基于SWRL规则的简单对等本体关联方法研究与实现

王若梅[1,2] 彭丽仪[1,2] 王 众[2] 陈 杰[1,2] 刘 林[2]
(中山大学计算机科学系 广州 510275)[1] (中山大学数字家庭教育部重点实验室 广州 510275)[2]

摘 要 提出的"基于SWRL规则的简单对等本体关联方法"通过对SWRL规则的应用和扩充实现多个简单对等本体之间在类上的对应关联,从而可以达到构建简单的本体并通过本体之间的关联来获取更大的、更多的知识的目的。本法为本体的重用和映射提供了有效的方法。

关键词 本体关联,SWRL规则,对等本体

SWRL-based Mapping Methodology for the Simple Peer-to-Peer Ontologies

WANG Ruo-mei[1,2] PENG Li-yi[1,2] WANG Zhong[2] CHEN Jie[1,2] LIU Lin[2]
(Department of Computer Science, Sun Yat-Sen University, Guangzhou 510275, China)[1]
(Key Lab of Digital Life(Sun Yat-sen University), Ministry of Education, Guangzhou 510275, China)[2]

Abstract This paper presented the SWRL based mapping methodology for the simple peer-to-peer ontologies, implementing the mappings and alignments among the classes belong to multiple simple peer-to-peer ontologies by applying and extending the SWRL rules. The methodology gives an effective way to reuse the ontologies.

Keywords Ontology mapping, SWRL rules, Peer-to peer ontology

1 概述:简单对等本体的关联

本体的构建采取的方式多是在领域专家的帮助下,手工获取本体中的概念、属性等词汇进行进一步的组织和建立关系。虽然近些年来对本体的自动、半自动获取的研究取得了一些进展,如通过知识、数据挖掘等手段实现本体的自动构建等,但是由于领域知识的范围不易界定,追求知识完备所带来的影响不但使得本体构建成为耗时和困难的过程,同时会造成本体规模越来越庞大,增加了维护难度。

对于上述问题,M. C. Rousset在2004年的第3届国际语义网大会(International Semantic Web Conference)上提出了一种基于"简单对等(peer-to-peer)本体"的设想。简单对等本体是指本体由少数最基本的概念、属性组成,本体与本体之间地位是平等的,可以相互访问[1]。要实现M. C. Rousset所提出的"对等本体"的设想,最关键的问题是实现对等本体之间"在类上的映射关系"。现有本体研究一般是通过计算本体间类的相关度来实现这种映射关系,但现有的相关度算法的计算量大、不完善和具有片面性的缺点[2]决定了现有方法无法有效地实现类上的映射关系。本文提出一种基于SWRL规则的简单对等本体关联方法,并给出了实际算例和分析结果。结果表明,使用本文提出的关联方法能有效实现对等本体间类上的映射关系。

2 基于SWRL规则的简单对等本体关联方法

2.1 SWRL简介

2.1.1 SWRL的背景

SWRL(Semantic Web Rule Language)是以语义的方式表示规则的一种语言,SWRL基于DAML[3]语言系列中DAML2Rule[3]或DAML2L[3]规则语言的一些初步设想。SWRL结合了OWL DL[4]和RuleML[5]的子语言Datalog[5]。SWRL是对OWL[6]的扩充,它使OWL语法中可以包含Horn子句[7]。

2.1.2 SWRL的抽象语法描述

SWRL规则的基本形式是表达前提和结论的推导关系,它保留了在RuleML中以结论(规则头部)表示推论结果、前提(规则尾部)表示推论前提的基本形式,其中的语义可以表达为:如果前提中表示的前提成立了,那么结论中表示的推论也必然成立。前提与结论都可以包括0个或者多个原子(基本命题),而多个原子之间是一种逻辑与的关系。规则中的原子可以是以下的形式:$C(x)$,$P(x,y)$,$sameAs(x,y)$或者是$differentFrom(x,y)$。其中,C是一个OWL的类描述(本文集中讨论这种形式的语义分析),P是一个OWL的属性,而x和y分别是变量,或者是OWL的实例,又或者是OWL的数据值[7]。W3C规范中为了丰富SWRL对规则的描述能力,提出一些内置(built-in)谓词,用于以后的规则扩展。本文的论述暂不考虑含有内置谓词的规则。

为了把要讨论的问题简化并基于以下规则,本文只讨论结论中只含一个原子的SWRL规则,如表1所列。一条结论为具有合取命题形式的SWRL规则可以被改写为结论为一个原子的多条SWRL规则,如(A,$C1$,$C2$分别是3个不同的

到稿日期:2008-04-29 本项目受国家自然科学基金(60473131)和广州市科技计划项目(2006Z1-D6131)支持。

王若梅 教授,主要研究方向为计算机辅助设计、计算机仿真、数据库与信息处理,E-mail:isswrm@mail.sysu.edu.cn;**彭丽仪** 硕士生,主要研究方向为语义网与本体应用;**王 众** 博士后,主要研究方向为热湿仿真与热舒适性、本体与语义网;**陈 杰** 硕士生,主要研究方向为语义网与本体应用。

类，x 是规则中唯一的变量)[8,9]：

$$A(?\ x)\text{->}C1(?\ x)\wedge C2(?\ x) \tag{1}$$

可以改写成以下两条规则：

$$A(?\ x)\text{->}C1(?\ x) \tag{2}$$

$$A(?\ x)\text{->}C2(?\ x) \tag{3}$$

表1 SWRL规则原子形式示例

原子形式	说明
C(x)	类谓词，它为真当且仅当x是C的实例
P(x,y)	属性谓词，它为真当且仅当x与y通过P相关
sameAs(x,y)	它为真当且仅当x和y是同一个对象
differentFrom(x,y)	它为真当且仅当x和y不为同一个对象

2.1.3 SWRL规则分析器

本文使用SWRL规则来实现简单对等本体的关联。对于SWRL规则的解释，构造一个“SWRL规则分析器”，并对SWRL规则进行了扩充，使它不仅能实现逻辑与，还能实现逻辑或和逻辑非的语义。分析器完成以下两个基本功能：1)分析SWRL规则的语义；2)根据上一步分析的结果执行相应的处理查询操作，并返回最终的结果。本文2.2节将阐述分析器如何对未扩充的SWRL规则进行语义分析，2.3节主要讲述对SWRL规则进行扩充的原理，2.4节阐述扩展后分析器是如何实现SWRL语义分析的。如上所述，一条结论为具有合取命题形式的SWRL规则可以被改写为结论为一个原子的多条SWRL规则，所以本文讨论的分析器规则分析对象仅为结论是一个原子的SWRL规则。

2.2 分析器对SWRL规则的语义分析

2.2.1 单条SWRL规则的语义分析

考虑一般性SWRL规则：

$$A(?\ x)\wedge B(?\ x)\rightarrow C(?\ x) \tag{4}$$

如果以一阶谓词逻辑的角度来分析以上的规则，可以得到如下的语义：

如果变量 x 是类 A 的实例，同时也是类 B 的实例，那么变量 x 也是类 C 的实例。

类在本体中代表着某个集合，而该集合的元素正是该类的实例。也就是说，上面SWRL规则的语义用集合论方法可以表达为：

如果变量 x 是类 A 所代表的集合的元素，同时也是类 B 所代表的集合的元素，那么变量 x 也是类 C 所代表的集合的元素。

下面用标记法 $SetX$ 表示类 X 所代表的实例集合，则式(4)可以表示为以下的式子：

$$SetA\cap SetB\subseteq SetC \tag{5}$$

由以上的分析可以得到对SWRL规则(4)的语义分析结果：对本体中某个类，如规则(4)中的类 C 的实例集的查询请求，实际上就是要求返回集合 $SetC$，则在分析器处理查询请求的时候，将求出的 $SetA$ 与 $SetB$ 的交集，作为 $SetC$ 的一部分返回。

2.2.2 一组SWRL规则的语义分析

对一组一般性SWRL规则：

$$\begin{cases}A(?\ x)\wedge B(?\ x)\rightarrow C(?\ x)\\ D(?\ x)\rightarrow C(?\ x)\\ E(?\ x)\wedge F(?\ x)\rightarrow C(?\ x)\end{cases} \tag{6}$$

由2.2.1节的分析，从集合论的角度，得到：

$$\begin{cases}SetA\cap SetB\subseteq SetC\\ SetD\subseteq SetC\\ SetE\cap SetF\subseteq SetC\end{cases} \tag{7}$$

$$\Rightarrow SetA\cap SetB\cup SetD\cup SetE\cap SetF\subseteq SetC$$

进一步可以得到对SWRL规则(6)的语义分析结果：对本体中某个类，如规则(6)中的类 C 的实例集的查询请求，实际上就是要求返回集合 $SetC$，那么在处理查询请求的时候，分析器将求出以下集合，作为 $SetC$ 的一部分返回。

$$S:=SetA\cap SetB\cup SetD\cup SetE\cap SetF \tag{8}$$

2.3 SWRL规则的语义扩展

在实际的应用当中，为了实现本体之间的对应关系，SWRL规则的撰写者极有可能需要描述类谓词之间比简单的逻辑与操作更为复杂的逻辑结合关系(根据文献[7]，SWRL规则前提只能包含逻辑与)，所以本文对SWRL规则的语义进行了扩充，使之能够在前提body中包含类谓词的逻辑非操作(¬)和类谓词之间的逻辑或操作(∨)。在一阶命题逻辑中，所有的谓词公式都可以转换成析取范式，所以在扩充的SWRL规则中，先把规则转换成析取范式[8](形如$(A(?\ x)\wedge\neg B(?\ x))\vee(\neg C(?\ x)\wedge D(?\ x))\vee E(?\ x)$)，再实现规则的语义分析。

本文论述的SWRL规则语义的扩充不是通过改变现有SWRL的语法以及语义的定义来实现的。在实际的应用当中，当想要表达对类谓词的逻辑非操作时，SWRL规则的撰写者不能在SWRL规则的写作上引入¬符号。本文采用如下的标记法来代替¬符号：

当需要表达对类谓词 $A(?\ x)$ 的逻辑非操作，

1)在OWL文件中建立一个辅助类 Not_Class；

2)建立一个类 Not_Class 的子类 Not_A，并把OWL提供的该类的备注的第一项写上类“A”，以此代替“$\neg A(?\ x)$”。

当分析器分析到类谓词 $Not\_A(?\ x)$ 时，判断类谓词中的类 Not_A 是类 Not_Class 的子类。于是，分析器就读出类 Not_A 备注中的第一项类“A”，并按照下面论述的 $\neg A(?\ x)$ 的语义对SWRL规则加以处理。对于逻辑或符号的处理将在2.4.2节中详述。

考虑类谓词命题公式：

$$\neg A(?\ x) \tag{9}$$

从一阶谓词逻辑的角度来分析，式(9)的语义是：

变量 x 不是类 A 所代表的集合的元素变量

如果采用集合标记法，式(9)可用集合的语言表示为

$$x\notin SetA \tag{10}$$

2.4 分析器对扩充的SWRL规则的语义分析

2.4.1 分析器对前提body中含有0个∨谓词的SWRL规则的分析

首先考虑析取范式的SWRL规则前提body中只含0个∨符号的情况。不失一般性，考虑SWRL规则：

$$\bigwedge_{y=0}^{n}A_y(?\ x)\wedge\bigwedge_{y=0}^{m}\neg B_y(?\ x)\rightarrow C(?\ x) \tag{11}$$

从集合论的角度，可以得到(A,B 的下标 $y\rightarrow j,k$，参看规则11)：

$$\bigcap_{y=0}^{n}SetA_y-\bigcup_{y=0}^{m}SetB_y\subseteq SetC \tag{12}$$

由于对本体中的某个类，如规则(11)中的类 C 的实例集

的查询请求，实际上就是要求返回集合 $SetC$。那么在分析器处理查询请求的时候，应该求出的 $\bigcap_{y=0}^{n} SetA_y$ 与 $\bigcup_{y=0}^{m} SetB_y$ 的差集，作为 $SetC$ 的一部分返回。

2.4.2 分析器对前提 body 中含有多个 ∨ 谓词的 SWRL 规则的分析

考虑前提中含有多个 ∨ 谓词的 SWRL 规则（A,B 的下标 $y \to j,k$，参看规则 11，z 不变）：

$$\bigvee_{z=0}^{p}\left(\bigwedge_{y=0}^{n_z} A_{zy}(?\ x) \wedge \bigwedge_{y=0}^{m_z} \neg B_{zy}(?\ x)\right) \to C(?\ x) \tag{13}$$

规则(13)可以转换成一组 SWRL 规则：

$$\begin{cases} \bigwedge_{y=0}^{n_0} A_{0y}(?\ x) \wedge \bigwedge_{y=0}^{m_0} \neg B_{0y}(?\ x) \to C(?\ x) \\ \bigwedge_{y=0}^{n_1} A_{1y}(?\ x) \wedge \bigwedge_{y=0}^{m_1} \neg B_{1y}(?\ x) \to C(?\ x) \\ \cdots \\ \bigwedge_{y=0}^{n_p} A_{py}(?\ x) \wedge \bigwedge_{y=0}^{m_p} \neg B_{py}(?\ x) \to C(?\ x) \end{cases} \tag{14}$$

由于规则(14)中的每条规则都是形如规则(11)的 SWRL 规则，则由 2.3.1 节的分析可以得到

$$\begin{cases} \bigcap_{y=0}^{n_0} SetA_{0y} - \bigcup_{y=0}^{m_0} SetB_{0y} \subseteq SetC \\ \bigcap_{y=0}^{n_1} SetA_{1y} - \bigcup_{y=0}^{m_1} SetB_{1y} \subseteq SetC \\ \cdots \\ \bigcap_{y=0}^{n_p} SetA_{py} - \bigcup_{y=0}^{m_p} SetB_{py} \subseteq SetC \end{cases}$$

$$\Rightarrow \bigcup_{z=0}^{p}\left(\bigcap_{y=0}^{n_z} SetA_{zy} - \bigcup_{y=0}^{m_z} SetB_{zy}\right) \subseteq SetC \tag{15}$$

在这种情况下，分析器应该求出以下集合，作为 $SetC$ 的一部分返回：

$$\bigcup_{z=0}^{p}\left(\bigcap_{y=0}^{n_z} SetA_{zy} - \bigcup_{y=0}^{m_z} SetB_{zy}\right) \tag{16}$$

基于本节的分析可以看出，一条前提(body)中含多个 ∨ 符号的 SWRL 规则可以转换成一组结论(head)相同而前提(body)中只含 0 个 ∨ 符号的 SWRL 规则。经过转换后的每条规则，分析器可以使用 2.4.1 节的方式进行语义解释。

3 基于 SWRL 规则的本体关联方法用例分析

3.1 测试用例说明与实现

为了测试 SWRL 规则分析器实现简单对等本体关联方法的可行性，即是否可以实现简单对等本体间类的关联，本文设计了一个简单的测试用例：建立 3 个简单的对等本体 DemoA，DemoB 和 DemoC，每个本体里都含有一个或几个类（由 AClass，BClass，CClass，DClass 和 EClass 组成），当应用程序需要向某个本体查询某个类的实例时，经过上述分析器对该本体中 SWRL 规则进行分析后，返回要查询的类的实例以及该本体查找该类的实例而访问过的本体。

本文所构建的 3 个简单对等本体，相互之间的地位是平等的，不存在主次之分，两两之间可以相互访问、查询，并且每个本体都只包含部分的、简单的信息，符合了简单对等本体的要求。

3 个演示本体分别简单介绍如下：

· 本体 *DemoA*，有类 *AClass* 和 *DClass*，分别有实例集 $\{a,b,c,d,e\}$ 和 $\{f,g\}$，本体 *DemoA* 通过 SWRL 规则集与本体 *DemoB*、本体 *DemoC* 关联。

$$\begin{cases} BClass(?\ x) \wedge \neg CClass(?\ x) \to AClass(?\ x) \\ \neg DClass(?\ x) \wedge EClass(?\ x) \to AClass(?\ x) \end{cases} \tag{17}$$

· 本体 *DemoB*，有类 *BClass*，有实例集 $\{b,c,d\}$，本体 *DemoB* 通过 SWRL 规则

$$CClass(?\ x) \to BClass(?\ x) \tag{18}$$

与本体 *DemoC* 关联。

· 本体 *DemoC*，有类 *CClass* 和 *EClass*，分别有实例集 $\{c,d,e,f\}$ 和 $\{g,h\}$，本体 *DemoC* 通过 SWRL 规则

$$AClass(?\ x) \wedge DClass(?\ x) \to EClass(?\ x) \tag{19}$$

与本体 *DemoA* 关联。

本文使用 Web service 来实现分析器提供的查询本地本体的接口，将 Tomcat 服务器与相应的本体 OWL 文件部署在同一台机器上面，再把分析器作为部署在 Tomcat 服务器上的一个组件，其它本体通过 Web service 公开自己的本体的内容。测试用例的 3 个演示本体及其本体所携带的 SWRL 规则都采用 Protégé-OWL[10,11] 编辑器构建。

根据前面所述的本体关联方法，若要查询本体 *DemoA* 中类 *AClass* 的实例，则分析器具体处理流程和中间结果集，如表 2 所列（仅列出了查询 *DemoA* 的具体流程，查询 *DemoB*，*DemoC* 的具体流程雷同）。

表 2 用例本体 DemoA 查询处理流程

流程顺序	具体的操作	待返回结果集	待接受中间结果集	待拒绝中间结果集	中间查询过的本体
1	查询本地本体 DemoA，得到类 AClass 的本地实例集{a,b,c,d,e}，并把之作为“待返回的结果集”	{a,b,c,d,e}	null	null	DemoA
2	分析规则 BClass(? x) ∧ ¬CClass(? x)→AClass(? x)	{a,b,c,d,e}	null	null	
3	查询关联本体 DemoB 得到类 BClass 的实例集{b,c,d,e,f}，并把之作为“待接受的中间结果集”	{a,b,c,d,e}	{b,c,d,e,f}	null	DemoB
4	查询关联本体 DemoC，得到类 CClass 的实例集{c,d,e,f}，并把之作为“待拒绝的中间结果集”	{a,b,c,d,e}	{b,c,d,e,f}	{c,d,e,f}	DemoC
5	求出“待接受的中间结果集”相对于“待拒绝的中间结果集”的差集{b}，并把之并入“待返回的结果集”中	{a,b,c,d,e}	null	null	
6	分析规则 ¬DClass(? x) ∧ EClass(? x)→AClass(? x)；	{a,b,c,d,e}	null	null	
7	查询本地本体 DemoA，得到类 DClass 的本地实例集{f,g}，并把之作为“待拒绝的中间结果集”	{a,b,c,d,e}	null	{f,g}	DemoA
8	查询非本地本体 DemoC，得到类 EClass 的实例集{g,h}，并把之作为“待接受的中间结果集”	{a,b,c,d,e}	{g,h}	{f,g}	DemoC
9	求出“待接受的中间结果集”相对于“待拒绝的中间结果集”的差集{h}，并把之并入“待返回的结果集”中	{a,b,c,d,e,h}	null	null	
10	返回“待返回的结果集”{a,b,c,d,e,h}	{a,b,c,d,e,h}	null	null	

程序的运行结果如图 1 所示。

1）返回的类 *AClass* 的实例集为 $\{a,b,c,d,e,h\}$，是本地本体 *DemoA* 中类 *AClass* 的实例集 $\{a,b,c,d,e\}$ 的一个超集，

（下转第 188 页）

For Support Vector Machine[A]// Proceedings of the First International Conference on Machine Learning and Cybernetics. Beijing, November 2002: 4-5

[9] Mangasarian O L, Musicant D R. Successive overrelaxation for support vector machines[J]. IEEE Tangasarian on Neural Networks, 1999, 10: 1032-1037

[10] Vapnik V N. Statistical Learning Theory[M]. New York: Wiley, 1998

(上接第 128 页)

证明该关联方法确实通过 SWRL 规则关联了另一简单对等本体；2) 分析器根据 SWRL 规则进行相关的查询操作，并记录了整个查询过程中访问到的本体，其中 count 是查询深度，即查询某个类的实例时最多可关联的其它类的个数。本程序设定 count 为 2，此值可根据具体情况进行修改。

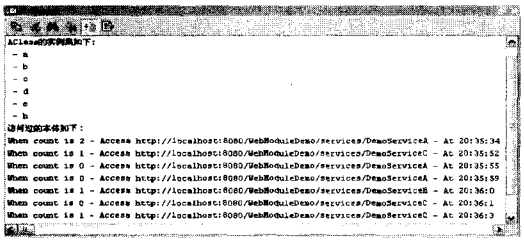

图 1 程序运行结果图

3.2 结果分析与讨论

3.2.1 SWRL 规则扩展的有效性

在本实验中，分析器对形如$\neg CClass(?\ x)$的规则的语义能正确解释，把查询返回的实例$\{\{c,d,e,f\}\}$作为待拒绝的结果集。这使扩展后的 SWRL 规则基本上能支持类间的逻辑与、逻辑或和逻辑非操作，符合更多具体应用的要求。用于测试的本体应用程序最终得到的类 *AClass* 的实例集是$\{a,b,c,d,e,h\}$，与推论出的最终被返回的"待返回的结果集"$\{a,b,c,d,e,h\}$相同。表明基于 SWRL 规则的本体关联方法能够实现简单对等本体之间类的映射。

3.2.2 本体自动动态更新

实验中返回的 *AClass* 的实例集与初始给出的 *DemoA* 中 *AClass* 的实例集并不相同，这是因为分析器根据用户编写的 SWRL 规则关联了其它的简单对等本体，从关联的本体中进行类 *AClass* 的实例查询，从而获得并返回更完整的类 *AClass* 的实例和信息。

测试用例中的规则分析器实现了对 SWRL 规则的语义分析，是一个基于一阶谓词逻辑的 SWRL 规则的推理机。在测试用例的实现过程中，不同本体之间是一种对等的关系，每个本体只存放了高一层大本体的部分知识，本体的构建者只需要维护和处理自己的本体以及不同本体间映射关系的构建。

3.2.3 基于关联的重用机制

如果本体的使用者想重用已有本体中构建的概念(类)，或者从已有本体中查找出与自己的本体相关的概念(类)知识，那么我们不需要使用相关度计算来实现本体之间概念(类)的映射。从上述测试用例看出，可以通过 SWRL 规则去关联不同的本体中的类，实现异构本体间类的通信，从而从不同的角度去解决本体中类的重用问题，而且这种方法并不针对某一领域本体，具有灵活性和通用性。

结束语 本文提出的"基于 SWRL 规则的本体关联方法"，对以下几方面问题进行了初步研究：通过 SWRL 规则实现了多个本体之间在类层面的对应关系，由此实现了在"简单对等(peer-to-peer)本体"设想中简单本体之间的关联；理论上，根据本体的一般定义，SWRL 规则支持的一般逻辑谓词的语义可以被解释为有名的类或者是匿名的类[7]，类是本体中的重要组成部分，开发一个本体的基本过程是围绕着类来进行的[12]。本文即是利用 SWRL 规则，从集合论的角度分析了 SWRL 规则的语义，对其语义基于查询结果进行了扩展，并设计了"SWRL 规则分析器"实现之。在现阶段，本文提出的"基于 SWRL 规则的本体关联方法"实现的只是本体之间在类上的对应关系。在后续的研究中，可以通过增加相应的把 SWRL 规则支持的一般逻辑谓词的语义解释为类的功能，使得本文提出的"SWRL 规则分析器"能够处理更加一般的由 SWRL 规则描述的本体之间的关联。

参 考 文 献

[1] Rousset M - C. Small Can Be Beautiful in the Semantic Web. 2004. http://www.springerlink.com/(qqu30yz2tpautoadmmfoax45)/app/home/contribution.asp?referrer=parent&backto=issue,2,57

[2] 郑丽萍. 本体映射的研究. 硕士学位论文. 山东科技大学, 2005

[3] Office, D. s. I. E. The DAML Rules. 2004. http://www.daml.org/swrl

[4] W 3 C. OWL Web Ontology Language Semantics and Abstract Syntax. 2004. http://www.w3.org/TR/2004/REC-owl-semantics-20040210

[5] Boley H B G, Tabet S. Rule Markup Tutorial. 2005. http://www.ruleml.org/papers/tutorial-ruleml-20050513.html

[6] Smith M K, Deborah C W, McGuinness L. OWL Web Ontology Language Guide. 2004. http://www.w3.org/TR/owl-guide

[7] Horrocks I, Harold Boley P F P-S, Tabet S, et al. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. 2004. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521

[8] 左孝凌，李为监，刘永才. 离散数学. 第一版. 上海：上海科学技术文献出版社，1982

[9] Ian Horrocks P F P-S, Bechhofer S, Tsarkov D. OWL rules: A proposal and prototype implementation. Web Semantics, 2005: 23-40

[10] Knublauch H, et al. The Protege OWL Plugin: An Open Development Environment for Semantic Web Applications. 2004

[11] Horridge M, et al. A Practical Guide To Building OWL Ontologies Using The Protege-OWL Plugin and CO-ODE Tools Edition 1.0. 2004. http://protege.stanford.edu/doc/users.html

[12] Vet P E v d a N J I M. Bottom-Up Construction of Ontologies. 2002. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=706054