第 35 卷 第 1 期 计 算 机 工 程 2009 年 1 月
Vol.35 No.1 Computer Engineering January 2009
·软件技术与数据库· 文章编号：1000—3428(2009)01—0055—03 文献标识码：A 中图分类号：TP311

领域本体的一致性检查

许 勇，王智学，李宗勇
(解放军理工大学指挥自动化学院，南京 210007)

摘 要：针对领域本体中概念集的一致性检查问题，给出领域本体的形式化定义，阐述领域本体中的公理集，分析概念集的一致性检查方法，将 OWL 的本体表达方式转化为基于 UML 的表达方式，对 UML 进行适当的扩展，提供一种基于图形化的本体一致性检查机制。该方法基于图形化的形式，简洁直观、容易操作，克服了原有检查方法形式化程度高及复杂难操作的缺点。

关键词：领域本体；UML 语言；一致性检查

Consistency Check for Domain Ontology

XU Yong, WANG Zhi-xue, LI Zong-yong
(Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007)

【Abstract】 This paper focuses on the concepts' consistency in the domain ontology. It presents the formal definition of the domain ontology, describes the set of axioms in the domain ontology, analyzes nowaday consistency check method about concepts, and converts OWL ontology into UML description, and develops the UML description. It provides a consistency check method based on graph, which is laconic and easy to handle, and conquers the disadvantage of high formalization and complexity of the old check mechanism.

【Key words】 domain ontology; UML; consistency check

1 概述

本体是语义网的基础[1]，已应用到知识工程、信息综合、需求工程等众多领域。在这些领域中，本体的使用有效地增强了系统的推理能力、查询能力及系统之间的协作与综合能力。领域本体是用于描述指定领域的一种专门本体，由领域实体概念相互关系及其定义、属性、公理和约束等组成。从开发者的角度看，领域本体定义了开发者之间需要共享的领域信息的公共词汇；从系统的角度看，领域本体定义了能被机器理解的领域概念及其关系。在需求工程中，利用领域本体在知识表示方面形式化、规范化的优势，可以很好地解决领域内知识共享的问题，有效消除需求获取过程中的“通信鸿沟”，并在一定程度上实现知识的复用和推理。

由于用户需求的改变以及设计过程中缺陷的纠正等原因，现实世界中建立的领域本体往往不是静止不变而是不断演化发展的。在领域本体的演化过程中，一个很重要的问题是确保领域本体的一致性。目前，OWL是一种被业界广泛关注的本体语言，已经成为W3C组织推荐的标准本体建模语言。并且OWL本体的一致性检查也得到各种推理机的支持，如RACER[2], Fact[3], Pellet[4]。然而，OWL是一种严格的逻辑描述语言，其复杂的符号仅为少数专业人员理解和掌握，OWL的理解难度制约了基于OWL本体应用的发展，使得针对它的一些推理机和推理方法操作起来比较复杂，不太直观。如果能够利用图形化的界面表示OWL中的元素，并在此基础上开发基于图形化的一致性检查方法，将会很大程度地提高本体一致性检查的可操作性。

UML 是软件工程中的标准建模语言，它不仅受到学术界和软件开发领域的广泛关注，而且在实际应用中得到了大量工具和领域专家的支持。因此，要降低 OWL 在理解上的复杂性、构建高质量的图形化模型、提高本体一致性检查的可操作性，UML 无疑是一种理想的选择。

2 领域本体

2.1 领域本体的定义

定义 一个领域本体由 6 个部分组成：

Domain::=(<DomName>,<DomConS>,
<AttDomConS>,<DomConAssS>,
<AttDomConAssS>,<DomAxiomS>)
DomConAssS::={association,SubClassOf,
UnionOfRelation,EquivalentClass,
DisjointWith}

其中，DomName 是领域本体的名称，表示所描述的领域；DomConS 是领域内概念的集合；AttDomConS 是概念的属性的集合；DomConAssS 是领域内概念间关系的集合；AttDomConAssS 是关系的属性的集合；DomAxiomS 是领域内公理的集合。本文定义了 5 种关系，association 表示一般意义上的关联关系，其名称可以自定义；SubClassOf 表示“子类”关系，形成了概念之间的逻辑层次结构；UnionOf Relation 表示“整体－部分”关系；EquivalentClass 表示概念“等价”关系；DisjointWith 表示概念之间“不交”关系。

2.2 领域本体中的公理集

领域本体中的公理是对领域内概念以及领域内概念间关系的约束的集合，它包含的是领域知识中的一些永真断言，如概念之间的继承、等价和不相交及属性之间的继承、等价。下面是对领域本体中主要公理的描述，其中，$C_i(i=1, 2, \cdots, n)$ 表示的是一个概念。

基金项目：国家部委预研基金资助项目
作者简介：许 勇(1983－)，男，硕士研究生，主研方向：需求工程，软件工程；王智学，教授、博士生导师；李宗勇，博士研究生
收稿日期：2008-06-14 **E-mail：**xylgdx1983@163.com

SubClassOf是一种概念间的二元关系，用于指出概念间的类属关系，它形成了概念之间的逻辑层次结构。SubClassOf(C_1,C_2)表示概念C_1继承于概念C_2，其中，概念C_1是子概念；概念C_2是父概念。

公理1 SubClassOf(C_1,C_2)∧ SubClassOf(C_2,C_3) →SubClassOf(C_1,C_3) (子类关系的传递性)

公理2 SubClassOf(C_1,C_2)∧…∧SubClassOf(C_i, C_{i+1})… ∧ SubClassOf (C_n,C_1)=False

上述公理说明，在领域本体中“子类”关系不允许存在循环，即不允许出现父概念继承子概念的情况出现。

UnionOfRelation是概念间的整体和部分关系，它表示一个概念是另一个概念的一部分。UnionOfRelation (C_1,C_2)表示C_1是C_2的一部分。

公理3 UnionOfRelation(C_1,C_2) ∧UnionOf Relation(C_2, C_3) → UnionOfRelation (C_1,C_3) (整体-部分关系的传递性)

公理4 UnionOfRelation (C_1,C_2) ∧…∧UnionOf Relation (C_i,C_{i+1}) …∧UnionOfRelation (C_n,C_1)=False

上述公理说明了在领域本体中“整体-部分”关系不允许存在循环，即不允许出现部分包含整体的情况出现。

EquivalentClass表示概念之间的等价关系，它表明 2 个概念在逻辑上是等价的。Equivalent Class(C_1,C_2)表示C_1和C_2是等价的，例如，概念¬∀enrolledIn.¬ Course 和概念∃enrolledIn.Course 都表示参加上课的学生，将概念¬∀enrolledIn.¬Course经过等价转换，变为标准否定范式的形式后就得到了概念∃enrolledIn.Course，所以，这 2 个概念是等价的。

公理5 EquivalentClass(C_1,C_2)∧EquivalentClass(C_2, C_3)→EquivalentClass(C_1,C_3) (等价关系的传递性)

公理 6 SubClassOf(C_1,C_2)∧EquivalentClass(C_2,C_3)→SubClassOf(C_1,C_3)

公理7 SubClassOf(C_1,C_3)∧EquivalentClass(C_1,C_2)→SubClassOf(C_2,C_3)

公理 6 和公理 7 表明了等价概念具有相同的特性，它们有共同的子概念和父概念。

DisjointWith表示概念之间的不交关系，它表明一个概念和另一个概念不存在关联。DisjointWith(C_1,C_2)表示概念C_1和概念C_2是不交的。

公理8 DisjointWith(C_1,C_2)∧(SubClassOf(C_3,C_1)∨EquivalentClass(C_3,C_1))∧(SubClassOf(C_4,C_2)∨Equivalent Class(C_4, C_2))→DisjointWith(C_3,C_4)

上述公理表明了如果 2 个概念在逻辑上不相交，那么它们的子概念或等价概念之间在逻辑上也是不相交的。

2.3 领域本体的一致性

领域本体是一致的，也就是要满足领域本体中的公理的集合。本文将领域本体的一致性分为 3 种：语法一致性，语义一致性和用户自定义的一致性。

(1)语法一致性：如果对一个本体的描述符合相应的本体描述语言(如 OWL)的语法规则，则认为该本体是语法一致的，因此，本体的语法一致性可以通过检查本体的构造过程是否违背本体语言中定义的语法规则进行判断。语法规则的例子有：类的补集必须也是类，类之间不能循环继承等，如上述的前 7 个公理。

(2)语义一致性：如果一个本体符合相应的本体描述语言的逻辑基础，则认为该本体是逻辑一致的。如公理 8。

(3)用户自定义的一致性：这类一致性实际上涉及到具体的领域规则，比如：如果教师甲编写了教材乙，课程丙使用了教材乙，那么必须要求教师甲教授课程丙。

3 描述逻辑作为本体语言的推理

OWL 是与描述逻辑等价的本体表示语言，它们之间可以相互转化。目前，OWL 的推理工具都是将 OWL 本体转化为描述逻辑知识库进行推理的。描述逻辑的知识库包含表示术语公理的 TBox 和表示断言事实的 ABox。TBox 表示描述逻辑概念间的蕴含和等同关系等背景知识；ABox 陈述领域个体和概念以及个体对和关系间的隶属关系。

对描述逻辑知识库的一致性检查分为 2 类：(1)只关注本体中的概念集，而不关注个体集，也就是表示术语公理的 TBox(terminological or concept knowledge)的一致性检验；(2)在关注概念集合的基础上，关注个体集合，也就是表示断言事实的 ABox(assertional or instance knowledge)的一致性检验。本文主要关注的是本体中概念集合的语义一致性检验，也就是领域本体的语义一致性检查。

目前已有一些检查本体一致性的方法，它们大都是基于 Tableau 算法的。

(1)试建一个概念 C 的模型。

(2)模型用一棵树T表示：T的节点是描述逻辑中$\triangle^I$的元素；节点用C的子概念的集标示；边表示$\triangle^I$的元素间的关系；边用角色名标示。

(3)根节点被标示为$\{C\}$。

(4)转换规则对应语言结构。

(5)规则应用到节点标签直到转换规则不适用→树表示了一个有效的模型及一个冲突(矛盾)被发现$\{A,\neg A\}$。

转换规则如下：

(1)∩规则

$S\rightarrow\{x:C_1, x:C_2\}\cup S$，若$x:C_1\cap C_2$在$S$中，且$x:C_1$和$x:C_2$不在$S$中同时出现。

(2)∪规则

$S\rightarrow\{x:D\}\cup S$，若$x:C_1\cup C_2$在$S$中，$x:C_1$和$x:C_2$都不在$S$中，且$D=C_1$或者$D=C_2$。

(3)∃规则

$S\rightarrow\exists\{x\ P_1y,\ P_2y,\cdots,sP_ky,y:C\}\cup S$，若$x:\exists R.C$在$S$中，$R=P_1\cap\cdots\cap P_k$，没有$z$使得$xRz$在$S$中成立，且$z:C$ 在S中，y为一个新变量。

(4)∀规则

$S\rightarrow\forall\{y:C\}\cup S$，若$x:\exists R.C$在$S$中，$xRy$在$S$中成立，且$y:C$不在$S$中。

例如，TBox 集为：$\{C\subseteq B,\ B=\exists R.D,\ D\subseteq E,\ E\subseteq A\cap F,\ F\subseteq\neg A\}$，其一致性检查过程如下：

(1)$a:\{C\}$；

(2)$a:\{C,B\}$；

(3)$a:\{C,B,\exists R.D\}$；

(4)利用∃规则得 $R(a,b),b:\{D\}$；

(5)$b:\{D,E\}$；

(6)$b:\{D,E,A\cap F\}$；

(7)利用∩规则得 $b:\{D,E,A\cap F,A,F\}$；

(8)$b:\{D,E,A\cap F,A,F,\neg A\}$，该集合中既有 A，又有$\neg A$，所以发现冲突，TBox 集合是不一致的。

从上述过程来看，形式化程度较高，增加了用户的理解

难度。下面把本体的 OWL 描述转换为 UML 的描述形式，然后在此基础上提出一种基于图形化的本体概念集一致性检查方法。

4 OWL 向 UML 的转化

目前，已有不少关于 OWL 向 UML 转化的研究，如关于领域本体概念向 UML 概念映射的规则。文献[5]阐述了利用 UML 的 Profile 扩展技术建立 OWL 语法元素与 UML 建模元素之间的映射关系等。由于本文主要考虑本体中概念集合的语义一致性检验，因此只阐述了与之相关的 OWL 中的一些元素与相应的 UML 建模元素之间的转换。同时，为了使一致性检查的更加直观和方便，在原有的 UML 中引入了一些新的图形元素。

在表 1 中，建立了一些 OWL 本体元素与相应 UML 元素的映射表，其中，OWL 中的 DisjointWith 与 EquivalentClass 都对应 UML 中的 Association 关系，但是为了有所区别及更加直观，对原有的 UML 中 Association 关系的图形表示进行了修改，引入了新的图形表示。

表 1 OWL 与 UML 元素对照表

OWL 本体元素	UML 对应的元素	图形化表示
Class	Class	
SubClassof	Generalization	
UnionOfRelation	Aggregation	
DisjointWith	Association	
EquivalentClass	Association	

5 领域本体一致性检查

领域本体出现不一致的情况往往是由不交关系 DisjointWith 引起的，因此，在判断领域本体一致性时，主要是围绕不交关系进行考虑的。它的大致流程如图 1 所示。

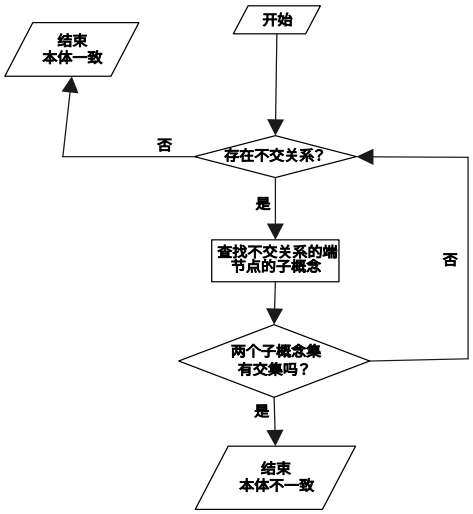

图 1 领域本体一致性检查流程

具体步骤如下：

(1)判断是否存在不交关系，若是，进入(3)，否则进入(2)。

(2)退出，领域本体一致。

(3)依次采用深度优先搜索算法 DFS 查找不交关系的 2 个端节点的子概念，并将这些子概念归入到各自的子概念集中。在查找过程中，把等价类也归入子概念集中。

(4)将不交关系的 2 个端点并入各自的子概念集中。

(5)判断 2 个子概念集是否有交集，若有，进入(6)，否则，进入(1)。

(6)退出，领域本体不一致。

6 一致性检查的设计与实现

笔者开发了自己的需求获取工具，在该工具中实现了上述一致性检查机制。用类 C++形式描述的一致性检查算法如下，其中，Start 和 End 分别是不交关系的源端和目的端；strStart 和 strEnd 分别用来保存它们的子概念集合；relation List 表示关联列表。

```
Bool ConsistencyCheck()
{   if(!HasDisjoint)
        return true;
   for(int i=0;i<relationList.GetSize();i++)
{if(relationList.GetAt(i)==Disjoint)
{
SearchChildSet(relationList.GetAt(i).Start,strStart);
//深度优先搜索
SearchChildSet(relationList.GetAt(i).End, strEnd);
//深度优先搜索
            if(IsIntersectant(strStart,strEnd))
return false;}}}
```

在采用深度优先搜索算法搜索子集的过程中，需要重复地从关系列表中寻找等价和子类关系，为了提高效率，在搜索过程中要对搜索过的关系进行标记，以免重复判断。

下面用一个实例进行说明。该实例采用 OWL 的抽象语法进行描述：

本体 O:

Human = Person,

Man ⊆ Person,

Woman ⊆ Person

Woman ⊆ ¬Man

Father ⊆ Man

Father ⊆ Woman

上述实例描述的 Human 与 Person 是等价的，都表示人的概念；Man 和 Woman 是 Person 的子概念，分别表示男人和女人，并且在逻辑上是不相交的；Father 是 Man 和 Woman 的公共子概念。显然，上述本体是不一致的。采用的工具描述见图 2。然后运用领域本体的一致性检验算法进行检验，获得的结果见图 3。

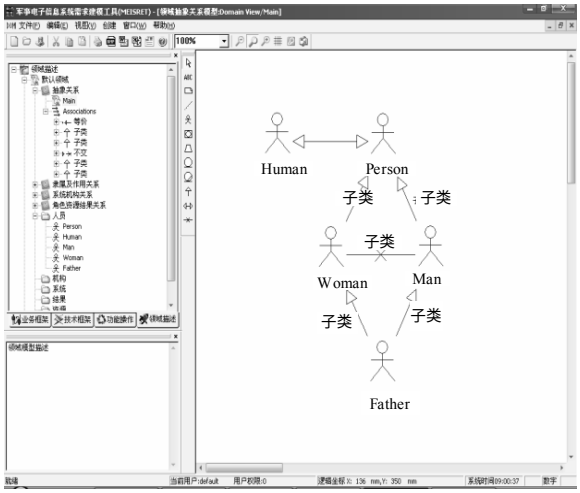

图 2 描述工具

（下转第 64 页）