自定义SWRL知识图谱推理补全插件的实现

陈 光[1,2,3,4],蒋同海[1,3],王 蒙[1,4],唐新余[1,4],季文飞[1,2,4]
1.中国科学院 新疆理化技术研究所,乌鲁木齐 830011
2.中国科学院大学,北京 100049
3.新疆民族语音语言信息处理实验室,乌鲁木齐 830011
4.江苏中科西北星信息科技有限公司,江苏 无锡 214135

摘 要:知识图谱是人工智能应用的基石,基于规则进行推理是知识图谱知识补全的重要方式。SWRL推理插件的局限性成为了知识推理补全的瓶颈。打破了SWRL有限的推理能力,论述了在SWRL规则中编写自定义知识推理插件,并在知识图谱建模和推理工具中实现对自定义插件推理支持的方法。介绍了知识图谱知识建模和推理的方法与工具,结合一个具体的知识补全需求建模了包含自定义知识推理插件的SWRL推理规则;在Pellet推理机中实现和注入了此自定义推理插件的推理支持源码,并通过与Protégé知识建模工具进行集成从而完成知识补全需求;应用包含自定义插件的SWRL推理规则完成了老人健康小屋物联网系统资源组成和资源故障诊断的知识补全。以此论述了使用SWRL自定义知识推理插件进行知识图谱知识补全的方法和实践。

关键词:知识图谱;知识补全;推理;SWRL语言;Protégé知识建模工具;Pellet推理机;自定义插件
文献标志码:A **中图分类号:**TP391 doi:10.3778/j.issn.1002-8331.1910-0342

Custom SWRL Knowledge Graph Completion Reasoning Built-ins Implementation Method

CHEN Guang[1,2,3,4], JIANG Tonghai[1,3], WANG Meng[1,4], TANG Xinyu[1,4], JI Wenfei[1,2,4]
1.Xinjiang Technical Institute of Physics & Chemistry, Chinese Academic of Sciences, Urumqi 830011, China
2.University of Chinese Academy of Sciences, Beijing 100049, China
3.Xinjiang Laboratory of Minority Speech & Language Information Processing, Urumqi 830011, China
4.Jiangsu CAS Nor-West Star Information Technology Co., Ltd., Wuxi, Jiangsu 214135, China

Abstract: Knowledge Graph(KG) is the cornerstone of AI application, and rule-based reasoning is still an important way to complete knowledge in KG. The limitation of SWRL reasoning built-ins has become the bottleneck of knowledge reasoning and complete. This paper breaks the limited reasoning ability of SWRL, and discusses the method of compiling custom knowledge reasoning built-ins in SWRL rules, and realizing reasoning support for these plug-ins in knowledge graph modeling and reasoning tools. Firstly, the paper introduces the methods and tools of knowledge modeling and reasoning in KG, and establishes SWRL reasoning rules including custom reasoning built-in with a specific knowledge completion requirement. Then, the reasoning support source code of this custom reasoning built-in is implemented and injected into Pellet reasoner, and is integrated with Protégé knowledge modeling tool to fulfill the requirement. Finally, the SWRL inference rules including the custom plug-ins are applied to complete the knowledge completion of the resource composition and resource fault diagnosis of the Internet of things system in the elderly healthy cabin. In this way, it discusses the method and practice of using SWRL custom reasoning built-ins to complete knowledge graph.

Key words: knowledge graph; knowledge complete; reasoning; Semantic Web Rule Language(SWRL); Protégé knowledge modeling tool; Pellet reasoner; custom built-ins

基金项目:无锡市科技发展资金项目(N20191005)

作者简介:陈光(1988—),通信作者,男,博士研究生,主要研究方向为物联网系统技术与应用、物联网体系结构建模与模型检测、模式识别和人工智能,E-mail:cheng@ciotc.org;蒋同海(1963—),男,博士,教授,博士生导师,主要研究方向为计算机网络与流量技术、大数据技术、物联网技术;王蒙(1980—),男,副研究员,主要研究方向为信息处理、大数据融合、大数据清洗;唐新余(1975—),男,博士,副研究员,硕士生导师,主要研究方向为数据参考技术、数据挖掘、大数据清洗;季文飞(1991—),男,博士研究生,主要研究方向为知识图谱技术、大数据融合、大数据清洗。

收稿日期:2019-10-23 **修回日期:**2020-02-16 **文章编号:**1002-8331(2021)01-0261-10

知识图谱是由具有属性的实体通过关系边进行连接而形成的图型数据结构[1-2],其中图的节点代表实体(Individual)或者概念(Concept),而图的边代表实体或者概念之间的关系(Property)。知识图谱的本质是一种揭示实体之间关系的语义网络,是对现实世界的事物及其相互关系进行结构化描述而形成的知识库[1-2]。在人工智能领域,知识图谱逐渐成为知识表示、知识链接和机器认知的重要手段,在智能搜索、机器翻译、机器理解、自然语言问答中发挥着重要作用[2],并在大数据风控、推荐系统、金融以及教育等领域应用中得到了广泛应用[1,3]。

知识图谱技术经历了语义网络、描述逻辑和本体论以及开放链接数据[4]等发展阶段。当前主流的知识图谱表达方式[3]是由W3C制定的基于资源描述框架RDF[5](Resource Description Framework)的网络本体语言OWL[6](Web Ontology Language)。由于知识图谱能够提供高质量的结构化知识数据,因此成为了人工智能应用的基石。但由于大部分的开放知识图谱,例如Freebase[7]和DBPedia[8]等,都是由人工或者使用半自动的方式构建的,导致这些图谱中的实体关系通常比较稀疏,大量实体之间的关系没有被充分的挖掘出来。因此,对知识图谱中的实体及其关系进行知识补全,从而发现潜在的知识,丰富和扩大知识图谱的知识含量是知识图谱的一个重要研究方向。

面向知识图谱的知识推理是指根据知识图谱中已有的知识,采用某些方法,推理出新的知识或识别知识图谱中错误知识的技术,是进行知识图谱补全和知识图谱去噪的主要手段[9]。知识推理的方法包含了基于规则的推理、基于分布式表示的推理、基于神经网络的推理以及混合推理[9]。其中基于推理规则进行知识推理,仍是知识图谱知识补全的重要手段之一。鉴于推理规则在知识补全中的重要性,W3C在本体描述语言OWL[6]的基础上,定义了专门用于描述本体推理规则的语言SWRL[10](Semantic Web Rule Language)。

基于SWRL进行推理推理表述的一大特色是能够使用SWRL的插件原语(Built-ins Atom)语法来拓展SWRL知识推理规则的表述能力,SWRL官方已经定义了一些插件[10](Core Build-ins),这些插件能被主流的推理机如Pelle[11]或者Hermit[12]等支持,从而极大地丰富了SWRL在数学计算、字符串处理和时间处理等方面的推理规则表述能力。

但是由于这些SWRL插件是官方为通用推理场景设计的,因此不能很好地满足丰富的知识图谱应用场景下不断拓展的具体知识推理规则的建模表述需求,使得推理插件的自定义性十分局限。特别是与实时信息获取相关的推理需求,官方定义的SWRL推理插件不能很好的进行支持,甚至无法通过插件取得当前系统的实时工作时间。这为一些需要依赖当前时间判断系统组件状态的推理场景造成了很大的不便。SWRL官方推理插件的局限性成为了基于推理规则进行知识图谱知识补全的瓶颈。探索一种将SWRL知识推理规则中应用的插件进行进一步拓展,使其能够根据具体的推理需求进行个性化自定义,并实现主流推理机对自定义插件推理支持的方法,是本文的研究动机。

本文首先对知识图谱知识表示和建模的相关知识进行论述,介绍本体知识图谱的建模工具和知识推理工具。然后以完成人类本体中人的年龄知识补全的需求为例,探讨了基于SWRL插件原语语法建模包含自定义推理插件原语推理规则的建模方法,并通过集成经过注入自定义插件实现源码的Pellet推理机到Protégé知识图谱建模工具,实现自定义SWRL知识推理插件的推理支持。最后,结合一个具体的智慧养老老人健康小屋知识图谱知识推理补全需求,应用基于自定义插件实现的SWRL推理规则,实现了健康小屋物联网系统的实时工作资源组成和资源工作状态知识推理。从而论述了基于自定义的SWRL拓展推理插件进行知识图谱推理补全的实现方法以及在具体知识图谱推理补全中的应用实践。

1 相关工作

在应用SWRL进行本体推理研究方面,文献[13]将SWRL推理规则应用到故障树推理领域,将故障树中事件之间的逻辑关系转化成SWRL推理规则,最后通过Jess推理引擎进行推理,挖掘出故障树中的隐含知识。文献[14]在物品的认知及推理研究中,通过制定SWRL规则实现了物品功能属性和操作动作的自动获取。以上研究在其推理规则编写中,只用到了基础的类表达式原语和属性表达式原语就完成了建模的需求,并没有对SWRL的建模能力进行进一步的拓展。文献[15]侧重于应用SWRL的拓展查询规则语言SQWRL进行知识本体的查询检索,并实现了一个知识查询检索框架和界面,但并不涉及知识推理补全和内容。文献[16]将SWRL应用到城市公交线路途经站点的知识推理补全中,证明了通过SWRL推理可以对本体中蕴含的知识进行有效的推理,改善和优化知识查询的结果,但是其仅仅是基于SWRL的基础原语进行SWRL规则建模,而没有在其应用场景中应用到一些SWRL中不存在的特殊推理插件原语。文献[15]在基于本体的知识库研究中提到了SWRL内置原子(Built-ins)可以通过开放式假设增加新的概念定义提升本体的推理能力并拓展本体知识库,但是并没有给出明确的实现新增推理概念以提升SWRL推理能力的具体方法。

本文在OWL本体语法规则和SWRL规则的基础上,在第2章中给出了基于本体推理规则进行推理需要

使用的推理工具以及推理方法。在第3章中详细讨论了SWRL自定义插件的实现方法,并在第4章中详细的给出了支持自定义SWRL插件原语的推理机在本体建模工具中集成方法。从而详细的描述了在知识图谱中建模自定义SWRL插件原语并在推理机中实现推理支持的方法,丰富了SWRL的规则建模表述和推理能力。

在应用SWRL推理规则进行本体物联网推理的研究方面,文献[17]侧重于将不同本体物联网模型进行语义协同,从而为用户提供满足其偏好的特殊服务,但没有提供物联网本体模型的建模方法和建模实例。文献[18]通过实验验证了本体物联网模型结合推理技术获得更加丰富的蕴含知识以更好地满足用户信息查询的可能性。但在时间推理和多事件联合推理的实现方法上表述的不够明确,虽然提到使用了Jena的推理机和时间推理链的算法,但是对于使用何种技术基于编程还是推理规则实现此推理链没有明确的说明。

本文在第5章应用SWRL进行知识图谱补全的实践中,提出了一个明确的物联网系统本体维度体系结构建模的自动建模框架,建模了一套用于进行物联网系统资源和行为自动分类、组成检测和故障诊断的包含自定义推理插件的SWRL推理规则,基于物联网系统实时本体模型进行物联网系统知识图谱的自动知识推理补全。从实践的层面上对所提出的自定义SWRL知识图谱推理补全插件实现方法进行了验证。

2　预备知识

本体知识图谱建模理论经历了一个不断发展完善的过程,并且到目前还在持续发展的进程中。涉及的内容广泛而丰富,由于篇幅限制,本文中不会将具体的建模理论语法规则一一列出,但是会给出较为官方的理论规则介绍文献引用。

早在2000年,万维网之父蒂姆·伯纳斯-李(Tim Berners-Lee)就提出了语义网的概念和层次结构[19]。在URI[20]和XML[21]的基础上,W3C于2004年正式发布了第一个基础本体建模语言RDF[5]。

RDF确定了以三元组的形式来表示实体资源(Resource)与资源之间的连接关系(Property),成为了语义网和知识图谱进行知识实体与关系表示的基础。直到现在,主流知识图谱和语义本体中的原子知识表示都是以三元组的形式存在的。但RDF的表述能力十分有限,并不具备表达实体的概念类和类属性的能力,只能算作原始的本体语言。

为了进一步拓展本体的建模表述能力,在继承RDF语法的基础上,W3C制定了面向语义网的本体语言OWL[6],OWL拓展了表述基本类(Simple Class)、个体(Individuals)、属性(Simple Properties)以及属性特征(Property Characteristics)和属性约束(Property Restrictions)的建模能力。OWL还定义了本体映射(Ontology Mapping)的描述类规则,使得本体建模能够在已有类和属性的基础上,以等价(Equivalence)的形式衍生新的类或属性。

在RDF的基础上引入本体(Ontology),极大地丰富了知识图谱本体建模的表述能力,但OWL仍然不能表述"IF-THEN"逻辑的知识产生式规则,仅仅依靠OWL进行知识表示的能力有限。比如:即使已经有基本类"人"(Person)和数据属性"有年龄"(hasAge),仍然不能通过本体映射表达"属猪的人"(PigSignPerson)这样的概念。为了表述"属猪的人(在2019年)是年龄对12取余为0这样的概念",仍需要进一步拓展OWL的建模表述能力。为此,W3C基于RuleML[22]定义了SWRL[10]。

SWRL[10]通过描述本体建模中的公理(Axioms)使得OWL描述语言的表述能力进一步得到扩展,一个SWRL推理规则分为前驱(Antecedent)和结论(Consequent)两部分,两部分都由若干个无序的原语(Atom)组成。在本体建模中能够同时创建多个SWRL规则,规则之间构成逻辑或的关系,规则内的原语之间构成逻辑与的关系。SWRL原语可以分为以下几类[23]:类表达式原语(Class Expersion Atom)、属性表达式原语(Property Expersion Atom)、数据范围限制原语(Data Range Restriction Atom)和SWRL核心插件原语(Core Build-ins Atom)。其中核心插件原语(Core Build-ins Atom)是通过由W3C官方定义的一系列核心拓展插件(Core Built-ins)来进一步拓展SWRL规则的建模表述能力,这些插件在SWRL的插件原语(Built-ins Atom)中使用,从而丰富了SWRL在数学计算,字符串处理和时间处理等方面的推理规则表述能力。

基于OWL使用SWRL已经能够对知识图谱本体建模中的很多概念以及推理规则进行表述,但SWRL仍然存在一定的局限性。比如,虽然SWRL通过时间拓展插件[10](Built-ins for Date,Time and Duration)能够很好地处理与时间相关的操作,但是却没有办法获得当前的时间,而在很多知识图谱建模场景下,都需要获得当前的时间来判断一些实体的实时状态。为此,需要再进一步拓展SWRL的表述能力,实现一些自定义的SWRL推理拓展插件,从而更好地满足更广泛的知识图谱推理补全应用需求。SWRL从语法上支持用户自定义插件,使用与Core Build-ins一样的插件语法进行拓展,为了加以区分,将这类用户自定义原语称为自定义插件原语(Custrom Build-ins Atom)。自定义插件的使用方式与核心拓展插件(Core Build-ins)的使用方法完全一致,在SWRL插件原语中进行声明。

为了更好地说明使用自定义SWRL插件进行知识图谱建模和知识推理的方法,以一个简单的人(Person)类本体知识图谱中,通过自定义插件补全人的年龄的例

子进行说明。鉴于RDF和OWL语法规则不是本文的论述重点,为了提高建模和说明的效率,使用经典建模工具Protégé[24]对人类本体知识图谱进行建模如图1所示,Protégé的菜单功能介绍和详细使用方法可以参考文献[25]。

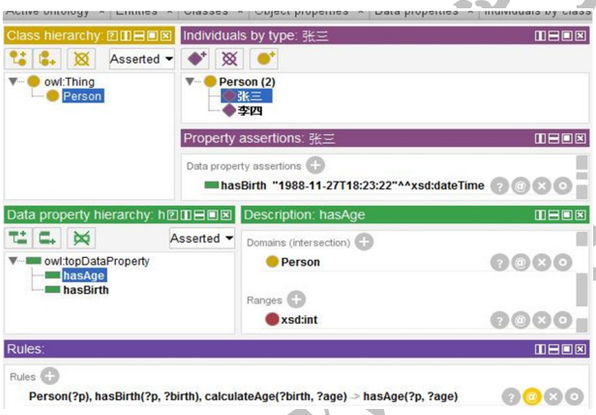


图1 使用Protégé可视化建模本体“人”

在图1所示的本体知识图谱模型中,创建了人的概念(Person),并创建了2个人的个体实例张三和李四。定义了2个数据属性(Data Property)分别表示“有年龄(hasAge)”和“有生日(hasBirth)”,并规定其定义域均为Person,值域分别为标准整形和日期数据类型。然后为张三和李四分别添加生日,张三的生日为1988-11-27日。希望根据当前的年份通过知识推理的方式,动态地计算并补全张三和李四的hasAge属性。因此基于SWRL的语法,定义了一个补全年龄属性的SWRL推理规则如图1下侧所示。

在这个推理规则中,首先通过类表达式原语和属性表达式原语取出一个人的生日?birth。然后将生日装入一个自定义插件原语calculateAge,此插件原语接收人的生日?birth并根据当前的系统时间计算出年龄?age。最后在此SWRL规则的结论部分,通过属性表达式原语hasAge将计算得到的年龄赋值给对应的人,从而完成人的年龄属性的知识补全。

需要特别注意的是Protégé 5.5版本的SWRLTab页面目前不能很好地支持自定义插件原语的录入,这些自定义插件原语是通过修改对应的基础本体OWL文件,基于SWRL的RDF/XML语法规则,以手动的方式修改录入的。Protégé 5.5可以展示出这些插件原语,但无法进行修改和保存。

另外SWRL只对本体中的推理规则进行描述,但并不实际执行这些规则的推理,如图1的模型中可以看出,个体张三只有hasBirth而没有hasAge属性,这是由于基于OWL语法规则和SWRL推理规则的推理执行都是由推理机完成的。知识图谱本体模型的常见推理机包含了Jess[26]、Racer[11]、Fact++[11]、Hermit[12]、Jena[27]和Pellet[11]等等。Protégé 5.5已经自带了Hermit推理机,通过“File”菜单中的“Check For Plug-in”还可以安装其他的推理机包括Pellet。

不同的推理机对于OWL语法规则和SWRL推理规则的支持各不相同,甚至对于SWRL中不同类型的组成原语的支持也有所差别。各个推理机对于OWL+SWRL的推理支持情况如表1所示。其中“未知”表示没有找到相关的文献或者实验方法,“编码支持”表示需要修改推理机源代码并重新编译打包才能够支持。另外Jena不能够直接支持SWRL,但是提供了另外一种形式的推理规则表述方法JenaRules,并提供了API进行推理规则的推理支持,规则的录入和推理都依赖于编码,并且不能与Protégé进行集成。

表1 推理机对本体推理规则支持情况表

推理机	OWL Ontology Rules	Class & Property Experation Atom	DataRange Restriction Atom	Core Build-ins Atom	Custrom Build-ins Atom
Jess	不支持	支持	支持	支持	未知
Racer	支持	不支持	不支持	不支持	不支持
Fact++	支持	不支持	不支持	不支持	不支持
Jena	支持	不支持	不支持	不支持	不支持
Hermit	支持	支持	不支持	不支持	不支持
Pellet	支持	支持	支持	支持	编码支持

本文选择能够支持用户自定义插件原语的Pellet推理机进行自定义插件实现。原生的Pellet推理机不能直接支持自定义插件的推理,需要通过修改推理机源代码的方式,使用Pellet插件注册类注入自定义插件才能实现推理支持。

3 Pellet推理机实现SWRL自定义插件

为了在Pellet中集成自定义插件calculateAge,首先需要从GitHub上(https://github.com/stardogunion/pellet/tree-/maven)下载Pellet的源代码,需要注意的是GutHub上提供了很多Pellet的分支源码下载,一些较新的分支版本例如Pellet-Master不支持在自定义插件中传入个体实例(Individual),插件编写的接口API也有所变化和调整。本文下载的是Pellet2.3.2版本,工程名称为“Pellet-Maven”,此版本能够较好地支持在自定义插件中传入个体变量作为参数,且运行相对稳定。

Pellet源码包是一个Maven项目[28],需要使用集成了Maven功能的IDE进行导入,本研究使用的是Eclipse Mars+Maven 2.4.3作为实验集成环境。Pellet的根项目Pellet-Maven中包含了若干个子项目,包括Core(Pellet核心推理包,包含了Pellet的核心推理算法实现),Jena(Pellet与Jena集成的API工程)等等,为了在推理算法中注入我们的插件实现代码,需要修改Pellet的Core工程。在此工程中新建一个自定义的Java源码包(cas.

ucas.chengaung.swrl.plugins),并新建一个插件实现类(CalculateAgePlugIn),让它实现Pellet的BuiltIn和BindingHelper接口,Eclipse IDE就会自动生成需要实现的接口函数。

BuiltIn和BindingHelper接口实现了Pellet推理机针对SWRL中各个插件组成原语的推理实现回调接口。Pellet推理机处理SWRL推理规则的处理方法如算法1所示。

算法1 Pellet推理机处理SWRL组成原语算法

输入:SWRL规则,包含各插件原语集合$A=\{\cdots I,J,K\cdots\}$

输出:SWRL推理规则是否成功执行

1.获取各原语插件实现类,完成插件是否正确配置的检测

BindingHelper helper=createHelper(BuiltInAtom atom)

2.获得各原语需要的已绑定才能执行的变量集合

Collection N=getPrerequisiteVars(Collection bound)

3.获得各原语插件可以绑定的变量

Collection C=getBindableVars(Collection bound)

4.根据SWRL推理规则中,各个组成原语Atom的N集合和C集合,生成SWRL原语调用顺序序列。

设当前已经绑定的变量集合为R,对于调用序列中的任意2个原语I和J,必须满足:$R\cup C_I\supseteq N_J$如果能够满足,继续执行步骤5

如果无论如何调整调用顺序都不能满足$R\cup C_I\supseteq N_J$,:

返回:不能执行(忽略此SWRL的执行)

5.按照调用序列循环调用各个原语实现插件,针对其中的每个插件实现:

(1)通过回调接口,传入已经绑定的所有变量和变量值:rebind(VariableBinding newBinding)

(2)通过回调接口,获取此插件是否按照传入变量成功处理其功能业务:boolean selectNextBinding()

(3)如果插件功能处理成功,通过回调接口获取新的绑定变量值:setCurrentBinding(VariableBinding currentBinding)

6.若所有的插件都成功执行并绑定变量,返回:执行成功

对于建模时在SWRL推理规则中使用的自定义插件,只需编码此插件实现BuiltIn和BindingHelper的实现类,就可以实现Pellet推理机的推理加载支持了。此实现类中已经包含了算法1中所列出的关键回调函数。如算法1中createHelper接口用于获取一个进行数据绑定的帮助者实例,由于已经实现了BindingHelper接口,因此可以直接返回本类引用this。createHelper回传了一个atom原语,它代表在建模本体知识图谱SWRL规则时,编写在本体文件中的calculate-Age插件原语,通过这个原语可以获得在本体模型中声明的变量参数,比如?birth和?age。在回调函数createHelper中一般完成自定义原语声明变量的合法性检查和初始化工作。

如算法1中getBindableVars函数和getPrerequ-isiteVars函数分别返回了此自定义插件能够绑定的变量和需要预先绑定才能执行的变量。Pellet会依据这两个函数的返回结果,确定一个SWRL推理规则中前驱部分包含的所有原语的安全调用顺序序列。具体的讲,如算法1中第4步所示,Pellet推理机根据getBindableVars确定了一个插件原语能够绑定的变量,根据getPrerequisiteVars确定一个插件原语需要已经绑定变量的条件,从而动态地调整各个原语的调用顺序。在SWRL推理规则被执行的时候,这两个函数会被反复调用,如果无论各原语的调用顺序如何调整都不能使得通过getBindableVars确定的可绑定变量集合满足某个插件的getPrerequisiteVars变量要求,则Pellet会给出推理异常警告,忽略此SWRL规则的执行。在计算年龄自定义插件的例子中getBindableVars返回?age(AgeKey),而getPrerequisiteVars返回?birth(BirthDayKey),表达此插件在执行前需要保证?birth变量被绑定,并且此插件在执行后能够绑定?age作为返回结果。

如算法1中的rebind函数会在一个具体的推理变量集合被装填时调用,比如装填了张三的生日1988-11-27作为?birth(BirthDayKey)的值。通过此回调函数回传的newBinding参数,可以取到对应的数值并进行自定义计算。如算法1中的selectNextBinding在rebind函数之后执行,返回一个布尔型的表示自定义计算rebind是否执行成功的结果。如果成功,则算法1中的setCurrentBinding被调用,用于注入计算的返回结果到绑定变量集合中。

4　集成自定义Pellet推理机到Protégé

Pellet以插件拓展包的形式被Protégé本体知识图谱建模工具加载。为了编译注入了在第3章中实现的PelletCore核心推理源代码的Pellet插件包Pellet-CG,需要首先从GitHub(https://github.com/stardogunion/pellet/tree/master)上下载Pellet的Protégé插件源码包Protege,导入Eclipse并修改其pom文件的工程依赖为2.3.2如图2所示。

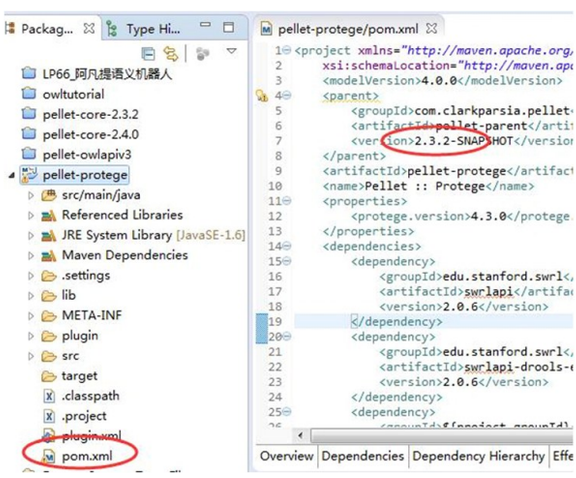


图2　导入Pellet-Protege工程并修改工程依赖

```
package com.clarkparsia.protege.plugin.incremental;
import org.semanticweb.owlapi.reasoner.BufferingMode;
……
public class PelletCGReasonerFactory extends AbstractProtegeOWLReasonerInfo
{
    public OWLReasonerFactory getReasonerFactory()
    {
        //注入自定义插件
        BuiltInRegistry.instance.registerBuiltIn("http://cas.ucas.chenguang/swrl#calculateAge",new CalculateAgePlugIn());
        return com.clarkparsia.pellet.owlapiv3.PelletReasonerFactory.getInstance();
    }
    public BufferingMode getRecommendedBuffering()
    {
        return BufferingMode.BUFFERING;
    }
}
```

图3 新建推理机获取接口并注入自定义插件

```
<?xml version="1.0" ?>
<plugin>
<extension id="pellet.incremental.reasoner.factory" point="org.protege.editor.owl.inference_reasonerfactory">
<name value="Pellet-2.3"/>
<class value="com.clarkparsia.protege.plugin.incremental.PelletReasonerFactory"/>
</extension>
      <!-- 修改Protege插件自述,模仿Pellet2.3的写法注入Pellet-CG -->
     <extension id="pellet.reasoner.factory" point="org.protege.editor.owl.inference_reasonerfactory">
<name value="Pellet-CG"/>
<class value="com.clarkparsia.protege.plugin.pellet.PelletCGReasonerFactory"/>
     </extension>
</plugin>
```

图4 修改Protégé插件自述XML文件

需要注意的是Pellet-Protege工程编译上需要依赖pellet-core、pellet-owlapi3、pellet-modularity等工程依赖，这些在Pellet-Maven的子项目中都提供了工程源代码，请确保在IDE中打开了它们或者使用Maven的intall命令安装了这些基础依赖包，才能顺利地编译Pellet-Protege工程。

在工程中新建PelletCGReasonerFactory推理机工厂类，将第3章中编写好的自定义插件注入到Pellet的推理机获取接口中，如图3所示。

然后修改Protégé的插件自述文件“plugin.xml”如图4所示，仿照Pellet-2.3的配置注入了自定义插件Pellet-CG，配置其在Protégé中显示的名称，对应实现类配置为上一步中的PelletCGReasoner-Factory的工程全路径。

这样就可以通过导出jar包的形式，导出经过推理插件源码注入的Pellet插件jar包pellet-cg，并通过将此jar包拷贝到Protégé安装目录下的plugins文件夹下，完成自定义Pellet-CG推理机的安装。注意在导出jar包的时候必须选择使用工程中的META-INF文件，否则将导致Protégé不能正确的加载Pellet推理机。

最后，重新启动Protégé本体知识图谱建模工具，就可以使用经过注入了自定义SWRL插件实现的Pellet-CG推理机，基于包含自定义插件原语calculateAge的SWRL推理规则进行推理，完成人类知识图谱中人的年龄的知识补全，如图5所示。

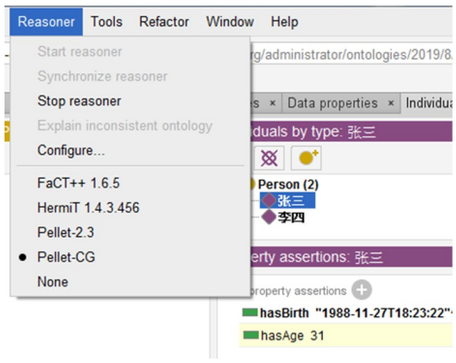


图5 使用Pellet-CG推理机完成年龄推理

5 基于SWRL自定义插件进行知识补全

通过第3章和第4章的讨论，已经明确了在知识图谱SWRL推理补全规则中使用自定义插件，并实现推理支持的方法。由于插件所需要实现的功能是使用者根据实际知识图谱的推理补全需求灵活自定义的，因此这种方法适用于任何应用SWRL推理规则进行知识图谱知识补全的应用场景。

本章将应用自定义插件到一个智慧养老老人健康小屋物联网系统的知识图谱推理补全实践中，以应用基于自定义插件的SWRL推理规则从低级的系统传感数据，推理和补全出系统组成和系统工作状态以及故障状态的高级知识。本章内容仅仅作为一个实际应用案例，研究人员可以根据自己的应用需求场景灵活自定义

SWRL插件,并通过编码注入的方式实现推理机支持,从而实现对应的知识图谱推理补全需求。

在智慧养老老人健康小屋物联网系统中,接入了3个温度传感器用于感知室内温度,接入了1台空调设备进行制冷,接入了1台暖气设备用于制热。所有的设备和传感器都由云端中控服务统一调度。系统需要保障室内温度在有效的时间范围内达到老人生活的最适宜温度26 ℃。此物联网系统的基础本体知识图谱概念模型如图6所示。

如图6所示,智慧养老老人健康小屋物联网系统模型共建模了17种物联网系统概念,包括代表物联网系统本身的物联网系统类(IOTSystem),代表物联网系统所处物理环境的环境类(Environment),代表物联网系统中各个组件资源(包括传感器和云端控制服务)的资源类(Resource),以及代表资源所产生的物联网系统行为的行为类(Behavior)。各个类别又细分为若干子类,例如资源类(Resrouce)从资源的功能角度上看,又可以分为传感器资源(Sensor)、中控服务资源(Service)和控制类实体资源(Controller)。从资源的状态角度上看,可以把资源细分为处于工作中的资源(WorkingResource)、已经发生故障的资源(FaultResource)和当前不能判定其状态的资源(UnknownResource)。

健康小屋物联网系统中的概念类分别衍生出13种不同的对象属性(Object Propertity)和12种数据属性(Data Prpoerty)。对象属性连接到本体模型中的其他个体,数据属性则直接连接到一个具体的值。以资源(Resource)为例,它包含可以连接到一个物联网系统的属性"属于系统(belongSystem)",表示这个资源所属的物联网系统。也包含了连接到行为类的"执行了行为(execBehavior)"和"接收了行为(takeBehavior)",分别表示此物联网系统资源产生了某种行为,和接收了其他物联网系统资源发送的行为。例如,传感器产生了温度感知的行为,空调控制实体资源接收了来自中控服务的控制命令。资源(Resrouce)同时也包含了一些数据属性,例如,代表唯一标识的"标识地址(identyURI)"以及代表此资源被判定为故障资源的原因"故障原因(faultReason)"等等。

健康小屋的基础本体模型中包含了对物联网系统所处的物理环境以及物联网系统资源的抽象概念表述,但基础模型中不包含具体的实例和数据。比如,已经明确了存在传感器的概念,描述传感器应该有哪些数据属性或者对象关联属性,但是这个时候我们还不能确定物联网系统中有哪些实际的传感器实例。

为了在对物联网系统的侵入性最小的前提下完成本体物联网系统的自动实例建模,将基于Java语言编写一个本体建模中间件,本体建模中间件以代理的模式接入到健康小屋物联网系统的传感控制资源与云端中控服务之间,实现感知数据代理和控制命令的转发,具体的实现架构如图7所示。

健康小屋物联网系统自动建模实现架构的工作原理概括如下:自动建模中间件首先加载基础本体模型中的概念,并依据这些模型中的概念和接受到的感知数据与控制命令,使用Jena Ontology API[29]自动创建对应的物联网系统感知与控制行为实例,并提供一个HTTP服务,用于输出当前实时的带有感知行为实例的物联网本体基础事实模型。在实验环境中通过运行中间件,一共从老人健康小屋物联网系统中采集到3个温度感知实体,2个温度控制实体和1个中控服务实体,并采集到这

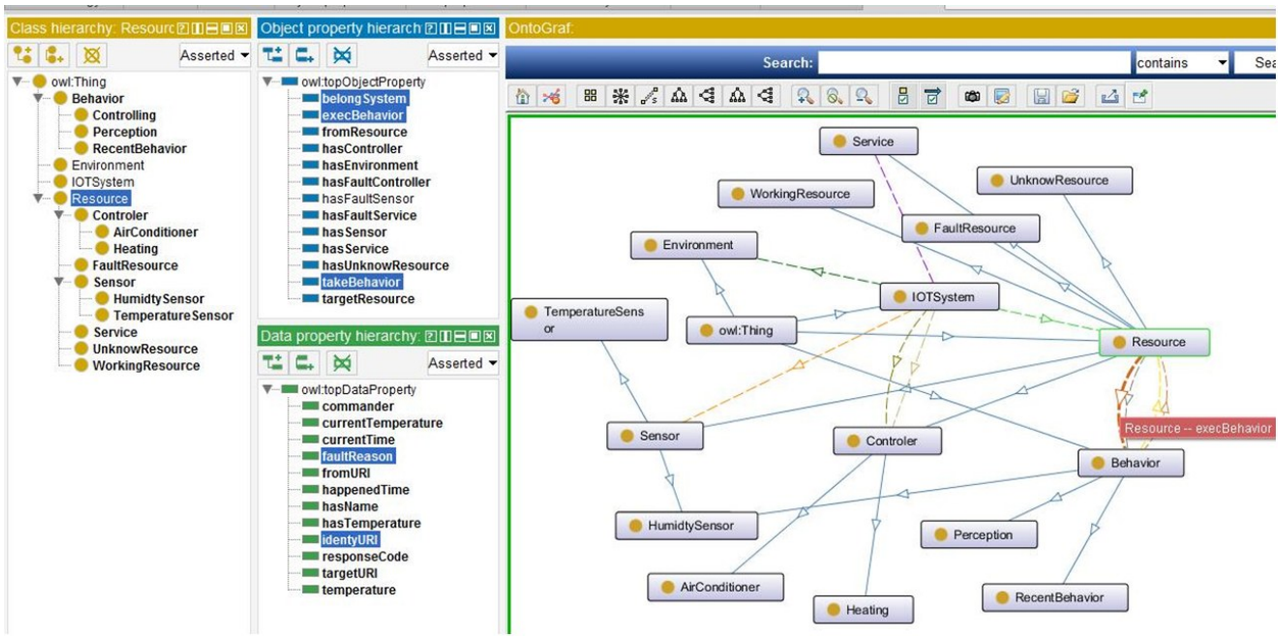


图6　健康小屋物联网系统基础本体模型

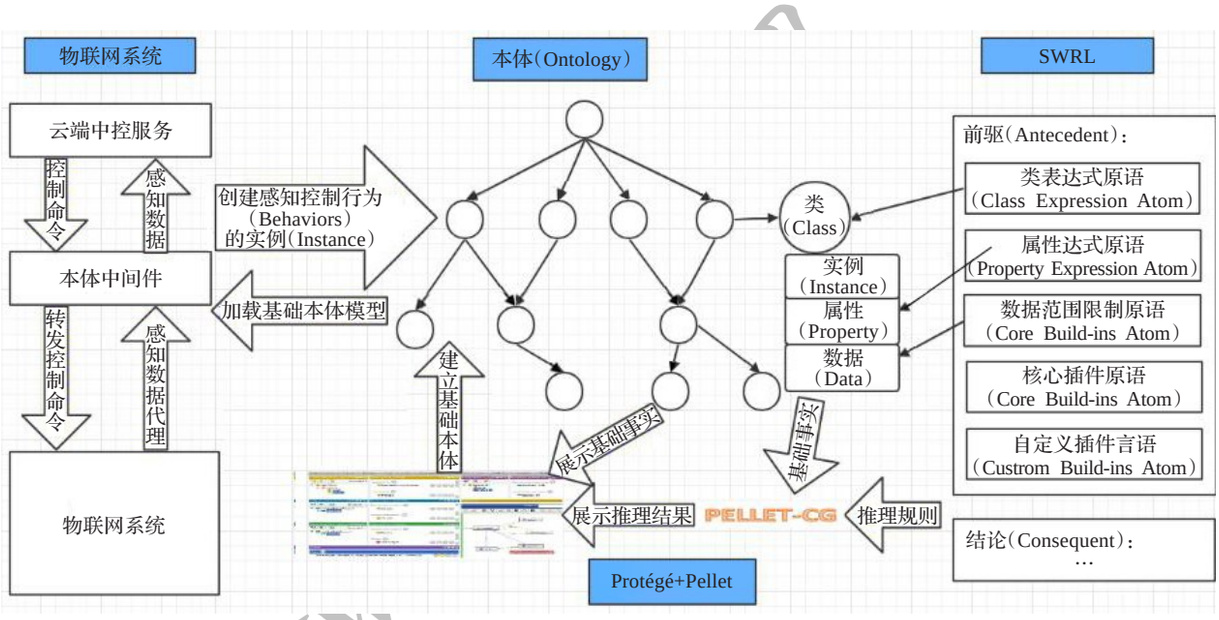


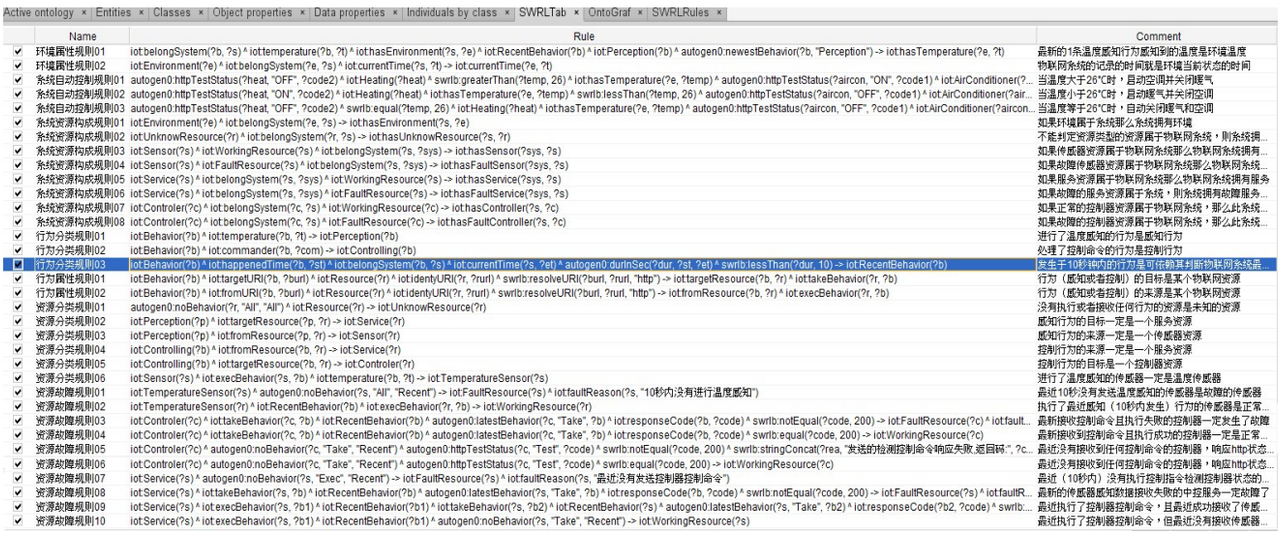
图7　物联网系统实例建模实现框架

些实体之间相互发生和作用的行为(Behavior)实例共计531个。

物联网实时HTTP输出服务包含了物联网系统的实时资源行为信息,但不包含系统构成,系统组成和各资源工作状态判定的高级知识,需要通过SWRL推理规则来对这些高级知识进行知识图谱推理补全。首先通过Protégé工具读取本体中间件HTTP输出服务提供的带有感知数据和控制命令的物联网系统最新的行为信息。然后参照第3章和第4章中提出的方法,使用集成在Protégé中经过源代码修改和重新编译的Pellet-CG(Pellet推理机注入了自定义推理插件的实现)进行模型推理,从而最后补全所有能够被推理的物联网系统组成和各资源故障状态的高级知识,实现物联网系统状态的实时运行状态监控。

健康小屋物联网系统建模了34条知识补全推理规则如图8所示,在这些推理规则中使用了“durInSec”和“httpTestStatus”这两种自定义插件原语。推理规则中包含了用于将物联网系统和行为关联起来的“行为属性规则”;用于完成健康小屋物联网系统资源和行为自动分类的“行为分类规则”与“资源分类规则”;用于进行健康小屋物联网系统资源故障诊断的“资源故障规则”;基于系统资源工作状态的推理结果进行物联网系统自动控制的“系统自动控制规则”;以及基于行为和资源分类与资源故障推理结果进行物联网系统资源构成知识补全的系“统资源构成规则”。限于篇幅,本文不能将这些规则的功能进行一一论述,但会详细说明使用到自定义推理插件实现知识图谱推理补全的推理规则,以突出自定义SWRL推理插件在健康小屋知识图谱推理补全中发挥的作用。

如图8中的“行为分类规则”用于对建模中间件捕



Active ontology × | Entities × | Classes × | Object properties × | Data properties × | Individuals by class × | SWRLTab × | OntoGraf × | SWRLRules ×

	Name	Rule	Comment
✓	环境属性规则01	iot:belongSystem(?b, ?s) ^ iot:temperature(?b, ?t) ^ iot:hasEnvironment(?s, ?e) ^ iot:RecentBehavior(?b) ^ iot:Perception(?b) ^ autogen0:newestBehavior(?b, "Perception") -> iot:hasTemperature(?e, ?t)	最新的1条温度感知行为感知到的温度是环境温度
✓	环境属性规则02	iot:Environment(?e) ^ iot:belongSystem(?e, ?s) ^ iot:currentTime(?s, ?t) -> iot:currentTime(?e, ?t)	物联网系统的记录的时间就是环境当前状态的时间
✓	系统自动控制规则01	autogen0:httpTestStatus(?heat, "OFF", ?code2) ^ iot:Heating(?heat) ^ swrlb:greaterThan(?temp, 26) ^ iot:hasTemperature(?e, ?temp) ^ autogen0:httpTestStatus(?aircon, "ON", ?code1) ^ iot:AirConditioner(?...	当温度大于26℃时,自动空调并关闭暖气
✓	系统自动控制规则02	autogen0:httpTestStatus(?heat, "ON", ?code2) ^ iot:Heating(?heat) ^ iot:hasTemperature(?e, ?temp) ^ swrlb:lessThan(?temp, 26) ^ autogen0:httpTestStatus(?aircon, "OFF", ?code1) ^ iot:AirConditioner(?air...	当温度小于26℃时,自动暖气并关闭空调
✓	系统自动控制规则03	autogen0:httpTestStatus(?heat, "OFF", ?code2) ^ swrlb:equal(?temp, 26) ^ iot:Heating(?heat) ^ iot:hasTemperature(?e, ?temp) ^ autogen0:httpTestStatus(?aircon, "OFF", ?code1) ^ iot:AirConditioner(?aircon...	当温度等于26℃时,自动关闭暖气和空调
✓	系统资源构成规则01	iot:Environment(?e) ^ iot:belongSystem(?e, ?s) -> iot:hasEnvironment(?s, ?e)	如果环境属于系统那么系统拥有环境
✓	系统资源构成规则02	iot:UnknowResource(?r) ^ iot:belongSystem(?r, ?s) -> iot:hasUnknowResource(?s, ?r)	不能判定资源类型的资源属于物联网系统,则系统拥...
✓	系统资源构成规则03	iot:Sensor(?s) ^ iot:WorkingResource(?s) ^ iot:belongSystem(?s, ?sys) -> iot:hasSensor(?sys, ?s)	如果传感器资源属于物联网系统那么物联网系统拥有...
✓	系统资源构成规则04	iot:Sensor(?s) ^ iot:FaultResource(?s) ^ iot:belongSystem(?s, ?sys) -> iot:hasFaultSensor(?sys, ?s)	如果故障传感器资源属于物联网系统那么物联网系统...
✓	系统资源构成规则05	iot:Service(?s) ^ iot:belongSystem(?s, ?sys) ^ iot:WorkingResource(?s) -> iot:hasService(?sys, ?s)	如果服务资源属于物联网系统那么物联网系统拥有服务
✓	系统资源构成规则06	iot:Service(?s) ^ iot:belongSystem(?s, ?sys) ^ iot:FaultResource(?s) -> iot:hasFaultService(?sys, ?s)	如果故障的服务资源属于系统,则系统拥有故障服务...
✓	系统资源构成规则07	iot:Controller(?c) ^ iot:belongSystem(?c, ?s) ^ iot:WorkingResource(?c) -> iot:hasController(?s, ?c)	如果正常的控制器资源属于物联网系统,那么此系统...
✓	系统资源构成规则08	iot:Controller(?c) ^ iot:belongSystem(?c, ?s) ^ iot:FaultResource(?c) -> iot:hasFaultController(?s, ?c)	如果故障的控制器资源属于物联网系统,那么此系统...
✓	行为分类规则01	iot:Behavior(?b) ^ iot:temperature(?b, ?t) -> iot:Perception(?b)	进行了温度感知的行为是感知行为
✓	行为分类规则02	iot:Behavior(?b) ^ iot:commander(?b, ?com) -> iot:Controlling(?b)	处理了控制命令的行为是控制行为
✓	行为分类规则03	iot:Behavior(?b) ^ iot:happenedTime(?b, ?st) ^ iot:belongSystem(?b, ?s) ^ iot:currentTime(?s, ?et) ^ autogen0:durInSec(?dur, ?st, ?et) ^ swrlb:lessThan(?dur, 10) -> iot:RecentBehavior(?b)	发生于10秒钟内的行为是可依赖其判断物联网系统后...
✓	行为属性规则01	iot:Behavior(?b) ^ iot:targetURI(?b, ?burl) ^ iot:Resource(?r) ^ iot:identyURI(?r, ?rurl) ^ swrlb:resolveURI(?burl, ?rurl, "http") -> iot:targetResource(?b, ?r) ^ iot:takeBehavior(?r, ?b)	行为(感知或者控制)的目标是某个物联网资源
✓	行为属性规则02	iot:Behavior(?b) ^ iot:fromURI(?b, ?burl) ^ iot:Resource(?r) ^ iot:identyURI(?r, ?rurl) ^ swrlb:resolveURI(?burl, ?rurl, "http") -> iot:fromResource(?b, ?r) ^ iot:execBehavior(?r, ?b)	行为(感知或者控制)的来源是某个物联网资源
✓	资源分类规则01	autogen0:noBehavior(?r, "All", "All") ^ iot:Resource(?r) -> iot:UnknowResource(?r)	没有执行或者接收任何行为的资源是未知的资源
✓	资源分类规则02	iot:Perception(?p) ^ iot:targetResource(?p, ?r) -> iot:Service(?r)	感知行为的目标一定是一个服务资源
✓	资源分类规则03	iot:Perception(?p) ^ iot:fromResource(?p, ?r) -> iot:Sensor(?r)	感知行为的来源一定是一个传感器资源
✓	资源分类规则04	iot:Controlling(?b) ^ iot:fromResource(?b, ?r) -> iot:Service(?r)	控制行为的来源一定是一个服务资源
✓	资源分类规则05	iot:Controlling(?b) ^ iot:targetResource(?b, ?r) -> iot:Controler(?r)	控制行为的目标是一个控制器资源
✓	资源分类规则06	iot:Sensor(?s) ^ iot:execBehavior(?s, ?b) ^ iot:temperature(?b, ?t) -> iot:TemperatureSensor(?s)	进行了温度感知的传感器一定是温度传感器
✓	资源故障规则01	iot:TemperatureSensor(?s) ^ autogen0:noBehavior(?s, "All", "Recent") -> iot:FaultResource(?s) ^ iot:faultReason(?s, "10秒内没有进行温度感知")	最近10秒没有发送温度感知的传感器是故障的传感器
✓	资源故障规则02	iot:TemperatureSensor(?r) ^ iot:RecentBehavior(?b) ^ iot:execBehavior(?r, ?b) -> iot:WorkingResource(?r)	执行了最近感知(10秒内发生)行为的传感器是正常...
✓	资源故障规则03	iot:Controler(?c) ^ iot:takeBehavior(?c, ?b) ^ iot:RecentBehavior(?b) ^ autogen0:latestBehavior(?c, "Take", ?b) ^ iot:responseCode(?b, ?code) ^ swrlb:notEqual(?code, 200) -> iot:FaultResource(?c) ^ iot:fault...	最新接收控制命令且执行失败的控制器一定发生了故障
✓	资源故障规则04	iot:Controler(?c) ^ iot:takeBehavior(?c, ?b) ^ iot:RecentBehavior(?b) ^ autogen0:latestBehavior(?c, "Take", ?b) ^ iot:responseCode(?b, ?code) ^ swrlb:equal(?code, 200) -> iot:WorkingResource(?c)	最新接收到控制命令且执行成功的控制器一定是正常...
✓	资源故障规则05	iot:Controler(?c) ^ autogen0:noBehavior(?c, "Take", "Recent") ^ autogen0:httpTestStatus(?c, "Test", ?code) ^ swrlb:notEqual(?code, 200) ^ swrlb:stringConcat(?rea, "发送的检测控制命令响应失败,返回码:", ?c...	最近没有接收到任何控制命令的控制器,响应http状态...
✓	资源故障规则06	iot:Controler(?c) ^ autogen0:noBehavior(?c, "Take", "Recent") ^ autogen0:httpTestStatus(?c, "Test", ?code) ^ swrlb:equal(?code, 200) -> iot:WorkingResource(?c)	最近没有接收到任何控制命令的控制器,响应http状态...
✓	资源故障规则07	iot:Service(?s) ^ autogen0:noBehavior(?s, "Exec", "Recent") -> iot:FaultResource(?s) ^ iot:faultReason(?s, "最近没有发送控制器控制命令")	最近(10秒内)没有执行控制指令检测控制器状态的...
✓	资源故障规则08	iot:Service(?s) ^ iot:takeBehavior(?s, ?b) ^ iot:RecentBehavior(?b) ^ autogen0:latestBehavior(?s, "Take", ?b) ^ iot:responseCode(?b, ?code) ^ swrlb:notEqual(?code, 200) -> iot:FaultResource(?s) ^ iot:faultR...	最新的传感器感知数据接收失败的中控服务一定故障了
✓	资源故障规则09	iot:Service(?s) ^ iot:execBehavior(?s, ?b1) ^ iot:RecentBehavior(?b1) ^ iot:takeBehavior(?s, ?b2) ^ iot:RecentBehavior(?s) ^ autogen0:latestBehavior(?s, "Take", ?b2) ^ iot:responseCode(?b2, ?code) ^ swrlb:...	最近执行了控制器控制命令,且最近成功接收了传感...
✓	资源故障规则10	iot:Service(?s) ^ iot:execBehavior(?s, ?b1) ^ iot:RecentBehavior(?b1) ^ autogen0:noBehavior(?s, "Take", "Recent") -> iot:WorkingResource(?s)	最近执行了控制器控制命令,但最近没有接收传感器...

图8　健康小屋物联网系统SWRL推理规则

获到的物联网系统行为进行分类。其中最为重要的就是判定物联网系统的实时最新行为,如“行为分类规则3”所示,这条规则的核心思想是把发生在当前时间10 s内的规则归类为最近发生的行为(RecentBehavior)。这就涉及到取出当前的系统时间,并与行为发生的时间做出比较,如果单纯的基于SWRL核心推理插件(Core Built-ins)是无法实现这个功能的,因此使用了一个自定义插件原语“durInSec”插件原语,这个插件原语的功能是根据系统当前的时间和物联网行为的发生时间(happenedTime)计算从发生时间到当前时间已经经过了多少秒钟,从而进一步通过比较此结果是否小于10而判断此物联网行为是否为近期行为。

在健康小屋物联网系统中,如果控制资源(空调或者暖气)接收了最近行为(RecentBehavior),则根据最近行为中最新的一条行为的执行情况,即可判定其工作状态:如果控制资源在10 s内接收到控制指令并成功处理,可以充分地判定控制资源处于工作状态;相反如果接收到控制命令却处理失败,则可以判定控制资源发生故障。但如果控制资源最近10 s没有处理过任何行为(控制指令),那么不能简单地判定其处于故障状态,因为在云端中控服务故障的情况下,也会导致空调或者暖气控制实体不处理任何的控制行为。

为了进一步判断控制资源的工作状态,我们编写了一个能够发送HTTP请求的自定义原语插件“httpTestStatus ”,如图8中“资源故障规则05和06”所示,传入控制器实体(?c),和需要发送的命令(Test),即可发送一条http请求,获得对应的返回结果(?code),从而进一步判定控制实体的工作状态。这个自定义插件实现的代码并不困难,因为基于资源分类的推理结果,对于传入的控制资源实例(?c),已经可以明确地取出其服务地址和端口信息。但是如果不能进行插件代码的自定义编写而仅仅依靠SWRL官方定义的插件原语是根本无法实现这个推理补全需求的。

基于这些包含自定义插件的SWRL推理规则,可以从整体上完成一个物联网系统的工作状态的知识补全,如图9所示,展示了经过知识推理补全后老人健康小屋物联网系统的当前实时的系统组成和工作状态。

图9　健康小屋物联网系统资源构成知识补全结果

如图9所示,可以看出此系统目前接入了一个传感器,一个中控服务和一个控制器,并且控制器已经出现了故障。点击对应的资源链接,如图9右侧蓝色下划线部分所示,可以直接打开对应的物联网控制器资源,展示其具体的资源分类和故障原因如图10所示。可以看出一个控制器资源由于最近没有正确处理所接收到的控制指令而被推理为故障状态。

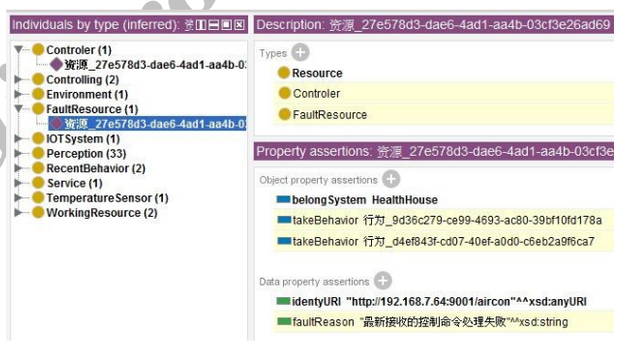


图10　健康小屋物联网系统资源故障规则知识推理结果

应用包含自定义推理插件的SWRL推理规则,成功地完成了老人健康小屋中系统资源组成和系统故障诊断的高级知识推理补全,使得老人健康小屋知识图谱的知识含量进一步丰富,并且基于推理知识补全的结果,为基于老人健康小屋知识图谱实现系统实时自动控制打下了很好的基础。

6　结束语

知识图谱是人工智能应用的基石,基于推理规则进行知识推理是知识图谱知识补全的重要方法。本文提出了一个在SWRL知识推理规则中使用自定义插件原语,并实现Pellet推理机对这些自定义原语推理支持的实现方法。并对如何基于Pellet中插件的实现接口进行自定义推理插件的代码实现方式,以及在Protégé知识建模工具中集成经过源代码修改的推理机的插件集成方法进行了比较完整和详细的论述。为使用知识推理进行知识图谱知识补全提供了更加丰富和自由的推理规则建模表述和推理支持实现能力。在老人健康小屋知识图谱的建模推理实践中,充分地证明了基于自定义推理插件的SWRL推理规则能够有效地从低级传感数据中挖掘和补全高级的系统工作知识,实现系统状态的自动感知和故障的推理判断。

未来的研究工作可以考虑将一些更加复杂的知识推理算法,例如机器学习和深度学习算法的实现为一个集成的自定义知识推理插件,以进一步拓展使用SWRL自定义规则进行知识补全的表述和推理能力。未来的研究工作也需要研究如何应对在大规模使用自定义SWRL推理规则的应用场景中进行推理规则的冲突消解,以及推理规则执行的并行优化的问题。

未来的研究工作还需要研究如何将SWRL自定义推理规则,应用到更多具体的知识图谱建模场景之下,从而丰富基于SWRL自定义推理规则进行知识图谱建模和知识图谱推理补全的实践应用层面,体现基于知识图谱和推理技术解决实际应用问题的研究价值。

参考文献:

[1] 侯梦薇,卫荣,陆亮,等.知识图谱研究综述及其在医疗领域的应用[J].计算机研究与发展,2018,55(12):2587-2599.

[2] 蒋秉川,万刚,许剑,等.多源异构数据的大规模地理知识图谱构建[J].测绘学报,2018,47(8):1051-1061.

[3] 王勇超,罗胜文,杨英宝,等.知识图谱可视化综述[J].计算机辅助设计与图形学学报,2019(10):1666-1676.

[4] 徐增林,盛泳潘,贺丽荣,等.知识图谱技术综述[J].电子科技大学学报,2016,45(4):589-606.

[5] FRANK M,ERIC M.RDF primer[EB/OL].(2004)[2019-09-15].https://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[6] SMITH M K,CHRIS W,MCGUINNESS D L.OWL Web ontology language guide[EB/OL].(2004)[2019-09-15].https://www.w3.org/TR/2004/REC-owl-guide-20040210/.

[7] BOLLACKER K D,COLIN E,PRAVEEN P,et al.Freebase:A collaboratively created graph database for structuring human knowledge[C]//Proceedings of Sigmod Conference,2008.

[8] SÖREN A.DBpedia:A nucleus for a Web of open data[C]//Semantic Web,International Semantic Web Conference,Asian Semantic Web Conference,Iswc+Aswc,Busan,Korea,November,2007.

[9] 官赛萍,靳小龙,贾岩涛,等.面向知识图谱的知识推理研究进展[J].软件学报,2018,29(10):2966-2994.

[10] IAN H,PATEL-SCHNEIDER P F,HAROLD B,et al.SWRL:A semantic Web rule language combining OWL and RuleML[EB/OL].(2004)[2019-09-15].https://www.w3.org/Submission/SWRL/Overview.html.

[11] HUANG T,LI W,YANG C.Comparison of ontology reasoners:Racer,pellet,Fact++[C]//Proceedings of Agu Fall Meeting,2008.

[12] BIRTE G,IAN H,BORIS M,et al.HermiT:An OWL 2 reasoner[J].Journal of Automated Reasoning,2014,53(10):245-269.

[13] 周亮,黄志球,倪川.基于SWRL规则的本体推理研究[J].计算机技术与发展,2015,25(10):67-70.

[14] 李洮洮,田国会,张梦洋,等.基于本体的物品属性类人认知及推理[J].浙江大学学报(工学版),2018,52(7):1231-1238.

[15] 王飞,张应中,罗晓芳.基于SQWRL的本体知识库语义查询[J].计算机技术与发展,2017,27(2):15-19.

[16] 翟社平,马传宾,李威,等.基于SWRL规则推理的知识发现研究[J].信息技术,2016(2):76-79.

[17] 丁亚飞,李冠宇,张慧.语义物联网中基于语义空间的语义协同方法研究[J].计算机应用与软件,2016,33(2):1-6.

[18] 冯建周,宋沙沙,孔令富.物联网语义关联和决策方法的研究[J].自动化学报,2016,42(11):1691-1701.

[19] 李永超,罗钧旻.语义Web中的本体推理研究[J].计算机技术与发展,2007,17(1):101-103.

[20] BERNERS-LEE T.Uniform Resource Identifiers(URI):Generic Syntax[EB/OL].(1998)[2019-09-15].https://www.ietf.org/rfc/rfc2396.txt.

[21] LIAM.Extensible Markup Language(XML)[EB/OL].(2016)[2019-09-15].https://www.w3.org/XML/.

[22] NICK B.Rule markup language initiative[EB/OL].(2019)[2019-09-15].http://wiki.ruleml.org/index.php/RuleML_Home.

[23] MARTIN K.OWL 2 and SWRL Tutorial[EB/OL].(2012)[2019-09-15].http://dior.ics.muni.cz/~makub/owl/.

[24] NOY N F,SINTEK M,DECKER S,et al.Creating semantic Web contents with Protege-2000[J].IEEE Intelligent Systems,2005,16:60-71.

[25] MATTHEW H.A practical guide to building OWL ontologiesusing protege 4 and CO-ODE tools[EB/OL].(2011-03-24)[2019-09-15].http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf.

[26] ERIKSSON H.Using JessTab to integrate Protege and Jess[J].Intelligent Systems IEEE,2003,18:43-50.

[27] ALVES M B,DAMÁSIO C V,CORREIA N.SPARQL commands in jena rules[C]//Proceedings of International Conference on Knowledge Engineering & the Semantic Web,2015.

[28] 邓志强,邓林强.Maven在Java项目开发中的应用[J].电子元器件与信息技术,2019,3(5):1-4.

[29] BRIAN M.Jena:A semantic Web toolkit[J].IEEE Internet Computing,2002,6:55-59.