本体的推理研究

纪兆辉

(淮海工学院 计算机工程学院,江苏 连云港 222005)

[摘要] 针对 OWL 所描述的本体,介绍了本体推理机的构成,重点阐述了 Jena 语义 Web 编程框架中内置的几种推理模式,并通过叠加外部推理机 Pellet 来弥补单一推理机在推理能力方面的不足. 以一个小型本体为例,通过运用 Jena 内置的推理模式和叠加外部推理机的方式对本体的推理进行了探索和验证.

[关键词] 本体,推理,OWL,Jena,Pellet

[中图分类号] TP391 [文献标志码] A [文章编号] 1672-1292(2012)03-0054-06

Research on Ontology Based Inference

Ji Zhaohui

(School of Computer Engineering, Huaihai Institute of Technology, Lianyungang 222005, China)

Abstract: In relation to the ontology described by OWL, the paper firstly introduces the architecture of common ontology reasoners, then some built-in inference schemas in Jena semantic Web framework are explained in details. Meanwhile, the technique called reasoner cascading is also introduced in order to improve the capability of a single reasoner. Taking a mini-ontology as an example, we explore the ontology inference by using Jena built-in inference schemas and the outer reasoner Pellet.

Key words: ontology, inference, OWL, Jena, Pellet

本体是共享概念模型的明确的形式化规范说明[1]. 本体的本质表现在:(1)概念化:指通过抽象出客观世界中一些现象的相关概念而得到的模型;(2)明确:指概念及它们之间的联系都有明确的定义;(3)形式化:指精确的数学描述,计算机可读;(4)共享:指本体中体现的知识是被不同的人和组织所共同认可的,可以跨应用系统进行互操作.

本体在发展过程中先后出现多种描述语言,目前 OWL[2] 已成为被广泛接受的本体语言,大部分开发和研究人员都基于 OWL 来描述本体中的概念及其语义联系. 使用 OWL 本体作为信息检索的载体,用户能够方便地在概念上描述信息需求,构造复杂的语义查询. 本体的查询需要借助于 SPARQL[3] 语言,但由于该语言只能查询 OWL 知识库中显式定义的知识,本身不具备推理的功能,因此需要借助推理引擎,将本体中具有隐含语义关联的数据提取出来,获得所有相关联的数据作为 SPARQL 查询的数据源. 对本体中蕴含的知识进行有效地查询与智能化推理,是本体在各个领域能够成功应用的基础和关键.

一些研究机构也提供了各种基于本体的推理机来对本体蕴含的知识进行推理. 本体的推理从根本上说就是把隐含在显式定义和声明中的知识通过一种处理机制提取出来. 对本体的开发人员来说,本体的推理可以用于检测本体定义中存在的冲突,消除不一致性,优化本体表达和实现本体融合;而对于知识管理、语义检索、自然语言理解等诸多领域的本体使用者来说,本体的推理可以获得本体中特定形式的知识集合并用于解决实际问题.

本文研究了推理机的基本结构和工作原理,并以一个小型本体为例,结合 Jena[4] 语义 Web 编程框架中几种内置的推理模式,实现了基于 RDFS、OWL 和自定义规则的推理. 同时,为了弥补单一推理机在推理能力上的不足,提高其推理的完备性,对通过 Jena 推理机叠加外部推理机 Pellet[5] 的工作模式也进行了分

收稿日期:2012-06-28.

通讯联系人:纪兆辉,副教授,研究方向:语义 Web 与本体论. E-mail:jzhqjhui@126.com

析. Pellet 提供了对 OWL 和 SWRL 更全面的支持,是一款合理完备的本体推理机.

1 OWL 样例本体

OWL 样例本体描述了 Mammal、Human 和 Canine 等概念及其关系,另外还建立了相应的个体. 同时为了便于阅读,本文采用 Turtle[6] 语法,基于三元组的形式对此 OWL 本体进行序列化. 每一个三元组用谓词将不同的概念联系起来,同一个概念可以通过多个谓词与其他多个概念相联系. 用 Turtle 语法表示的语句具有形式"subject predicate$_1$ object$_1$; predicate$_2$ object$_2$; …predicate$_n$ object$_n$. ". OWL 样例本体的代码如下:

```
#以下定义命名空间前缀
@ prefix ex:  < http: //example. org# > .
@ prefix rdf:  < http: //www. w3. org/1999/02/22-rdf-syntax-ns# > .
@ prefix rdfs:  < http: //www. w3. org/2000/01/rdf-schema# > .
@ prefix owl:  < http: //www. w3. org/2002/07/owl# > .
#以下定义了 7 个类( 概念)
ex: Mammal rdf: type owl: Class.
ex: Human rdf: type owl: Class;  rdfs: subClassOf ex: Mammal.
ex: Canine rdf: type owl: Class;  rdfs: subClassOf ex: Mammal;
        owl: equivalentClass  [rdf: type owl: Restriction;  owl: onProperty ex: breed;
            owl: someValuesFrom ex: Breed] .
ex: PetOfRyan rdf: type owl: Class;  owl: intersectionOf
            ( ex: Mammal  [rdf: type Owl: Restriction;  owl: onProperty ex: hasOwner;
                owl: hasValue ex: Ryan] ) .
ex: Breed rdf: type owl: Class.
ex: LargeBreed rdf: type owl: Class;  rdfs: subClassOf ex: Breed.
ex: SmallBreed rdf: type owl: Class;  rdfs: subClassOf ex: Breed.
#以下定义了 2 个数据类型属性
ex: name rdf: type owl: DatatypeProperty.
ex: registeredName rdf: type owl: DatatypeProperty;  rdfs: subPropertyOf ex: name.
#以下定义了 3 个对象属性
ex: breed rdf: type owl: ObjectProperty.
ex: hasOwner rdf: type owl: ObjectProperty.
ex: owns rdf: type owl: ObjectProperty;  owl: inverseOf ex: hasOwner.
#以上部分构成了 OWL 知识库的 TBOX
#以下定义了 5 个实例( 个体) ,构成了知识库的 ABOX
ex: GoldenRetriever rdf: type ex: LargeBreed.
ex: Chihuahua rdf: type ex: SmallBreed.
ex: Ryan rdf: type ex: Human;  ex: : name "Ryan Blace";  ex: owns ex: Daisy.
ex: Daisy rdf: type ex: Canine;  ex: name "Daisy";  ex: breed ex: GoldenRetriever;
ex: registeredName "Morning Daisy Bathered in Sunshine".
ex: Amber rdf: type ex: Mammal;  ex: name "Amber";  ex: breed ex: GoldenRetriever.
```

2 本体推理机的构成

本体中蕴含了丰富的语义知识,对本体知识库进行推理,是以本体作为信息组织形式比传统数据库更智能化的方面[7]. 通过推理可以获得本体中特定形式的知识集合以及运用本体中的知识来辅助解决涉及语义的应用.

本体推理机由本体解析器、查询分析器、推理引擎、结果展现等模块组成,构成如图 1 所示.

其中,本体解析器提供了对文件形式或者存储在数据库中的本体进行解析;查询分析器为用户提供了分析用户查询命令的接口;推理引擎是整个推理机的核心,其内部结构对本体推理能力的影响至关重要.推理机常用的推理模式可以分为3种[8]:

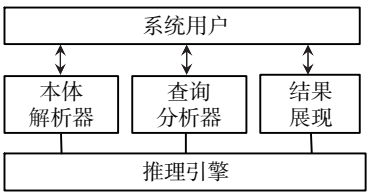

图1　本体推理机的构成

Fig.1　The composition of an ontology inference engine

(1) RDFS 推理模式:这种推理模式意味着推理应当在所有RDFS词汇表元素(属性和类层次关系、定义域、值域)之上进行.

(2) OWL 推理模式:这种推理模式可以基于OWL类的公理、属性公理以及各种约束条件进行推理,可以得到OWL语义所蕴含的新信息.

(3) 基于规则的推理模式:这种推理模式可以按照某种规定语法所明确定义的规则灵活地构造各种关系.推理规则的定义可以用SWRL来完成,然后交给某种推理引擎来解释执行.但目前大多数的推理引擎除了Pellet外一般不支持SWRL规则.像Jena这样的推理引擎可以将规则写在应用程序里或者单独保存在规则文件中.

目前一些研究机构已开发了一些基于本体的实际使用的推理机,比较流行的有Racer、Pellet和Jena.

3　基于Jena语义框架的本体推理

3.1　Jena推理机的结构

Jena[9]是专门用来构建语义Web应用的编程框架,它为RDF、RDFS和OWL提供了一个可编程实现的环境,具有很好的稳定性和通用性,通过Jena提供的OWL API接口、SPARQL查询接口和本体推理机接口,可以编写基于本体的数据集成、语义检索等智能应用程序.其中Jena框架中推理机的整体结构如图2所示.

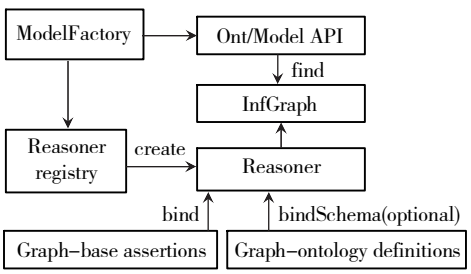

图2　Jena推理机的整体结构

Fig.2　The architecture of Jena inference engine

应用程序通过ModelFactory将本体数据集同特定的推理模式进行关联以创建推理模型.对于所创建的推理模型的查询不仅返回原有数据中的三元组信息,同时返回通过推理机制所导出的隐含语义信息.

从图2看出,推理机(Reasoner)是实现于Graph SPI层之上,因此可以围绕InfGraph创建任意不同的模型接口.Graph是保存语义Web数据的基本方法,支持基本的添加、删除、查找和包含操作.一般来说,一个应用程序并不直接对Graph对象进行处理,Graph接口支持不同类型存储机制的实例化,就语义Web存储来说,这提供了低层次上的灵活性.特别应指出,Jena的本体编程接口提供了便捷的方式来把合适的推理机同它所创建的本体模型(OntModel)进行关联.作为通用的RDF编程接口的一部分,系统还提供了InfModel类用于对标准的模型接口进行扩展,InfModel类提供了对底层的推理图的额外的控制和访问.

推理机的编程接口通过调用bindSchema方法,将特定的推理机与本体模式或本体数据进行绑定,然后特定的推理机就可以通过调用bind方法附加到不同的实例数据集上.同时,Jena具有很好的开放性,通过ReasonerRegistry可以获取目前可用的各种推理机,并且可以注册新的推理机类型和查找给定类型的推理机.

利用Jena进行本体推理的基本步骤如下:

(1) 声明本体模型:OntModel model = null;

(2) 创建特定的推理机:以创建OWL层次的推理机为例:

```
Reasoner reasoner = ReasonerRegistry.getOWLReasoner();
```

(3) 获取推理模型:

```
Model infModel = ModelFactory.createInfModel(reasoner,ModelFactory.createDefaultModel());
```

(4) 将推理模型与本体模型进行绑定:

```
model = ModelFactory.createOntologyModel(OntModelSpec.OWL_DL_MEM,infModel);
```

(5) 从文件或数据库中把本体数据加载到本体模型中:

```
model. read( ) ; //此去省略了 read( ) 方法的相应参数
```

(6) 执行推理,查询输出推理结果:

此处以输出个体信息为例,示例代码如下:

```
ExtendedIterator iIndividuals = model. listIndividuals( ) ;
while( iIndividuals. hasNext( ) )
{ Individual i = ( Individual) iIndividuals. next( ) ;
  printIndividual( i,writer) ; }
```

3.2 Jena 的内置推理机

考虑到不同层次的推理需求,Jena 语义 Web 编程框架支持 4 种内置的推理机,在实际应用中,应根据推理的需求选用适当的推理模式.

(1) RDFS 推理机: 支持 RDFS 蕴含,实现基于 RDFS 中类和属性的关系进行推理. 对样例 OWL 本体采用 RDFS 推理机进行推理,可以得到如表 1 所示的关于个体的隐含信息.

表 1 对样例本体进行 RDFS 推理后得到的蕴含信息

Table 1 Implicit information obtained by RDFS reasoning

个体名称	推理得到的蕴含信息	说明
Chihuahua	type: http: //example. org#Breed	个体 Chihuahua 属于 SmallBreed 类,而 SmallBreed 类是 Breed 类的子类
GoldenRetriever	type: http: //example. org#Breed	个体 GoldenRetriever 属于 LargeBreed 类,而 LargeBreed 类是 Breed 类的子类
Daisy	(1) name: Morning Daisy Bathered in Sunshine (2) type: http: //example. org#Mammal	(1) registeredName 属性是 name 属性的子属性; (2) 个体 Daisy 属于 Canine 类,而 Canine 类是 Mammal 类的子类
Ryan	type: http: //example. org#Mammal	个体 Ryan 属于 Human 类,而 Human 类是 Mammal 类的子类

(2) 传递推理机: 仅支持 RDFS 中对称属性和传递属性的蕴含. 如可以对家族本体中的 hasSibling 属性进行这种类型的推理.

(3) OWL 推理机: 支持 OWL 蕴含的推理,涉及到的 OWL 构造包括 sameAs、SymmetricProperty 和 maxCardinality 等. 目前,Jena 对 OWL 推理的支持还不够完备. 对示例本体数据采用 OWL 推理机进行推理,可以推出如表 2 所示的关于个体的隐含信息.

表 2 对样例本体进行 OWL 推理后得到的蕴含信息

Table 2 Implicit information obtained by OWL reasoning

个体名称	推理得到的蕴含信息	说明
Chihuahua	(1) type: http: //www. w3. org/2002/07/owl#Thing (2) sameAs: http: //example. org#Chihuahua	(1) 任何类都是 owl: Thing 的子类 (2) 任何个体都与自身相同
GoldenRetriever	(1) type: http: //www. w3. org/2002/07/owl#Thing (2) sameAs: http: //example. org#GoldenRetriever	(1) 任何类都是 owl: Thing 的子类 (2) 任何个体都与自身相同
Ryan	(1) type: http: //www. w3. org/2002/07/owl#Thing (2) sameAs: http: //example. org#Ryan	(1) 任何类都是 owl: Thing 的子类 (2) 任何个体都与自身相同
Daisy	(1) hasOwner: http: //example. org#Ryan (2) type: http: //www. w3. org/2002/07/owl#Thing (3) type: http: //example. org#PetOfRyan (4) sameAs: http: //example. org#Daisy	(1) ex: owns 属性和 ex: hasOwner 属性是逆反属性 (2) 任何类都是 owl: Thing 的子类 (3) Daisy 是 Mammal 类的个体,且 hasOwner 属性的值为 Ryan (4) 任何个体都与自身相同
Amber	(1) type: http: //www. w3. org/2002/07/owl#Thing (2) type: http: //example. org#Canine	(1) 任何类都是 owl: Thing 的子类 (2) ex: Canine 类与 breed 属性取值为 Breed 对象的类相等价

(4) 通用规则引擎: 该引擎包含了其自身的规则语言和实用的方法,可以构建出 if/then 形式的规则. 一般将规则保存在程序中的字符串或项目的 rules 文件中,规则的格式如下:

[规则名: (三元组$_1$),(三元组$_2$),…(三元组$_n$)→(三元组$_{n+1}$)].

其中,前 n 个三元组为规则的前提,第 $n+1$ 个三元组为规则的结论,每个三元组具有形式: subject predicate object. 其中 subject 和 object 分别为主语和宾语,一般为概念,而 predicate 为谓词,一般为对象属性. 如家族本体中,已知 A hasSibling B,B has Daughter C,则蕴含有 A hasNiece C,但基于前面几种内置的

推理机无法实现这一功能. 此处定义推理规则 rule_hasNiece,规则定义如下:

[rule_hasNiece: (? A family: hasSibling? B),(? B family: hasDaughter? C) →(? A family: hasNiece? C)]

通过以下语句即可以创建通用规则引擎:

```
Reasoner ruleReasoner = new GenericRuleReasoner( Rule. parseRules( rules) ) ;
```

3.3 在 Jena 中集成外部推理机

外部推理机可以通过两种方式集成到 Jena 中:在 Java 的 classpath 中直接添加. jar 文件或类文件,或者通过远程 DIG 接口来实现. 一般通过设置 classpath 来实现外部推理机集成的方法比较方便高效,因为不需要任何网络来和推理机进行交互. Pellet 是基于 Java 的 OWL-DL 推理机,因此提供了非常方便的包含. jar 的接口,非基于 Java 的推理机通过提供 JNI 接口也可以提供这样的访问[5]. 由于 Pellet 几乎支持 OWL1 和 OWL2 的所有特征,是一款合理完备的推理机,且对 SWRL 提供很好的支持,所以通过叠加 Pellet 可以明显提高 Jena 推理的完备性,弥补了单纯使用 Jena 推理机带来的不足. 对 Pellet 推理机的集成方式和 Jena 内部推理机很类似,使用 PelletReasonerFactory. theInstance( ) . create( ) 创建一个 Jena 推理机对象,推理机对象绑定到模式的方法也和 Jena 内部推理机一样. 示例代码为:

```
Reasoner reasoner = PelletReasonerFactory. theInstance( ) . create( ) ;
reasoner = reasoner. bindSchema( ontModel) ;
InfModel infmodel = ModelFactory. createInfModel( reasoner,ontModel) ;
```

如果一个推理机不允许通过 Java 的. jar 文件来直接访问,则 Jena 框架还支持 DIG 接口,这是一种标准的推理机接口,是关于通过 HTTP 协议访问描述逻辑处理的一个 XML 标准. 叠加了 Pellet 推理机后,以上家族本体中的 hasNiece 规则,可以写成以下的 SWRL 规则:

$$\text{hasSibling}(?x,?y) \wedge \text{hasDaughter}(?y,?z) \rightarrow \text{hasNiece}(?x,?z)$$

实验和研究表明,通过推理机叠加,一方面利用了外部推理机 Pellet 对具有推理完备性和可判定性的 OWL DL 的支持,另一方面也充分发挥了自定义规则灵活广泛的优势. 这种叠加是目前使用开源软件能够实现的一种功能较强、结果较完备的语义推理机解决方案,但如果实现商业化的基于大型本体知识库的语义推理和应用,Racer 推理引擎则是更好的选择.

4 结语

本文在介绍本体推理机结构的基础上,重点阐述 Jena 语义 Web 编程框架中内置的几种推理模式,并介绍通过叠加外部推理机 Pellet 的方式来弥补单一推理机在推理能力方面的不足. 文中以一个小型本体为例,通过运用 Jena 内置的推理模式和叠加外部推理机的方式对本体的推理进行了探索和验证. 在编写基于本体的应用程序时,可以充分利用 Jena 的稳定性、通用性,结合 Pellet 对 OWL 和 SWRL 的强大支持,通过内外部推理机的层叠,可以实现功能完备的推理应用. 该类型的推理机在本体一致性检测、新知识获取等方面都有重要的作用,可在一定范围内实现语义检索等基于本体的智能应用.

今后的研究将围绕本体推理机的性能和推理完备性的验证,并基于 Jena 语义 Web 框架实现具有知识管理功能的本体应用开发.

[参考文献](References)

[1] Knowledge System Laboratory,Stanford University. What is an Ontology [EB/OL]. (2009-01-28) [2011-12-24]. http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.

[2] World Wide Web Consortium. OWL Web Ontology Language Overview [EB/OL]. (2009-11-12) [2011-12-24]. http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[3] World Wide Web Consortium. SPARQL Query Language [EB/OL]. (2008-01-15) [2011-12-26]. http://www.w3.org/TR/rdf-sparql-query/.

[4] The Apache Software Foundation. Apache Jena [EB/OL]. (2009-04-23) [2012-01-22]. http://incubator.apache.org/jena/.

[5] Clark & Parsia,LLC. Pellet: OWL 2Reasoner for Java [EB/OL]. (2010-08-21) [2012-01-22]. http://clarkparsia.com/pellet/.

[6] World Wide Web Consortium. Turtle-Terse RDF Triple Language [EB/OL]. (2011-03-28) [2012-01-22]. http://www.w3.org/TeamSubmission/turtle/.
[7] 纪兆辉. 本体的查询与推理研究 [J]. 微电子学与计算机,2011,28(10):52-55.
Ji Zhaohui. Research on ontology querying and inference [J]. Microelectronics and Computer,2011,28(10):52-55. (in Chinese)
[8] 宋晓峰,唐发根. 基于本体的推理技术的相关研究 [EB/OL]. (2007-09-27) [2011-12-28]. http://www.paper.edu.cn.
Song Xiaofeng,Tang Fagen. Research on Ontology Based Inference Technology [EB/OL]. (2007-09-27) [2011-12-28]. http://www.paper.edu.cn. (in Chinese)
[9] 唐富年,唐荣年. Web 3.0 与 Semantic Web 编程 [M]. 北京: 清华大学出版社,2010.
Tang Funian, Tang Rongnian. Web 3.0 and Semantic Web Programming [M]. Beijing: Tsinghua University Press,2010. (in Chinese)

[责任编辑: 严海琳]

(上接第 53 页)

[3] 王继瑞. 基于 J2EE 平台的代码生成器 [D]. 济南: 山东大学计算机科学与技术学院,2006.
Wang Jirui. Code generator base on J2EE [D]. Jinan: School of Computer Science and Technology, Shandong University,2006. (in Chinese)
[4] 张敏. 数据驱动的 J2EE 应用开发方法的研究及其自动化代码生成工具的设计 [D]. 长沙: 国防科学技术大学计算机学院,2005.
Zhang Min. Research on the development of data driven application and design of the automatic code generating tools [D]. Changsha: School of Computer, National University of Defense Technology,2005. (in Chinese)
[5] 李文华,杨亚仿,吴昊. Ext JS 框架下表格组件的应用与改进 [J]. 电脑编程技巧与维护,2012(4):63-64.
Li Wenhua,Yang Yafang,Wu Hao. The application and improvement of grid component based on Ext JS frame [J]. Programming Skills & Maintenance,2012(4):63-64. (in Chinese)
[6] 徐会生,何启伟,康爱媛. 深入浅出 Ext JS [M]. 北京: 人民邮电出版社,2009.
Xu Huisheng,He Qiwei,Kang Aiyuan. Head First Ext JS [M]. Beijing: Posts & Telecom Press,2009. (in Chinese)
[7] 朱敏,贾长云. FrontPage 网页制作自动阅卷系统的研究与实现 [J]. 计算机时代,2010(11):60-61.
Zhu Min,Jia Changyun. Research and realization of automatic examining system for webpage making with front page [J]. Computer Era,2010(11):60-61. (in Chinese)

[责任编辑: 严海琳]