From: Dean Allemang, James Hendler. Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL=实用语义网：RDFS 与 OWL 高效建模. 人民邮电出版社，2009

曾新红译　　2010.1

译者的话：本文内容摘译自 Dean Allemang 和 James Hendler 合著的“Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL”一书。这两位资深的语义网培训专家使用比较通俗的语言，试图将语义网工具从逻辑学家和技术人员手中，移交到各个领域的能够真正建立应用程序的睿智的实践者手中。本文摘译的内容集中在语义 Web 建模语言的基本知识上，以及一些与 SKOS、NCI 有关的评论上，这些内容可以让我们快速了解语义 Web 的基本原则、建模的意义、建模语言的种类和基本结构及它们的作用，以及该领域常用术语的含义，并站在语义 Web 的大背景下来看待 SKOS 的作用以及对 SKOS 进行的扩展。

1　语义建模

语义 Web，就像在它之前的文档（document）Web，是基于一些信息共享的基本观念（radical notions）的。这些思想——AAA（Anyone can say Anything about Any topic）口号，开放世界假设以及非唯一命名——提供给一个环境，在其中信息共享可以繁荣，知识协同的网络工作成为可能。但是这种形式的信息聚集产生了一种充满混乱、不一致和冲突的无秩序前景。Web 的基础设施怎样才能支持它从这种无秩序状态向以信息共享、合作和协同为特色的状态发展呢？

这个问题的答案在于建模。建模是一个为了团体共用而组织信息的过程。建模以三种方式支持团体共用：它提供一个人类交流的框架，它提供一种解释结论的方法，以及提供一个用于管理可变观点的结构。在语义 Web 的背景中，建模是一个正在进行的过程。在任何时间点，一些知识将是良好构造和可理解的，这些结构可以用语义 Web 的建模语言来表示。同时，其他知识将仍处于无序、不协调的阶段，每个人都在不同地表达自己。典型地，当不同的人在太阳下提供它们自己关于任何主题的意见时，Web 将同时包含组织好和未经组织的关于同一主题的知识。建模行为就是一种从混乱信息中提取公共知识的行为。

语义 Web 提供了许多建模语言，它们在表达程度上有区别；也就是说，它

们构成了不同的工具，允许不同的人表达不同种类的信息。语义 Web 标准是按每一个语言层次建立在前一层次之上来组织的，所以这些语言本身是分层的（layered）。以下是语义 Web 的语言，表达性从低到高。

RDF——Resource Decription Framework，资源描述框架。这是语义 Web 其余部分立足的基本框架。RDF 提供了一种机制，允许任何人对任何事作基本的声明，并将这些声明分层放进一个模型中。RDF 自 2003 年起已是 W3C 的推荐标准（recommendation）。

RDFS——RDF Schema language，RDF 模式语言。RDF 是一种具有描述共性和变性的基本概念的表达力的语言，这种概念常见于对象语言和其他类系统——即：类（class），子类（subclass）和属性（property）。RDFS 自 2003 年起已是 W3C 的推荐标准（recommendation）。

RDFS-Plus——RDF-Plus 是 OWL 的一个子集，它比 RDFS 更有表达力，但没有 OWL 的复杂性。没有 RDFS-Plus 的标准在制定，但是越来越多的人意识到 RDFS 和 OWL 之间有某些东西可能是工业相关的（即可能在工业上实现）。我们选择了 OWL 的一个子集来递增地展示 OWL 的能力。RDFS-Plus 包含足够的表达力来描述某些属性如何被使用以及它们如何彼此相关。RDFS-Plus 是具有表达力的，足够用来显示某些 RDFS 之上结构（construct）的效用，但它没有让许多建模初学者感到沮丧的复杂性。首选符号的唯一性问题是 RDFS-Plus 表达力的一个例子。

OWL——OWL 把逻辑表达力带给了语义 Web。它允许建模者表达类、实体、属性之间的详细限制。OWL 在 2003 年被 W3C 接受为推荐标准。

2 RDF——语义 Web 的基础

首先和最重要的，RDF 是一个用于数据建模的系统。它放弃了简洁性而获得了灵活性。任意两个数据之间的每一种关系都被明确表示，从而允许一种非常简单的合并数据的模型。关系（以一种熟悉的形式“主体/谓词/客体”表示）要么存在要么不存在。于是合并数据简化为一件简单的事情：在一个地方一起考虑来自所有来源的所有这样的声明。

在这样一个系统中残留的唯一挑战就是*标识*（identity）的挑战。我们如何能

够让每一个实体拥有一个全球符号作为身份标识？幸运的是，这个问题并不只针对 RDF 数据模型。Web 本身的基础就有同样的问题并且有一个标准解决方案：URI。RDF 借用了这个解决方案。

因为 RDF 是一种 Web 语言，一个基本的考虑就是 Web 上来自多个来源的信息的分布（distribution）。在 Web 上，支持 AAA 口号：Anyone can say Anything about Any topic。RDF 支持这个口号，允许任何数据来源引用在任何命名域的资源。甚至一个三元组都可以引用多个命名域的资源。

作为一个数据模型，RDF 对于合并来自多个来源的信息一定会发生什么提供了明确的规范说明。它没有提供算法或技术来实现那些步骤。

基本概念

•RDF（Resource Description Framework，资源描述框架）——这分布数据在 Web 上。

• Triple（三元组）——RDF 的基本数据结构。三元组由主体（subject）、谓词（predicate）、客体（object）构成。

• Graph（图）——RDF 数据的一种节点与链接（nodes-and-links）结构视图。

• Merging（合并）——将两个图视为似乎它们是一个图的这样一个过程。

•URI（Uniform Resource Indicator，统一资源标识符）——URL（Uniform Resource Locator，统一资源定位符）的一个泛化（generalization），它是 Web 上的全局名称。

• namespace（命名域）——属于某一管理机构（authority）的一组名称。命名域允许不同的使用者（agent）以不同的方式使用同一个词。

• qname——URI 的缩写版，它由一个命名域标识符和一个名称组成，中间以冒号间隔。

• rdf:type——实例（instance）及其类型之间的关系。

• rdf:Property——RDF 中任意属性的类型。

• Blank nodes（空节点）——没有 URI 的 RDF 空节点，不能被全球引用。它们被用来表示匿名实体。

3 RDFS

RDFS 是 RDF 的模式（schema）语言；它描述了对象类型结构（Class），彼此之间的相关类型（subclass），描述对象的属性（Property）以及它们之间的关系（subProperty）。RDFS 的 Class 系统包括一个简单而精致的继承观念（基于集合包含（set inclusion））；一个类是另一个类的子类意味着这个类的实例也是另一个类的实例。

RDFS 语言通过用 RDF 本身来表示而受益于 RDF 的分布式特性。所有的模式信息（类，子类，子属性，定义域（domain），值域（range）等）都用 RDF 三元组表示。特别地，这使得模式信息（以及数据）都服从 AAA 口号：任何人可以就任何话题说任何事——甚至是就模式本身说任何事。

RDFS 的语义通过推理机制来表达；也就是说，RDFS 中任何结构的语义都由可从中推断出来的推理给定。例如，正是这种简单而强大的指定语义的机制考虑到了（allow for）简短而精致的子类和子属性定义。

RDFS 也包括结构 rdfs :domain 和 rdfs:range，描述属性和类之间的关系。这些结构的含义由很简单的规则给出，但这些规则有微妙而深远的影响。规则可能简单，但声明是强大的。

即使只有小型的结构集合和简单的规则，RDFS 也考虑到了广泛的集成问题的解决。无论何时你想要在一个结构化数据集中做一个全局的查找-替换，考虑改用 rdfs :subPropertyOf 或 rdfs :subClassOf。这样说可能有些琐碎：人们应该只合并那些没有重要区别的多来源的实体。使用 RDFS 的推理机制，我们可以只确定我们在合并事物时发生了什么并判断结果是令人满意的还是危险的。虽然 RDFS 没有提供并集和交集这样的逻辑原语（primitive），通过使用 subClassOf 和 subPropertyOf 的特定模式，还是常常有可能获得满意的推理结果。RDFS 提供了一个信息可以在其中流动的框架；我们可以把 subClassOf 和 subPropertyOf 看作是语义建模的 IF/THEN 程序。甚至当我们改用 OWL 建模时，这种效用仍然可以持续。事实上，以这种方式使用 subClassOf 提供了一块 OWL 建模的基石。

当用于很小心的结合时，RDFS 的结构特别有效，用于定义不同结构的信息如何能够以一种统一的方式一起使用。

基本概念：

• rdfs:subClassOf——类之间的关系，一个类的成员包含在另一个类的成员之中。

• rdfs:subPropertyOf——属性之间的关系，一个属性关联的对包含在另一个属性关联的对之中。

• rdfs:domain 和 rdfs:range——一个属性的描述，确定由那个属性关联的个体的类成员（membership）。

•RDFS 中的逻辑操作(Union,Intersection 等)——RDFS 结构可以用来模拟集合和属性的某些逻辑结合。

4 RDFS-Plus

RDFS-Plus 的结构是 OWL 结构的一个子集。这个子集为语义 Web 上的建模提供了可观的灵活性。在下一节中，我们会看到一些例子，看看 RDFS-Plus 是如何用在一些大规模的语义 Web 项目上的。

基本概念：

• rdfs:subClassOf——子类的成员也是超类的成员。

• rdfs:subPropertyOf——由子属性描述的关系超属性也拥有。

• rdfs:domain——一个三元组的主体被分类为这个谓词的 domain。

• rdfs:range——一个三元组的客体被分类为这个谓词的 range。

Annotation Properties（注释属性）

• rdfs:label——没有推理性的语义，可打印的名称

• rdfs:comment——没有推理性的语义，为此模型的读者提供的信息。

OWL Features: Equality

• equivalentClass——每个类的成员也是另一个类的成员。

• equivalentProperty——每个属性拥有的关系另一个属性也拥有。

• sameAs——一个实例的所有声明另一个也拥有。

OWL Features（功能部件）：Property Characteristics（属性特征）

• inverseOf——互换主体和客体

• TransitiveProperty——一串关系压缩成一个关系

- SymmetricProperty——一个属性是它自己的逆属性
- FunctionalProperty——只有一个值被允许（为客体）
- InverseFunctionalProperty——只有一个值被允许（为主体）
- ObjectProperty——可以有一个资源为客体的属性
- DatatypeProperty——可以有数据值为客体的属性

5 RDFS-Plus 的“在野”使用：SKOS 和 FOAF

我们已经看到了大量的例子，以一种动态和灵活的方式使用 RDFS-Plus 为合并多来源信息建模。本章我们描述两种 RDFS-Plus 结构（constructs）的扩展应用。这两种 RDFS-Plus 的应用在它们各自的领域都已经吸引了可观的用户群。它们都对 RDFS-Plus 的结构进行了基本的利用，虽然常常是以非常不同的方式。这些是真实的建模应用，由起初对 RDFS 或 OWL 并无技术承诺的团队构建（虽然两者都被认为是 RDF 应用）。

SKOS 和 FOAF 证明了一个相当简单的建模结构（construct）集合如何可以用来创建可扩展的分布式的信息网络。它们都利用了 RDF 的分布式特性来允许一个分布在 Web 上的信息网络的扩展。它们都依赖于 RDFS-Plus 的推理结构（structure）来给它们的信息结构加入完备性。它们都使用了 owl:InverseFunctionalProperty 来确定关键元素的识别。

虽然它们在这些方面类似，但 FOAF 和 SKOS 在它们支持其期望用户团体进行扩展方面有很不同的组织。FOAF 对信息扩展采用一种进化的方法。许多概念都有大量的术语（如“name”的若干变体）。FOAF 可以扩展为新的需要的功能部件（feature）。例如，foaf:weblog 在博客流行之前并不重要，但最近，其重要程度几乎超越了更经典的 foaf:hompage。

相比之下，SKOS 则采用了一种更有条理的方法进行扩展。SKOS 以三个部分出现：SKOS Core，这里已经描述了；SKOS Mapping，它包括用于映射不同来源词表的词表；SKOS Extensions，用于 SKOS 的特殊垂直（vertical）应用。SKOS Core 计划作为叙词表的一种中间语言，它由一个小型的委员会设计，试图将其他叙词表的主要成分（fundamentals）合并成一个单一的语义 Web 模型。其他两个文档引入了 SKOS Core，并在其上建立了进一步的语义。

当我们考虑它们将如何被扩展时，这两种方法之间的不同会变得更加明显。FOAF 很严格地执行了 AAA 口号，因此该表示的真正受欢迎的部分将在其使用中被确定为一个很大的范围。而另一方面，SKOS 有一个相当稳定的核心，它由一个见识广博的委员会设计，他们已经对现存的词表系统进行了详细的共性/变性分析。SKOS 的体系结构已确定和公布，作为其发展的路线图。

RDF 的技术结构支持这两种模式。FOAF 的自由扩展风格和 SKOS 的有序分层都通过使用同样的 RDF 图覆盖机制来实现。不同之处在于这种覆盖是如何组织和管理的。每一种方法都不是本来就超过另一种方法；它们每一个都实现了对自己的项目来说重要的一些目标。

SKOS 和 FOAF 研究计划在许多方面类似于标准研究计划。它们每一个都由一个委员会来维护，这个委员会制订和公布有关的政策决定。但是它们在一个重要的方面不同于标准。它们都不是计划作为一项完整的工作，将给一个设计词表控制系统或社会网络系统的人提供说明性的建议。它们的任务是为共享这类信息提供一种 Web 上的交换机制。这是语义 Web 上一个模型的力量；它不是说明如何去表示事物，而是提供一个从一种表示到另一种表示的转换方法。每一个研究计划都提供了一个例子：一个模型如何扮演这种中间角色。

已经存在若干种叙词表标准，人们可能会很奇怪为什么这个群体觉得有必要创建另一个标准。SKOS 与叙词表之间的关键区分点在于，它的根基在语义 Web 上。和标准不同，SKOS 从一开始就设计成允许建模者创建模块化的知识组织，这些知识组织能够被重用和在 web 上被引用。SKOS 不是设计来代替任何叙词表标准的，事实上，通过将语义 Web 的分布式特性带给叙词表和受控词表，SKOS 增强了标准。朝着这个方向，可能将任意的叙词表以一种相当直接的方式映射到 SKOS 也是 SKOS 的一个设计目标。

使用 SKOS 表示叙词表的一个好处是实现叙词表之间的映射：用 SKOS 可以很容易输出叙词表，不同的机构可以独立地维护这些 SKOS 输出；有了这些 SKOS 表示（使用 URI），可以很直接地表示出两个词表之间的映射。如联合国的 AGROVOC 与各成员国图书馆的农业叙词表之间的映射。

基本概念

• foaf—— 一个社会网络信息系统的命名域；“friend of a friend”的缩写。

• SKOS——一个信息管理表示系统的命名域，代表“Simple Knowledge Organization System”。

6 OWL

OWL 的一个重要的功能性是定义限制类的能力。未命名的类通过在类的特定属性值上施加限制来定义。使用这种机制，OWL 可以用来为这样的情形建模：一个特定类的成员必须有某些属性。在 RDFS 中，domain 和 range 限制允许我们进行有关一个类的所有成员的推理（例如 playFor 将一个棒球队员和一个棒球队联系起来）。在 OWL 中，人们可以使用限制声明来区分这样的情况：某个事物是应用于一个类的所有成员还是一些成员，甚至还可以为一个类的所有成员的一个特定属性坚持一个特殊的值。

当限制用来与 RDFS 的结构（尤其是 rdfs:subPropertyOf 和 rdfs:subClassOf）相结合时，以及当它们彼此串联时（限制引用其它限制），它们可以用来为属性、类和个体之间的复杂关系建模。以这种方式（胜过非形式化的规范说明）为关系建模的好处是，多个规范说明的交互可以被理解甚至被自动处理。

OWL 也提供其它种类的限制，可以放在一个使用其它种类的 onProperty 限制的类的成员上。

RDF 回答了“我如何能够在一个地方获取有关一个事物的所有信息？”这个问题。对于 RDFS，我们引入了推理的概念，回答了“假定我知道有关我的数据的某些事物，我怎样计算出（figure out）其他的？”RDFS-Plus 和 OWL 的基本应用给了我们更多的从旧信息推出新信息的综合能力。当我们继续到 OWL 的高级功能部件时，我们仍然是在作为我们的模型含义来源的推理范式（paradigm）中工作，但我们扩展了这种推理，不仅包括有关我们的数据的推理，还包括关于模型本身的推理。有了 OWL 的高级结构（construct），我们就可以基于枚举和消除（elimination）参数（以及基于属性和类型的参数，如同我们用 RDFS 和 RDFS-Plus 做的）推断出结论，开放世界的影响会变得更加明显。

基本概念

• owl:Restriction——OWL 的建筑部件，通过限制某些属性的值来描述类。

• owl:hasValue——一种限制，它为一个属性指定一个单一的值。

• owl:someValuesFrom——一种限制，它指定一个集合，一个属性的一些值必须来自这个集合。

• owl:allValuesFrom——一种限制，它指定一个集合，一个属性的所有值必须来自这个集合。

• owl:onProperty——从一个限制到其限制的属性的链接。

• owl:unionOf, owl:intersectionOf, owl:complementOf——应用于类的基本集合操作。每一个都用来产生一个新的类，基于应用于一个或多个已定义类的指定集合操作。

• Open World Assumption——这个思想在第一章引入，但为某些目的关闭世界的策略在本章介绍。

• owl:oneOf——指定一个类只包含列出的成员

• owl:differentFrom——指定一个个体不 owl:SameAs 另一个。当构造计算参数时尤其有用。

• owl:disjointWith——指定两个类不共享一个成员。常常用作 owl:differentFrom 的批发版本。

• owl:cardinality, owl:minCardinality, owl:maxCardinality——基数指定有关某一属性的不同值的数量的信息。与 owl:oneOf, owl:differentFrom, owl:disjointWith 等合用，它可以是基于属性值数量计算的推理的基础。

• Contradiction（矛盾，永假式）——有了 OWL 的高级结构，一个模型就有可能表示一个永假式——也就是，一个在逻辑上不一致的模型。

• Satisfiability(unsatisfiability，不可满足性)——使用 OWL 的高级结构，有可能推理出一个没有成员的类，这样的类是不可满足的。

7 OWL 的“在野”使用：NCI

NCI 本体（译者注：美国癌症研究所基于 NCI Thesaurus 转换成的一个 OWL 本体），由一个单一团体管理的集中式受控词表。

模型的重要方面是它可以被许多有着两种冲突需求的不同的人使用：一方

面，他们需要在他们的工作中有一些共同性（在 NCI 的案例中，全世界的研究者要关联他们的成果）；另一方面，模型的每一个用户都有一些独立的需求，研究者有他们自己的方法和议程。

每一个本体都通过提供一个形式化的、无歧义的和可重用的各自领域概念间的限制模型来调解这些冲突需求。利益攸关方（stakeholder）之间共享的核心概念在一个核心模型中表示，这些核心概念之间拥有的限制也被表示出来。在 NCI 中，这些限制有助于为术语消歧以及跟踪哪个术语是以什么方式被使用的。这些本体都利用了 OWL 的形式化语义来平衡它们的冲突的需求。

这些本体都一再重复一种特定的本体设计模式（pattern）。在 NCI 本体中，连接一棵树中的类与另一棵树中的类的是 owl:someValuesFrom 模式，该模式重复了 5 万次！这样的重复在本体设计中扮演了什么角色？

各自的模式揭示了一种底层模式：信息在该本体描述的领域是如何被组织的。NCI 本体描述的术语空间有许多分面（facets）。在每个分面中的某些术语与另一分面的术语之间有知名的关系。每一对类之间的关系是相同的——也就是，我们如何能够通过术语空间找到我们的路。

从模型维护的角度来看，这些重复的模式给未来的建模者提供了一个机会了解模型是如何工作的以及可以做些什么来修改它们。没有哪一个人可以一次性了解一个有 5 万个类的模型。如果有 5 万个不同的逻辑关系，那么从这个模型中了解由什么推理出结果是一项令人生畏的任务。当然，推理引擎可以计算出推理结果，但是从一个较高的层次上了解现在在干什么，对某些要维护或修改模型的人来说仍然是一个挑战。建模模式的重复简化了这项任务，使模型变得可维护。