**语义网语义学**

Semantic web semantics

(M2R SWXO lecture notes)

Jerome Euzenat

INRIA & Univ. Grenoble Alpes,

Laboratoire d’informatique de Grenoble,

Montbonnot, France

Jerome.Euzenat@inria.fr

April 28, 2017

Copyrightc Jerome Euzenat, 2007-2017

This document can be distributed as such or used in part with attribution.

1.引言

Web经历从分布式超文本系统到一个非常大型的信息处理机器一直在演变。尽管如此，这一演变是基于借用计算机科学领域的几个原则的理论原理，如程序设计语言、数据库、结构化文档、逻辑和人工智能。过去和未来网络的大规模平稳运作都依赖于这些基础。

本课程讲义介绍了Web上知识表示的语义（语义Web技术）。此外，本课程讲义的目的是让学生了解知识表示技术，以便在其他情况下进行使用。特别是，我们的目标是提供：

* 对这些技术的深刻理解，至少比简单的语言方法更深；
* 逻辑的一个具体应用例子，逻辑本身。

语义Web以更丰富和更精确的信息扩展Web，因为它是使用本体中定义的词汇（以一种逻辑形式定义的概念和属性的结构化词汇）以一种形式化语言表达的。本体（ontology）用于描述Web资源内容，并对这些资源进行形式化推理。我们介绍了语义Web语言（RDF，RDFS，OWL），用知识表示的形式化（概念图，描述逻辑）和XML揭示它们的关系。这种方法为用本体进行推理特别是评估查询提供了工具。然而，Web的分布式本质导致在使用之前必须进行匹配的异构本体（heterogeneous ontologies）。我们揭示了如何匹配本体以及如何在语义上解释本体之间的关系。最后，将这种方法应用到对等点的网络中。

该材料是为了尽可能通用，但它是通过具体和实用的语义Web技术（主要是语言）说明。

* 1. **语义Web的历史注释**

随着万维网的发展，在万维网上使用知识表示的思想迅速出现。像SHOE [LUKE et al. 1997] 和Ontobroker [FENSEL, DECKER, et al. 1998]这样的系统，在网页中集成了形式化知识表示，与此同时Ontoserver [FARQUHAR et al. 1995]和HyTroeps [EUZENAT 1996]用Web浏览和编辑知识。

1998年，Tim Berners Lee描述了他称之为“语义网（Semantic Web）”的原则。他已经在W3C1的RDF语言推出，其最初的目标是描述Web页面的开发。当时，他写道“一个语义Web不是人工智能（A Semantic Web is not Artificial Intelligence）”，但是后来他补充说，这是“知识表示走向全球（Knowledge Representation goes Global）”[BERNERS-LEE 1998]。

2000年，这一想法并没有取得成功，也没有受到雄心勃勃的发展的影响。但美国DARPA推出DAML（*DARPA Agent Markup Language*）计划，由SHOE的发起人James Hendler执导。此后，举办了一场“*Semantics for the Web*”研讨会 [FENSEL, HENDLER, et al. 2003]，指导欧盟OntoWeb专题网络，本身导致了卓越的Knowledge Web网络。在语义Web技术的发展方面[EUZENAT (ED.) 2002]的许多人工智能研究者都已经参与了这些常识。

语义（Semantic Web）网通常被称为“机器的网络（Web for Machines）”：它的目标是用可由机器处理的形式化语言来表示在Web上的知识。为此目的，使用人工智能开发的技术是很自然的。

事实上，语义Web的主要目标应用是信息检索（Information Retrieval）：能够在Web上检索精确的信息，因为文档是用“语义元数据（Semantic Metadata）”标注的。尽管这一目标在2000年看来很遥远，但现在却更为实际可触知的。在这期间，我们目睹了Facebook网开放图Open Graph的使用，通过四个主要的搜索引擎（Bing, Google, Yahoo, Yandex）定义的轻量级本体schema.org，以及Google的知识图（Knowledge Graph）的应用，提供了对查询结构化的应答而不是文件列表清单。

已经出现了许多其他类型的应用程序，每个应用程序都基于相同的基础：使用相同的知识表示语言对资源进行注释。这些应用程序被称为语义技术的应用程序。尤其是：

* 语义Web服务（Semantic Web Services）：其中，服务Web接口（输入/输出）用语义方式进行标注；
* 语义对等系统（Semantic Peer-to-Peer Systems）：其中，共享资源用语义方式进行标注；
* 语义社会网络（Semantic Social Networks）：其中，社会关系和人物简介（people profiles）用语义方式进行标注；
* 语义桌面（Semantic Desktop）：其中，个人信息（议程、地址簿等）用语义方式进行表示；
* 语义物联网（Semantic Web of Things）：其中，传感器、效应器以及它们交换的信息用语义方式进行标注；
* 数据网络（Web of Data）：其中，以语义Web语言表示和链接各种数据源。

语义网已经从最初的园地扩展开来。现在，它是研究各种问题的一个广泛的实验领域，如对等系统的信任，统计数据集成，或智能城市监控。作为Web技术，语义技术逐步影响到所有的计算机的发展[JANOWICZ et al. 2015]。

**1.2课堂讲稿格式和主题依赖**

这个文档由一个硕士课程的课堂讲稿构成（见x1.3），继我以前的关于知识表示语义的讲座[EUZENAT 2002] (法国)之后，已经在2007年到2017之间，以不同格式完整地或部分地在Grenoble大学讲授过。

整个课程是由7个小节（每小节3小时）组成的。

3h00表达信息（RDF）

6h00建模知识（RDFS和OWL）

3h00（SPARQL查询网络数据）

3h00 SPARQL的扩展

3h00本体对齐

3h00网络本体

它进一步分为三个部分：表达知识（数据和本体），查询和本体网络。

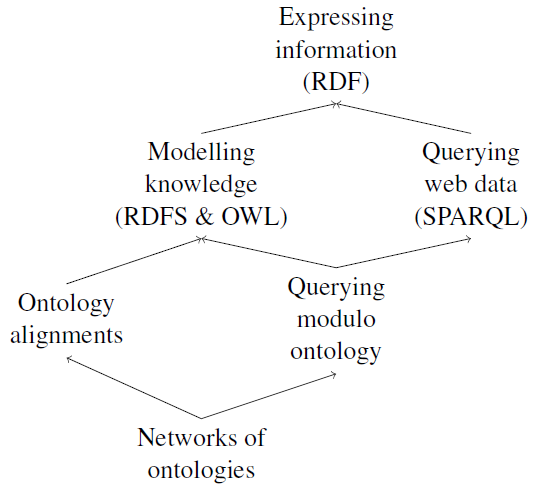


图1.1：这些小节的不同主题之间的依赖关系。

主要部分之间的依赖关系如图1.1所示。不过，也可以只考虑第2, 3, 4、第5, 8、第8.2、第10和第12.1章，以减少的形式给予或阅读本课程。

本课程的演化是一个对抗势力间竞争、简化的不完善的结果，例如，大多忽略了数据类型和复合化，例如，添加OWL2和SPARQL1.1的特征。

**1.3 语义网课程中的语境**

本节致力于提供一个与Pierre Geneves与Nabil Layaıda 讲授的《XML基础》课程部分的连接。

他们考虑了在XML语言中表达信息。XML不是一种表达能力很强的语言：它提供了一种存储信息的结构，为了便于访问和利用，必须提取这些信息。它不允许用描述来表达知识，而这些描述将被用来指导这种解释。

XML提供了一个关于知识表示的数据库观点，其中可以用语言完成的能力取决于查询语言的能力（XPath、XQuery等），而表示是一个存储。

在语义Web的语境中，该语言提供了一种语义，指定如何解释结构。这允许定义一般的语句，不管语法形式如何，它都适用于相关数据。

这样做需要定义一个逻辑，指定语句（或公式）的语义，并确定文档或一组文档的可能模型。回答查询将不得不通过推理来进行，这取决于它们对语义的正确性和完整性。

无论如何，这些对Web数据的处理之间存在连续性。它首先通过导航从文档中提取数据：

信息也可以从文档模式中提取出来。尽管模式首先指定可接受的文档结构，但它们通常提供快捷方式，可以帮助生成更多的查询应答保证：

此外，复杂的查询语言提供了通过它们的代数结构和查询以更好的方式提取信息的机会来推理查询的可能性：

这里所描述的语义Web语言大致以与XML中相同的方式操作数据，因此：

但是，我们使用来强调提取信息的操作是在语义上定义的。同样的SPARQL查询语言XQuery查询，被查询RDF数据定义：

虽然SPARQL中推荐在语法定义，但是最好是在语义上定义SPARQL：

并展示了RDF语义的语法描述的完备性。此外，语义Web技术允许表达本体论，这些理论限制了数据必须如何解释。RDF和RDFS扩展（RDF Schema）和OWL形成W3C推荐的语义Web知识表示的三种形式逻辑。在本演示中，将要考虑的唯一操作是测试某些本体的数据是否包含特定的公式：

我们将集中于蕴涵试验并在后续考虑进一步阐述查询语言。最后，语义Web继承了Web的分布式方面。这意味着，独立的数据和本体资源提供了必须一起解释的信息。为此，可以考虑一组本体和数据源（依旧以OWL和RDF形式描述）和表示本体中的实体间关系的比对Λ的一个集合Ω。因此，从语义Web中提取信息，是在这样一个本体网络中获得的，通过：

随着XML增益表达（使用XML模式和XQuery）和RDF允许最小推理，“语法网（Syntactic Web）”和语义网（Semantic Web）之间的边界逐渐消失。XML查询语言的定义，如XPath，在逻辑上，如λ演算，需要对两个世界取得一个更好的理解。这就是为什么把这些讲座放在一起的主要原因之一。语义Web应用开发的，像在许多其他情况下，特点是对语言运用与复杂推理能力之间的权衡。

因此，在网络上表达知识和信息取决于不同的语言：

* 数据语言（XML，RDF）；
* 本体语言（XML模式，RDF模式，OWL）
* 查询语言（XPath，XQuery，SPARQL）
* 分布式方面（对齐Alignments）

每一个都需要在语义上进行定义。

**1.4习题**

我们提供的运动通常是以以前的考试为基础的。它们已在章节结尾处分发，以便在本章一章结束时予以考虑。考试（90分钟）分别由：

– 习题 1, 8, 18 (并附加28);

– 习题2, 9, 10, 20, 和 22;

– 习题3, 11, 21, 23, 和 27.

– 习题 12, 13, 和 24.

– 习题 4, 14, 和 25.

– 习题 5, 15, 和 26.

以及 60分钟的习题，构成：

– 习题 6 和 16

– 习题 7, 17 和 19

**1.5进一步的信息**

本课程网站http://exmo.inria.fr/teaching/sw/。它包含对其他材料的参考以及课程第一部分的信息。

任何评论、问题或澄清询问都可以通过电子邮件发送给作者：

mailto:Jerome.Euzenat@inria.fr

进一步的图书涵盖这些主题部分[ANTONIOU and VAN HARMELEN 2008; HITZLER et al. 2009]。

Contents

**1 Introduction 3**

1.1 Historical notes on the semantic web . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.2 Lecture note format and topic dependencies . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1.3 Context in the semantic web course . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

1.4 Exercises . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

1.5 Further information . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

1.6 Acknowledgements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

**I Graphs and ontologies 13**

**2 RDF: Resource description framework 17**

2.1 IRI (and XML): the basic layer . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

2.2 RDF Syntax . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

2.3 Simple RDF Semantics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

2.4 Inference mechanism . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

2.5 Computational properties . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

2.6 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

2.7 Exercises . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26

**Interlude 1: The web of data 31**

**3 Ontology languages 33**

3.1 RDF Schema . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

3.2 The web ontology language OWL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41

3.3 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

3.4 Exercises . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

**Interlude 2: Rules 59**

**II Queries 61**

**4 Querying the semantic web 63**

4.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63

4.2 What is a query language? . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63

**5 Querying RDF with SPARQL 65**

5.1 Syntax . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65

5.2 SPARQL Semantics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68

5.3 Algebraic manipulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71

5.4 Entailment regimes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72

5.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73

5.6 Exercises . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73

**6 Extending SPARQL 75**

6.1 NSPARQL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75

6.2 PSPARQL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78

6.3 cpSPARQL and CPSPARQL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80

**Interlude 3: Streams and navigation 87**

**7 Querying modulo ontologies 89**

7.1 RDF(S) closure and query answering . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 90

7.2 Ontologies expansion as path navigation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91

7.3 Querying modulo DL-lite . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96

7.4 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96

7.5 Exercises . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96

**III Networks of ontologies 99**

**8 Alignments and networks of ontologies 101**

8.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 101

8.2 Alignment syntax . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 103

8.3 Networks of ontologies . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 108

8.4 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 111

**9 Alignment algebra 113**

9.1 Algebras of relations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114

9.2 Aggregating matcher results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116

9.3 Composing alignments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 117

9.4 Algebraic reasoning with alignments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 118

9.5 Algebra granularity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 119

9.6 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 120

**Interlude 4: Ontology matching 121**

**10 Alignment semantics 123**

10.1 Reminder: ontology semantics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123

10.2 General alignment semantics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 124

10.3 Consistency, consequence and closure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 127

10.4 Local models from the standpoint of an ontology . . . . . . . . . . . . . . . . . . . . . . . 130

10.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131

10.6 Exercises . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132

10 J´E RˆO ME EUZENAT

**11 Equalising semantics for alignments 133**

11.1 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 135

**Interlude 5: Data interlinking 137**

**12 Application: Distributed query evaluation 139**

12.1 Semantic peer-to-peer systems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 139

12.2 The semantics of semantic peer-to-peer systems . . . . . . . . . . . . . . . . . . . . . . . . 140

12.3 Exercises . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 141

12.4 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 142

**13 Conclusion 145**

**IV Answers to exercises 147**

**部分 I**

图与本体

Graphs and ontologies

**第二章** **RDF: 资源描述框架**

在语义Web上提出的第一个问题是， Web上事实数据（factual data）的表达。为了解决这个问题，需要一种适应Web特性的数据表达式语言（data expression language）。资源描述框架（Resource Description Framework, RDF）是W3C推荐的语言在语义网[MANOLA and MILLER 2004]上表达数据。它在实践中得到了广泛的应用。

本章介绍简单的RDF，即，RDF没有具体的（RDF和RDFS）词汇。我们首先回顾（2.2节）的抽象语法、语义（2.3节），利用简单的解释模型，简单的蕴涵，然后2.4节使用同态刻画简单的RDF蕴涵，替代等效插值引理（equivalent interpolation lemma）的概念。2.5部分介绍了RDF蕴涵问题及其复杂性。

**2.1 IRI (与 XML): 基本层**

国际化资源标识符（Internationalized Resource Identifier, IRI）推广了统一资源标识符（Uniform Resource Identifier, URI[BERNERS-LEE et al. 1998]），URI 本身又推广了统一资源定位器（Uniform Resource Locater, URL），用于不仅识别网页而且识别任何资源（人、书、作者属性）。一个IRI代表一个精确的资源，但一个资源可能有几个IRI。

在人工智能实践中，IRI的使用是特定于语义Web技术的。IRI表现为简单标识符。但是，它们通常与命名空间（name space）相关联，以确定它们的出处（词汇表或数据源）。例如，foaf:人识别在friend-of-a-friend的命名空间http://xmlns.com/foaf/0.1/（简称foaf:）中人的概念。因此，RDF图可以以很小冲突或歧义的风险使用不同的词汇表。

语义Web松散地依赖于XML。XML仅被认为是一种传输格式。RDF被确切地定义为抽象对象（abstract object）。收集的RDF语句（RDF三元组）可以直观地理解为一种有向标记图（directed labelled graph）：资源（resources）是节点（nodes）和语句（statements）是连接节点的弧（arcs）（从主题节点到对象节点）。

**2.2 RDF 语法**

RDF可以用多种格式表示。最初的格式是RDF/XML [CAROTHERS and SEABORNE 2014]和N-triple[GANDON and SCHREIBER 2014]。RDF 1.1也开始接受各种格式用于表达RDF图：超越RDF/XML和N-Triples。新定义的格式是RDFa—使RDF嵌入在HTML—，Turtle [BECKETT 2006]，N-quads通过四边形扩展了N-triple—，以及JSON-LD。这些格式通常由RDF数据管理系统支持，所谓的三元组存储（*triple stores*）系统。

在这里，我们使用它的抽象语法（三元组格式），这对于大多数目的来说是足够的。

**RDF术语**

要定义RDF的语法，我们需要介绍RDF图构造的术语（terminology）。

**定义1** (RDF术语 [HAYES 2004]). RDF术语是三个两两不相交的无限集的联合：

* IRI集合 ，
* 文字/文本（literals）集合（它自己分为两个集合，普通文字（plain literals）集合和类型文字（typed literals）的集），即，以及
* 变量（variables.）集合。

集合的名字称为词汇（vocabulary）。

从现在开始，我们对这些集合的元素使用不同的符号：每个变量都会被加上？前缀（像?b1），文字将处于引号之间（像"27"），而其余的就将是IRI（如foaf:Person—foaf:具体见<http://xmlns.com/foaf/spec/>，是一个名字空间前缀，用于表示个人信息—如：ex:friend或friend）。

为了简化符号，并且不损失一般性，我们不区分简单（普通）文字和类型文字。

**2.2.1 RDF图为三元组集合**

RDF图，是建立在IRIs，空白（blanks）与文字（literals）之上的三元组的集合[CARROLL and KLYNE 2004]。因为我们想强调RDF结构与经典逻辑（classical logic）的兼容性，我们将使用术语变量（term variable），而不是“空白”这个词，它是RDF的一个特有词汇。空白对变量的特殊性就是量化（quantification）。的确，在RDF上的一个空白，是量化存在于一个特定图上的一个变量。我们更倾向于保留这种经典的解释，当RDF图放置在不同的上下文中时，这种解释非常有用。

**定义2** （RDF图）. 一个RDF三元组（RDF triple），是的一个元素。一个RDF图（RDF graph），是RDF三元组一个的有限集合。

在RDF设计中，排除变量作为谓词和文字是不必要的限制，RDF 1.1中已经放宽了限制。这些约束使得语法规范复杂化，松弛了这些语法规范，既不会改变RDF语义，也不会影响推理的计算性能。因此，我们采用引自[TER HORST 2005] 称为广义RDF图（Generalized RDF Graphs）的一种扩展，或者干脆简称GRDF图。

**定义 3**（GRDF图）. 一个GRDF三元组，是的一个元素。一个GRDF图是GRDF三元组的一个有限集合。 is an element of T \_ (U [ B) \_ T .

所以，每一个RDF图都是一个GRDF图。

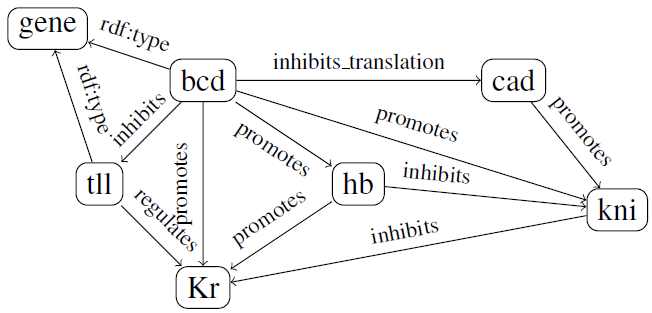


图2.1：一个表示果蝇胚胎早期发育过程中基因间相互作用的RDF图

{

< ?b1 foaf:name "Faisal" >,

< ?b1 ex:daughter ?b2 >,

< ?b2 ?b4 ?b3 >,

< ?b3 foaf:knows ?b1 >,

< ?b3 foaf:name ?name >

}

直观地说，这张图意味着，存在一个实体命名为（foaf:name）"Faisal"，它有一个女儿（ex:daughter）与另一个名字不确定的实体有某种关系，并且该实体认识（foaf:knows）名字为"Faisal"的实体。

**标记法**  在三元组中，称为主语（subject），是谓语（predicate），是宾语（object）。

如果是一个RDF图，我们用；；；；分别表示术语（terms）、IRI、文字（literals）、变量（variables）或在上至少一个三元组出现的名字（names）的集合。我们用表示做为在*GRDF*图的一个三元组中的一个主题出现的元素集合。和是相同方式定义的在谓语和宾语。我们称为的节点，是在的一个三元组中做为主语或宾语出现的元素的集合，即。一个术语（term）是的一个元素。如果是一个术语（term）集合，我们用代表。例如，是出现于中的名字（names）集合。

一个基础GRDF图（ground GRDF graph）是一个没有变量GRDF图，即（见例子2）。

**示例2**（ RDF图）。RDF可以用于揭示大量的各种各样的数据。例如，图2.1显示了代表果蝇胚胎基因调控网络部分的RDF图。节点（node）代表基因且属性（property）表达调控环节，即源基因表达对目标基因表达的影响。这幅图的三元组形式如下：

**dm:bcd rdf:type rn:gene.**

**dm:bcd rn:inhibits translation dm:cad.**

**dm:bcd rn:promotes dm:hb.**

**dm:bcd rn:promotes dm:kni.**

**dm:bcd rn:promotes dm:Kr.**

**dm:bcd rn:inhibits dm:tll.**

**dm:cad rn:promotes dm:kni.**

**dm:hb rn:inhibits dm:kni.**

**dm:hb rn:promotes dm:Kr.**

**dm:kni rn:inhibits dm:Kr.**

**dm:tll rn:regulates dm:Kr.**

**dm:tll rdf:type rn:gene.**

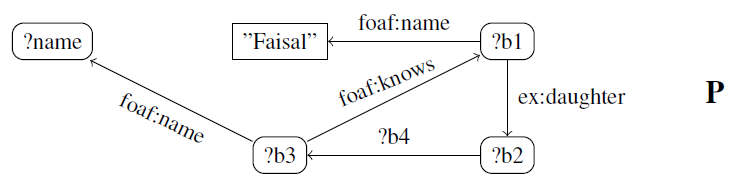


图2.2：一个GRDF图

此示例仅使用IRI。

**2.2.2 RDF三元组的图表示**

一个简单的GRDF图，可以以图形化方法表示为一个有向标记多重图（directed labeled multigraph），在该图中，节点集是在图中的至少一个三元组中做为一个主语或宾语出现的术语的集合，弧集是图的三元组集，把节点对（弧的端点）连接每条弧上，在此，是弧的源头（source）且是弧的目标（target），标记图的所有节点和弧：如果是中的一个节点，即一个术语，那么，并且如果是中的一个弧，即一个三元组，那么，即，如果是一个三元组，那么就是一条弧。绘制这种图形时，由文字（literal）产生的节点用矩形（rectangle）表示，而其他节点用圆角矩形（rounded corner）表示。

例如，在例1的GRDF元组，可以用图2.2所示。

RDF 1.1已经确认RDF数据集（Dataset）的概念，它是RDF图和命名图（named graph）的集合，这些图是由IRI标识的图。这种识别（identification）可以断言为关于图也即三元组的多个语句（三元组）。这样的语句（statement）可能是关于该图的起源（origin）或是对它断言的断言（confidence）。这个功能是以前所完成“四边形”，即通常由一个IRI标记的三元组。

接下来，我们不再在关于RDF语法的这两种观点之间做区分（做为三元组集合或者有向标记图）。然后，我们将会交替着地说它们的节点、它们的弧或者构成它们的三元组。

**2.2.3 RDF三元组的逻辑表达**

一些RDF图可以转化为正（无否定）、合取、存在和无函数的一阶逻辑的公式。这些图中谓词不作为主语或宾语使用。对每个三元组，对应着原子公式（atomic formula），因此，是一个谓词名称，如果这些元素是IRI或文字，则和是常量，否则为变量。一个图转化为与其三元组相关的原子公式的联合的存在闭包（existential closure）。因此，图2.2中带有由rel:worksWith的?b4的图，就被转化为：

?*b*1, ?*b*2, ?*b*3, ?*name*, foaf:name(?*b*1, "*Faisal*")

rel:daughter(?*b*1, ?*b*2) rel:worksWith(?*b*2, ?*b*3)

foaf:name(?*b*3, ?*name*) foaf:knows(?*b*3, ?*b*1)

The models of such a formula are isomorphic to those of the simple RDF semantics of the graph (we state it without proof).

这样一个公式的模型与图的简单RDF语义的模型是同构的（我们不证明地声明它）。

**2.3 简单RDF 语义学**

[HAYES 2004]介绍了RDF图的几种语义。在本节中，我们只展现没有RDF/RDFS词汇[BRICKLEY and GUHA 2004]的简单语义。解释、模型、可满足性和蕴涵的定义，对应于[HAYES 2004]的简单解释，简单模式，简单可满足性和简单蕴涵。一个RDF图的语义，在模型理论[HAYES and PATEL-SCHNEIDER 2014]中定义。因此，RDF是一个适当的逻辑。**可以在多项式内**，通过RDF或RDFS规则[BAGET 2005; TER HORST 2005]（见第3.1.3），将RDF和RDFS的结果（或蕴涵/框架）约减为简单蕴涵。

本演示文稿的主要是基于[HAYES 2004]，但可以借助于[HAYES and PATEL-SCHNEIDER 2014]进行重新考虑。

**2.3.1解释（Interpretations）**

一种解释（interpretation）描述了世界可能的方式，以便确定任何基础RDF图（ground RDF graph）的真值（truth-value）。它通过指定每个IRI（和每个文字）的外延（denotation）来实现这一点。此外，如果IRI用于指示一个属性（property），则它指定了在该论域（universe）中哪些事物（things）与此属性相关。

在一个给定的词汇（vocabulary）中将特定的含义分配给某些名称的解释，将依据该词汇表进行命名，例如，RDFS解释（见3.1节）。在词汇（包括RDF词汇表本身）中，一个没有特定附加条件的解释，将被称为解释（interpretation）。

**定义 4** (**词汇的解释**). 令 为一个词汇（vocabulary）, 的一个解释（interpretation）是一个四元组 〈〉，以致：

* 是包含的一个资源（resource）集合；
* 是一个属性（property）集合；
* 将每个属性关联于一个称为属性外延（extension）的资源对（pairs of resources）集合；
* 解释函数，将中的每个名字关联于一个资源，如果，那么。

这种语义学（Semantics）具有这种性质，谓词（predicate），自然对应于经典二元谓词（dyadic predicate），也可以看作是资源（resource）。因此，三元组在RDF中是合法的，并且可以解释（它实际上意味着rdf:type表示一个谓词）。这种解释，通过解释三元组以两个步骤来实现：第一步（）将一个外延（denotation）关联于每个被使用的IRI，第二步（）将在谓语位置中使用的那些外延解释为二元关系。这是RDF语义学，相对于一阶逻辑语义学的主要特点。

**2.3.2 模型（Models）**

通过提供RDF形式语义，[HAYES 2004; HAYES and PATEL-SCHNEIDER 2014]表示了各种条件（conditions），在这些条件下，一个**解释**（interpretation）就**是**一个RDF图的一个**模型**（model）。通常的有效性（validity）、可满足性（satisfiability）和结果（consequence）概念，则完全取决于这些条件。

直观地，在一个GRDF图中的一个基础三元组（ground triple），在解释下，将取得真值，如果被解释为一个属性（例如）且资源对属于属性的外延（扩展，extension）。一个三元组具有变量，在解释下，将取得真值，如果存在一个资源，这样资源对就属于外延（扩展）。当一个变量总是可以依据同一资源进行解释，那么该解释函数（interpretation function）的外延定义如下：

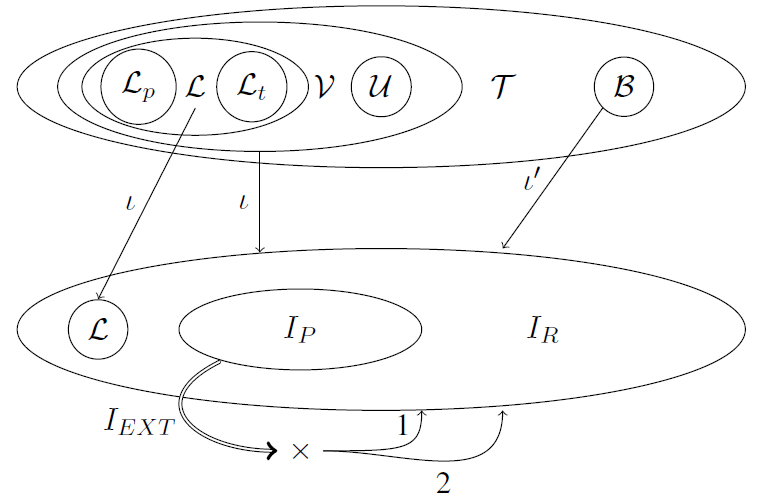


图2.3：简单RDF语义的域结构

**定义5 (变量的外延**（扩展）**，Extension to variables)**. 设〈〉为一个词汇（vocabulary）的一个解释（interpretation），并且是一个变量集合。解释函数对的外延（扩展）就是一个映射，以致，。

一个解释就是一个GRDF图的一个模型，在下所有的三元组都为真。

**定义 6 (一个GRDF的模型)**. 设为一个词汇，为一个GRDF图，那么在中出现的每个名字（name），也在中。的一个解释〈〉是的一个模型，当且仅当，存在一个外延（扩展）将到，这样对的每个三元组，并且。映射被称为在中的一个证据（proof）。

**2.3.3可满足性、有效性、蕴涵及结果**

以下定义是可满足性（satisfiability）、有效性（validity）、蕴涵（entailment）和结果（consequence）的标准模型理论定义。

**定义 7 (可满足性, 有效性, 结果)**. 一个图是可满足的，当且仅当它有一个模型。当且仅当对一个词汇每种解释，图是有效的，那么就是图的一个模型。一个图是一个图的结果，或者蕴含（），表示为，当且仅当图的每个模型也是图的一个模型。

**命题（Proposition）1（空图，子图，实例引理[HAYES 2004]）。**

**空图引理**（Empty graph lemma）三元组的空集合被任何图所决定，除它本身之外，不包含任何图；

**子图引理**（Subgraph lemma）一个图决定其所有的子图（即：三元组的子图）；

**实例引理**（Instance lemma）一个图由其任何实例决定（即：由值来替换变量）。

**证明框架**（Sketch of proof）.

子图引理之所以成立，是因为一个子图的任何一个子模型对图是一个子图的解释强加了较少的约束。因此，任何的解释，将构成图的解释。这也需要空引理（Empty lemma），因为空图（empty graph）是任何图的子图（subgraph）。关于实例引理，实例化的图比初始图施加更多的约束，因为相应变量的赋值是固定的。因此，实例的任何模型都是一般图（general graph）的模型（带有对每个实例化变量分配实例化值的空白的扩展）。最后，任何图都是与其他图结合的子图，由子图引理构成。

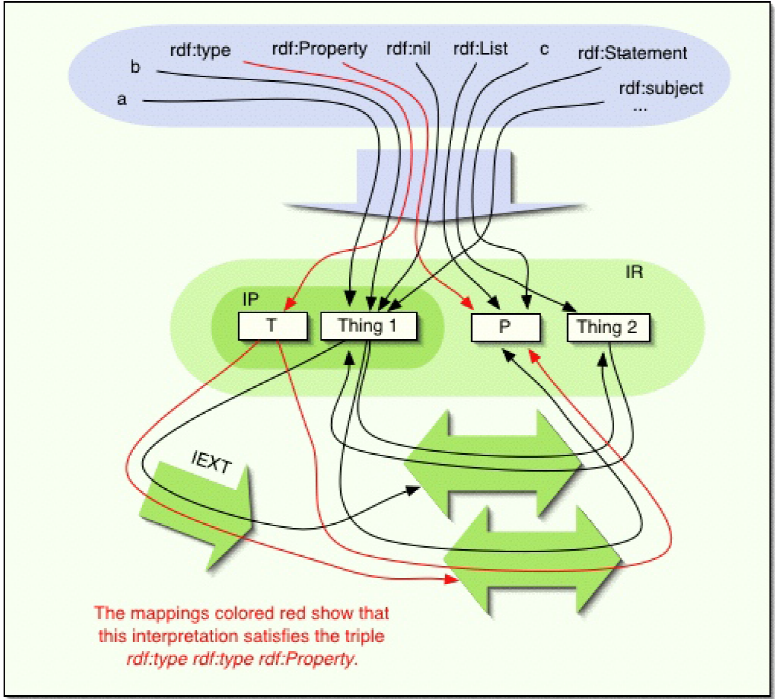


图2.4：域结构解释的示例（来自[HAYES 2004]）

目前，以下两个命题依靠相同的构造：Herbrand模型或同构模型。

**命题2** (可满足性, 有效性 [BAGET 2005; TER HORST 2005]). 每个GRDF图是可满足的. 唯一有效的GRDF图是空图。

**证明**. (可满足性) 见命题3的证明。

（有效性）一个非空GRDF图在一个解释中都没有证据，在此，所有属性都被解释为一个空集。

**命题3** (插值定理Interpolation lemma [HAYES 2004]).

**证明**. ：因子图和实例引理，这显然是真的依据：如果是的一个子图，那么，因为的所有模型平凡地满足的三元组（子图引理）。如果是的一个实例，那么，因为，对的每个模型〈〉来说，对满足所有三元组的变量都存在一个外延（扩展），并且对每个三元组实例化一个三元组来说，对所有的变量存在一个外延（扩展），这样，对在该三元组中的每个元素，，如果，否则（实例引理）。

：图的Herbrand插值由来定义，这样：

* ；
* ；
* ，以致，；
* 插值函数是在上的识别函数（identity function）。

明显是的一个模型，因为它可以被识别函数在空白上进行扩展，而且平凡地满足所有约束 。定义6的条件直接遵循这种构造。这就证明了命题2的可满足性部分。

如果，那么也是的一个模型。这就意味着，存在对空白的一个扩展，满足中的每个三元组。所以，一定存在一个映射，这样，对于每个三元组，。但是假定的定义，这就意味着。对于该三元组的每个元素，或者（并且有）或者（并且有），存在它的一个实例化，这就证明了的一个子图是的一个实例化。

**2.4 推理机制（Inference mechanism）**

简单RDF蕴涵[HAYES 2004]可以被描述为一种图同态（graph homomorphism）。

一个从RDF图到RDF图的图同态，按照[BAGET 2005; GUTIERREZ et al. 2004]中的定义，是一个从图的节点到图的节点保持图结构的映射，即，对于每个节点，如果那么；并且有每个边被映射到。这个定义，类似于用于刻画**概念图**（conceptual graph）蕴含的**投影**（projection）[CHEIN and MUGNIER 2009]（见[CORBY et al. 2000]用于RDF与概念图之间的一个精确关系）。我们该定义修改成将映射到的定义。映射被用于确保一个变量永远映射到同一术语（term），正如对解释（interpretation）的扩展（extension）所做的那样。

**定义8 (映射Map).** 设且是两个术语集，从到的一个映射（map）就是一个映射，以致，。

显然，只要，就存在这样一个映射。一个RDF同构就是一个保持该图结构的映射。

**定义9 (GRDF同态).** 设和为两个GRDF图。一个从到的同态（homomorphism），就是一个从到的映射，以致，。

[GUTIERREZ et al. 2004] 没有证明地提供了一个在RDF蕴涵与映射之间的等价定理（定理3）。在[BAGET 2005]中，也对RDF图提供了一个证明，但所设计的同态（homomorphism）只是一个从节点到节点（nodes-to-nodes）的映射，而不是从术语到术语（terms-to-terms）的映射。在RDF中，这两个定义是等价的。然而，这个术语到术语（terms-to-terms）协议版本扩展RDF的定理（定理4）到在[ALKHATEEB, BAGET, et al. 2009]中研究的PRDF图是必要的。所有证明均可在在[ALKHATEEB 2008]获得。

**例 3 (GRDF 同态)**. 图 2.5显示两GRDF图和（注意图是图2.2中的图，其中，以下三元组加入到该图。由{(*"Faisal"*, *"Faisal"*), (*?b1*, *?c1*), (*?name*, *"Natasha"*), (*?b2*, *?c2*), (*?b4*, *ex:friend*), (*?mbox*, *"natasha@yahoo.com"*), (*?b3*, *ex:Person1*)}定义的映射，是一个从到的GRDF同态。并且，由{(*"Faisal"*, *"Faisal"*), (*?b1*, *?c1*), (*?name*, *"Deema"*), (*?b3*, *ex:Person2*), (*?b4*, *ex:friend*), (*?b2*, *?c2)*}定义的映射，是从到的一个GRDF同态。注意，无法扩展到从到 的一个GRDF同态，因为根本没有邮箱对应在中的*"Deema"*。

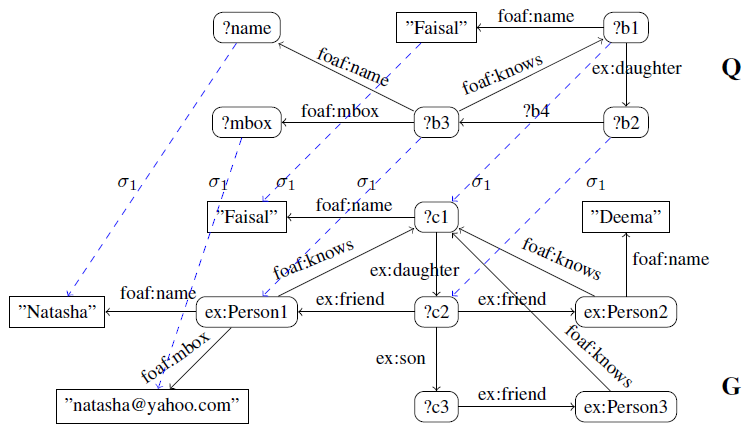


Figure 2.5: 一个从到的GRDF同态

**定理（Theorem）4.** 设和是两个GRDF图，那么，当且仅当，存在一个从到的GRDF同态。

**证明（Proof）**. 这个定理的证明被留作习题（它是在[ALKHATEEB 2008]给出的）。它依赖于显示从到的一个同态，确定的一个子图（由同态所表示的中的映像），并且根据定义9，这个子图必然是的一个实例。然后，根据命题3，就解释了等价性。

在蕴涵的语义概念和同态的句法概念之间的这种等价关系（equivalence）是基础（ground），借此，可以设计一个正确和完整的查询应答过程（query answering procedure）。更确切地说，在RDF知识库上，对一个GRDF图查询的应答集，是从到的RDF同态集，依据定理4，对应于RDF的结果（consequence）。

**2.5 计算的性质（Computational properties）**

与简单RDF语义相关的决策问题，称为简单RDF蕴涵（SIMPLE RDF ENTAILMENT），并定义如下：

简单 (G)RDF蕴涵 SIMPLE (G)RDF ENTAILMENT

**实例**: 两个GRDF图和. **Instance**: two GRDF graphs G and H.

**提问**: 是否? **Question**: Does ?

对于RDF图来说，简单 (G)RDF蕴涵，是一个NP-完全问题[GUTIERREZ et al. 2004]。对于GRDF图来说，它的复杂性仍然不变[P´EREZ et al. 2009]。

**证明提示**（*Hint of proof*）。图的匹配，一个NP-完全问题，利用源图（source graph）的具有不同IRI所有的节点、目标图（target graph）空白的所有节点以及每个边上的相同谓词，可以约减到在RDF图中的蕴涵。

基于查询的结构或标记，已经展示了问题的多项式子类（Polynomial subclasses）：

* 当查询是范围（ground）[TER HORST 2005]，或更一般地，如果它有一个有界的变量数，
* 查询是一棵树或允许有界分解为一棵树，依据在[BAGET 2005]中阐明的[GOTTLOB et al. 1999]中的方法。

**2.6 结论（Conclusion）**

RDF（或简单RDF）是一种语言，允许将信息表示为一个图，其中资源由谓词关联。这种语言，具有模型理论语义，允许在这样的图之间定义结果（或蕴含）的概念。此外，蕴涵等价于在已蕴含图（entailed graph）和待蕴含图（entailing graph）之间的一个同态的存在。就其一般性而言，蕴涵检查是一个NP-完全问题。

这就要考虑去检查一个公式（查询）是否是RDF图中所表示的信息的一个结果。

**2.7 习题（Exercises）**

**习题1** (RDF断言). 考虑图2.6的四个图。

1. 它们都是良好的RDF图吗？为什么？
2. 将图2.6(a) 表示为一个三元组集合。你要列出在该图中发现的文字集、IRI集和变量（或空白）集。
3. 用谓词演算形式，给出图2.6的一个非正式意义或它的表达式。
4. 考察图2.6中哪个图是格式良好的RDF图，其中有些图是否蕴含其它的图？解释为什么。

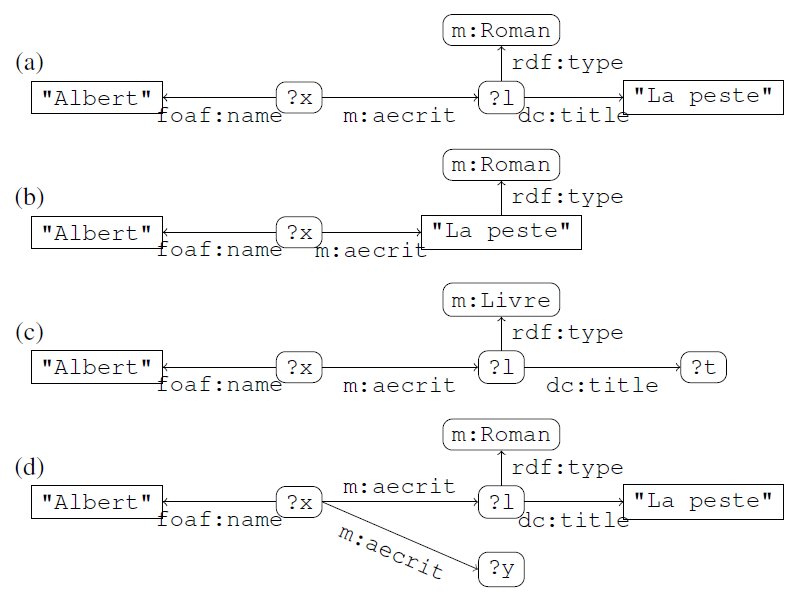


图2.6: 四个RDF图

**习题2** (RDF与断言). 考虑图2.7中关于社交关系表达的两个图。

1. 将图2.7(b) 表示为一个三元组集合。你要给出在本图的文字、IRI和变量（或空白）列表。
2. 用非形式化方式，解释图2.7(b)中图的意思是什么（用英语或法语或谓词演算来描述它）？
3. 这些图中有一个图蕴含另一个图吗？（解释原因）

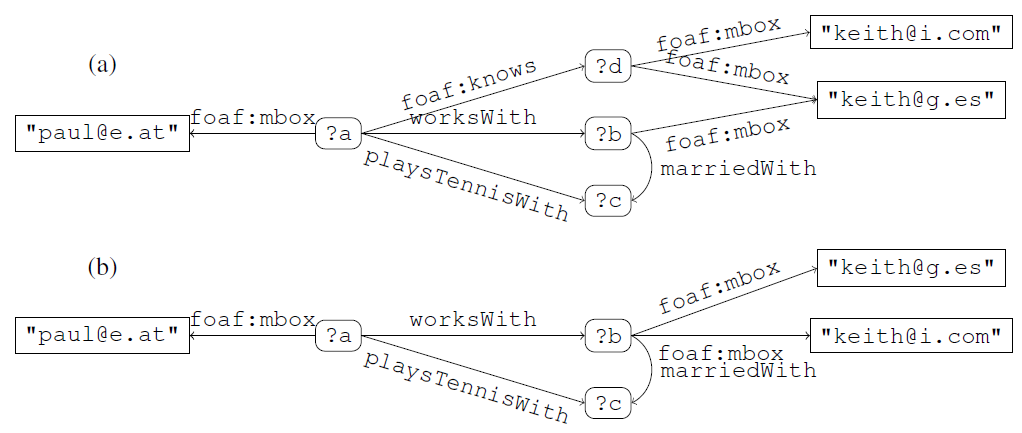


图2.7: RDF图

**习题 3** (RDF图). 这里有一个RDF图关于作家及其作品的8个三元组：（所有标识都事实上对应于IRI，*\_:b*是一个空白节点）：

1. 绘制与这些语句相对应的RDF图
2. 用英语表达这些语句的意思。

**习题4** (RDF操纵). 考虑构成RDF图的下列语句：

ex:book1 rdf:type mr:Book .

ex:book1 dc:title "For whom the bell tolls" .

ex:book1 dc:date 1940 .

ex:book1 dc:creator ex:eh .

ex:book1 mr:storedIn "Living room" .

ex:book2 rdf:type mr:Book .

ex:book2 dc:title "Pour qui sonne le glas" .

ex:book2 mr:translationOf ex:book1 .

ex:book2 dc:creator ex:eh .

ex:book2 dc:date 1948 .

ex:book2 dc:publisher ex:gal .

ex:movie1 rdf:type mr:Movie .

ex:movie1 dc:title "For whom the bell tolls" .

ex:movie1 mr:adaptedFrom ex:book1 .

ex:movie1 mr:director ex:sw .

ex:movie1 mr:cast ex:ib .

ex:movie1 mr:cast ex:gc .

ex:movie1 mr:storedIn "Computer drive" .

ex:movie1 dc:date 1943 .

ex:eh rdf:type foaf:Person .

ex:eh foaf:name "Ernest Hemingway" .

ex:sw rdf:type foaf:Person .

ex:sw foaf:name "Sam Wood .

ex:ib rdf:type foaf:Person .

ex:ib foaf:name "Ingrid Bergman" .

ex:gc rdf:type foaf:Person .

ex:gc foaf:name "Gary Cooper" .

ex:gal rdf:type foaf:Organization .

ex:gal foaf:name "Gallimard" .

1. 图中有哪些不同的类？他们为什么是类？
2. 用图形形式将表示为一个图形。

将以下语句考虑为RDF图 ( 是空白标识):

\_:x rdf:type mr:Movie .

\_:x mr:adaptedFrom \_:y .

\_:y dc:creator \_:p .

\_:p foaf:name "Ernest Hemingway" .

1. 解释图。
2. 图是否被图蕴含？详细解释为什么。

**习题5** (RDF 蕴含). 考虑图2.8中的图。

1. 解释该图意味着什么（用英语或法语，考虑到RDFS词汇的含义）。
2. 把它改写为一组三元组。

给定GRDF图*P*={*?x rdf:type a:CHU. ?x a:hasService ?y. ?y rdf:type a:Maternity.*}

1. 吗？（解释如何以及/或为何）

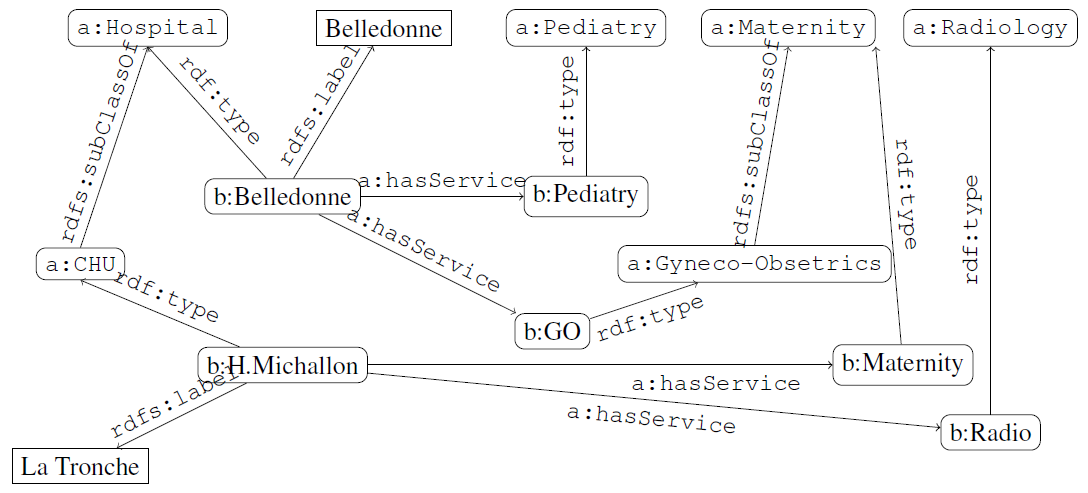


图2.8: RDF 图 *G*.

**习题6** (RDF 蕴含). 我们有来自两个不同资源的两个图。考虑图来自由下列三元组构成的自治市（municipality）:

a:Pierre o1:father a:Carole . a:Marie o1:mother a:Carole .

a:Pierre o1:father a:Kevin . a:Marie o1:mother a:Kevin .

a:Jacques o1:father a:Jean . a:Marie o1:mother a:Jean .

\_:b1 o1:father a:Sylvie . a:Stephanie o1:mother a:Sylvie .

\_:b1 o1:father a:William . a:Stephanie o1:mother a:William .

a:Jacques o1:father a:Julie . a:Nabila o1:mother a:Julie.

a:Sven o1:father a:Laurent . a:Lucie o1:mother a:Laurent .

且图为来自由下列信息构成的学校（school）:

a:Carole o2:attendsClass b:4e3. a:Carole rdf:type o2:Female .

a:Kevin o2:attendsClass b:6e1 . a:Kevin rdf:type o2:Male .

a:Sylvie o2:attendsClass b:5e2 . a:Sylvie rdf:type o2:Female .

a:William o2:attendsClass b:5e2 . a:William rdf:type o2:Male .

a:Julie o2:attendsClass b:5e2 . a:Julie rdf:type o2:Female .

a:Laurent o2:attendsClass b:4e3 . a:Laurent rdf:type o2:Male .

a:Jasmine o2:attendsClass b:5e1 . a:Jasmine rdf:type o2:Female .

1. 把这两个图（一起）画出来；
2. 为了处理这两个图，我们想回答贯穿它们两个的查询。考虑下面的图：

\_:x o2:attendsClass \_:w .

\_:y o2:attendsClass \_:w .

\_:x rdf:type o2:Male .

\_:y rdf:type o2:Female .

\_:z o:parent \_:x .

\_:z o:parent \_:y .

用英语表达的含义。被或蕴含吗？（解释原因）

1. 表达与英语对应的图：“有两个人出现在同一个班上至少共享一个父母吗？”或吗？

**习题7** (RDF解释和蕴涵). 考虑图描述假日包（holiday packages）:

\_:b1 rdf:type o:Package . d:PousadaDesArts rdf:type o:Pousada .

\_:b1 o:destination d:Salvador . \_:b1 o:activity \_:b2 .

\_:b1 o:accomodation d:PousadaDesArts . \_:b2 rdf:type o:Swimming .

\_:b3 rdf:type o:Package . d:Metropol rdf:type o:GrandHotel .

\_:b3 o:destination d:Moskow . \_:b3 o:activity d:VolgaCruise .

\_:b3 o:accomodation d:Metropol . d:VolgaCruise rdf:type o:Cruise .

\_:b4 rdf:type o:Package . d:ToyofukuRyokan rdf:type o:Ryokan .

\_:b4 o:destination d:Kobe . \_:b4 o:activity \_:b5 .

\_:b4 o:accomodation d:ToyofukuRyokan . \_:b5 rdf:type o:SwordFighting .

1. 绘制图。
2. 定义的词汇()的RDF-解释。
3. 给定下面的图：

\_:x rdf:type o:Package .

\_:x o:accomodation \_:acc .

\_:x o:activity \_:act .

你的解释是否满足（换句话说，是的一个模型）吗？

1. ？给出解释。
2. 给定下面的图：

\_:y rdf:type o:Package .

\_:y o:accomodation \_:acc .

\_:acc rdf:type o:Local .

\_:y o:activity \_:act .

\_:act rdf:type o:Sport .

？解释为什么。

**数据网络（The web of data）**

在2006年，根据观察，由于缺乏资源，语义Web的发展已经相对落后，Tim Berners Lee提出了在Web上发布大量数据的一个方法学，这样就对语义网有所帮助[BERNERS-LEE 2006]。

这实际上通过在网络上发布数据的四个原则定义了“数据的网络（web of data）”，概括为[BERNERS-LEE 2006; HEATH and BIZER 2011]：

1. 依据IRI识别各种资源（resources）。
2. IRI是可以引用的（dereferenceable），即可以在Web上查找。
3. 当IRI是可以引用的，就应该返回关于已识别资源的一个描述，理想地适合通过内容协商（content negotiation）。
4. 已发布的Web数据集必须包含与其它Web数据集的链接。

虽然没有明确的规定，链接数据源（linked data sources）是更可用的，如果它们与语义Web技术一同发布： IRI用于识别资源，RDF用于描述它们，OWL用于定义所使用的词汇，SPARQL用于访问数据，并通过owl:sameAs谓语声明（asserting）各数据源（sources）之间的链接。

“五星级评估（five stars rating）”，被解释为增量步骤（incremental steps），已经被引入来衡量这种可用性（usability）究竟有多大，即：

★以任意格式在Web上发布数据，如PDF中对一个表的一个扫描；

★★使用结构化数据格式，例如Excel中的表而不是PDF中表扫描；

★★★使用非专有格式，例如CSV，而不是Excel，以致用户就可以直接访问原始数据；

★★★★使用通用格式来表示数据，如RDF，它封装语法和语义；

★★★★★将数据链接到Web上的其他数据集，从而提供上下文（context）。

前三个星级是很容易到达，这些已经使某些数据重用。然而，人类仍然必须处理与集成相关的所有语义问题。为了有更易于发现（discoverable）和可互操作（interoperable）的数据，就有必要达到第四和第五星级。

因此，数据Web是一个巨大的RDF图，可以通过类似Web的HTTP协议来访问。很快地，数据Web就围绕DBPedia进行了开发[BIZER et al. 2009]，以RDF形式，对维基百科（Wikipedia）的一种大规模提取。许多成果都将他们的资源链接到了DBpedia。不久之后，一些政府鼓励以RDF形式发布他们（公开）数据。从那时起，图书馆、博物馆、各种研究机构，都加入了链接开放数据云（linked open data cloud）。在2016年，链接开放数据云已经包含了9960个数据源，提供超过1500亿个三元组，使用576个词汇或本体(<http://stats.lod2.eu/> 和 <http://lov.okfn.org>)。

**第三章**

**本体语言（Ontology languages）**

一旦能够在Web上表达数据，下一个问题，就是RDF图中使用的词汇表的定义。这意味着，列出用于描述数据的术语（term），以及表示约束使用这些术语的公理（axiom），从而允许机器正确地解释它们。这些词汇定义为本体（ontology）。以下，我们将本体视为特定逻辑中的公理集。我们讨论用于设计Web网页的主要本体语言：RDF Schema（X3.1）和OWL（X3.2）。

**3.1 RDF模式（RDF Schema）**

我们迄今所考虑的都仅仅是简单的RDF语义。完整的RDF，是通过识别特定词汇、RDF词汇以及在与该词汇相关的模型定义中添加约束来定义的。因为这些变化是细微的，我们就不考虑它们，而是直接定义包含完整RDF的RDFS模式。

RDFS（RDF Schema）[BRICKLEY and GUHA 2004]是RDF的一个扩展设计，用于描述资源之间关系，使用一组称为RDFS词汇的保留的IRI（reserved IRIs）。在上面的例子中，保留字（reserved word）rdf:type可用于将实例（instance）关联到类（class），例如，book1属于类型publication。

这一部分主要关注RDF和RDFS，作为第2章中提出的简单RDF语言扩展。这两个扩展都是以相同方式定义的：

* 它们采用分别以rdf:及rdfs:为前缀的词汇的一个IRI的特定集合。
* 它们在解释中添加与这些术语相关联的资源的附加约束。

在向RDFS解释添加新的约束中，RDFS文档可能有较少的模型，从而会有更多的结果。这是可能的，例如，在RDF中，依据〈ex:person1 ex:author "Alkhateeb"〉推断〈ex:author rdf:type rdf:Property〉；在RDFS中，依据{〈ex:document1 rdf:type rdf:Autobiography〉, 〈ex:Autobiography rdfs:subClassOf ex:Biography〉}推断〈ex:document1 rdf:type ex:Biography〉。

像往常一样，我们首先提出词汇（§3.1.1），然后提出对语言的解释添加的各种约束（§3.1.2）。此外，我们也介绍一种“规范”的推理机制（§3.1.3）。

**3.1.1 RDFS 作为一种 RDF 词汇**

在RDF和RDF Schema中，存在一组保留字（reserved words），RDF和RDF S词汇，用来描述资源间的关系，比如类（class），例如geo:City subClassOf geo:PopulatedArea，以及属性（property）之间的关系，例如country subPropertyOf is-contained-in。

RDF已经在rdf:命名空间中引入了几个关键字来构建知识（具体见表3.1）：

* 〈ex:Sonia rdf:type ex:Employee〉断言资源 ex:Sonia 是类ex:Employee的一个实例；
* 〈rel:worksWith rdf:type rdf:Property〉表明rel:worksWith 是一个谓语, 即, 用于标记边的一个资源。

用rdfs:名称空间中的少数几个关键词，将RDFS表示为RDF三元组形式，如：

* 〈ex:Employee rdf:type rdfs:Class〉表示资源 ex:Employee 属于类型 rdfs:Class，因此它是一个类。
* 〈ex:Employee rdfs:subClassOf foaf:Person〉意味着ex:Employee是foaf:Person的一个子类，因此ex:Employee的每个实例也是foaf:Person的一个实例，就像ex:Sonia；
* 〈ex:worksWith rdfs:range ex:Employee〉 断言 作为一个边（edge）的端点（extremity）标记以rel:worksWith的任何资源都是ex:Employee class的一个实例。

表3.1给出了RDFS词汇，以在[HAYES 2004]出现的形式。我们将使用的快捷方式（shortcut）在括号[]中给出具体词汇。

用于XML/RDF的某些术语，如rdf:about，rdf:parseType和rdf:resource，由于它们不以三元组形式出现，所以没有出现在表3.1中。它们只是用来说明如何将XML转换为三元组。从现在开始，我们用RDFSV表示RDFS词汇。

**RDF 类型归类词汇（RDF Typing vocabulary）**

rdf:type[type] rdf:Property[prop] rdf:XMLLiteral[xmlLit]

**具体化的词汇（Reification vocabulary）**

rdf:Statement[stat]

rdf:subject[subj] rdf:predicate[pred] rdf:object[obj]

**容器词汇（Container vocabulary）**

rdf:first[first] rdf:rest[rest] rdf:nil[nil]

rdf: 1[1] rdf: 2[2] rdf: i[i]

rdf:List[list] rdf:Bag[bag] rdf:Seq[seq]

rdf:Alt[alt]

**杂项词汇（Miscellaneous vocabulary，不要使用)**

rdf:value[value]

**RDFS类型归类词汇（RDFS Typing vocabulary）**

rdfs:Class[class] rdfs:Resource[res] rdfs:Literal[literal]

rdfs:domain[dom] rdfs:range[range] rdfs:Container[cont]

rdfs:subClassOf[sc] rdfs:subPropertyOf[sp] rdfs:Datatype[datatype]

**容器词汇（Container vocabulary）**

rdfs:ContainerMembershipProperty[contMP] rdfs:member[member]

**文档词汇（Documentation vocabulary）**

rdfs:comment[comment] rdfs:label[label] rdfs:seeAlso[seeAlso]

rdfs:isDefinedBy[isDefined]

表3.1: RDFS 词汇

**例4 (RDFS-图).** 图3.1 显示了带有下列断言的 RDFS 图：

geo:Grenoble departement geo:Isere

departement rdfs:subPropertyOf is-contained-in

is-contained-in rdfs:range geo:GeographicArea

is-contained-in rdfs:domain geo:GeographicArea

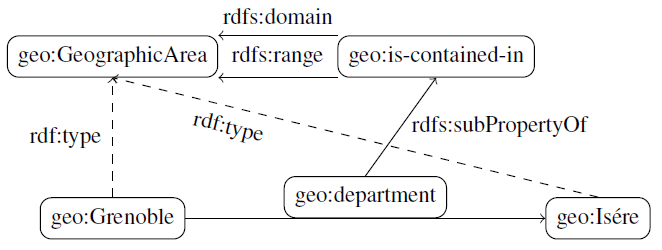


图3.1: 一个RDFS图. 虚线箭头是RDFS语义的结果

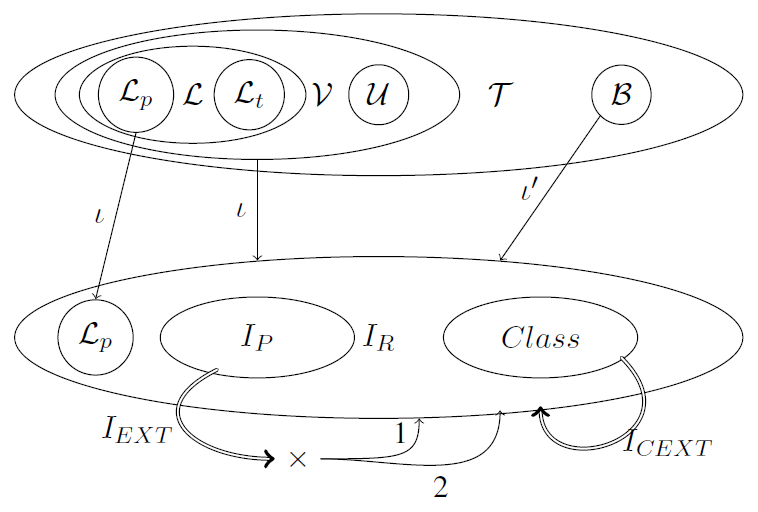


图3.2: RDFS 语义的域结构（Domain structure）

一些作者介绍*ρ*DF [MUN OZ et al. 2009]作为语言对应的RDF和RDFS类型归类词汇，而且一些文件提到“RDFS的描述逻辑片段”，这很可能是同一事物。这里，我们将集中讨论这种片段，因为它是语义相关的片段。

**3.1.2 RDFS语义（RDFS semantics）**

根据语言的特定词汇的RDFS图模型的设定限制了RDFS语义。通过将某些IRI视为识别类而形成的RDF词汇作为信息资源IR的一个子集，从而实现了扩展。这些就要以两个步骤将它们解释为属性（见§2.3）。

**定义10 (RDFS 解释).**一个词汇 的一个RDFS解释是一个元组〈, , *Class*, , , *Lit*,〉，这样：

* 〈, , , 〉是一个RDF解释；
* 如果一个资源代表一个资源类，则*Class*是识别的一个区别子集（distinguished subset）；
* 是一个映射（map），它将一组资源分配给每一个代表类的资源；
* *Lit*是文字值（literal values）集合，*Lit*包含所有在中的普通文字。

在一个RDFS解释成为一个RDFS图的RDFS模型中，特定条件（specific conditions）要添加到与RDFS词汇的术语相关的资源。这些条件包括对出现在RDF规范语义[HAYES 2004]中的RDF和RDF Schema公理三元组（axiomatic triples）的满足（定义11、12）。

**定义11** (RDF 公理三元组). RDF 公理三元组是在下述无限集（infinite set）中的三元组：

〈type, type, prop〉 〈subj,type,prop〉

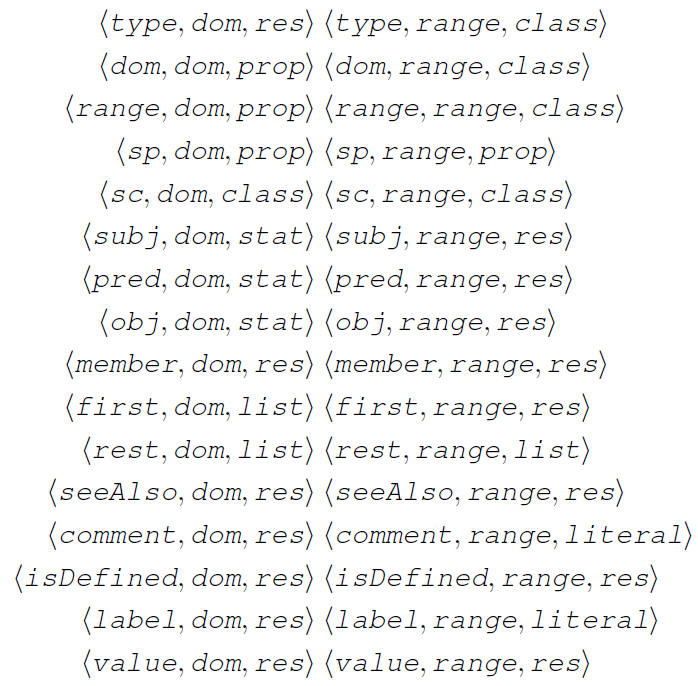
〈obj,type,prop〉 〈pred,type,prop〉

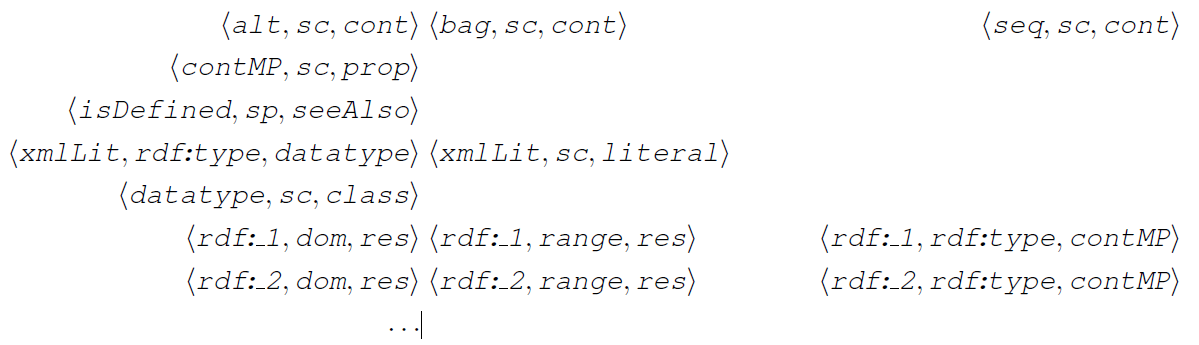
〈first,type,prop〉 〈rest,type,prop〉

〈value, type, prop〉 〈nil,type,list〉

〈rdf:1,type,prop〉 〈rdf:2,type,prop〉 …

**定义12** (RDFS 公理三元组). RDFS 公理三元组在下述无限集（infinite set）中的三元组：





根据这个定义，很明确的，任何RDFS解释都是一个RDF解释，而且RDF解释都可以扩展为不同的RDFS解释：它足以识别各种类。

**定义13** (RDFS 模型). 设是一个RDFS图，〈, , *Class*, , , *Lit*,〉为词汇的一个RDFS解释，以致。那么，就是的一个RDFS模型，当且仅当满足下列条件：

1. 简单语义（Simple semantics）：
2. 存在对的一个扩展，以致对于的每个三元组，，并且。
3. RDF语义（RDF semantics）：
4. 〈,(*prop*)〉∈((*type*))。
5. 如果是一个用词汇表（lexical form）确定类型的XML文字，那么就是的XML文字值，，并且〈,(*XML Lit*)〉∈((*type*))。
6. 满足全部RDF公理化三元组（定义11）。
7. RDFS类（RDFS Classes）：
8. ，((*res*))。
9. ，((*Class*))。
10. ，((*literal*))。
11. RDFS 子属性（RDFS Subproperty）：
12. ((*sp*))在上是传递的（transitive）和反射的（reflexive）。
13. 如果〈,〉∈((*sp*))，那么并且()()。
14. RDFS 子类（RDFS Subclass）：
15. ((*sc*)) 在*Class*上是传递的（transitive）和反射的（reflexive）。
16. 如果〈,〉∈((*sc*))，那么并且()()。
17. RDFS 类型归类（RDFS Typing）：
18. ()，〈,〉∈((*type*))。
19. 〈,〉∈((*dom*))并且〈,〉∈()，那么()。
20. 〈,〉∈((*rang*))并且〈,〉∈()，那么()。
21. RDFS 附加（RDFS Additionals）：
22. 如果，那么〈,(*res*)〉∈((*sc*))。
23. 〈,〉∈((*datatype*))，那么〈,(*literal*)〉∈((*sc*))。
24. 〈,〉∈((*contMP*))，那么〈,(*member*)〉∈((*sp*))。
25. 满足全部RDFS公理化三元组（定义12）。

任何RDFS模型都是一个RDF模型。这是不正确的，任何RDF模型都可以支持一个RDFS模型，由于RDFS词汇的使用在RDFS模型上施加额外的限制。

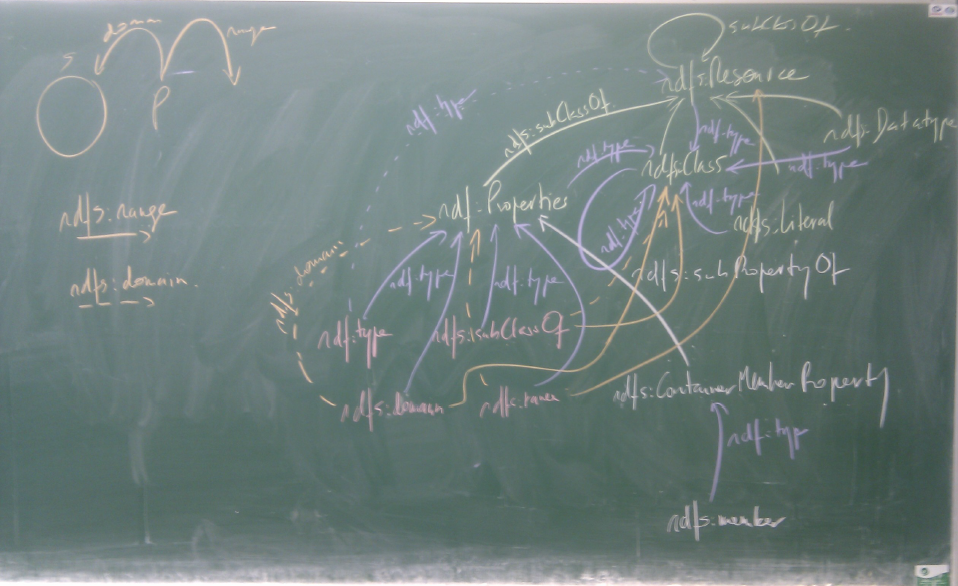


图3.3: 现场演示的部分公理三元组（习题：找到究竟存在多少处错误）.

**定义14** (RDFS 结果). 设和为两个RDFS图，那么 RDFS蕴含（表示为），当且仅当的每个RDFS模型也是的一个RDFS模型。

[HAYES 2004] 指出：“由于每个RDFS解释都是一个RDF解释，如果 RDFS蕴含，那么它就RDF蕴含”。在勘误表（并且在[海因斯和[HAYES and PATEL-SCHNEIDER 2014]）中，给出了逆命题（converse）：

**命题5**. 如果，那么。

**证明**（*Proof*）. 意味着，，；，，但，，并且比起来，不会对其RDFS模型施加更多的限制，因为它被它蕴含了。所以，的所有RDFS模型，都是的RDFS模型。 ⬜

即使是空图RDFS也要比RDF蕴含更多的断言[HAYES 2004]，例如所有三元组的形式：

x rdf:type rdfs:Resource .

在任何含有IRI x词汇的所有RDFS解释中，都为真。

**例 5** (RDFS 语义). 说明为什么例4中图的RDFS蕴含：

geo:Isere rdf:type geo:GeographicArea

因为存在各种RDF模型，所以同样的图并不简单RDF蕴含这个断言，这些RDF模型不是RDFS模型，而且不满足这个三元组（图一）。这对*geo:Grenoble*也适用吗？

**例 6** (RDFS 语义). 考察图的组成:

ex:Pierre rdf:type ex:TennisPlayer

ex:TennisPlayer rdfs:subClassOf foaf:Person

ex:playsWith rdfs:domain ex:TennisPlayer

ex:defeated rdfs:subPropertyOf ex:playsWith

\_:b ex:defeated ex:Pierre

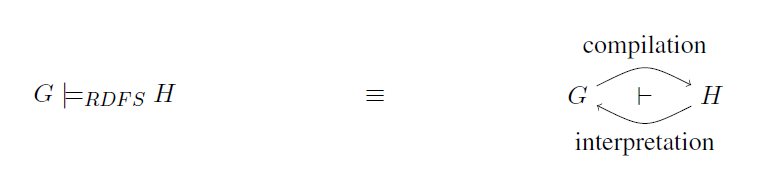
*\_:b rdf:type foaf:Person*，因为没有的一个子图是这个三元组的一个实例。

但是 *\_:b rdf:type foaf:Person*，因为对于的任何模型，〈(\_*:b*),(*ex:Pierre*)〉∈((*ex:defeated*))。依据6c，这意味着，(\_*:b*)∈((*ex:defeated*))。此外，〈(*ex:TennisPlayer*),(*foaf:Person*)〉∈((*rdfs:subClassOf*))。依据5b，我们有((*ex:TennisPlayer*))((*foaf:Person*))。所以，(\_*:b*)∈((*foaf:Person*))。

**3.1.3 ter Horst闭包**

在RDF中，依据发现与之间的一个RDF同态来确定是否是可能的。在RDFS中，这是不容易做到的。

要确定，就必须设计过程。原则上，可以考虑这两种方法：编译（或数据驱动的正向推理）过程，将生成的全部结果，同时确定是否属于它们；评价（或查询驱动或反向链接）过程，从开始，并寻找它是的一个结果的证据。（见下图）。

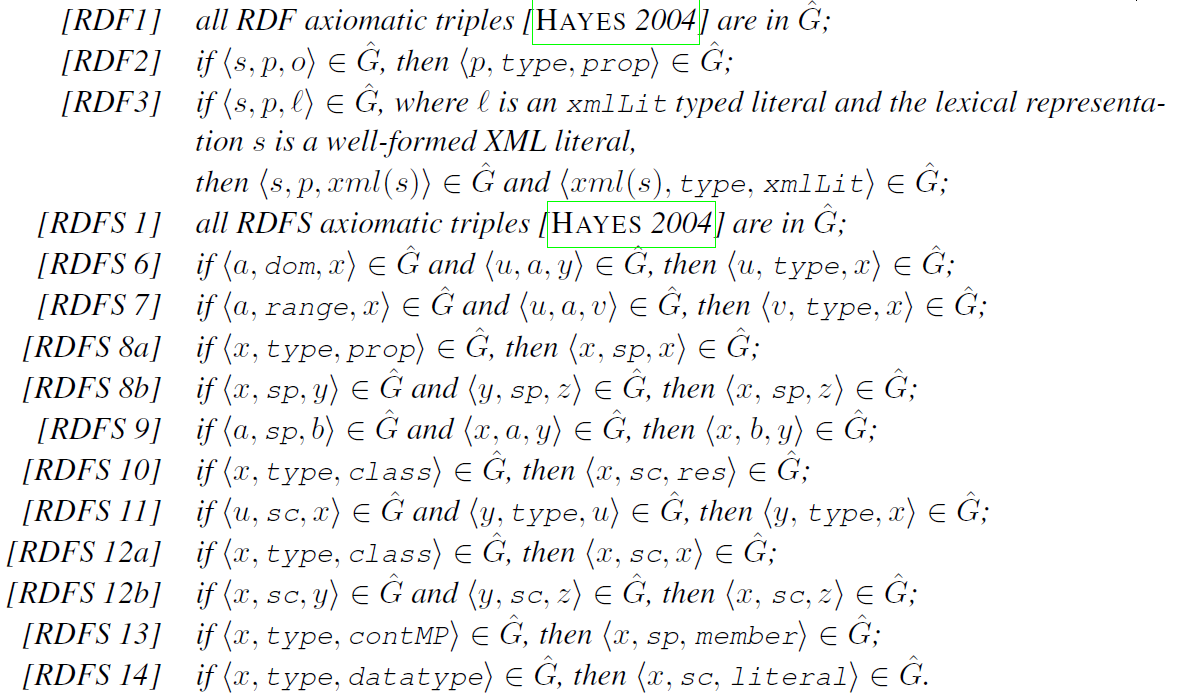


如果图（数据和本体）不经常变化，并且必须快速回答查询，那么编译（compilation）方法是适当的。如果它们不断发展，并且有些延迟是可以接受的，那么解释（interpretation）方法就更适合了。

当然，可能有中间策略（intermediate strategies）：例如，编译与本体相对应的稳定知识，并对编译的本体和数据（不太稳定）的查询进行解释。

一个可能的方法是，查询一个RDFS图的一个健全和完善的方式是，通过计算的封闭图（closure graph），即利用所有信息对进行饱和得到的图，可以使用一组预定义的规则称为RDFS规则进行推断，然后再对该封闭图上的查询进行评价（evaluating）。

**定义15** (RDFS 闭包). 设是在一个RDFS词汇上的一个RDFS图。的RDFS闭包（closure），记为，是包含并满足以下约束的三元组的最小集：



很容易证明这个闭包总是存在的，可以通过将约束变为规则来获得，从而定义闭包操作。

**例7**（RDFS闭包）. 得到例24的RDFS三元组（p.90）增强的例2的RDF图的RDFS闭包（p.19），特别地包含下列断言（assertions）：

dm:bcd rn:inhibits dm:cad. // [RDFS 9]

dm:hb rn:regulates dm:kni. // [RDFS 9]

dm:hb type rn:gene. // [RDFS 6]

由于公理化三元组（axiomatic triples）的缘故，这个闭包可能是无限的，但却是有限多项式闭包（finite and polynomial closure），称为部分闭包（partial closure），已经在[BAGET 2003]和[TER HORST 2005]独立提出。

**定义16** (部分RDFS闭包). 设和是在一个RDFS词汇上的两个RDFS图。给定的的部分RDFS闭包，记为，以下列方式获得：

1. 设*k*是*i*的最大值，即：*rdf:\_i*是或的一个术语（term）；
2. 将规则 [*RDF 1*]，替换成规则

[*RDF 1P*] 除那些使用*rdf:\_ i* 且*i* >*k*的三元组外，加入所有的RDF公理三元组[HAYES 2004]；

同理，将规则[RDFS 1]，替换成规则

[*RDFS 1P*] 除那些使用*rdf:\_ i* 且*i* >*k*的三元组外，加入所有的RDFS公理三元组；

1. 应用修正后的规则。

将部分闭包（partial closure）用于一个RDFS图，就可以将RDFS蕴涵约减为简单RDF蕴涵。

**命题6** (部分RDFS闭包的完整性[HAYES 2004; TER HORST 2005]). 令为一个可满足的RDFS图，且为一个RDFS图，那么，当且仅当。

**不完全的证明**（*Incomplete proof*）. 证明必须依赖于两个论点（arguments）：（1），这留作习题，用于说明关于定义13，每条规则都为真，因此，；（2）是最大化的，即，，，并且更微妙的是，即；，这蕴含着。*only*部分是简单的，每个元组，是依据 RDF蕴含的，那么由于命题5，它是依据 RDF蕴含的。*if*部分必须用于全部模型，因此就更复杂，并在[TER HORST 2005]中具体给出。

限于可能影响的某些结果，因此这个完整性（completeness）是相对于的。如果是不可满足，完整性就不成立，因为在这种情况下，任何图是的一个结果，且并不反映这一点（没有RDF图可以不一致）。一个RDFS图可能是不可满足的，只有当它包含可以在多项式时间（polynomial time）内发现的数据类型冲突[TER HORST 2005]。

这种方法是互调解的（intermediate），因为它对编译图，然后就对评价（通过寻找一个RDF同态）。这似乎会使编译的目的作废，因为它必须重新计算每一个查询。然而，这个编译是达到最高值*n*的一个完整编译，然后它将可用于具有较低的这个*n*的任何图，并且如果它有一个较大的*n*，则可能要对以前的编译进行扩展。

**例8**. 考察由下列三元组组成的图：

a:Book rdfs:subClassOf a:Document .

a:writtenBy rdfs:subPropertyOf a:createdBy .

a:writtenBy rdfs:range a:Writer .

a:createdBy rdfs:range a:Creator .

a:Creator rdfs:subClassOf a:Person .

b:WMLForDummies a:writtenBy b:Pierre .

该图是否RDF蕴含任何一个三元组：

?x rdf:type a:Book . // there exists a book

?x rdf:type a:Person . // there exists a person

a:Writer rdfs:subClassOf a:Creator . // writers are creators

b:Pierre rdf:type a:Person . // Pierre is a person

计算三元组的初始集合的闭包。这四个元组中的哪一个是RDFS蕴含的？

**3.1.4计算性能（Computational properties）**

正如我们定义SIMPLE RDF ENTAILMENT那样（见§2.5），以相同方法，我们可以定义RDFS ENTAILMENT问题：

RDFS ENTAILMENT

**Instance**: two RDFS graphs and .

**Question**: Does?

RDFS ENTAILMENT 是一个NP-完全问题[GUTIERREZ et al. 2004]。对于GRDF图（简单RDF图）， 其复杂性保持不变 [PEREZ et al. 2009]。

**证明提示**（*Hint of proof*）.图匹配（graph matching），是一个NP-完全问题，可以约减到在RDF图中的蕴涵，拥有源图（source graph）的具有不同IRI的所有节点、目标图（target graph）空白的所有节点并且每个边上的谓词相同。 ⬜

基于查询的结构或标记，展示了问题的多项式子类（Polynomial subclasses）：

* 当查询是基础的[TER HORST 2005]，或更一般地，它拥有有限数量的变量，
* 当查询是一颗树或者允许有界分解成一棵树，根据在[BAGET 2005]显示的[GOTTLOB et al. 1999]方法。

**3.1.5 结论（Conclusion）**

完整RDF（Full RDF）用特定词汇表扩展了简单RDF（Simple RDF）。类似地，RDF模式（RDF Schema）用另一个词汇表扩展了RDF。这些词汇表（vocabulary）表示RDF图中实体（entity）的约束（constraint）：它们的类型和结构，对应RDF、属性域上的进一步约束以及类/属性特化（specialisation）。这些约束进一步扩大RDFS解释（interpretation）的定义，进而约减了这些解释，这些解释实际上就是RDF和RDFS图模型。因此，结果（consequence）的概念就不同于Full RDF、RDFS图。

减少蕴涵同态的存在是不可能的了。然而，通过计算蕴含图的一个闭包，就有可能产生一个更大的图，以防止RDFS蕴涵对应于简单RDF 蕴涵（因此同态测试又是完整和正确的）。尽管闭包可能是无限的，但可以计算一个较小的、有限的有用子集，称为部分闭包（partial closure），以防止同态检查（homomorphism checking）又是完整和正确的。

**3.2 Web本体语言OWL**

RDF和RDF Schema允许断言类（classe）和属性（property）之间的关系，例如subClassOf，但不允许从内部构建它们。OWL语言[HORROCKS, PATEL-SCHNEIDER, and VAN HARMELEN 2003; DEAN and SCHREIBER 2004]致力于类和属性定义。受描述逻辑的启发[BAADER et al. 2003]，它提供构造子（constructor）来精确地约束它们。W3C发布了一个新版本的OWL（OWL 2.0）[BECKETT 2009]。

下面，我们展现OWL的语法、语义和不同标准子语言。

**3.2.1 OWL语法**

OWL语法，基于RDF，在owl中引入了特定词汇表：名称空间（name space）。但是，OWL不仅仅是一个词汇表：有许多带有OWL词汇的OWL图，不能用OWL解释。因此，并非所有使用此词汇表的RDF图都必须是有效的OWL本体：必须满足进一步的约束条件。

更合理的做法是，认为OWL有一个语法，用RDF语法来表示，用术语的更经典的含义来解释。OWL 2规定了五种不同的语法[BECKETT 2009]：

**XML/RDF** 是惟一的强制交换语法，它基于XML，具有非常规则但冗长的结构；

**OWL/XML** 一个丑陋但直接的XML语法；

**Functional syntax**（函数语法） 顾名思义，这种语法表示OWL概念的抽象树，更适合于指定其语义（并以归纳方式对其进行操作）；

**Manchester syntax**（曼彻斯特语法） 被认为是一种“用户友好”语法，使用语法糖（syntactic sugar）；

**Turtle** 本身的RDF N3语法的延伸，它也应该是用户更易读的。

很明显，不论它们的名字如何，这些语言的输入不是它们的语法（syntax），而是它们的结构（structure）和语义（semantics）。我们将主要使用XML/RDF和Turtle语法。

特别地，OWL抽象地定义类描述（class descriptions）和属性描述（property descriptions）。这些描述，可以像术语一样，递归地进行定义。

**定义17** (OWL术语). 在OWL中， 术语（terminology）仍然分为文字、IRI 和变量，然而，集合被划分为多个集合：

含有*owl:Thing* and *owl:Nothing*的类名（class names）集合；

含有*rdfs:Literal* 的数据类型（data types）集合；

个体名（individual names）集合；

数据类型属性名（data type property names）集合；

对象属性名（object property names）集合；

在最初的定义中，还有其他类别：

含有rdfs:label, rdfs:comment, rdfs:seeAlso, rdfs:versionInfo和rdfs:isdefinedby的标注性能名称（annotation property names）集合。这些都已经在OWL 2中重新命名；

本体名（ontology names）集合。它们在OWL 2语义中已被删除。

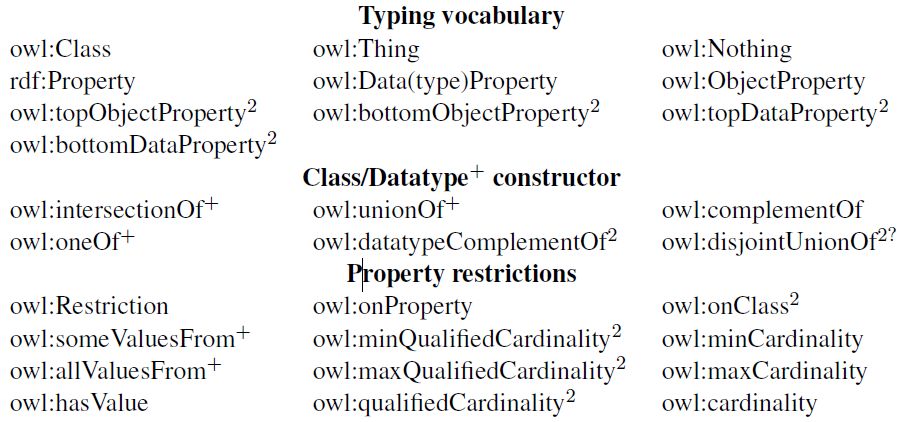
但是，在下面，我们只考虑，，，，，因为它们是唯一有意义语义的。同时，我们也会忽略文档词汇（documentation vocabulary）。

OWL词汇在表3.2中给出遵循用于OWL 的[DEAN and SCHREIBER 2004]以及用于OWL 2 的[BAO et al. 2009]。我们只覆盖主要术语，杂项术语在此不予考虑。OWL与OWL 2之间的一个主要区别，是数据类型的一致性处理（uniform treatment），在此也轻易地得到覆盖。

主要OWL构造子（constructors）的直观语义如下所示：

* RDF 关键字 (rdf:type, rdf:Property) 以及 RDFS的关键字 (rdfs:subClassOf, rdfs:subPropertyOf, rdfs:range, rdfs:domain) 都使用相同的语义。
* owl:Class 是一个新的（元）类。
* owl:sameAs 和 owl:differentFrom 被用于断言两个资源相等或不同。
* owl:inverseOf 断言属性是属性的逆 （在这种情况下，三元组蕴含；还可以给属性分配其他特征，如自反性reflexivity (owl:ReflexiveProperty)，传递性transitivity (owl:TransitiveProperty)，对称性symmetry (owl:SymmetricProperty) 或者功能性functionality (owl:FunctionalProperty)。
* owl:allValuesFrom将一个类关联到一个关系。这就定义了对象的类，那么如果有效，则属于类（这是在描述逻辑中的一个普遍量化作用）。owl:someValuesFrom编码存在量化作用。
* owl:minCardinality（分别地owl:maxCardinality）允许用一个基数（cardinality），过一个给定的属性，来定义相关对象的类至少（或至多）通关联于一个特定数量的对象。此外，这些构造子的限定版本，还强制这些对象属于某个特定的类。
* owl:oneOf 通过枚举实例的集合在内涵上定义了一个类。
* owl:hasValue 强制一个属性拥有一个特定的个体作为值。
* owl:disjointWith 断言两个类不能有一个共同的实例。
* owl:unionOf，owl:intersectionOf 和 owl:complementOf定义一个类为其他类的析取（disjunction）、合取（conjunction）或补（negation）。
* owl:hasSelf通过特定关系定义与自己相关的对象类。
* owl:hasKey断言一组属性是一个类的关键字，即对这些属性而言，两个不同的实例不能共享相同的值。
* owl:propertyChainAxiom将若干关系和属性进行组合以获得新的关系或属性。

我们没有提到所有构造子（constructors）。它们中的许多，可以通过使用引用的构造子得以平凡实现，如owl:equivalentClass断言两个类等效，可以使用两个rdfs:subClassOf断言来表达。而且，OWL也使用不在这里考虑的数据类型。虽然它们是重要的，但是它们可能会导致不一致。



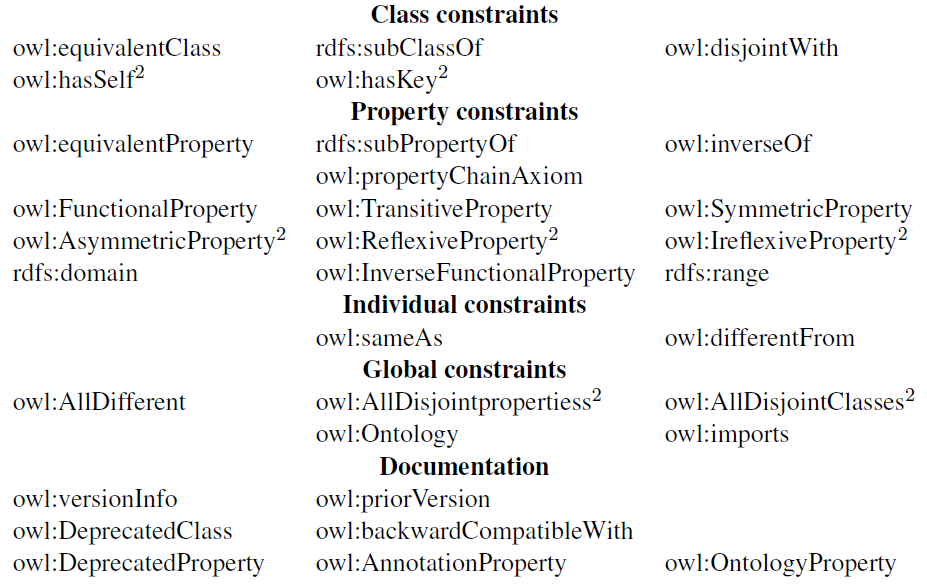


表3.2: OWL 词汇. OWL 2原语被接以一个指数”2” ，适用于数据类型的原语依据OWL 2被标以指数”+”

**例9**. 以下的例子：

*ChemistryProfessor* *Professor* *teaches.ChemistryLecture*

*teaches* =

*teaches.Lecture* .*Professor*

可以用OWL表达为

<owl:Class rdf:about="#ChemistryProfessor">

<rdfs:subClassOf>

<owl:Class>

<owl:intersectionOf rdf:parseType="Collection">

<owl:Class rdf:about="#Professor" />

<owl:Restriction>

<owl:onProperty rdf:resource="#teaches" />

<owl:allValuesFrom rdf:resource="#ChemistryLecture" />

</owl:Restriction>

</owl:intersectionOf>

</owl:Class>

</rdfs:subClassOf>

</owl:Class>

<owl:ObjectProperty rdf:about="#teaches">

<owl:inverseOf rdf:resource="#taughtBy" />

<rdfs:domain rdf:resource="#Professor" />

<rdfs:range rdf:resource="#Lecture" />

</owl:ObjectProperty>

值得注意的是，最后一个陈述句是rdfs:domain和rdfs:range的一个编码。特别是，定义了OWL的语义，从而使这个语句表明：

= () (D)

同样可以用三元组的可怕语法来表达：

ex:ChemistryProfessor rdf:type owl:Class

ex:ChemistryProfessor rdfs:subClassOf \_:b1

\_:b1 rdf:type owl:Class

\_:b1 owl:intersectionOf \_:b2

\_:b2 rdf:type rdf:List

\_:b2 rdf:\_1 ex:Professor

ex:Professor rdf:type owl:Class

\_:b2 rdf:\_2 \_:b3

\_:b3 rdf:type owl:Restriction

\_:b3 owl:onProperty ex:teaches

\_:b3 owl:allValuesFrom ex:ChemistryLecture

ex:teaches rdf:type owl:ObjectProperty.

ex:teaches owl:inverseOf ex:taughtBy

ex:teaches rdfs:domain ex:Professor

ex:teaches rdfs:range ex:Lecture

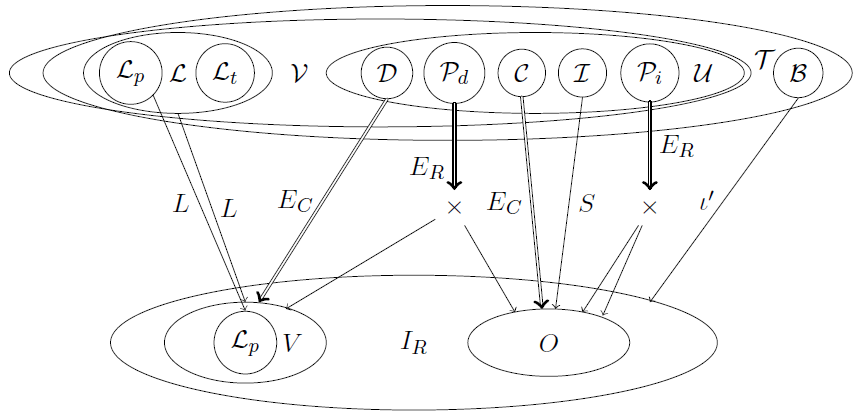


图3.4: OWL语义的域结构

**3.2.2 OWL 语义**

OWL构造子的语义，在[PATEL-SCHNEIDER et al. 2004; MOTIK, PATEL-SCHNEIDER, et al. 2009]中给出，遵循描述逻辑风格。

**定义18** (OWL 解释). 在词汇上的一个OWL解释，是一个元组，那么：

* ；
* ；
* ；
* ；
* （加额外）；
* ；
* ；
* ；
* ；
* （加额外）；
* （加额外）；
* ；
* 更多关于数据类型

根据经典描述逻辑语义学解释复合表达式（compound expressions）。

**定义19** (复合表达式的解释). 令为在词汇上的一个OWL解释，依据对复合类表达式的解释，依据以下进行定义（所有关键字都在OWL命名空间中）：

*EC*(*owl:Thing*) = *O*

*EC* (*owl:Nothing*) = ∅

*EC* (*rdfs:Literal*) = *V*

*EC* (*complementOf*(*c*)) = *O* *EC* (*c*)

*EC* (*unionOf*(*c*, *c*’)) = *EC* (*c*) *EC* (*c*’)

*EC* (*intersectionOf*(*c*, *c*’)) = *EC* (*c*) *EC* (*c*’)

*EC* (*oneOf*(*i*1, …, *in*)) ={*S*(*i*1), …, *S*(*in*)}

*EC* (*oneOf*(*v*1, …, *vn*)) ={*L*(*i*1), …, *L*(*in*)}

*EC* (*restriction*(*p*, *allValuesFrom*(*r*))) = {; (*p*), *EC* (*r*)}

*EC* (*restriction*(*p*, *someValueFrom*(*r*))) ={; (*p*)*EC* (*r*)}

*EC* (*restriction*(*p*, *hasValue*(*v*))) = {; *EC* (*p*)}

*EC* (*restriction*(*p*, *hasValue*(*i*))) = {; *EC* (*p*)}

*EC* (*restriction*(*p*, *minCardinality* (*n*))) ={;}

*EC* (*restriction*(*p*, *maxCardinality* (*n*))) ={;}

*EC* (*restriction*(*p*, *cardinality* (*n*))) ={;}

*EC* (*restriction*(*p*, *minQCardinality* (*n*, *C*))) =

*EC* (*restriction*(*p*, *maxQCardinality* (*n*, *C*))) =

*EC* (*restriction*(*p*, *qCardinality* (*n*, *C*))) =

(*inverseOf*(*p*))

(*propertyChainAxiom*)

与描述逻辑（description logics）一样，它也可以解释其他类型的复合表达式（如属性路径表达式）。

**定义20** (公理满足Axiom satisfaction). 令为在词汇上的一个OWL解释，一个公理（axiom）被称为满足（所有关键字都在OWL命名空间中）：

*equivalentClass*(*c*, *c*’) ***iff*** (*c*) = (*c*’)

*subClassOf* (*c*, *c*’) ***iff*** (*c*)(*c*’)

*disjointWith*(*c*, *c*’) ***iff*** (*c*)(*c*’) =∅

*equivalentProperty*(, ) ***iff*** =

*subPropertyOf*(, ) ***iff***

*sameAs*(, ) ***iff*** () =()

*differentFrom*(, ) ***iff*** () ()

*AllDifferent* ***iff*** , [ ], () ()

*SymmetricProperty* ***iff***

*TransitiveProperty* ***iff***

*FunctionalProperty* ***iff***

*InverseFunctionalProperty* ***iff***

一个模型通常被定义为满足所有公理的解释。它也满足每个术语类别所引起的限制。

**定义21** (OWL 模型). 在词汇上的一个OWL解释，是某个OWL本体的一个OWL模型，当且仅当（***iff***）：

* 用于指定中的类位置的每个IRI（分别为：数据类型的属性datatype property，对象属性object property，个体individual，数据类型datatype），属于(分别为, )；
* 中的每个文字（literal）属于；
* 存在到变量（空白）的一个扩展，使得满足中的所有公理。

结果（consequence），不一致（inconsistence）等，像往常一样定义。

**例10.** 这是用三元组集合的可怕语法形式表达的OWL本体的一个实例：

ex:Grenoble rdf:type geo:City

geo:City rdfs:subClassOf geo:GeographicArea

ex:is-contained-in owl:inverseOf ex:contains

geo:City rdfs:subClassOf \_:x

\_:x rdf:type owl:Restriction

\_:x owl:onProperty ex:is-contained-in

\_:x owl:maxCardinality "1"ˆˆxsd:Integer

geo:City rdfs:subClassOf \_:y

\_:y rdf:type owl:Restriction

\_:y owl:onProperty ex:country

\_:y owl:someValuesFrom geo:Country

它蕴含以下三元组集合：

ex:Grenoble rdf:type geo:GeographicArea

ex:Isere ex:contains ex:Grenoble

ex:Grenoble ex:country \_:z

\_:z rdf:type ex:Country

此外，如果一个增加：

ex:Grenoble ex:country ex:France

ex:country rdfs:subPropertyOf ex:is-contained-in

这就蕴含：

ex:France owl:sameAs ex:Isere

这显然是错误的。那么问题是什么呢？

**例11.** 考察由下列陈述句（statement）组成的本体：

OWL语义使：因为对任何模型，成立

因此，在所有模型中都为真。

*:b rdf:type ex:ChemistryLecture*不为真。然而，如果一个模型加入陈述句*ex:Paul rdf:type ex:ChemistryProfessor*，那么*:b rdf:type ex:ChemistryLecture*就成立。

**3.2.3 OWL的表方法**（Tableau method for OWL）

OWL一般也具有无限闭包（例如，最大基数限制蕴含所有更大整数的最大基数限制），但已蕴含的语句的模式更宽泛。因此，不建议使用闭包。

OWL中的推理（inference），通常是由专用的描述逻辑证明器（dedicated description logic provers）基于tableau方法、解析（resolution）或其他转换（transformation）来提供。Tableau方法（或语义tableau方法）是一种试图为一组公式创建一个模型的方法[BAADER et al. 2003]。为此，它使用规则（rule）来探索可能模型空间，这些规则保证了探索是详尽的并且符合语言的语义。

在描述逻辑（description logics）中，表推理（tableau reasoning）通常以反驳（refutation）的方式进行。为了证明，而不是去研究的各种模型，检查在所有这些模型中都成立，而是先从与的一个否定编码（）开始，并尝试建立一个模型满足。如果找到了这样一个模型，就意味着，存在的一个模型不满足，所以不由蕴含。如果没有发现这样的模型，那么就意味着在的所有模型中成立，因此由蕴含。

要做到这一点，逻辑必须是完全否定的，也就是说，语言的任何公式的否定，都可以用该语言的公式来表达。

A tableau procedure has been designed for most of the OWL language. We provide below an example of tableau for the ALC language which is a small subset of OWL (see [HORROCKS and SATTLER 2007] for a fuller treatment). An ALC TBox is a set of general concept inclusion axioms of the form C⊆C’. Concepts are defined by:

针对大多数OWL语言，都设计了一个表过程（tableau procedure）。以下，我们提供语言的一个tableau实例，这是OWL的一个小子集（为了更全面的处理，参见[HORROCKS and SATTLER 2007]）。一个 TBox是形式的一个一般概念包含公理（general concept inclusion axioms）的集合。概念被定义为：

所有角色（role）都仅仅是原子角色（atomic role）（）。

用于的Tableaux方法，分割公式中正确的本体内容（ontological content）----大致地，OWL部分----作为TBox ，同时公式中断言内容（assertional content）---- RDF部分----作为一个ABox 。此外，ABox必须表示为正常形式的否定（negation normal form），即否定（negation）只适用于原子类（atomic classes）。它的进行，就是通过应用规则来建立模型。这种模型，可以被认为是RDF图：它们是具有由角色（role）标记的边（edge）和由节点所属的类表达式集（）标记的节点所构成的标记图（labelled graphs）。正是这些规则，形成了可能模型的一个树。某些特定规则，称为冲突（clashes），表示一个不可能模型（impossible model），并因此关闭树的分支。如果满足这样的冲突（clash），那么当前的表示就不能变成一个模型，并且算法必须探索最终替代表示（eventual alternative representations）。一种特殊的技术，称为阻塞（blocking），是为了避免因无限扩展规则而产生无限的分支。的规则和冲突，在表3.3中提供。所有规则一直使用到没有规则适用为止。如果有一个不包含冲突的分支，那么就建立了一个模型。

表3.3 用于的完整性规则与冲突条件

|  |  |
| --- | --- |
| 规则 | 含义 |
| -rule | Condition: ，没有被阻止；  Action: |
| - rule | Condition: ，没有被阻止；  Action: ，或 |
| -规则 | Condition: ，没有被阻止；；  Action: 用和，创建一个新节点 |
| - rule | Condition: ，没有被阻止；；  Action: |
| - rule | Condition: ，没有被阻止；  Action: |
| Clash-rules | -clash: ； |
| -clash: ； |

**3.2.4 OWL 2 与配置文件**（profiles）

最初的OWL规范，定义了三种不同的语言（OWL，OWL DL和OWL Lite）。幸运的是，人们意识到描述逻辑的框架已经足够丰富，可以容纳更多的语言。在这里，我们考虑OWL规范的状态。此外，关于许多小的变化，OWL 2介绍了配置文件（profile）的概念[MOTIK, CUENCA GRAU, et al. 2009]，让人联想到描述OWL的一个特定子集的描述逻辑模块化（description logic modularity）。这些覆盖以前OWL子语言的配置文件，引入了三个新的OWL 2配置文件。一个配置文件的特点是它使用的词汇和它接受的图的语法形式。

这些配置文件是：

**OWL Lite**

* 具有分立的属性和类的论域（domains of discourse）；
* 包含所有RDF构造子，也就是说，它允许声明一个个体作为类的成员，并通过属性关联个体和数据值；
* 采用部分RDFS词汇（rdfs:subClassOf, rdfs:subPropertyOf, rdfs:range, rdfs:domain），具有相同的语义；
* 使新类（owl:Class）的定义更为具体或等价于其他类的合取（conjunction）；
* owl:sameIndividualAs 与 owl:differentIndividualFrom断言两个个体是等价的或不同的；
* 属性特点可以通过owl:inverseOf断言，陈述属性是另一个的逆（converse）（因此，三元组蕴含）；其它特征是传递性（owl:TransitiveProperty）、功能性（owl:FunctionnalProperty）、对称性（owl:SymmetricProperty）或逆功能性（owl:InverseFunctionalProperty）；
* owl:allValuesFrom定义对象类的属性的全部（）值都属于类；owl:someValuesFrom定义对象类存在（）属性的一个值属于类；
* owl:minCardinality（或。owl:maxCardinality）定义对象类至少（或至多）对属性取个值。在OWL Lite中，只能是0或1。

OWL Lite对应的描述逻辑（概念合取，析取和否定+全称角色限制+量化存在角色限制+传递角色+角色层次+逆角色+函数角色+数据类型）。蕴涵可以在EXPTIME[HORROCKS, PATEL-SCHNEIDER, and VAN HARMELEN 2003]中决定，并且存在许多解决该问题的有效算法。

**OWL 2 RL** 是OWL Full的一个片段，可以使用规则语言实现。

**OWL 2 QL**大致相当于DL-Lite（§3.2.5）；

**OWL DL**包含所有的构造子，对这些构造子使用特定约束，保证蕴涵测试的可判定性（decidability）。所以，OWL DL：

* 包含所有OWLLite构造子；
* 在基数约束上允许任何正整数，但在传递属性或其逆或其任何子属性上，对它们予以禁止；
* owl:oneOf通过类的实例来定义一个类，
* owl:hasValue定义对象类对某个属性仅具有特定值；
* owl:disjointWith断言两个类没有共同实例；
* owl:unionOf 和 owl:complementOf允许类的定义为两个类的并（union）或一个类的补（complement）。

OWL DL对应描述逻辑（+ nominals）。描述逻辑SHOIN（D）的蕴涵测试具有不确定性指数时间复杂度（NEXPTIME）[HORROCKS, PATEL-SCHNEIDER, and VAN HARMELEN 2003]。到目前为止，还没有一个完整的算法可以解决这个问题。

**OWL 2 EL**保留类定义和局限于存在量化的属性中的表达能力（它放弃了通用量化、基数限制、析取、否定和关系属性）；

**OWL Full**不需要任何约束使用所有构造子。由于它的非标准语义，它不符合一个很好识别的描述逻辑类。因为它包含了否定和集可以包含定义集的定义，它最有可能是不可判定的。

* 包含所有OWL DL构造子；
* 包含所有RDF Schema构造子；
* 允许使用类代替构造子中的个体。

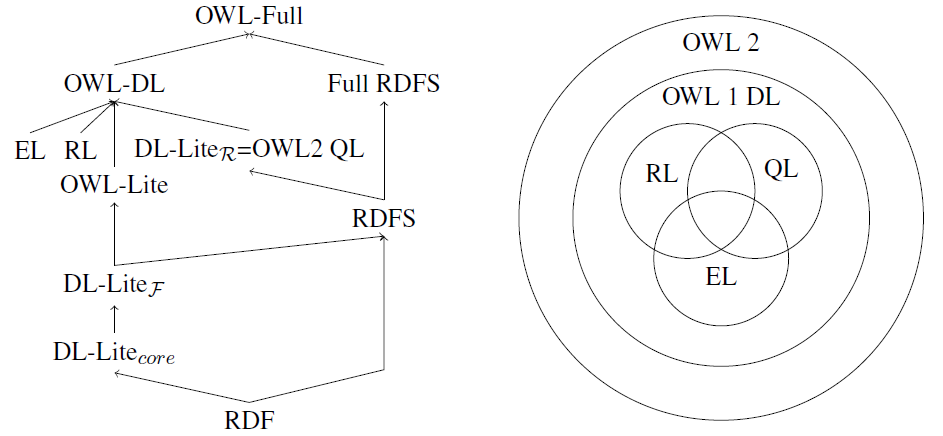


图3.5：OWL语言的层次结构

推理机（Inference engines）已经实现了OWL DL（特别是Pellet）的重要子集。此外，在OWL语言族中，仍然可以找到令人高兴的地方。其中之一，就是DL Lite，如下所示。

**3.2.5 DL-Lite一个属于自己的子族**

DL-Lite [CALVANESE, DE GIACOMO, LEMBO, et al. 2007] 是一个描述逻辑的家族，它被设计用来支持有效（多项式）合取查询应答。它是基于OWL语言的表达约减。这是通过选择子语言的查询评估可以约减为简单的关系数据库查询的实现。

**定义22** (DL-Lite 语法). DL-Lite术语（terms）基于以下语法：

其中，和是IRI，分别识别概念（concept）和属性（property）。*DL*-语言包含形式的断言，此外，*DL*-像与一样对公理进行授权，*DL*-添加了功能性公理（functionality axioms）。

这相当于使用以下OWL构造子： rdfs:subClassOf，owl:minCardinality (具有值1)， owl:Thing， owl:Nothing，owl:complementOf，owl:inverseOf，对其添加了一个属性的补（complement）。这种简单语言允许形如（等价于并且）表达断言。

事实上，*DL*-是OWL Lite和RDFS的严格子集（它不能表达rdfs:range或owl:allValuesFrom），显然*DL*-是RDFS（的描述逻辑部分）一个严格超集\*。

---------------------------------------------------------------------------

\* 同一作者介绍了*DL*-作为*DL*-和*DL*-的并（union）以及*DL*-作为 *DL*-的一个子集，做出进一步限制，诸如角色（role）和特征（feature）[POGGI et al. 2008]之间的分离。

-------------------------------------------------------------------------------------------------------------------------------------

**例12**. 在Marie-Christine Rousset的演示文稿[ABITEBOUL et al. 2011]中给出的实例：

*DL*-

*DL*-

可以用OWL表示：

<owl:Class rdf:about="#Professor">

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#teachesTo" />

<owl:minCardinality>1</owl:minCardinality>

</owl:Restriction>

</rdfs:subClassOf>

<owl:disjointWith rdf:resource="#Student" />

</owl:Class>

<owl:Class rdf:about="#Student">

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasTutor" />

<owl:someValuesFrom rdf:resource="&owl;Thing" />

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

<owl:Class>

<owl:equivalentClass>

<owl:Restriction>

<owl:onProperty>

<owl:ObjectProperty>

<owl:inverseOf rdf:resource="#teachesTo" />

</owl:ObjectProperty>

</owl:onProperty>

<owl:someValuesFrom rdf:resource="&owl;Thing">

</owl:Restriction>

</owl:equivalentClass>

<rdfs:subClassOf rdf:resource="#Student" />

</owl:Class>

<owl:Class>

<owl:equivalentClass>

<owl:Restriction>

<owl:onProperty>

<owl:Property>

<owl:inverseOf rdf:resource="#hasTutor" />

</owl:Property>

</owl:onProperty>

<owl:someValuesFrom rdf:resource="&owl;Thing">

</owl:Restriction>

</owl:equivalentClass>

<rdfs:subClassOf rdf:resource="#Professor" />

</owl:Class>

<owl:Property rdf:about="#hasTutor">

<rdfs:subPropertyOf>

<owl:ObjectProperty>

<owl:inverseOf rdf:resource="#teachesTo" />

</owl:ObjectProperty>

</rdfs:subPropertyOf>

</owl:Propert>

<owl:FunctionalProperty rdf:about="#hasTutor" />

hastutor的定义是正确的，因为如果，那么（证明它，看看它是如何影响我们的定义）。

你也可以试着将这个片段表达为一个三元组集合，并检查这个描述逻辑片段的语义和相同的OWL语义。

**3.2.6 计算结果（Computational results）**

表3.4总结了从[MOTIK, CUENCA GRAU, et al. 2009]和[CALVANESE, DE GIACOMO, LEMBO, et al. 2007]关于公理蕴涵问题的复杂性（一致性，类可满足性，类包含，实例检测）和合取查询应答。

表3.4：最坏情况下OWL复杂问题的组合复杂性结果

|  |  |  |
| --- | --- | --- |
| 问题 | 公理蕴含 | 合取查询 |
| OWL Full | 不可判定的 | 不可判定的 |
| OWL 2 | 2NEXPTIME-complete | 开放的 |
| OWL 1 DL | NEXPTIME-complete | 开放的(可判定的) |
| OWL 1 Lite | EXPTIME-complete |  |
| DF | NP-complete, PTIME (ground) |  |
| OWL 2 EL | PTIME-complete | PSPACE-complete |
| OWL 2 RL | PTIME-complete | NP-complete |
| OWL 2 QL | NLOGSPACE-complete | NP-complete (UCQ), AC0 (CQ) |
| *DL*- | PTIME | NP-complete (UCQ) |
| *DL*- | PTIME |  |

因为我们保留了组合复杂性（combined complexity），所以特定逻辑的选择，也可能取决于要处理的问题的主要因素（数据的大小、查询的大小或本体的大小）。当然，这个选择应该主要取决于要表达什么。

**3.3结论（Conclusion）**

本体语言允许定义RDF图中使用的词汇，并限制使用此词汇表。由于这些限制，可以对RDF图进行推理并推断所包含的信息。在推理中实现完整性（completeness），即找到全部和唯一的条件并不一定是一件容易的事。因此，定义了不同表达能力的不同本体语言。为了表达一个本体（ontology），必须选择最合适的语言。

尽管RDF模式允许定义简单的本体，但它不允许强大的约束，例如通过析取（disjunction）或补（complement）定义类或通过类型（type）或基数（cardinality）约束它们的属性。OWL语言引入更多的词汇（和结构约束）来定义更具表现力的本体。这进一步扩展了这些语言的语义结构。

当查询RDF图时，可以使用蕴含（entailment），即，当将本体与RDF图连接在一起时，不只查询该图（graph），而且查询它蕴含着什么。这将在本文档的其余部分加以考虑。

**3.4 习题**

**习题8 (OWL 本体).**

1. 用OWL（RDF或XML / RDF）描述包含以下断言的本体：

* *All authors are persons;*
* *A book (m:Livre) has exactly one year of publication (m:annee);*
* *A novel (m:Roman) is a book (m:Livre) and a book is a work (m:Oeuvre);*
* *The title (dc:title) of a work is a character string (xsd:string);*
* *The relation ”a ecrit” (m:aecrit) relates an author to a work.*

1. 如果一个图关联了习题1中图2.6（p26）的图(a)及根据以前问题造成的本体，那么它可能推断出*?x*的类型(*rdf:type*)吗？

你能在语义上证明怎么做吗？还可以推断出什么呢？

1. 定义图2.6图（p26）使用这个本体是否会对回答习题1中的问题4改变些什么？是什么呢？

**习题9** (DL-Lite 本体). 考虑由以下DL-Lite断言组成的本体论：

worksWith foaf:knows

playsTennisWith playsSportWith

playsSportWith foaf:knows

marriedWith foaf:knows

Person foaf:mbox

1. 其中，这个本体所表示的是哪个DL-Lite的专业用语（子语言）（*DL*-，*DL*-，*DL*-）？
2. 将本体重写为OWL或RDFS形式。
3. 其中，你的本体所表示的专业用语（语言）是RDFS还是OWL？
4. 给定习题2(p27)中图2.7(b)中的图，向其增加公理。计算其闭包。与部分闭包有什么区别？
5. 给定图2.7(p27)中的图，向其增加公理，如果我们考虑RDFS蕴涵，这会对习题2中的问题3的解答带来哪些变化呢？（解释原因）

**习题10** (OWL 本体).

1. 提供由习题2（P27）的图2.7中的两个图所满足的本体。

我们想表达的是，依据属性foaf:mbox，一个电子邮件地址不能对应于一个以上的人。

1. 如何在OWL中表示：基数约束、函数属性、逆函数属性或对称属性？
2. 可以在DL-Lite中表达这个约束吗？如果是，如何做？
3. 考虑向图2.7（P27）中的图添加此约束，这是否改变了习题2中的问题3的答案？（解释原因）

**习题 11** (RDFS 本体). 考虑RDFS本体，除了包括习题3中的图G的那些陈述句，还包括下面的陈述句：

〈o:Novel, rdfs:subClassOf, o:Literature〉

〈o:Poem, rdfs:subClassOf, o:Literature〉

〈o:translated, rdfs:range, o:Literature〉

〈o:wrote, rdfs:domain, o:Writer〉

1. 这是否可以得出这样的结论：d:Poe，d:Baudelaire或d:Mallarme是o:Writer？解释为什么。
2. 你能用OWL来表达“任何写文学的人都是作家（anyone who write Literature is a Writer）”的陈述句吗？

**习题12** (OWL 2 受限的基数约束). OWL 2 引入了受限的基数约束（qualified cardinality restrictions），owl:qualifiedCardinality、 owl:maxQualifiedCardinality以及owl:minQualifiedCardinality，其解释是通过扩展定义19的函数获得的：

(restriction(, minQualifiedCardinality(, )))

(restriction(, maxQualifiedCardinality(, )))

(restriction(, qualifiedCardinality(, )))

考虑以下表达式（在OWL 2中）：

ex:SmallTeam rdfs:subClassOf :a .

\_:a rdf:type owl:Restriction .

\_:a owl:onProperty ex:member .

\_:a owl:maxCardinality 5 .

ex:ModernTeam2 rdfs:subClassOf ex:SmallTeam .

ex:ModernTeam2 rdfs:subClassOf :b .

\_:b rdf:type owl:Restriction .

\_:b owl:onProperty ex:member .

\_:b owl:minQualifiedCardinality 4 .

\_:b owl:onClass ex:Woman .

1. 画出这三元组集对应的图。
2. 用OWL/XML表示它。
3. 解释这个图的意思（用英语意译）
4. 如果我们交换5和4，会发生什么？

习题13 (从OWL 2 到OWL 1 并反过来).

1. 如何才能按最小和最大的受限基数约束的函数形式重写qualifiedCardinality？用语义解释它。
2. 是否可能表达minCardinality、maxCardinality、cardinality及someValuesFrom这些新的受限基数约束？如何解释。

除了前面的RDF图，考虑以下陈述句（用OWL 1表示的）：

ex:womanmember owl:subPropertyOf ex:member .

ex:womanmember rdfs:range ex:Woman .

ex:ModernTeam1 rdfs:subClassOf ex:SmallTeam .

ex:ModernTeam1 rdfs:subClassOf :b .

\_:b rdf:type owl:Restriction .

\_:b owl:onProperty ex:womanmember .

\_:b owl:minCardinality 4 .

1. ex:ModernTeam1包含ex:ModernTeam2或者反过来？证明。
2. 这是否表明，在OWL 1中，也可以表示受限基数约束？解释。
3. 受限基数约束对OWL 1提供额外的表达性（additional expressivity）吗？

**习题14** (RDFS 中的FRBR). *FRBR*现在是类库（libraries）中一个行之有效的词汇（vocabulary）。FRBR的区别：

1. 是关于作品是什么的一个抽象概念的工作；
2. 以特定的形式（诗歌、音乐、绘画）实现一个作品的表达；
3. 表现形式（manifestation），是一种表达（一个文本的版本，一部电影的发行；一幅绘画的复制）的清楚体现；
4. 一个项（item）通常是一个表现形式的物理样本（一本书的范例；一个mp3文件的副本）。

考虑其RDF模式显式的片段：

frbr:Work rdf:type rdfs:Class .

frbr:Expression rdf:type rdfs:Class .

frbr:Manifestation rdf:type rdfs:Class .

frbr:Item rdf:type rdfs:Class .

frbr:Performance rdfs:subClassOf frbr:Expression .

frbr:adaptation rdfs:domain frbr:Work .

frbr:adaptation rdfs:range frbr:Work .

frbr:realization rdfs:domain frbr:Work .

frbr:realization rdfs:range frbr:Expression .

frbr:translation rdfs:domain frbr:Expression .

frbr:translation rdfs:range frbr:Expression .

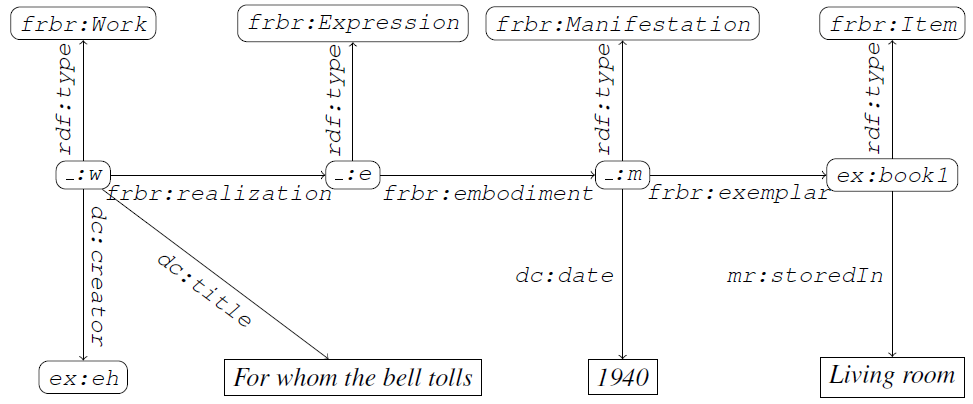
frbr:embodiment rdfs:domain frbr:Expression .

frbr:embodiment rdfs:range frbr:Manifestation .

frbr:exemplar rdfs:domain frbr:Manifestation .

frbr:exemplar rdfs:range frbr:Item .

例如，习题4中图G的前5个陈述句，以FRBR形式，对应下面的图：



1. 你如何用习题4中的图中的前缀mr确定的词汇来扩展上面的？即给出三元组，用RDFS词汇（rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, rdfs:range）扩展FRBR与对应的实体。

**习题15** (RDFS 蕴含). 给定={*?x* rdf:type a:Hospital. *?x* a:hasService *?y*. *?y* rdf:type a:Maternity. *?x* a:hasService *?z*. *?z* rdf:type a:Pediatry.} 与图2.8（p.29）的图 ，

1. 是否？（解释如何以及/或为什么）
2. 哪个过程可以用来推导出？

**习题16** (RDFS 与 OWL 解释).

1. 解释两个异构源（heterogeneous sources）的一个方便的方法是，通过一个共同的本体（common ontology）来解释它们。考虑以下陈述句构成的本体：

o:parent rdfs:domain foaf:Person .

o:parent rdfs:range foaf:Person .

o1:mother rdfs:subPropertyOf o:parent .

o1:father rdfs:subPropertyOf o:parent .

o1:mother rdfs:domain o2:Female .

o1:father rdfs:domain o2:Male .

考察习题6（p.28）中的图1, , 及，是否成立？

1. 是否成立？（解释你的答案）给所有的映射（变量/空白指派）支持这一蕴涵。RDFS-蕴含什么附加事实？（提供一个例子）
2. 你能以OWL形式，用本体的概念和属性，将类o:ParentOfNumerousChildren，表示为拥有三个以上孩子的那些父母的类？给出这个（复合）类的解释。

**习题17** (RDFS 与 OWL蕴含). 考虑由下列陈述句构成的本体：

o:accomodation rdfs:range o:Accomodation .

o:Local rdfs:subClassOf o:Accomodation .

o:Pousada rdfs:subClassOf o:Local .

o:Ryokan rdfs:subClassOf o:Local .

o:GrandHotel rdfs:subClassOf Accomodation .

o:activity rdfs:range o:Activity .

o:Sport rdfs:subClassOf o:Activity .

o:Swimming rdfs:subClassOf o:Sport .

o:SwordFighting rdfs:subClassOf o:Sport .

o:Visit rdfs:subClassOf o:Activity .

o:Cruising rdfs:subClassOf o:Visit .

以及习题7中的图

1. o:Package rdf:type rdfs:Class？

o:Package rdf:type rdfs:Class？

1. ？？解释原因。
2. 给定OWL公理（确定OWL本体）：

o:TonicPackage o:Package

o:accomodation:(o:Localu o:swimmingPool)

o:activity:o:Sport

给出TonicPackage ((o:TonicPackage))的OWL解释。

1. *\_:b1 rdf:type o:TonicPackage*？解释原因。

**规则（Rules）**

关于规则，情况并不比其他语义Web语言更清楚。W3C建议的“规则交换格式”（rule interchange format, RIF）”（RIF [BOLEY et al. 2013]），RIF的目标是提供逻辑编程规则（logic programming rules）和“业务规则（business rules）”，这是在满足条件时触发一个动作的一种规则。两种专业用语（dialect），分享他们的语法部分（RIF-Core），已被定义为：BLD“基本逻辑专业用语（Basic Logic Dialect）”和PRD“生产式规则专业用语（Production Rule Dialect）”。RIF有一个不同于其他语言提供的语法。特别是，对RuleML和SWRL的RDF语法[HORROCKS, PATEL-SCHNEIDER, BOLEY, et al. 2004]没有保留RIF。

从理论的角度来看，以逻辑编程方式开发的所有工作都自然地应用于语义Web，并且可以用RIF-BLD专业用语进行换位。此外，规则和本体之间的联系已具体规定[DE BRUIJN and WELTY 2013]。

结合描述逻辑（description logics）和Datalog（数据记录）进行了研究[LEVY and ROUSSET 1998]。在语义网的情况下，工作从缺乏表现力的片段（fragments）开始：DLP（描述逻辑程序[GROSOF et al. 2003]）是一种结合霍恩子句（Horn clause）和描述逻辑的最小语言。因此，DLP不允许既不表达所有OWL Lite，也不表达所有逻辑编程。然而，它却可以被用于大规模的推理。DLP一直是OWL 2 RL的灵感来源。

一个关于描述逻辑和关于“应答集编程（answer set programming）” 的紧密集成，这再次提及诸如否定即失败（negation-as-failure）和封闭世界假设（closed world assumption）方面，也被 [EITER et al. 2008; MOTIK and ROSATI 2010] 考虑到。其他工作，都试图在逻辑编程语言的基础上定义描述逻辑[HUSTADT et al. 2007; HUSTADT et al. 2008]。

从概念图（conceptual graph）方面，将逻辑规则作为简单概念图[BAGET et al. 2011]已经进行了研究，因为这些形式接近，所以它可以直接应用于RDF。

第二部分（Part II）

**查询（Queries）**

**第4章**

**查询语义Web（Querying the semantic web）**

**4.1 动机（Motivation）**

我们知道如何在网络上表达信息。现在有必要好好利用它。

如今，由于RDF具有简单的数据模型、形式化的语义和健全完整的推理机制，所以更多的资源，都通过RDF进行注释。RDF本身，可以作为一个使用RDF蕴涵测试RDF知识库查询语言。尽管如此，回答查询的结果仍然有限。特别是，回答那些包含复杂关系的需要复杂的结构。例如，用一个简单的蕴涵测试，回答查询“查找姓名和地址，如果它们存在，关于或者从事查询语言（query languages）或者从事本体匹配（ontology matching）的那些人”，是不可能的。

因此，在查询中添加表达性（expressivity）的需要，已经导致了在图模式（graph patterns）的顶部，对几个查询语言进行定义，基本上都是RDF图和更精确的GRDF图。下一章的重点，是概述已设计或可用于查询RDF图的一些语言，并讨论它们在表现性和局限性方面的主要区别。

**4.2 什么是一种查询语言?**

基本上，一种查询语言（query language）的目标，是表示根据一种表达进行评估的各种查询。查询评（query evaluation）估涉及各个方面：

* 从某个结构中抽取信息；
* 从一种表达中推断信息；
* 聚合查询答案；
* 构造新结构。

**4.2.1 数据抽取语言（Data extraction languages）**

基本的数据抽取范例是Web。你知道URL，你得到了HTML页面。这是非常基本的，不是很好结构化的：HTML页面的合并操作不是很自然的。

在XML世界中，XPath是提取信息的合适语言。XML文档和答案（answers）都是节点集（node set），对节点集可以应用查询并因此组成它们。XPath允许在结构中进行导航，并提供强大的选择操作。

**4.2.2 数据操纵语言（Data manipulation language）**

数据操纵语言可以做的不仅仅是抽取语言（extraction languages）。在SQL之后，XQuery和XSLT通常提供XML作为输出。他们不仅评估在XML文档中抽取信息的XPath查询，而且利用了答案构建新XML文档。这些文档可以在XQuery或XSLT引擎中重新加载。

**4.2.3 查询语言作为一种代数: SQL**

SQL的一个典型方面，就是把自己变成一个代数，也就是说，查询（query）是一个可以根据代数规则操作的代数的表达式。这被用于分发（distributing）或重写查询（rewriting queries）以进行优化（关于实例的某个模式）。

**4.2.4 查询作为一种测试结果**

到目前为止，我们尚未考虑从表达中推断结果：在这里，查询语言只是提取和转换信息。

由于我们为语义Web语言提供了一种语义（semantics），因此应该对这个语义进行查询评估。因此，不只是从结构中提取信息，一个查询语言的基本操作，应该是知道信息是否是数据的结果。由于我们提供了不同的逻辑用于在Web上表达知识，所以很自然地会考虑到，关于这些不同的逻辑的参数化查询（以及它们所提供的蕴涵种类）。因此，查询语义Web数据，可以表达为：

这样，是一个特定的表达语言（简单RDF，RDF，RDF Schema，GRDF，OWL Lite，OWL DL，OWL Full或其他）。在我们的情境中，公式将基于RDF图。

第5章

**用SPARQL 查询RDF**

已经有针对特定RDF查询语言的早期建议，例如RDQL [SEABORNE 2004]，RQL [KARVOUNARAKIS et al. 2002]或SeRQL[BROEKSTRA 2003]。依据这些早期的尝试[PRUD HOMMEAUX and SEABORNE 2008]，在2004年，W3C推出了数据访问工作组（Data Access Working Group）所设计的一种RDF查询语言，称为SPARQL。SPARQL受启发于SQL关系数据库查询语言。

在以下小节中，我们将定义SPARQL的语法和语义。对SPARQL的完整描述，读者可以参考SPARQL规范[PRUD HOMMEAUX and SEABORNE 2008]或[PEREZ et al. 2009; POLLERES 2007]的SPARQL形式语义。除非另有说明，我们重点集中于SPARQL 1，但在已目前语言中考虑到的一些特点，现在已经集成在了SPARQL 1.1 之中[HARRIS and SEABORNE 2013; GLIMM and OGBUJI 2013]。

**5.1 语法**

在下面，按照SPARQL的官方规范中所做的那样，不考虑它们的存在语义（existential semantics），我们将RDF中空白节点（blank nodes）简单地当作常数（犹如它们是IRI）处理。然而，如果当RDF查询时考虑空节节点的存在语义，那么就要利用图形同态技术（graph homomorphism technique）间接地应用本文的结果[BAGET 2005]。

**5.1.1 SPARQL 图模式（graph patterns）**

SPARQL查询的核心是各种**图模式**（graph patterns）。

通俗地说，一个**图模式**（graph pattern）可以是下列情况之一（更详情见[PRUD HOMMEAUX and SEABORNE 2008]）：

* **三元组模式**(triple pattern)：一个三元组模式在RDF中对应于一个GRDF三元组；
* **基本图模式**(basic graph pattern)：一个三元组模式（或一个GRDF图）集合，称为一个基本图模式；
* **联合图模式**(union of graph patterns)：我们在SPARQL中使用关键词UNION来表达各种选择（alternatives）；
* **可选图模式**(optional graph pattern)：SPARQL允许依据关键词OPTIONAL确定将返回的可选结果；
* **常量**(constraint)：在SPARQL中的常量是布尔值表达式（boolean-valued expressions），这限定了待返回的答案的数量。常量可以用关键词FILTER进行定义。作为一个原子FILTER表达式，SPARQL允许像BOUND的一元谓词；二元（不）等式谓词（和）；比较运算符，像；数据类型转换（data type conversion）以及此处省略的各种字符串函数（string functions）。复杂FILTER表达式的构建，可以使用、和；
* **群图模式**(group graph pattern)：是在｛和｝内部分组归类的一个图模式，并确定诸如FILTER与变量节点这样的SPARQL构造子的范围。

**定义23** (SPARQL图模式). 一个SPARQL图模式，按以下方式归纳地定义：

* 每个GRDF图都是一个SPARQL图模式；
* 如果, 是SPARQL图模式，并且是一个SPARQL图约束，那么(*P* AND *P'*), (*P* UNION *P'*), (*P* OPT *P'*), 以及(*P* FILTER *P'*) 都是SPARQL图模式。

一个SPARQL图约束，是涉及来自的术语一个布尔表达式，例如，一个数字测试。我们没有进一步指定这些表达式。(至于更完整的处理，见[MUNOZ et al. 2009])。

**例13**. 下列图模式：

{ ?person foaf:knows "Faisal" . }

是一个基本的图形模式，可用于查询寻找那些认识Faisal的人。

{

{ ?person ex:liveIn ex:France . }

UNION

{ ?person ex:hasNationality ex:French . }

}

是两种基本图形模式的联合，旨在搜索那些居住在法国或具有法国国籍的人。

下列图模式

{

?person foaf:knows "Faisal" .

OPTIONAL

{ ?person foaf:mbox ?mbox . }

}

包含一个搜索邮箱的可选的基本图模式，如果它们存在，谁知道Faisal。

{

?person ex:liveIn ex:France .

?person ex:hasAge ?age .

FILTER ( ?age < 40 ) .

}

这个图模式中的约束，将应答限定为居住在法国的年龄小于40岁的人。

{

{ ?person foaf:knows "Faisal" . }

{

?person ex:liveIn ex:France .

?person ex:hasAge ?age .

FILTER ( ?age < 40 ) .

}

}

是两个群图模式的一个图模式。在该图模式中约束的范围，是第二个群图模式。因此，它只适用于居住在法国的人。

**5.1.2 SPARQL 查询**

SPARQL 提供了几种查询范式（query forms）。

**定义24** (SPARQL 查询). 给定一个SPARQL图模式，中变量的一个元组，一个IRI集合，和一个基本图模式，

*ASK FROM WHERE P*

*SELECT FROM WHERE P*

*CONSTRUCT FROM WHERE P*

是SPARQL查询。

直观地说，对SELECT查询的一个答案，是用RDF图中的术语对中的变量进行赋值，这样，在这些赋值下，就被由确定的图所蕴含。一个CONSTRUCT查询，用于依据应答集构建一个RDF图。如果对给定的查询存在一个应答，则ASK返回TRUE，否则返回FALSE。此外，DESCRIBE用于描述一个资源RDF图。

下面的示例查询提供了对这些查询范式（query forms）的领悟。

**例14** (查询). 下面的查询搜索在图2.1的调控网络（regulatory network）中的一个基因*?x*，抑制一种产品----调节*?x*促进的一个产品，并返回这三个实体：

SELECT ?x, ?y, ?z

WHERE

?x rn:inhibits ?y

?x rn:promotes ?z

?y rn:regulates ?z

?x rdf:type rn:gene.

下列ASK查询：

ASK

WHERE { ?person foaf:name "Faisal" .

?person ex:hasChild ?child .

}

如果有一个叫Faisal的人至少有一个孩子，返回TRUE，否则为FALSE。

下列CONSTRUCT查询：

CONSTRUCT { ?son1 ex:brother ?son2 .}

WHERE {

?son1 ex:sonOf ?person .

?son2 ex:sonOf ?person .

FILTER ( ?son1 != ?son2 ) .

}

通过将每个定位的答案替换为变量*?son*1和*?son*2的值的方式，构造RDF图（包含兄弟关系）。

下列查询：

DESCRIBE <example.org/person1>

返回由给定的IRI所确定的一个资源的描述，即返回涉及这个IRI的三元组集。

SPARQL使用后过滤子句（post-filtering clauses），允许对一个查询的答案，例如，进行排序（带有限定词ASC / DSC的ORDER BY子句）、分组（GROUP BY）或限制（LIMIT和/或OFFSET(补偿)子句）。读者可以参考SPARQL规范[PRUD HOMMEAUX and SEABORNE 2008]的更多细节或[P EREZ et al. 2009]对SPARQL查询的形式语义。

**例15** (后过滤 SPARQL 查询). 下面的 SPARQL查询：

SELECT ?name

WHERE {

?person ex:liveIn ex:France .

?person foaf:name ?name .

}

ORDER BY ?name ASC

LIMIT 10

OFFSET 5

返回居住在法国的人数最多为10人的名字，按他们的名字排序，并从第五个答案开始。

SPARQL 1.1 [HARRIS and SEABORNE 2013]还介绍了将查询（queries）和源（sources）相结合的条件。特别是，查询可以集成子查询，对特定的命名图（具有FROM NAMED和GRAPH构造子）的一个查询部分进行评估，并评估联合查询（federated queries），即，对其结果进行合并的多数据源的查询（用SERVICE构造子）。

由于在SPARQL查询语言中的图模式被所有SPARQL查询范式共享，所以我们就使用SELECT . . .FROM . . .WHERE . . .查询来说明它们。

**5.2 SPARQL语义**

在下面，我们将查询刻画为[PEREZ et al. 2009]中所做的那样的SPARQL应答。该方法依赖于GRDF在蕴涵和从查询图模式的RDF图到RDF知识库（见定义8，p.24）的映射之间的对应。

**定义25** (应用一个映射到一个基本图模式). 将一个映射应用到一个基本图模式，定义为：

* 如果是一个GRDF图，；
* 如果是一个三元组，；
* 如果，；
* 否则。

SPARQL查询构造子，通过在映射上的代数运算定义：从一组变量到保留名字的术语的赋值。

**映射上的操作**（Operations on maps） 如果是一个映射，代表在其上定义的的子集。对一个术语集的限制，定义为，且对一个术语集的完全（completion），定义为\*\*。

如果是一个图模式，那么就是出现在中的一个变量集，且是由替换为每个变量所获得的图模式。两个映射和是兼容的，当，。否则，就说它们是不兼容的，并用表示。如果和是两个兼容的映射，那么就可以表示为，该映射定义为，=，并且，=。

在下面，我们使用SPARQL查询的另一种表征，依赖GRDF蕴涵和映射之间从查询图模式到RDF图的对应 [PEREZ et al. 2009]。对基本图模式查询的应答，就是依靠被查询的图确保该图模式的蕴涵的那些映射。在SPARQL的情况下，这种蕴含关系（entailment relation）是GRDF蕴涵。

--------------------------------------------------------------

\*\* null符号用于表示OPTIONAL子句引入的NULL空值。

--------------------------------------------------------------------------------------------------------------------------------------

**定义26** (图模式蕴含). 令⊨是在基本图模式上的一个蕴含关系，,是SPARQL图模式，是一个SPARQL约束，并且是一个RDF图，那么，由一个RDF图取模（modulo）一个映射的图模式蕴含，归纳地定义为：

(AND ) *iff* and

(UNION ) *iff* or

(OPT ) *iff* and

(FILTER ) *iff* and

条件被解释为布尔函数，根据它们涉及的术语。因此，，意味着，一旦中的变量被替换，这个函数就被求值为真true。如果并非中所有变量都是绑定的（bound），那么。一个特定的运算符，“*bound*(*?x*)”，可以使用SPARQL过滤器条件。

如果变量*?x*是绑定的，这个操作符返回真值true，在这种情况下，当一个变量是非是绑定的，就不为真值true。

SPARQL FILTER表达式的语义，定义如下：给定一个映射和SPARQL约束，我们说满足（记为=），如果：

* = BOUND with ；
* = with and ；
* = with and ；
* = with and ；
* = with and ；
* = with and ；
* = with and ；
* = with ( does not satisfy )；
* = with or ；
* = with and 。

上面给出的关于蕴涵关系的定义，可被在基本案例（base case）即基本图模式上所定义的一个具体关系所取代。

像往常一样，对于这种查询语言，一个查询的回答，是对不同变量的一个赋值（查询的SELECT部分中的那些变量）。这样的赋值，是从查询中的变量到该图的节点的一个映射。所定义的答案，可能只对变量中足以证明蕴涵的一部分赋值。答案是被扩展到所有不同变量的这些赋值。

**Definition 27** (对一个SELECT SPARQL查询的应答[ALKHATEEB, BAGET, et al. 2009]). 令SELECT FROM WHERE 为关于一个SPARQL图模式的一个SPARQL查询，并且为由IRI确定的 (G)RDF图，则对应这个查询的应答集就是

。

这个定义就是一个SPARQL答案的语义表征。

空值（null）的完成（completion）不会阻止在答案中保留空白节点：空值只替换那些不匹配的变量（unmatched variables）。[POLLERES 2007]定义当映射包含空值时关于连接（join）操作的一个不同语义。可以使用这种语义而不改变其余部分。

**例16** (查询评估). 对例2的RDF图的例14的查询的评价，仅返回一个答案：

〈dm:bcd,dm:tll,dm:Kr〉

**定义28** (对一个ASK/CONSTRUCT SPARQL查询的应答). 令ASK FROM WHERE 为一个关于一个SPARQL图模式的一个SPARQL查询，并且为由IRI确定的 (G)RDF图，则对应这个查询的（布尔）应答集就是

令CONSTRUCT FROM WHERE 为一个关于一个SPARQL图模式的一个SPARQL查询，为一个RDF图，为由IRI确定的 (G)RDF图，则对应这个查询的应答集就是

**例17** (查询评估). 下列查询：

SELECT ?z ?y ?k

WHERE {

{ ?x ex:hasDaughter ?y . } UNION { ?x ex:hasSon ?y . }

AND { ?x ex:hasSon ?z . } UNION { ?x ex:hasDaughter ?z . }

AND { ?w ex:hasChild ?z . ?v ex:hasChild ?y . }

FILTER { ?w != ?v && ?v != ?x && ?x != ?w }

OPTIONAL { ?x foaf:lastname ?k . }

}

待评估于：

ex:Peter foaf:lastname "MacCartney" .

ex:Peter ex:hasDaughter ex:Mary . ex:Shirley ex:hasChild ex:Mary .

ex:Peter ex:asSon ex:Paul . ex:Kate ex:hasChild ex:Paul .

ex:Peter ex:hasDaughter ex:Julia. ex:Kate ex:hasDaughter ex:Julia .

将返回值元组：

ex:Mary ex:Paul "MacCartney"

ex:Paul ex:Mary "MacCartney"

如果与查询类型一起使用：

CONSTRUCT { ?z ex:hasHalfSibling ?y . }

它将会加入三元组：

ex:Mary ex:hasHalfSibling ex:Paul .

ex:Paul ex:hasHalfSibling ex:Mary .

它似乎是正确的兄弟姐妹吗？有什么问题需要解决吗？

为了评估查询应答（query answering）的复杂性，我们使用以下问题，通常命名为**查询评估**（query evaluation），但更适合命名是**ANSWER CHECKING**（**查询检验**）：

**Problem:** -**ANSWER CHECKING**

**Input:** an RDF graph , a SPARQL graph pattern , a tuple of variables , and a map .

**Question:** Does ?

这个问题通常与检查答案是否存在一样复杂。对于SPARQL，问题已被证明是PSPACE -complete。

**命题7** (-**ANSWER CHECKING**的复杂性 [PEREZ et al. 2009]). -**ANSWER CHECKING** 是PSPACE –complete。

检查RDF-蕴涵和GDRF-蕴涵是NP-complete [GUTIERREZ et al. 2004]。这意味着，-**ANSWER CHECKING**当查询被约减到基本图模式时是NP-complete。事实上，加入AND、FILTER和UNION不会增加复杂性，仍然是NP-complete。PSPACE复杂性，来自OPT构造的加入[PEREZ et al. 2009]。

因此，对于每一个其蕴涵是NP-complete的语言，作为一种基本图模式语言，由于定义26解释了子查询评估的独立性，这种语言的问题将会有同样的复杂性。

**5.3代数操作（Algebraic manipulation）**

回答SPARQL查询，可以通过直接操纵图和映射来获得。SPARQL原有语义是以这样方式给出的。为了定义这些操作，我们需要进一步定义关系数据库理论。**join操作**和**两个映射**和的**差**（difference），**定义**如下[PEREZ et al. 2009]：

* (join)
* (difference)

一个基本图模式查询的答案，就是那些映射，借助查询的图，这些映射确保该图模式的蕴涵成立。在SPARQL的情况下，这种蕴含关系就是RDF-蕴涵。通过在映射上的操作，得到复合图模式（compound graph patterns）的答案。

**定义29** (对复合图模式的应答). 令是在基本图模式上的一个蕴含关系，,是SPARQL图模式，是一个SPARQL约束，并且是一个RDF图。那么，对中的应答集，是按以下方式定义的从到的映射集：

如果是一个基本图模式

((*P* **AND** *P’*), *G*)=

((*P* **UNION** *P’*), *G*)=

((*P* **OPT** *P’*), *G*)=

((*P* **FILTER** *K*), *G*)=

下面的推论（corollary）的获得，得益于插值引理（interpolation lemma）和及其对同态（homomorphisms）的应用。

**推论 8.** 令SELECT FROM WHERE 为一个关于一个基本图模式的一个SPARQL查询，为由IRI确定的 (G)RDF图，且是对中用GRDF-蕴含的应答集，则

。

**证明**（*Proof*）. 根据插值引理，如果存在从到的一个同态，则，此外，这个同态决定了的一个实例，因此。因此，，所以，这正好就是 [参见定义27中的公式]。 🞏

**借助对图模式的结构归纳**，**这一推论可以扩展到任何图模式**。

**命题 9** (对SPARQL查询的应答 [ALKHATEEB and EUZENAT 2013]). 令SELECT FROM WHERE 为一个SPARQL查询，为由IRI确定的 (G)RDF图，且是对中用GRDF-蕴含的应答集，则对查询的应答，是对中的应答的制约（restriction）与完善（completion），即

。

**证明**（*Proof*）. 证明是通过归纳得到的：基础案例（base case）是推论8的基本案例：

如果是一个基本图模式

归纳步骤是基于这样的想法，如果基础步骤（base step）关于子组件（和）为真，然后：

((*P* **AND** *P’*), *G*)

(*P* **AND** *P’*)

((*P* **UNION** *P’*), *G*)

(*P* **UNION** *P’*)

((*P* **OPT** *P’*), *G*)

(*P* **OPT** *P’*)

((*P* **FILTER** *K*), *G*)⟺

(*P* **FILTER** *K*)

🞏

这揭示了语义定义（定义26）和代数定义之间的等价性（定义29）。

**5.4 蕴含机制（Entailment regimes）**

有关语义定义的好消息是，SPARQL的答案是用语义定义的，独立于任何评价机制。它也是关于一个简单蕴涵关系（simple entailment relation）的定义。因此，通过改变这种蕴含关系，就有可能获得某种不同的查询语言。

SPARQL规范[PRUD’HOMMEAUX and SEABORNE 2008]介绍了蕴含机制（entailment regimes）的概念。目前，这些都被 [GLIMM and OGBUJI 2013]进一步细化（但是，仍然在“图模式同态”的定义下，而不是一个语义定义）。一个蕴含组织方法，定义的不仅仅是蕴涵关系的变化。它涵盖了：

**图语言**()：允许哪些类型的图（例如，RDF，RDFS，OWL）；

**查询语言**()：允许哪些类型的查询（例如限制或扩展约束）；

**蕴含关系**()：用于定义答案的蕴涵关系（例如，RDF-蕴涵，RDFS-蕴涵，经OWL调味的）；

**应答类型** ()：这个将作为一个应答（answer）（例如，允许的空白节点，skolemisation空白节点，正是此解决方案是否导致一致理论）；

**语法错误处理**：如果在定义出现错误时，需要做什么（例如，引发异常，忽略不正确部分）；

**不一致性处理**：如果是不一致的，需要做什么（例如，引发异常，回答应答的特定子集）。

我们已经定义了SPARQL语义，因此就很容易改变蕴含机制。

**5.5 结论**

SPARQL是数据库查询语言，主要是允许查询RDF图。正如我们所看到的，它可以提供基于RDF语义的语言的语义描述。

在下面，我们将说明如何才能改变查询语言（§6）以及如何才能改变蕴涵关系，针对图模本体（graphs modulo ontologies）的查询（§7）。

**5.6 习题**

**习题18** (SPARQL 查询). 给定下列SPARQL 查询：

**SELECT** ?t

**PREFIX**

foaf: http://xmlns.com/foaf/0.1/

m: http://mydomain.com/myExample#

dc: http://purl.org/dc/elements/1.1/

rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

**WHERE** {

?x foaf:name "Albert".

?x ?r ?l.

?r rdf:type rdf:Property.

?l rdf:type m:Roman.

?l dc:title ?t.

}

1. 这个查询的非正式含义是什么？
2. 绘制对应该查询的图模式的RDF图。
3. 这种图模式与简单RDF图有什么区别？
4. 评估习题1中图2.6的每个有充分根据的图上的查询，并提供答案。
5. 针对要返回m:Roman的出版年份的这个查询，必须要添加入什么，如果它是可以获得的？

**习题 19** (简单SPARQL查询).

1. 你能表达一个SPARQL查询，按照习题17中问题3的OWL公理定义，返回所有o:TonicPackage？

**习题20** (SPARQL 与 查询). 给定下列 SPARQL 查询：

**SELECT** ?m, ?n

**PREFIX** foaf: http://xmlns.com/foaf/0.1/

**WHERE** {

?x foaf:mbox ?m.

?x foaf:knows ?y.

?z foaf:mbox "keith@i.com".

?z marriedWith ?y.

**OPTIONAL** { ?x foaf:name ?n }

}

1. 这个查询的意义的非正式是什么（用英语或法语说）？
2. 画出对应于这个查询的图模式的GRDF图。
3. 评估对习题2的图2.7中图的查询，并提供结果。
4. 对在习题9问题4获得的闭包上的查询进行评估，并给出结果。

**习题21** (SPARQL 查询蕴含). 考虑下列在习题3中的RDF图上的查询和：

**SELECT** *?w* **FROM** *G* **WHERE** ( **AND** )

**UNION**

**SELECT** *?w* **FROM** *G* **WHERE** ( **UNION** )

**AND**

1. 在这个过程中，我们定义了区分变量（distinguished variables），查询图和查询模式。在和中识别它们。
2. 提供和关于图的应答。

在两个查询= **SELECT** **FROM** **WHERE** 与=**SELECT** **FROM** **WHERE** 之间，由关于任意RDF图的事实定义的查询包含（query containment），对的应答被包含在对的应答之中。

1. 前一个问题的答案告诉了你和之间的查询包含吗？
2. 你认为查询包含保持在和之间的某个方向吗（或者或者）？
3. 为此提供证明。这可以用语义方式通过使用查询模式的解释来完成，或者用语法方式通过将查询翻译成逻辑来完成，进而说明查询包含语句是一个定理。

第6章

**扩展SPARQL**

SPARQL是基于三元组的基本查询语言。然而，对于许多应用程序，需要更精细地使用查询。因此，不同的工作试图在某些方面改进该语言。

可以有不同的方法对SPARQL进行扩展（extending）或约减（reducing）：

* 增加操作 (**AND**, **OR**, 等)，像 **COUNT**(.) 或 **MINUS** 或，
* 改变基本图模式的定义，如使用路径表达式（path expressions）代替三元组。

以上提到的扩展为例，在SPARQL 1.1[HARRIS and SEABORNE 2013]中给出了详细的介绍。

在查询数据时考虑本体的背景下，我们考虑下面几个这种方式的扩展。

**6.1 NSPARQL**

我们在这里提供一个稍微不同的对这些表达式的使用，我们命名为NSPARQL以区别于它起源的nSPARQL [PEREZ et al. 2010]. 然而，这种方法的优点是直接继承了原先的nSPARQL。我们的陈述在两点上有所不同：

* 我们对嵌套正则表达式（nested regular expressions）使用了简化但等价的描述；
* NSPARQL就是在图模式的谓语位置中使用嵌套的正则表达式的SPARQL，而nSPARQL却不考虑投影操作 (SELECT) [PEREZ et al. 2010]。

**6.1.1 nSPARQL 语法**

**定义30** (正则表达式). 正则表达式，是依据以下语法建立的一个表达式：

具有并且{self, next, next-1, edge, edge-1, node, node-1}。

在下面，我们用表示一个路径表达式（path expression）的正闭包（positive closure），定义为。

关于正则表达式运算符的优先级，规则顺序如下：\*，/，然后⎮。圆括号()可用于打破优先级规则。

该模型构成nSPARQL的基础，是XPath的模型，用于在XML结构内进行导航。因此，轴axis表示分别地在每一步所选的节点对象的类型，当前节点（或），通过一个出射三元组（outbound triple） 代表该节点可达到，这些节点通过一个入射三元组（incident triple）可以到达当前节点、出射三元组的属性（）、入射三元组的属性（）、一个谓词的对象（）和一个对象的谓词（）。图6.1对此进行了说明。

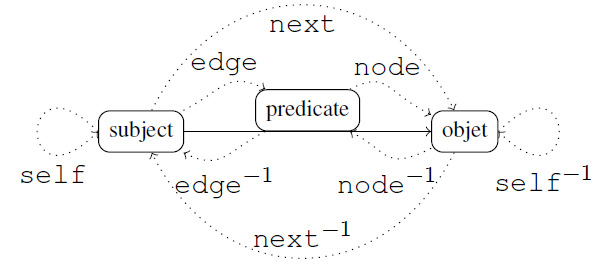


图6.1: nSPARQL 轴线

**定义31** (嵌入式正则表达式). 嵌入式正则表达式（nested regular expression）是依据下列语法构建的一个表达式（具有）：

与简单正则表达式相反，嵌套式正则表达式可能约束节点以满足额外的辅助路径。

嵌套式正则表达式被用于三元组模式中的谓语位置，定义nSPARQL三元组模式。

**定义32** (nSPARQL三元组模式). nSPARQL三元组模式，是一个三元组 使得，并且是一个嵌套式正则表达式。

**例18** (nSPARQL 三元组模式). 假设一个人想要检索城市对，使得存在一种借助任何交通工具的旅行方式。以下nSPARQL模式表示该查询：

这个模式表示一系列属性，使得每个属性（谓词）都是属性“transport”的子属性。

**例19** (nSPARQL三元组模式). 在分子生物学中，nSPARQL表达可能是非常有用的。例如，用于例14查询的部分图模式可以用以下方式表示：

?x next::rn:inhibits / next::rn:regulates ?z

它发现所有对节点，使得第一个节点抑制第二个节点的调节器。它可以通过使用传递闭包（transitive closure）进一步增强：

?x next::rn:inhibits / next::rn:regulates+ ?z

表示我们希望在两个节点之间有一个路径，通过一个首次抑制，然后是一个任意的非空数的调节步骤（+是通常的表示法，使得a+与a/a\*对应）。嵌套式表达式允许进一步约束路径中的任何步骤。所以，

?x next::rn:inhibits[ next::rdf:type/self::dm:gap ] / next::rn:regulates+ ?z

此外，路径中的第二个节点需要一个间隙基因。

依据nSPARQL三元组模式，也可以创建一个查询语言。像在SPARQL中一样，一组nSPARQL三元组模式，称为一个nSPARQL基本图模式。nSPARQL图模式，可以按通常的方式进行定义，即通过nSPARQL三元组模式替代三元组模式。

**定义33** (nSPARQL图模式). 一个nSPARQL图模式，可归纳地定义为：

* 每一个nSPARQL三元组模式都是一个nSPARQL图模式；
* 如果和是两个nSPARQL图模式，且是nSPARQL约束，那么( **AND** )，( **UNION** )，( **OPT** )，和( **FILTER** ) 都是nSPARQL图模式。

由于时间复杂度的原因，nSPARQL语言的设计者选择定义更多的限制比SPARQL语言[PEREZ et al. 2010]. 与SPARQL查询相反，nSPARQL查询被约减成根据nSPARQL三元组模式构建的nSPARQL图模式，附加上SPARQL算子AND、UNION、FILTER及OPT。它们不允许投影操作（SELECT）。这可以防止，在检查答案时，必须对未受控制的变量进行评估。

**6.1.2 nSPARQL 语义**

为了确定nSPARQL的语义，我们需要知道嵌套式正则表达式的语义[PEREZ et al. 2010]. 这里，我们背离了[PEREZ et al. 2010]给出的语义。在解释中添加第三个变量，其唯一目的是压缩规则集。这两个定义都是等价的。

**定义34** (嵌套式路径解释). 给定一个嵌套式路径（nested path）及一个RDF图，中的解释（表示为）定义为：