国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第13卷第4期 1991年12月 Vol. 13 No. 4

知识库调试工具的技术原理与设计思想

杨 莉 胡守仁
（电子计算机系）

摘 要 知识库调试工具旨在解决因知识不完全和噪音等因素而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 中知识库调试工具 KBDT 实现方案的讨论，论述知识库调试工具的技术原理与设计思想。

关键词 人工智能，知识求精，知识库调试
分类号：TP311

知识获取是研制智能系统的瓶颈问题。根据 A. Ginberg 等的观点，知识获取过程可分为两个阶段：

(a) 初始知识库建造阶段；(b) 初始知识库调试和求精阶段。

由于开始得到的初始知识库常常存在某些问题，如知识不完全、知识不正确、知识之间的不一致性等等，因此需要对知识库进行调试、修改和补充。一般来说，知识库调试和求精可分为以下三步：

(1) 调试初始知识库，获得求精的信息；(2) 对初始知识库进行精化；(3) 调试求精后的知识库，如已符合要求则停止，否则重复(2)，(3)两步，直至知识库符合要求，初始知识库被精化成高性能的知识库为止。

因此，在知识库建造工具 GKD-KBST 中，为了提高它所建立的知识库的性能和运行效能，需要专门设计一个知识库调试工具 KBDT 作为 GKD-KBST 的核心模块。KBDT 知识库调试工具主要包括如下关键技术：

(a) 矛盾跟踪技术；(b) 特殊化和一般化求精技术；(c) 正向推理技术；(d) 规则间构成环的判定和消环技术；(e) 冗余知识的判定和删除、规则的最优化排列技术。

1 KBDT 中知识求精技术

模型驱动法是知识求精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上，进行一些实验，并试图去发现能够解释这些实验结果的一个理论。其框架如下：

给定：一阶语言 L 的某个未知模型 M；

L 中用于验证模型 M 正确性的测试子句。

求： 一个假设集 H，它在 M 中为真并能蕴涵所有正确的测试子句。

* 1990年9月20日收稿

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：

（1）如果一个假设（子句）太强，如何去减弱它。

（2）如果一个假设（子句）太弱，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。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。

1.1 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，先给出关于归结的基本概念。

定义 1.1.1 令 $A\leftarrow B$，$C\leftarrow D$ 是 L 的两个子句，$R_1\subset B$，$R_2\subset C$，如果对于 R_1 和 R_2，存在一个最一般的一致化置换 Q，使得 $\{P\}=R_1Q=R_2Q$，那么 $[A\cup(C-R_2)\leftarrow(B-R_1)\cup D]Q$ 是 $A\leftarrow B$，$C\leftarrow D$ 的一个归结子句，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 或 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。对 P 进行实例化的方法是任意的，但一旦它被实例化后，进一步的后续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 1.1 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两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

$K=0$；$Q_0=\{\}$（Q_0 为初始置换集）；

N_0 指示二叉树的树根；

while N_k 不是叶结点时 *do*；

令 P 为 N_k 使用最一般的一致化置换 Q 所得到的归结原子；

选择一置换 Q'，使得 PQ_k 被实例化到一基本原子 p_k；

令 $Q_{k+1}=Q\circ Q_k\circ Q'$（这里“$\circ$”为复合置换操作）；

测试 p_k 在 M 中的正确性；

Case p_k

true：置 N_{k+1}等于 N_k 的左儿子。

false：置 N_{k+1}等于 N_k 的右儿子。

$k_1=k+1$；

输出 N_k.

下面更明确地给出实验结果是如何构造一个假设的反例的。对两个子句 $A\leftarrow B$，$A'\leftarrow B'$，如果有置换 Q，使得 $A'Q\subset A$ 且 $B'Q\subset B$，称子句 $A'\leftarrow B'$ 包含在子句 $A\leftarrow B$ 中，即 $(A'\leftarrow B')Q\subset(A\leftarrow B)$. 设基本原子 $p_1,\cdots,p_k$ 被测试，令 $B=\{p_i\mid p_i$ 在 M 中为真，$i=1,\cdots,k\}$，$A=\{p_i\mid p_i$ 在 M 中为假，$i=1,\cdots,k\}$，则基本子句 $A\leftarrow B$ 在 M 中为假。类似地可得任何包含在 $A\leftarrow B$ 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 Q 所构出。

定理 1.1.1　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 1.1 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 Q_K 构造了 N_K 的一个反例（证明从略）。

1.2 求精算子

上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，解决了设计知识求精技术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，给出在假设子句集 L_h 上设计的一个求精算子。

根据 Reynold 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 1.2.1　设 L 为一阶语言，p 和 q 为 L 的子句，若 p 蕴涵 q，即 size(p)$\leqslant$size(q)，则称 q 为 p 的精化。

定义 1.2.2　求精算子 ρ 是 L 中子句到它们的精化子集的映射，使得对任一 $p\in L$ 和任何 $n>0$，集合 $\rho(p)(n)=\{a\mid$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 $p<_\rho q$. 对任意两个子集：$T\subset L$，$S\subset L$，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 1.2.3　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 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。

下面定义“$\rightarrow$”关系，并考虑由 $\rho_1(p)=\{q\mid p\rightarrow q\}$所定义的操作 ρ_1. 设 $p\in L$ 为一子句，如果下面条件之一被满足，则 $p\rightarrow q$.

(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，…，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、…、x_n 为 p_0 中不出现的 n 个相互独立的变量。

定理 1.2.1 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（证明从略）。

然而，仅包含原子的系统是过于简单化了。下面给出 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 1.2.4 令 ρ 和 ρ' 为一阶语言 L 上的两个求精算子，称 ρ' 比 ρ 更一般，如果 $\rho^*\subset\rho'^*$ 成立。

考虑如下所示的 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，…，t_n 为项，$q=a(t_1,\cdots,t_n)\leftarrow a(x_1,\cdots,x_n)$. 这里 x_1，…，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 1.2.2 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。(证明从略)。

2 KBDT 中知识库调试技术

初始知识库经过第一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链，因而需要对知识库进行调试。下面给出其基本概念和方法。

定义 2.1 冗余的知识定义如下：

(1) 两条一模一样的事实或规则；

(2) 两条规则间有下列联系：

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

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

(c) $P(X)\rightarrow Q(X)\rightarrow\cdots\rightarrow R(X)$，　　$\underline{P(X)\rightarrow R(X)}$

这里，划横线者为冗余。

定义 2.2 若规则 R_1，…，R_k 之间存在如下所示的依赖关系，则称规则 R_1、…、R_k 构成一个环，如图 1.

定义 2.3 若知识库中的规则集为 $R=\{R_1,\cdots,R_n\}$，对应 R 可以构造一个有向图 $G_R=<V_R,E_R>$ 如下：

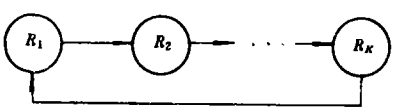

图 1

(1) 顶点集 $V_R=R$；

(2) 边集：$E_R=\{<R_i,R_j>|$，若规则 R_j 的条件部分包含规则 R_i 的结论部分$\}$。

定义 2.4 设图 $G=<V,E>$ 的结点为 $V=\{v_1,\cdots,v_n\}$，则 n 阶方阵 $A=(a_{ij})$ 称为 G 的邻

接矩阵。这里，若$<v_i,v_j>\notin E$，则$a_{ij}=0$. 若$<v_i,v_j>\in E$，则$a_{ij}=1$.

定义 2.5 设矩阵$A=(a_{ij})_{n\times n}$，称其元素所形成的链：$a_{k_1l_1}\cdots a_{k_ml_m}$构成一个环，如果：$l_1=k_2$，$l_2=k_3$，…，$l_m=k_1$.

下面给出利用图论的方法来判定知识库中存在环的方法。

定理 2.1 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必然存在一条不多于$n-1$条边的路径。

定理 2.2 设A是有向图G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的路径数目。特别地，其对角线元素$a_{ij}^{(K)}$等于经过结点v_i，长度等于K的循环数目。

由定理2.1和2.2可知，要判断一个具有n条规则的知识库中是否有环存在，只需判断其对应的有向图的结点之间是否存在循环，即通过计算A(A对应有向图的邻接矩阵)，…，A^n，再根据A，…，A^n的对角线元素进行判断。

3 KBDT的实现方案

KBDT针对智能系统的知识库进行调试和求精，待调试的知识库用一阶语言L中的Horn子句形式来表示。KBDT作为一个独立的调试工具其总体框图如图2所示，其数据流图如图3所示。

由图2所示，整个KBDT包含五个功能模块和一个实例库。实例库中所存放的知识分为如下三种情况：

(1) 系统中已有正例集和反例集（可为空）；

(2)矛盾知识的定义，初始情况为contradiction：—p，not (p).

(3) 通过预言者专家给出的事实性知识，形如：fact (p，v)，p为当前事例，v为true或false.

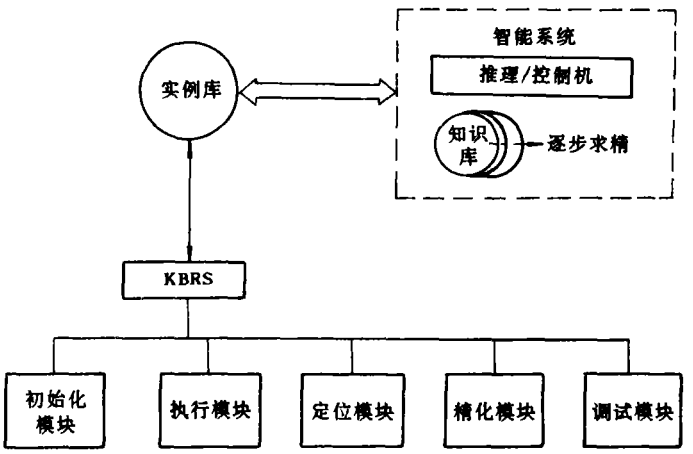

图2 KBDT系统框图

KBDT的运行方式是实时求精，即每当外界知识加入知识库时，都激活KBDT对这些知识进行调试和求精。下面逐个模块地介绍KBDT的实现方案。

3.1 初始化模块

此模块为KBDT的运行做一些准备工作。此时系统请求用户输入知识库名；然后系统将此知识库中的规则进行重排，以使控制程序搜索该知识库所用的可能时间达到最小；接着将知识库中的知识变成内部表示装入系统，同时将实例库也装入系统。在该功能模块中，定义操作：(1) 输入知识库名 (get-kb-name)；(2) 重新排列规则 (reorder-rule)；(3) 变知识库为内部表示 (meta-knowledge)；(4) 装入文件 (insert-file)。

在 reorder-rule 操作中，使用的是无环规则集的最优排列算法。这里所说的最优是指：若知识库中的知识按这种算法进行排列，则控制程序搜索该知识库所用的可能时间将达到最小。下面给出其方法：

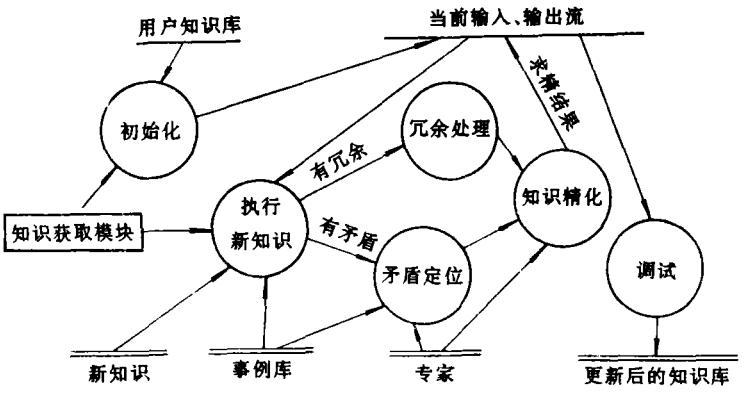

图 3　KBRS 的数据流图

定义 3.1.1　若规则 R_j 的条件部分包含规则 R_i 的结论部分，则规则 R_j 依赖规则 R_i.

这时，为了提高控制程序的搜索效率，在知识库中，应把 R_i 放在 R_j 之前。不难看出，依赖关系具有传递性。

定义 3.1.2　如果规则 R_i 被 m_i 个其它规则所直接或间接地依赖，则称 m_i 为规则 R_i 的被依赖次数。

规则 R_i 的被依赖次数 m_i 越大，说明 R_i 被激活的可能性越大。我们设计的最优排列算法，就是按其被依赖次数从大到小来排列规则的。

3.2　执行模块

此模块当新知识（外界知识）加入知识库时，对整个知识库进行正向推理，验证正向推理的结果是否引起矛盾：若引起矛盾，则转定位模块；否则把知识库中推出的新知识加入事例库。接着从事例库中选择典型事例运行知识库，若出现下列两种情况之一，则进入定位模块，即(1) 生成某些假解（推出错误的知识）；(2) 丢失某些真解（没有推出应得的结论）。否则，直接进入调试模块。

在此模块中所用的正向推理的思想是从已知数据信息出发，正向使用规则（即让规则的前提与已知数据相匹配），求解待解的问题。

在执行模块中定义如下操作：(1) 正向推理 (inference)；(2) 验证是否有矛盾 (judge-contradiction)；(3) 选例运行检查 (run-check)。

3.3　定位模块

在知识库中，有两种情况需要定位。情况一：系统导致矛盾，要找出引起矛盾的规则；情况二：若知识库在真的实例上失败，要找出导致失败的规则。

我们使用矛盾回溯法来进行定位模块的设计。因为在 KBDT 中，假定知识库中的知识用一阶语言 L 中的 Horn 子句（规则）来加以表示，故矛盾回溯法可用于知识库的定位操作：

(1) 正向追踪 (error-solve1)：寻找导致矛盾的规则；(2) 反向追踪 (error-solve2)：寻找导致矛盾的规则；(3) 丢失真解追踪 (error-solve3)；(4) 预言者（人工界面）：用于对 L 中的基本原子进行正确与否的判断。

3.4 精化模块

在系统已定位或找到不符合要求的规则后，需要对此规则进行精化。在KBDT中，提供了两种求精方式：一是手工求精：由专家对知识进行修改；另一种是自动求精：由系统对已定位的规则进行精化。

在精化模块中，定义了如下操作：

(1) 手工求精 do-correct1)；(2) 系统求精 (do-correct2)：实例化 (instantiate)，加子目标 (add-goal)，删子目标 (delete-goal)。

3.5 调试模块

此模块用于知识库中冗余的检查和删除，规则间循环支持的检查和删除。

在调试模块中，对于前面几个模块中所加入的新知识进行如下三种操作：

(1) 如果新知识是冗余的，则去掉它；(2) 如果新知识不包含在当前知识库中，则把它加进知识库中；(3) 如果新知识比知识库中一已知的知识更一般，或更具体，或和已知知识相互矛盾，则用新知识取代知识库中的当前知识。

此外，在调试模块中还定义了循环的检查和删除操作 (do-circle-check，circle-delete).

4 结 束 语

本文论述的知识库调试工具KBDT作为知识库建造工具GKD-KBST的核心部分已在micro VAX Ⅱ上使用Prolog语言实现。实际运用结果表明，KBDT即可作为一个独立的系统用于智能系统知识库的求精，也可用于逻辑程序系统的调试，其实用面比较宽，且其关键技术具有坚实的理论基础。

参 考 文 献

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

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

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

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

[5] 杨莉等. GKD-KBST 知识库建造工具鉴定会资料. 国防科技大学，1990

Technical Principles and Design Ideas of the Knowledge Base Debugging Tool

Yang Li Hu Shouren

(Department of Computer Science)

Abstract

The knowledge base debugging tool is used to solve the problem of correctness and efficiency of knowledge base caused by the uncompleteness and noises, etc. This paper discusses the technical principles and design ideas of knowledge base debugging tool through the introduction of implementing method of knowledge base debugging tool KBDT in the knowledge base constructing tool GKD—KBST.

Key words artificial intelligence, knowledge base refinement, knowledge base debugging