本体推理机及应用①

潘 超 古 辉 (浙江工业大学 计算机科学与技术学院 浙江 杭州 310023)

摘 要: 随着语义 web 的迅速发展，基于本体的应用越来越多，本体推理机的应用也越来越为重要。本体推理机可用于推理和查询语义，是实现语义 Web 的关键技术之一。介绍了本体推理机及其分类、结构以及相关技术，介绍了几种典型的本体推理机；分析了本体推理机的应用及其发展的趋势。

关键词: 本体; 推理; 本体推理机

Ontology Reasoner and Its Application

PAN Chao, GU Hui

(School of Computer Science and Technology, University of Technology, Hangzhou 310023, China)

Abstract: With the rapid development of Sematic Web, application of ontology spring up and ontology reasoner application are widely used. Ontology reasoner is used to reason and query semanteme, and is one key technology in implementing Sematic Web. This paper introduces the techniques, classification, and structure of Ontology reasoner, and then introduces several Ontology reasoners. Lastly, this paper summarizes the application and the future work to be done on this topic.

Keywords: ontology; reasoner; ontology reasoner

1 引言

本体推理机是实现语义 Web 的关键技术之一。目前一些本体推理系统已经用于推理和查询语义 Web (Sematic Web)[1]，其中比较典型的有 W3C 对本体进行测试的本体推理机[2]，DIG 基于描述逻辑实现的本体推理机[3]，一些集成在语义网开发平台(如 HP 实验室的 Jena[4]、德国 Karlsruhe 大学的 KAON2[5])和本体管理系统(如 IBM 的 SNOBASE 系统)中的推理引擎。

各种本体推理系统所使用的推理算法和最优化技术存在差异。本文对当前一些主流本体推理机及其结构分类进行研究，分析了典型的推理技术及常用推理机，然后给出本体推理机的应用。。

2 本体推理机及其分类

推理是指依据一定的规则从已有的事实推出结论的过程。基于知识的推理则强调知识的选择和运用，完成问题求解。推理机(Inference Engine)常见于专家系统，它是对知识进行解释的程序，根据知识的语义，按一定策略找到的知识进行解释执行。

本体[6](Ontology)是共享概念模型的的形式化规范说明。这个定义包含了四层含义：概念模型、明确、形式化和共享。概念模型是指通过抽象出客观世界中的一些现象的相关概念而得到的模型，概念模型所表现得含义独立于具体的环境状态；明确指所使用的概念及使用这些概念的约束都有明确的定义；形式化指本体是能被计算机处理；共享指本体中体现的是共同认可的知识，反映的是相关领域中公认的概念集。即本体是对应用领域概念化的解释说明，为某领域提供了一个共享通用的理解，从而无论是人还是应用系统之间都能够有效地进行语义上的理解和通讯。一个非常简单的例子就是分类的层次结构，指明了类和它们之间的包含关系。

① 收稿时间:2009-12-28;收到修改稿时间:2010-01-28

本体推理机主要是针对本体进行推理，目前针对本体的推理，越来越多地集中在了几种标准的本体语言上，如 OWL[7]、DAML[8]、RDFS/RDF[9]等。

2.1 本体推理技术

目前而言，实现本体推理的主要方法有以下四种:

1) 基于传统描述逻辑[10]的推理方法。典型代表有 Pellet[11]、Racer[12]和 FaCT++[13]，它们都是基于传统 Tableaux 算法设计并实现的本体推理机，同时也引入了许多 Tableaux 算法的优化技术，从而使得它们的推理效率很高。

2) 基于规则的方法。本体推理作为一类应用，可以映射到规则推理引擎上进行推理。目前已经存在了许多现成的实现 OWL 到规则的转化工具。目前基于规则方法实现的本体推理机系统典型代表有 Jess[14]和 Jena。

3) 利用逻辑编程方法。基于演绎数据库(Deductive database)技术实现，典型的系统项目有 F-OWL[15]，德国卡尔斯鲁厄大学的 KAON2 也是一个采用这种技术方法实现的典型例子。

4) 基于一阶谓词证明器的方法。由于 OWL 声明语句能够很方便地转化为 FOL，因此也就可以很方便地利用传统的一阶谓词证明器实现对 OWL 的推理，例如 Hoolet[16]本体推理机就是利用了 Vampire 一阶谓词证明器来实现本体推理。

2.2 本体推理机分类及其基本应用

本体推理机系统的有各种分类方法。通常，按照本体推理机是否针对某些具体本体描述语言，将本体推理机划分为专用和通用两大类:

1)专用本体推理机。如 Racer 、FaCT++、Pellet 等就专用本体推理机，它们支持的主要本体语言有 RDFS，OWL 等，专用本体推理机效率较高，使用方便；只是它将推理能力限定在几种具体的本体语言上，较难进行扩展。

2) 通用本体推理机。典型的有 Jess，它是开放的，用户只需要提供不同领域的推理规则，Jess 就可以对不同领域进行推理。通用本体推理机效率低，也不能提供针对各种具体领域的优化能力，使得这种推理机的效率很难被优化。

推理机都应实现以下的两个基本推理功能:

1) 检查本体的一致性。保证本体一致性就是保证本体中已获得的类和个体逻辑上的一致性，检验实例是否与类、属性和个体的所有公理约束相冲突。

2) 得到隐含的知识。本体创建一般遵循在尽量简化本体的同时使得本体尽量包含足够多的信息。因为如果要在一个本体中声明出所有的语义关系，那么构建本体将是一件非常复杂而又繁琐的任务，也会导致本体过于庞大而难以处理；若本体设计简单，在实际应用中又需要本体中蕴含的语义信息，这时就需要本体推理机来获取本体中隐含的信息。

3 典型本体推理机

3.1 本体推理机的一般结构

一般地，本体推理机的系统结构由本体解析器、查询解析器、推理引擎、结果输出模块和 API 五大模块组成。其系统结构图见图 1[17]。

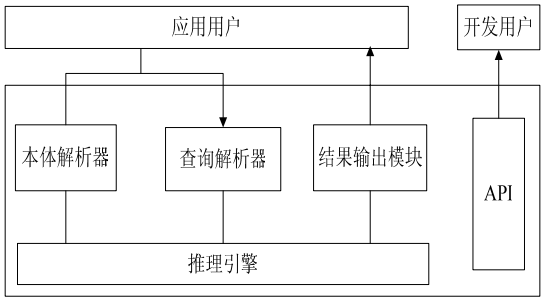

图 1　本体推理机的系统结构图

本体解析器：负责读取和解析本体文件，它决定了推理机能够支持的本体文件格式。解析性能的好坏决定了推理机是否支持对大本体文件的解析。

查询解析器：负责解析用户的查询命令。

推理引擎：是本体推理机的核心部件，负责接受解析后的本体文件和查询命令，并执行推理流程，推理引擎决定本体推理机的推理能力。目前大部分推理引擎是基于描述逻辑的。

结果输出模块：对推理引擎所推导出来的结果进行包装，满足用户的需求。它决定了本体推理机能够支持的文件输出格式。

API 模块：主要面向开发用户，一般有三部分，OWL-API、DIG（DL Implementation Group）接口以及编程语言开发接口。OWL-API 为用户操作 OWL 本体文件提供了一种标准接口。DIG 接口为描述逻辑推理机系统向外提供服务提供了一组标准的接口，作用类似于数据库中的 ODBC，它允许前端（如本体编

辑器 Protege）挂接到后台不同的推理引擎上。另外本体推理机提供的常见编程语言接口主要有 Lisp 和 Java 两种，大部分本体推理机是采用这两种编程语言实现的。

3.2 典型本体推理机系统

Jena 是由 HP Labs(http://www.hpl.hp.Com)开发的 Java 开发，是一种产生式规则的前向推理系统，针对本体的推理。Jena 框架包含一个本体子系统(Ontology Subsystem)，它提供的 API 支持基于 OWL，DAML+OIL 和 RDFS 的本体数据； Jena 提供了 ARQ 查询引擎，它实现 SPARQL 查询语言和 RDQL，从而支持对模型的查询。Jena 的主要特点是它开源的。

Pellet 是由美国马里兰大学 College Park 分校 MinSwap 实验室开发的一个本体推理机。是一个基于 Java 的开放源码系统，以描述逻辑作为理论基础，采用 Tableaux 算法。Pellet 是一个较完善的 OWL-DL 推理机，广泛支持个体推理，包括名义(nominal 枚举类)推理和合取查询，用户自定义数据类型和本体的调试支持。Pellet 主要应用在本体开发、发现和构建 Web Service 等方面。Pellet 效率较高，但是缺乏对本体规则语言 SWRL 的支持并且支持的本体查询语言不够全面；一般只是进行 A-Box 推理查询的时候考虑使用 Pellet，而查询如果牵涉到 T-Box 推理则推荐使用 Racer。

Racer 最初由德国的汉堡大学开发的基于描述逻辑系统的知识表达系统，采用 Tableaux 算法，它的核心系统是 SHIQ(描述逻辑的一种，它主要包含交、并、存在、任意、数量约束等构造算子)。Racer 也可以对基于 RDFS\OIL+DAML 和 OWL 知识库进行处理。它提供支持多个 TBox(术语公理)和 ABox(断言事实)的推理功能。给定一个 TBox 后，Racer 可以完成各种查询服务。Racer 具有较强的本体一致性检查功能，在 TBox 方面推理能力较强，能够对大本体文件提供良好的支持，而且具有图形用户界面，并有详细的开发文档和示例代码，但是 Racer 不支持对枚举类和用户自定义数据类型的推理。

Jess 是一个经过扩充的 CLIPS(C Language Integrated Production System)版本，由美国实验室分布式系统计算机组用 Java 实现的基于产生式的前向推理引擎。Jess 是性能良好的开放式推理机，采用经典的 Rete 算法，支持正向和逆向推理，在提供这个领域的相关规则和事实信息的前提下，原则上可以处理各种领域的推理服务。由于 Jess 是通用推理引擎，使得这种推理机制的效率很难优化。

KANO2 是管理 OWL-DL,AWRL,和 F-Logic 本体的基础设施，它由信息技术研究中心(FZI)的 IPE、卡尔斯鲁厄大学的 AIFB 和曼切斯特大学的 IMG 共同努力开发的。KANO2 是 KANO(也即 KANO1)的新一代产品，与 KANO1 最主要的不同在于支持的本体语言：KANO1 采用 RDFS 的扩展,而 KANO2 是基于 OWL-DL 和 F-Logic，KANO2 是一个完全新的系统。KANO2 的 API 能够处理 OWL-DL 本体、F-Logic 本体，可以读取 OWL-XML 语法和 OWL RDF 语法。在推理方面，KANO2 支持 OWL-DL 的子集 SHIQ(D)和 OWL Lite，同时还支持 SWRL 子集 DL-safe，和其它的推理机不同的是 KANO2 不采用 tableaux 算法。但是 KANO2 不能处理名义(nominals)，即通常所说的枚举类(enumerated classes)，也不能处理大批的集的势声明。

FaCT++ 是 FaCT(Fast classification of Terminologies)的新一代产品，是英国曼城斯特大学开发的一个描述逻辑分类器，提供对模型逻辑(model logic)的可满足性测试，采用了客户端/服务器模式。FaCT++采用 FaCT 的算法，二者均采用 tableaux 算法，但是内部结构是不同的。FaCT++是基于描述逻辑的推理机，支持 OWL DL 和 OWL 2 标准，为了提高效率和获得更好的平台移植性，FaCT++采用了 C++而非 FaCT 的 Lisp 语言来实现，这是 FaCT++和其它推理机不同的一个地方。FaCT++推理机在本体一致性检查上具有很好的表现。但是 FaCT++没有提供 OWL 接口也不支持对实例的查询。FaCT++也是开源的，但是开发文档和示例代码都不详细，而且没有友好的用户界面。

4 本体推理机的应用

随着语义 Web 的发展，基于本体的应用越来越多。由于本体开发本身具有分布式的特点，不同组织开发的本体可能覆盖相同或者相交的领域，它们存在一定程度上的相似性。也因此本体推理机的应用将会越来越广泛。

本体推理机主要有以下几个方面的应用：检测冲突、优化表达和实例的归类等，以获得本体中隐含的

知识和运用本体中的知识解决问题 。

下面用定义一个简单本体(部分代码)来说明本体推理机的作用。

```
<owl:Class rdf：ID="man">
  <owl:disjointWith>
     < owl:Class rdf: =" woman "/>
  < / owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="humans"/>
  </ rdfs:subClass0f>
</owl:Class>
<owl:Class rdf:ID="woman">
  <rdfs:subClassOf >
  <owl:Class rdf: ID ="humans"/>
```

```
<owl:Class rdf:ID=" mother ">
  <owl:ObjectProperty
rdf:ID="haschild"/>
  <rdfs:subClassOf>
     < owl:Class rdf: ="woman "/>
  </ rdfs:subClass0f>
</owl:Class>
 <owl:Class rdf:ID="father">
   <rdfs:subClassOf rdf: ID= "man"/>
</owl:Class>
<woman rdf:ID="Jerry"/>
```

(1) 检测冲突：本体建立者要想建立正确、一致的本体就需要借助推理，这一般由推理机完成。好的推理机能检查出本体中的冲突，一般包括实例体系的冲突和定义体系的冲突。例如上例中,Jerry 是 woman 的一个实例，后来又有其他本体建立者在这个本体中加入以下实例:

<man rdf:ID="Jerry"/>

这个代码声明了 Jerry 是 man 的是一个实例，因为 Jerry 在本例中已经做了 woman 的实例，而 man 和 womam 没有交集，这时就产生了不一致情况，形成了实例体系的冲突。

(2) 优化表达

例如，在本体建立的时候经常有以下情况：

<owl：Class rdf：ID=“mother”>

<owl：disjointWith>

<owl：Class rdf：ID=“man” />

</ owl：disjointWith>

<rdfs：subC1assOf>

<owl：Class rdf：ID=“humans” />

</ rdfs：subClassOf>

</ owl：Class>

这段代码定义了 mother 类和 man 类是没有交集的，还定义了 mother 是 humans 的子类，定义 mother 是 humans 的子类是多余的，因为在本例中已经 mother 是 woman 的子类,woman 是 humans 的子类，所以 mother 一定是 humans 的子类。在一个本体这样声明类似的关系，会使本体过于庞大，难以实现自动处理。

在实际应用中，类、属性和实例之间的关系是非常复杂的，推理机能够把这些错综复杂的关系整理清楚，用符合应用需求的格式组织本体中的信息，通常有用树结构来描述类和属性的层次关系，用图来表达实例之间的联系等等。这样在获取本体信息的时候就可以使用成熟的、低复杂度的算法，从而提高效率。

(3) 实例的归类：实例归类是指把一个已经描述好的实例，要归结到一个最能反应它特征的类中。推理机可以根据类具有的属性对实例进行归类处理。

例如有如下描述：

<woman rdf:ID="teacher"/>

<owl:Individual rdf:about=“lucy”>

<haschild rdf:resource= “teacher” / >

</owl:Individual>

加到刚才的本体中。这时交给推理机分类的话，因为 lucy 具有 haschild 属性，在本体定义中只有 mother 有 haschild 属性，所以在本体中描述 lucy 的具体类应该是:mother 类。

5 总结

目前大部分本体推理机都能够实现两大基本推理功能，但是也存在着一些不足，如 Racer 不支持对枚举类和用户自定义数据类型的推理，Pellet 支持的本体查询语言不够全面，由于 Jess 是通用推理引擎，但效率较低等。另外现在的本体推理机大多数用户界面不够友好等。因此，本体推理机的发展趋势应该是在功能方面，开发更完善的推理算法等，如支持用户自定义数据类型和逆关系属性类型等；允许用户自定义推理规则以及支持多个、多版本和不一致本体的推理；提高推理引擎的效率；向用户提供友好的图形用户界面和良好的程序开发接口；另外把本体推理机更好地集成到大型本体服务器中都是本体推理机应用的最新趋势。

参考文献

1 Bemers-Lee T, Hendler J, Lassila O. The Semantic Web.Scientific American, 2001,284(5):34－43.

2 OWL Test Results(Semi-Official Semi-Static View). [2006-9-1].http://www.w3.org/2003/08/owl-systems/test-results-out#systems.

3 Description Logic Reasoners.[2006-9-1].http://www.es.man.ae.uk/-sattler/reasoners.html.

4 Jena – A Semantic Web Framework for Java.[2009-4-9]. http:// jena. sourceforge.net/ KAON2.[2009-4-9]. http://kaon2. semanticweb.org/.
6 Studer R. Benjamins VR, Fensel D. Knowledge Engineering, Principles and Methods. Data and Knowledge Engineering ,1998 ,25(12):161 − 197.
7 OWL Web 本体语言指南. [2009-4-9]. http://zh. transwiki.org/cn/owlguide.htm.
8 DAML. http://baike.baidu.com/view/386985. htm?fr= ala0.
9 RDF 入门推荐标准.[2009-4-9]. http://zh.transwiki. org/cn/rdfpri- mer.htm.
10 Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider P. Description Logic Handbook, Cambridge press, 2003.01
11 Sirin E. Pellet:A practical OWL-DL reasoner. Journal of Web Semantics, 2006.
12 Haarslev V, Moller R. Racer: A Core Inference Engine for the Semantic Web.In Work-shop on Eval uation on Ontology-based Tools, the 2nd International Semantic Web, 2003.
13 Tsarkov D, Horrocks I. Description Logic Reasoner: System Description. IJCAR 2006:292 − 297.
14 Friedman-Hill EJ. Jess, The Rule Engine for the Java Platform[2003-11-21]. http:// herzberg.ca.sandia.gov/ jess/,21 November2003.
15 Zou Y, Finin T, Chen H. F-OWL: an Inference Engine for the Semantic Web. http://fowl.sourceforge.net/ 2003.
16 Tsarkov D, Riazanov A, Bechhofer S. Using Vampire to Reason with OWL,International Semantic Web Conference 2004:471 − 485.
17 徐德智,汪智勇,王斌.当前主要本体推理工具的比较分析与研究.现代图书情报技术, 2006,(12).

(上接第 207 页)

度。同时，根据语义锁中的语义信息，本文还提出了能够最大限度地减少企业经济损失的死锁解除机制，智能选择回滚事务，可以在 Microsoft .NET 平台上模拟多 Web 服务集成的业务流程，并且设计出基于语义锁的事务处理过程以及死锁解除算法来验证该方法的优越性，下一步我将编程实现这种方法，并且进一步分析实验结果在 Web 服务中的作用。

参考文献

1 BPEL4WS.[2009-10-24].http://www-128.ibm.com/ developerworks/library/ws-bpel/.
2 WSFL.[2009-10-24].http://www-306.ibm.com/ software/solutions/webservices/pdf/WSFL.pdf.
3 Tong L, Carla SE, Alvin RL, Danie JS. Pulse: A Dynamic Deadlock Detection Mechanism Using Speculative Execution. Proc. of the 2005 USENIX Annual Technical Conference Anaheim, California, April 10–15, 2005.
4 赵明霞. Web 服务集成的协调框架中的死锁检测机制研究[硕士学位论文].武汉:武汉大学, 2004.
5 Antoine G, Mathieu B. A resource-control model based on deadlock avoidance.[2009-9-20]. http://cdc.ioc.ee/ appsem04/webproc/indust/galland.pdf, 2004.
6 Jaehwan L, Vincent JM. A Novel Deadlock Avoidance Algorithm and Its Hardware Implementation. [2009-9-20].http://codesign.ece.gatech.edu/publications/jaehwan/paper/codes_2004.pdf, 2004.