中 国 科 学
1992 年 10 月　　A 辑　　第 10 期

基于项重写技术的机器发现逻辑*

孙怀民　　梁　群
(北京航空航天大学计算机科学与工程系，北京　100083)

摘　　要

机器学习与机器发现的研究需要借鉴机器定理证明技术的成果和现代科学哲学的方法论. 遵循这两个观点，本文扩展了基于项重写规则法的机器定理证明技术，使之能解决某些有关机器发现的问题. 在此基础上给出了 Lakatos 发现方法论的一个形式系统，并建立了实现 Lakatos 方法的算法.

关键词　证伪方法论、重写系统、关系变元、*BN* 合一、*PN* 合一、假说演算

一、Lakatos 的发现逻辑

1. 证伪方法论

Lakatos 在他的名著《证明与反驳——数学的发现逻辑》[1]中，通过对数学史上两个案例(围绕 Euler 猜想和 Cauchy 猜想的论战)的分析，揭示了证明与反驳在数学发现过程中的作用，发展了他的基于反驳与证明分析的证伪方法论，他称之为数学发现逻辑. 随后，他又把这种证伪方法推广到自然科学哲学，发展成科学研究的纲领方法论[2].

Lakatos 首先批判了数学史上对待反驳的简单做法，即他所谓的怪物除外法和例外除外法. 他指出，这些方法只限于修改概念以排除反例，他们忽略了反驳与证明之间的联系. Lakatos 强调，在遇到反驳时，必须对猜想的原证明过程进行逻辑分析，才能使猜想得到真正的改进[1].

为发展这种证明分析方法，Lakatos 把他所关心的反例分成两类:

(1) 全局非局部反例　这种反例反驳了猜想的结论，但没有反驳原来证明猜想时所用的引理(前提)，这才是对猜想的真正反驳. 因为原来的证明是

引理→结论.

现在反例给出了一个引理为真、后件为假的实例，所以上述关系在逻辑上不真. 因此，Lakatos 称这一类反例为逻辑反例.

(2) 局部非全局反例　这种反例只反驳了引理，而没有反驳结论. 这种反例不是对猜想的逻辑反驳，但它却揭示了原猜想的有效范围过窄. 因此，Lakatos 称它们为启发反例.

1991-10-29 收稿，1992-05-21 收修改稿
* 国家 "863" 高科技计划和国家自然科学基金资助项目

Lakatos 提出两种方法，即“一证多驳法”和“多证多驳法”来处理这二类反例. 他用了一系列启发规则来说明他的方法. 这些启发规则可以概括为如下 3 条:

启发规则 1　引理并入法 (一证多驳法): 如果遇到一个逻辑反例，就分析原证明过程中所用的引理. 选 1 条被当前反例所驳倒的适当引理，把它作为前提条件并入到猜想中，使得这个反例变成新猜想的局部兼全局反例. 在机器学习研究中所谓的 “Specialization”, 可以看成是这一启发规则的一个简化的特例.

启发规则 2　引理替换法 (多证多驳法): 如果遇到一个局部非全局反例，就设法用未被否证的引理换掉被驳倒的引理，以改进证明分析.

启发规则 3　演绎推测法(多证多驳法): 对待任何一类反例，分析导出原猜想的条件，试图用演绎推测的手段去推导出一个更为深刻的猜想.

2. 关于 Lakatos 方法的形式化

能否在计算机上实现 Lakatos 的方法呢? 这当然是一个十分诱人的前景. 为此，首先需要将 Lakatos 的方法论形式化. 要这样做，自然需要利用目前已发展得相当成熟的机器定理证明技术. 但在这样做的时候，我们必需注意到，在机器定理证明与 Lakatos 发现逻辑之间，存在如下的区别: 一阶谓词演算所关心的问题是对一个给定的前提集合 Φ 和结论 α，是否有 $\Phi\vdash\alpha$. 而 Lakatos 的证明分析却超出了一阶逻辑的范畴，它所关心的是前者的逆过程，即: 如果已知 $\Phi\nvdash\alpha$，是否存在着一个公式 P，使得 $\Phi\cup\{P\}\vdash\alpha$. 因此，这个待求的公式 P 将作为一个变元在证明分析过程中出现. 这就是二阶逻辑中所谓的关系变元.

为了解决上述问题，我们将机器定理证明技术中的重写规则加以扩展，使之能在一定程度上解决 Lakatos 的证明分析问题，并以此为手段建立了 Lakatos 方法论的形式系统. 为了检验我们的理论，我们还开发了一个基于本文方法的发现逻辑实验系统 DL.

二、假 说 演 算

本节将利用项重写技术建立一种受限的二阶谓词演算，并称之为假说演算. 假说演算所要解决的问题是: 设 Φ 是一个一致的合式谓词公式集合，α 是一个合式谓词公式，已知由 Φ 不能推得 $\alpha(\Phi\nvdash\alpha)$，要求构造一个公式 P_x，使得或者有 $\Phi\cup\{P_x\to\alpha\}\vdash\alpha$，或者有 $\Phi\cup\{P_x\}\vdash\alpha$.

这相当于要对下述两种形式的断言:

$$\text{存在着一个公式 } P_x,\ \Phi\cup\{P_x\to\alpha\}\vdash\alpha;$$

或者，

$$\text{存在着一个公式 } P_x,\ \Phi\cup\{P_x\}\vdash\alpha$$

作出关于 P_x 的构造证明，所以它是一种二阶演绎. 又因为我们规定 P_x 只能被二阶存在量词约束，所以说它是受限的.

1. 基于重写规则法的一阶演绎

基于重写规则系统的一阶演绎方法是 Hsiang[3] 给出的，他的方法的原理是将前提集合 Φ 中的谓词公式以及待证结论的否定转换成 Bool 环上的一组 Bool 表达式，以这些 Bool 表达式作为重写规则集，如果能用该规则集导出一条规则 $1\Rightarrow 0$，则有 $\Phi\vdash\alpha$.

对于前提集合及待证结论都是子句型公式的情况，文献[3]中给出了一个很简便的转换函数，其定义如下.

定义 2.1 设子句 $C = L_1 \vee L_2 \vee \cdots \vee L_k$，其中 L_i 是文字(原子谓词或原子谓词的否定)，则

$$\Psi(C) = \begin{cases} 1, & \text{如果 } C \text{ 是空子句;} \\ P + 1, & \text{如果 } C \text{ 是 } P; \\ P, & \text{如果 } C \text{ 是} \sim P; \\ \Psi(L_1) * \Psi(L_1 \vee \cdots \vee L_k) & \text{其它情况,} \end{cases}$$

其中，"+"是 Bool 环的异或运算，"*"是与运算.

Hsiang 给出了实现上述方法的算法，他称为 N 算法.限于篇幅，我们不在此作详细介绍.

2. 二阶项重写系统

定义 2.2 关系变元 P_x 是 Bool 变元.

定义 2.3 重写规则集 R 是二阶的，如果 R 中有一个(且仅有一个)规则 $l_i \Rightarrow r_i$，在 l_i 中有 Bool 变元 P_x 出现.

定义 2.4 重写规则 $N \Rightarrow \delta$ 中，如 N 是若干个一阶文字的乘积(不包含 P_x)，$\delta = 0$ 或 1，则称之为 N' 规则.

定义 2.5 (BN 合一) 如果重写规则集 R 中有两条规则 $L_1(|t, P_x|) \Rightarrow \delta$ (其中，δ 等于 0 或 1，$|t, P_x|$ 表示 L_1 中有子项 t 和关系变元 P_x 出现)和 $L_2 \Rightarrow 0$，满足

(1) $L_2 \Rightarrow 0$ 是 N' 规则；

(2) 存在两个 Bool 项 u 和 v 和一个最一般合一 θ，使得 $(ut)^\theta = (vL_2)^\theta$，则可推得一条新规则

$$u(L_1[t \leftarrow 0,\ P_x \leftarrow P_x^\theta])^\theta \Rightarrow \delta,$$

式中用 $x \leftarrow y$ 表示用项 y 替换 x.

例 2.1 规则

$$P_x(x, y)\ \text{grandpa}(x, y) + P_x(x, y) \Rightarrow 0$$

与规则

$$\text{grandpa (billy, teddy) grandpa (edward, tom)} \Rightarrow 0$$

有 BN 合一：

$$\theta = \{\text{billy}/x,\ \text{teddy}/y\},$$
$$u = \text{grandpa (edward, tom)},$$
$$v = P_x(x, y).$$

由此可得到一条新规则

$$P_x^\theta(\text{billy, teddy})\ \text{grandpa (edward, tom)} \Rightarrow 0.$$

定义 2.6 令 $\theta_1, \theta_2, \cdots, \theta_n$ 是一个 BN 合一序列，由此得到的规则形如

$$P_x^{\theta_1} \cdots P_x^{\theta_n} \Rightarrow \delta \text{ (其中，} \delta \text{ 等于 0 或 1.).}$$

如规则左端全是关系变元的乘积，则称它是 PN 规则，称由 $\theta_1, \cdots, \theta_n$ 中出现的常元所构成的集合是该 PN 规则的 Herbrand 子域，简称 H 子域.

定义 2.7 令 E 是一个表达式，θ 是一个替换 $\{t_1/x_1, \cdots, t_n/x_n\}$，则 $E^{-\theta}$ 是将 E 中所有常元 $t_i (1 \leqslant i \leqslant n)$ 的出现用变元 x_i 替换所得的结果.

定义 2.8 设 $P_x^{\theta_1} \cdots P_x^{\theta_n} = \delta$ 是一 PN 规则. 对一组基 N' 规则

$$p(\bar{t}_1) \Rightarrow \delta',$$
$$\vdots$$
$$p(\bar{t}_n) \Rightarrow \delta',$$

其中 $p(\bar{t}_1)$ 是基原子 $p_1(\bar{t}_{i1})$, …, $p_m(\bar{t}_{im})$ 的合取. 如果能按下法构造一组 θ_i':

对任一 $p_x^0 i$, 如果 $p(\bar{t}_i)$ 中的常项 t 在 θ_i 的分式 t/x 中出现,则令 $t/x \in \theta'$; 如果 t 不在 θ 中出现,则指定一新变元 u, 令 $t/u \in \theta'$. 且 θ' 满足

$$p(\bar{t}_1)^{-\theta_1'} = p(\bar{t}_2)^{-\theta_2'} = \cdots = p(\bar{t}_n)^{-\theta_n'} = p(\bar{t})^{-\theta'},$$

则称替换

$$\sigma = \{p(\bar{t})^{-\theta'}/P_x\}$$

是上述诸规则的 PN 合一. 称

$$P_x = \begin{cases} p(\bar{t})^{-\theta'}, & \text{如果 } \delta \neq \delta', \\ \sim p(\bar{t})^{-\theta'}, & \text{如果 } \delta = \delta' \end{cases}$$

是 P_x 的 PN 合一解.

引理 2.1　令 $C[P_x]$ 是一个有二阶项 P_x 出现的子句集, P 是 P_x 的 PN 合一解,则 $C[P_x \leftarrow p]$ (用 P 替换 C 中 P_x 的出现)必不可满足.

定义 2.9　二阶项 P_x 的 PN 解可递归定义如下: (1) P_x 的 PN 合一解是 P_x 的 PN 解; (2) 如果 P_1, P_2 是 P_x 的 PN 解,则 $P_1 \wedge P_2$ 是 P_x 的 PN 解.

3. 基于二阶项重写系统的假说演算算法

下面将用二阶项重写系统构造一个假说演算和算法. 为此, 我们还需要定义一些概念.

定义 2.10　一个假说演算可用五元组 $H\text{-}C\langle C_0, \alpha, E, H_i, C_R\rangle$ 表示. 其中 C_0 是有限个子句集,称为前提集,它表示我们的知识背景. α 是一个原子谓词, 称为一个问题. 它表示我们所关心的一种关系, E 是有限个形如 α 或 $\sim\alpha$ 的基文字所组成的集, 称为经验集,这里 $C_0 \cup E$ 是 C_0 的一致扩充. 它表示我们根据经验或直觉所作的判断, E 中的正文字和负文字分别称为问题 α 的正例和反例. H_i 是一个待求的子句集合,称为假说集合. 要求 H_i 能满足: E 中的所有正例都是 $C_0 \cup H_i$ 的有效推论, 且 E 中没有任一反例是 $C_0 \cup H_i$ 的有效推论.

下面我们给出假说演算的算法,在 $H\text{-}C(C_0, \alpha, E, H_i, C_R)$ 算法中, 我们用表结构表示集,用[ ]表示空集.

$H\text{-}C(C_0, \alpha, E, H_i, C_R)$: ($H_i$ 初值=[ ]).

步骤 1　在 E 中取一正例 α_e, 如果 C_0 中有含 α_e 的子句,则取 $C_1 = C_0 \cup \{\sim\alpha_e\} \cup \{P_x\}$; 否则,取 $C_1 = C_0 \cup \{\sim P_x \cup \alpha(\bar{t})\} \cup \{\sim\alpha_e\}$.

步骤 2　用 Ψ 转换函数将 C_1 转换成重写规则集 R.

步骤 3　实行 BN 合一直至求出一 PN 规则为止,记结果为 R_1.

步骤 4　对 R_1 中所有 N' 规则求出其在 PN 规则的 H 子域上的基实例. 然后利用它们对 R_1 化简,求出所有在 H 子域上的基 N' 规则,令结果为 R_2.

步骤 5　对 R_2 实行 PN 合一,求出所有的 PN 合一解,记入表 $\sum PN$ 中.

步骤 6　执行算法 $PN(C_1, \sum PN, E, [\,], \sum E)$ 去求最大 PN 解 S 和 $C_1[P_x \leftarrow S]$ 所能推出的经验正例,将这些正例记入表 $\sum E$ 中,令 $H_i' = H_i \cup \{S\}$.

步骤 7 如果 $\sum E$ 已经包含了 E 中所有正例(此时，称 PN 合一解为 PN 规则的全解)，则算法成功，令 $C_R = C_0 \cup H'$。否则，令 $C_1' = C_1[P_x \leftarrow S]$，$E' = E - \sum E$，$H_1' = H_1 \cup C[P_x \leftarrow S]$ (式中 $[P_x \leftarrow S]$ 表示用 S 代换原来 C_0 的公式中的 P_x 后的结果)，并回到步骤 1 执行 $H\text{-}C(C_0', \alpha, E', H', C_R)$。

在上述算法中调用了一个子算法 PN，其过程如下：

$$PN(C_1, \sum PN, E, \sum S, \sum E).$$

步骤 1 按定义 2.9，用 $\sum PN$ 中的 PN 合一解递归地构造一个 PN 解 S_i (如已穷尽所有的 PN 解，则直接转步骤 5)。

步骤 2 令 $C_2 = C_1[P_x \leftarrow S_i] - \{\sim\alpha_e\}$ (即把原来的否证目标 $\sim\alpha_e$ 除掉)。

步骤 3 对 E 中的每个反例 $\sim\alpha_i$，用 N 算法检查 $C_2 \cup \{\sim\alpha_i\}$ 是否能导出规则 $1 \Rightarrow 0$(即检查是否有 $C_2 \vdash \alpha_i$)。如 C_2 能推出一个反例，就放弃解 S_i 回到步骤 1 去求另一个解。否则，执行步骤 4。

步骤 4 对 $KE\text{-}\{\alpha_e\}$ 中的每个正例 α_i，用 N 算法检查是否有 $C_2 \vdash \alpha_i$，设所能推出的正例为 $\{\alpha_1, \cdots, \alpha_n\}$ $(n \geqslant 0)$，将对偶 $\langle S_1, \{\alpha_e, \alpha_1, \cdots, \alpha_n\}\rangle$ 放到 $\sum S$ 中。如果步骤 1 中已穷尽了所有的 PN 解，则执行步骤 5。否则，回归到步骤 1 去求另一个 PN 解。

步骤 5 在 $\sum S$ 的对偶中，找一个能推出最多正例的对偶 $\langle S_i, \alpha_e, \alpha_1, \cdots, \alpha_n\rangle$，令 $S = S_i$，将 $\{\alpha_e, \alpha_1, \cdots, \alpha_n\}$ 中各元素加到 $\sum E$ 中。

为了说明 $H\text{-}C$ 算法，下面举出一个简单的例子。一个更复杂的例子将在第 3 节给出。

例 2.2 设 $\alpha =$ grandpa (x, y)，C_0 由以下子句组成：

C_1. father (tom, teddy);

C_2. father (billy, tom);

C_3. father (edward, billy).

取二阶公式为

$$C_4.\ \sim P_x(x, y) \vee \text{grandpa } (x, y),$$

$$E = \{\text{grandpa (billy, teddy), grandpa (edward, tom)}\}.$$

转换成重写规则集 R 如下：

r_1. father (tom, teddy) $\Rightarrow 1$,

r_2. father (billy, tom) $\Rightarrow 1$,

r_3. father (edward, billy) $\Rightarrow 1$,

r_4. $P_x(x, y)$ grandpa $(x, y) + P_x(x, y) \Rightarrow 0$,

r_5. grandpa (billy, teddy) grandpa (edward, tom) $\Rightarrow 0$.

r_4 与 r_5 BN 合一：

$\theta_1 = \{\text{billy}/x', \text{teddy}/y'\}$ (用“ ' ”标志对变元改名)，

$u =$ grandpa (edward, tom)，$v = 1$。

得规则

r_6. $P_x(\text{billy, teddy})^{\theta_1}$ grandpa (edward, tom) $\Rightarrow 0$。

r_4 与 r_6 BN 合一：

$$\theta_2 = \{\text{edward}/x'', \text{tom}/y''\},$$

$$u = P_x,\ v = 1.$$

得 PN 规则

$$r_7.\ P_x^{\theta_1}(\text{billy, teddy})\ P_x^{\theta_2}(\text{edward, tom}) \Rightarrow 0.$$

它的 H 子域为 {billy, teddy, edward, tom}，H 子域上的基 N' 规则为 r_1，r_2，和 r_3，所构造的短语为 P^θ = father (billy, tom) father (tom, teddy)，$\theta = \theta_1 \cdot \theta_2$，$P_x^{\theta_1}$ 的 PN 合一解为 $P_x^{\theta_1}$(billy, teddy) = father (x', y'') father (y'', y'). 用 x 代 x'，z 代 y''，y 代 y'，这个解可写为 $P_x(x, y)$ = father (x, z) father (z, y). 它是规则 r_7 的全解.

因此，得到假说 father (x, z) father $(z, y) \rightarrow$ grandpa (x, y).

定理 1　如果假说演算 $H\text{-}C(C_0, \alpha, E, H, C_R)$ 求出子句集 C_R，则对 E 中所有正例 α_i，均有 $C_R \vdash \alpha_i$；对 E 中所有反例 $\sim\alpha_i$，均有 $C_R \nvdash \alpha_i$.

4. 假说演算与 Lakatos 证伪方法论

我们利用假说演算建立了一个 Lakatos 方法的形式系统，并实现了一个实验系统 DL. 在此，我们不拟详细介绍 DL 的设计，仅给出它实现 Lakatos 启发规则的原理；在第 3 节中将给出一个运用 DL 系统所做的案例实验.

引理并入法(一证多驳法)　如果对 H_i 中的某个假说 H_i

$$L_i \rightarrow \alpha \tag{1}$$

有一个逻辑反例 $L_i^\theta \rightarrow \alpha_i^\theta$，则用二阶公式

$$P_x L_i \rightarrow \alpha_i \tag{2}$$

替换 H_i，令 $H_i' = H_i[(1) \leftarrow (2)]$，$E' = E$ 中所有 α_i 的正例和反例，对 $C_0 \cup H_i'$，E'，α_i^θ 实施 $H\text{-}C$ 算法去求 P_x 的 PN 解.

引理替换法(多证多驳法)　如果对 H_i 中的某个假说

$$H_i = L_i - \alpha_i$$

有一启发反例

$$L_i^\theta \rightarrow \alpha_i^\theta,$$

就用二阶公式

$$P_x \rightarrow \alpha_i$$

去替换原来假说，通过 $H\text{-}C$ 算法求 P_x 的 PN 解.

演绎推测法(多证多驳法)　如果对假说 $L_i \rightarrow \alpha_i$ 有任何一种反驳，则用新引理和新结论去构造一个含 P_x 的新假说，然后用 $H\text{-}C$ 算法求 P_x 的解.

新引理的构造方式：(1) 用 $P_x L_i$ 替换 L_i (当对原假说既有逻辑反驳又有启发反驳时，这种方式往往是很有效的)；(2) 用 P_x 替换 L_i.

新结论的构造方式：(1) 用某些新变元替换 α_i 中的某些常量；(2) 用某些新变元去替换 α_i 中的某些变元的重复出现；(3) 用某些新变元去替换 α_i 中某些函项.

(系统以试错方式去构造新假说，直到成功或穷尽可能构造方案均失败为止.)

三、案例研究：Chuchy 猜想

这是 Lakatos[1] 讨论过的一个案例.

原始猜想　如果连续级数函数是收敛的，则它的极限函数是连续的，即如果

1) 对任意 m, $f_m(x)$ 连续;且

2) $\sum f_m(x)$ 收敛,则 $\sum f_m(x)$ 连续.

1821 年, Chuchy 给出了这个猜想的一个证明. 1822 年 Fourier 发表了他的著名的级数

$$\sin(x) - \sin(2x)/2 + \sin(3x)/3 - \cdots.$$

1826 年 Abel 注意到 Fourier 级数是 Chuchy 猜想的一个反例. 它是一个收敛的连续级数函数,但它的极限函数

$$S(x) = \begin{cases} \pi/4, & \text{如果 } x < \pi/2, \\ 0, & \text{如果 } x = \pm\pi/2, \\ \pi/4, & \text{如果 } \pi/2 < x < \pi \end{cases}$$

不连续. 因此,这是一个全局非局部反例. Abel 建议把原猜想的"级数"改成"幂级数", 这是典型的"例外除外法". 一直过了 30 年, Seidel 对 Cuchy 的证明进行了分析,才发现了绝对收敛的概念.

为了用前节的方法对这个问题进行假说演算,我们先简要地介绍 Chuchy 的证明.

Chuchy 在其证明中把 $\sum_{m=0}^{\infty} f_m(x)$ 分成两部分:

$$\sum_{m=0}^{\infty} f_m(x) = s_m(x) + r_m(x),$$

式中 $s_m(x) = \sum_{m=0}^{m'} f_m(x)$, $r_m(x) = \sum_{m=m'}^{\infty} f_m(x)$. 然后, 他利用显然的算术关系: 对任意 u, v, w, z 如果 u, v, w 均小于 z, 则 $u + v + w < 3z$,
即

$$\forall u \forall v \forall w \forall z (w < z \wedge v < z \wedge u < z \to u + v + w < 3z) \tag{3}$$

及他的 ε-δ 定义,作出了一个证明,他先由猜想的前提条件推导出两条引理:

1. 存在—δ, 对任何 b, 如果 $|b| < \delta$, 则 $|s_m(x+b) - s_m(x)| < \varepsilon$ (这是由于 $s_m(x)$ 是连续的),也即

$$\forall \varepsilon \forall x \forall m \exists \delta \forall b (|b| < \delta) \to (|s(m, x+b) - s(m, x)| < \varepsilon). \tag{4}$$

2. 存在—N, 对所有 $m \geqslant N$, $|r_m(x)| < \varepsilon$ (这是由于 $\sum f_m(x)$ 收敛)也即

$$\forall \varepsilon \forall x \exists N \forall m (m \geqslant N) \to |r(m, x)| < \varepsilon. \tag{5}$$

(在以上各个形式表达中,为了符合 ε-δ 定义,我们把 ε, δ 当作变元符使用)

Chuchy 认为,由(3)—(5)式可推得猜想的结论:

3. 对所有 b, δ, 如果 $b < \delta$, 则

$$\begin{aligned} |f(x+b) - f(x)| &= |s_m(x+b) - r_m(x+b) - s_m(x) - r_n(x)| \\ &\leqslant |s_m(x+b) - s_m(x)| + |r_x| + |r_m(x+b)| \\ &< 3\varepsilon, \end{aligned}$$

也即

$$\forall \varepsilon \forall x \forall m \exists \delta \forall b (b < \delta) \to (|s(m, x+b) - s(m,x)| + |r(m, x)|$$

$$+|r(m,\ x+b)|<3\varepsilon). \tag{6}$$

所以，Chuchy 的证明等价于由前提(3)企图去证明猜想

$$(4)\wedge(5)\rightarrow(6).$$

既然猜想遇到了逻辑反驳，根据第 4 节启发规则 1，需要把原猜想改为

$$P_x\wedge(4)\wedge(5)\rightarrow(6).$$

由演绎定理知，要证

$$\{(3)\}\cup P_x\wedge(4)\wedge(5)\rightarrow(6)$$

等价于证

$$\{(3),\ P_x,\ (4),\ (5)\}\cup\{\sim(6)\}.$$

所以，我们把{(3),(4),(5)}作为 KB，$\{P_x\}$ 作为 H_x，输入到我们的试验系统 DL 中，要求证明(6)。

机器的演算过程如下：

首先，它将 {(3)，P_x，(4)，(5)，(6)} 变换成子句集，再利用 Ψ 转换函数得到重写规则集 R：

r_1. $((w<z)(v<z)(u<z)(w+v+u<3z)+(w<z)(v<z)(u<z)\Rightarrow 0$,

r_2. $(|b|<\delta_c(\varepsilon,x,m))(|s(m,x+b)-s(m,x)|<\varepsilon)+(|b|)<\delta_c(\varepsilon,x,m)\Rightarrow 0$,

r_3. $(m\geqslant n_c](\varepsilon,\ x))(|r(m,\ x)<\varepsilon)+(m\geqslant n_c(\varepsilon,\ x))\Rightarrow 0$,

r_4. $P_x\Rightarrow 1$

r_5. $(b_c(\delta)<\delta)\Rightarrow 1$,

r_6. $(|s(m_c,x_c+b_c(\delta))-s(m_c,x_c))|+|r(m_c,x_c)|+|r(m_c,\ x_c+b_c(\delta)|<3\varepsilon_c)\Rightarrow 0$.

在以上各式中 δ_c，n_c，b_c 是 Stolem 函数符，x_c，m_c，ε_c 是 Stolem 常元。其它函数符均采用了数学上习惯的中缀表达。机器执行 BN 合一的结果如下。

r_1 与 r_6 BN 合一：

$$\theta_7=\{|s(m_c,x_c+b_c(\delta))-s(m_c,x_c)|/w,|r(m_c,x_c)|/v,|r(m_c,x_c+b_c(\delta))|/u,\varepsilon_c/z\},$$

得

r_7. $(|s(m_c,x_c+b_c(\delta))-s(m_c,x_c)|<\varepsilon_c)(|r(m_c,x_c)|<\varepsilon_c)(|r(m_c+b_c(\delta)|)<\varepsilon_c)\Rightarrow 0$.

r_2 与 r_7 BN 合一：

$$\theta_8=\{m_c/m,\ x_c/x,\ \varepsilon_c/\varepsilon,\ b_c(\delta)/b,\ \varepsilon_c/\varepsilon\},$$

得N关键对

$$\langle(|r(m_c,\ x_c)|<\varepsilon_c)(|r(m_c,\ x_c+b_c(\delta)|\varepsilon_c))(|b_c(\delta)|)<\delta),\ 0\rangle.$$

用 r_5 简化得

$$r_8.\ (|r(m_c,\ x_c)|<\varepsilon_c)(|r(m_c,\ x_c+b_c(\delta)|)<\varepsilon)\Rightarrow 0.$$

r_3 与 r_8 BN 合一：

$$\theta_9=\{m_c/m,\ x_c/x,\ \varepsilon_c/\varepsilon\},$$

得

$$r_9.\ (m_c\geqslant n_c(\varepsilon_c,\ x_c))(|r(m_c,\ x_c+b_c(\delta)|)<\varepsilon_c)\Rightarrow 0.$$

r_3 与 r_9 BN 合一：

$$\theta_{10}=\{m_c/m,\ x_c+b_c(\delta)/x,\ \varepsilon_c/\varepsilon\},$$

得

$$r_{10}\cdot (m_c \geqslant n_c(\varepsilon_c, x_c))(m_c \geqslant n_c(\varepsilon_c, x_c + b_c(\delta))) \Rightarrow 0.$$

r_4 与 r_{10} PN 合一,并按 θ_i 至 θ_{10} 恢复变元符,得 P_x 解:

$$P_x = (m \geqslant n_c(\varepsilon, x))\wedge(m \geqslant n_c(\varepsilon, x + b)). \tag{7}$$

这个解表明,对原猜想还需要增加一个前提条件,即对任何 ε, Stolem 函数 $n_c(\varepsilon, x)$ 都存在极值 $\max_x n_c(\varepsilon, z)$. 这正是 Seidel 所发现的条件,也即现在的绝对收敛概念.

在机器实际运行中,并不是仅仅给出 P_x 的一个解,它还对所有导出的N规则(r_7, r_8, r_9)施行 PN 合一,给出一组 P_x 的解. 当然,从逻辑上说,所有的解都推出结论(6),而且它们也都隐含了(7). 不过仅有解(7)直接揭示了绝对收敛的概念. 这表明,机器发现逻辑的演算结果需要由人来选择. 一个有希望的途径是,建立一个人-机合作系统,把机器的形式推理能力与人的直觉推理能力结合起来.

参 考 文 献

[1] Lakatos, I., *Proof and Refutations: The Logic of Mathematical Discovery*, Cambridge University Press, 1967.

[2] Lakatos, I., *The Methodoligy of Scientific Reseach Programmes*, Cambridge University Press, 1987.

[3] Hsiang, J., *Artifical Intelligence*, 25(1985).