A Graph Model for RDF

Diploma Thesis

Jonathan Hayes

Technische Universität Darmstadt
Universidad de Chile

A Graph Model for RDF

Diploma Thesis

A Graph Model for RDF

Diploma Thesis

Jonathan Hayes

Supervision: Claudio Gutierrez
Ricardo Baeza-Yates
Alejandro Buchmann

August 31, 2004

Technische Universität Darmstadt
Universidad de Chile

Abstract

The Resource Description Framework (RDF) is a language for metadata assertions about information resources on the World-Wide Web, and is thus a foundation for a future *Semantic* Web. The atomic construct of RDF are *statements*, which are triples consisting of the resource being described, a property, and a property value. A collection of RDF statements can be intuitively understood as a graph: resources are nodes and statements are arcs connecting the nodes. The graph nature of this abstract triple syntax is indeed appealing, but the RDF specification does not distinguish clearly among (1) the term of *RDF Graph* (merely a set of triples, thus not a standard graph), (2) the *mathematical concept* of graph, and (3) the graph-like *visualization* of RDF data ("node and directed-arc diagrams").

This thesis argues that there is need for an explicit graph representation for RDF, which allows the application of technics and results from graph theory and which serves as an *intermediate model* between the abstract triple syntax and task-specific serialization of RDF data. Directed labeled graphs currently used by default suffer from an ambiguous definition and, furthermore, have limitations inherent in any approach representing RDF triple statements by essentially binary (although labeled) edges.

As an alternative, it is natural to consider hypergraphs with ternary edges; from this, we derive *RDF bipartite graphs* as an intermediate graph-based model for RDF. This proposal is complemented by studies of its transformation cost and its "size" compared to a directed labeled graph representation.

The thesis furthermore investigates some issues of RDF's graph nature in the light of the new model: RDF maps are studied as maps on graphs and an approach to decompose an RDF Graph into data and schema layers is presented. For the processing of RDF data the notions of *connectivity* and *paths* in RDF Graphs are essential; because RDF bipartite graphs incorporate statements and properties as nodes into the graph, it turns out that this model conveys a richer sense of connectivity than the standard directed labeled graph representations. Finally, we explore the perspectives of enhancing the expressivity of RDF query languages by a proposal of graph-based query primitives.

Keywords: RDF, Resource Description Framework, RDF Model, RDF Graph, RDF Databases, RDF Query Languages, Semantic Web, Metadata, Bipartite Graph, Hypergraph, Graph Theory

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Santiago de Chile, im August 2004

Jonathan Hayes

Contents

List of Figures

Acknowledgments

This thesis is one of the requirements for the diploma in Computer Science of Darmstadt University of Technology, Germany. I would like to express my thanks to several persons who have supported my work.

Professor Alejandro Buchmann is this thesis' supervisor at Darmstadt University. He provided me with the exceptional opportunity to undertake this work at the department of his colleague, Professor Ricardo Baeza-Yates at Universidad de Chile, Santiago, Chile. I am grateful to Professor Buchmann for this opportunity and for his confidence, and to Professor Baeza-Yates for cordially receiving me and providing me with all the necessary means for the research undertaken for this thesis.

My tutor at Universidad de Chile was Dr. Claudio Gutierrez. Dr. Gutierrez was involved into the progress of this work at all stages: Despite his obligations as lecturer and his involvement in several projects I always found an open door to discuss problems and to ask for advice. The tuition I enjoyed from him by far exceeded my expectations. (I sincerely hope that this will have left its mark in the quality of this work.)

I profited from Dr. Gutierrez' guidance not only with respect to my thesis. I fondly recall many interesting discussions on related subjects. Through his encouragement and substantial help we produced a paper with some of the results of this work which was—successfully—submitted to an international conference.

For all this I would like to express my special thanks to Claudio: for his commitment to the thesis topic and to my progress, for all the essentials I have learned concerning how to conduct research, and, quite generally, the friendly reception and the atmosphere that sustained our cooperation.

During my year's stay in Santiago, I was received as a member of the family and enjoyed the greatest and most generous hospitality and support by Mrs. Rita Schickhardt and her husband, Dr. Oskar Fuenzalida—relatives,

whose ancestors had immigrated to Chile in 1911. I am deeply indebted to them for their warm and sustained welcome.

I enjoyed financial support by the Klaus-Murmann-Foundation. I thank researchers, staff, and students at the Computer Science department of Universidad de Chile for kind integration and help in many situations. A remarkable experience were the activities with the mountaineering club of Universidad de Chile's engineering school; especially our final winter expedition will be a long-lasting memory.

1. Introduction

The Resource Description Framework (RDF) is a framework to annotate information resources in a machine-understandable way. This thesis contributes a graph model in order to enhance reasoning over the RDF model and the processing of RDF data.

A *graph* is a generalization of the simple concept of a collection of *nodes*, connected pair-wise by *edges*. It is very common to represent structures of any sort as graphs, because many practical questions can be reduced to graph problems [Wik04b]. For example, one of the first contributions to graph theory is Leonhard Euler's discussion of the *Seven Bridges of Königsberg*: Is it possible to walk a route through the town which returns to the starting point, but crosses each bridge only once? (cf. Figure 1.1)

From Euler's day to the present, a wealth of results has been produced by graph theorists, many of whom helped enhance real-life applications which were stated as graph problems. This thesis contributes such a formulation: *RDF*—to be introduced below—is a currently evolving technology; here we analyze it from a graph-theoretical point of view, we attempt to (re)phrase parts of it in "graph language", with the obtained concepts we further study some foundational issues of that technology, we present results of this graph-based investigation, and, finally, we point to open problems whose exploration appears promising.

RDF is an abbreviation for *Resource Description Framework* [MSB04]. The purpose of RDF is to describe information resources; to make RDF as general as possible, it is a framework for making *statements*, rather than a language in itself. The intended audience for such descriptions is *machine agents*, that is, programs running on computers connected by a network. This framework is a vital building block for a development towards an enhanced version of today's World-Wide Web, which we shall take advantage of in the near future.

The Web was built principally for human consumption, but due to its enormous size and its continuing growth it appears promising to make use of software agents for organizing, searching, and processing its content—already

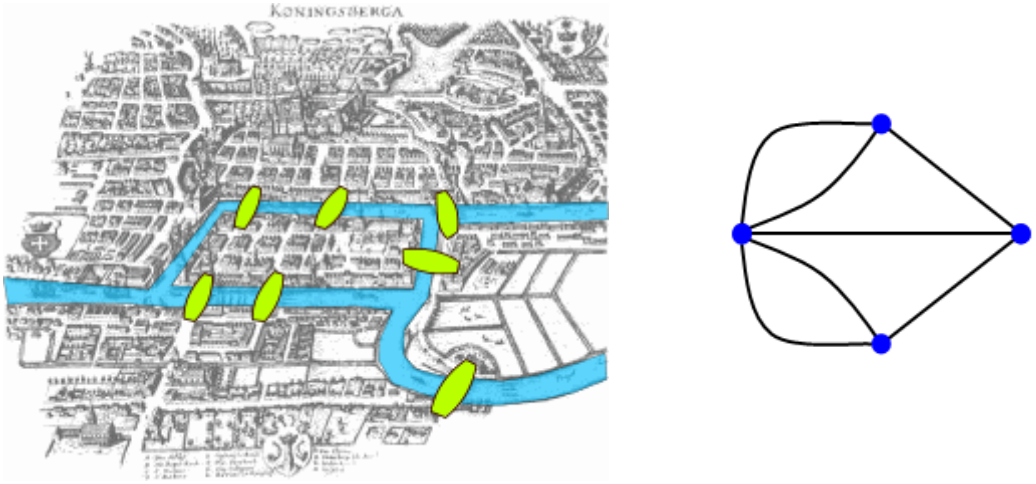


Fig. 1.1: The seven Königsberg bridges. Leonhard Euler studied the question whether it is possible to find a round-trip walk which crosses each bridge only once. He represented the problem by a graph, shown at right: points ("nodes") represent pieces of land (the island, and the upper, the lower and the right-hand city parts); lines ("edges") represent bridges. (Images from [Wik04a])

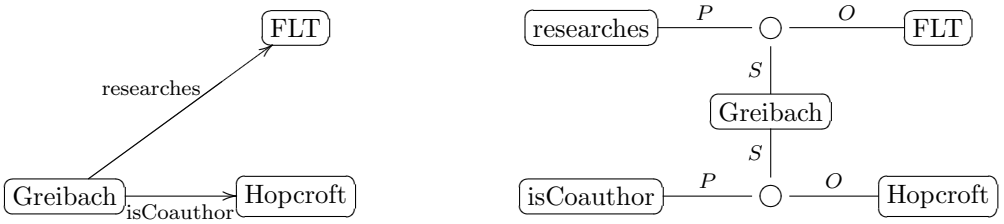
for the simple reason that computers have substantially facilitated access to information in the last decades. Although the data displayed on the Web is machine-*readable*, it is not machine-*understandable*, which is the fundamental requirement for meaningful processing of it.

A commonly accepted solution [BL98] towards such a *Semantic* Web is the enrichment of human-targeted Web resources (Web pages, etc.) with machine-intelligible information, also referred to as *metadata annotation*. The Resource Description Format provides a simple triple syntax to express such annotations: a *resource* (the subject) is described by a *property* (the predicate) and its *property value* (the object).

For example, <`Greibach isCoauthor Hopcroft`> specifies that the researcher Greibach (humans, too, are considered as resources) is a coauthor of Hopcroft—this is an RDF *statement*. Another assertion could be made about Greibach, for example, that she researches Formal Language Theory: <`Greibach researches FLT`>.

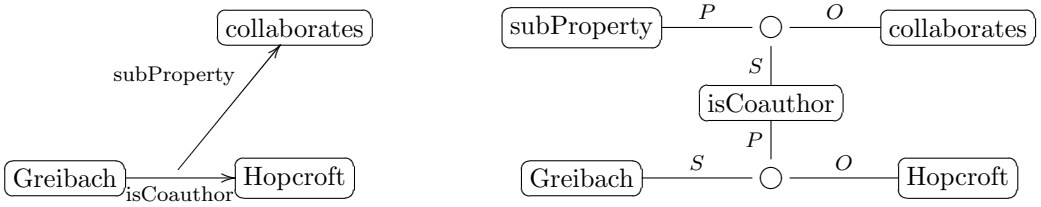
This is the so-called *abstract syntax* of RDF. Intuitively, we can visualize statements as graphs—we consider subjects and objects as nodes, and connect them by lines (edges) which are labeled by the statement predicate.

The left half of the diagram below is a graph representing the two statements introduced above:



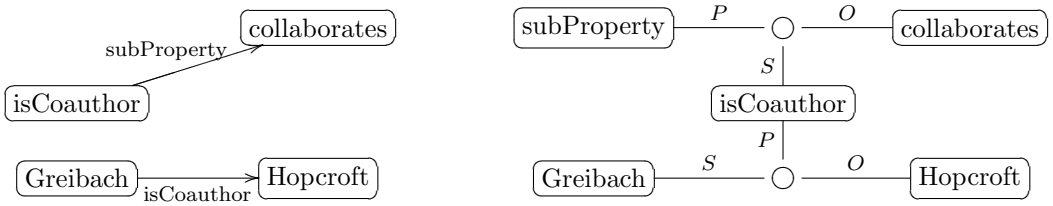
This example shows how *directed labeled graphs* can be employed to represent RDF. As indicated above, there are various purposes for this: By means of graphs RDF data can be conveniently visualized. Also, graphs convey the essence of RDF in that resources are interconnected by statements. The study of RDF can be enhanced by making use of well-established graph concepts, such as *path* or *degree*. Results for problems stated for graphs in general apply equally to RDF graphs, for example, the computation cost of whether an RDF graph contains a certain type of pattern. Finally, programming libraries providing graph data structures and algorithms are available to facilitate the implementation of applications using RDF.

However, such benefits can be taken advantage of only to the extent to which the graph representation really reflects the entity which is to be modeled with all its peculiarities and features. The graph representation used above—which is given in the RDF specification—has certain limitations. For example, RDF permits properties to be described just like other resources. For example, consider <`isCoauthor subProperty collaborates`>, which is a so-called *schema* statement: `isCoauthor` is a more specialized property than `collaborates`, that is, a sub-property. If we replace the second statement of the previous example with this one, we obtain the following diagram (consider the left half):



The new graph is somewhat strange: one of the edges connects an edge label with a node. The definition of graphs, however, implies that nodes and edges are distinct sets. While simple collections of RDF data may be visualized that way, other advantages of a graph model, such as the validity of theoretic results and the support by programming libraries are lost.

There is another way to visualize this example,



which avoids the non-standard edges of the previous example. Here, edges connect only nodes, but the *labels* of edges and nodes intersect. The disadvantage of this is that the obtained graph does not truly represent the *connectivity* of the RDF data, i.e., in this example, that the property `isCoauthor` (as in the statement describing Greibach) is related to `collaborates`.

Part I of this thesis discusses these issues in detail to give a motivation for the study undertaken (chapter 4). Chapter 2 presents the vision of a Semantic Web and specifies the role of the Resource Description Framework within it; technical preliminaries of both RDF and graph theory are introduced in chapter 3.

Part II contributes two approaches towards a formal representation of RDF by graphs. Chapter 5 formalizes a map to directed labeled graphs as outlined by the RDF specification [KC04]. As this representation does not solve limitations inherent in essentially binary edges of standard graphs, the first section of chapter 6 explores the use of hypergraphs with ternary edges to represent RDF. Hypergraphs can be represented by *incidence graphs*, and by this we derive *RDF bipartite graphs* as a proposal for a graph model for RDF.

RDF bipartite graphs have two node classes: *value nodes* represent all RDF resources (including properties) and literal values, unlabeled *statement nodes* represent statements. Three edges labeled S, P, and O indicate which value node has the role of the subject, the predicate, and the object for every statement. For example, each of the preceding figures presents a RDF bipartite graph representation in the right half.

The third section of that chapter formally presents a map from RDF to the class of RDF bipartite graphs and proves that it is indeed a 1:1 relation between sets of RDF statements and RDF bipartite graphs. Finally, results are given for the transformation cost from RDF to its RDF bipartite graph representation (which is in $O(n \lg n)$), and its "size" (the number of nodes and edges) in comparison to that of a directed labeled graph representing the same RDF data (it is up to 7 times as large).

The fact that RDF can be represented by standard (bipartite) graphs is considered as the principal result of this thesis. Part III uses this model for a more in-depth study of the graph nature of RDF.

Chapter 7 formalizes RDF Maps as defined in the RDF specification, and translates this concept to maps on graphs. That the question of *equivalence* of RDF Graphs can be reduced to the graph isomorphism problem has been noted before [Car01], but here a proof is provided for both directed labeled graph and RDF bipartite graph models of RDF. With *RDF Schema* statements vocabularies can be defined within a collection of RDF data itself. Chapter 8 presents an approach to *stratify* RDF data in order to analyze the schema part(s) and the "data" part of it.

Central for meaningful processing of RDF is the notion of *connectivity* which is discussed in chapter 9. The representation of RDF by graphs provides us with the notion of *paths*, which, applied to RDF, allows grasping the relation between resources. By introducing restrained types of paths it is shown that the directed labeled graph representation of RDF supports only a restricted concept of connectivity, while the RDF bipartite graph representation accounts for the full connectivity as conveyed by the RDF model.

Chapter 10 is meant to be a motivation towards further investigation of RDF storage and, especially, querying systems based on the graph features of the RDF model. After a short summary on the challenges of RDF to database systems and a presentation of two existing storage/query solutions (Jena and Sesame) a set of graph-based query primitives and use cases is given.

Contribution This thesis presents a graph-based intermediate model for RDF. In contrast to the default model of directed labeled graphs the proposed RDF bipartite graphs are unambiguously defined and incorporate explicitly statement properties and statements themselves as nodes into the graph. This approach truly reflects the connectivity as imposed by the RDF model; thus, it has the potential to enhance the processing of RDF data.

Notes on this document

This document was produced by `pdflatex` using the `thesis` style by Wenzel Matiaske. Thanks to the `hyperref` package PDF hyperlinks have been added to facilitate the following of cross-references in the document itself and to WWW resources. Care has been taken that all relevant link information is reflected in the printed version as well. This applies to colors, too: where colors have been used in figures it has been verified that the essential information is reflected by a gray-scale printing.

The bibliography contains so-called "back-references" after each entry: the section number of each quotation of the corresponding publication is given to facilitate literature lookup.

This document is my diploma thesis, submitted to the Department of Computer Science of Darmstadt University of Technology, Germany. I would be happy to receive any kind of questions or feedback.

Jonathan Hayes
`Jonathan.Hayes@gmx.de`
`http://purl.org/net/jhayes`

Part I

The Context

2. The Semantic Web: Vision and Reality

> "*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and humans to work in cooperation.*" (Tim Berners-Lee)

Chapter 4 presents the motivation for the research of a graph model for the Resource Description Framework (RDF) which was undertaken in this thesis. In this chapter, reasons shall be given why it is at all relevant to study this standard. It turns out that RDF is a fundamental building block for a future "semantic" World-Wide Web. The purpose of the following text is to convince the reader that this development is indeed about to happen, and that RDF will fulfill its commonly foreseen role in it.

The evolution of content presentation on the World-Wide Web (WWW) can be described by a three-generation model representing its past, present, and future [DvHB+00]. In the beginning of the WWW, textual information was displayed in the form of individually authored and often hand-written HTML pages. Today, after more than ten years of enormous growth, the amount of data retrievable as well as the significance for the daily life of individuals has vastly increased. However, the underlying technology (the internet protocol (IP), the hypertext protocol (HTTP), the hypertext markup language (HTML)) is essentially the same. What has changed is the way the content is produced: a large number of web pages is generated by programs—be it on-the-fly, to respond to a user's input, or off-line, when an entire website is generated from data specified in a more abstract schema. The reason for this is evident: the cost for input of data or its reformatting to suit other presentation needs required by humans is considerable. Compared to this, the careful definition of a schema to capture data semantics and the development of programs aware of those semantics is a sound investment. The success of the extensible markup language (XML) proves the importance of storing data in a structured, presentation-independent manner.

However, the vast majority of information displayed on the web is marked up only for visual consumption by human users. This present-day fact is not consistent with ongoing trends. The web will continue to grow, more people will participate in it, and more every-day procedures will be performed as web applications. How much more can the web grow? The technical infrastructure has proven to be quite scalable. So the access to enormous amount of content may be granted, but along with the growth in size, the topics displayed became more diverse, the services offered more complex, and the (human) languages used more numerous. Today's methods for information access rely on the use of directories and search engines. However, in spite of the awesome increase of search engine performance, key problems already identified years ago persist: the amount of pages indexed is only a fraction of the pages available (due to the dynamic nature of the web its growth can not be followed by a centralized index), search by keywords is imprecise (especially so in more and more diversified contexts), and ways of indexing non-textual content, such as multi-media resources and web services, is rudimentary. The trends identified here make evident that the weaknesses of classical search engines will persist.

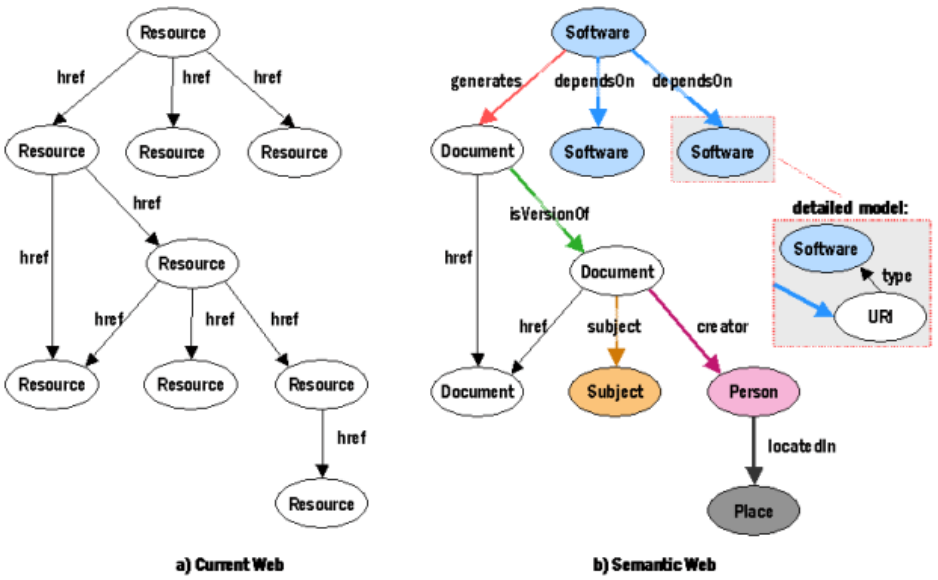


Fig. 2.1: A machine agent view of the semantics of resources and links. Essential for the vision of a Semantic Web is that resources and relations between resources are characterized in a machine-understandable way. (image from [KM02])

The *Semantic Web* is the vision of a *machine-understandable* (understandable as a contrast to the contemporary machine-*processable* content) web of information resources. This means that resources as well as relations between resources (such as hyperlinks) are characterized in a formal way (see Figure 2.1). This process is referred to as *metadata annotation*. As a method for organizing knowledge it has a long tradition, e.g., in the form of library catalogs or artifact descriptions in museums.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dc ="http://purl.org/dc/elements/1.1/">

   <rdf:Description rdf:about="http://www.w3schools.com">
      <dc:title>W3Schools Online Web Tutorials</dc:title>
      <dc:description>
         W3Schools is a collection of web building tutorials
      </dc:description>
      <dc:publisher>Refsnes Data</dc:publisher>
      <dc:date>1999-09-01</dc:date>
      <dc:type>World Wide Web Home Page</dc:type>
      <dc:format>text/html</dc:format>
      <dc:language>en</dc:language>
   </rdf:Description>
</rdf:RDF>
```

Fig. 2.2: An example of metadata annotation: The website `www.w3schools.com` is described using the Dublin Core vocabulary in RDF/XML.

Even though there is still much advance to be done at the level of technology, conceptually thinking, today's WWW is not too far from a Semantic Web: although the data displayed on the web now is marked up only for visual consumption, it is generated from data which the content provider stores in a structured format, such as in databases or in XML.

As the technical infrastructure evolves, resource providers can migrate from human-only presentation of content to forms accessible also by machine agents, thus enabling semantic-aware applications to be built on top. So far, the Internet contains only "semantic islands" of metadata enrichment which uses a shared vocabulary, for example the Friend-Of-A-Friend (FOAF)

network[1]. Programs such as the FOAF Explorer[2] or the Photo Metadata Co-Depiction Project[3] can be seen as first Semantic Web Applications already running today.

Other web applications will be led to an unforeseen potential as the "semantization" of the web continues. For example, annotating electronic documents with "standard" document metadata, such as the vocabulary[4] specified by the Dublin Core Metadata Initiative, is already possible today: methods exist for web pages [Pal02] and Adobe's Portable Document Format (PDF) [Ado03, Ado04]. Once a critical breakthrough is reached (for example, the usage of metadata for digitalized music is already well-established [HG04]) sophisticated services—such as Citeseer[5], a search engine for computer science publications—will become available for broad communities. More examples include intelligent agents for travel planning or online shopping—those existing today rely strongly on laboriously implemented extraction technics.

Various perspectives for new applications have been studied, for example in [BL98, All01]. Especially, "semantic searching" is addressed in [GMM03, Bra03b]

What are the *enabling technologies* for the Semantic Web? It is doubtful if there will ever be a formal Semantic Web specification—just as there is no single standard specifying the World-Wide Web. In the same way as the WWW is—on a technical level—the composition of a set of technology standards (HTTP, HTML, ...) which continue to evolve, it is probable that improving technologies for machine-understandable semantics will gradually enable Semantic Web applications. Considering the application perspectives sketched above, we may consider the Semantic Web as a vision; regarding the continuing work—which first results further motivate—we can state that it is an ongoing and already present process.

The evolution stages of the Web outlined above can be identified with content formats which proved characterizing for them. The Hypertext Markup Language (HTML) [W3C04] provides a standard for document visualization independent of the underlying machine architecture and software environment. The overwhelming success of the HTML format for data visualization motivated the creation of a presentation-independent standard for structured documents: the Extensible Markup Language (XML) [BPSM+04]. Key is-

[1] `http://www.foaf-project.org/`
[2] `http://xml.mfd-consult.dk/foaf/explorer/`
[3] `http://www.rdfweb.org/2002/01/photo/`
[4] `http://dublincore.org/documents/1998/09/dces/#`
[5] `http://citeseer.ist.psu.edu`

sues for the success of XML are its tree-like structuring of data [TBMM01] which suits an enormous breadth of applications, its solution in dealing with different character encodings [BM01], and the globally unambiguous referencing scheme it provides, Uniform Resource Identifiers [BLFM98, Con03]. These concepts provide an ease of reusability and portability of data unmatched by traditional database technology. Although the second stage of the WWW evolution—the machine-generated display of content stored in a presentation-independent way—relies physically on data stored in conventional databases as much as in XML formats, it is safe to say that the design principles of XML enabled much of this development.

The impetus triggered by the wide-spread adaptation of XML and its underlying motives leads the way towards the future. XML offers a rich and extensible expressivity for its documents, enables far-reaching syntactical validation, addresses exotic script encoding issues and provides a universal object referencing scheme—all of which are vital Semantic Web requirements already solved. However, the excitement over XML's potential resulted also in unrealistic assumptions, some implying that XML by itself already fulfills the requirements for the Semantic Web. It must be remembered that XML provides a standardization of document structure (as HTML standardized device-independent document presentation) and addresses in no way topics such as the "meaning" of the stored data [Gut02].

As argued above, a complete list of prerequisites for the Semantic Web can not be given. The fundamental role of metadata annotation, however, is evident from the description of the evolution of the WWW provided here; for a more reputed source see, e.g., the *Semantic Web Road Map* by Tim Berners-Lee [BL98].

As HTML and XML are accepted standards for document visualization and structure, there is need for a commonly accepted standard to express metadata. Of course, metadata is data, too, so it could be specified within the framework of XML. However, it turns out that there is a greatest common denominator for all kinds of metadata specifications: the assertion.

The *Resource Description Framework* (RDF)—comprehensively presented in the next chapter and discussed throughout this thesis—provides the infrastructure for the expression, the exchange and the extension of metadata. The core of it is the RDF *statement*, a subject-predicate-object triple which formulates the relation between an information resource, a property, and a property value [KC04].

The RDF model specifies the expression of assertions in an abstract graph syntax. It is neither bound to a concrete serialization syntax (e.g.,

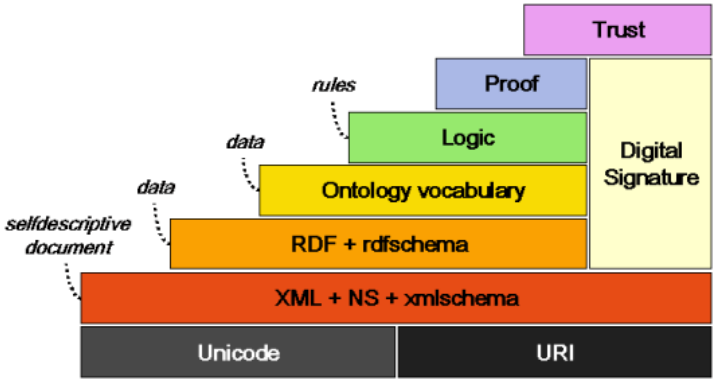


Fig. 2.3: Semantic Web Layers (from [KM02])

for storage in a file or exchange over a network) nor predefines more than the most elementary concepts and properties. Consider Figure 2.3 which proposes a layered architecture as a foundation for Semantic Web applications. RDF/XML [Bec04] is an XML-based serialization syntax for RDF, which itself relies on Unicode and URIs. The advantages of these technologies as presented above are thus provided at the assertion-making level of RDF even though the issues of character encoding and document validation are not addressed in the context of the RDF model. For an example, see Figure 2.2 in which RDF assertions are made about the resource `www.w3schools.com` in RDF/XML syntax.

Conversely, the greatest-generality approach of RDF—it is an universal framework for making assertions, and nothing more—enables other layers to be built on top, thus permitting any prospective application with whatever technology and data architecture by whatever community to be mapped on the model [Bra03a, BL98].

A machine-understandable mechanism for the definition of vocabularies is of crucial importance so that a Semantic Web attains the same degree of technical interoperability and growth-ensuring extensibility as the World-Wide Web showed in the past. Thus, RDF provides the "skeleton" for making assertions, but not the "flesh": the meaning of an RDF expression depends on the understanding of the concepts the RDF statements are made up of. RDF by itself provides only the most basic predefined concepts, the so-called RDF core vocabulary, and RDF Schema.

RDF Schema—which is equally mature as the core RDF—permits elementary structuring of a vocabulary, which enables a type-focused understanding of a vocabulary for machine agents and can support "semantic"

querying at a first stage. Its limited expressivity, however, falls short of what shall be provided by true ontology specification languages such as OWL [MvH04].

Conclusion This chapter presented a comprehensive overview of the evolution of the World-Wide Web towards its current state and identified obvious trends towards a future "semantic" web. The fundamental role of metadata annotation for this transformation and the importance of the Resource Description Framework as the main infrastructure for such assertions were made evident. As it is one of several layers which will very likely characterize Semantic Web applications, the function of RDF in this composition of enabling technologies was explained in order to disambiguate it from XML on the one hand, and to caution expectations that RDF alone provide "the semantics" on the other.

We hope that this overview gave a thorough motivation why it is at all relevant to study RDF. In that way this chapter may be understood as the "meta-motivation" for the proper motivation in chapter 4, where the tasks for this thesis are specified and justified.

3. Preliminaries

This section provides preliminaries for this thesis, which requires the familiarity with some basics of the Resource Description Framework, and largely common notions of graph theory. After a very comprehensive introduction into these fields examples which will be used throughout this document will be presented.

Readers aware of basic graph theory and who are interested primarily in the results of this thesis may prefer just to consider the first three paragraphs of the Resource Description Framework introduction which provides the essential background.

3.1 The Resource Description Framework[1]

The Resource Description Framework is an extensible infrastructure to express, exchange and re-use structured metadata [Mil98]:

"Everything is URI" Information resources are commonly identified by Uniform Resource Identifiers (URIs). By generalizing the concept of "resource", whatever is *identifiable* by an URI can be described in RDF. In this way, URIs can be assigned to anything, even physical objects, living beings, abstract concepts, etc. It is important to note that the *identifiability* does not imply *retrievability* of the resource.

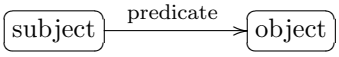
The principal advantages of this approach are that URIs are a globally unambiguous way to reference resources, and that no centralized authority is necessary to provide them. A common way to abbreviate it is the *XML qualified name* (or QName) syntax of the form `prefix:suffix`. For example, an URI such as `http://www.w3.org/TR/rdf-primer/` would be written as `w3:rdf-primer/` if it has been agreed that `w3` stands for `http://www.w3.org/TR/`.

RDF Statements The atomic structure for RDF specifications is the *statement*, which is a <`subject predicate object`>-triple. The information re-

[1] For a more thorough introduction see the "RDF Primer" [MM04] or other references given in subsection A.1.3

source being described is the subject of the statement and is denoted by an URI. The predicate of a statement is an URI reference representing a *property*, whose *property value* appears as the statement object. The property value can be a resource as well as a *literal* value. A literal is a string (e.g., a personal name) of a certain datatype and may only occur as the object of a statement.

RDF triples can be visualized as a directed labeled graph,



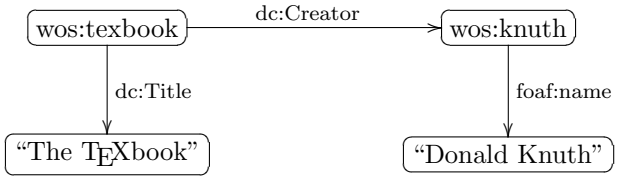
in which subjects and objects are represented as nodes, and predicates as arcs. This will be thoroughly discussed in chapter 4 and chapter 5, so we provide just this intuitive definition here.

In [KC04] a drawing convention is given—which will be neglected in this document. According to this convention nodes representing literals are drawn as rectangles and nodes representing URIs as ovals. In the drawings of this study, however, we need not to make these distinctions as we equally treat them as nodes. For an example of a graph drawn following that convention, see page 52.

RDF Graph *A set of RDF statements is an RDF Graph.* For example,

```
<wos:texbook dc:Creator wos:knuth>
<wos:texbook dc:Title "The TEXbook">
<wos:knuth foaf:name "Donald Knuth">
```

form an RDF Graph of three statements[2]. The TEXbook by Knuth is represented by the URI `wos:texbook` (`wos` is the namespace prefix of an—imaginary—"Web of Scientists" vocabulary, which shall be presented later in this chapter) and described in two statements. The object of the first statement, `wos:knuth`, is an URI representing the person Knuth. This RDF Graph would be visualized as follows:



[2] Note that this term—defined in the RDF specification [KC04]—is not a graph in the proper sense (see definition on page 44). This will be discussed in chapter 4.

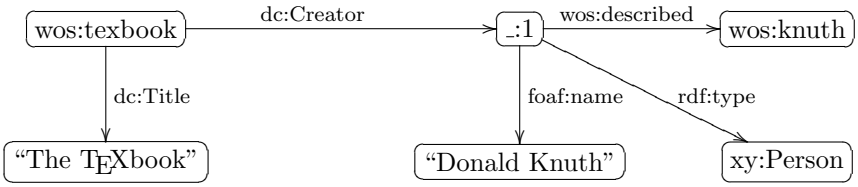
The meaning of this RDF Graph is "an information resource, identified by `wos:texbook`, has the title `"The TEXbook"` and was created by something which is identified by `wos:knuth` and whose name is `"Donald Knuth"`.

Anonymous Resources There are situations in which we wish to describe information using more complex structures of data than using a literal string or an URI pointer. For this, "anonymous" resources are used: the object of a statement can be an anonymous resource—or a *blank node*—which itself is the subject of other statements. Such a resource is represented by a *blank node identifier*, which is usually denoted as _:n, with n being an integer.

For example, a more sophisticated version of the above example about Knuth's authorship of the TEXbook would be

```
<wos:texbook dc:Creator _:1>
<wos:texbook dc:Title "The TEXbook">
<_:1 foaf:name "Donald Knuth">
<_:1 rdf:type xy:Person>
<_:1 wos:described wos:knuth>
```

which states more clearly that the author of the TEXbook is a human, which has a personal name and is further described in another resource (`wos:knuth`). The corresponding diagram is



It is important to note that the blank node identifiers carry no meaning; they are used merely for the purpose of serialization (e.g., file storage).

RDF Concepts We can now state more formally the triples which are syntactically correct: let "uris" be the set of URIs, "blanks" the set of blank node identifiers, and "lits" the set of possible literal values of whatever datatype (we consider all these sets as infinite). Then

$$(s, p, o) \quad \in \quad (\text{uris} \cup \text{blanks}) \times (\text{uris}) \times (\text{uris} \cup \text{blanks} \cup \text{lits})$$

is an RDF statement. Observe that there is no restriction to what URIs may appear as statement property.

We say that x is a *resource* if $x \in \text{uris} \cup \text{blanks}$, and everything occurring in an RDF statement is a *value* ($x \in \text{uris} \cup \text{blanks} \cup \text{lits}$). In this document, most of the time it will be referred to values because the type—URI, blank, literal—is not of interest.

To recall, a *RDF Graph* T is a set of RDF statements (T abbreviates triples). A *subgraph* of T is a subset of T. A *ground* RDF Graph is an RDF Graph without blank nodes.

With $\text{univ}(T)$ we denote the set of all values occurring in all triples of T and call it the *universe* of T; and $\text{vocab}(T)$, the *vocabulary* of T, is the set of all values of the universe that are not blank nodes. The *size* of T is the number of statements it contains and is denoted by $|T|$. With $\text{subj}(T)$ (respectively $\text{pred}(T)$, $\text{obj}(T)$) we designate all values which occur as subject (respectively predicate, object) of T.
Let V be a set of URIs and literal values. We define

$$\text{RDFG(V)} := \{\, T \;:\; T \text{ is RDF Graph and } \text{vocab}(T) \subseteq V \,\}$$

i.e. the set of all RDF Graphs with a vocabulary included in V.

Let M be a map from a set of blank nodes to some set of literals, blank nodes and URI references; then any RDF Graph T' obtained from the RDF Graph T by replacing some or all of the blank nodes N in T by $M(N)$ is an *instance* of T.

Consider an RDF Graph T_1, and a bijective map $M : B_1 \to B_2$ which replaces blank node identifiers of T_1 with other blank node identifiers. Then $T_2 = M(T_1)$ is an instance of T_1, and T_1 is an instance of T_2 (by the inverse of M which is trivially defined). Two such RDF Graphs are considered as *equivalent*. Equivalent RDF Graphs are treated as identical RDF Graphs, which is in conformance with the notion of blank nodes as "anonymous resources" whose identifier is assigned only in a temporary manner.

Reification It is also possible to make statements about other statements, which is called *reification*. A blank node symbolizes the statement to be described, while four other statements are used to provide the link between the blank node and the statement to be described.

Figure 3.1 shows the reification of a statement <a b c>. Reification is a good example to see that any property is also an information resource

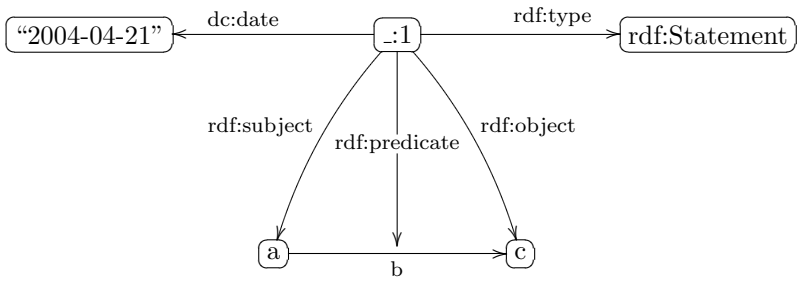
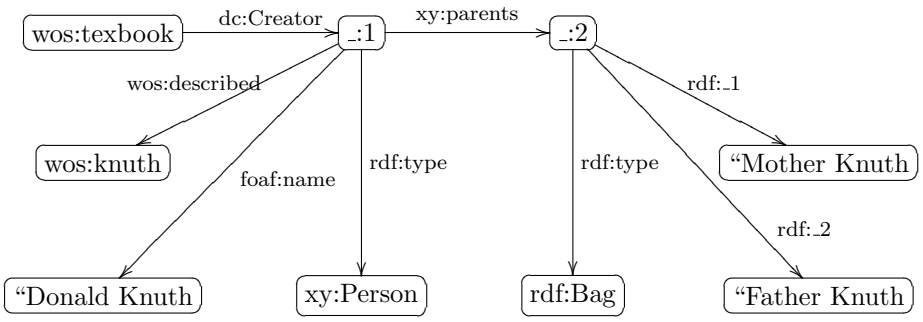


Fig. 3.1: Reification of a statement

which can be the subject or object of descriptions. In reifications, the predicate of the reified statement—`b` in this case—becomes the object of another statement.

Bag, List, Alternative RDF provides *Bags*, *Lists* and *Alternatives* as additional means for more complex descriptions. However, these structures are not of special interest in this study and are just mentioned for completeness. The following diagram shows the usage of a Bag construct:



Vocabularies As defined above, the vocabulary of an RDF Graph is the set of all URI resources and literals which occur in it. In a broader sense, we refer to a vocabulary as a set of concepts with a well-understood meaning to make assertions in a certain domain. Some RDF vocabularies are defined only in a human-understandable way, while others use RDF Schema (see below) or ontology specification languages to add machine-understandable semantics.

An example is the vocabulary of the *Dublin Core Metadata Initiative* [KS02] which is used to describe media. Some of the DC properties are:

Title The name given to a resource

Creator The person or organization responsible for the creation of the intellectual content of the resource

Subject The subject of the resource

Language Language(s) the resource is provided in

Date Date on which the resource was made publicly available in its current form

The examples presented above already make use of the Dublin Core vocabulary (e.g., `dc:title`). The prefix `dc` stands for the namespace identifier `http://dublincore.org/documents/1998/09/dces/#`.

Also elements of the *RDF core vocabulary* appeared in the examples, such as `rdf:type` and `rdf:subject`. This is a minimum set of concepts to provide the relevant RDF features of reification or bag constructs.

RDF Schema There is a distinguished vocabulary, *RDF Schema* [BG04], that may be used to define classes and properties of a vocabulary. The drawing on page 94 shows an RDF Graph together with a schema. For example, with the `rdf:type` property resources can be specified to be members of a *class*: <`picasso type Painter`> (cf. above example). Furthermore, classes (and properties) can be related to each other by the `rdfs:subClassOf` (`rdfs:subPropertyOf`) property: <`Painter subClassOf Artist`>

When querying RDF the schema specification can be taken advantage of, for example: "select all resources which are painters". *Schema-aware* query systems (e.g., RQL [FOR03]) are aware of the RDF Schema semantics, so subclass and subproperty relations are automatically incorporated in the queries. For example, "select all artists" would select all members of classes which are (possibly transitively) subclasses of `Artist`.

The semantics of the RDF and RDF Schema vocabulary are defined in the specification documents [BG04, KC04]. The *axiomatic triples* [Hay04] don't have normative character, but are nevertheless interesting to study to understand the meaning of the schema vocabulary.

A fragment of the axiomatic triples:

<`rdf:type rdfs:domain rdfs:Resource`>
<`rdfs:domain rdfs:domain rdf:Property`>
<`rdfs:range rdfs:domain rdf:Property`>

```
<rdfs:subPropertyOf rdfs:domain rdf:Property>
<rdfs:subClassOf rdfs:domain rdfs:Class>
<rdf:subject rdfs:domain rdf:Statement>
<rdf:predicate rdfs:domain rdf:Statement>
<rdf:object rdfs:domain rdf:Statement>
<rdfs:member rdfs:domain rdfs:Resource>
<rdf:first rdfs:domain rdf:List>
<rdf:rest rdfs:domain rdf:List>
...
```

RDF/XML So far, the *abstract syntax* was presented. There are a number of (concrete) serialization syntaxes; the most prominent among them is RDF/XML, which is an XML language to serialize RDF [Bec04]. RDF/XML has been criticized for not being easily readable; what is more, various XML documents can be generated for the same RDF Graph [CS04]. However, this study hardly considers RDF/XML for mainly two reasons, which are (1) said ambiguity of serialization, and (2) the fact that a tree data structure such as XML can not truly represent a graph data structure as RDF is. Finally, the task of this thesis is to establish an *intermediate* model for RDF, which resides between the abstract triple syntax and a concrete application-specific serialization. Figure 2.2 on page 31 shows an RDF description in RDF/XML.

Summary and Glossary This section presented the essential concepts of RDF which are required for this thesis.

Some notions shall be repeated to avoid confusion:

- An *RDF statement* is a subject-predicate-object triple with the semantics as described above. While, strictly, an RDF statement and an RDF triple are distinct concepts, they are treated synonymously throughout this document.

- An *RDF Graph* is defined as a set of triples and is, by itself, not a graph in the established sense (chapter 4). For this reason, "RDF Graph" will always be written with a capital 'G' to distinguish it from the mathematical concept of graph.

- *RDF data* is a somewhat more loosely used concept, but means essentially RDF Graph, or a set of RDF Graphs.

- *RDF specification* is ambiguously used; *the* RDF specification [documents] refers to the documents specifying RDF, while *an* RDF specification is a synonym for RDF Graph.

- An *RDF description* is (1) synonymous to RDF Graph or (2) synonymous to a single RDF statement. Later, the terms *description base* and *description schema* (from [KAC+02]) are introduced which are subsets (actually a partition) of an RDF Graph.

- The *RDF Model*, or *RDF abstract syntax*, refers to the abstract foundations of RDF as introduced in this section.

3.2 Graph Concepts

This section will recall some basic concepts of graph theory. The definitions follow established notations as presented, e.g., in [Die97] and [Kuc90].

Definition 1 (Graph): A *graph* is a pair $G = (N, E)$, where N is a set whose elements are called *nodes*, and E is a set of unordered pairs $\{u, v\}$, $u, v \in N$— the *edges* of the graph.

Two edges e_1, e_2 are said to be *incident* if they share a node ($e_1 \cap e_2 \neq \emptyset$). The *degree* of a node n, denoted degree(n), is the number of edges incident to it.

A graph $G' = (N', E')$ is a *subgraph* of G, denoted $G' \subseteq G$, if $N' \subseteq N$ and $E' \subseteq E$. □

Observe that the definition implies that the sets N and E are disjoint.

Definition 2 (Bipartite Graph): A graph $G = (N, E)$ is said to be *bipartite* if $N = U \cup V, U \cap V = \emptyset$ and for all $\{u, v\} \in E$ it holds that $u \in U$ and $v \in V$. A bipartite graph is *regular* if for every $v_1, v_2 \in V$ degree(v_1) = degree(v_2). □

Definition 3 (Multigraph): A graph G is a *multigraph* if multiple edges are permitted between two nodes (Figure 1.1 on page 22 shows an example). □

Definition 4 (Directed Graph): A *directed graph* is a graph where the edges have a direction, i.e., we distinguish between edges connecting the same nodes, but being oriented differently. We choose not to define the set E

as a set of pairs, as above, but rather as a set of edge elements with two maps

$$\text{from} : E \to N$$

$$\text{to} : E \to N$$

which yield the source and the target of each edge. Edges e_1, e_2 of a directed graph are incident if $\text{to}(e_1) = \text{from}(e_2)$ □

Definition 5 (Labeled Graph): A graph (N, E), together with a set of labels L_E and an edge labeling function $l_E : E \to L_E$ is an *edge-labeled* graph. A graph is said to be *node-labeled* when there is a node label set and a node labeling function, as above. We will write (N, E, l_N, l_E) for an edge- and node-labeled graph. □

The notions of path and connectivity will be important in what follows.

Definition 6 (Path): A *path* in a graph $G = (N, E)$ is a sequence of edges $e_1, \ldots, e_n$ where each edge e_i is incident to e_{i+1}, for $1 \leq i < n$.

A path is said to be *simple* if it additionally holds that $e_i \neq e_j$ for $i, j \leq n$, $i \neq j$; it is furthermore *cycle-free* if it holds for all nodes $n \in N$: $(n \in e_i, n \in e_j) \Rightarrow (i = j)$.

Two nodes x, y are *connected* if there exists a path $e_1, \ldots, e_n$ with $x \in e_1$ and $y \in e_n$. The *length* of a path is the number of edges it consists of.

The *label* of the path in an edge-labeled graph is the concatenation of the edge labels the path consists of: $l_e(e_1) \cdot l_e(e_2) \cdot \ldots \cdot l_e(e_n)$. □

3.3 Hypergraphs

Hypergraphs are systems of sets which are conceived as natural extensions of graphs. Set elements correspond to nodes in classical graphs, sets to edges—thus edges in a hypergraph can connect any number of nodes. The following definitions are inspired by [Ber87] and [Duc95].

Definition 7 (Hypergraph): Let $V = \{v_1, ..., v_n\}$ be a finite set, whose members are called *nodes*. A *hypergraph on* V is a pair $\mathcal{H} = (V, \mathcal{E})$, where $\mathcal{E}$ is a family $(E_i)_{i \in I}$ of subsets of V. The members of $\mathcal{E}$ are called *edges*. □

Definition 8 (Hypergraph Concepts):

- A hypergraph is *simple* if no edges are repeated (a hypergraph is generally a multiset of sets)

- The *rank* of a hypergraph $\mathcal{H}$ is defined as the maximum cardinality of an edge. Similarly, the *co-rank* of $\mathcal{H}$ denotes its minimum cardinality
- A hypergraph whose rank and co-rank are both r is called *r-uniform* ($|E_i| = r$ for all i)
- A simple, r-uniform hypergraph is called an *r-graph*
- An r-uniform hypergraph is said to be *ordered* if the nodes of every edge are numbered from 1 to r

□

Thus, a *simple ordered r-uniform hypergraph* can be conceived as a set of r-tuples.

Definition 9 (Graph Incidence Matrix): Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph with $m = |\mathcal{E}|$ edges and $n = |V|$ nodes. The edge-node *incidence matrix* of $\mathcal{H}$ is

$$M_{\mathcal{H}} \in \mathcal{M}_{m \times n}(\{0, 1\})$$

and defined as

$$m_{i,j} = \begin{cases} 1 & \text{if } v_j \in E_i \\ 0 & \text{else} \end{cases}$$

□

From a graph's incidence matrix the *bipartite incidence graph* can be derived:

Definition 10 (Bipartite Incidence Graph): For a hypergraph $\mathcal{H} = (V, \mathcal{E})$ with an incidence matrix $M_{\mathcal{H}}$ the bipartite incidence graph

$$B_{\mathcal{H}} = (N_V \cup N_{\mathcal{E}}, E)$$

is defined as follows:

$$N_{\mathcal{E}} = \{m_i : E_i \in \mathcal{E}\}$$
$$N_V = \{n_j : v_j \in V\}$$
$$E = \{\{m_i, n_j\} : m_i \in N_{\mathcal{E}}, n_j \in N_V, \text{ and } m_{i,j} = 1\}$$

□

The obtained graph $B_{\mathcal{H}}$ can be read to have an edge $\{e, v\}$ exactly when the hypergraph node represented by v is member of the hypergraph edge represented by e. It is evident that B is bipartite.

Example 1: Figure 6.2 on page 73 shows the incidence matrix of a hypergraph and the bipartite incidence graph derived from it. □

3.4 Running Examples

This section introduces some examples which will be used throughout the document.

Web Of Scientists *A Metadata Repository of Bibliographical Information*[3]. Such a knowledge base containing bibliographic information on scientific publications could be imagined as an RDF version of Citeseer[4] and DBLP[5] combined. Interesting queries would include paths of citations and other forms of cooperation.

Example RDF Graph 1: A small "Web of Scientists" example. The prefix `wos` represents the corresponding namespace prefix

1: <wos:Ullman> <wos:coauthor> <wos:Aho>
2: <wos:Greibach> <wos:coauthor> <wos:Hopcroft>
3: <wos:coauthor> <rdfs:subPropertyOf> <wos:collaborates>
4: <wos:Greibach> <wos:researches> <wos:topics/formalLanguages>
5: <wos:Valiant> <wos:researches> <wos:topics/formalLanguages>
6: <wos:Erdös> <wos:researches> <wos:topics/graphTheory>
7: <wos:Aho> <wos:collaborates> <wos:Kernighan>
8: <wos:Hopcroft> <wos:coauthor> <wos:Ullman>

The WordNet Ontology WordNet[6] is an online lexical reference system of the English language. Words are organized into synonym sets, which are ordered by the hyponym (subconcept) relation.

Museum Example *A Cultural Portal.* This classic example from [FOR03] is a good illustration of some of RDF's peculiarities and schema-aware querying.

[3] The similarity in name and concept to the *ISI Web of Science*® is not intended; here we make use of the term *Web of Scientists* just for examples.

[4] `http://citeseer.ist.psu.edu`

[5] `http://www.informatik.uni-trier.de/~ley/db/index.html`

[6] See `http://www.cogsci.princeton.edu/~wn/` for the WordNet project. The RDF representation considered here can be found at `http://www.semanticweb.org/library/`

US National Security Domain Use cases for querying for complex relationships between entities in large amounts of metadata to discover potential dangers has recently gained interest [AMHAS03].

Biology / Human Genome Project Excellent examples for the need for complex queries in large datasets come from biology. Examples include biopathways data, protein interaction networks, taxonomies and phylogenetic trees, chemical structure graphs, food webs, laboratory protocols, genetic maps, multiple sequence alignments. Queries against such graphs often include various types of path queries where regular expressions, shortest paths, and matching of subgraphs play a central role [Olk03].

Gene Ontology. The Gene Ontology Database[7] is a large ontology for the domain of genome concepts.

Photo Metadata Co-Depiction Experiment The website[8] of the project provides an interface to explore relations between people depicted on photographies. Two people are co-depicted if there exists some digital image that depicts them both.

[7] `http://www.godatabase.org/dev/database/`
[8] `http://www.rdfweb.org/2002/01/photo/`

4. Motivation

This chapter gives the motivation for the work undertaken in the scope of this diploma thesis.

The definition of RDF Graph and the representation scheme as directed labeled graph as provided in the RDF specification documents are discussed, followed by a critique of this approach. Then reasons are given why graph representations of RDF are at all relevant. We conclude that they are relevant and provide a list of requirements a graph representation should satisfy. Finally, a literature survey on other work related to the (graph) foundations of RDF is given.

The RDF specification is a suite of six documents now in the status of a WWW Consortium Recommendation: [MM04, KC04, Hay04, GB04, Bec04, BG04] which jointly replace the 1999 document by Lassila and Swick [LS99].

The most important document with respect to the understanding of RDF as a graph is *Resource Description Framework (RDF): Concepts and Abstract Syntax* [KC04], so reference to the RDF specification in this chapter will be understood as reference to this document unless otherwise specified.

4.1 The Graph Nature of RDF

The RDF specification does not clearly distinguish among the term "RDF Graph", the mathematical object of graph, and the graph-like visualization of RDF data.

RDF statements are triples consisting of a subject (the resource being described), a predicate (the property) and an object (the property value). The values of a statement are URI references; in addition, blank nodes may occur as subjects and objects and literals as objects. This simple model of assertions leads to a network of information resources, interrelated by properties which establish relations between resources and property values.

Thus, one can intuitively understand a collection of information resources and RDF statements describing them as a graph. To emphasize this charac-

teristic, the term *RDF Graph* is defined as a set of RDF triples; hence, any collection of RDF data is an RDF Graph.

As much as this is convincing for an intuitive understanding, the definition falls short of a definition of a graph in a mathematical sense (section 3.2 of this thesis). However, the very next paragraph after the definition of *RDF Graph* in [KC04] (in section 6.2 of [KC04]) "attaches" a graph definition

> *The set of nodes of an RDF graph is the set of subjects and objects of triples in the graph.*

which is somewhat fragmentary, as no definition for the edges is given. Section 3.1 of the same document—a section which is not marked as *normative*—contains a "visual explanation" of the edges of such a graph, and characterizes this construct—named "node and directed-arc diagram[1]"—as an *illustration* of an RDF Graph.

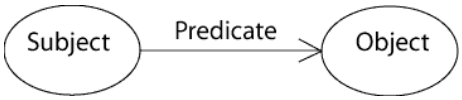


Fig. 4.1: Edges represent RDF statements (from [KC04])

This document's chapter 5 attempts to put these fragments together to offer a formal definition of the representation of an RDF Graph as a directed labeled graph. Still, this representation form leaves open some issues which are central to the RDF model which will be discussed in the next section.

To summarize: RDF has an *abstract graph syntax*, from which three concepts can be derived: An (1) *RDF Graph* is a technical term (explicitly defined as a set of statements) referring to the underlying graph data model of RDF but is not a graph by itself[2]. For the (2) *visualization of RDF* graphs or graph-like structures can be used; however, the usage of graphs in the context of RDF is not limited to drawing (cf. section 4.3). Directed labeled graphs are graphs but are not full-featured (3) *representations of RDF* and suffer from ambiguities, as will be explained in the following section.

[1] Also referenced to as"directed labeled graph", e.g., in [GCG99] and [LS99]

[2] To emphasize the difference between the mathematical concept of graph and the term *RDF Graph* as defined in the specification, the latter will always be written with a capital 'G' throughout this document

4.2 Shortcomings of Directed Labeled Graphs

This section shall address shortcomings of the representation of RDF as directed labeled graphs. This includes (1) the discussion of some features of RDF which are not explicitly treated in the RDF specification, and (2) limits which are inherent in directed labeled graphs.

RDF statements are subject-predicate-object triples, and the only constraint ruling the formation of RDF triples concerns the occurrence of blank nodes and literals. In particular, URI references may occur as any part of a triple.

The illustration scheme as directed labeled graph presented above does not address the issue that in a given set of RDF data an URI reference may occur at the same time as the predicate of one statement and as the subject or object of others. It shall be emphasized that this is not an exotic case, as, e.g., the RDF Schema declarations for some RDF vocabulary use properties as subjects. Also, every reification of a statement lets the statement's property appear as the object of another statement.

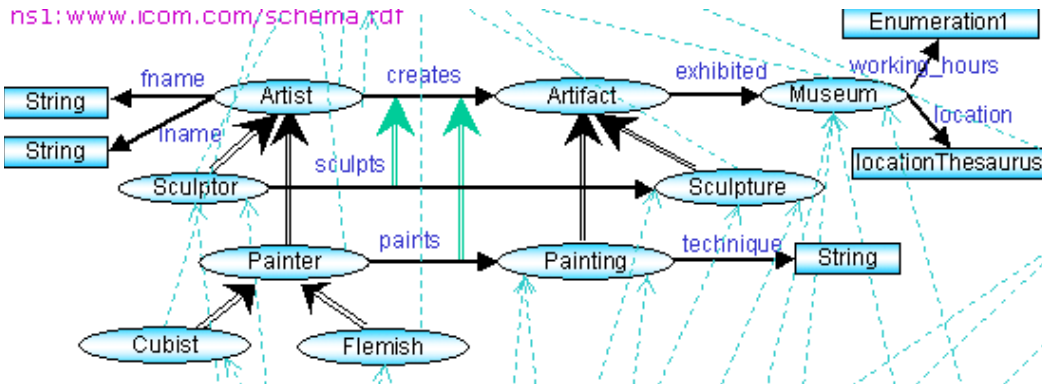


Fig. 4.2: Part of the Schema of the "Museum Example" of the *RQL Manual*. The turquoise (light gray) double-line arcs in this non-standard drawing indicate that `paints` and `sculpts` are sub-properties of `creates`.

So, consider an RDF Graph in which some information resource p occurs as a predicate of some statements and as the subject or object of others—how shall the edges representing the statements of the latter be drawn? The problem can be solved in two different ways:

1. The edges of the statements using p as subject (or object) are represented by arcs originating at (leading to) the edge label p. A drawing

using this solution is the museum example in figure 1 of the RQL manual [FOR03] (<`ns1:paints rdfs:subPropertyOf ns1:creates`>)—see Figure 4.2.

2. The information resource p occurs multiple times in the graph, once for each usage as a predicate (as edge label), and once for all uses as a subject or object (as node). An example for this solution is the figure explaining reification in section 4.3 of the RDF Primer [MM04]—see Figure 4.3 (which is—to the author's awareness—also the only drawing in the entire RDF specification suite where such a case occurs).

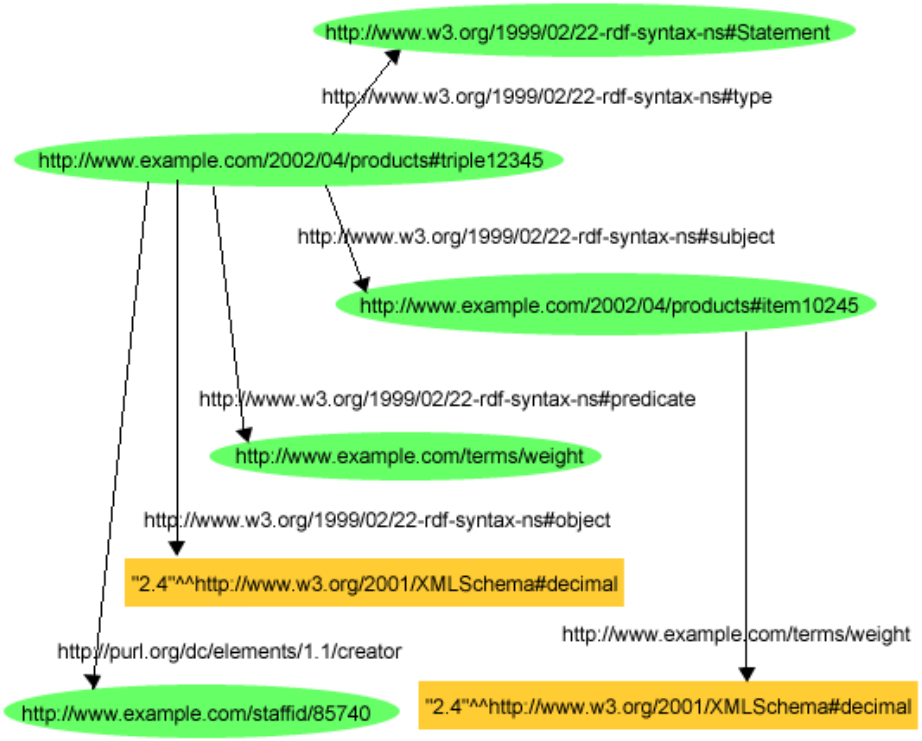


Fig. 4.3: Reification of a statement (from [MM04]). The predicate of the reified statement (`weight`) appears two times in the graph, once as edge label and once as node.

Both solutions have problems. Allowing arcs connecting nodes with other arcs can lead to puzzling drawings. What is more, the graph drawn this way has sets of arcs and nodes which intersect, which does not correspond to the commonly accepted definition of graphs. Defining a graph representation which is not a graph reduces this task to visualization for humans and gives

away advantages of a true graph representation, which will be presented in section 4.3.

On the other hand, duplicating the visual representation of an information resource for its role as predicate (representation as edge label) and as subject / object (representation as node) has a disadvantage. Information about a property (its sub- and super-properties, its domain and range) are disconnected from the actual usage of the property. This might result in users drawing misleading conclusions; allowing such multiple occurrences of resources jeopardizes one of the most important aspects of graph visualization, which is the implicit assumption that the complete information regarding an entity in a graph is obtained by examining its place in the drawing and its incident edges. Furthermore, chapter 9 introduces the notion of *connectivity*, which is central to querying RDF data. Duplicating properties in the graph representation of an RDF Graph makes it unsuitable for the study of connectivity.

However, it is not so much the ambiguity of the definition as the limitation of directed labeled graphs as such which causes problems in modeling RDF. RDF triples establish ternary relations which cannot be truly represented by the binary edges of classic graphs. Labeling the edges neglects the fact that properties are information resources in their own right, leading to the problem of either non-standard graphs or repeated occurrences of the same resource throughout the graph.

Thus, it can be said that the graph-like nature of RDF is intuitively appealing, but its naive formalization as directed labeled graph presents problems inherent in this type of graph. The recognition of the insufficiency of binary edges to model RDF statements led to an approach based on ternary edges (hypergraphs), which will be presented in section 6.1.

4.3 Reasons for Graph Representation of RDF

Graphs are mathematical objects which enjoy wide-spread usage for many tasks, which include the visualization and analysis of data for humans, mathematical reasoning, and the implementation as a data structure for developing software. These tasks are relevant in the context of RDF data as well, as this section shall present.

4.3.1 Fixing the Specification

The first specification of RDF in the status of a *WWW Consortium Recommendation* appeared in 1999 [LS99]. Since then, it has taken five years to revise the original specification and to replace it by a suite of six documents which gained recommendation status just recently, in February 2004 [MM04, KC04, Hay04, GB04, Bec04, BG04]. The success of RDF appears to take place at a rather modest pace, and one is tempted to conclude that the arduously advancing process of specification is one reason for this. The fact that the 2004 WWW Consortium Recommendation still contains ambiguities as described above gives motivation to supply a constructive critique and a proposition for refinement, with the hope to contribute to future revisions of the specification.

The issue—an incomplete definition of a graph representation, and a representation with certain limitations—might appear trivial. However, it has considerable impact: Numerous publications, including tutorials, exist which claim that RDF "is" a directed labeled graph. The immediate result is the—artificial—distinction between resources and properties which many people make. This prevents users from recognizing the actual simplicity of the RDF model. The results of the understanding of RDF bounded by the directed labeled graph model becomes especially evident in the limitations of current RDF query languages as studied in [AGH04].

4.3.2 Graphs as a Concept of Human Understanding

Graphs are a successful method to visualize and understand complex data. RDF, as a language developed to annotate and describe information resources and their relations among each other, allows the expression of potentially highly interconnected collections of metadata assertions.

For the visualization of RDF data directed labeled graphs may be employed successfully for not too complex RDF Graphs. Also, to explain the RDF model it is natural to use graphs. While the examples provided, e.g., in the *RDF Primer* [MM04] are simple and therefore above-mentioned limitations of directed labeled graphs are not as relevant, care should be taken that the (abstract) graph nature of RDF, the well-defined concept of *RDF Graph* and the representation of RDF as directed labeled graph are not confused.

4.3.3 Exploiting Graph Theory's Results

When reasoning formally over the RDF model, e.g., as described in *RDF Semantics* [Hay04], one has to operate with sets of triples. A set of triples (an RDF Graph) is a structure that due to multiple occurrences of the same resource leads to undesirable redundancies and does not capture the graph-like nature of RDF data, particularly regarding connectivity of resources.

Representing RDF by standard graphs could have several other advantages by reducing problems to well-studied topics from graph theory. A few examples at hand: The difference between RDF Graphs: when are two RDF Graphs the same? [BLC04, BL01b, Car01] Entailment: determining entailment between RDF Graphs can be reduced to graph maps: is graph A isomorphic to a subgraph of graph B? [Hay04]. Minimization: finding a minimal representation of an RDF Graph is important for compact storage and update in databases [GHM04].

4.3.4 Implementing Semantic Web Applications

From a programmer's point of view, graphs appear also as an appropriate data structure to model an RDF Graph. This has numerous advantages, as programming languages may feature (or facilitate) generic implementations of graphs and common graph algorithms. Furthermore, results from graph theory can be used, such as correctness proofs and complexity bounds for algorithms.

This motivation will be explored in more depth to connect the requirements of RDF applications with graph concepts and problems. Before setting out looking for graph representations of RDF Graphs, it is important to examine more closely the concrete usage of graphs by applications. It is doubtful—but nevertheless desirable—if one general graph model can be found to satisfy all kinds of tasks applications could require.

Below a list of possible "use cases" of RDF applications is given, which characterizes needed features.

Difference between RDF Graphs When are two RDF Graphs the same? How can it be verified? Tim Berners-Lee published a note on the so-called "diff-problem" ([BLC04]) and it has been studied in the context of graph isomorphism by Jeremy Carroll [Car01].

The "fine-granularity" of the diff problem is the question "what is the shortest sequence of edit operations which would transform RDF Graph 1 into RDF Graph 2?

Entailment Determining entailment between RDF Graphs can be reduced to graph isomorphism, too: "is graph A isomorphic to a subgraph of graph B?". (see [Hay04], section 7.1)

Minimization Finding a minimal representation of an RDF Graph is important for a compact storage and update in databases [GHM04].

Semantic Associations The relation between two documents can be deduced from the collection of paths through the graph of RDF statements. There is a series of papers concentrating on this, e.g., [AMHAS03]. We study this in the context of *connectivity* in subsection 9.3.4.

Clustering Well-established methods from graph clustering could be applied to an RDF Graph, yielding interesting patterns, such as *hubs* and *authorities* among information resources [Kle99]. Another application would be the optimization of the schema of an RDF database, as required in [WSKD03].

Related to this, a *structural decomposition* of an RDF Graph into, e.g., cycles or (spanning) trees might be useful to understand the structure of the RDF data.

Drawing RDF Graphs For the visualization of RDF a graph-based approach might be employed successfully.

This section presented some tasks of applications in which the graph representation of RDF could be taken advantage of.

4.3.5 Enhanced Querying

This section shall give reasons why a more thorough consideration of RDF's graph nature could enhance querying.

Annotating metadata can be found in two ways: (1) directly embedded, attached or linked to an information resource (e.g., referencing an RDF/XML file in the header of a HTML file via a `<link>` element [Pal02], or embedding it in a SVG image file (section 21 of [FJJ03])) it describes, and (2) as metadata collections in databases. Examples for the latter include large repositories such as WordNet and the Gene Ontology (presented in 3.4).

A considerable number of query languages have been developed to access the information stored in such metadata repositories. Several, among them

RQL [FOR03], underline the importance of graph-based querying. A critical investigation of this claim conducted in the process of this thesis resulted in a short paper [AGH04]—the findings will be summarized in section 10.3.

Motivating to study RDF's graph foundations in the light of querying is the desire of enhancing query languages towards an expressivity richer than offered today, especially considering the connectivity of resources.

This section gave reasons why the study of the graph foundations of RDF is promising with respect to applications and research leading to the Semantic Web. In particular, contributing towards a precise RDF specification, providing a model based on classic graphs for reasoning and software development, and improving the expressivity of RDF query languages have been identified as key motives for the investigation of the graph aspects of the RDF model.

4.4 Goals

The study of graph foundations of RDF should consider specifically

- A reflection upon the definitions provided in the RDF specification
- An improved definition of directed labeled graphs as a representation of RDF
- In the light of shortcomings of directed labeled graphs: are there alternatives which would be graphs in the traditional sense?

  If so, provide a formal graph-based intermediate model for RDF, which intends to be more concrete than the abstract RDF model to take advantage of results from graph theory, but still general enough to allow specific implementations (cf. Figure 4.4).
- In the case of another graph model, what are the transformation costs? Study its usefulness with respect to applications
- Explore the central notion of *connectivity* of information resources, in particular its impact on database storage and querying

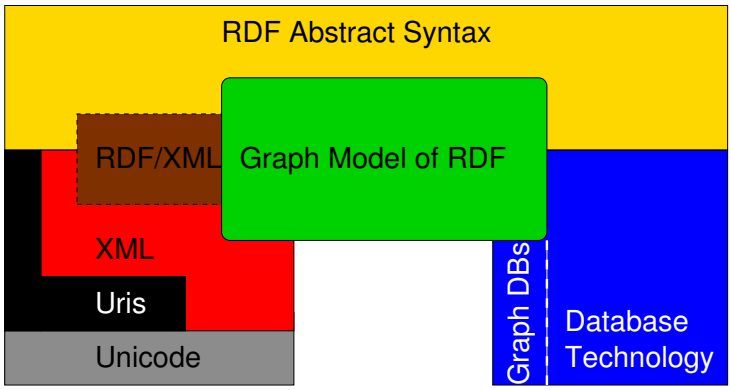


Fig. 4.4: A graph-based model for RDF as intermediate layer between the abstract triple syntax and a concrete serialization (compare with the figure on page 34).

4.5 Related Work

There is little work on foundations of the RDF model apart from the official documents of the WWW Consortium, specifically *RDF Concepts and Abstract Syntax* [KC04] and *RDF Semantics* [Hay04].

The semantics of RDF have received considerable attention, studies cover, e.g., the expressivity of RDF Schema [BKD+01, BKD+00, SEMD00], the relation to Topic Maps [LD01, Gar03, Ogi01], to UML [Cha98], or to Conceptual Graphs [CDH00, BL01a]. However, this has little to do with the more "syntactical" side of RDF Graph representations.

Studies of the RDF model in the context of storage and querying (e.g., [KCP00, KMA+04, BKvH02, WSKR03]) naturally touch the graph nature of RDF. However, despite frequent conjurations of the graph features of RDF, established notions such as the representation as directed labeled graph are not challenged. Furthermore, there exits a study oriented to querying [KAC+02], which contributes a formal typing to the RDF model, and a logical approach that gives identities to statements and so incorporates them to the universe [YK02].

However, the graph-theoretical aspects of RDF Graphs in particular have scarcely been studied. The author is aware of graph-related work of Jeremy Carroll [Car01, Car03], but graph-theoretic issues about RDF representations as discussed here are hardly present. Finally, there are results on normalization of RDF Graphs for compact database storage [GHM04], which directly influenced this work.

4.6 Conclusion

This chapter motivated the work undertaken in the course of this thesis. Concepts relevant to the RDF model have been clarified; especially, a distinction between RDF's abstract graph syntax, the defined term *RDF Graph*, the mathematical notion of graph, and graph representations of RDF data has been provided. A section on the shortcomings of directed labeled graphs challenges the concrete (or not so concrete) definition in the RDF specification documents and questions their usage as an all-purpose graph representation due to inherent limitations. The relevance of these issues became evident as the benefits of graphs as mathematical objects and standard structures were explored to serve the scientific foundations as well as concrete applications of RDF. Finally, sections on the objectives and related literature set the stage for the actual contribution now to follow.

Part II

RDF Graph Representations

5. Directed Labeled Graphs

This chapter proposes a map from RDF Graphs to directed labeled graphs. This proposal is thought to complement the "default" representation of RDF by graphs as provided by the RDF specification, which was reviewed in chapter 4. Even though this proposal claims to be formal and unambiguous, the fundamental limitations inherent in the modeling of RDF as directed labeled graph persist.

5.1 Mapping RDF to Directed Labeled Graphs

As outlined in the motivation chapter, the common usage of directed labeled graphs to represent RDF data falls into two categories:

1. One allows edges to be incident with both nodes and other edges
2. One distinguishes between occurrences of a resource as subject / object (represented by a node) and as property (represented by edges)

The second approach is followed here, because this results in a standard graph.

We formalize a representation of RDF based on *directed labeled graphs*, in the lines of what has been described in [KC04].

Definition 11 (Directed Labeled Graphs (DLG)): Let V be a vocabulary, T be an RDF Graph with $\text{vocab}(T) \subseteq V$ and $\mathcal{G}_{\text{dir, label, multi}}$ the set of directed, edge- and node-labeled multigraphs. We then define a map

$$\delta : \text{RDFG(V)} \rightarrow \mathcal{G}_{\text{dir, label, multi}}$$

as follows: $\delta(T) = (N, E, l_N, l_E)$, where

$$N = \{n_x : x \in \text{subj}(T) \cup \text{obj}(T)\} \text{ with}$$

$$l_N(n_x) = \begin{cases} (x, d_x) & \text{if } x \text{ is literal } (d_x \text{ is datatype identifier}) \\ x & \text{else} \end{cases}$$

$$E = \{\, e_{s,p,o} : (s,p,o) \in T \,\} \text{ with}$$
$$\text{from}(e_{s,p,o}) = n_s, \;\; \text{to}(e_{s,p,o}) = n_o, \quad \text{and} \quad l_E(e_{s,p,o}) = p$$

□

A directed labeled graph representing an RDF Graph is a multigraph, because arcs representing statements (s,p,o) and (s,p',o) differ only in their edge labels. Note that the same resource will be mapped to a node as well as to any number of edges if it occurs as subject / object and as property in the RDF Graph. This results in sets of edge and node labels which are not disjoint, leading to problems as mentioned before.

For convenience, we write $\delta_N(x)$ for the node which represents $x \in \text{subj}(T) \cup \text{obj}(T)$, and $\delta_E(y)$ for the set of edges representing the property $y \in \text{pred}(T)$. By $\delta(t)$, $t \in T$ we denote the (single) edge representing the statement t.

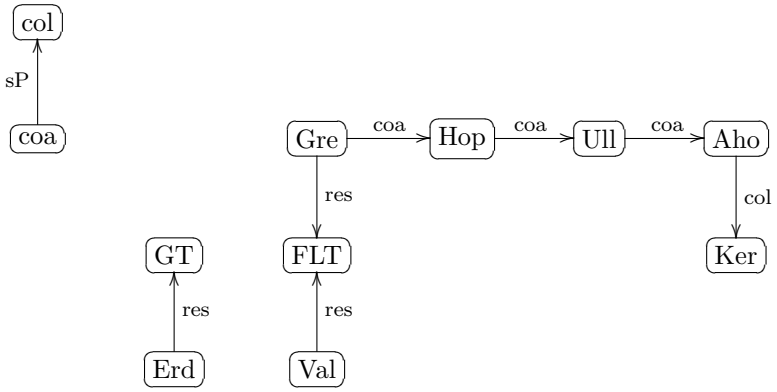


Fig. 5.1: The RDF Graph on page 47 represented by a directed labeled graph. URI prefixes have been omitted, and labels have been abbreviated as follows: **col**laborates, **s**ub**P**ropertyOf, **coa**uthor, **G**raph **T**heory, **res**earches, **Erd**ős, **Gre**ibach, **F**ormal **L**anguage **T**heory, **Val**iant, **Hop**croft, **Ull**man, **Aho**, **Ker**nighan.

Example 2: Figure 5.1 presents an example of such a graph. Consider the resource `coa` (`coauthors`) which appears as node as well as several time as edge. □

5.2 The Size of an RDF Graph

In this section the size $|T|$ (i.e., the number of statements it contains) of an RDF Graph T is expressed as a function of T's universe, and vice versa. This discussion for RDF Graphs is conducted in order to give size bounds for their directed labeled graph representation in the next section. The results of this section will be used in later chapters, too.

For the following studies, we make use of several example RDF Graphs with respect to a certain set of values $V_n = \{v_1, \ldots, v_n\}$, which will be the universe of these graphs.

Example 3 (Minimum RDF Graph): Let $T_{\min}(n)$ be an RDF Graph (with respect to a set of values V_n) as follows:

$$T = \{t_1, \ldots, t_k\} \quad \text{with } k = \left\lceil \frac{n}{3} \right\rceil$$

$$t_i = (v_{3i-2}, v_{3i-1}, v_{3i}) \quad \text{for } 1 \leq i < k$$

$$t_k = \begin{cases} (v_{n-2}, v_{n-1}, v_n) & n \equiv 0 \pmod 3 \\ (v_{n-1}, v_n, v_n) & n \equiv 2 \pmod 3 \\ (v_n, v_n, v_n) & n \equiv 1 \pmod 3 \end{cases}$$

(In this RDF Graph every value occurs exactly once; only in the last statement v_n might appear 2 or 3 times if n is not divisible by 3.)
We refer—informally—to $T_{\min}$ as a *minimum* RDF Graph because it has the smallest amount of statements with respect to its universe, the set of values V_n (See Figure 5.2, left). □

Example 4 (Maximum RDF Graph): Let $T_{\max}(n)$ be an RDF Graph (with respect to a set of values V):

$$T_{\max}(n) = \bigcup_{i,j,k \in \{1,\ldots,n\}} (v_i, v_j, v_k)$$

We consider $T_{\max}(n)$ as a *maximum* RDF Graph because it contains the largest amount of statements with respect to its universe (see Figure 5.2, right). □

Proposition 1 (RDF Graph Size Bounds): Let T be an RDF Graph. Then it holds

$$\left\lceil \frac{|\operatorname{univ}(T)|}{3} \right\rceil \leq |T| \leq |\operatorname{univ}(T)|^3.$$

□

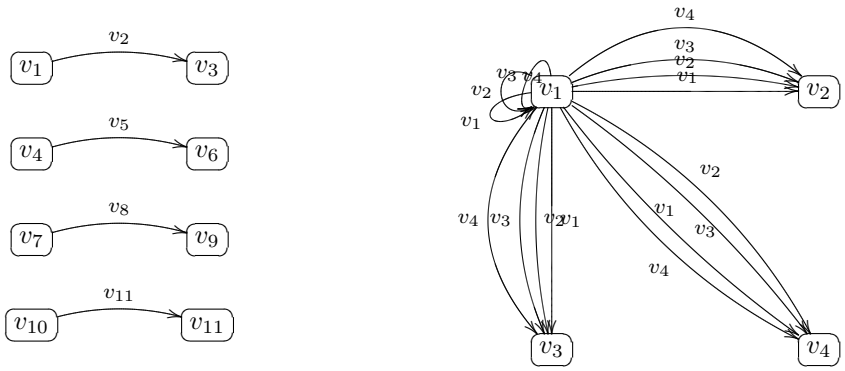


Fig. 5.2: Directed labeled graph representations of RDF Graphs: to the left, $T_{min}(11)$, to the right a fragment of $T_{max}(4)$. For reasons of visibility only the statements $<v_1\ v_x\ v_y>$ were drawn in the latter.

Proof 1: (1) $\left\lceil \frac{|\operatorname{univ}(T)|}{3} \right\rceil \leq |T|$: Consider $T_{\min}(n)$ as presented in example 3. The RDF Graph has a minimum amount of statements because every value of the universe occurs only once (except for, possibly, in the last statement). Its number of statements is $\left\lceil \frac{|\operatorname{univ}(T)|}{3} \right\rceil$. (2) $|T| \leq |\operatorname{univ}(T)|^3$: The amount of statements in $T_{\max}(n)$ of example 4 is maximal, because its statements are a complete enumeration of triples on the universe. □

Corollary 1:

$$\sqrt[3]{|T|} \leq \operatorname{univ}(T) \leq 3\,|T|$$

□

We now introduce two more example RDF Graphs in order to give bounds for its subjects (and predicates, objects). Again, $V = \{v_1, \ldots, v_n\}$ is the set of values of its universe.

Example 5 (One-Subject RDF Graph): Let $T_{\text{one-sub}}(n)$ be an RDF Graph on V whose statements have all the same subject, that is, $\operatorname{subj}(T_{\text{one-sub}}(n)) = \{v_1\}$ (see Figure 5.3, left). □

Example 6 (Full-Subject RDF Graph): Let $T_{\text{full-sub}}(n)$ be an RDF Graph on V for which it holds $\operatorname{subj}(T_{\text{full-sub}}(n)) = \operatorname{univ}(T_{\text{full-sub}}(n))$ (see Figure 5.3, right). □

RDF Graphs for a minimal and maximal amount of predicates and objects are similarly defined. These examples do not put into account restrictions for the occurrence of blank nodes and literals as subjects and predicates.

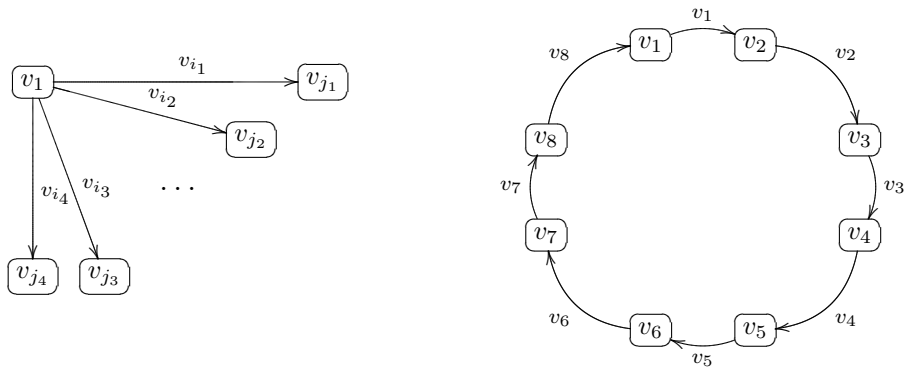


Fig. 5.3: Examples for $T_{\text{one-sub}}(n)$ and $T_{\text{full-sub}}(8)$ (there are others).

However, the motivation for giving the bounds is to compare the directed labeled graph representation of RDF with other graph representations, thus, in our context, these constraints are not relevant.

Proposition 2 (Subject Bounds): Let T be an RDF Graph. Then it holds

$$\begin{aligned} 1 \leq |\operatorname{subj}(T)| &\leq |T| \\ |\operatorname{subj}(T)| &\leq |\operatorname{univ}(T)| \end{aligned}$$

□

Proof 2: The lower bound is obvious because of $T_{\text{one-sub}}(n)$ of example 5. The upper bound $|T|$ is trivial, and $|\operatorname{univ}(T)|$ follows from example 6. (Recall that, by proposition 1, neither $|T|$ nor $|\operatorname{univ}(T)|$ are larger in the general case.) □

The same bounds are valid for predicates and objects (neglecting the restrictions of the RDF model as explained above).

Example 7 (Maximum-Node RDF Graph): Let T be an RDF Graph on a set of values $V = \{s_1, \ldots, s_n, o_1, \ldots, o_n\}$. Then

$$T = \bigcup_{1 \leq i \leq n} (s_i, p, o_i) \text{ for any } p \in V$$

(As will be seen later, this is an RDF Graph which has the maximum number of nodes in a directed labeled graph representation.) □

Proposition 3 (Subject/Object Bounds): Let T be an RDF Graph. Then it holds

$$\begin{aligned} 1 \leq |\operatorname{subj}(T) \cup \operatorname{obj}(T)| &\leq 2\,|T| \\ |\operatorname{subj}(T) \cup \operatorname{obj}(T)| &\leq |\operatorname{univ}(T)| \end{aligned}$$

□

Proof 3: We can define an RDF Graph similar to $T_{\text{one-sub}}(n)$ with the same single subject and object, which implies the lower bound. The RDF Graph of example 7 has the highest $|\operatorname{subj}(T) \cup \operatorname{obj}(T)|$ with respect to an RDF Graph T, because its subjects and objects are disjoint and meet each the upper bound given by proposition 2. $|\operatorname{univ}(T)|$ remains the upper bound with respect to an RDF Graph's universe because each $\operatorname{subj}(T)$ and $\operatorname{obj}(T)$ are $\subseteq \operatorname{univ}(T)$. □

This section presented bounds for RDF Graphs which go beyond the simple notion of "size" as the number of statements. The results will be useful to compare graph representations of RDF, as conducted in the following section for directed labeled graphs (and later for other representations).

5.3 The Size of a Directed Labeled Graph

We now give bounds for the number of nodes and edges for directed labeled graphs representing RDF Graphs.

Proposition 4: Let T be an RDF Graph, and $\delta(T) = (N, E, l_N, l_E)$ the directed labeled graph representing it. Then:

1. $|E| = |T|$
2. $|N| \leq 2\,|T|$

□

Proof 4: $\delta(T)$ contains exactly one edge per statement in T, so (1) is obvious. $N = \operatorname{subj}(T) \cup \operatorname{obj}(T)$, so proposition 3 implies (2). □

Conclusion This chapter provided an explicit definition for the representation of RDF as a directed labeled graph.

Resources occurring as subject / object as well as as predicate are represented by a node as well as by (possibly several) edges. This approach avoids that non-standard edges connecting nodes with other edges are necessary. While this leads to a "standard" directed labeled graph, it is unavoidable that edge and node labels intersect. This is not desirable because it breaks with the implicit assumption for graph representations that all information "related" to an entity is found in the direct vicinity (incident nodes / edges) of the graph element(s) modeling that entity.

However, we argue that any graph representation of RDF modeling statements by edges has shortcomings, because essentially binary edges can not truly represent the ternary relation a statement triple establishes. This limitation which is inherent in the approach of directed labeled graphs is the core motivation to look for alternate graph representations for RDF (chapter 6).

6. RDF Bipartite Graphs

This chapter shall study an alternate graph representation for RDF. Naturally, hypergraphs permitting edges which connect three nodes (in fact, any number of nodes can be connected by hypergraph edges) appear as a promising model for RDF. The first section discusses this approach. However, hypergraphs are not what could be considered as "common" graphs, so many advantages of representing RDF by graphs as discussed in chapter 4 do not apply, or apply only to a limited extent.

Because of this, the second section explores a class of bipartite graphs which are derived from hypergraphs; section 3 formally presents a map from RDF Graphs into this class. It is shown that *RDF Bipartite Graphs* provide an adequate model for RDF. Finally, the transformation cost to obtain this representation is studied.

6.1 Hypergraphs

One of the major problems encountered in trying to model RDF Graphs as classical graphs is the fact that an edge or labeled edge cannot represent the ternary relation given by an RDF triple. Therefore it is natural to turn the attention to graphs with 3-node-connecting edges instead of classical 2-node edges, that is, hypergraphs.

Definition 12: The *representing hypergraph* $\gamma(T) = (V, \mathcal{E})$ of an RDF Graph T is defined as follows:

$$V = \{v_x : x \in \text{univ}(T)\}$$

$$\mathcal{E} = \{(v_x, v_y, v_z) : (x, y, z) \in T\}$$

□

We understand the hypergraph edges to be ordered. It is also 3-*uniform*, because each edge represents an RDF triple. Furthermore, it can be seen that the obtained hypergraph is *simple* (no multi-edges), because T is a set in which each triple occurs only once.

Example 8: Figure 6.1 shows a hypergraph representing the first three statements of the example on page 47. □

$\mathcal{E}$ = { { coauthor, subPropertyOf, collaborates }, { Ullman, coauthor, Aho },{ Greibach, coauthor, Hopcroft } }

V= { collaborates, coauthor, subPropertyOf, Aho, Greibach, Hopcroft, Ullman }

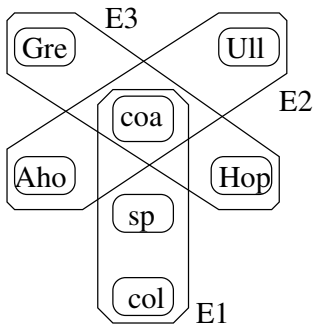


Fig. 6.1: Example of a simple 3-uniform hypergraph.

Proposition 5: Any RDF Graph T can be represented by a simple ordered 3-uniform hypergraph $\gamma(T)$. (Follows trivially from the definition) □

The converse of the proposition also holds when imposing constraints on the occurrences of blank nodes and literals: blank nodes may not be predicates and literals may not serve as subjects or predicates.

Hypergraphs are closed under set operations. The fact that an RDF Graph, too, is a set of triples allows us to apply set operations to an RDF-representing hypergraph in the same way as to the RDF Graph itself.

6.2 Deriving Incidence Graphs from Hypergraphs

The representation based on hypergraphs introduced in the previous section has the disadvantage that hypergraphs are not as common as conventional graphs. We now present a map that transforms hypergraphs representing RDF as studied above to a class of common graphs.

As stated in the preliminaries chapter, hypergraphs can be represented by incidence matrices. Such a matrix can be understood as the node adjacency

matrix of a bipartite graph. Thus, the idea is to represent RDF by means of bipartite graphs.

Hypergraph incidence matrices represent membership of a node in an edge with a '1' in the corresponding entry (see Figure 6.2). In the case of the hypergraph representing an RDF Graph, the nodes of an edge are ordered. This ordering must be preserved in the incidence matrix: instead of using an integer according to the occurrence of a node in an edge, we choose to label them by S, P or O to represent the role (subject, predicate, or object) of the information resource in that statement-edge.

Hence, when deriving the bipartite incidence graph of this incidence matrix, an edge will be added for every S, P, O entry of the matrix, and this edge will be labeled with the corresponding character. Thus, the only difference between the graph derived from the incidence matrix of any hypergraph and an RDF Graph hypergraph is the fact that each edge has one out of three labels.

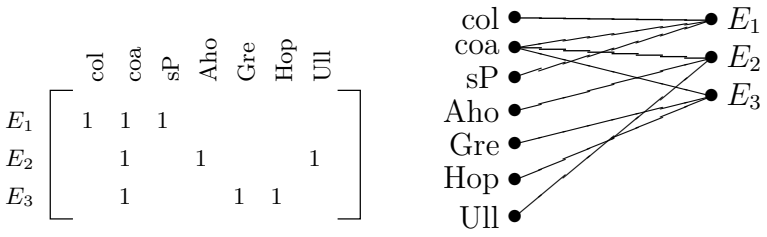


Fig. 6.2: Incidence matrix representing the hypergraph of Example 6.1 and the corresponding incidence graph. In the case of an ordered hypergraph, matrix entries will indicate the position of the occurrence of the node in the edge.

This section presented the underlying idea for representing RDF by bipartite graphs. In the following, the map will be formally introduced.

6.3 Mapping RDF Graphs to Bipartite Graphs

Before formally defining the map from RDF Graphs to these incidence graphs, we study the graph class which is the range of this transformation.

Definition 13 (Bipartite Labeled Graphs): Let $\mathcal{B}$ be the set of bipartite labeled graphs $G = (V \cup St, E, \mathrm{nl}, \mathrm{el})$, $V \cap St = \emptyset$, where each edge in E connects a node in V with a node in St, and $\mathrm{el} : E \to \mathrm{EL}$ and $\mathrm{nl} : V \to \mathrm{NL}$ are labeling functions. The elements of V are called *value nodes* and those of St

are *statement nodes*. To allow multiple edges between nodes we understand E as a set of edge elements and introduce functions stat : $E \to St$ and val : $E \to V$ which yield the statement node stat(e) and the value node val(e) which are connected by an edge e [1]. □

The above defined set of graphs $\mathcal{B}$ represent the bipartite incidence graphs derivable from hypergraphs—although the naming of the node classes already reflects the intended usage. To specify the graphs to be obtained from hypergraphs representing RDF Graphs—that is, the incidence graphs of (a certain subset of) simple ordered 3-uniform hypergraphs—the following restrictions are made:

Definition 14 (RDF Bipartite Graphs (RBG)): We define the graph class RBG $\subset \mathcal{B}$ as follows. Let $B = (V \cup St, E, \mathrm{nl}, \mathrm{el}) \in$ RBG. The set of edge labels is EL $= \{S, P, O\}$ and the set of node labels NL is partitionable into the three sets uris(B), blanks(B), and lits(B). Then it holds

(1) for all $st \in St$:

$$\mathrm{degree}(st) = 3$$

and (2): for all $e \in E$ with val(e) $= v$:

$$\mathrm{nl}(v) \in \begin{cases} \mathrm{uris}(B) \cup \mathrm{blanks}(B) & \text{if } \mathrm{el}(e) = S \\ \mathrm{uris}(B) & \text{if } \mathrm{el}(e) = P \\ \mathrm{uris}(B) \cup \mathrm{blanks}(B) \cup \mathrm{lits}(B) & \text{if } \mathrm{el}(e) = O \end{cases}$$

and

$$\forall e' \in E,\ \mathrm{stat}(e') = \mathrm{stat}(e) : \quad \mathrm{el}(e') \neq \mathrm{el}(e).$$

□

Now the map of RDF Graphs into the class RBG of bipartite graphs is defined.

Definition 15 (RDFG to RBG Map): Let T be an RDF Graph on a vocabulary V. Then we define a map β : RDFG(V) $\to$ RBG as follows: $\beta(T) = (V \cup St, E, \mathrm{nl}, \mathrm{el}) \in$ RBG is the *RDF bipartite graph representing* T, with

$$V = \{v_x : x \in \mathrm{univ}(T)\},$$

$$St = \{st_t : t \in T\}.$$

[1] Example 12 (1) on page 98 shows a case where multiple edges are required.

The labeling of the nodes is given by:

$$\mathrm{nl}(v_x) := \begin{cases} (x, d_x) & \text{if } x \text{ is literal } (d_x \text{ is the datatype identifier of } x) \\ x & \text{else} \end{cases}.$$

The set of edges E is defined as follows: for each triple $t = (x, y, z) \in T$ there are edges

$$e_1 : \ \mathrm{val}(e_1) = v_x, \ \ \mathrm{stat}(e_1) = st_t, \ \ \mathrm{el}(e_1) = S,$$
$$e_2 : \ \mathrm{val}(e_2) = v_y, \ \ \mathrm{stat}(e_2) = st_t, \ \ \mathrm{el}(e_2) = P,$$
$$e_3 : \ \mathrm{val}(e_3) = v_z, \ \ \mathrm{stat}(e_3) = st_t, \ \ \mathrm{el}(e_3) = O.$$

□

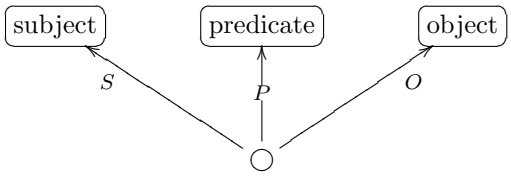


Fig. 6.3: A statement as RDF bipartite graph. The circle node is the *statement node* representing the statement <`subject predicate object`>, the nodes above are the *value nodes* representing the three components of a statement. Edge labels S, P, and O indicate the role of the value nodes in a statement.

Note that $\beta(T)$ is a *3-regular* bipartite graph, because the degree of each node in St is 3. For convenience, for the statement node st representing an RDF statement t, we write $\beta(t)$. Similarly, the value node v_x representing a value x is denoted by $\beta(x)$ (recall that *value* is the abstract term for URIs, blanks, and literals).

Example 9: Figure 6.3 illustrates how a single statement is represented as RDF bipartite graph. Figure 6.4 shows the RDF bipartite graph representation of the Web Of Scientists example. The drawing convention is as follows: unlabeled circles represent statement nodes, boxes with rounded corners value nodes. Edge labels S, P, and O indicate the subject, predicate, and object of a statement. □

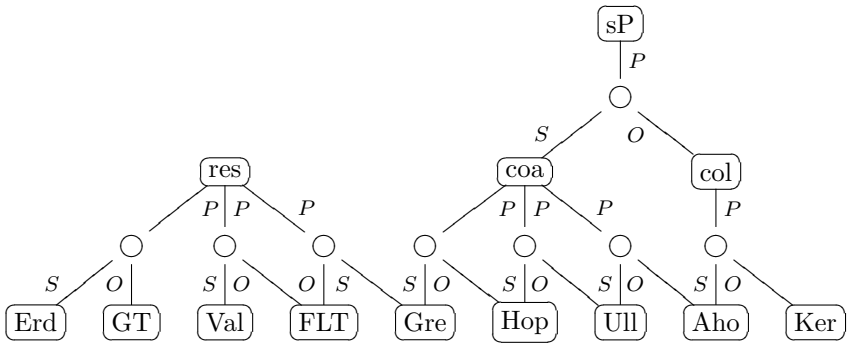


Fig. 6.4: The RDF bipartite graph of the RDF Graph on page 47.

The map from RDF Graphs to RDF bipartite graphs is well-defined, and one can go back and forth between T and $\beta(T)$, as the following proposition states:

Proposition 6: (1) For each RDF Graph T there is a uniquely defined RDF bipartite graph $\beta(T)$ representing it. Moreover, (2) there exists a function

$$\beta^{-1} : \mathrm{RBG} \to \mathrm{RDFG(V)}$$

satisfying

$$\beta^{-1}(\beta(T)) = T.$$

□

Proof 6: (1) is implied by the definition of β. (2a) *Definition of* β^{-1}: Let B be an RDF bipartite graph $(V \cup St, E, \mathrm{nl}, \mathrm{el})$. Then

$$\begin{aligned}\beta^{-1}(B) = \{ \quad (a,b,c) \quad : \quad & e_S, e_P, e_O \in E, \\ & \mathrm{stat}(e_S) = \mathrm{stat}(e_P) = \mathrm{stat}(e_O), \\ & \mathrm{el}(e_S) = S, \quad \mathrm{nl}(\mathrm{val}(e_S)) = a, \\ & \mathrm{el}(e_P) = P, \quad \mathrm{nl}(\mathrm{val}(e_P)) = b, \\ & \mathrm{el}(e_O) = O, \quad \mathrm{nl}(\mathrm{val}(e_O)) = c \quad \}\end{aligned}$$

(2b) Let $T \in \mathrm{RDFG(V)}$ and $\beta(T) = (V \cup St, E, \mathrm{nl}, \mathrm{el})$. By definition 15 is

$St = \{\ st_t : t \in T,\ \ t = (a_t, b_t, c_t)\ \}$, and

$$E = \bigcup_{t \in T} \{\ e_{tS},\ \ e_{tP},\ \ e_{tO} : \\ \mathrm{stat}(e_{tS}) = \mathrm{stat}(e_{tP}) = \mathrm{stat}(e_{tO}) = st_t, \\ \mathrm{el}(e_{tS}) = S,\ \mathrm{nl}(\mathrm{val}(e_{tS})) = a_t, \\ \mathrm{el}(e_{tP}) = P,\ \mathrm{nl}(\mathrm{val}(e_{tP})) = b_t, \\ \mathrm{el}(e_{tO}) = O,\ \mathrm{nl}(\mathrm{val}(e_{tO})) = c_t \quad \}.$$

It follows immediately that $\beta^{-1}(\beta(T)) = T$. □

This section introduced a map from sets of RDF statements to bipartite graphs. So-called *statement nodes* incorporate explicitly the statements as nodes into the graph. As *value nodes*, there is no distinction between resources being described, and properties. The role of values (their appearance as a subject, predicate, or object) is entirely characterized by the labels of the edges incident to it.

In the next section, the cost of this transformation shall be studied.

6.4 Transformation Cost

In this section we study the number of nodes and edges of an RDF bipartite graph $\beta(T)$ as a function of the size of an RDF Graph T, as well as the cost of the actual transformation from an RDF Graph to the corresponding RDF bipartite graph.

Proposition 7 (RDF Bipartite Graph Size): Let T be an RDF Graph and $\beta(T) = (V \cup St, E, \mathrm{nl}, \mathrm{el})$ the corresponding RDF bipartite graph. Then

1. $|St| = |T|$
2. $|V| = |\,\mathrm{univ}(T)|$
3. $|E| = 3\,|T|$

□

Proof 7: (1) and (2) are stated in the RDF bipartite graph definition. Every statement node has exactly three edges; this implies (3). □

Now we can compare the size of a representation as a directed labeled graph (which was provided in chapter 5) and as RDF bipartite graph:

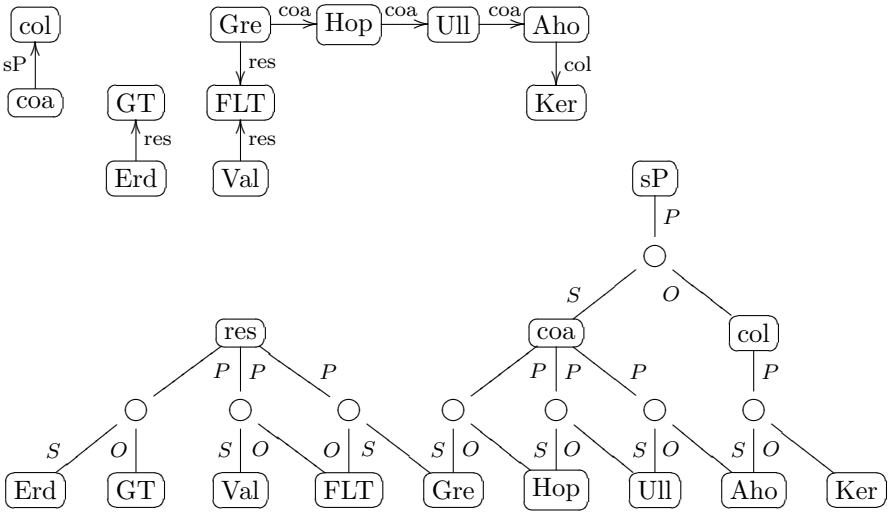


Fig. 6.5: A comparison: the directed labeled graph and the RDF bipartite graph of the Web of Scientists example on page 47.

Proposition 8: For any RDF Graph T, representations as RDF bipartite graph $\beta(T) = (N_{\mathrm{RBG}}, E_{\mathrm{RBG}}, \mathrm{nl}, \mathrm{el})$ (with $N_{\mathrm{RBG}} = V \cup St$) and as directed labeled graph $\delta(T) = (N_{\mathrm{DLG}}, E_{\mathrm{DLG}}, l_N, l_E)$ compare as following:

$$|N_{\mathrm{RBG}}| + |E_{\mathrm{RBG}}| \quad \leq \quad 7\ (|N_{\mathrm{DLG}}| + |E_{\mathrm{DLG}}|)$$

□

Proof 8: The following transformation was obtained by proposition 7, corollary 1 (page 66), and proposition 3 (page 68):

$$\begin{aligned}
|N_{\mathrm{RBG}}| + |E_{\mathrm{RBG}}| &= \\
|V| + |St| + |E_{\mathrm{RBG}}| &= \\
|\operatorname{univ}(T)| + |T| + 3\ |T| &\leq \\
7\ |T| &\leq 7\ (1 + |T|) \\
&\leq 7\ (|\operatorname{subj}(T) \cup \operatorname{obj}(T)| + |T|) \\
&= 7\ (|N_{\mathrm{DLG}}| + |E_{\mathrm{DLG}}|)
\end{aligned}$$

□

Example 10: Figure 6.5 shows a directed labeled graph and an RDF bipartite graph representation of the Web of Scientists example. Directed labeled graph edges and RDF bipartite graph statement nodes—both representing RDF statements—number 8. There are 11 directed labeled graph nodes and 13 RDF bipartite graph value nodes: the discrepancy arises from the fact that `res` (researches) and `sP` (subPropertyOf) never appear as subjects or objects and are thus not incorporated as nodes into the directed labeled graph representation. □

Algorithm 1: BETA: $T \rightarrow \beta(T)$

for each statement $t = (s, p, o) \in$ T **do**
  $St \leftarrow St \cup \{st_t\}$
  $V \leftarrow V \cup \{v_s, v_p, v_o\}$
  $\mathrm{nl}(v_s) \leftarrow s$; $\mathrm{nl}(v_p) \leftarrow p$
  if (o is literal) **then** $\mathrm{nl}(v_o) \leftarrow (o, d_o)$ $\{d_o$ is datatype identifier of o$\}$
  else $\mathrm{nl}(v_o) \leftarrow o$
  $E \leftarrow E \cup \{st_t, v_s\}$; $\mathrm{el}(\{st_t, v_s\}) \leftarrow$ 'S'
  $E \leftarrow E \cup \{st_t, v_p\}$; $\mathrm{el}(\{st_t, v_p\}) \leftarrow$ 'P'
  $E \leftarrow E \cup \{st_t, v_o\}$; $\mathrm{el}(\{st_t, v_o\}) \leftarrow$ 'O'
end for

Proposition 9: Algorithm BETA takes an RDF Graph T as input and computes the RDF bipartite graph $\beta(T)$. (Trivial) □

Proposition 10: $\beta(T)$ can be computed in $O(|T| \lg |T|)$. □

Proof 10: The most expensive operation performed in the loop of algorithm 1 is a set-add for the value nodes. If V is implemented as a sorted set of the value labels, this operation can be performed in log n time, $n \leq |\mathrm{univ(T)}| \leq 3|T|$. □

Conclusion. A map from RDF Graphs to common (bipartite) graphs was introduced, which was motivated by the fact that hypergraphs can be represented by incidence graphs. A study of the transformation cost revealed that RDF bipartite graphs can be obtained efficiently. Evidence for the adequacy of RDF bipartite graphs as representation for RDF is also given in the context of RDF maps in chapter 7.

Part III

Applications of RDF Bipartite Graphs

This part on "Applications of RDF Bipartite Graphs" concentrates on some foundational issues of the graph nature of RDF.

RDF maps transform blank nodes to URIs and literals or replace blank node identifiers by other blank node identifiers. Such mappings are fundamental for advanced topics such as computing the *core* of an RDF Graph (see, e.g., [GHM04]), or the question of RDF Graph *equivalence*. Chapter 7 provides a definition for RDF maps on standard RDF triples as well as on RDF bipartite graphs.

RDF Graphs consist, in the terms of [KMA+04], of a *description schema* and a *description base*. However, can there be several *schema layers*, and how can such layers be identified without relying on the `rdfs` namespace prefix? This is addressed in chapter 8.

A central issue for applications "understanding" an RDF model is the *connectivity* of its resources. This is intuitively convincing as the statements a resource occurs in—which are the adjacent nodes and edges in the graph representation—form the part of the RDF Graph which directly describes that resource, and the relation between two resources is entirely represented by the collection of paths which exist between them. Chapter 9 studies connectivity and paths.

Likewise, for querying RDF models the expressivity of the query language is of interest. We find that aforementioned notions of connectivity and paths give an interesting perspective to formulate querying primitives. Such primitives are presented in chapter 10 and are complemented by arguments why an enhancement of current RDF query languages would be useful.

7. Maps

This chapter discusses maps of RDF Graphs as defined in *RDF Semantics* [Hay04] and how to apply them to RDF bipartite graphs. It is connected to chapter 6 in that it gives further evidence that RDF bipartite graphs are an adequate model for RDF. Because of its amount of material, however, it was chosen to dedicate a separate chapter to RDF maps.

7.1 RDF Graph Maps

We now define *RDF maps* after what is outlined in [Hay04].

Definition 16 (RDF Map): An RDF map between RDF Graphs T_1 and T_2 is a map

$$\varphi : T_1 \to T_2$$

satisfying

$$\varphi(T_1) \subseteq T_2$$

$$(\text{ where } \varphi(T) \text{ is an abbreviation for } \{\ \varphi(t) \mid t \in T\ \})$$

and

$$\varphi((s,p,o)) = (\varphi'(s), \varphi'(p), \varphi'(o))$$

with

$$\varphi' : \text{univ}(T_1) \to \text{univ}(T_2), \quad \varphi'(x) = \begin{cases} \varphi''(x) & \text{if } x \in \text{blanks}(T_1) \\ x & \text{else} \end{cases}$$

with

$$\varphi'' : \text{blanks}(T_1) \to \text{univ}(T_2)$$

□

As blank nodes of T_1 are mapped into the universe of T_2, it must be prevented that a blank node, which appears as a subject, is mapped to a literal. However, this is already implied by $\varphi(T_1) \subseteq T_2$ and the fact that T_2 is a (valid) RDF Graph.

For RDF Graph equivalence, only a restrained notion of RDF map is required, which we name a *blank node bijection*.

Definition 17 (Blank Node Bijection): We constrain the notion of an RDF map to obtain a *Blank Node Bijection*. An RDF map φ is a blank node bijection, if

1. φ'' is injective
2. φ'' is surjective on the blank nodes of T_2: $\text{range}(\varphi'') = \text{blanks}(T_2)$

The inverse φ^{-1} of a blank node bijection φ is trivially defined. □

Definition 18 (RDF Graph Equivalence): Let φ be a blank node bijection and T_1, T_2 RDF Graphs. We say that T_1 and T_2 are *equivalent* [Hay04] and write $T_1 \cong_{\text{RDF}} T_2$ iff[1]

$$\varphi(T_1) = T_2.$$

□

The notion of RDF Graph equivalence is of high relevance, because it describes a class of RDF Graphs whose meaning is equivalent. Because blank node identifiers are generated in an arbitrary way, RDF Graphs produced by different applications or by the same application under different circumstances might syntactically differ although they actually mean the same.

7.2 Equivalence of RDF Graphs and Graph Isomorphism

We now study how equivalence between RDF Graphs can be reduced to the classic graph isomorphism problem: Given two graphs A and B, does there exist a bijective map from the nodes of A to the nodes of B and from the edges of A to the edges of B, such that A is a subgraph of B?

In the case of labeled graphs, one additionally requires that the map observes the labeling. In our context, directed (node- and edge-) labeled graphs, we want to show that two directed labeled graphs are isomorph if and only if the corresponding RDF Graphs are equivalent (i.e. differ only by renaming blank nodes, see [Hay04]). For this, we adapt the classic definition of graph isomorphism to ensure that blank nodes may be mapped to any other blank node, and all other nodes (and edges) get mapped in a way consistent with their label.

[1] *iff* is an abbreviation for *if and only if*

Definition 19 (Directed Labeled Graph Isomorphism): Let T_1 and T_2 be RDF Graphs, and let $\delta(T_1) = (N_1, E_1, l_{N_1}, l_{E_1})$ and $\delta(T_2) = (N_2, E_2, l_{N_2}, l_{E_2})$ be the corresponding directed labeled graph representations. We say that $\delta(T_1)$ and $\delta(T_2)$ are *isomorph* and write $\delta(T_1) \cong \delta(T_2)$ if there exist bijective maps $\phi_N : N_1 \to N_2$ and $\phi_E : E_1 \to E_2$ satisfying

$$\forall\, \delta(x) \in N_1 \begin{cases} l_{N_1}(\delta(x)) = l_{N_2}(\phi_N(\delta(x))) & \text{if } x \in \text{uris}(T_1) \cup \text{lits}(T_1) \\ & \text{if } x \in \text{blanks}(T_1) \end{cases}$$

and

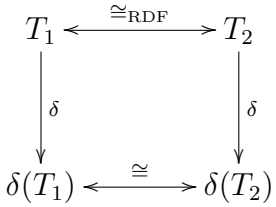
$$\begin{aligned} \forall e \in E_1 : \phi_E(e) = e' \in E_2, \text{ with } \text{from}(e') &= \phi_N(\text{from}(e)), \\ \text{to}(e') &= \phi_N(\text{to}(e)), \\ l_{E_2}(e') &= l_{E_1}(e)\,; \end{aligned}$$

(This definition follows the established definition of graph isomorphism).

□

Proposition 11: Let T_1 and T_2 be RDF Graphs. T_1 and T_2 are equivalent iff $\delta(T_1) \cong \delta(T_2)$. □

The correspondence between RDF Graph equivalence and graph isomorphism is not new—e.g., Jeremy Carroll studies the applicability of graph isomorphism algorithms to compute whether two RDF Graphs are equivalent [Car01][2]. However, to the author's knowledge, neither a formal model of graphs representing RDF, nor the reduction of RDF Graph equivalence to graph isomorphism has yet been stated in a formal way (cf. the following diagram).



Proof 11: RDF Graphs T_1 and T_2 are equivalent, if there exists a blank node bijection φ such that $\varphi(T_1) = T_2$; directed labeled graphs are isomorph

[2] "Determining if two graphs are isomorphic is thought to be neither an NP-complete problem nor a P-problem, although this has not been proved. In fact, there is a famous complexity class called *graph isomorphism complete* which is thought to be entirely disjoint from both NP-complete and from P." [Wei]

if there exist maps ϕ_N, ϕ_E as described in definition 19. We now define ϕ_N, ϕ_E by means of φ:

$$\begin{aligned} \text{for all } x \in \text{univ}(T_1) \quad &: \quad \phi_N(\delta(x)) = \delta(\varphi'(x)) \\ \text{for all } t \in T_1 \quad &: \quad \phi_E(\delta(t)) = \delta(\varphi(t)) \end{aligned}$$

It can be easily verified that ϕ_N, ϕ_E observes the conditions of definition 19: blanks are only mapped to blanks because φ is a blank node bijection, and by definition 16 the edge condition is accounted for. □

This section formally stated that the problem of RDF Graph equivalence can be reduced to the question of graph isomorphism. For this, the directed labeled graph representation was used. In the following, we study RDF maps on RDF bipartite graphs, and show that isomorphism between RDF bipartite graphs, too, represents RDF Graph equivalence (proposition 12 and corollary 2).

7.3 RDF Bipartite Graph Maps

In this section we introduce RDF bipartite graph maps in order to reflect RDF maps in the RDF bipartite graph model. The underlying motivation is to give further evidence that RDF bipartite graphs are a useful representation for RDF.

Definition 20 (RDF Bipartite Graph Map): Let $B_1 = (V_1 \cup St_1, E_1, \text{nl}_1, \text{el}_1)$ and $B_2 = (V_2 \cup St_2, E_2, \text{nl}_2, \text{el}_2)$ be RDF bipartite graphs. An RDF bipartite graph map $\psi : B_1 \to B_2$ is a triple of maps

$$\psi_V : V_1 \to V_2, \qquad \psi_{St} : St_1 \to St_2, \qquad \psi_E : E_1 \to E_2$$

which observe the following conditions:

1. The value node map has only effect for blank nodes:

$$\forall\, v \in V_1 : \text{nl}_1(v) \in \text{uris}(B_1) \cup \text{lits}(B_1) \quad \Rightarrow \quad \text{nl}_1(v) = \text{nl}_2(\psi_V(v))$$

2. It must be taken care that, by the map, literals never appear as subjects:

$$\begin{aligned} &\forall\, v \in \text{blanks}(B_1) : \exists\, e \in E_1 \quad \text{with} \\ &\quad \text{val}(e) = v \text{ and } \text{el}_1(e) = S \quad \Rightarrow \quad \psi_V(v) \notin \text{lits}(B_2) \end{aligned}$$

3. Edges are mapped to the corresponding nodes and keep their labels:

$$\begin{aligned} \forall\, e \in E_1 \quad & : \\ \mathrm{val}(\psi_E(e)) &= \psi_V(\mathrm{val}(e)) \\ \mathrm{stat}(\psi_E(e)) &= \psi_{St}(\mathrm{stat}(e)) \\ \mathrm{el}_2(\psi_E(e)) &= \mathrm{el}_1(e) \end{aligned}$$

4. $\psi(B_1) \subseteq B_2$

□

This definition provides the measures to perform RDF maps on RDF bipartite graphs, a correspondence which will be formally stated in proposition 12. Condition 1 takes into account that RDF maps as defined in the RDF specification only map blank nodes. Conditions 2 and 3 are necessary to ensure that the result of the map will again be an RDF bipartite graph. There is no corresponding constraint for RDF maps because the condition $\varphi(T_1) \subseteq T_2$ already implies that $\varphi(T_1)$ is a valid RDF Graph—in fact, any set of RDF triples is an RDF Graph. The $\subseteq$ operator in condition 4 of the RDF bipartite graph map definition denotes the standard subgraph relation.

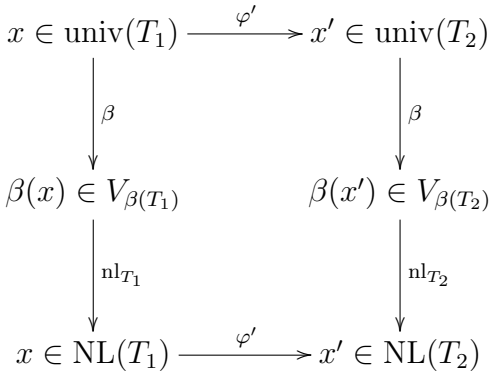
Definition 21 (RDF Bipartite Graph Isomorphism): Consider RDF bipartite graphs $B_1 = (V_1 \cup St_1, E_1, \mathrm{nl}_1, \mathrm{el}_1)$ and $B_2 = (V_2 \cup St_2, E_2, \mathrm{nl}_2, \mathrm{el}_2)$. We say that B_1 and B_2 are *isomorph* and write $B_1 \cong B_2$ if there exists an RDF bipartite graph map ψ from B_1 into B_2 whose ψ_V, ψ_{St} and ψ_E are bijective. □

7.4 RDF Maps on RDF Bipartite Graphs

Before stating the correspondence of RDF and RDF bipartite graph maps, we formulate a lemma which will be useful for the proof.

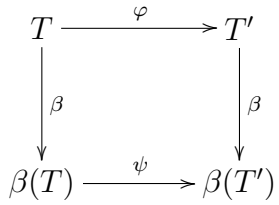
Lemma 1: Let $\varphi : T_1 \to T_2$ be an RDF map. The following diagram com-

mutes:



(Trivial) □

Proposition 12 (Correspondence of Maps): Let φ be an RDF map. For every $T \in \text{RDFG(V)}$ there exists an RDF bipartite graph map ψ such that the following diagram commutes:



□

Proof 12: First, we define $\psi : \beta(T) \to \beta(\varphi(T))$. Let $\psi_V, \psi_{St}, \psi_E$ be such that

$$\begin{aligned}
\forall\, u \in \text{univ}(T): &\quad \text{nl}(\psi_V(\beta(u))) = \varphi'(u) \\
\forall\, t \in T: &\quad \psi_{St}(\beta(t)) = \beta(\varphi(t)) \\
\forall\, e \in E_{\beta(T)}: &\quad \psi_E(e) = e' \text{ such that} \\
&\qquad \begin{aligned} \text{el}(e') &= \text{el}(e), \\ \text{val}(e') &= \psi_V(\text{val}(e)), \\ \text{stat}(e') &= \psi_{St}(\text{stat}(e)) \end{aligned}
\end{aligned}$$

We now prove that ψ is indeed an RDF bipartite graph map as specified by definition 20. Condition 1 is evident because φ maps only blank nodes. Condition 2 is fulfilled because an RDF map was defined to produce only valid triples. Condition 3 is exactly reflected by the definition of ψ_E. The fourth condition, $\psi(\beta(T)) \subseteq \beta(\varphi(T))$, is implied by the definitions of $\psi_V, \psi_{St}, \psi_E$ together with lemma 1. For the same reason it holds also $\psi(\beta(T)) \supseteq \beta(\varphi(T))$, which ensures $\psi(\beta(T)) = \beta(\varphi(T))$. □

Corollary 2: RDF Graph equivalence can be reduced to the standard graph isomorphism problem for bipartite graphs:

$$T_1 \cong_{\mathrm{RDF}} T_2 \quad \text{iff} \quad \beta(T_1) \cong \beta(T_2)$$

□

Conclusion This chapter phrased definitions of RDF maps and RDF Graph equivalence from [Hay04] in a more formal way. The intent was twofold: it was shown that the problem of RDF Graph equivalence can be reduced to the graph isomorphism—a result, which is not new, but which has been stated formally here. Second, it turns out that maps on RDF bipartite graphs can fully represent RDF maps.

It is interesting to note that isomorphism among both directed labeled graphs and RDF bipartite graphs correspond to RDF Graph equivalence. This is due to the fact that RDF Graph equivalence relies on a map of blank nodes, which never appear as statement properties. As the main difference between directed labeled graphs and RDF bipartite graphs consist in the incorporation of properties (and statements) as nodes into the graph, it is obvious that both graph models are adequate to represent RDF Graph equivalence.

8. The Structure of an RDF Graph

This chapter attempts an investigation of the *structure* of an RDF Graph. Of special interest is the relation between the data and the schema-defining part of an RDF specification, which will be examined from a graph point of view.

RDF is an extensible framework to express assertions about information resources. The core feature of extensibility implies that only a minimal vocabulary with predefined semantics is provided (the vocabulary with `rdf` namespace). RDF Schema is a special vocabulary (`rdfs` namespace) introduced by the designers of RDF to provide some elementary means to define other vocabularies.

It is common practice to distinguish an RDF Graph into a data and a schema part. However, what is the schema part? This issue is seldomly addressed. The implicit answer is that the set of statements making use of the `rdfs` namespace vocabulary is the schema part. However, this does not consider the extensibility of RDF, especially the ongoing work on ontology definition languages, which one day might provide a more sophisticated mechanism than the `rdfs` vocabulary for vocabulary definition.

In this chapter, an attempt is made to distinguish the schema part from the data part of an RDF Graph without relying on the `rdfs` namespace prefix.

8.1 Schema, Metaschema, Axiomatic Triples

A *Schema* is an "an internal representation of the world" [Mil].

Primarily, the RDF Schema is a special vocabulary to structure other vocabularies. The semantics of this vocabulary as well as the core RDF vocabulary are defined in the RDF specification. On this principle—a relatively small vocabulary with predefined semantics to define new vocabularies—grounds the extensibility of RDF. At the time of writing, many RDF vocabularies exist which make use of RDF Schema to provide a machine-accessible understanding of it. However, RDF Schema's expressiveness is limited to basic assertions about class and property relationships. "Richer" schema-

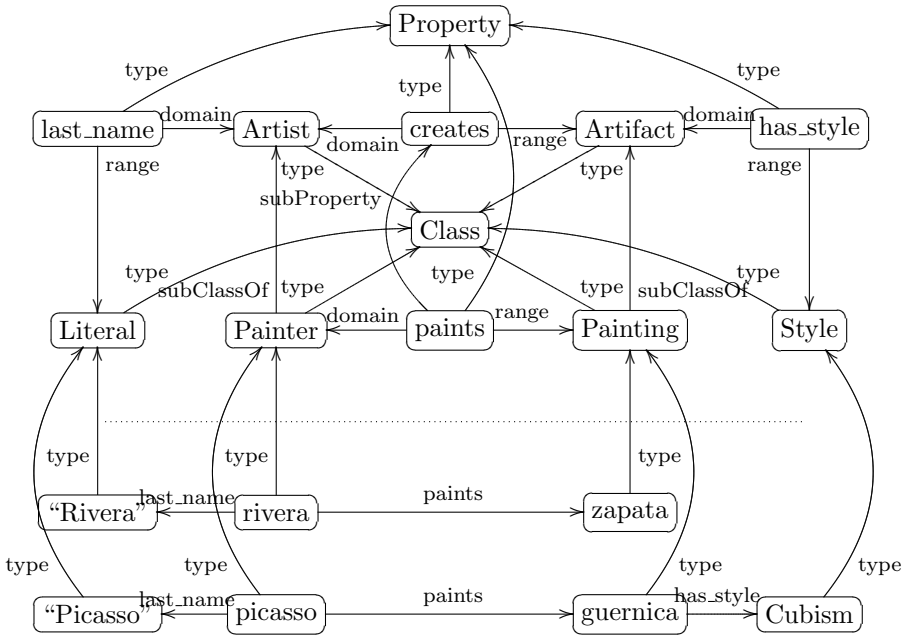


Fig. 8.1: Directed labeled graph drawing of the museum example. The dotted line divides the *data part* (below) from the *schema part* of the graph.

defining languages (e.g., DAML+OiL[1], OWL[2]) continue to evolve, enhancing the way agents can automatically process RDF data.

The Resource Description Framework provides the means to express the meaning of an RDF specification within itself, if only a part of it (be it the RDF Schema vocabulary or something more expressive) is known. But how can the schema-defining statements be identified? The preceding arguments motivated the need of other criteria than just relying on the `rdfs` namespace.

Numerous publications discuss RDF schema, but very few address the question how data and schema of an RDF Graph are interwoven. Among those which came to the author's attention is *RQL: a functional query language for RDF* [KMA+04], where an RDF Graph is decomposed into a *description base* and a *description schema* (an approach discussed more in detail below). However, it is not addressed which part of an RDF Graph is description base and which is description schema.

The authors of said work also introduce the notion of *metaclass* for

[1] DAML+OIL, `http://www.daml.org/2001/03/daml+oil-index.html`
[2] Web Ontology Language (OWL), `http://www.w3.org/2004/OWL/`

`rdfs:Class` and `rdf:Property`. The idea of a "meta-schema" is appealing, because the so-called (they are characterized only as "informative") axiomatic triples [Hay04] form a layer above the actual schema.

A different approach is followed in [PH03, PH01], where a stratification of RDF in different schema levels is presented, which is, by default, not bounded. However, it is based on an alternative RDF Schema (*RDFS(FA)*).

The latter approach is the one favored by this author, as it embraces RDF's extensibility. For example, in figure 8.2 an RDF graph with four layers is depicted. While it may seem unorthodox to use the data part of one RDF Graph (here: WordNet) as the schema of another, this is a perfectly valid RDF example.

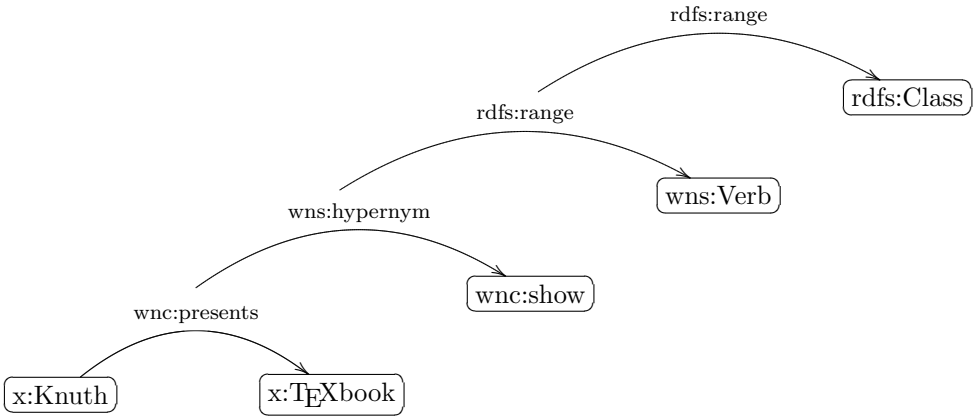


Fig. 8.2: This example shows a statement ("Knuth presents the TEXbook") using as a schema the description base of another RDF specification (here: WordNet). The schema of the latter forms the third layer, while RDF axiomatic triples form the fourth, or meta-meta schema layer.

8.2 An Approach for Data/Schema-Partition

RDF Graphs consist of values and statements. A first coarse-grained division of the statements is the following: those that define a schema, and those that are data structured under this schema (see figure 8.1—the dotted line represents this division). Although RDF does not distinguish between schema-

defining and data statements, this distinction is natural when considering storage [MAYU03] and querying [KAC+02] in databases. Unfortunately, a plain distinction between the data and schema parts of an RDF specification is not always possible. Moreover, features like extensibility of specifications and reification make this divide difficult to grasp formally. In this and the following section we present two approaches to this issue.

Definition 22: A *data subgraph* of an RDF Graph T is a maximal subgraph T' satisfying $(\text{subj}(T') \cup \text{obj}(T')) \cap \text{pred}(T') = \emptyset$. The *schema subgraph* associated to T' is $T \setminus T'$. □

The motivation for this definition is that the data subgraph of conventional RDF Graphs is usually easily partitioned into resources being described and properties describing them. This holds for many of the examples around, including the museum example or WordNet. In these examples description base properties only occur as subject or object in schema definitions—as, for example, `paints` in figure 8.1. However, a problem is the fact that reification statements would be categorized as part of the schema, too.

Notice that an RDF Graph T does not have a uniquely defined data-subgraph. For example, consider $\{(a, b, c), (b, d, e)\}$ where each statement alone is a data subgraph. Moreover, an RDF Graph could have exponentially many different ones:

Proposition 13: The number of different data subgraphs of T can be exponential in $|T|$. □

Proof 13: Let n be a positive integer. Consider an RDF Graph

$$T = \bigcup_{1 \leq i \leq n} \{(a_i, b_i, c_i), (b_i, d_i, e_i)\}$$

which contains n 2-statement fragments which are connected only among themselves. For each, two choices of data/schema partition can be made, yielding 2^n possible partitions for the entire RDF Graph. □

This section explored an intuitive approach for partitioning an RDF Graph into data/schema parts. As it did not perform very favorably, we consider another approach in the following section.

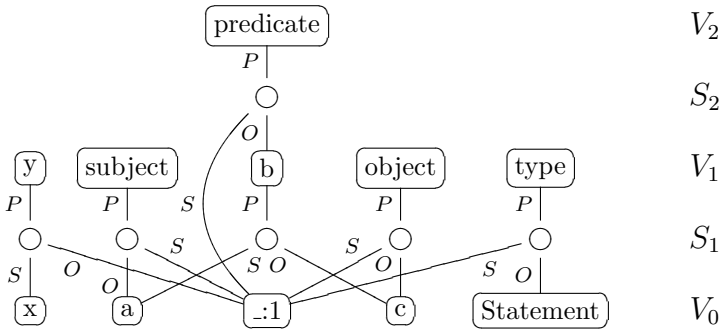


Fig. 8.3: Stratified drawing of a reification. A resource `x` makes a proposition `y` about the statement <`a b c`>.

8.3 Stratifying RDF Graphs

An alternative approach is to decompose the RDF Graph T into *strata*. This is grounded on the idea that a property describing basic-level entities is a predicate, and a property describing predicates is a predicate of higher order.

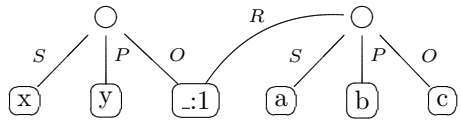
Definition 23 (Stratification): Let T be an RDF Graph. Define $V_i(T)$ and $St_j(T)$ by mutual recursion as follows: $V_0(T)$ is the set of values of T that are not predicates, $V_n(T)$ is the set of values of T that are predicates of statements in $St_n(T)$, and $St_{n+1}(T)$ is the set of statements of T whose subject and object are elements of $\bigcup_{j \leq n} V_j(T)$.
The *order* of T is the maximum n such that V_{n-1} is not empty.

T is *stratified* if $\text{univ}(T) = \bigcup_{j \geq 0} V_j(T)$.

We denote $v_i \prec v_j \Leftrightarrow v_i \in V_i(T), v_j \in V_j(T)$ and $i < j$. □

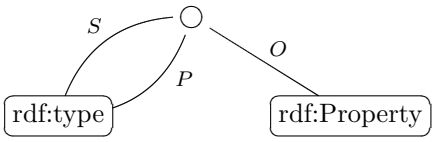
Example 11 (Reification): The reification[3] of a triple <`a b c`> is stratified

[3] Although unrelated to the issue of stratification, it shall be noted that the RDF bipartite graph model offers an economic way to represent reifications (space consumption of reifications can be a problem [Jai04]). Instead of the standard way which adds four more statements as depicted in Figure 8.3, a "shortcut" would be to add a fourth edge to the statement node to be reified (labeled in a distinct way, e.g., by 'R') directly to the blank node representing that statement:



(see Figure 8.3). □

Example 12 (Axiomatic Triples): $T = \{(a, a, b)\}$ can not be stratified. This matters, e.g., for the axiomatic triple <`rdf:type rdf:type rdf:Property`>:



It results in $V_0(T) = \{b\}$ and $V_j(T) = \emptyset$ for $j > 0$. □

Example 13 (Museum Example): The museum example (figure 8.1) can be partitioned in three levels (see figure 8.6). □

We now characterize unstratified RDF Graphs:

Proposition 14 (Unstratifiedness): T is not stratified iff there is a cycle from a value node in $\beta(T)$ with label in

$$[(O|S)P]^+.$$

□

To prove the proposition, we make use of the following lemma.

Lemma 2: Let v_i and v_j be values in an RDF bipartite graph $\beta(T)$, and p be a path connecting v_i with v_j, and label(p) $\in ((O|S)P)^+$. This is equivalent to $v_i \prec v_j$. □

Proof 2 (Lemma): From definition 23 it follows that in a stratified RDF Graph T for a value $v_i \in V_i(T)$ all statements using v_i as predicate are member of some $St_k(T)$ with $k \leq i$, and all statements using v_i as a subject or object belong to a $St_j(T)$ with $j > i$. This implies $v_i \prec v_j$ if there exists a path connecting v_i with v_j which has a label in $((O|S)P)^+$ (cf. Figure 8.4). □

Proof 14: Lemma 2 gave a characterization of "vertical paths" (crossing layers) in a stratified RDF Graph by means of a regular edge expression in the RDF bipartite graph representation. A cycle with label $((O|S)P)^+$ signifies a circular $\prec$ relation between the value nodes of the cycle which can not exist in a stratified RDF Graph. Hence, T is not stratified. □

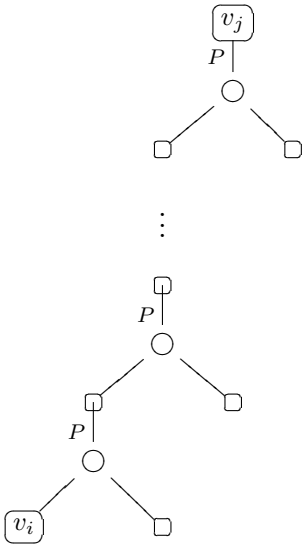


Fig. 8.4: Illustration for proof of lemma 2. The edge labels S and O were omitted, because the order of their appearance is without significance here.

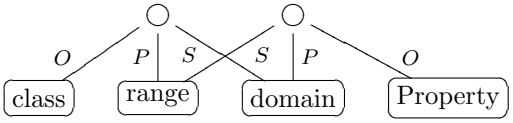


Fig. 8.5: Example of an unstratified RDF Graph.

Although the complete axiomatic specification of RDF is not stratified (see, e.g., Figure 8.5), most RDF data currently found in practice is stratified—if RDF axiomatic triples are not considered—and has small order (no bigger than 3). Also, if T is stratified, the RDF Graph obtained by reifying each of its triples is stratified. However, the union of two stratified RDF Graphs is not necessarily stratified (for example, consider again Figure 8.5: each of the statements by itself is stratified, but not their union).

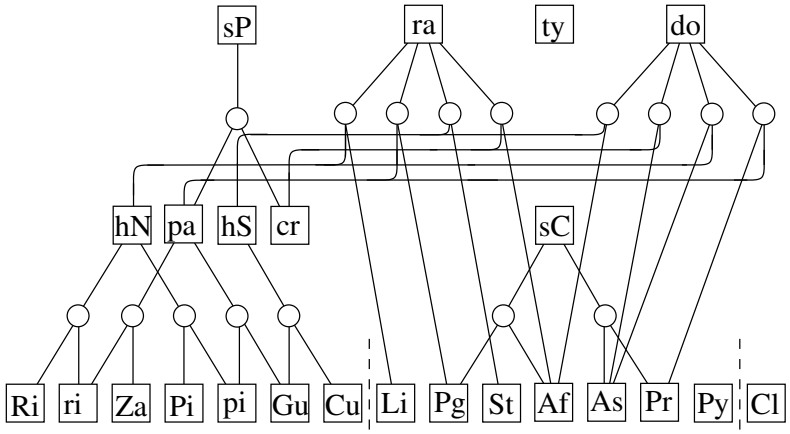


Fig. 8.6: RDF bipartite graph of the museum example. Edge labels have been omitted for clarity. Drawing levels indicate the order of the values nodes. At the bottom level are values which never occur as predicates: class instances like (bold letters indicate the abbreviations) the literal "**Ri**vera", **ri**vera (resource), **Za**pata, classes such as **P**ainting, **A**rti**f**act, **A**rti**s**t, **P**ainte**r**, **P**roperty, and the meta-class **Cl**ass. Simple properties include **h**as_**N**ame and **pa**ints, and properties of properties are **s**ub**P**roperty, **do**main, **ra**nge and **ty**pe.
Declarations with property "type" are shown in figure 8.7

Conclusion

This chapter presented two approaches to partition a RDF Graph in data and schema parts. Especially the second one is interesting, because it is conform with the idea of higher level predicates from logic, and because of a certain similarity to what is presented in [PH03, PH01]. This deserves a more thorough study in order to re-conciliate the approaches.

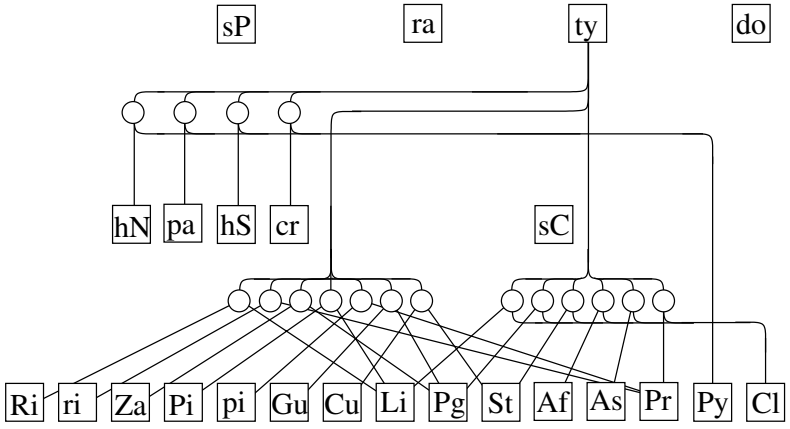


Fig. 8.7: Type declarations of the RDF bipartite graph in figure 8.6

9. Connectivity of RDF Data

This chapter formally introduces the notion of "connectivity" and studies the relevance of paths in an RDF Graph.

Preamble The abstract triple syntax of statements consisting of a subject, a predicate and an object reflects the assertional nature of RDF. However, for meaningful processing of RDF data a more fine-grained view concentrating on individual information resources rather than entire statement triples is desirable.

Guha et al. distinguish between a *logical* and a *physical* model in the classic article *Enabling Inferencing* [GLMB98]. In the context of RDF such a logical model—to be used for, e.g., querying or inferencing—would have to truly represent the graph nature of RDF in contrast to a concrete serialization syntax (the "physical model") such as RDF/XML. However, currently, database schemas for RDF storage systems (see section 10.2) and the design of RDF query languages (section 10.3) reflect a statement-centric view of the RDF model.

Because of this fact we experience the following inconveniences:

- There are conceptually very simple queries which are essential for common RDF use cases, which are not or not directly supported in contemporary query languages. Example: *"are resources X and Y connected?", "what is the degree of node Z?"*
- Directed labeled graphs convey only a restricted notion of connectivity, because a distinction is made between properties and information resources.
- The transitive closure of a relation is relevant for some RDF queries.
- Schema specifications instantiate properties as statement subjects, so schema-aware querying combines the needs for the transitive closure computation (for, e.g., the `rdfs:subPropertyOf` property) with support for "vertical connectedness" (to be defined in this chapter).

Paths appear as the lever to approach the meaning of an RDF Graph. This has been recognized for RDF querying, however a critical examination in subsection 10.3.1 shows limitations for contemporary query languages. This chapter explores the fundamental notion of *connectivity* on which proposals for advanced querying (subsection 10.3.2) rely.

9.1 Paths in an RDF Graph

This section will introduce the notion of paths and connectivity for RDF Graphs as collection of triples and RDF bipartite graphs.

The term "connectivity" comes from graph theory, where *k-connectedness* of a graph describes that at least k nodes must be removed to make the graph unconnected. In the context of RDF one rather cares for the mere fact of connectedness of two resources, and then by which sequences of RDF statements they are connected.

The intuitive notion of "connected resources" in an RDF Graph relies on the existence of a *path*:

Definition 24 (Triple Path): Let T be an RDF Graph. A *path in an RDF Graph*—or *triple path*—P is a sequence of RDF triples $(t_1, t_2, \ldots, t_n)$ with $t_k = (s_k, p_k, o_k) \in T$, for which it holds that

$$\text{for all } i < n, \{s_i, p_i, o_i\} \cap \{s_{i+1}, p_{i+1}, o_{i+1}\} \neq \emptyset.$$

A path is said to be *statement-loop-free* if

$$t_i = t_j \quad \Rightarrow \quad i = j,$$

and *value-loop-free* if

$$(x \in t_i \text{ and } x \in t_j) \quad \Rightarrow \quad j = i \text{ or } j = i + 1.$$

The *length* $|P|$ of a path in an RDF Graph is the number of statements it contains. □

Definition 25 (RDF Connectedness): A triple path $(t_1, t_2, \ldots, t_n)$ is said to be *connecting* resources x and y if it holds that $x \in \{s_1, p_1, o_1\}$, $x \notin \{s_i, p_i, o_i : 1 < i \leq n\}$ and $y \in \{s_n, p_n, o_n\}$, $y \notin \{s_i, p_i, o_i : 1 \leq i < n\}$. □

These definitions reflect the broadest sense possible for the connectedness of resources in an RDF Graph. In the course of the following studies, this notion will be restrained to match common use cases.

The concepts "triple path" (or "path in an RDF Graph") and "RDF connectedness" stress the anchoring in the RDF model and the independence to the use of whatever graph representation (e.g., RDF bipartite graph or directed labeled graph) or serialization syntax.

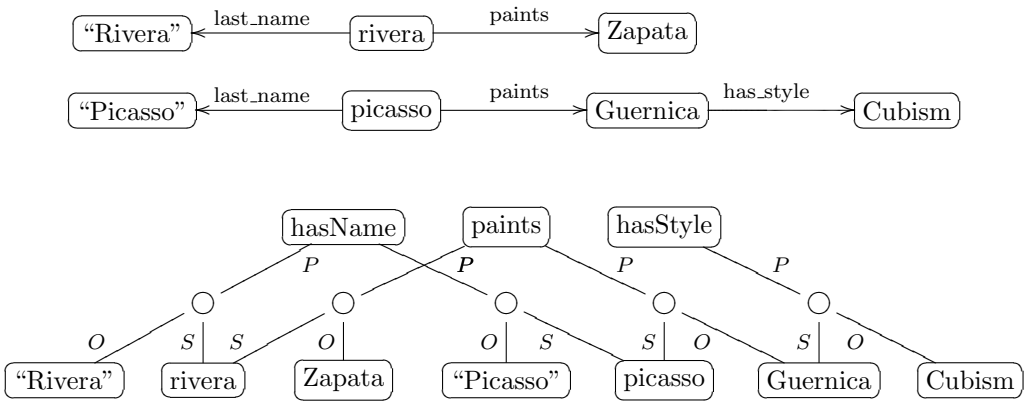


Fig. 9.1: Fragment of the museum example, presented first as a directed labeled graph, and below as RDF bipartite graph.

Example 14 (Connectivity): Figure 9.1 depicts a fragment of the museum example. For example, `picasso` and `Cubism` are connected by a path of length 2 in the directed labeled graph representation. □

The definition of RDF triple paths corresponds precisely to the well-established notion of paths and connectivity in graphs:

Proposition 15: Let T be an RDF Graph. The resources x, y are connected in T iff there exists a path between $\beta(x)$ and $\beta(y)$ in $\beta(T)$. □

Proof 15: ($\rightarrow$) Let $x, y \in \text{univ}(T)$ be connected values of an RDF Graph T. Then a sequence of statements $P = (t_1 \ldots t_n)$ exists such that $x \in \{s_1, p_1, o_1\}$ and $y \in \{s_n, p_n, o_n\}$; furthermore, there exist values $z_1, \ldots, z_{n-1} \in \text{univ}(T)$ with $z_i \in t_i$ and $z_i \in t_{i+1}$. By definition 15 there exist in $\beta(T)$ value nodes $\beta(x), \beta(y)$ and $\beta(z_1), \ldots, \beta(z_{n-1})$ representing these RDF Graph values; and statement nodes $\beta(t_1), \ldots, \beta(t_n)$ representing the statements $t_1, \ldots, t_n$. Because $x \in t_1$ and $y \in t_n$ there exist edges $\{\beta(x), \beta(t_1)\}$ and $\{\beta(y), \beta(t_n)\}$, and $\{\beta(z_i), \beta(t_i)\}$, $\{\beta(z_i), \beta(t_{i+1}\}$ because of $z_i \in t_i$ and $z_i \in t_{i+1}$ for all

$i < n$ (edge labels are not considered). From this, it follows that there exists a path $\{\beta(x), \beta(t_1)\}$, $\{\beta(t_1), \beta(z_1)\}$, $\{\beta(z_1), \beta(t_2)\}$, $\ldots \{\beta(t_{n-1}), \beta(z_{n-1})\}$, $\{\beta(z_{n-1}), \beta(t_n)\}$, $\{\beta(t_n), \beta(y)\}$ connecting $\beta(x)$ and $\beta(y)$ (cf. figure 9.2). (This path is said to be the *corresponding path* to the triple path P, denoted $\beta(P)$.)

($\leftarrow$) Let $\beta(T)$ be an RDF bipartite graph representing an RDF Graph T. Let $e_1, \ldots, e_{2n}$ be a path connecting value nodes $\beta(x)$ and $\beta(y)$. The path has the form $\beta(x), \beta(t_1), \beta(z_1), \beta(t_2), \ldots, \beta(t_{n-1}), \beta(z_{n-1}), \beta(t_n), \beta(y)$ ($\beta(t_i)$ are statement nodes, $\beta(z_i)$ are intermediary value nodes)[1]. It follows that the statements t_i which are represented by the statement nodes $\beta(t_i)$ are connected in the sense of definition 25, so the values x and y represented by the connected value nodes $\beta(x), \beta(y)$ are equally connected. □

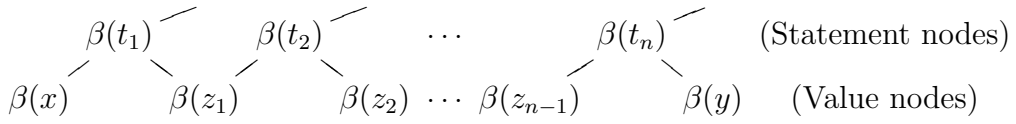


Fig. 9.2: A path in an RDF bipartite graph corresponding to a triple path in an RDF Graph as used in proof 15. Edge labels have been omitted, as well as the third value node every statement node is connected to.

Corollary 3 (Correspondence of Path Length): Let P be a triple path in an RDF Graph T, and $\beta(P)$ the corresponding path in $\beta(T)$. Then $|\beta(P)| = 2\,|P|$. □

Example 15 (Path in an RDF bipartite graph): Figure 9.1 shows in the lower part the RDF bipartite graph version of the directed labeled graph depicted above. All paths of the former also exist in the latter, but due to corollary 3 their length is exactly twice that given in example 14.

This drawing reveals an interesting feature of RDF bipartite graphs: the resources `picasso` and `rivera` now appear connected (via the property `paints`, or `hasName`), although the directed labeled graph version does not show this. In the sense of the definition of RDF Graph connectivity given above, these resources are indeed connected, and the directed labeled graph version fails to represent that. □

[1] Note that paths in bipartite graphs with node classes U and V which originate and end in U have even length. Furthermore, if the path length is $2k$, the path alternates between $k+1$ nodes of U and $k-1$ nodes of V.

This section introduced the fundamental notions of path and connectivity for RDF Graphs. The RDF bipartite graph is a model for RDF which provides a 1 : 1 representation of RDF triple paths by graph paths (proposition 15).

9.2 Restrained Paths

After having introduced the notion of RDF connectivity, we study in this section the connectivity conveyed by a directed labeled graph representation of RDF.

Intuitively, paths in a directed labeled graph correspond to triple paths making use only of the connectedness of statement subjects and objects. The following definition describes such paths:

Definition 26 (Horizontal Path): A *horizontal (triple) path* P in an RDF Graph T is a sequence of RDF triples $(t_1, t_2, \ldots, t_n)$, $t_k = (s_k, p_k, o_k) \in T$, for which it holds that for all $i < n$, $\{s_i, o_i\} \cap \{s_{i+1}, o_{i+1}\} \neq \emptyset$.

A horizontal path is said to be *oriented* if $o_i = s_{i+1}$ for all $i < n$. In an analogous way, it is *inversely oriented* if $s_i = o_{i+1}$. A horizontal path is said to be *unoriented* if it holds for all $1 < i < n$ that $s_i \in t_j$ and $o_i \in t_k$ with j, k equaling either $i + 1$ or $i - 1$.

The concepts of loop-freeness and length of horizontal paths are inherited from definition 24. □

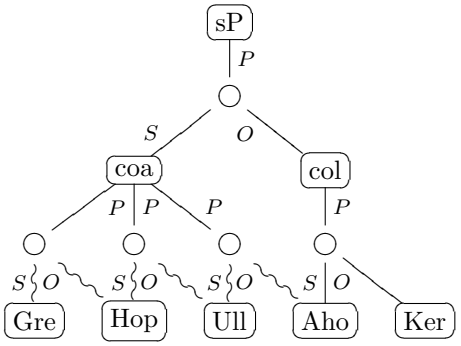


Fig. 9.3: Example of an horizontal path (curly edges) in the stratified drawing of an RDF bipartite graph.

The word "horizontal" was chosen because these paths are standard paths in a directed labeled graph and do not consider, e.g., schema specifications

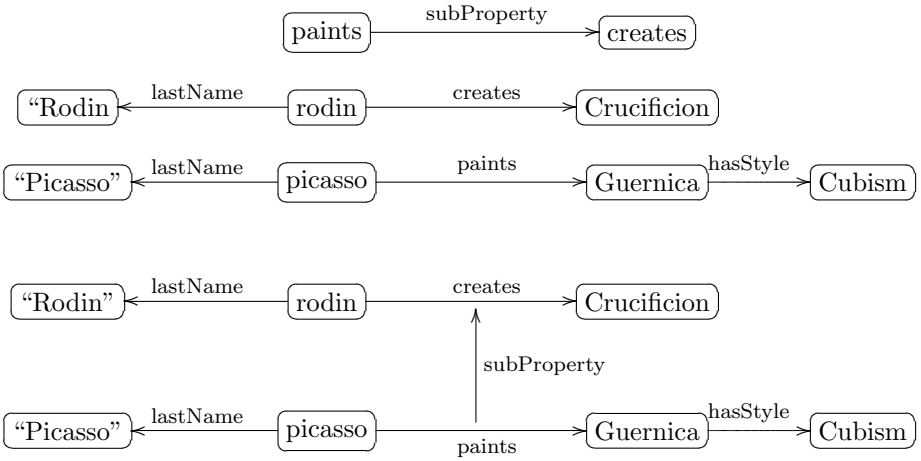


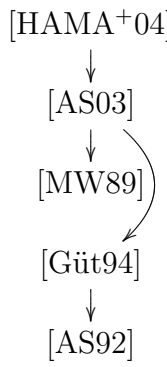
Fig. 9.4: Fragment of the museum example (slightly altered from the previous examples). The non-standard directed labeled graph drawing below emphasizes the (RDF) connectedness of the resources `rodin` and `picasso` in contrast to the restrained notion of horizontal connectivity as conveyed by directed labeled graphs.

of properties (see figure 9.4). In contrast, a path involving at least one triple predicate is said to make use of *vertical connectivity*.

If we compare to relational databases, the notion of horizontal path corresponds roughly to values of tuples linked via joins. Paths making use of vertical connectedness link tuples within the same table (subject-object pairs using the same predicate `p`) or pass through the schema of the database (e.g., via `rdfs:subPropertyOf` relationships).

Is it really necessary to emphasize the "orientedness" of a horizontal path when specifying directed labeled graph paths? It seems that horizontal paths in general would correspond to the paths in a directed labeled graph if the edge direction is not considered. However, this leads to problems, as there are horizontal triple paths which do not correspond to classic graph paths, be they directed or not, as illustrated by figure 9.5.

We now state the equivalence of oriented horizontal paths in an RDF Graph and the paths in a directed labeled graph. If it holds for a triple path $P = (t_1, \ldots, t_n) \in T^n$ and for a path $P' = (e_1, \ldots, e_n) \in E^n$ that $\delta(t_i) = e_i$



RDF Graph "Citations"
<[HAMA+04] cites [AS03]>
<[AS03] cites [MW89]>
<[AS03] cites [Güt94]>
<[Güt94] cites [AS92]>

Fig. 9.5: Example of an RDF Graph and a horizontal path, which consists of all the four statements of the graph in the order they are given at right. From the second to the third statement, the condition of orientedness is violated, thus, the directed labeled graph representation of the graph does not contain a path containing all four statements.

for $1 \leq i \leq n$, we say that P' is the corresponding path to P and write $P' = \delta(P)$.

Proposition 16: For every oriented horizontal path P in an RDF Graph T there exists a corresponding path $\delta(P)$ in $\delta(T)$, and vice versa. □

Proof 16: The existence of a path $\delta(P)$ in $\delta(T)$ for an oriented horizontal path $P \subseteq T$ is obvious. Let $P' = (e_1, \ldots, e_n)$ be a path in $\delta(T)$. By definition of δ there is a $t_i \in T$ for every e_i such that $\delta(t_i) = e_i$. Because of $\text{to}(e_i) = \text{from}(e_i + 1)$ for $i < n$ it follows that $o_i = s_{i+1}$ for the $t_i = (s_i, p_i, o_i)$, so $(t_1, \ldots, t_n)$ is an oriented horizontal path in T, and $P' = \delta((t_1, \ldots, t_n))$.

□

This section examined the relation between paths in a directed labeled graph representation of RDF in contrast to RDF triple paths as introduced in the previous section. It was shown that directed labeled graph paths correspond to *oriented horizontal paths* in an RDF Graph. This class of paths is a proper subset of RDF triple paths, which are fully present in an RDF bipartite graph representation. One can thus conclude that directed labeled graphs convey only a limited sense of connectivity, what makes RDF bipartite graphs the model of choice for advanced studies.

9.3 Relevancy of Paths

The last section introduced a number of concepts to characterize restrained paths in order to grasp the notion of connectedness for directed labeled graphs. This section shall motivate to take advantage of the "full" amount of RDF connectivity as presented in definition 25 by means of examples for the relevancy of each class of paths defined above.

9.3.1 Various Horizontal Paths

Oriented Horizontal Paths This class of paths in an RDF Graph is what usually is referred to when discussing path expressions for RDF. Several examples have been provided above.

Unoriented Horizontal Paths Oriented horizontal paths and inversely oriented horizontal paths do not differ conceptually. It is, however, interesting to note that current query languages do not support path expressions *without* specifying the subject or object role of a resource:

> Consider a query for the relationship between two scientists A and B in an RDF Graph with the only property `cites`. The user is interested in any path between A and B, regardless of the "direction" of the citation.

It will be explained on page 123 that this query must be formulated as a union of an exponential number of sub-queries, which is, of course, unsatisfying.

As mentioned above, there is a class of paths which is horizontal but not unoriented, Figure 9.5 provides an example. The author is not aware of relevant use cases.

9.3.2 Vertical Paths

Paths making use of "vertical connectedness" via statement predicates shall here be explored to some extent. The most common cases of a property appearing as a statement subject or object is by

- Schema specification. `range`, `domain`, and `type` are schema properties which are applied to (data) properties. The most relevant is

`rdfs:subPropertyOf` because paths via this property are relevant for schema-aware querying.

- Reification. If <`s p o`> is a statement to be reified, one of the statements added in this process is <`_:x rdf:predicate p`>.

These are surely the two most common cases, however, by the RDF specification, there is no restriction governing this. For example, in figure 8.2 on page 95 WordNet is used as a schema for common concepts, and the proper RDF schema of WordNet represents a second schema level.

Paths through the schema of an RDF Graph are relevant for meaningful, or "schema-aware" querying:

For the museum example, consider a query for all artists which have produced artifacts of Cubism. In the preceding examples `creates` as well as `paints` appear as producing activities. Making use of the fact that `paints` is a subproperty of `creates`, one can formulate the query as (in a RQL-like [FOR03] syntax) `select X where X creates Y, Y hasStyle Cubism`.

Such a query can be implemented by first computing the transitive closure of the `rdfs:subPropertyOf` relation for the `creates` property, and then computing the conventional (horizontal) paths for the real query.

An almost trivial use of vertical connectedness is made in the following example:

Consider figure 9.6. The RDF bipartite graph shows the connectedness of `Erdös` and `Valiant` via the `researches` property.

Although somewhat straightforward, the relationship between the two resources relies on vertical connectedness.

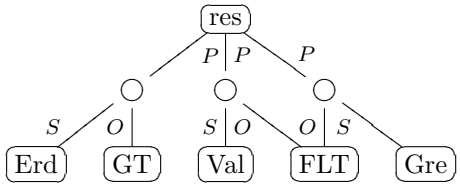


Fig. 9.6: Representing statements 4-6 of the RDF Graph on page 47.

It turns out that this example is the simple a case of a more complex usage of vertical connectedness. In [AS03] *similarity* between paths is introduced. Informally, two paths of equal length are considered similar, if every i-th resource (i-th property) of both paths share a common ancestor class (ancestor property) by means of the `rdfs:subClassOf` (`rdfs:subPropertyOf`) relation.

In the last example, Erdős and Valiant and Graph Theory and Formal Language Theory have common ancestor classes (although not stated in the example), and the usage of the same property `researches` makes them a trivially similar path.

A more general example is shown in Figure 9.7. Restriction of the ancestor classes for the subject/objects have been omitted.

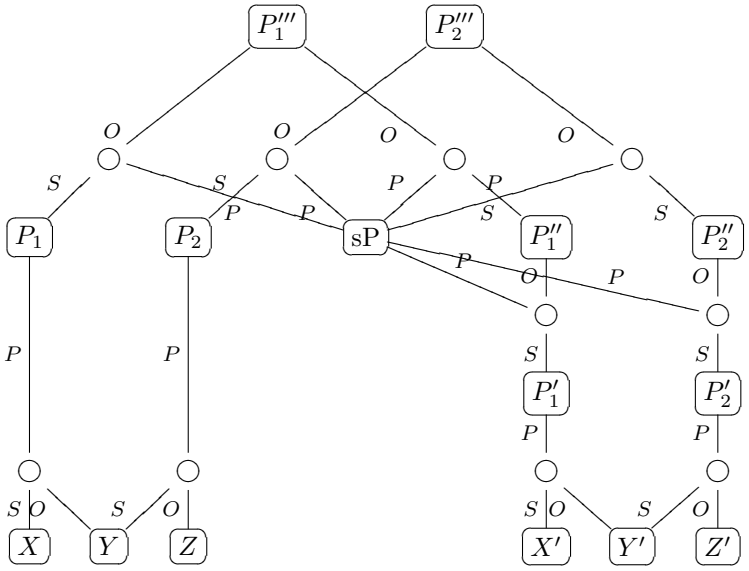


Fig. 9.7: Similar Paths: paths `X`,`Y`,`Z`, and `X'`,`Y'`, `Z'` are similar, because each property has a common super-property.

Vertical Paths: Conclusion This section illustrated through various examples the usefulness of vertical paths for querying and analyzing RDF data. Relevant applications include schema-aware querying and mining for semantic associations.

9.3.3 Arbitrary-Length Path

The discussion above hardly touched the question if the paths in study would be of fixed, or of arbitrary length. From a graph theoretic point of view this does not matter, standard query languages for relational databases, however, normally don't offer so-called *recursive* queries (see, e.g., [CM90, Agr88, BFS00]). Use cases for arbitrary-length querying are presented in greater depth in chapter 10, here we just present a very brief summary of two relevant notions in order to complete the above study of paths in an RDF Graph.

Transitive Closure of a Relation Many properties commonly used in RDF data are transitive or have, if understood as transitive, a derived meaning. Examples include `cites` or `coauthors` which are by themselves not transitive relations, but their transitive closure conveys a certain meaning, such as the *influence* of an article (all other articles relying on it by repeated citation). The schema properties `rdfs:subClassOf` and `rdfs:subPropertyOf` are transitive, and their transitive closure is relevant for schema-aware querying.

Connectedness The very basic question *"are resources X and Y connected?"* requires that it can be queried for a path of any length between X and Y.

9.3.4 Metrics and Semantic Associations

This subsection shall briefly explore how relations between information resources can be characterized through paths in RDF data.

In mathematics, a metric space is a set of points with an associated distance function. In the context of RDF, it is useful to introduce some kind of *distance metric* between information resources to approach the intuitive notion of relatedness between resources. The relation between resources is founded on the paths which connect them. However, simply taking the path length as a distance metric would be a naive approach if the meaning of the connecting properties is not taken into account[2].

A study of such a distance or similarity function is conducted in [RE03], which concentrates on the integration of different ontologies to deduce the

[2] The length of the shortest path between resources hardly conveys anything, as any two resources `X` and `Y` are connected by a path of length 2: <`X rdf:type rdf:Resource`> and <`Y rdf:type rdf:Resource`>

relation between entities. Sheth et al. presented an approach for this in *Context-aware semantic association ranking* [AMHAS03] for RDF / RDF Schema.

"Semantic associations" are complex relationships between entities and rely on the *connectivity* and *similarity* of "property sequences" (unoriented horizontal paths in RDF Graphs) [AS03]. The work is formally founded on the typing introduced by [KAC+02] and to a lesser extent on notions from graph theory. Ongoing work of Sheth et al. includes [AMHAS03, HAMAS04, SAMA+04].

Conclusion

This chapter formally introduced the notion of *connectivity* as a fundamental concept for the interpretation of RDF data. It was proven that directed labeled graph represent only a restrained sense of connectivity. *Vertical connectivity*—which is not conveyed by directed labeled graphs—is essential for schema-aware RDF querying and complex semantic association finding. On the other hand, it was proven that RDF bipartite graphs reflect the full extent of RDF connectivity, which therefore appear as an appropriate model for querying and interpreting RDF data.

10. Observations on RDF Graph Storage and Querying

This chapter approaches storage and querying aspects of RDF. First, in a "motivating preamble", some challenges of RDF to storage and query systems are summarized. Then the contemporary storage systems Sesame and Jena are presented. A third section—the actual contribution of this chapter—proposes graph primitives as a basic construct for RDF query languages.

10.1 Preamble

As the web continues its growth, and the use of structured metadata emerges from its current niche role towards common practice, databases technology will be challenged to provide adequate solutions. The problem is not alone the sheer amount of data resulting from these two trends—the important role of connectivity and peculiarities when merging RDF data result in difficulties despite the very simple syntax. Finally, some obvious use cases let current querying techniques appear insufficient.

Reasons supporting these claims shall be given in this section.

Harvesting Metadata The main method of serving metadata is already implied by the phrase "metadata annotation". Thus, a large portion of metadata will reside either "in" the media to be described, or be linked to it. Search engines today capture a representation of the web in their indexes. As the quantity of machine-understandable metadata grows, it is probable that search engines will harvest metadata (and distill, transform, process, infer) into databases for offering advanced search services to their users.

It is interesting to note that intelligence agencies are showing a strong interest in related technologies, as manifested, e.g., in [SAMA+04]. As a difference to the outlined scenario of semantic search the emphasis is to use

RDF as a common format to integrate data collected by different organizations with diverse intentions, tools, etc. [Bra03a].

If intelligence agencies elected RDF as a greatest common denominator for representation of any kind of information, it may be speculated if RDF will emerge as the standard back-end data format for storing heterogeneous information; applications could include Data Warehousing or business intelligence knowledge discovery tools (e.g., [Sem04]).

Wild Data At a first view, RDF data is extremely simple: it consists of uniform triples of resources or literal strings. However, as argued before, the triple-centric view does not take into account the essential features of RDF, which are, for storage and querying issues, the prominent role of connectivity, and the possibly heterogeneous sources of RDF datasets which are merged into one to be stored and queried together.

The fundamental role of RDF connectivity and paths for the interpretation of RDF data became evident in the preceding chapter. As an important use case of RDF consists in the collection and analysis of metadata, it is of central interest how the *union* of diversely obtained RDF data can be produced. A tough (NP-complete) issue for obtaining the union of RDF datasets is how anonymous resources can be identified with each other, as their *blank node identifier* is only of local scope for the dataset it was generated in. Opting for a complete disambiguation (*merging* in RDF terminology) is naive, as anonymous resources often represent complex identities such as persons, and their correct identification is a central requirement for several use cases.

But leaving aside the issue which arises from blank nodes when uniting or updating RDF models, determining the *format* in which RDF data is provided is a problem, too. RDF Schema allows the definition of vocabularies, and RDF descriptions making use of only one well-defined vocabulary can be easily processed and queried. However, if several vocabularies are mixed, as it is practiced already commonly, how shall the various schemas be integrated?

Another problem arising when merging RDF data from various sources is that reification may be used to mark the origin, or, e.g., the level of confidentiality, of the data (for example, [WSKR03] reports that some applications reify each statement). As every reification adds five more statements, the overall quantity and complexity of the data rises considerably.

We see that despite a trivially simple syntax, RDF data presents problems when integrating it from various sources. But not only the data, also the use cases are demanding. This makes wonder if standard databases are

appropriate: multiple-source data, essentiality of full connectivity support, demand for complex queries, and no clear schema are a real challenge beyond the scope of classical business database applications (e.g., airline reservation systems, accounting systems, ...) [Tuc04].

Data and Schema Entangled Other than in conventional storage systems, for RDF, the schema is not separated from the data: the schema appears within the data. Chapter 8 already addressed the issue of how to determine the schema of an RDF Graph, assuming that relying simply on the `rdfs` namespace prefix, given the extensible nature of RDF, is naive. What is more, several levels of a schema may be present, as depicted by the not-so-exotic example in the figure on page 95.

Relevant Queries As argued, the composition of RDF data may be complicated. In addition, already common use cases require sophisticated queries. It is doubtful whether traditional SQL-like query languages can provide meaningful querying, as they are geared towards tuple storage, and, for example, typically do not offer recursion on query results.

The "semantic associations" as introduced by Sheth et al. give a good guideline to formulate the requirements for adequate query languages. This chapter proposes query primitives based on graph concepts as a first step into this direction. We conclude that *subgraph matching*, as already provided in some RDF query languages, (e.g., RQL) gives the core of the required functionality, but also the opportunity to formulate indirection and the ability to compute arbitrary-length paths are needed. (Regular) path expressions on node/edge labels therefore appear as an interesting perspective to study.

Schema-aware querying is an essential feature already provided by some languages. However, one of the first storage/query systems to support this characteristic, Sesame, infers the transitive closure of the `rdfs:subClassOf` and `rdfs:subPropertyOf` relationship and stores it explicitly, an approach for which it is doubtful whether it scales for large schemas.

Another issue is that even if the schema of diverse RDF datasets is accessible for machine agents, human users may not be able to get an intuitive understanding for the structure of the data by just seeing the schema. A query system aiming for insightful queries should therefore allow various degrees of impreciseness.

Conclusion, Outlook The intent of this section was to recall the challenges of *adequate* (a term to be defined in this chapter) RDF storage and querying.

This chapter presents a first approach for enhancing the processing of RDF data.

10.2 RDF Storage

This section gives an overview of three common ways to store RDF data: the two openly available storage systems Jena and Sesame, and the standard XML serialization, RDF/XML.

10.2.1 RDF/XML

RDF/XML is a syntax to serialize an RDF Graph into XML[1]. A graph-like data structure as RDF can be mapped onto a tree-model as offered by XML, but can not be truly represented by it: an XML encoding of RDF does not properly represent the connectivity of the RDF Graph, and is thus, as a data structure, unfit for querying. Another issue is that RDF/XML is ambiguous in the sense that many XML serializations of the same RDF Graph can be obtained.

These facts are stated more formally in [CS04], along with a presentation of alternate XML serializations. However, the fundamental issue of the inadequateness of a tree-model to represent a graph structure such as RDF persists.

10.2.2 Jena

Jena is a "Semantic Web Framework", offering a Java programming interface, a database subsystem, and a query language (RDQL) [CDD+03, WSKR03].

Jena's original design (Jena-1) used two alternative approaches to store an RDF Graph: (1) Three tables: one for statements, one for literals and one for resources. Here the main problem was the heavy use of joins to answer queries. (2) One statement table, with indexes by subject, by predicate and by object.

Experience with the first version led to a simplified schema for Jena-2. Essentially, in Jena-2 [WSKR03] triples are stored into a statement table. Only values whose string length exceeds a given threshold value are stored only once and into a separate table; a reference is stored for every occurrence

[1] Often it is not properly distinguished between RDF and RDF/XML: RDF/XML is a serialization syntax governed by the (abstract) RDF model (and not vice versa).

in the statement table instead. Thus, by varying the threshold applications may trade off space consumption for speed.

Space consumption is optimized using compression of common URI prefixes. Also, there is special treatment for *common statement patterns*, for which special property tables account for. Such patterns are the result of high-level RDF constructs as bags, sequences, or reifications, and patterns induced from regularities in the user data.

One weakness of the query system is that it does not automatically incorporates the semantics of RDFS vocabulary.

10.2.3 Sesame

The main feature of Sesame [BKvH02] is that it provides query languages (SeRQL ([Adu04], chapter 5), and a subset of RQL) which incorporate the RDF Schema semantics. The concrete data storage is implemented differently according to the underlying database system:

- PostgreSQL: this object-relational database allows explicit modeling of class hierarchies. The drawback is that RDF Schema statements which are added to the RDF Graph at a later point lead to a costly refinement of the database schema, possibly including several new table creations. The RDF data is stored into property tables with keys referring to the actual resource URIs in a resource table, according to the class hierarchy.

- MySQL is a relational database, so RDFS information is stored in separate tables. RDF Schema information which is added to a later moment does not lead to a refinement of the database schema. Inferred statements are stored together with user data, but are marked as inferred. All RDF data is stored into an RDF statement table.

The two different database implementations and the different usage of them is interesting to study the two different maps. However, the most convincing argument for Sesame remains the expressiveness and schema-awareness of its query languages.

10.3 RDF Querying

This section discusses the current state of RDF query systems and proposes improvements, based on the preceding results.

As it is widely acknowledged that the Semantic Web will largely be built on RDF, a stable and scalable infrastructure for RDF query systems is required. Especially effective and expressive querying of RDF specifications is important with respect to Semantic Web applications. Under the impression of the large metadata indexes maintained already today, e.g., by search engine providers, the issues of efficiency and scalability of RDF querying have been addressed (e.g., [WSKR03]).

However, in how far have the Semantic Web and database communities recognized the importance of "rich" or "meaningful" RDF querying[2]? Given the importance which is attributed to automated reasoning on RDF data [GLMB98] and the finding of complex associations [AS03] the expressivity of the language is of great importance. In some cases the lack of adequate query constructs may be encountered by repeated submission of simpler queries, however, there exists at least one relevant case in which as much as exponential many sub-queries are required.

10.3.1 Current State of RDF Querying

In [GLMB98] *subgraph matching* is postulated as the relevant RDF querying mechanism. Several query languages (e.g., RQL [KMA+04, FOR03], SeRQL [Adu04] (Ch. 5), RDQL [Sea04]) support graph-based querying to some extent.

Recent surveys on RDF query languages [PG01, HBEV04] contain only query use cases of limited complexity. For example, the need for the support of arbitrary-length paths—relevant, e.g., for determining if two resources are connected or not—is not mentioned. Current query languages also lack expressiveness in querying for patterns of fixed length: as stated in [AGH04] there is need for the ability to query for paths of fixed length in a manner not taking into account the subject / object role of resources (see the remark on page 123). In chapter 9 we argued that the semantics of an RDF Graph relies on the connectivity of the resources described. This is not properly taken into account in a mere triple storage, therefore it appears natural to allow resources as a basic query entity and not, as in SeRQL, only statements. Querying for the degree of a node—the number of statements a resource appears in—is an example for a query which is not built of expressions on

[2] Recently, the "RDF specific aspects" were characterized as "distribution, scalability and schema-aware querying" [Stu04]—without any reference to the expressiveness of the query model.

statements.

Aforementioned authors (Guha et al.) emphasize the "deductive closure" of an RDF model. This implies that RDF Schema information is considered for querying, especially making use of `rdfs:subPropertyOf` (subsumption of properties) and `rdfs:subClassOf` (subclassing of resource classes) specifications. SeRQL and RQL, for example, support schema-aware querying. It is interesting to note that this feature relies on the transitive closure of these schema properties; Sesame, for example, stores explicitly every relation inferred from the transitive nature of `rdfs:subPropertyOf` and `rdfs:subClassOf`.

These remarks on graph properties or RDF query languages (see [AGH04] for a more detailed survey) motivate the proposals of the next section.

10.3.2 Graph-Based RDF Query Primitives[3]

The survey [HBEV04] formulates criteria for RDF query languages to meet. Among them is *adequacy*, which is defined as

> *A query language is called adequate if it uses all concepts of the underlying data model. This property therefore complements the closure property: for the closure, a query result must not be outside the data model, for adequacy the entire data model needs to be exploited.*

Given the fact that RDF has a graph-based model, it appears natural to consider graph operations as the basic query construct in order to *adequate* support of RDF. This section proposes a collection of such operations.

Degree of a Node The degree of a value is the number of statements in which that value occurs. One can distinguish the *S, P, O-degree*, according to the number of statements the value has the role of a subject, predicate, or object. This can lead to conclusions on the role of the value, e.g., if it is a simple information resource being described, a property, a property with schema information, a part of the schema vocabulary, etc.

No current query language supports this feature directly. Some allow the retrieval of all triples the value occurs in by three queries and use of the

[3] In [AGH04] Renzo Angles, Claudio Gutierrez and this author conducted a survey on the support for graph features in current RDF query languages. This subsection is based on results of that paper, and, thus, builds on cooperative work of Renzo Angles and this author.

union operator. Using COUNT gives the number of triples returned (the degree), but this information is not presented as a RDF statement, thus not complying with the query language property of *closure*.

(Museum example) *"What is the number of predicates involving the resource* `Guernica` *?"*

Adjacent Nodes Computing the adjacent nodes of a value node v results in the set of statements that value occurs in, or, at level 2, the set of values occurring in the same statements as v. We refer to this as the *semantic neighborhood* of a node. It can be used, e.g., to retrieve the assertions which have been made about a particular resource.

(Museum example) *"What are the resources adjacent to the resource* `Guernica` *?"*

K-Neighborhood of a Node Related to the previous:

Given a value or a statement node, *"what are the nodes at most (exactly) k edges away?"*

The notion can be also extended to sets of nodes instead of a single root node. Observe that the node classes (statement or value) being reached alternate for even and uneven k.

Path Patterns Path expressions of a fixed length are supported by several query languages, such as SeRQL and RQL.

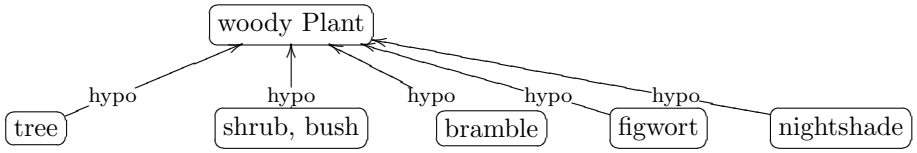


Fig. 10.1: Coordinate terms of "tree" in WordNet

(WordNet) One common use of path expressions is to find meaning-related clusters of words, such as the query for the *coordinate terms* of a word (i.e., the "sister words", all immediate sub-concepts of the super-concept of a word)—see figure 10.1. This corresponds to a path

<tree hyponymOf X>, <Y hyponymOf X>.

However, to query for such a pattern of arbitrary length (k hyponyms up, k down again) is not possible in any of today's query languages.

It appears that the triple or directed edge model results in a serious limitation of the path querying capability of current query languages. If the orientation of triples shall not be considered—e.g., when querying for paths of length k between two resources

(Museum example) *What are the paths of length* 2 *between* "Picasso" *and* Guernica*?"*

one is forced to formulate it as the union of 2^k sub-queries which explicitly gives every possible combination of triple directions:

<"Picasso" X Y> <Y Z Guernica> OR
<"Picasso" X Y> <Guernica Z Y> OR
<Y X "Picasso"> <Y Z Guernica> OR
<Y X "Picasso"> <Guernica Z Y>

Arbitrary-Length Paths Of utter importance to meaningful Semantic Web querying is the arbitrary-length path. Already the—conceptually—very simple query for mere connectedness of two resources

(WordNet) *"Are the concepts* professor *and* master *related?"*

can not be answered by current RDF query languages. More advanced, but nevertheless very essential queries [AS03] than for the *existence* of a path is the question for a *shortest path*

(Web of Scientists) *"What is the Erdős number of Alberto Mendelzon?"*—this is the shortest path via the co-authorship property between Pál Erdős and Alberto Mendelzon (3).

(Photo Co-Depiction Experiment) *What is the co-depiction path between Tim Bray and John F. Kennedy?"*—the answer is a path consisting of three photos, depicting (1) Tim Bray and Tim Berners-Lee, (2) Tim Berners-Lee and Bill Clinton, and (3) Bill Clinton and John F. Kennedy.

as well as the query for *independent paths* and *all paths* between two resources:

(Web of Scientists) *"What is the relation between scientists A and B?"*

An interesting extension of what was outlined above would be the support of regular expressions for path labels [MW89]. Also, matching subgraphs by simple graph grammar expressions deserves to be studied.

Transitive Closure of Properties The transitive closure of a property is the special case of an arbitrary-length path. For example, the transitive closure of the `rdfs:subClassOf` and `rdfs:subPropertyOf` relationship is required for schema-aware querying.

> (Web of Scientists) *"What is the* influence *of article C?"*—this requires the computation of the transitive closure of the *isReferencedBy* relation (the inverse of the *cites* property) from the root node C.

Aggregate Functions Apart of the natural COUNT on triples and/or nodes retrieved by the query, aggregate functions dealing directly with the structure of the underlying graph, such as the degree of a node, the highest degree of a set of nodes, the diameter of the graph (or a set of nodes), the distance between nodes, etc. would be useful before submitting more expensive queries.

This section presented some thoughts about how to enhance RDF querying by basic graph operations. However, a principal question remained untouched: How could this be implemented?

The modeling of graphs on conventional databases has been studied, e.g., in [GPST94, Güt94, MS90]. These approaches deserve a more thorough study, especially if they would support "richer" querying of RDF than it is provided today. On the same lines, other graph-theoretic problems, like graph pattern mining are important for storage techniques, e.g., common statement patterns in Jena [WSKD03].

Conclusion

This chapter opened with a section on the challenges database technology faces in order to meet standard use cases for metadata processing of the near future. Although this short presentation revealed only issues which are to some extent obvious, today's storage and query systems satisfy them only partially. As a possible solution, a set of graph primitives has been proposed in order to contribute towards more expressive query languages.

11. Conclusion

The task of this thesis was to provide a *graph-based intermediate model* for RDF. The class of *RDF bipartite graphs* proposed here as a representation for RDF resides between the abstract triple model and a concrete serialization syntax (e.g., RDF/XML): concepts and results from graph theory may support reasoning on the RDF model, but applications are free to choose their own task-specific implementations of RDF bipartite graphs in order to take advantage of the graph foundation of RDF.

Chapter 4 argued why it is relevant to study explicit graph representations of the RDF (abstract) graph model. It also argued that it would be beneficial to find an alternate graph representation to the directed labeled graphs which are commonly used by default. Several reasons were given why RDF bipartite graphs have advantages compared to directed labeled graphs, some of which are recalled below. However, RDF bipartite graphs were introduced to *enhance* the understanding of RDF where directed labeled graphs do not suffice, and not to entirely replace them. For example, directed labeled graphs are frequently used for the visualization of RDF examples in specification documents or manuals (e.g., [KC04, FOR03]) and in drawing programs [Say04].

The principal difference between directed labeled graphs and RDF bipartite graphs is that the latter *incorporate properties and statements as nodes* into the graph. In a directed labeled graph, properties, although information resources in their own right, occur possibly several times as labels of statement-edges in contrast to subjects / objects which are represented by nodes. Also the representation of statements themselves as (unlabeled) nodes in RDF bipartite graphs has advantages: for example, the difference of two RDF Graphs is essentially a (partial) map of statements from one graph to the other, which could be very well represented by maps among the sets of statement nodes of RDF bipartite graphs.

The issue of properties being represented by nodes arises again when studying the connectivity of RDF Graphs. It was shown that directed labeled graphs convey only a restrained notion of connectedness in contrast to the

RDF model, while RDF bipartite graphs account for it fully. The term *vertical connectedness* grasps what directed labeled graphs miss and RDF bipartite graphs/ the RDF model have. Chapter 9 presented examples where this notion is of importance; especially advanced use cases involving schema relations rely on it.

To take advantage of this finding and to contribute towards an enhancement of RDF query languages *graph-based query primitives* were proposed in chapter 10. It turns out that several query use cases formulated in "graph language" account for common examples of queries, while others, although rather straightforward given typical applications of RDF, are not supported in current RDF query languages.

The discussion of RDF querying left open issues of implementation. On the one hand, there are already many contributions to graph-based querying and database storage, so an attempt to further investigate graph-based RDF querying should carefully consider this previous work. On the other hand, extending the expressiveness of RDF query languages would be rewarding for various applications. Thus, enhancing RDF query languages and the underlying storage system by means of graphs appears as a very promising subject for further research.

Bibliography

[Ado03] Adobe Systems Incorporated. *PDF Reference, Version 1.5*. World Wide Web, `http://partners.adobe.com/asn/tech/pdf/specifications.jsp`, 2003. 2

[Ado04] Adobe Developer Technologies. *XMP - Extensible Metadata Platform*. World Wide Web, `http://www.adobe.com/products/xmp/pdfs/xmpspec.pdf`, 2004. 2

[Adu04] Aduna B.V. *User Guide for Sesame*. World Wide Web, `http://www.openrdf.org/doc/users/userguide.html`, 2004. 10.2.3, 10.3.1

[AGH04] Renzo Angles, Claudio Gutierrez, and Jonathan Hayes. RDF Query Languages Need Support for Graph Properties. Technical Report TR/DCC-2004-3, Universidad de Chile, `http://www.dcc.uchile.cl/~cgutierr/ftp/graphproperties.pdf`, 2004. 4.3.1, 4.3.5, 10.3.1, 3

[Agr88] Rakesh Agrawal. Alpha: An Extension of Relational Algebra to Express a Class of Recursive Queries. *IEEE Trans. Softw. Eng.*, 14(7):879–885, 1988. 9.3.3

[All01] Joshua Allen. *Making a Semantic Web*. World Wide Web, `www.netcrucible.com/semantic.html`, 2001. 2

[AMHAS03] Boanerges Aleman-Meza, Chris Halaschek, Ismailcem Budak Arpinar, and Amit P. Sheth. Context-Aware Semantic Association Ranking. In I. F. Cruz, V. Kashyap, S. Decker, and R. Eckstein, editors, *Proceedings of SWDB'03*, 2003. 3.4, 4.3.4, 9.3.4

[AS92] Bernd Amann and Michel Scholl. Gram: A Graph Data Model and Query Language. In *ECHT '92: European Conference on Hypertext Technology, November 30 - December 4, 1992, Milan, Italy*, pages 201–211. ACM, 1992. 9.2

[AS03] Kemafor Anyanwu and Amit Sheth. ρ-Queries: Enabling Querying for Semantic Associations on the Semantic Web. In *Proceedings of the Twelfth International World Wide Web Conference*, pages 690–699. ACM Press, 2003. 9.2, 9.3.2, 9.3.4, 10.3, 10.3.2

[Bec04] Dave Beckett. *RDF/XML Syntax Specification. W3C Recommendation.* World Wide Web, `http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/`, 10 February 2004. 2, 3.1, 4, 4.3.1, A.1.1

[Ber87] Claude Berge. *Hypergraphes. Combinatoires des Ensembles Finis.* Gauthier-Villars, Paris, 1987. 3.3

[BFS00] Peter Buneman, Mary Fernandez, and Dan Suciu. UnQL: a Query Language and Algebra for Semistructured Data Based on Structural Recursion. *The VLDB Journal*, 9(1):76–110, 2000. 9.3.3

[BG04] Dan Brickley and R.V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation.* World Wide Web, `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`, 10 February 2004. 3.1, 4, 4.3.1, A.1.1

[BKD+00] J. Broekstra, M. Klein, S. Decker, D. Fensel, and I. Horrocks. Adding Formal Semantics to the Web: Building on Top of RDF Schema. In *Proceedings of ECDL 2000, Workshop on the Semantic Web*, `http://www.ics.forth.gr/isl/SemWeb/program.html`, 2000. World Wide Web. 4.5

[BKD+01] Jeen Broekstra, Michel C. A. Klein, Stefan Decker, Dieter Fensel, Frank van Harmelen, and Ian Horrocks. Enabling Knowledge Representation on the Web by Extending RDF Schema. In *Proceedings of the Tenth International World Wide Web Conference*, pages 467–478. ACM Press, 2001. 4.5

[BKvH02] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In I. Horrocks and J. A. Hendler, editors, *The Semantic Web - ISWC 2002, Sardinia, Italy, Proceedings*, volume 2342 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2002. 4.5, 10.2.3

[BL98] Tim Berners-Lee. *Semantic Web Road Map*. World Wide Web, `http://www.w3.org/DesignIssues/Semantic.html`, September 1998. 1, 2, 2

[BL01a] Tim Berners-Lee. *Conceptual Graphs and the Semantic Web*. World Wide Web, `http://www.w3.org/DesignIssues/CG.html`, 2001. 4.5

[BL01b] Tim Berners-Lee. *The RDF-DIFF Problem*. World Wide Web, `http://www.w3.org/DesignIssues/Diff`, 2001. Replaced by newer document [BLC04]. 4.3.3

[BL03a] Tim Berners-Lee. *Semantic Web Status and Direction. ISWC2003 Keynote*. World Wide Web, `http://www.w3.org/2003/Talks/1023-iswc-tbl/Overview.html`, 2003. A.1.4

[BL03b] Tim Berners-Lee. *Standards, Semantics and Survival. Talk at SIIA Meeting*. World Wide Web, `http://www.w3.org/2003/Talks/01-siia-tbl/slide1-0.html`, 2003. A.1.4

[BLC04] Tim Berners-Lee and Dan Connolly. *Delta: an Ontology for the Distribution of Differences between RDF Graphs*. World Wide Web, `http://www.w3.org/DesignIssues/Diff`, 2004. 4.3.3, 4.3.4, 11

[BLFM98] T. Berners-Lee, R. Fielding, and L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax (RFC 2396)*. World Wide Web, `http://www.ietf.org/rfc/rfc2396.txt`, 1998. 2

[BM01] Paul V. Biron and Ashok Malhotra. *XML Schema Part 2: Datatypes (W3C Recommendation)*. World Wide Web, `http://www.w3.org/TR/xmlschema-2/`, 2001. 2

[BPSM+04] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. *Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation*. World Wide Web, `http://www.w3.org/TR/2004/REC-xml-20040204`, 2004. 2

[Bra03a] Tim Bray. *Ongoing: DC Intel*. World Wide Web, `http://www.tbray.org/ongoing/When/200x/2003/09/06/ICMWG`, 2003. 2, 10.1

[Bra03b] Tim Bray. *Ongoing: On Search: Metadata*. World Wide Web, `http://www.tbray.org/ongoing/When/200x/2003/07/29/SearchMeta`, 2003. 2

[Bra03c] Tim Bray. *Ongoing: The RDF.net Challenge*. World Wide Web, `http://www.tbray.org/ongoing/When/200x/2003/05/21/RDFNet`, 2003. A.1.4

[Car01] Jeremy J. Carroll. Matching RDF Graphs. Technical Report HPL-2001-293, Digital Media Systems Laboratory, HP Laboratories, Bristol, `http://www.hpl.hp.com/techreports/2001/HPL-2001-293.html`, 2001. 1, 4.3.3, 4.3.4, 4.5, 7.2

[Car03] Jeremy J. Carroll. Signing RDF Graphs. Technical Report HPL-2003-142, Digital Media Systems Laboratory, HP Laboratories, Bristol, 2003. 4.5

[CDD+03] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: Implementing the Semantic Web Recommendations. Technical Report HPL-2003-146, Digital Media Systems Laboratory, HP Laboratories, Bristol, 2003. 10.2.2

[CDH00] Olivier Corby, Rose Dieng, and Cedric Hebert. A Conceptual Graph Model for W3C Resource Description Framework. In Bernhard Ganter and Guy W. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues, 8th International Conference on Conceptual Structures, ICCS 2000*, volume 1867 of *Lecture Notes in Computer Science*, pages 468–482. Springer, 2000. 4.5

[Cha98] Walter W. Chang. *A Discussion of the Relationship Between RDF-Schema and UML (W3C Note)*. World Wide Web, `http://www.w3.org/TR/NOTE-rdf-uml/`, 1998. 4.5

[Cla04] Kendall Grant Clark. *Putting ISBNs to Work*. World Wide Web, `http://www.xml.com/pub/a/2004/06/02/dijalog.html`, 2004. O' Reilly XML.com, June 02. A.2

[CM90] Mariano P. Consens and Alberto O. Mendelzon. GraphLog: a Visual Formalism for Real Life Recursion. In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, April 2-4, 1990, Nashville, Tennessee*, pages 404–416. ACM Press, 1990. 9.3.3

[Con03] Dan Connolly. *Naming and Addressing: URIs, URLs, ...*. World Wide Web, `http://www.w3.org/Addressing/`, 2003. 2

[CS04] Jeremy J. Carroll and Patrick Stickler. TriX: RDF Triples in XML. Technical Report HPL-2004-56, Digital Media Systems Laboratory, HP Laboratories, Bristol, `http://www.hpl.hp.com/techreports/2004/HPL-2004-56`, 2004. 3.1, 10.2.1

[Die97] Reinhard Diestel. *Graph Theory*. Springer-Verlag, New York, 1997. 3.2

[DMvH+00] Stefan Decker, Sergey Melnik, Frank van Harmelen, Dieter Fensel, Michel C. A. Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. The Semantic Web: The Roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–74, 2000. A.1.4

[Duc95] Pierre Duchet. Hypergraphs. In R.L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, pages 381–432. Elsevier Science B.V., Amsterdam, 1995. 3.3

[DvHB+00] Stefan Decker, Frank van Harmelen, Jeen Broekstra, Michael Erdmann, Dieter Fensel, Ian Horrocks, Michel Klein, and Sergey Melnik. *The Semantic Web – On The Respective Roles of XML and RDF*. World Wide Web, `http://www.ontoknowledge.org/oil/downl/IEEE00.pdf`, 2000. Published in *IEEE Internet Computing*, 4(5):63–74, 2000. 2

[Ebe02] Andreas Eberhart. Survey of RDF Data on the Web. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI2002)*, 2002. A.4

[FJJ03] Jon Ferraiolo, Fujisawa Jun, and Dean Jackson. *Scalable Vector Graphics (SVG) 1.1 Specification (W3C Recommendation)*. World Wide Web, http://www.w3.org/TR/SVG11/, 2003. 4.3.5

[FOR03] FORTH Institute of Computer Science, `http://139.91.183.30:9090/RDF/RQL/Manual.html`. *RQL v2.1 User Manual*, 2003. 3.1, 3.4, 1, 4.3.5, 9.3.2, 10.3.1, 11

[Gar03] Lars Marius Garshol. *Living with Topic Maps and RDF*. World Wide Web, `http://www.ontopia.net/topicmaps/materials/tmrdf.html`, 2003. 4.5

[GB04] Jan Grant and Dave Beckett. *RDF Test Cases*. World Wide Web, `http://www.w3.org/TR/2004/`

REC-rdf-testcases-20040210/, 10 February 2004. 4, 4.3.1, A.1.1

[GCG99] R.V. Guha, Robert Churchill, and John Giannandrea. *RDF Technical Overview*. World Wide Web, http://www.mozilla.org/rdf/doc/api.html, 1999. 1

[GHM04] Claudio Gutierrez, Carlos Hurtado, and Alberto O. Mendelzon. Foundations of Semantic Web Databases. In *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), June 14-16, 2004, Paris, France*. ACM, 2004. 4.3.3, 4.3.4, 4.5, III

[GLMB98] R.V. Guha, Ora Lassila, Eric Miller, and Dan Brickley. *Enabling Inferencing*. World Wide Web, http://www.w3.org/TandS/QL/QL98/pp/enabling.html, 1998. 9, 10.3, 10.3.1, A.1.4

[GMM03] R.V. Guha, Rob McCool, and Eric Miller. Semantic Search. In *Proceedings of the Twelfth International World Wide Web Conference*, pages 700–709. ACM Press, 2003. 2

[GPST94] Alejandro Gutiérrez, Philippe Pucheral, Hermann Steffen, and Jean-Marc Thévenin. Database Graph Views: A Practical Model to Manage Persistent Graphs. In J.B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of VLDB'94*, pages 391–402. Morgan Kaufmann, 1994. 10.3.2

[Güt94] Ralf Hartmut Güting. GraphDB: Modeling and Querying Graphs in Databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 297–308. Morgan Kaufmann, 1994. 9.2, 10.3.2

[Gut02] Claudio Gutierrez. Introducción a la Web Semántica. World Wide Web, http://www.dcc.uchile.cl/~cgutierr/websemantica/eci/apuntes.pdf, 2002. ECI 2002. 2

[HAMA+04] Chris Halaschek, Boanerges Aleman-Meza, Ismailcem Budak Arpinar, C. Ramakrishnan, and Amit P. Sheth. A Flexible Approach for Analyzing and Ranking Complex Relationships on the Semantic Web. In *The Semantic Web - ISWC 2004, Hiroshima, Japan, Proceedings*, Lecture Notes in Computer Science. Springer, 2004. (Submitted). 9.2

[HAMAS04] Chris Halaschek, Boanerges Aleman-Meza, Ismailcem Budak Arpinar, and Amit P. Sheth. Discovering and Ranking Semantic Associations over a Large RDF Metabase (Demonstration Paper). In Mario Nascimento, editor, *VLDB 2004, Proceedings of 30th International Conference on Very Large Data Bases*. Morgan Kaufman, 2004. 9.3.4

[Hay04] Patrick Hayes. *RDF Semantics. W3C Recommendation.* World Wide Web, `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`, 10 February 2004. 3.1, 4, 4.3.1, 4.3.3, 4.3.4, 4.5, 7, 7.1, 18, 7.2, 7.4, 8.1, A.1.1

[HBEV04] Peter Haase, Jeen Broekstra, Andreas Eberhart, and Raphael Volz. *A Comparison of RDF Query Languages.* World Wide Web, `http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/rdfquery.pdf`, 2004. 10.3.1, 10.3.2

[HG04] James Howison and Abby Goodrum. *Why can't I manage academic papers like MP3s? The evolution and intent of Metadata standards.* World Wide Web, `http://freelancepropaganda.com/archives/MP3vPDF.pdf`, 2004. 2

[Jai04] Eric Jain. *Reification—What's Best Practice?* World Wide Web, RDF Interest Mailing List, `http://lists.w3.org/Archives/Public/www-rdf-interest/2004Aug/0172.html`, August 2004. 3

[KAC+02] Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. RQL: A Declarative Query Language for RDF. In *Proceedings of 2002 WWW Conference*, pages 592–603. ACM Press, 2002. 3.1, 4.5, 8.2, 9.3.4

[KC04] Graham Klyne and Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation.* World Wide Web, `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`, 10 February 2004. 1, 2, 3.1, 2, 3.1, 4, 4.1, 4.1, 4.3.1, 4.5, 5.1, 11, A.1.1

[KCP00] Greg Karvounarakis, Vassilis Christophides, and Dimitris Plexousakis. Querying Semistructured (Meta)Data and Schemas on the Web: The case of RDF and RDFS. Technical Report 269, FORTH Institute of Computer Science, 2000. 4.5

[Kle99] Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632, 1999. 4.3.4

[KM02] Marja-Riitta Koivunen and Eric Miller. W3C Semantic Web Activity. In E. Hyvonen, editor, *Semantic Web Kick-Off in Finland - Vision, Technologies, Research, and Applications*, `http://www.cs.helsinki.fi/u/eahyvone/stes/semanticweb/kick-off/proceedings.html`, 2002. Helsinki Institute for Information Technology (HIIT). 2.1, 2.3

[KMA+04] Gregory Karvounarakis, Aimilia Magkanaraki, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, Michel Scholl, and Karsten Tolle. RQL: A Functional Query Language for RDF. In P.M.D. Gray, L. Kerschberg, P.J.H. King, and A. Poulovassilis, editors, *The Functional Approach to Data Management*. Springer-Verlag, 2004. 4.5, III, 8.1, 10.3.1

[KS02] Stefan Kokkelink and Roland Schwänzl. *Expressing Qualified Dublin Core in RDF / XML*. World Wide Web, `http://dublincore.org/documents/dcq-rdf-xml/`, 2002. 3.1

[Kuc90] Ludek Kucera. *Combinatorial Algorithms*. IOP Publishing Ltd, Bristol, 1990. 3.2

[LD01] Martin S. Lacher and Stefan Decker. On the Integration of Topic Maps and RDF Data. In Isabel F. Cruz, Stefan Decker, Jérôme Euzenat, and Deborah L. McGuinness, editors, *Proceedings of SWWS'01, The First Semantic Web Working Symposium, Stanford University, California, USA, July 30 - August 1, 2001*, pages 331–344, 2001. 4.5

[LS99] Ora Lassila and Ralph R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation*. World Wide Web, `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222`,, 1999. 4, 1, 4.3.1

[MAYU03] Akiyoshi Matono, Toshiyuki Amagasa, Masatoshi Yoshikawa, and Shunsuke Uemura. An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays. In I. F. Cruz, V. Kashyap, S. Decker, and R. Eckstein, editors, *Proceedings of SWDB'03*, pages 151–168, 2003. 8.2

[Mil] George A. Miller. WordNet—A Lexical Database for the English Language. World Wide Web, `http://www.cogsci.princeton.edu/~wn/`. 8.1

[Mil98] Eric Miller. An Introduction to the Resource Description Framework. *D-lib Magazine*, Mai, 1998. 3.1, A.1.3

[MM04] Frank Manola and Eric Miller. *RDF Primer. W3C Recommendation.* World Wide Web, `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`, 10 February 2004. 1, 4, 2, 4.3, 4.3.1, 4.3.2, A.1.1, A.1.3, A.3

[MS90] Michael V. Mannino and Leonard D. Shapiro. Extensions to Query Languages for Graph Traversal Problems. *IEEE Trans. Knowl. Data Eng.*, 2(3):353–363, 1990. 10.3.2

[MSB04] Eric Miller, Ralph Swick, and Dan Brickley. *Resource Description Framework (RDF) / W3C Semantic Web Activity.* World Wide Web, `http://www.w3.org/RDF/`, 2004. 1

[MvH04] Deborah L. McGuinness and Frank van Harmelen. *OWL Web Ontology Language Overview. W3C Recommendation.* World Wide Web, `http://www.w3.org/TR/owl-features/`, 10 February 2004. 2

[MW89] Alberto O. Mendelzon and Peter T. Wood. Finding Regular Simple Paths in Graph Databases. In Peter M. G. Apers and Gio Wiederhold, editors, *Proceedings of the Fifteenth International Conference on Very Large Data Bases, August 22-25, 1989, Amsterdam, The Netherlands*, pages 185–193. Morgan Kaufmann, 1989. 9.2, 10.3.2

[Ogi01] Nikita Ogievetsky. *XML Topic Maps through RDF Glasses.* World Wide Web, `http://www.cogx.com/xtm2rdf/extreme2001`, 2001. Slides presented at *Extreme Markup Languages 2001*, 12-17 August 2001, Montréal, Canada. 4.5

[Olk03] Frank Olken. Tutorial on Graph Data Management for Biology. *IEEE Computer Society Bioinformatics Conference (CSB)*, Aug 2003. 3.4

[Pal02] Sean B. Palmer. *RDF in HTML: Approaches.* World Wide Web, `http://infomesh.net/2002/rdfinhtml/`, 2002. 2, 4.3.5

[PG01] E. Prud'hommeaux and B. Grosof. *RDF Query and Rules: A Framework and Survey*. World Wide Web, `http://www.w3.org/2001/11/13-RDF-Query-Rules/`, 2001. 10.3.1

[PH01] Jeff Pan and Ian Horrocks. Metamodeling Architecture of Web Ontology Languages. In Isabel F. Cruz, Stefan Decker, Jérôme Euzenat, and Deborah L. McGuinness, editors, *Proceedings of SWWS'01, The first Semantic Web Working Symposium, Stanford University, California, USA, July 30 - August 1, 2001*, LNCS. Springer-Verlag, 2001. 8.1, 8.3

[PH03] Jeff Pan and Ian Horrocks. RDFS(FA) and RDF MT: Two Semantics for RDFS. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 30–46. Springer, 2003. 8.1, 8.3

[RE03] A. Rodríguez and M. Egenhofer. Determining Semantic Similarity Among Entity Classes from Different Ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):442–456, 2003. 9.3.4

[SAMA+04] A. Sheth, B. Aleman-Meza, I. B. Arpinar, C. Halaschek, C. Ramakrishnan, C. Bertram, Y. Warke, D. Avant, F. S. Arpinar, K. Anyanwu, and K. Kochut. Semantic Association Identification and Knowledge Discovery for National Security Applications. *Special Issue of Journal of Database Management on Database Technology for Enhancing National Security*, 2004. 9.3.4, 10.1

[Say04] Craig Sayers. Node-Centric RDF Graph Visualization. Technical Report HPL-2004-60, Enterprise Systems and Data Management Laboratory, HP Laboratories, Palo Alto, `http://www.hpl.hp.com/techreports/2004/HPL-2004-60.html`, 2004. 11

[Sea04] Andy Seaborne. *RDQL - A query Language for RDF (W3C Member Submission)*. World Wide Web, `http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/`, January 2004. 10.3.1

[Sem] Semaview. XML and RDF Illustrated. World Wide Web, `http://www.semaview.com`. A.1.3

[Sem04] Semagix. *Semagix Freedom: Creating a Three Dimensional View of Your Information*. World Wide Web, `www.semagix.com`, 2004. 10.1

[SEMD00] S. Staab, M. Erdmann, A. Maedche, and S. Decker. An Extensible Approach for Modeling Ontologies in RDF(S). In *Proceedings of ECDL 2000, Workshop on the Semantic Web*, `http://www.ics.forth.gr/isl/SemWeb/program.html`, 2000. World Wide Web. 4.5

[Stu04] Heiner Stuckenschmidt. ISWC'04 Tutorial: Theory and Practice of RDF Query Processing. World Wide Web, `http://www.cs.vu.nl/~heiner/ISWC04Tutorial.html`, 2004. 2

[TBMM01] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. *XML Schema Part 1: Structures*. World Wide Web, `http://www.w3.org/TR/xmlschema-1`, 2001. 2

[Tuc04] Tucana Technologies, Inc. *Information Fusion... Tucana*. World Wide Web, `www.tucanatech.com`, 2004. Whitepaper. 10.1

[W3C04] W3C World Wide Web Consortium. *HyperText Markup Language (HTML) Home Page*. World Wide Web, `http://www.w3.org/MarkUp/`, 2004. 2

[Wei] Eric W. Weisstein. Isomorphic graphs. From MathWorld—A Wolfram Web Resource. `http://mathworld.wolfram.com/IsomorphicGraphs.html`. 2

[Wik04a] Wikipedia: The Free Encyclopedia. Seven bridges of königsberg. `http://en.wikipedia.org`, 18 Jul 2004. 1.1

[Wik04b] Wikipedia: The Free Encyclopedia. Graph (mathematics). `http://en.wikipedia.org`, 25 Jun 2004. 1

[WSKD03] Kevin Wilkinson, Craig Sayers, Harumi Kuno, and Luping Ding. Application-Specific Schema Design for Storing Large RDF Datasets. Technical Report HPL-2003-170, Mobile and Media Systems Laboratory, HP Laboratories, Palo Alto, `http://www.hpl.hp.com/techreports/2003/HPL-2003-170.html`, 2003. 4.3.4, 10.3.2

[WSKR03] Kevin Wilkinson, Craig Sayers, Harumi Kuno, and Dave Reynolds. Efficient RDF Storage and Retrieval in Jena2. In I. F. Cruz, V. Kashyap, S. Decker, and R. Eckstein, editors, *Proceedings of SWDB'03*, 2003. 4.5, 10.1, 10.2.2, 10.3

[YK02] G. Yang and M. Kifer. On the Semantics of Anonymous Identity and Reification. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE, Irvine, USA, Proceedings*, volume 2519 of *Lecture Notes in Computer Science*. Springer, 2002. 4.5

Appendix

A. RDF Resources

A.1 RDF in General

A.1.1 Specification Documents

RDF Specification Documents (RDF Primer [MM04], RDF Semantics [Hay04], RDF Concepts and Abstract Syntax [KC04], RDF Test Cases [GB04], RDF Vocabulary Description Language [BG04], RDF/XML [Bec04])

A.1.2 RDF Portals

- RDF / W3C Semantic Web Activity
  `http://www.w3.org/RDF/`
  RDF portal of the WWW Consortium. Links to the RDF specification documents and other resources

- Dave Beckett's Resource Description Framework (RDF) Resource Guide
  `http://www.ilrt.bris.ac.uk/discovery/rdf/resources/`

A.1.3 RDF Tutorials

- RDF Primer
  `http://www.w3.org/TR/rdf-primer/`
  The official RDF Primer by the WWW Consortium ([MM04])

- RDF Introduction by Eric Miller
  `http://www.dlib.org/dlib/may98/miller/05miller.html`
  Very good, but quite old (May 1998) ([Mil98])

- RDF Introduction by Champin
  `http://www710.univ-lyon1.fr/%7Echampin/rdf-tutorial/`

- RDF: in Fifty Words or Less (Mozilla)
  `http://www.mozilla.org/rdf/50-words.html`

- RDF and XML Illustrated [Sem] is a nice collection of explanation fragments for RDF and RDF/XML

A.1.4 Miscellaneous

- Some notes on the history of RDF by Tim Bray [Bra03c]

- Recent (2003) talks of Tim Berners-Lee: [BL03b], [BL03a]

- A clarifying examination of the roles of RDF and XML in the construction of a Semantic Web: [DMvH+00]

- "Enabling Inferencing" from 1998 [GLMB98]

- Here are links to the archives of the W3C RDF mailing lists:
  `http://lists.w3.org/Archives/Public/www-rdf-interest/`
  `http://lists.w3.org/Archives/Public/w3c-rdfcore-wg/`
  `http://lists.w3.org/Archives/Public/www-rdf-logic/`

- Tim Berners-Lee, *The RDF-Diff Problem*
  `http://www.w3.org/DesignIssues/Diff`

- Jeremy Carroll, *Matching RDF Graphs*
  `http://www.hpl.hp.com/techreports/2001/HPL-2001-293.html`

A.2 Developing with RDF

- *Putting RDF to Work* by Edd Dumbill, August 2000 on xml.com
  `http://www.xml.com/pub/a/2000/08/09/rdfdb/index.html`

- *Putting ISBNs to Work* by Kendall Grant Clark on xml.com [Cla04]

A.2.1 Libraries

- RDFlib
  `http://rdflib.net/`
  A Python library for RDF. See also `http://redfoot.net/`

- Jena Semantic Web Framework
  `http://www.hpl.hp.com/semweb/jena.htm`
  Hewlett Packard's Java RDF programming library and storage architecture

- Sesame
  `http://www.openrdf.org:80/`
  An open source RDF Schema-based storage and querying facility (Java)

- Lisp and RDF
  `http://www.semanticweb.org/SWWS/program/full/paper32.pdf`
  A paper by Ora Lassila on programming with RDF in Lisp

- RDFStore - Perl API for RDF Storage
  `http://rdfstore.sourceforge.net/`

- kowari : metastore
  `http://kowari.sourceforge.net/docs/index.php`

- Redland RDF Application Framework
  `http://www.redland.opensource.ac.uk/`
  A C-library for RDF

A.3 Projects Using RDF as Data Format

Some links are given also in section 6 of the RDF Primer [MM04]

- The Open Directory Project
  `http://dmoz.org/`
  The Open Directory Project is the largest, most comprehensive human-edited directory of the Web

- RDF at Mozilla
  `http://www.mozilla.org/rdf/doc/`
  Usage of RDF in the Mozilla web browser and mail tool

- The friend of a friend (foaf) project
  `http://www.foaf-project.org/`

- Haystack Home
  `http://haystack.lcs.mit.edu/index.html`

- Dublin Core Metadata Initiative
  `http://dublincore.org/documents/dcmi-terms/`
  Metadata terms maintained by the Dublin Core Metadata Initiative

- Expressing Simple Dublin Core in RDF/XML
  `http://dublincore.org/documents/dcmes-xml/`

- RDF Site Summary (RSS) 1.0
  `http://web.resource.org/rss/1.0/`
  RSS is an RDF/XML format summarizing website content

- Syndic8.com
  `http://www.syndic8.com/`
  A site collecting RSS channels

- Vcard RDF/XML format
  `http://www.w3.org/TR/2001/NOTE-vcard-rdf-20010222/`

- Creative Commons
  `http://creativecommons.org/`
  A website promoting the open-source idea for artist's creative works. The focus is especially on licensing and metadata markup

- Describing and retrieving photos using RDF and HTTP
  `http://www.w3.org/TR/photo-rdf/`
  This page describes how RDF metadata can be stored in a JPG image

A.4 RDF Data on the Web

Consider the 2002 survey [Ebe02]. It does not, however, consider the validity of the RDF/XML used, an issue I experienced repeatedly.

- the *OpenDirectory Project* (`http://www.dmoz.org`) uses RDF, but in an awkward manner which is not valid RDF/XML. The *ODP Data Parser* (`http://www.ohardt.com/computer/dev/java/`) is a custom-made parser.

- The *Chefmoz project* (`http://chefmoz.org`)is related to the OpenDirectory project, but I am not sure if it is valid RDF/XML, either.

- The RDF representation of *WordNet* (`http://www.cogsci.princeton.edu/~wn/`) can be found at `http://www.semanticweb.org/library/`. The files are valid RDF/XML. A Schema definition is provided. It contains about 470 000 statements. Sesame benchmarks can be found in the Openrdf forum (`http://www.openrdf.org/forum/mvnforum/viewthread?thread=38`), including a tip to add two statements to the schema to ensure correct inferencing.

- The *Gene Ontology Database* (`http://www.godatabase.org/dev/database/`) is a large gene concept database. It cannot be validated by Sesame because they use a non-RDF attribute in their RDF/XML. It can be, however, parsed by Jena (with warnings). My conversion attempts failed due to its excessive size.

- The *Dynamic Research Cooperation Ontologies* (`http://orlando.drc.com/DAML/Ontology/Ontologies.htm`) contain together 7 186 statements (1324 distinct subjects, 59 predicates and 3367 objects. There are 1766 resources and 1970 literals.) which are valid RDF/XML.

- *Openguides* (`http://openguides.org`) is a network of free, community-maintained city guides based on the Wiki system. The technical policy is to provide the maximum possible amount of metadata about things. An RDF/XML summary is provided for each node. At the time of writing the entire RDF of a guide can not be downloaded in one bit, but there is an RDF index for each guide (e.g., for the London guide the index is at `http://london.openguides.org/index.cgi?action=index;format=rdf`), so the data can be retrieved using a simple crawler.

- The *Museum* example of the makers of RQL is downloadable from the Sesame system (`http://www.openrdf.org`) and contains 166 statements.

- The VCard example is also provided by Sesame and contains 177 statements (with inferred statements).