使用约束条件支持领域本体的重用*)

明　仲[1,2]　蔡树彬[1]　李师贤[1]

(中山大学计算机科学系　广州510275)[1]　(深圳大学信息工程学院　深圳518060)[2]

摘　要　在利用领域本体知识库(DOKB)中已有领域本体构造新的领域模型或领域本体后，知识工程师需要检查新领域本体以确保其符合规则前提。使用约束条件可以令计算机自动解决这个问题，从而减少知识工程师的工作量，并加快知识的重用过程。

关键词　约束条件，领域本体，重用，演化

Supporting Reuse of Domain Ontology by Using Constraint

MING Zhong[1,2]　CAI Shu-Bin[1]　LI Shi-Xian[1]

(Department of Computer Science, SUN Yat-Sen University, Guangzhou, 510275)[1]

(Faculty of Information Engineering, Shenzhen University, Shenzhen, 518060)[2]

Abstract　Having used domain ontology in DOKB (Domain Ontology Knowledge Base) to construct new domain model or domain ontology, Knowledge Engineer need to confirm that the new domain ontology fulfills the rule bases. Computer can do this automatically by using constraints, which will reduce the workload of the KE and hasten the reusing process of knowledge.

Keywords　Constraint, Domain ontology, Reuse, Evolution

自从20世纪80年代末本体的概念引入计算机领域后，本体的研究越来越受到计算机专家的重视。有关本体的研究和应用在知识工程、自然语言理解、知识表示、软件工程等领域日益受到重视。特别是由于它在智能信息集成、Internet 信息获取、大规模知识库工程等方面取得的成功更使其成为引人瞩目的热点之一[1]。在软件工程领域，本体作为 CASE 工具的基础在 Protégé[2]和 PROMIS[3]等项目中得到广泛应用。在开发 PROMIS 工具的系列项目中，提出了面向本体的领域需求分析 OORA 方法，设计实现了领域描述语言 DODL 和领域分析语言 ONONET 以及基于本体的多领域知识库管理系统 DOKM。PROMIS 工具实现了从领域本体(模型)到可执行软件系统的自动化开发过程。利用 DOKB 中已有的领域知识构造新的领域模型或领域本体具有重要意义[1]。在文[1]解决这个问题后，需要使用计算机自动检测判断新的领域本体是否符合给定的规则前提，领域知识库中知识是否一致，而避免由知识工程师人工检查判断，从而减少知识工程师的工作量并提高新领域本体的质量，加速知识复用的进程，提高开发的自动化程度。

1　本体及本体语言

在知识工程界，Neches 等人在文[4]第一次引进了本体的概念，他们将本体定义为“相关专题的基本术语和关系，以及利用这些术语和关系构成该专题的规划的集合”[4]。后来，Gruber 给出本体更流行的定义，即“本体是领域概念模型的显式表示”[5]。虽然到目前为止，本体仍未有一个统一的定义，但对什么是本体还是形成了一个共识。具体地说，某个领域的本体就是关于该领域的一个公认的概念集以及概念间关联集的一个明确的形式化规格说明。概念含有公认的语义，语义通过概念间的关联体现。领域本体提供了某一领域可共享的、通用的理解。

本体需要用知识表示语言进行编码，并得到相应工具的支持。传统的表示语言有很多，例如基于框架的 Ontolingua、OKBC、OCML，基于框架和一阶谓词的 Flogic、CYCL，基于描述逻辑的 LOOM。还有 OXL (Ontology eXchange Language)，SHOE (Simple Html Ontology Extension)，OML (Ontology Markup Language)，RDF (Resource Description Framework)，RDFs (RDF schema)，OIL (Ontology Interchange Language)，DAML+OIL 等。OIL 是一种目前比较流行的本体表示语言，它结合了基于框架表示的建模原语和描述逻辑简单清晰的语义，并且语法综合了 OKBC、XML、RDF 等。

在 PROMIS 软件自动化 CASE 工具中，领域本体是用于描述指定领域知识的一种专门本体。它给出了该领域实体概念及相互关系、领域活动以及该领域所具有的特性和规律的一种形式化描述[1]。陆汝钤院士等人在 PROMIS 系列工具中，提出了面向本体的领域需求分析 OORA 方法，设计实现了领域描述语言 DODL 和领域分析语言 ONONET 以及基于本体的多领域知识库管理系统 DOKM。使用 ONONET 语言表示的领域本体具有下列特点：①关系是一个独立的知识单元。除继承关系外，其他关系存在于对象外部，允许有限多元关系出现；②将对象组织到本体内部；③将对象作为本体的基本元素。对象是有内部结构的，而这是通常作为本体基本元素的“概念”所不具备的。使用对象作为本体的基本元素可以具有更强的描述和建模能力；④本体组成继承层次；⑤允许本体嵌套；⑥使用“不相关度”组织本体非必需但相关的组成部分[3]。知识工程师在重用 DOKB 中已有的领域知识构造新领域模型(本体)后，需要检查确保新领域本体符合规则前提并保持 DOKB 中领域知识的一致性。通过给本体加上的约束条

*)本文由国家自然科学基金项目(60373084)和广东省科技攻关项目(2003A1030403)资助。明　仲　副教授，博士研究生，主要研究方向为本体论及软件工程。李师贤　教授，博导，主要研究方向为形式语义学及软件工程。蔡树彬　硕士生，主要研究方向为软件工程。

件的方法，一方面可以自动检测领域知识出现不一致的情况，另一方面可以在本体语义相关度匹配[1]的基础上使用约束条件裁剪匹配结果，减少人机交互，从而进一步减少知识工程师的工作量，加速知识的重用。

文[3]中，本体表示为一个六元组(D,P,U,V,H,I)，我们将其扩展为九元组(D,P,U,V,H,I,T,C_E,C_C)，其中D，P,U,V,H,I的定义保持不变，T是一个时间戳，C_E和C_C是约束条件的有限集合。有关约束条件情况见下节，T及其用途见第3节后半部分。

2 约束条件

本体论表示观认为：表示是对自然世界的描述，自然世界唯一绝对精确的表示是其自身，其他表示都不是绝对的逼真，任何表示不可避免地包含简化或人为的规定。本体论中任何表示也都是不完全的知识理论，要做到绝对精确是不可能的[6]。因此，本体的继承只有允许例外的发生，才能更好地支持本体的表示。使用杂交(crossover)等方法构造新领域本体的方法[1,3]，也要求允许本体的多继承。因此，本体的继承应该支持允许例外的多继承机制。文[7]利用生物学上的例子，做了生动的说明，表明允许继承例外和多继承是本体表示所必需的。

允许例外的多继承机制会引入领域知识的不一致，文[8,9]已做介绍。文[1]使用同用户交互的方法解决该问题，系统不提供辅助工具。随着领域知识库描述知识的增加，保持领域知识描述的一致性的工作将非常繁重，会限制领域知识的重用。文[9]提出了一种半自动解决一致性问题的模型，但在将该模型引入到ONONET表示的领域本体时，为解决一致性问题而引入的额外信息将急剧增多，导致知识工程师工作量大幅增加，得不偿失。为了尽量减少知识工程师的工作，并且维持领域知识库描述知识的一致性，本文使用约束条件来检测不一致，把对不一致信息的取舍交由用户交互解决。这种用于检测多继承不一致的约束条件组成的集合，我们称之为存在约束条件集合，记为C_E。

经典逻辑(一阶谓词逻辑)难以表示继承的例外，因此我们使用带模态词M(M表示逻辑一致或相容的)的模态逻辑来表示约束条件。这里，我们用到两个谓词$ISA(X,Y)$(表示X是Y)和$CONTAIN(X,Y)$(表示X包含Y)。约束条件是由这两个谓词构成的一阶模态谓词逻辑的子集L^M表示。在DOKB中的每个元素都必须满足其存在约束条件集合。假设X共继承n_1个祖先，包含n_2个元素，具有n_3个存在约束条件，使用n_4个模态词，同时每个约束条件平均由a个ISA谓词和b个$CONTAIN$谓词组成，则存在约束条件集合的最坏计算复杂度为$O((an_1+bn_2+a+b)*(2^{n_4}+n_3))$。最坏情况的出现是$n_4$个模态词M都互相影响的结果。假设将带模态词的约束条件按相互将存在影响关系的条件分组，平均每组约束条件的个数为c，则复杂度可以改写为$O((an_1+bn_2+a+b)*(2^c*n_4/c+n_3))$，若c与$n_4$无关，则复杂度是$n_4$的线性函数。但是最坏情况出现的概率很低，下文随机测试的结果可证明这一点。

以下是使用存在约束条件集合检测多继承不一致并解释用户可以如何交互解决的一个例子。

下面是关于"孙悟空"等的3个陈述：S1：和尚是素食者；S2：孙悟空是和尚；S3：孙悟空是妖精。显然，"孙悟空"多继承了"和尚"和"妖精"。陈述S1～S3是一致的信息。但如加入下面这样一个规则

S4：妖精不是素食者

此时，陈述S1～S4将产生不一致信息。孙悟空能否是素食者?这就是"孙悟空"多继承引入的不一致信息。使用约束条件检测这种不一致的方法如下：

令$Y(x)$表示$ISA(x,$妖精$)$，即命题x是妖精，$S(x)$表示$ISA(x,$素食者$)$，即命题x是素食者，$H(x)$表示$ISA(x,$孙悟空$)$，即命题x是孙悟空。将规则$S4$表示为"妖精"的约束条件"$\forall x.Y(x)\cap M(\neg S(x))\rightarrow\neg S(x)$"(记为$\sigma_1$，使用模态词$M$是因为这个规则不是必然判断，或表示为$\forall x.ISA(x,$妖精$)\cap M(\neg ISA(x,$素食者$))\rightarrow\neg ISA(x,$素食者$)$)。"孙悟空"因$Y(x)\cap M(\neg S(x))$(继承到"妖精"的约束条件$\sigma_1$)满足，而要求$\neg S(x)$成立。但由于S1和S2则"S5：孙悟空是素食者"，即$S(x)$成立，从而约束条件σ_1不满足，这就检测到了多继承冲突。此时，用户通过交互，若给"孙悟空"增加约束条件"$H(x)\rightarrow S(x)$"，则σ_1将因为$M(\neg S(x))$不成立而得到满足，从而解决了冲突。反之，若希望得到"孙悟空不是素食者"的判断，这是对S1、S2从而S5的继承例外，则可以使用继承例外机制和文[8]的"继承路径"方法解决。

由此可见，使用存在约束条件及允许例外的继承机制可以检测多继承不一致并可通过用户交互解决，从而保证了领域知识库描述知识的一致性。

约束条件的另一个作用，就是在本体语义相关度匹配[1]的基础上使用规则条件进一步约束匹配结果，减少人机交互，从而减少知识工程师的工作量，加速知识的重用过程。由这种用途的约束条件组成的集合称为容器约束条件集合，记为C_C。若元素a是元素e的组成部分，即e包含a，则称元素e及其各组成部分为元素a的容器。若容器不满足元素a的容器约束条件集合，则a自动不在e中出现，即e最终不包含a。

使用容器约束条件的原因是：使用ONONET描述的领域本体，本体的各组成部分都具有对本体的不相关度。除了不相关度为0的组成部分外，本体可以不包含其他组成部分。这些非必需的组成部分的存在具有依赖关系。语义相关度与不相关度具有紧密联系[1]。文[1]使用语义相关度匹配的方法构造虚拟领域本体。若使用预先定义的规则描述这些组成部分间的依赖关系，则可以自动对构造得到的虚拟领域本体进行剪裁，减少软件工程师的工作。假设X共继承n_1个祖先，包含n_2个元素，每个元素平均具有n_3个容器约束条件，使用n_4个模态词，同时每个约束条件平均由a个ISA谓词和b个$CONTAIN$谓词组成，则容器约束条件集合的最坏计算复杂度为$O((an_1+bn_2+a+b)(2^{n_4}+n_3)*n_2)$。下面是使用容器约束条件进行自动剪裁的一个例子。

例如，若"西餐厅"对"餐饮部门"的不相关度是2(即语义相关度是1/2)，"咖啡厅"、"冷饮室"对"餐饮部门"的不相关度都是3(即语义相关度都是1/3)。按语义相关度为1/3匹配，则得到的新领域本体包含"西餐厅"、"咖啡厅"和"冷饮室"。若"冷饮室"存在容器约束条件"西餐厅必须不存在"，按语义相关度为1/3匹配，得到的新领域本体将仅包含"西餐厅"和"咖啡厅"。"冷饮室"因为容器约束条件不满足而不在新领域本体中出现。若用户通过交互删除"西餐厅"，则"冷饮室"可因约束条件满足而出现。

容器约束条件表示组成部分间的依赖关系。元素e通过继承所包含的元素，若其容器约束条件不被满足，则该元素最后将不被元素e所包含。至于容器约束条件的表示和判断则与存在约束条件相同，这里就不复述了。

为了避免由于容器约束条件的使用而产生二义性信息，规定新增容器约束条件不能跟系统现有容器约束条件形成环状依赖。如对上例，若给"西餐厅"增加新的容器约束条件"冷

饮室必须不存在”，则由于这与“冷饮室”已有容器约束条件“西餐厅必须不存在”形成环路，导致新约束条件不能增加。若系统允许同时出现这样两个约束条件，对“餐饮部门”按语义相关度为1/3匹配，将得到两个(允许并行判断则三个)不同的新领域本体。

总之，领域本体包含两种类型的约束条件集合，存在约束条件集合和容器约束条件集合。这两种约束条件的使用，可以加速领域知识的重用过程。

3 约束条件的应用算法

整个领域本体知识库的约束条件并不组成一个约束条件集合，约束条件附加在与其相关的元素上。这样可以减少每次需要判断的约束条件的数量，从而提高判断的速度，但需增加获取元素相关约束条件的获取算法。

约束条件的获取算法

给定领域本体 O_1 的祖先集合 ANCESTOR(O_1)

1. 对 ANCESTOR(O_1)，使用文[8]继承路径的方法，按 *IS-A* 关系拓扑排序，得到父亲本体的集合 FATHER(O_1)。
2. 将 FATHER(O_1)中元素的存在约束条件集合合并为 O_1 的存在约束条件集合。
3. 将 FATHER(O_1)中元素的容器约束条件集合合并为 $O1$ 的容器约束条件集合。

由存在和容器约束条件集合的获取算法可知，对祖先本体约束条件或继承关系的修改，将影响到其后代本体约束条件的变化。这些变化通过不断对直接后代本体使用上述算法获取新约束条件集合得到。步骤1使用文[8]继承路径的方法，限于篇幅省略。

判断约束条件集合是否满足，即判断模态一阶逻辑命题集合是否矛盾，这里我们担心的是约束条件数量增加时，判断耗费时间的大量增加。但由于我们将约束条件分散到各个相关元素中去，每个元素所必须判断的约束条件数量则大大减少，因此判断所耗费的时间仍是可以接受的。

使用约束条件是为了应用到 DOKB 中构造新的本体，在使用文[1]杂交等方法构造出新领域本体 O 后，需要再执行下面的剪裁及检测算法：

剪裁及检测算法

新领域本体 *O*

1. 对领域本体 *O* 所包含的每个元素 *e*，判断 *e* 的容器约束条件集合是否满足。若 *e* 的容器约束条件不满足则分两种情况：
   a) 若 *e* 的语义相关度不为1，则在 *O* 中删除元素 *e*。
   b) 若 *e* 的语义相关度为1，则产生不一致冲突，要求用户交互处理。
2. 调用约束条件的获取算法，获取 *O* 的存在和容器约束条件集合。
3. 判断 *O* 的存在约束条件集合是否满足，若不满足则表明检测到多继承冲突，同用户交互解决冲突问题。
4. 保存修改后的领域本体 O 及其存在和容器约束条件集合到 DOKB 中。

注意到除杂交方法外，使用合成、转基因等方法构造新领域本体后也可继续执行剪裁及检测算法。

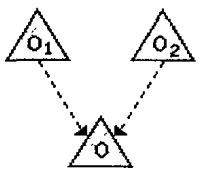

图1 领域本体 O_1 和 O_2 的杂交

图2 目标领域本体（黑圆圈为裁剪部分）

在上述方法中，每个本体存在一个虚拟拷贝保存其指向组成部分的指针和约束条件的集合，虽消耗更多存储空间，并增加了维持虚拟拷贝一致性的开销，但可以加快本体的查找、浏览速度。

领域本体在不断应用中会有一个学习演化的过程。在 DOKB 中，领域本体允许嵌套，一个领域本体的变化，将会影响到其他本体。这些影响有些是我们需要的，有些则不是。容器约束条件的另一个应用可用来解决这个问题。

我们称内容不再随时间变化的本体(对象)为静寂本体(对象)，否则称为活动本体(对象)。静寂本体不受本体演化的影响，活动本体则受本体演化影响。

当我们标记一个本体为静寂本体时，系统获取当前时间为静寂时间戳，并传播静寂时间戳：标记父亲本体静寂和所包含各组成部分静寂，收到静寂时间戳的本体(对象)继续传播静寂时间戳。此后对静寂本体的修改，则转变为增加以静寂本体为父亲的新本体。

令 T_e 表示容器的静寂时间戳，T_x 表示 x 自身静寂时间戳，*NOW* 表示当前时间，每个元素增加的容器约束条件为“$(Te<NOW)\rightarrow(Tx\leqslant Te)$”，即“如果容器静寂时间戳小于当前时间(即存在容器静寂时间戳)，则要求自身静寂时间戳不大于容器静寂时间戳”。同时，令 $NAME(x)$ 表示元素名字，对每个静寂元素 x 增加容器约束条件“$\forall\, y.\ (NAME(x)=NAME(y))\rightarrow(Ty<NOW)\cap(T_y\leqslant T_x)$”。即“如果存在与自身同名的元素，则其静寂时间戳存在并且不大于自身静寂时间戳”。通过增加这样的约束条件，则静寂本体将不会再随时间变化，而活动本体仍可随其他本体变化而变化。

例如，本体“人事部”“餐饮部门”包含对象“员工”，“员工”包含属性“年龄”。我们于2003/1/1将“人事部”标记静寂，则对象“员工”因被“人事部”包含而收到静寂时间戳被标记静寂。2003/6/1我们修改对象“员工”，去掉“年龄”属性而改为“出身年月”属性。因“员工”是静寂对象，此时的修改变为增加“员工”的同名后代对象(下面分别称为“新、旧员工”)，再通过继承例外及增加内容反映修改。“餐饮部门”因仍是活动本体，对象“旧员工”因不满足“$\forall\, y.\ (NAME(x)=NAME(y))\rightarrow(T_y<NOW)\cap(T_y<T_x)$”而不被包含，“新员工”的容器约束条件被满足。同时，“新员工”的容器约束条件不被“人事部”满足，“人事部”包含的仍是“旧员工”。这就解决本体演化的影响问题。

4 OIL 的规则前提(rule-base)

上述约束条件是在 PROMIS 的 DOKB 中使用 ONONET 表示本体的基础上增加的。由于 OIL 是本体交换语言，具有可将其他各种本体语言转换为 OIL 的特色，而且 ONONET 表示的领域知识是从软件工程角度出发表示的一个领域知识，因此 ONONET 可以转换为 OIL。但由于 ONONET 具有的特点，使得由 ONONET 向 OIL 的转换具有一定难度。例如 ONONET 的关系可以是有限多元关系，而 OIL 对多元关系的表示能力不强；ONONET 将有内部结构的对象作为本体的基本元素，而 OIL 采用的是无内部结构的“概念”；ONONET 允许本体嵌套，而 OIL 不支持；ONONET 使用“不相关度”组织的非必需成分必须转换成一定语义相关的必需成分后才能使用 OIL 表示。对这些问题的具体解决方法另文讨论。尽管存在转换的困难，但 ONONET 向 OIL 本体交互语言的转换是可行的。约束条件在转换后变为 OIL 的规

则前提，然而 OIL 并不能自动处理这些规则前提[10]。存在约束条件是本体自身必须满足的规则前提，而容器约束条件则是引入(IMPORT)该本体的本体必须满足的规则前提。对这两种规则前提的区分有利于对规则前提的自动判断处理。由于约束条件使用模态词 M 的非单调一阶逻辑表示，因此具有很强的表达能力。而将约束条件分散到各个相关本体的方法，避免了由大量约束条件组成的全局约束条件或公理集合导致判断时进行的大量计算。这种大量计算是一般非单调逻辑系统在实现时所不得不面对的困难。利用本体描述事物概念化模型的性质，分散约束条件，则可保证这些约束条件既起到作用，又减少计算，从而为 OIL 规则前提的自动判断提供了支持。具体的算法另文讨论。

5 约束条件获取及判断的模拟结果

我们使用(伪)随机数产生约束条件，这阶段耗费的时间模拟为继承(或合并等)旧本体时新本体获取约束条件所耗费的时间。同时，使用20次运行的平均运行时间作为运行时间。下文如无特别指出，运行时间即指20次运行的平均运行时间。

图3是模拟本体库存在10或255个本体时，获取并判断20000条约束条件(不含模态词)的运行时间。子句个数是指将约束条件表示为合(析)取范式时的子句个数。从图3可以看出，本体数量对约束条件运行时间的影响小，约束条件可以运用于大规模本体库。约束条件长度(子句个数)同运行时间成正比。该结果完全在意料之中。

图4是在255个本体，20000个约束条件中有200(1%)、2000(10%)、4000(20%)个约束条件含有1个模态词时的运行时间图。如图4所示，模态词数量的增加对运行时间的影响不大。这表明，模态词的数量同约束条件平均相互影响个数 c (详见本文第2部分分析)不密切相关。这里有部分原因是255个本体可以表示65025个关于 ISA 和 CONTAIN 的不同谓词命题，4000/130050≈3%，出现约束条件相互影响的概率小。这表明引入模态词并不轻易导致指数级增长的计算复杂度，在约束条件中引入模态词增强表示能力是可以接受的。

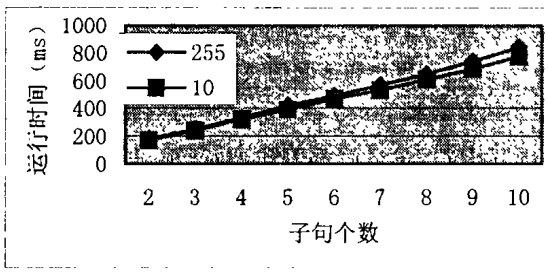

图3 本体个数对运行时间的影响

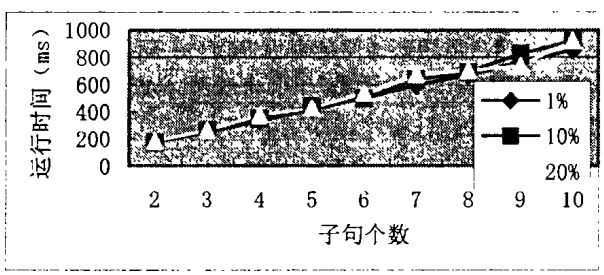

图4 模态词 M 个数对运行时间的影响

图5是在含模态词 M 的约束条件占总数的30%的情况下，20次模拟运行(在不同约束条件集合上)的最短、平均和最长运行时间。均值曲线靠近最短曲线，表明运行出现最短或接近最短运行时间的次数多。这也表明算法达到最坏复杂度的概率小。

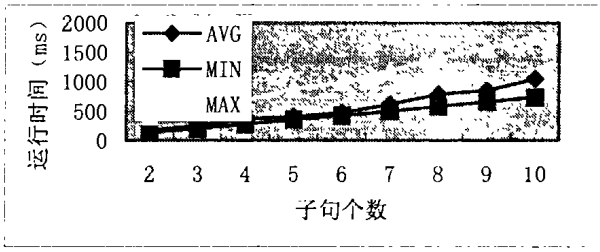

图5 最短、平均和最长运行时间

我们引入模态词 M 来表示约束条件，增强了约束条件的表示能力，但担心由此而导致计算量的大幅增长。模拟实验的结果消除了我们的这一忧虑，表明该方法的使用是可行的。

小结 使用约束条件检测多继承冲突，保证多继承一致性，并解决本体组成部分在重用领域本体以构造新领域本体时的自动裁剪问题。这将进一步减少知识工程师使用 PROMIS 工具开发软件系统的工作量，推动软件自动化开发工具的发展。同时，使用约束条件解决了本体演化所带来的影响问题，并且提出了支持解决 OIL 规则前提自动判断的一个方法。

参 考 文 献

1 陈刚，陆汝钤，金芝. 基于领域知识重用的虚拟领域本体构造. 软件学报，2003，14(3)：350～355

2 Musen M A. Ontology-Oriented Design and Programming：[SMI Report Number：SMI-2000-0833]. http://www-smi.stanford.edu/pubs/SMI_Reports

3 Lu R，Jin Z. Domain Modeling-Based Software Engineering. Boston：Kluwer Academic Publishers，2000

4 Eches R，Fikes R E，Finin T，et al. Enabling technology for knowledge sharing. AI Magazine，1991，12(3)：36～56

5 Gruber T R. A translation approach to portable ontology specifications. Knowledge Acquisition，1993，5(3)：199～220

6 余以胜，张玉峰. 基于本体论的知识库系统研究. 情报杂志，2003，7：2～3

7 Jones D M，Paton R C. Some Problems in the Formal Representation of Hierarchical Knowledge Formal Ontology in Information Systems，N. Guarino(Ed.) IOS Press，1998. 135～147

8 Touretzky D S. The Mathematics of Inheritance Systems. Pitman，London Morgan Kaufmann Publishers，1986

9 Valetian A，Tamma M，Trevor J，Bench-Capon M. Supporting Inheritance Mechanisms in Ontology Representations. In：R. Dieng，O. Corby，eds. EKAW 2000，LNAI 1937，Springer-Berlin Heidelberg，2000. 140～155

10 Horrocks，et al. The Ontology Interchange Language OIL. http://www.ontoknowledge.org/oil