构建领域本体的方法

李　景[1],苏晓鹭[2],钱平[2]

([1]中国科学院文献情报中心,北京　100080,[2]中国农业科学院科技文献信息中心,北京　100081)

摘　要:本文介绍了开发领域本体(Domain ontology)的基本方法和实施步骤,进一步强调了定义类的等级体系、类的实例以及属性过程中的实施要点,成为我们建立农业知识组织体系实践的指南。

关键词:领域本体;ontology 开发方法;分类等级体系

[中图分类号] S126　　[文献标识码]A　　[文章编号]1007-6581(2003)07-0007-04

前 言

本体(ontology)是关于一些主题的清晰规范的说明。它是一个规范的、已经得到公认的描述,它包含词表(或名称表/术语表)。词表中的术语与某一领域相关,词表中的逻辑声明用来描述术语的含义和术语间关系(它们是怎样和其它术语相关联的)。本体提供了用来表达和交流某些主题知识的词表和把握着词表中这些术语间的联系的关系集。领域本体(Domain ontology),是专业性的本体,提供了某个专业学科领域中概念的词表以及概念间的关系,或在该领域里占主导地位的理论。

构建本体,可以解决在用户间或软件代理间达成对于信息组织结构的共同理解和认识,可以复用专业领域知识,使专业领域内的假设变得更加明确,将专业领域的知识从运筹学、知识管理的环境中剥离出来,并且可以分析专业领域的知识。

一个本体包括一套关于某一领域概念的规范而清晰的描述称为类(classes)或概念(concepts),描述了有关概念的各种特征的属性(properties)和属性插件(slots,有时也被称为 roles 或 properties),还包括属性插件的限制条件(restrictions)和分面(facets,有时也被称作 role restrictions),以及一系列与某个类相关的实例(instances),这些实例组成了一个知识库(knowledge base)。类是本体的核心,它描述了某一领域的概念。属性插件描述了类的属性和实例。

开发一个本体的过程包含:定义本体中的类、在某一分类学等级体系中安排本体中的类、定义属性插件并描述其允许的赋值,为实例的属性插件赋值。通过定义这些类的单个实例,并添加特定的属性插件赋值信息和限制条件,就可以建立起一个知识库。

1　构建领域本体的知识工程方法

构建领域本体的方法依赖于所采用的应用软件以及可以预见的扩展功能。本体的开发和完善是一个反反复复的叠加过程。本体中的概念应该贴近于要研究的专业领域(物理的或者是逻辑的)中的客观实体(objects)和关系法则(relationships)。对应于客观实体的概念,其词性应该是名词;对应于关系法则的概念,其词性应该是动词。

1.1　确定本体的专业领域和范畴

可以通过确定专业领域和范畴作为开发领域本体的起点,首先,要明确构建的本体将覆盖的专业领域、应用本体的目的、本体应该能回答哪些类型的问题以及它的系统维护者与应用对象。这些问题可以随时调整,但是由于他们限制模型的范畴,所以需要相对稳定。

确定本体范围的方法之一是列出基于本体的知识库能够回答的问题清单(Competency questions)。通过这些问题可以得到对这个本体是否包含回答这些类型问题的足够的信息、问题的答案是否需要特定层次的详细信息或特定专业领域的表达法、这些问题是否保留成为提纲形式,而不需要细化等的解释。

1.2　复用现有的本体

如果系统需要和其它的应用平台进行互操作,而这个应用平台又与特定的本体或受控词表连锁在一起,那么复用现有的本体是行之有效的方法。许多现成的本体,例如:Ontolingua 的本体文库、DAML 的本体文库、UNSPSC、和 DMOZ 等,可以导入到本体开发系统中,本体的格式转换也并不困难。

1.3　列出本体中的重要术语

列出一份所有术语的清单,这上面的术语是需要声明或解释的。首先,需要一份最全的术语清单,而不要担心概念间会有属性及表达上的重复。

1.4　定义类和类的等级体系

建立一个等级体系有几种可行的方法。自顶向下法:由某一领域中最大的概念开始,而后再通过添加子类将这些概念细化。自底向上法:由最底层、最细小的类的定义开始,然后将这些细化的类组织在更加综合的概念之下。综合法:首先定义很多非常显而易见的概念,然后分别将它们进行恰当地归纳和细化。从一些顶层概念着手,将它们与一些中级概念关联起来。采取什么方法

主要依赖于开发人员对专业领域的个人观点。由于“中级概念”在领域的概念中更具代表性,所以综合法对本体开发者而言最便捷。如果想要收集到更多更广泛的实例,那么自底向上的方法更加适合。

无论选择哪种方法,都要从定义类开始。选择描述独立存在的对象的术语,作为本体中的类,而且嵌入分类等级体系中。也可以视类为一阶谓词逻辑结构(即只含有一个自变量)的提问式。如果类 A 是类 B 的母类,那么 B 中的每一个实例也是 A 中的实例。换言之,类 B 代表类 A 中的一个“Kind of”的概念。

1.5　定义类的属性插件(slots)

除了定义类外,还必须描绘概念间的内在结构。例如,确定哪条术语是描述哪个类的属性。这些属性会成为依附于类的属性插件。通常“内在的”属性(″intrinsic″ properties),“外在的”属性(″extrinsic″ properties)都能成为本体中的属性插件。如果对象是结构化的,那么它的一部分,可以是具体的或抽象的元素。同时也要描述类中的个体成员与其它类之间的关系。除了最初确定的一些属性之外,还需要添加一些其他的属性插件。任意类的所有子类都继承了该类的属性插件。一个属性插件应该被附加在拥有该属性的最大的类上。

1.6　定义属性插件的分面(facets)

属性插件可以有不同的分面(facets)来描述赋值类型(value type)、允许的赋值(allowed values)以及赋值的基数(cardinality),属性插件可以接受的赋值的其它特征。

属性插件的基数(cardinality)定义了一个属性插件可以有几个赋值。有些系统仅仅能够区分单一基数(只允许有一个赋值)和多元基数(允许有任何数量的赋值)。通过最大与最小基数的规范说明来描述属性插件赋值的个数,使描述更加精确。最小基数 N 是指一个属性插件至少有 N 个赋值。将最大基数定义为 0 表示某一子类的属性插件不能有任何赋值。

赋值类型(slot-value type)的分面描述了某一属性插件的赋值类型。赋值类型主要有字符型(String slots)是象 name 名称这样的最简单的赋值类型;数值型(Number slots)包含浮点数(Float)和整数(Integer);布尔型(Boolean slots)只有单纯的 yes-no 标记;枚举型(Enumerated slots)是某个属性插件的赋值清单;实例型(Instance-type slots)允许定义个体间的关系。

当某一属性插件被添加在一个特定的类时,允许限制属性插件的范围。一个属性插件所隶属的类集合,称为这个属性插件的域(domain)。确定一个属性插件的域或范围的规则通常是相同的:在为一个属性插件确定它的域时,找出最大的类或是分别能成为这个属性插件的域或范围的类。不要定义太过通用的域和范围。如果定义某个属性插件的范围或域的类列表清单包含了某个类以及它的子类,那么将子类去掉,因为它的存在并不会增加新的信息。如果定义某个属性插件域或范围的类清单包含类 A 中的所有子类,但是未包含类 A 本身,那么这一范围应该只包含类 A,而不是类 A 的那些子类。如果该清单几乎包含了类 A 中的所有子类,仅有少数几个子类未包含在内,那么应该考虑是否需要重新调整类 A 的范围。

1.7　创建实例

定义某个类的一个实例需要确定一个类,创建类的一个实例和添加属性插件的赋值。

2　定义类和分类的等级体系的原则

完善等级体系和定义概念属性(属性插件)是密不可分、互相交织的,二者必须同时进行。这两个步骤在本体的设计进程中最为重要。等级体系的确定依赖于本体的用途、应用平台、个性化特点,有时还要考虑和其它系统的兼容性。在定义大量新的类和逐渐形成等级体系过程中要随时检查是否符合下述原则。

2.1　分类等级体系的合理性

分类等级体系体现出“is-a”的关系:如果类 A 中的每个实例也是类 B 中的实例,那么类 A 是类 B 的子类。一种建模过程中易犯的通病是:在等级体系中,包含某一相同概念的单数和复数版本,而且把前者作为后者的子类,出现等级体系中的“kind-of”关系。等级体系关系具有传递性(Transitivity),母类-子类的关系是具有传递性的。如果 B 是 A 的子类且 C 是 B 的子类,那么 C 也是 A 的子类。有时,需要区别直接子类(direct subclass)和间接子类(indirect subclass)。直接子类是与类关系最近的子类:二者之间不加杂其它的类类。分类等级体系应考虑到专业领域的不断发展,体现兼容性和可维护性。

区分类和类名是极为重要的。类代表某一专业领域中的概念,而不是用于表示这些概念的词汇(words)。对于不同的术语体系,类名是可以改变的,但术语(term)本身却代表了存在于现实世界的客观实体。用于相同概念的同义词(Synonyms)不能表示不同的类。同义词是一个概念或一条术语的不同名称。它们表示的是而且只能是同一个类。许多系统允许将同义词、不同语种的译文,或同一个类的不同名称表示的列表关联在一起。要注意避免类的循环,如果类 B 是类 A 的子类,同时 B 还是 A 的母类,这个等级体系中就存在一种循环。这种循环相当于宣布类 A 就是类 B:A 的所有实例也是 B 的实例,B 的所有实例也是 A 的实例。

2.2　分类等级体系中的同属关系

分类等级体系中的同属关系（Siblings）是指同一个类中的若干直接子类之间的关系。等级体系中具有同属关系的类，应该是属于同一水平上的类。一个类需要有多少个直接子类，并没有硬性规定。结构良好的本体的直接子类的数目一般在 2~12 个左右。如果有一个类只有一个直接子类，建模过程中就有可能出现错误，或者就是本体不完整。如果某个类有多于 12 个子类，那么应考虑需要对它们做进一步的归纳。

2.3　多重的继承关系（Multiple Inheritance）

大多数知识表达系统都允许分类等级体系具有继承性，即一个类可以是若干个类的子类。子类将继承上位类的全部属性插件和分面。

2.4　新类的引入

在构建本体的整个过程中，决定在何时引入一个新的类，或者在何时利用不同的属性值来描述本质特征是最为困难的。某个类的子类通常有一些独特的而且是它们的母类不具有的属性，或者拥有不同于母类的制约因素限制条件，或者和母类比较起来，介入了更加复杂的关系之中。换言之，当有些内容只是某个类具有，而它的母类并不具备时，才可以在等级体系中引入一个新的类。实际上，应该为每一个子类添加新的属性插件，或者定义新的属性插件赋值，或者删除已继承的属性插件的某些分面。即使没有任何新的属性，有时也可以引入新的类。有些本体包含大规模的某一领域通用术语的参考等级体系。术语学等级体系中的类是不需要引入新属性的。引入不含任何新属性的新类也可以用于构建新的概念，利用新的概念，领域专家们就可以简单地找出类和类之间的本质区别。另外，没有必要为了每一个额外的限制条件创建子类。

2.5　新类与属性的赋值

在建模过程中，通常需要决定是否要找出一个本质区别作为属性赋值。构建一个类，还是只简单创建一个类，然后为它的属性插件给予不同的赋值，取决与所定义的本体的范围，及该概念在这个领域里的重要性。如果一个概念的不同属性插件赋值对于别的类的不同属性插件来说变成了限制条件，那么应该针对这一特征构建一个新的类。另外，还需要在属性插件的赋值中描述这种区别。在开发细节完备的本体时，这种特征区别非常重要。如果在某一专业领域里存在一个本质特征，而且认为对于这个特征而言，拥有不同赋值的对象，分属于不同的类，那么应该针对这个特征构建新的类。每个类都会有一个特定的实例从属于它，而这个类是不应该经常改变的。通常情况下，数量、色彩以及位置只可以作为属性插件的赋值，不能作为构建新类的理由，

2.6　实例与类目

要确定一个特定的概念是本体中的一个类还是依赖于本体潜在应用平台的单个实例并不容易。此外，还要决定类和实例的起始及表达的最低粒度水平。这种粒度水平取决于本体的潜在应用平台。换言之，就是决定知识库中需要表达的最为精确的术语。最小的单个实例是知识库中所表达的最精确的概念。如果概念被组织成为自然的等级体系（hierarchy），那么应该将这些概念表达为类。只有在等级体系中才可以设置类，知识表达系统中并没有子实例（sub-instance）这样的概念存在。因此，如果术语中有一种自然的等级体系，应该将这些术语定义为类，即使它们本身不包含任何实例。

2.7　限制范围

在考虑本体定义的完整性时，应该注意到一个本体不可能包含某一专业领域的所有信息。在应用平台上，没有必要演绎（或归纳）不需要的任何东西。同样地，本体也不应该包含等级体系的类中所有可能的属性和本质区别。只需要在本体中表达最为显而易见的属性就可以了。系统中所有术语间的关系也没有必要全部添加到本体中来。

2.8　互不相关的子类

如果，若干个类之间没有任何共同的实例，那么它们是互不相关的。不是互不相关的类有很多共用的实例。明确两个类是否是互不相关的会使系统更好地验证本体的性能和逻辑性。否则系统就会出现错误信息。

3　属性定义

一个属性插件的赋值可以由其它属性插件的赋值来决定。例如，如果某家厂商生产了某种产品，也就是说这种产品是由这家厂商生产的。厂商和产品被称为逆反关系（inverse relations）。在从知识获取的角度来看，具备正反两方面的信息而且可以方便地获取。知识获取系统可以为逆反关系自动添加赋值，以确保知识库的连贯性。

许多基于框架的系统（frame-based system）都允许为属性插件定义缺省值（default values）。如果类中的大多数实例都有一个相同的属性插件赋值，那么就把这个赋值定义为这个属性插件的缺省值。一旦某个类中，又有新的包含这个属性插件的实例加入，那么系统会自动将缺省值赋予这个属性插件。可以将缺省值改变为分面允许的其它赋值。缺省值的存在是为了方便起见：系统不会以任何方式为建模添加任何限制。缺省值是可以被更改的，这点与属性插件赋值截然不同。属性插件赋值一旦被确定，就不可以改变。

4　命名

在本体中为概念定义命名规则并严格地遵循它们，不仅会使本体易于理解，而且避免建模错误。规则的选

择可能并没有什么特别的理由。不过命名规则一旦定义了,就必须执行。在建立命名规则中应注意系统的命名空间(name space)对于类、属性插件和实例来说是否相同,系统对大小写的敏感性,单复数、前后缀和分隔符的使用等问题。避免对概念名使用缩写,不要对概念名添加诸如"class"、"property"、"slots"这样的字符串。

5　本体开发工具

由斯坦福大学研制开发的 PROTéGé 2000(Protege 2000)是基于 Java 的开发工具,可以免费下载,提供了较好的本体开发环境。Duineveld 等(Duineveld et al. 2000)还描述和比较了大量其它的本体开发环境。PROTéGé 2000 虽然没有中文版本,但是却支持中文的输入法。所以利用 PROTéGé 2000 可以构建中文本体。Gómez-Pérez (Gómez-Pérez 1998)和 Uschold (Uschold and Gruninger 1996)还介绍过其它的本体开发方法。Ontolingua 指南(Farquhar 1997)探讨了知识建模过程中的一些规范化工作。目前,研究的重点不仅在于本体的开发,还有本体的解析。随着越来越多的本体被开发和复用,能进行本体解析的工具也越来越多。Chimaera (McGuinness et al. 1994, 2000)为解析本体提供了诊断工具。Chimaera 可以检查本体的逻辑正确性并诊断本体设计中的常见错误。Chimaera 在本体开发的全过程中都可以使用。

结　语

在人工智能领域中,本体论已经受到广泛的关注,被应用于知识组织与管理、信息资源规划、智能检索系统设计等方面。本体的设计是一个创造性的过程,对任何专业领域来说,均不存在唯一的本体。本体的潜在应用平台、设计者对专业领域的理解和观点将会对本体的设计方案产生影响。通过设计应用系统,可以验证和评估本体的性能。本文提出的构建领域本体的方法成为我们建立农业知识组织体系实践的指南。本文的研究得到国家十五科技攻关计划"农业信息智能检索、发布与传播技术研究与开发"专题(2001BA513B01-03)的支持。

参考文献:

[1]Chimaera (2000). Chimaera Ontology Environment. http://www.ksl.stanford.edu/software/chimaera

[2] Duineveld, A.J., Stoter, R., Weiden, M.R., Kenepa, B. and Benjamins, V.R. (2000). WonderTools? A comparative study of ontological engineering tools. International Journal of Human-Computer Studies 52(6): 1111-1133.

[3]Gómez-Pérez, A. (1998). Knowledge sharing and reuse. Handbook of Applied Expert Systems. Liebowitz, editor, CRC Press.

[4]McGuinness, D.L., Fikes, R., Rice, J. and Wilder, S. (2000). An Environment for Merging and Testing Large Ontologies. Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000). A. G. Cohn, F. Giunchiglia and B. Selman, editors. San Francisco, CA, Morgan Kaufmann Publishers.

[5]Protege (2000). The Protege Project. http://protege.stanford.edu

[6]Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, Methods and Applications. Knowledge Engineering Review 11(2).

The methodology of developing domain ontology

LI Jing[1], SU Xiao-lu[2], QIAN Ping[2]

([1]Library of Chinese Academy of Sciences, Beijing, 100080, [2]Sciendific Documentation & Information Center, Chinese Academy of Agricultural Sciences, Beijing 100081, China)

Abstract: The methodology and process of developing domain ontology were introduced in this article. The definition of class hierarchy and instances & properties of classes were also emphasized.

Keywords: Domain ontology; The methodology of developing domain ontology; Class hierarchy