.

[Schaible et al., 2013] Schaible, J., Gottron, T., Scheglmann, S., and Scherp, A. (2013). LOVER: support for modeling data using linked open vocabularies. In Guerrini, G., editor, *Joint 2013 EDBT/ICDT Conferences, EDBT/ICDT '13, Genoa, Italy, March 22, 2013, Workshop Proceedings*, pages 89–92. ACM.

[Schenk, 2008] Schenk, S. (2008). On the semantics of trust and caching in the semantic web. In [Sheth et al., 2008], pages 533–549.

[Schenk et al., 2009] Schenk, S., Dividino, R. Q., and Staab, S. (2009). Reasoning with provenance, trust and all that other meta knowlege in OWL. In Freire, J., Missier, P., and Sahoo, S. S., editors, *Proceedings of the First International Workshop on the role of Semantic Web in Provenance Management (SWPM 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009), Washington DC, USA, October 25, 2009*, volume 526 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Schenk et al., 2011] Schenk, S., Dividino, R. Q., and Staab, S. (2011). Using provenance to debug changing ontologies. *J. Web Sem.*, 9(3):284–298.

[Schmachtenberg et al., 2014] Schmachtenberg, M., Bizer, C., and Paulheim, H. (2014). Adoption of the linked data best practices in different topical domains. In [Mika et al., 2014], pages 245–260.

[Schueler et al., 2008] Schueler, B., Sizov, S., Staab, S., and Tran, D. T. (2008). Querying for meta knowledge. In [Huai et al., 2008], pages 625–634.

[Sensoy et al., 2013] Sensoy, M., Fokoue, A., Pan, J. Z., Norman, T. J., Tang, Y., Oren, N., and Sycara, K. P. (2013). Reasoning about uncertain information and conflict resolution through trust revision. In Gini, M. L., Shehory, O., Ito, T., and Jonker, C. M., editors, *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6-10, 2013*, pages 837–844. IFAAMAS.

[Shchekotykhin et al., 2011] Shchekotykhin, K. M., Friedrich, G., Fleiss, P., and Rodler, P. (2011). Query strategy for sequential ontology debugging. *CoRR*, abs/1107.4303.

[Sheth et al., 2008] Sheth, A. P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T. W., and Thirunarayan, K., editors (2008). *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*, volume 5318 of *Lecture Notes in Computer Science*. Springer.

[Siberski et al., 2006] Siberski, W., Pan, J. Z., and Thaden, U. (2006). Querying the semantic web with preferences. In [Cruz et al., 2006], pages 612–624.

[Simmhan et al., 2005] Simmhan, Y., Plale, B., and Gannon, D. (2005). A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36.

[Simmhan et al., 2008] Simmhan, Y. L., Plale, B., and Gannon, D. (2008). Karma2: Provenance management for data-driven workflows. *Int. J. Web Service Res.*, 5(2):1–22.

[Sizov, 2007] Sizov, S. (2007). What makes you think that? the semantic web's proof layer. *IEEE Intelligent Systems*, 22(6):94–99.

[Smith et al., 2007] Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.-A., Scheuermann, R. H., Shah, N., Whetzel, P. L., and Lewis, S. (2007). The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–5.

[Straccia et al., 2010] Straccia, U., Lopes, N., Lukacsy, G., and Polleres, A. (2010). A general framework for representing and reasoning with annotated semantic web data. In Fox, M. and Poole, D., editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press.

[T. Berners-Lee, 2005] T. Berners-Lee, R. Fielding, L. M. (2005). Rfc 3986, uniform resource identifier (uri): Generic syntax. Technical report, Network Working Group.

[Tan and Mitra, 2010] Tan, Q. and Mitra, P. (2010). Clustering-based incremental web crawling. *ACM Trans. Inf. Syst.*, 28(4):17.

[Tan, 2007] Tan, W. C. (2007). Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull.*, 30(4):3–12.

[Tannen et al., 2013] Tannen, V., Wong, L., Libkin, L., Fan, W., Tan, W., and Fourman, M. P., editors (2013). *In Search of Elegance in the Theory and Practice of Computation - Essays Dedicated to Peter Buneman*, volume 8000 of *Lecture Notes in Computer Science*. Springer.

[Tappolet and Bernstein, 2009] Tappolet, J. and Bernstein, A. (2009). Applied temporal rdf: Efficient temporal querying of rdf data with sparql. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009 Heraklion, pages 308–322, Berlin, Heidelberg. Springer-Verlag.

[ten Teije et al., 2012] ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Aquin, M., Nikolov, A., Aussenac-Gilles, N., and Hernandez, N., editors (2012). *Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings*, volume 7603 of *Lecture Notes in Computer Science*. Springer.

[Theoharis et al., 2011] Theoharis, Y., Fundulaki, I., Karvounarakis, G., and Christophides, V. (2011). On provenance of queries on semantic web data. *IEEE Internet Computing*, 15(1):31–39.

[Thimm, 2013] Thimm, M. (2013). Dynamic preference aggregation under preference changes. In *Proceedings of the Fourth Workshop on Dynamics of Knowledge and Belief (DKB'13)*.

[Tran et al., 2008] Tran, T., Haase, P., Motik, B., Grau, B. C., and Horrocks, I. (2008). Metalevel information in ontology-based applications. In Fox, D. and Gomes, C. P., editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1237–1242. AAAI Press.

[Udrea et al., 2010] Udrea, O., Recupero, D. R., and Subrahmanian, V. S. (2010). Annotated RDF. *ACM Trans. Comput. Log.*, 11(2).

[Umbrich et al., 2010] Umbrich, J., Hausenblas, M., Hogan, A., Polleres, A., and Decker, S. (2010). Towards dataset dynamics: Change frequency of linked open data sources. In Bizer, C., Heath, T., Berners-Lee, T., and Hausenblas, M., editors, *Proceedings of the WWW2010 Workshop on Linked Data on the Web, LDOW 2010, Raleigh, USA, April 27, 2010*, volume 628 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Umbrich et al., 2012] Umbrich, J., Karnstedt, M., Hogan, A., and Parreira, J. X. (2012). Hybrid SPARQL queries: Fresh vs. fast results. In [Cudré-Mauroux et al., 2012], pages 608–624.

[Villata et al., 2015] Villata, S., Pan, J. Z., and Dragoni, M., editors (2015). *Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015*, volume 1486 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Völkel and Groza, 2006] Völkel, M. and Groza, T. (2006). SemVersion: An RDF-based Ontology Versioning System. In Nunes, M. B., editor, *Proceedings of IADIS International Conference on WWW/Internet (IADIS 2006)*, pages 195–202, Murcia, Spain.

[Walsh, 2007] Walsh, T. (2007). Representing and reasoning with preferences. *AI Magazine*, 28(4):59–70.

[Wang, 2008] Wang, J. T., editor (2008). *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*. ACM.

[Wang and Madnick, 1990] Wang, Y. R. and Madnick, S. E. (1990). A polygen model for heterogeneous database systems: The source tagging perspective. In *Proceedings of the 16th International Conference on Very Large Data Bases*, VLDB '90, pages 519–538, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Wille, 2009] Wille, R. (2009). Restructuring lattice theory: An approach based on hierarchies of concepts. In Ferré, S. and Rudolph, S., editors, *Formal Concept Analysis, 7th International Conference, ICFCA 2009, Darmstadt, Germany, May 21-24, 2009, Proceedings*, volume 5548 of *Lecture Notes in Computer Science*, pages 314–339. Springer.

[Williamson et al., 2007] Williamson, C. L., Zurko, M. E., Patel-Schneider, P. F., and Shenoy, P. J., editors (2007). *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. ACM.

[Wolf et al., 2002] Wolf, J. L., Squillante, M. S., Yu, P. S., Sethuraman, J., and Ozsen, L. (2002). Optimal crawling strategies for web search engines. In Lassner, D., Roure, D. D., and Iyengar, A., editors, *Proceedings of the Eleventh International World Wide Web Conference, WWW 2002, May 7-11, 2002, Honolulu, Hawaii*, pages 136–147. ACM.

[Wu et al., 2006] Wu, B., Goel, V., and Davison, B. D. (2006). Topical trustrank: using topicality to combat web spam. In Carr, L., Roure, D. D., Iyengar, A., Goble, C. A., and Dahlin, M., editors, *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 63–72. ACM.

[Wylot et al., 2014] Wylot, M., Cudré-Mauroux, P., and Groth, P. T. (2014). Tripleprov: efficient processing of lineage queries in a native RDF store. In Chung, C., Broder, A. Z., Shim, K., and Suel, T., editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, pages 455–466. ACM.

[Wylot et al., 2015a] Wylot, M., Cudré-Mauroux, P., and Groth, P. T. (2015a). A demonstration of tripleprov: Tracking and querying provenance over web data. *PVLDB*, 8(12):1992–1995.

[Wylot et al., 2015b] Wylot, M., Cudré-Mauroux, P., and Groth, P. T. (2015b). Executing provenance-enabled queries over web data. In Gangemi, A., Leonardi, S., and Panconesi,

A., editors, *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 1275–1285. ACM.

[Xin et al., 2006] Xin, D., Han, J., Cheng, H., and Li, X. (2006). Answering top-k queries with multi-dimensional selections: The ranking cube approach. In [Dayal et al., 2006], pages 463–475.

[Zadeh, 1965] Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8(3):338 – 353.

[Zeginis et al., 2011] Zeginis, D., Tzitzikas, Y., and Christophides, V. (2011). On computing deltas of RDF/S knowledge bases. *TWEB*, 5(3):14.

[Zhang et al., 2009] Zhang, Y., Jansen, B. J., and Spink, A. (2009). Time series analysis of a web search engine transaction log. *Inf. Process. Manage.*, 45(2):230–245.

[Zhao and Hartig, 2012] Zhao, J. and Hartig, O. (2012). Towards interoperable provenance publication on the linked data web. In Bizer, C., Heath, T., Berners-Lee, T., and Hausenblas, M., editors, *WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012*, volume 937 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Zimmermann et al., 2012] Zimmermann, A., Lopes, N., Polleres, A., and Straccia, U. (2012). A general framework for representing, reasoning and querying with annotated semantic web data. *J. Web Sem.*, 11:72–95.

Curriculum Vitae

Renata Dividino

Contact

Addresse	17, Halef Crt B3NOC1, Halifax Nova Scotia, Canada
E-mail	dividino@uni-koblenz.de

Education

2004 – 2007	Universität des Saarlandes, Germany	Master in Computer Sciences
2001 – 2002	Ecole Centrale de Lyon, France	Diplome d'Option en Informatique
1999 – 2004	Universidade Estadual de Campinas , Brazil	Bachelor in Computer Sciences

Work Experience

10/2008–10/2015	Institute for Web Science and Technologies, Germany	Research Associate
05/2007–09/2008	Fraunhofer-Institut für Graphische Datenverarbeitung IGD, Germany	Research Associate
03/2005–03/2007	Deutsche Forschungszentrum für Künstliche Intelligenz, Germany	Student Associate
02/2003–07/2004	Universidade Estadual de Campinas, Brazil	Student Associate
05/2002–01/2003	Avivias (University Startup), France	Student Internship

Teaching

Term	Course	
SS 2014	Seminar *Linked Data*	Universität Koblenz-Landau
WS 2013/2014	Exercises *Advanced Data Model*	Universität Koblenz-Landau
WS 2010/2011	Exercises *Database Systems*	Universität Koblenz-Landau
WS 2010/2011	Exercises *Advanced Data Model*	Universität Koblenz-Landau
WS 2010/2011	Exercises *Iteractive Multimedia Systems*	Universität Koblenz-Landau
SS 2010	Exercises *Semantic Web (Summer Academy)*	Universität Koblenz-Landau
WS 2009/2010	Exercises *Database Systems*	Universität Koblenz-Landau
WS 2009/2010	Exercises *Advanced Data Model*	Universität Koblenz-Landau
WS 2009/2010	Exercises *Iteractive Multimedia Systems*	Universität Koblenz-Landau
SS 2009	Seminar *Control, Trust and Privacy*	Universität Koblenz-Landau

Publications

- Renata Queiroz Dividino, Thomas Gottron, Ansgar Scherp: Strategies for Efficiently Keeping Local Linked Open Data Caches Up-To-Date. International Semantic Web Conference (2) 2015: 356-373 2014
- Renata Queiroz Dividino, Thomas Gottron, Ansgar Scherp, Gerd Gröner: From Changes to Dynamics: Dynamics Analysis of Linked Open Data Sources. PROFILES@ESWC 2014
- Renata Queiroz Dividino, Andre Kramer, Thomas Gottron: An Investigation of HTTP Header Information for Detecting Changes of Linked Open Data Sources. ESWC (Satellite Events) 2014: 199-203 2013
- Renata Queiroz Dividino, Gerd Gröner: Which of the following SPARQL Queries are Similar? Why? LD4IE@ISWC 2013
- Renata Queiroz Dividino, Ansgar Scherp, Gerd Gröner, Thomas Grotton: Change-a-LOD: Does the Schema on the Linked Data Cloud Change or Not? COLD 2013
- Kasjen Kramer, Renata Queiroz Dividino, Gerd Gröner: SPACE: SPARQL Index for Efficient Autocompletion. International Semantic Web Conference (Posters & Demos) 2013: 157-160 2012

- Renata Queiroz Dividino, Gerd Gröner, Stefan Scheglmann, Matthias Thimm: Ranking RDF with Provenance via Preference Aggregation. EKAW 2012: 154-163 2011
- Simon Schenk, Renata Queiroz Dividino, Steffen Staab: Using provenance to debug changing ontologies. J. Web Sem. 9(3): 284-298 (2011) 2010
- Thomas Franz, Jörg Koch, Renata Queiroz Dividino, Steffen Staab: LENA-TR : Browsing Linked Open Data Along Knowledge-Aspects. AAAI Spring Symposium: Linked Data Meets Artificial Intelligence 2010
- Renata Queiroz Dividino, Simon Schenk, Sergej Sizov, Steffen Staab: Provenance, Trust, Explanations - and all that other Meta Knowledge. KI 23(2): 24-30 (2009)
- Renata Queiroz Dividino, Sergej Sizov, Steffen Staab, Bernhard Schueler: Querying for provenance, trust, uncertainty and other meta knowledge in RDF. J. Web Sem. 7(3): 204-219 (2009)
- Renata Queiroz Dividino, Veli Bicer, Konrad Voigt, Jorge Cardoso: Integrating business process and user interface models using a model-driven approach. ISCIS 2009: 492-497
- Simon Schenk, Renata Queiroz Dividino, Steffen Staab: Reasoning With Provenance, Trust and all that other Meta Knowlege in OWL. SWPM 2009
- Renata Queiroz Dividino, Massimo Romanelli, Daniel Sonntag: Semiotic-based Ontology Evaluation Tool (S-OntoEval). LREC 2008 2004
- Ricardo da Silva Torres, Claudia Bauzer Medeiros, Renata Queiroz Dividino, Mauricio Augusto Figueiredo, Marcos Andre Goncalves, Edward A. Fox, Ryan Richardson: Using digital library components for biodiversity systems. JCDL 2004: 408