概念的最小上近似

陆建江[1,2,3], 徐宝文[2,3], 康达周[2,3], 李言辉[2,3]
(1.解放军理工大学指挥自动化学院,江苏南京 210007; 2.东南大学计算机科学与工程系,江苏南京 210096; 3.江苏省软件质量研究所,江苏南京 210096)

摘 要: 近似信息检索方法通过找到概念的最小上界,然后求出概念的上近似来解决本体异构问题.现有方法只考虑包含独立概念的最小上界,无法找到概念的最小上近似.引入概念的析取定义概念的多元最小上界,证明了基于多元最小上界得到的近似是概念的最小上近似.多元最小上界通常会存在大量的冗余,从而增加概念最小上近似表达式的复杂性.为此给出概念的最简多元最小上界的定义,并提供了寻找最简多元最小上界的有效算法.

关键词: 信息检索; 本体; 上近似; 多元; 最小上界
中图分类号: TP18 文献标识码: A 文章编号: 0372-2112 (2006) 05-0845-07

The Least Upper Approximation of Concept

LU Jian-jiang[1,2,3], XU Bao-wen[2,3], KANG Da-zhou[2,3], LI Yan-hui[2,3]
(1. *Institute of Command Automation, PLA University of Science and Technology, Nanjing, Jiangsu 210007, China;*
2. *Department of Computer Science and Engineering, Southeast University, Nanjing, Jiangsu 210096, China;*
3. *Jiangsu Institute of Software Quality, Nanjing, Jiangsu 210096, China*)

Abstract The approximate information retrieval approach finds least upper bounds of a concept and then uses them to get upper approximation of the concept to solve this problem of ontology heterogeneity. However, the current method considers the bounds only containing separate concepts, so it cannot get the least upper approximation of the concept. In this paper, disjunction of the concepts is introduced to define multielement least upper bounds, and the approximation based on them is proved the least upper approximation of a concept. In general, multielement least upper bounds may contain much redundancy, which will increase the expression complexity of the least upper approximation of a concept. We also define the simplified multielement least upper bounds and provide effective algorithm to find them last.

Key words information retrieval; ontologies; upper approximation; multielement; least upper bounds

1 引言

关键词的信息检索技术难以得到查询的精确回答[1].基于本体的信息系统可以通过语义来查找信息,从而增加查询的精度[2].本体是“共享概念模型的明确的形式化规范说明”,它的形式化定义可以用七元组 $O=(T, A^T, R, A^R, H, I, X)$描述.其中 T 是概念的集合,概念也称为类,从语义上讲,它是对现实世界中个体的抽象,表示的是个体的集合; A^T 是概念属性的集合,概念间之所以有差异正是由于它们有着不同的属性,才对应着不同的个体集合; R 是关系的集合,一个关系通常包含定义域和值域两部分,这两部分限定了该关系所适用的范围; A^R 是关系属性的集合,关系的属性描述了对关系的进一步限制; H 表示概念层次,概念层次是概念集合 T 上通过 *Kind-of* 或 *Is-a* 关系构成的概念层次结构; I 是实例的集合,实例是现实世界中具体的和唯一的个体,它对应着本体中的一个或多个概念; X 是公理的集合,公理集合中的每条公理代表领域知识中的永真断言.

现实世界中不存在一个覆盖万事万物的统一本体,不同的用户根据不同的应用需求和应用领域来构建或选择合适的本体.即使在同一个领域内也往往存在着大量的本体,这些本体所描述的内容在语义上往往重叠或关联,但

收稿日期: 2004-10-18; 修回日期: 2005-12-05
基金项目: 国家自然科学基金 (No. 60373066, No. 60303024, No. 60403016); 国家 973重点基础研究发展规划 (No. 2002CB312000); 高等学校博士学科点专项科研基金 (No. 20020286004); 江苏省高技术 (No. BG2005032).

所使用的本体表示语言和表示模型上却具有差异,这便造成了本体异构.本体的主要成分有概念、关系、实例和公理等,本文只讨论概念引起的异构问题,例如为了表示"汽车"这个概念,一个本体中使用词汇"Car",而另一个本体中使用词汇"Automobile".解决概念引起的异构问题的一种途径是查询重写.令用户本体为 O_1,系统本体为 O_2,查询重写把关于 O_1 的查询重写为关于 O_2 的等价查询[3],但对于 O_1 中的许多查询,往往只能重写为近似于原查询的查询[4],其中最典型的方法是基于概念蕴涵的近似信息检索[5].

2 近似信息检索

最基本的基于本体的检索是基于概念的检索[5],它包括信息源和查询两部分.信息源是网页、文档等信息项的集合.如果信息源 S 中的每个信息项都被分类到本体 O 中的一个或多个概念,则称 S 为基于本体 O 的信息源.令 O 中全部概念的集合为 T,$(\ )^{I(S)}$ 是解释函数,对于 T 中的每个概念 C,$C^{I(S)}$ 表示被分类到概念 C 的 S 中信息项的集合.顶概念 top 和底概念 bot 是特殊的概念,$top^{I(S)}$ 包含 S 中所有的信息项,$bot^{I(S)}$ 为空.本体中概念间存在蕴涵关系,令 C,D 为 T 中概念,如果 D 是 C 的超类,则称 D 蕴涵 C,记作 $C\subseteq D$.本文规定所有的信息源都与概念蕴涵关系兼容,即有 $C\subseteq D\rightarrow C^{I(S)}\subseteq D^{I(S)}$.

查询是本体中概念构成的布尔表达式.如果 C 是 T 中的概念,那么 C 是一个查询,它的解释为 $C^{I(S)}$;如果 Q,R 是查询,那么析取 $Q\wedge R$,合取 $Q\wedge R$ 也是查询,它们的解释分别为:$(Q\vee R)^{I(S)}=Q^{I(S)}\cup R^{I(S)}$,$(Q\wedge R)^{I(S)}=Q^{I(S)}\cap R^{I(S)}$;如果 Q 是查询,那么 $\neg Q$ 也是查询,它的解释为 $(\neg Q)^{I(S)}=top^{I(S)}-Q^{I(S)}$.查询之间也存在蕴涵关系[4],令 Q,R 是查询,如果 $Q^{I(S)}\subseteq R^{I(S)}$,则称 R 蕴涵 Q,记作 $Q\subseteq R$;如果 $R\subseteq Q$ 且 $Q\subseteq R$,称 Q 和 R 等价,记作 $Q\equiv R$;如果 $Q\subseteq R$ 但 $Q\equiv R$ 不成立,称 R 严格蕴涵 Q,记作 $Q\subset R$.

寻找查询在另一本体中的近似是一个十分耗时的过程.一个可行的方案是预先找到并记录 O_1 中所有概念在 O_2 中的近似,此过程虽然耗时但可以离线进行;当用户提出关于 O_1 的查询时,根据查询中概念的近似可以求得查询在 O_2 中的近似[6],此过程可以快速地在线进行.因此问题的关键在于如何找到并记录 O_1 中概念在 O_2 中的较好的近似.限于篇幅,本文仅讨论概念的上近似,下近似将另外讨论.

定义 1 令 C 为 O_1 的概念,R 是重写 C 得到的关于 O_2 的近似查询,R 在信息源 S 中的查全率和查准率定义为[6]:$\mathrm{recall}(C,R)=|C^{I(S)}\cap R^{I(S)}|/|C^{I(S)}|$ 和 $\mathrm{precision}(C,R)=|C^{I(S)}\cap R^{I(S)}|/|R^{I(S)}|$.如果 $C\subseteq R$,则称 R 是 C 的一个上近似,并有 $\mathrm{recall}(C,R)=1$.如果 R 是 C 的一个上近似,而且对于任何 C 的上近似 R_i,都有 $C\subseteq R\subseteq R_i$,则称 R 为 C 的最小上近似.显然,最小上近似 R 是查全率为 1 且查准率最高的上近似.

O_1 中的概念 C 可能有多个不同的上近似,文献[5]通过定义概念 C 的最小上界来寻找 C 的上近似.令 C 为 O_1 中概念,T 为 O_2 中全部概念的集合.C 的最小上界是 T 的一个子集,记作 $\mathrm{lub}(C,T)$,它满足:(1)对于任何 $D\in\mathrm{lub}(C,T)$,有 $C\subseteq D$;(2)对于任何 $A\in T$ 且 $C\subseteq A$,存在 $B\in\mathrm{lub}(C,T)$ 满足 $B\subseteq A$.找到 C 的最小上界后,定义 $\mathrm{ua}(C,T)=top\wedge\bigwedge_{D_i\in\mathrm{lub}(C,T)}D_i$,可知 $C\subseteq\mathrm{ua}(C,T)$,所以 $\mathrm{ua}(C,T)$ 是 C 的上近似.

$\mathrm{ua}(C,T)$ 通常不是概念 C 的最小上近似,它的近似质量通常是不可接受的.如果 C 远小于它在 T 中的超类,那么也可能远小于 $\mathrm{ua}(C,T)$,这时近似的查准率会很低;最坏情况是找不到 C 在 T 中的超类,那么 $\mathrm{ua}(C,T)$ 就是 top.异构本体常常有全异的概念集合和概念层次,因此最坏的情况也时常会出现.例如,有两个科技论文的分类本体,第一个本体 O_1 描述重点交叉学科的分类,包含概念:"生物信息学(C)"、"生物医学工程"、"纳米科技"等.第二个本体 O_2 描述一般学科的分类,包含概念:"学科(top)"、"医药卫生(A)"、"基础科学(B)"、"工业技术(D)"、"哲学政法"、"社会科学"、"经济财政"、"教科文艺"、"农业科学"等.对于第一个本体 O_1 中的概念 C,由于关于生物信息学的科技论文可能发表在关于医药卫生、基础科学、工业技术的期刊上,因此第二个本体 O_2 中除了顶概念 top 外,其他任意概念都不蕴涵概念 C,此时得到概念 C 的最小上界为 $\{top\}$,上近似为 top.但由于 $C\subseteq A\vee B\vee D\subset top$,显然 $A\vee B\vee D$ 是概念 C 的一个上近似,且比 top 更接近 C.这个例子表明考虑概念的析取通常会得到更好的上近似.本文将引入概念的析取来形式化地定义多元最小上界,并证明由多元最小上界生成的近似是概念的最小上近似.接着分析了多元最小上界通常会存在大量冗余的不足,给出了概念的最简多元最小上界的定义,并提供寻找最简多元最小上界的有效算法.

3 多元最小上界

令 O_1,O_2 为本体,C 为 O_1 中概念,T 是 O_2 中所有概念的集合且 $|T|=n$.称 T 中 i 个概念的集合为 T 中的 i-概念集.如果 $E=\{D_1,D_2,\cdots,D_i\}$ 为 T 中的 i-概念集,则有 $|E|=i$,E 中所有概念的析取称为 T 中 i-概念析取,记作 $E'=D_1\vee D_2\vee\cdots\vee D_i$.下面约定 A,B,D 是 T 中的概念;E,N,U 表示概念集,E',N',U' 表示对应的概念析取.如果有 j-概念集 N 是概念集 E 的子集,称 N 是 E 的 j-子集.多元最小上界的定义类似于最小上界,只是将最小上界中的成员从单个概念扩展到概念析取.

定义 2 令 $\mathrm{mlub}(C,T)$ 是 T 中概念析取组成的集合,如果满足:(1)对于 $\forall E'\in\mathrm{mlub}(C,T)$,有 $C\subseteq E'$;(2)对于 $\forall N\subseteq T$ 且 $C\subseteq N'$,都存在 $E'\in\mathrm{mlub}(C,T)$ 满足 $E'\subseteq N'$,则称

$\mathrm{mlub}(C,T)$是 C的多元最小上界.

令 $\mathrm{mua}(C,T)=top\wedge\bigwedge_{E_i\in\mathrm{mlub}(C,T)}E_i$, 由 $\mathrm{mlub}(C,T)$中的成员都蕴涵 C, 可知 $C\subseteq\mathrm{mua}(C,T)$, 所以 $\mathrm{mua}(C,T)$是 C的上近似. 下面证明 $\mathrm{mua}(C,T)$是 C的最小上近似.

定理 1　对 O_1中任何概念 C, $\mathrm{mua}(C,T)$是 C的最小上近似.

证明: 根据定义 1, 只需证明对任何 C的上近似 Q都有 $\mathrm{mua}(C,T)\subseteq Q$. 因为任何布尔表达式都可以化为等价的合取范式 (CNF), 令 R为与 Q等价的 CNF, 则有 $Q\equiv R=N_1\wedge N_2\wedge\ldots\wedge N_n$, N_i是析取项, 显然 $N_i\subseteq T$, $1\leqslant i\leqslant n$. 对于任意信息源 S和信息项 x, 如果 $x\notin Q^{I(S)}$, $Q^{I(S)}$是查询 Q在 S上的解释. 因为 $Q^{I(S)}=N_1^{I(S)}\cap N_2^{I(S)}\cap\ldots\cap N_n^{I(S)}$, 知至少存在一个 N_i, 使得 $x\notin N_i^{I(S)}$. 因为 Q是 C的上近似, 则有 $C\subseteq Q$, 由此得 $C\subseteq N_i$, 根据定义 2 存在 $E'\in\mathrm{mlub}(C,T)$满足 $E'\subseteq N_i$, 由 $x\notin N_i^{I(S)}$, 可得 $x\notin E'^{I(S)}$. $\mathrm{mua}(C,T)$是 $\mathrm{mlub}(C,T)$中所有成员和顶元素的合取, 现存在 $E'\in\mathrm{mlub}(C,T)$使得 $x\notin E'^{I(S)}$, 所以 $x\notin\mathrm{mua}(C,T)^{I(S)}$. 因此对于任意信息源 S和信息项 x, 由 $x\notin Q^{I(S)}$, 都能推出 $x\notin\mathrm{mua}(C,T)^{I(S)}$, 由此可得 $\mathrm{mua}(C,T)\subseteq Q$.

定理 1表明, 通过 C在 T中多元最小上界可以求得 C的最小上近似. 但是, 概念 C可能有多个 T中的多元最小上界, 通过它们求得的上近似具有相同的解释, 但表达式不同. 另外, 多元最小上界通常会存在大量的冗余成员, 比如, 如果 N, U都在多元最小上界内, 且 $N\subseteq U$, 根据 $\mathrm{mua}(C,T)$的定义, 删除 U不影响 $\mathrm{mua}(C,T)$的结果, 因此 U是冗余的成员; 一般地, 如果 E'在多元最小上界内, 那么把 E中的概念替换成概念的超类或者添加任意概念到 E中得到的概念析取都是冗余的. 大量的冗余成员增加了 $\mathrm{mua}(C,T)$表达式的长度, 降低了查询效率. 为此下面给出概念的最简多元最小上界的定义.

定义 3　令 $\mathrm{slub}(C,T)$是 C在 T中的多元最小上界, 如果对于 $\forall E'\in\mathrm{slub}(C,T)$都有: (1)$\neg\exists N\subseteq T$, 使得 $C\subseteq N\subset E'$; (2)$\neg\exists N'\in\mathrm{slub}(C,T)$, 使得 $N'\equiv E'$; (3)$\neg\exists N\subseteq T$, 满足 $N\equiv E'$且 $|N|<|E|$, 则 $\mathrm{slub}(C,T)$称为是最简的多元最小上界. 第一个条件说明 $\mathrm{slub}(C,T)$只包含蕴涵 C的最小概念析取, 第二、三个条件说明多个等价的概念析取中, $\mathrm{slub}(C,T)$只保留包含概念最少的一个.

4　求最简多元最小上界的算法

概念 C在 T中可能存在多个合法的最简多元最小上界, 约定 $\mathrm{slub}(C,T)$是由算法 1找到的最简多元最小上界. 令概念集集合 $ps=\{E\mid E\in\mathrm{slub}(C,T)\}$, ps为 ps中的 i概念集组成的集合. 算法的目标就是在 T中的所有概念集中找到 ps的所有成员. 判断某个概念集是否属于 ps需要检查它对应的概念析取与其他概念析取之间的蕴涵关系, 蕴涵检查是一个耗时的过程. T中共有 2^n个概念集, 除非 n很小, 否则全部检查是不现实的, 所以算法的目标是尽量减少蕴涵检查. 实际上, 部分概念集存在这样的特性: 它的所有超集都不在 ps中, 这些概念集组成的集合记为 ns, 即 $ns=\{E\mid\forall N\supset E\overrightarrow{\ }N\notin ps\}$. 因此只要证明一个概念集属于 ns, 那么它的所有超集就不再需要检查, 即这些概念集一定不属于 ps. 这样可大大减少蕴涵检查的次数, 比如只要证明概念集 $\{D\}$属于 ns, 则需要进行检查的概念集就只剩下 2^{n-1}个, 减少了一半. 减少搜索空间主要考虑两种途径: 一是利用迭代递增的过程从小到大测试概念集; 二是利用 T中的概念层次处理概念间的蕴涵关系.

4.1　算法主框架

图 1算法采用迭代递增的过程生成概念集并测试它们是否属于 ps. 在第一步, 处理 1-概念集; 在第二步, 处理 2-概念集; …, 在第 i步, 生成的 i-概念集的集合称为 i-候选集, 记作 c_i; 然后测试 c_i中的成员, 判断是否属于 ps_i或 ns. 所有可能属于 ps_i的放到一个集合中等待进一步验证, 称为 i-待验集, 记作 v_i; 一部分成员可以确定属于 ns, 从 c_i中删除这些成员后构成的集合为 r_i. 在第 $i+1$步, c_{i+1}由 r_i生成.

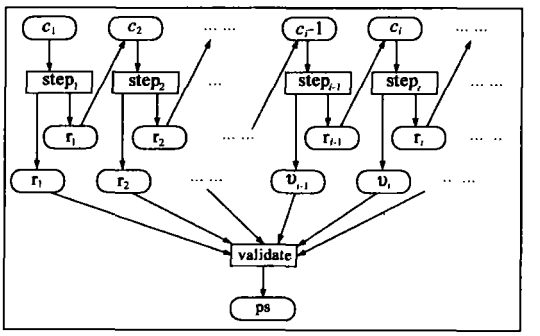


图 1　算法 1的流程图

为了了解算法主框架的基本思想, 下面分析如何从第一步过渡到第二步. 令 $c_1=\{\{D\}\mid D\in T\}$为第一步开始时生成的 1-候选集. 通过测试 c_1中的成员, 可以得到关于它们属于 ps_1或 ns的性质. 一部分成员可能属于 ps_1, 就把它们放到 v_1中; 一部分成员可以确定属于 ns, 从 c_1中删除这些成员后构成的集合为 r_1. 第二步的 2-候选集 c_2只需包含所有 1-子集都在 r_1中的 2-概念集. 因为如果一个 2-概念集 E, 它存在 1-子集不在 r_1中, 由 r_1定义, 则该 1-子集必属于 ns, 由 ns定义, $E\notin ps_2$, 且 $E\in ns$. 由此可知 E本身及其超集都不可能属于 ps, 而算法的目标是求 ps, 所以 E不需要放入 c_2中进行测试. 一般地 c_i为所有 $(i-1)$-子集都在 r_{i-1}中的 i-概念集组成的集合. 下面形式化地定义 c_i, v_i和 r_i, 并给出算法的完备性证明.

定理 2　令 $c_1=\{\{D\}\mid D\in T\}$, $c_i=\{E\mid(|E|=i)\wedge\forall N\subset E,\ |N|=i-1\overrightarrow{\ }N\in r_{i-1})\}$, $1<i\leqslant n$; $c_i\neg ns\subseteq r_i\subseteq c_i$, $c_i\cap ps_i\subseteq v_i\subseteq c_i$, 则对于任何 i, $1\leqslant i\leqslant n$, 有 $ps_i\subseteq v_i$.

证明: 要证 $ps_i\subseteq v_i$, 只需证 $ps_i\subseteq c_i$. 下面用归纳法证明对任何 i, $1\leqslant i\leqslant n$, 都有 $ps_i\subseteq c_i$, 且任何 i-概念集 $E\notin c_i$都满

足 $E \notin ps$ 和 $E \in ns$.

当 $i=1$ 时, 结论显然成立. 假设当 $i=k$ 时, 有 $ps_k \subseteq c_k$, 且任何 k-概念集 $E \notin c_k$ 都满足 $E \notin ps$ 和 $E \in ns$.

则当 $i=k+1$ 时, 对任何 $(k+1)$-概念集 $E \notin c_{k+1}$, 假设 E 的所有 k-子集都在 r_k 中, 那么根据 c_{k+1} 定义, E 一定属于 c_{k+1}, 矛盾. 因此 E 至少有一个 k-子集不在 r_k 中, 不妨令它为 N. 对于 N 有两种可能: 如果 $N \in c_k$, 则由 r_k 定义, 有 $N \in ns$; 如果 $N \notin c_k$, 则根据归纳假设, 有 $N \in ns$. 所以对任何 $(k+1)$-概念集 $E \notin c_{k+1}$, E 至少有一个 k-子集 N 属于 ns. 由 ns 定义可得 $E \notin ps$ 和 $E \in ns$. 因为不属于 c_{k+1} 的 $(k+1)$-概念集都不属于 ps 也不属于 ps_{k+1}, 所以有 $ps_{k+1} \subseteq c_{k+1}$.

定理 2的一个直接推论就是所有 v_i 的并集一定包含 ps. 这就保证了算法结果的完备性. 基于定理 2的算法 1的伪代码见图 2. 算法 1首先生成 c_1. 因为第一步中处理的 1-概念集是独立概念, 所以使用单独的测试函数 find1( )从 c_1 中找到 v_1, r_1. 函数 generate( )根据 c_i 定义从 r_{i-1} 中生成 c_i. 函数 find( )测试 c_i 中成员, 找到 v_i, r_i. 如果 r_{i-1} 是空集, 则满足 $k>i-1$ 的所有 c_k 都为空集, 可进入验证阶段. 函数 validate( )验证所有 v_i 中的成员找到 ps. 测试和验证过程中需要判定概念集之间的蕴涵关系, 令 check( )为判断蕴涵关系的函数. 对于任意概念集 E, N, check(E', N') 判断 E', N' 之间的蕴涵关系, 返回结果有 5种: $E' \equiv N'$, $E' \subset N'$, $N' \subset E'$, $E' \vee N'$ 为空, 或其他. check(C, E') 判断 C, E' 之间的蕴涵关系, 返回结果类似, 但与 check(E', N') 的计算有所不同. 由于 C 和 E' 是两个不同本体中的概念, 无法直接来判断两者之间的关系. 本体中的一个重要成分是实例的集合, 对于两个具有丰富实例的本体, 可以在两个本体的公共实例集上来计算函数 check( )的值. 如果实例不够丰富, 还可以采用人工或一些语义标注工具来为本体创建足够多的公共实例.

```
Algorithm 1. To find ps.
Input C; T.
Output ps
Procedure: begin
    c1 ← {{D} | D ∈ T};
    ps, v1, r1 ← find1(C, c1)
  if ps ≠ Ø then return ps;
  for i = 2 to n begin
        if r(i-1) = Ø then break;
        ci ← generate(i, r(i-1));
        ps, vi, ri ← find(C, ci);
        if ps ≠ Ø then return ps
    end
    v ← U(i=1..n) vi;
    ps ← validate(v);
    return ps
end
```

图 2 算法 1

由于 r_k 决定了 k步之后的搜索范围, 它成员的减少将极大地降低后续步骤需要测试的概念集数目. 另外所有 v_i 中的成员最后都要进行验证. 因此在算法的每一步, 希望得到尽可能小的同时满足定义要求的 v_k 和 r_k.

4.2 测试函数 find1( )

图 3中的函数 find1( )通过测试 c_1 中的成员找到 v_1, r_1. 在介绍函数 find1( )之前, 先引入一个容易证明的结论: 如果有概念 $A, B \in T$, 满足 $A \mid B \mid C$ 或 $A \equiv B$, 令 $T \triangleq T - \{A\}$, 则有 C 在 T 中的最简多元最小上界也是 C 在 T 中的最简多元最小上界, 即对求 C 在 T 中的最简多元最小上界来说, 概念 A 是冗余的. 根据这个结论, 如果 T 中有多个等价的概念, 则只需保留其中的一个, 其他都可以删除. 另外, 注意到如果有 $\{A\} \in c_1$ 满足 $A \equiv C$, 则 A 是 C 在 T 中对应的等价概念, 当然也是最小上近似, 所以有 $ps = \{\{A\}\}$, 算法可以结束. 下面假定 c_1 中没有等价的概念, 也没有与 C 等价的概念.

```
Function find1(C, c1) begin
    forall {A} ∈ c1 do
      forall {B} ∈ c1 that A ≠ B do begin
        check(A, B);
        if A ≡ B then delete {B} from c1
    end
    v1 ← Ø
    forall {A} ∈ c1 do begin
      check(C, A);
    if C ≡ A then return {{A}}, Ø, Ø
      else if C ∧ A ⊆ bot then delete {A} from c1
      else if C ⊂ A then begin
        add {A} into v1;
        forall {B} ∈ c1 and A ⊆ B do delete {B} from c1
    end
    else if A ⊂ C then
        forall {B} ∈ c1 and B ≠ A do delete {B} from c1
    end
    return Ø, v1, c1
end
```

图 3 函数 find1( )

为了避免反复测试 c_1 中的成员, 需要对 c_1 中的概念集定义一个序, 使概念总是先于其超类被测试.

定义 4 T 中概念的权定义为 $w: T \rightarrow N$, 其中 N 为自然数集合, 且 $\forall A, B \in T$, $A \subset B$ 时有 $w(A) < w(B)$. $\forall A, B \in T$, 如果 $w(A) < w(B)$, 则称在 c_1 中 $\{A\}$ 排在 $\{B\}$ 之前.

定义 4规定 c_1 中的概念集是有序的, $\forall \{A\}, \{B\} \in c_1$, 如果 $A \subset B$, 则在 c_1 中 $\{A\}$ 一定排在 $\{B\}$ 之前. forall语句遍历 c_1 时按这个顺序取其中的成员. 接下来是决定哪些 c_1 中的成员应放入 v_1 中. 由定义 3, 如果 $\{A\} \in ps_1$, 则有 $C \subseteq A$, 且不存在 $\{B\} \in c_1$ 使得 $C \subseteq B \vee A$. 因不考虑等价概

念, 规定 $v_1 = \{\{A\} \in c_1 \mid (C \subset A) \wedge (\neg \{B\} \in c_1, C \subset B \subset A)\}$. 容易知道 v_1 满足定理 2中的条件 $c_1 \cap ps_1 \subseteq v_1 \subseteq c_1$. v_1 可由下面的过程求得: 令 tc_1 和 tv_1 为概念集的集合, 初始化 $tc_1 = c_1$, tv_1 为空集. 按规定顺序将 tc_1 中成员和 C 比较, 如果 $C \vee A$, 则把 $\{A\}$ 放入 tv_1, 并从 tc_1 中删除所有满足 $A \subseteq B$ 的 $\{B\}$ (其中包括 $\{A\}$). 当每一个 tc_1 中成员都被取过或删除后, 有 $tv_1 = v_1$.

算法最后决定哪些 c_1 中的成员应放入 r_1 中, 这需要判定每个成员是否属于 ns, 把确定属于 ns 的成员从 c_1 中删除, 剩下的就是 r_1. c_1 概念集中的概念与概念 C 之间有 4种蕴涵关系. 对于 $\forall \{A\} \in c_1$: (1)如果 A 和 C 不相交, 则对 $\forall E \supset \{A\}$, 都有 $E \notin ps$, 所以 $\{A\} \in ns$, 可以从 c_1 中删除. 因为假设 $E \supset \{A\}$, E 属于 ps, 则必有 $C \subseteq E'$, 那么构造一个 $N = E - \{A\}$ 可得 $C \subseteq N' \subseteq E'$, 且 $|N| < |E|$; 根据定义 3, $E \notin ps$, 和假设矛盾. (2)如果 A 是 C 的超类, 即 $C \subset A$, 则对 $\forall E \supset \{A\}$, 都有 $C \vee A \subseteq E'$, 由定义 3, $E \notin ps$, 所以 $\{A\} \in ns$, 可以从 c_1 中删除. (3)如果 A 是 C 的子类, 即 $A \subset C$, 可分两种子情况: (a)如果存在 $\{B\} \in c_1$ 使得 $A \subset B \subset C$, 根据上面的结论, A 是冗余的, 规定 $\{A\}$ 可以从 c_1 中删除, $\{A\} \in ns$. (b)如果不存在 $\{B\} \in c_1$ 使得 $A \subset B \subset C$, 则可能存在 $\{A\}$ 的超集属于 ps, 需保留 $\{A\}$. (4)如果 A 和 C 重叠, 但不满足 $C \subset A$ 或 $A \subset C$, 则可能存在 $\{A\}$ 的超集属于 ps, 需保留 $\{A\}$. 结论是对 $\forall \{A\} \in c_1$, 如果满足情况(1), (2)和(3)(a), 那么 $\{A\}$ 可以从 c_1 中删除; 如果满足情况(3)(b)和(4), 则需保留在 r_1 中. 这样得到的 r_1 显然满足定理 2中的条件 $c_1 \neg ns \subseteq r_1 \subseteq c_1$.

4.3 生成函数 generate( )

图 4中的生成函数 generate( )根据 c_i 定义从 r_{i-1} 中生成 c_i. 为了避免反复测试 c_i 中的成员, 需要对 c_i 中的概念集规定一个序. 定义 4中定义了概念的权, 下面根据概念集中概念的权来定义概念集的序: 如果 E 为概念集, 将 E 中概念按权升序排列, 记 $E[k]$ 为 E 中的第 k 个概念, 概念集的序定义如下.

```
Function generate (i, r_{i-1}) begin
    c_i ← Ø;
    forall E_1 ∈ r_{i-1} do
      forall E_2 ∈ r_{i-1} do
        if E_1[j] = E_2[j] (j = 2 to i-1) and w(E_1[1] < w
(E_2[1]) then
          if not E_1[1] ⊂ E_2[1] then begin
          generate a new concept-set N ← {E_1[1]} ∪ E_2;
          if ∀ U ⊂ N, |U| = i-1 → U ∈ r_{i-1}
          then add N into c_i
        end
      return c_i
    end
```

图 4　函数 generate( )

定义 5　对任何 i, 概念集 E, N, 如果存在 $j \leqslant i$, 有 $w(E[j]) < w(N[j])$, 且对任何 k, $j < k \leqslant i$, 都有 $w(E[k]) = w(N[k])$, 则称 E 在 N 之前.

由 generate( )的执行过程可知, 当 $1 < i \leqslant n$ 时, 如果 r_{i-1} 满足定义 5规定的次序, 则生成的 c_i 自然保持该次序; 由于 r_i 是由 c_i 删除一部分成员产生的, r_i 也会保持该次序. 当 $i = 1$时, 定义 5的序退化为定义 4中的序, 因此 c_1 满足定义 5规定的次序, 同样 r_1 也满足. 由此归纳可得, 整个算法过程中只需在第一步对 c_1 中成员排序, 以后生成的 c_i, r_i 都自然满足定义 5中规定的序.

在生成函数 generate( )中, 由于 $E_1[1] \subset E_2[1]$ 成立时不生成新概念集, generate( )生成的 c_i 中的成员可能少于定理 2中定义的 c_i. 对于任何概念集 E, 只要它包含两个有蕴涵关系的概念, 如 $A, B \in E$, 且 $A \subset B$, 则可构造一个 $N = E - \{A\}$ 使得 $N' \equiv E'$, 且 $|N| < |E|$; 根据定义 3, $E \notin ps$, 而且任何 E 的超集同样包含 A, B, 也不属于 ps, 因此 $E \in ns$. generate( )少生成的 c_i 成员都属于 ns 且不属于 ps, 因此不影响算法的完备性.

4.4 测试函数 find( )

从 c_i 中求 v_i 和 r_i 需要检查概念析取之间的蕴涵关系, 这种检查十分耗时. 为此引入结构蕴涵的概念减少这类检查.

定义 6　令 E, N 是 i 概念集, 如果对 E 中的每一个概念, 都可在 N 中找到它的超类, 则称 E 被 N 结构蕴涵, 记作 $E' \subseteq N'$. 如果有 $E' \subseteq N'$ 但不满足 $E' \equiv N'$, 则记作 $E' \subset^* N'$.

易证, $E' \subseteq^* N' \rightarrow E' \subseteq N'$. 判断 T 中概念析取之间的结构蕴涵不必调用耗时的 check( )函数, 可由定义 6从概念间蕴涵关系推出. 检查结构蕴涵可以删除部分冗余, 但能节省时间, 其他冗余可在验证阶段删除. 易证结构蕴涵有如下性质: 对于 $\forall E, N \in c_i$, 如果 $E' \subset^* N'$, 则 E 在 N 之前.

```
Function find (C, c_i) begin
    v_i ← Ø;
    forall E ∈ c_i begin
      check(C, E');
      if C ≡ E' then begin
        ps ← {E};
        return ps, Ø, Ø
        end
        else if C ⊆ E' then begin
        add E into v_i;
        forall N ∈ c_i and E' ⊆* N' do delete N from c_i
      end
    end
    return Ø, v_i, c_i
  end
```

图 5　函数 find( )

图 5中的函数 find( )通过测试 c_i 中的每个成员判断它

是否属于 ps 或 ns, 从而决定 v_i, r_i. 首先, 如果有 $E\in c_i$ 满足 $E'\equiv C$, 则 E' 是 C 在 T 中对应的等价查询, 当然也是最小上近似, 一定有 $ps=\{E\}$, 算法结束. 因为如果 $ps\neq\{E\}$, 则必存在 N 使得 $|N|<|E|$ 且 $N'\equiv C$, 且算法在第 $|N|$ 步结束, 而 $|N|<i$, 与已测试到 c_i 矛盾. 以下假定 c_i 中不存在和 C 等价的概念析取. 接下来决定哪些 c_i 中成员应放入 v_i. 由定义 3, 如果 $E\in ps$, 则有 $C\subseteq E'$, 且不存在 $N\in c_i$ 使得 $C\subseteq N'\subset E'$. 因为不考虑和 C 等价的概念析取, 可以规定 $v_i=\{E\in c_i\subseteq (C\subset E')\subset(\neg\exists N\in c_i, C\subset N'\subset^{*} E')\}$, v_i 的求解过程与 v_1 类似.

最后决定哪些 c_i 中的成员应放入 r_i 中, 这需要判定每个成员是否属于 ns, 把确定属于 ns 的成员从 c_i 中删除, 剩下的就是 r_i. c_i 中成员对应的概念析取与概念 C 之间有 4 种蕴涵关系. 对于 $\forall E\in c_i$: (1)如果 $E'\wedge C$ 为空, 则 E 中概念都应与 C 不相交, 但这些概念已在第一步中删除, 所以这种情况不可能. (2)如果 $C\subset E'$, 则对 $\forall N\supset E$, 都有 $C\subset E'\subset N'$ 且 $|E|<|N|$, 由定义 3, $N\notin ps$, 所以 $E\in ns$, 可以从 c_1 中删除. (3)如果 $E'\subset C$, 可能存在 E 的超集属于 ps, 需保留 E. 因为不可能存在 $N\in c_i$ 使得 $E'\subset^{*} N'\subset C$, 这里没有类似第一步的子情况. (4)如果 E' 和 C 重叠, 但不满足 $C\subset E'$ 或 $E'\subset C$, 则可能存在 E 的超集属于 ps, 需保留 E. 结论是对 $\forall E\in c_i$, 只有 E 满足情况(2), 即 $C\subset E'$, E 才可从 c_i 中删除. 剩下的都留在 r_i 中. 最后所得的 r_i 一定满足定理 2中的条件 $c_i-ns\subseteq r_i\subseteq c_i$.

4.5 验证函数 validate()

当生成和测试阶段完成后, 有两种可能: 一是已经找到了和 C 等价的查询并赋给了 ps, 这时无需验证, 算法可以结束; 如果没有找到和 C 等价的查询, 由定理 2, 所有 v_i 的并集 v 必然包含 ps 的所有成员. 函数 validate()根据定义 3中最简多元最小上界的条件, 除去 v 中的冗余成员, 得到正确且完备的 ps. 验证函数 validate()见图 6

```
Function validate (v) begin
        forall E, N ∈ v that E≠N do
          if E' ⊂ N' or (E' ≡ N' and |E| ≤ |N|)
          then delete N from v;
        return v
end
```

图 6 函数 validate()

4.6 算法的复杂性分析

整个算法的主要运算是检查概念或概念析取之间的蕴涵关系, 即执行 check()函数. 在第一步 find1()中, 要求出所有 T 中概念间蕴涵关系, 并以此排序, 然后从 c_1 中求得 v_1 和 r_1, 时间复杂度为 $O(n^2)$, 其中 n 是 T 中概念总数. 由 4.2节中的分析知, 算法将 T 中所有冗余的等价概念, 和 C 不相交的概念, C 的超类, 以及 C 的间接子类都从 c_1 中删除, 只有很少的概念留在 r_1 中, 不妨设其数量为 m. 由于本体经常有一个好的组织结构, 实际中往往只有少数概念和 C 相交, 因此 m 远小于 n, 这样即使是在最坏情况下, 之后寻找 v_i 的时间复杂度也不会超过 $O(2^m)$. 假设找到的待验概念集数目为 l, validate()的最坏时间复杂度为 $O(l^2)$; 因为在算法中对 v_i 有严格的定义, 最终的 l 不会很大. 因此算法总的时间复杂度不会超过 $O(n^2+2^m+l^2)$. 以上是最坏情况下的分析, 实际上由于算法的每一步都尽可能地缩减搜索空间, 通常只需检查其中的很少一部分.

更为重要的是, 由于算法采用了迭代递增的过程, 算法执行中可以随时停机并给出一个近似的结果. 例如仅执行一次迭代, 即 $i=1$, 算法得到的结果就是文献[5]方法中使用的概念最小上界, 因此本文的方法是文献[5]方法的一种自然扩展. 随着迭代次数的增加, 结果越来越接近概念的多元最小上界. 通过这些近似结果求得的概念上近似也越来越接近概念的最小上近似. 这个性质说明, 即使在极少数较坏的情况下, 算法也能够在可接受的时间内得到较好的近似结果.

5 结束语

本文采用查询重写的途径来解决概念引起的异构问题, 主要有两方面的贡献. 首先, 文章引入概念的析取来形式化地定义概念的多元最小上界, 并从理论上证明了由多元最小上界生成的近似是概念的最小上近似; 其次, 为了解决多元最小上界中存在的大量冗余, 文章又给出了概念的最简多元最小上界的定义, 提供了寻找最简多元最小上界的算法, 并分析了算法的有效性. 本文仅讨论概念引起的异构问题, 关系等其他本体成分引起的异构问题将是今后有待研究的内容.

致谢 在此, 我们向对本文的工作给予支持和建议的评审专家, 以及汪鹏、周金和李珂玥等同学表示感谢.

参考文献:

[1] Shah U, Finin T, Joshi A, et al. Information retrieval on the semantic web[A]. Proc. of the 11th International Conference on Information and Knowledge Management[C]. Virginia: ACM Press, 2002. 461-468.

[2] Stuckenschmidt H. Ontology-based information sharing in weakly structured environments[D]. Amsterdam: Department of Mathematics and Computer Science, Vrije Universiteit Amsterdam, 2002.

[3] Papakonstantinou Y, Gupta A, Haas L. Capabilities-based query rewriting in mediator systems[J]. Distributed and Parallel Databases, 1998, 6(1): 73-110.

[4] Goasdoué F, Rousset M C. Answering queries using views: a KRDB perspective for the semantic web[J]. ACM Transactions on Internet Technology, 2004, 4(3): 255-288.

[5] Stuckenschmidt H. Approximate information filtering with

multiple classification hierarchies[J]. International Journal of Computational Intelligence and Applications, 2002, 2(3): 295-302

[6] Chang C, Garcia-Molina H. Approximate query mapping accounting for translation closeness[J]. The VLDB Journal, 2001, 10(2-3): 155-181.

[7] Chang C, Garcia-Molina H. Mind your vocabulary: query mapping across heterogeneous information sources[A]. Proc. of the 1999 ACM SIGMOD Conference[C]. Philadelphia: ACM Press, 1999. 335-346.

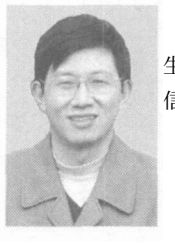
作者简介:

陆建江　男, 1968生于江苏无锡, 博士, 副教授, 主要研究方向为知识工程、数据挖掘等. E-mail: jjlu@seu.edu.cn

徐宝文　男, 1961生于江苏东台, 教授, 博士生导师, 主要研究方向为软件分析与测试、知识与信息获取技术等.

康达周　男, 1980生于江苏南京, 博士生, 主要研究方向为知识工程等.

李言辉　男, 1981生于江苏南京, 硕士生, 主要研究方向为知识工程等.



第二届和谐人机环境联合学术会议征文通知

(2006年 10月 31- 11月 2日, 浙江大学, 杭州, 浙江)

http://www.hhme2006.org.cn

中国计算机学会多媒体专业委员会, 中国图像图形学会多媒体专业委员会, 中国计算机学会普适计算专业委员会和ACM SIGCHI中国分会共同发起的第二届和谐人机环境联合学术会议 (第十五届全国多媒体技术学术会议 NCMT2006, 第二届全国普适计算学术会议, 第二届全国人机交互学术会议)将于 2006年 10月 31日 - 11月 2日在浙江杭州召开。本会议论文集将由清华大学出版社以中国计算机学会系列文集 (CCFP)正式出版。欢迎广大科技工作者踊跃投稿, 中英文论文皆可。会议优秀论文将推荐国内核心刊物发表。

■ 会议征文范围: 多媒体、普适计算、人机交互相关理论、技术与应用。

■ 重要日期:

投稿截止日期: 2006年 7月 1日

录用通知日期: 2006年 7月 31日

最终稿提交日: 2006年 8月 15日

■ 会议程序委员会:

主席: 杨士强 (清华大学), 吴朝晖 (浙江大学), 戴国忠 (中科院软件所)

■ 会议联系人:

多媒体技术:	董亚波 (浙江大学):	13819497136	dongyb@zju.edu.cn
	孙立峰 (清华大学计算机系):	010-62786910	sunlf@tsinghua.edu.cn
普适计算:	潘　纲 (浙江大学计算机学院):	0571-87951647	gpan@zju.edu.cn
	陈　渝 (清华大学计算机系):	62784141-801	yuchen@tsinghua.edu.cn
人机交互:	沈　琦 (浙江大学):	0571-88206681-401	shenq@cad.zju.edu.cn