知识求精的基本理论与方法的研究

杨莉 胡守仁
(浙江大学,杭州 310027) (长沙工学院,长沙 410073)

THE RESEARCH ON THE BASIC THEORY AND METHOD OF KNOWLEDGE REFINEMENT

Yang Li Hu Shouren
(*Zhejiang University*, *Hangzhou* 310027) (*Changsha Institute of Technology*, *Changsha* 410073)

Abstract Knowledge refinement is an important aspect of knowledge acquisition and is used to solve the problem of correctness and efficiency of knowledge base. This paper discusses the basic theory and method of knowledge refinement through the introduction of implementing method of refiner KBRS in the knowledge base constructing tool GKD—KBST.

摘要 知识求精属知识获取的一个重要方面,旨在解决因知识不完全和噪音等因素引起的知识库正确性与有效性问题.本文通过知识库建造工具 GKD—KBST 中求精器 KBRS 实现方案的介绍,论述知识求精的基本理论.

§1. 引 言

知识求精是知识获取必不可少的步骤.一般来说,初始知识库常常存在某些问题,如知识之间不一致、有的知识不正确等等,因而需要调试、修改和补充,即进行知识求精.实践证明,初始知识库经过求精后可以显著提高专家系统的性能和运行效能,例如用 SEEK$_2$ 系统对专家诊断风湿病理学的知识库求精后,其诊断正确率提高了 21.2%.因此,我们在设计知识库建造工具 GKD—KBST 的过程中,为了提高用 GKD—KBST 所构造的知识库的性能和运行效能,专门设计了一个求精子系统 KBRS 作为 GKD—KBST 的核心模块.本文通过介绍 KBRS 的实现方案,论述知识求精的基本理论和方法.

§2. KBRS 中知识库求精技术

模型驱动法是知识库求精(refinement)实用而高效的一种方法.其基本模式是指在某一固定的概念框架下,执行一些实验,并试图去发现能够解释这些实验所得结果的一个理论.其一般框架如下:

给定:· 一阶语言 L 的某个未知模型 M

· 测试 L 中的子句在模型 M 中的正确性

求:一个假设集 H,它在 M 中为真并能蕴含所有正确的测试子句.

因此,为了用模型驱动法来进行知识的求精(此时要求精的知识库中的知识以一阶语言 L 中的子句表示),主要涉及下面两个问题:

本文 1991 年 1 月 15 日收到,1991 年 3 月 25 日定稿.作者**杨莉**,1992 年博士毕业于国防科技大学,现为浙江大学计算机系博士后,主要研究领域为知识获取,非单调推理,神经网络等.**胡守仁**,教授,主要从事计算机系统结构,人工智能,人工神经网络等方面的工作.

(1)如果一个假设(子句)是太强时，如何去减弱它；

(2)如果一个假设(子句)是太弱时，如何去增强它.

我们说一个(推测)假设集相对于某个模型 M 来说是太强了，如果它包含了 M 中一个错误的观察实例，我们说它是太弱了，如果它没有包含 M 中一个正确的观察实例. 2.2，2.3 节我们介绍矛盾回溯算法用于解决第一个问题，求精算法用于解决第二个问题.

2.1 矛盾回溯算法

在当前推测太强的情况下，可知至少存在一个假设是错误的，矛盾回溯算法能够回溯假设和事实之间的矛盾，找到引起矛盾的一个错误假设，无论什么时候矛盾被假设和事实所推出，此算法都可以被用来找出一个错误的假设.

在一阶语言 L 上所定义的矛盾回溯算法是根据构造假设的反例来探测错误假设的思想，在描述此算法之前，我们沿用 Robinson 的定义给出关于归结的基本概念.

定义 2.1.1：令 $A\leftarrow B, C\leftarrow D$ 是 L 的二个子句，$R_1\subset B, R_2\subset C$，如果对于 R_1 和 R_2 存在一个最一般的一致化置换 θ，使得 $\{p\}=R_1\theta=R_2\theta$，那么 $[A\cup(C-R_2)\leftarrow(B-R_1)\cup D]\theta$ 是 $A\leftarrow B, C\leftarrow D$ 的一个归结，且 p 被称为归结原子，这里 $A\leftarrow B$ 称为归结的左部，而 $C\leftarrow D$ 称为归结的右部.

由于在谓词演算中，归结原子 p 不必是基本原子，因而我们不总是可以用 M 的预言者(user or expert)来直接测试其真值. 解决这一问题的办法是在把 p 送给预言者之前先把 p 实例化成一基本原子. 如何对 p 进行实例化，其方法是任意的，但一旦它被实例化后，进一步的后继工作应该在相同置换的条件下进行，其目标是使它们的结果能够构造出对一个假设的反例. 下面给出矛盾回溯算法.

算法 2.1：矛盾回溯算法

输入：模型 M 的一个预言者及满足下列特性的一个有序二叉树.

(1)二叉树的根是空子句；

(2)二叉树的树叶或者为一假设或者为一个在 M 中为真的观察子句，且没有二个叶结点共享同一变量；

(3)二叉树中不为叶结点的其它任一结点都是其左右儿子的一个归结，其中左儿子为归结的左部件，右儿子为归结的右部件.

输出：一个假设 H，它是二叉树的一个叶结点，且在 M 中为假.

算法：

```
k=0;θ0={},(θ0 为初始置换集)
N0 指示二叉树的树根
while  Nk 不是叶结点时  do
令 p 为 Nk 使用最一般的一致化置换 θ 所得到的归结原子
选择一置换 θ′,使得 pθk 被实例化为一基本原子 Pk
令 θk+1=θ·θk·θ′ (这里"·"为复合置换操作)
测试 Pk 在 M 中的正确性
case Pk
        true:置 Nk+1等于 Nk 的左儿子
        false:置 Nk+1等于 Nk 的右儿子
K:=K+1;
输出 Nk
```

下面我们更明确地定义实验结果是如何构造一个假设的反例的. 对二个子句 $A\leftarrow B, A'\leftarrow B'$，

如果有置换 θ,使得 $A'\theta\subset A$ 且 $B'\theta\subset B$. 我们称子句 $A'\leftarrow B'$ 包含在子句 $A\leftarrow B$ 中,即 $(A'\leftarrow B')\theta\subset(A\leftarrow B)$. 设基本原子 $P_1,\cdots,P_k$ 被测试,令 $B=\{P_i|P_i$ 在 M 中为真,$i=1,2,\cdots,k\}$,$A=\{P_i|P_i$ 在 M 中为假,$i=1,2,\cdots,k\}$,则基本子句 $A\leftarrow B$ 在 M 中为假,类似地我们可得任何包含在 $A\leftarrow B$ 之中的子句在 M 中也为假. 在此条件下可得 $A\leftarrow B$ 为 p 的反例,这是通过测试 $P_1,\cdots,P_k$ 的结果经 θ 所构出.

定理 2.1.1:令 M 为一阶语言 L 的一个模型,T 为 L 中带有空子句"□"的一个有序二叉归结树. 假如我们把算法 2.1 用于 T 并通过测试基本原子 $P_1,\cdots,P_k$ 后得到子句 N_k,则 N_k 在 M 中为假且上述测试结果通过 θ_k 构造了 N_k 的一个反例.

证明:对 k 应用归纳法来证明.

1. 当 $k=0$ 时,$N_0=$"□",这时 N_0 在 M 中为假且 $\theta_0=\{\}$,此时 N_0 的反例为 $\square\theta_0$——一个空子句.

2. 假设算法 2.1 测试基本原子 $P_1,\cdots,P_k$ 后到达了结点 N_k,并且测试结果通过置换 θ_k 构造了子句 $N_k=(A'\leftarrow B')$ 的一个反例 $A\leftarrow B$. 令 p 为 N_k 上的归结原子,这里 N_k 的左儿子为 $A_1\leftarrow B_1\cup R$,右儿子为 $A_2\cup R_2\leftarrow B_2$,且集合 R_1,R_2 有一个最一般的一致化置换 θ,使得 $R_1\theta=R_2\theta=\{p\}$ 且 $(A'\leftarrow B')=((A_1\cup A_2)\leftarrow(B_1\cup B_2))\theta$. 假定 $p\theta_k$ 通过置换 θ' 被实例化成一个基本原子 P_{k+1},则 $\theta_{k+1}=\theta\circ\theta_k\circ\theta'$. 测试 P_{k+1},它在 M 中有下列两种情况:

① P_{k+1} 在 M 中为真. 这时算法 2.1 置 N_{k+1} 为 N_k 的左儿子,即 $A_1\leftarrow B_1\cup R_1$. 由归纳假设:$A\leftarrow B$ 在 M 中为假且 P_{k+1} 在 M 中为真可知,子句 $A\leftarrow B\cup\{P_{k+1}\}$ 在 M 中为假.

因为 $(A_1\leftarrow B_1)\theta\subset N_k$,所以 $(A_1\leftarrow B_1)\theta\circ\theta_k\circ\theta'=(A_1\leftarrow B_1)\theta_k\subset N_k\theta_k\circ\theta'$ 成立.

又因为 $N_k\theta_k$ 为基本原子,所以 $N_k\theta_k\circ\theta'=N_k\theta_k$. 而 $N_k\theta_k\subset(A\leftarrow B)$,所以我们可知:

$(A_1\leftarrow B_1)\theta_{k+1}\subset(A\leftarrow B)$ 成立. 同时,由于 $R_1\theta=p$ 且 $p\theta_k\circ\theta'=P_{k+1}$,可得出:

$R_1\theta_{k+1}=P_{k+1}$. 因此,$N_{k+1}\theta_{k+1}=(A_1\leftarrow B_1\cup R_1)\theta_{k+1}\subset(A\leftarrow B\cup\{P_{k+1}\})$ 成立,即通过 θ_{k+1} 置换,使 N_{k+1} 包含在 $A\leftarrow B\cup\{P_{k+1}\}$ 中. 因此 N_{k+1} 在 M 中为假,且 $P_1,\cdots,P_k$ 的测试结果通过置换 θ_{k+1} 构成了 N_{k+1} 的一个反例.

由归纳法可知,在情况①,原结论成立.

② P_{k+1} 在 M 为假. 证明同①类似,不再叙述.

根据①和②,由归纳法可知,原结论成立.

2.2 求精算法

在 2.1 节中所述的矛盾回溯算法通过在太强的推测(假设集)中探测出一个错误的假设解决了设计递增的模型推理算法所遇到的第一个问题. 然而,从推测中移出这样一个错误假设可能导致一个太弱的推测,这样第二个问题就出现了. 本节讨论如何加强这种推测,给出了我们在假设子句 L_h 上设计的一个求精算子.

根据 Reynolds(1970)关于子句大小的定义,我们把 L 上的测度 size 定义为:一个子句 p 的 size,即 size(p)等于 p 中所有符号的总数(不包含标点符号)减去 p 中所出现的变量数.

定义 2.2.1:设 L 为一阶语言,p 和 q 为 L 的子句,若 p 蕴含 q,即 $size(p)\leqslant size(q)$,则称 q 为 p 的精化.

定义 2.2.2:求精算子 ρ 是 L 中子句到它们精化子集的映射,使得对任一 $p\in L$ 和任何 $n>0$,集合 $\rho(p)(n)=\{a|size(a)\leqslant n,a\in\rho(p)\}$ 是可计算的.

L 上的求精算子导出了 L 上的一个偏序关系 $\leqslant_\rho$,对 $\leqslant_\rho$ 而言,有最小元素"□"(空子句). 子句的一个有限序列 $p=p_0,p_1,\cdots,p_n=q$ 且 $p_{i+1}\in\rho(p_i)$,被称为一个有限完全 ρ 链,如果从 p 到 q 有一

个有限完全ρ链，则$p\leqslant_\rho q$成立. 若$p\leqslant_\rho q$且$q\not\leqslant_\rho p$则$p<_\rho q$. 对任意二个子集$T\subset L$，$S\subset L$，若对于任何$q\in S$，必有一$p\in T$使得$p\leqslant_\rho q$成立，则$T\leqslant_\rho S$. 对任一子句p，定义$\rho^*(p)=\{q\in L\mid p\leqslant_\rho q\}$，$\rho^*=\rho^*(\square)$.

定义 2.2.3：设S为一子句集且$S\subset L$，$\square\in S$，那么L上的一个求精算子ρ对S而言是完备的，仅当$\rho^*=S$.

根据求精算子的应用，当假设被反驳时，需要在该假设的精化子集中寻找其替换. 因此，要求求精算子对假设语言$L_h(L_h\subset L)$是完备的.

下面，我们定义"→"关系，并考虑由$\rho_1(p)=\{q\mid p\to q\}$所定义的操作$\rho_1$. 设$p\in L$为一子句，如果下面条件之一被满足，则$p\to q$.

(1)对L中的某一n元谓词符a，$n\geqslant 0$，$p=$"$\square$"，且$q=a(X_1,\cdots,X_n)$，这里$X_1,\cdots,X_n$为n个相互独立的变量；

(2)$p=p_0$，p_0为一原子，$q=p_0\{u/v\}$，这里u，v为p_0中相互独立的变量；

(3)$p=p_0$，p_0为一原子，$q=p_0\{v/f(X_1,\cdots,X_n)\}$，这里f为L中的n元函数符，$n\geqslant 0$，v为$p_0$中的变量，且$X_1,\cdots,X_n$为$p_0$中不出现的n个相互独立的变量.

定理 2.2.1：ρ_1为L上的一个求精算子，且ρ_1对L中的原子是完备的.

然而，仅包含原子的系统是过于简单化了，下面我们给出ρ_1的一个扩充ρ_2，使其能在更复杂的公理系统中适用.

定义 2.2.4：令ρ和ρ′为一阶语言L上的二个求精算子，我们说ρ′比ρ更一般，如果$\rho^*\subset\rho'^*$成立.

考虑如下所示ρ_1的一般化ρ_2，令$p\in L$为一子句，如果下列条件之一成立，则$q\in\rho_2(p)$.

(1)$q\in\rho_1(p)$；

(2)$p=a(t_1,\cdots,t_n)$，其中a为一n元谓词符，$t_1,\cdots,t_n$为项，$q=a(t_1,\cdots,t_n)\leftarrow a(X_1,\cdots,X_n)$. 这里$X_1,\cdots,X_n$为相互独立的变量且$X_i$出现在$t_i$中，$1\leqslant i\leqslant n$.

在上述定义中，我们把形如$\{p\}\leftarrow\{Q\}$的子句称为变形，且满足条件(2)的变形称为一上下文无关变形.

定理 2.2.2：ρ_2为L上的一个求精算子，且对L中的原子和上下文无关变形是完备的.

§3. 知识库调试技术

初始知识库经过§2节介绍的方案进行求精后，可能在知识库中存在某些冗余，并在知识库的规则(子句)间可能存在循环支持键，因而需要对知识库进行调试. 下面我们给出其基本概念和方法.

定义 3.1：冗余的知识定义如下：

(1) 两条一模一样的事实或规则

(2) 两条规则间有下列关系：

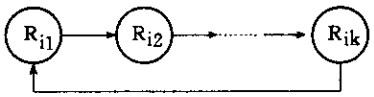
(a) $P(X)\to Q(X)$，　　$\underline{P(a)\to Q(X)}$

(b) $\underline{P(X),Q(X)\to R(X)}$，　　$P(X)\to R(X)$

(c) $P(X)\to Q(X)\to\ldots\to R(X)$，　　$\underline{P(X)\to R(X)}$

这里，划横线者为冗余.

定义 3.2：若规则$R_{i1},\cdots,R_{ik}$之间存在如下所示的依赖关系：则称规则$R_{i1},\cdots,R_{ik}$构成了一个环，如图：



定义 3.3:若知识库中的规则集 $IR=\{R_1,\cdots,R_n\}$,对应地可构造一个有向图 $G_{IR}=\langle V_{IR},E_{IR}\rangle$如下:

顶点集:$V_{IR}=IR$;边集:$E_{IR}=\{\langle R_i,R_j\rangle|$若 R_i 依赖 $R_j\}$.

定义 3.4:设图 $G=\langle V,E\rangle$的结点集为 $V=\{V_1,\cdots,V_n\}$,则 n 阶方阵 $A=(a_{ij})$称为 G 的邻接矩阵.其中,当$\langle V_i,V_j\rangle\in E$ 时,$a_{ij}=1$,否则 $a_{ij}=0$.

定义 3.5:设矩阵 $A=(a_{ij})_{n\times n}$,我们说它的元素积 $a_{k_1l_1},a_{k_2l_2},\cdots,a_{k_ml_m}$构成了一个循环项是指:$l_1=k_2,l_2=k_3,\cdots,l_m=k_1$.

为了给出判定规则之间构成环的算法,我们在此给出图论中的两个定理.

定理 3.1:在一个具有 n 个结点的图中,如果从结点 V_i 到结点 V_j 存在一条路径,则从结点 V_i 到结点 V_j 之间必然存在一条不多于 n−1 条边的路径.

定理 3.2:设 A 是有向图的邻接矩阵,则 A^k 的元素 $a_{ij}^{(k)}$等于结点 V_i 与结点 V_j 间长度等于 k 的路径的数目.特别地,其对角线元素 $a_{ii}^{(k)}$等于经过结点 V_i,长度等于 k 的循环的数目$(i,j,k=1,\cdots,n)$.

由上述两个定理可知,要判断一个具有 n 个结点的有向图的结点间是否存在循环,只需计算 $A,\cdots,A^n$,利用它们的对角线元素进行判断:$\sum_{k=1}^{n}tr(A^k)$包含了矩阵 A 元素之间所有可能的循环项(注:tr(A)表示矩阵 A 对角线上元素的和).

结束语:本文介绍的知识库求精和调试技术作为 KBRS 的核心部分已在 micro VAX Ⅱ上使用 prolog 语言实现,实际应用结果表明,KBRS 即可作为一个独立的系统用于智能系统的调试,也可用于逻辑程序系统的调试,其实用面比较宽,且其所使用的关键技术具有坚实的理论基础.

参考文献

[1] Politakis, P. and Weiss, S. M.: Using Empirial Analysis to Refine Expert System Knowledge Bases, AI, 1984, 22.

[2] Ginsberg, A. and Weiss, S. M.: Automatic Knowledge Base Refinement for Classification System, AI, 1988, 35.

[3] Popper, K. R.: Conjectures and Refutations: The Growth of Scientific Knowledge, Harper Torch Books, 1968.

[4] Reynold, J. C.: Transformational Systems and Algebraic Structure of Atomic Formulas, Edinburgh University Press, 1970.