计算机科学 1995Vol. 22№. 3

知识库更新的研究*)

马绍汉　陶雪红
(山东大学计算机系　济南 250100)

摘　要　This paper gives an outline of knowledge base revision and some recently presented complexity results about propositional knowledge base revision. Different methods for revising propositional knowledge base have been proposed recently by several researchers, some are formula-based methods and the others are model-based methods, but all methods are intractable in the general case. For practical application, this paper presents a revision method in special case, and gives its corresponding parallel algorithm by applying the divide-and-conquer strategy.

关键词　Propositional knowledge base, Revision, Divide-and-conquer strategy, Parallel algorithm, Binary search tree.

一、研究现状

在知识库管理中，当人们获取了新的领域知识时，就需对原有知识库进行更新。对知识库的更新，从理论上讲，主要有以下三种基本操作[4]：

1)扩充(expansion)：将一个新命题 p 直接加到原有知识库 T 中，这里新信息与原有信息之间没有矛盾。这种操作结果通常记作 T+p。

2)修改(revision)：一个与原知识库 T 不相容的新命题 p 被加入，为保证其所产生的新知识库的相容性，就得去掉原知识库中的某些命题。这种操作的结果通常表示为 T·p。

3)压缩(contraction)：当一个以前认为正确的命题 p 变得错误时，就要从知识库 T 中删去命题。这种操作的结果通常被表示为 T−p。

扩充操作容易实现，可定义为：T+p=T∪{p}；而修改和压缩操作则难于处理，因为它们涉及到删除原有某些命题，其中可有多种不同的选择和处理方式，很难在理论层次上把它们确定下来，修改和压缩这两个操作是可以互相定义的[3]。其一是：

(def ·)：T·p=(T−~p)+p

即加入 p 来修改 T，相当于先压缩去~p(p 的否定)，再扩充 p；其二是：

(def −)：T−p=T∩(T·~p)

即除去 p，可以被认为是先加入~p 来修改 T，再取与 T 的交集，其中求交运算保证了没向知识库中加入新命题。由于修改和压缩操作可以互相定义，因此，只研究其中的一种操作就够了。以下所说的更新都指修改操作。

近年来，对知识库的更新，学者们提出了许多具体的方法。其中，一类是基于公式的方法，当向知识库中加入新知识时，若发生矛盾，就删除原知识库中的一些公式，以维护其相容性。在这种更新方法下，一个公式或整个留在知识库中，或整个被抛弃。基于公式的方法主要有 Ginsberg 方法和 WIDTIO 方法等[1]；另一类是基于模型的方法，这种方法考虑知识库模型的变化，即如何从旧模型转换成更新后知识库的新模型。典型的方法有 Dalal 方法，Satoh 方法等[2]。所有这些方法都遵守修改最小化原则，即原则上不去掉不该去掉的信息，尽可能多地保留原有的知识。

每种更新方法都是针对某种具体应用领域而提出的，大多数学者认为，这些方法中没有一种可以看作是通用方法，而且现在普遍认为不存在通用的、独立于应用领域的知识库更新方法。Katsuno 和 Mendelzon 以及其他一些作者都指出[5,6,7,8]，对知识库更新的研究，虽已提出了许多方法，但在通常情况下都是难解的(intractable)。因此，我们不能期望找

*)本课题得到国家自然科学基金资助。马绍汉　教授，从事算法分析与设计、并行算法和人工智能等方面的研究工作。陶雪红　研究生，主要研究方向是知识库的更新。

到一种通常情况下知识库更新的多项式时间算法，只能寻找在某些限定条件下，特定算法的易解性。在命题逻辑中，Winslett 在文[8]中，限制 $\|p\| \leqslant K$（其中 $\|p\|$ 表示 p 的长度，K 是常量），得到用 Winslett 方法和 Forbus 方法进行知识库更新的算法，其时间复杂性为 O(Log t)，其中 t 为知识库中变元的数目。文[1]中限制 T 是 Horn 公式的集合，p 是 Horn 公式且 $\|p\| \leqslant K$（K 是常量），得到用 Satoh 方法和 Winslett 方法等求 $T \circ p$ 的算法，时间复杂性为 $O(\|T\|)$。以上这些可解的算法均是采用基于模型的更新方法，对基于公式的更新方法尚未见到有效算法，本文将讨论基于公式的更新方法，并给出一种特定条件下的并行更新算法。

二、算法设计与实现

2.1 基本概念

本文在命题逻辑的范围内讨论知识库的更新。我们用 L 表示有限命题变元集 U 上的语言，p，q 表示命题公式，T 表示知识库。知识库是命题公式的有限集。如果 $T \vdash q$，我们说公式 q 在知识库 T 上为真。每个 $x \in U$ 是个原子，原子或原子的否定叫文字，原子叫正文字，原子的否定叫负文字，文字的析取叫子句。以下将用符号"~"表示否定。

给定知识库 T 和公式 p，$T \circ p$ 表示向知识库 T 中加入新知识 p 更新后的知识库。"$\circ$"叫更新操作符，常用下标表示所采用的更新方法。

基于公式的更新方法主要有 Ginsberg 方法和 WIDTIO 方法。

定义 1[1](Ginsberg 更新方法) 令 T 是可满足的知识库，令

$$W(p,T)=\{T' \subseteq T \mid T' \nvdash \sim p, T' \subset S \subseteq T \Rightarrow S \vdash \sim p\}$$

即 W(p，T)是 T 中所有与 p 相容的极大命题公式集的集合。则

$$T \circ_G p=\{T' \cup \{p\} \mid T' \in W(p,T)\}$$

这就是说，$T \circ_G p$ 是由一些子知识库构成，每个子知识库等于 T 中与 p 相容的极大命题公式集再添加上 p。

例 $T=\{a, b, a \wedge b \Rightarrow c\}$，$p=\sim c$，则

$W(p,T)=\{\{a \wedge b \Rightarrow c, a\}, \{a \wedge b \Rightarrow c, b\}, \{a, b\}\}$，因此

$T \circ_G p=\{\{a \wedge b \Rightarrow c, a, \sim c\}, \{a \wedge b \Rightarrow c, b, \sim c\}, \{a, b, \sim c\}\}$。

一个公式 q 在 $T \circ_G p$ 上为真，当且仅当它在所有子知识库上为真。如 $a \vee b$ 在上例 $T \circ_G p$ 上为真。采用 Ginsberg 方法的缺点是更新后可能产生大量的子知识库。

定义 2[1](WIDTIO(When In Doubt Throw It Out)更新方法) 为解决 Ginsberg 方法中太量子知识库的问题，定义

$$T \circ_{wid} p=\bigcap_{w \in w(p,T)} W \cup \{p\}$$

即出现在所有与 p 相容的极大命题公式集中的公式才保留在 $T \circ_{wid} p$ 中，用这种方法更新后的子知识库只有一个。这种方法的缺点是：在最坏的情况下，T 中的所有公式均被抛弃。如上例中 $T=\{a, b, a \wedge b \Rightarrow c\}$，$p=\sim c$，则 $T \circ_{wid} p=(\sim c)$，T 中公式全被抛弃了，$a \vee b$ 在 $T \circ_{wid} p$ 上为假。

2.2 具有优先关系的更新方法

用以上提到的两种方法求 $T \circ p$，都要考虑所有与 p 相容的极大命题公式集，而这种极大命题公式集往往数目繁多，这使我们在决定公式的取舍时很困难。但在实际应用中，W(p，T)中的某些元素可能毫无意义。因此，当对具体的知识库进行更新时，就可以从实际出发，从 W(p，T)中选择符合我们要求的一个命题公式集构成 $T \circ p$。

在大多数情况下，T 中公式的重要程度并不相同，更新时可根据公式的重要程度决定其取舍。例如，某些公式可能比另外一些公式更可信，我们就希望舍弃那些不可靠的公式，保留可信度高的公式；再如，公式加入 T 的时间不尽相同，加入知识库时间越长，就越可能过时，就希望先舍弃加入 T 时间久的公式。公式的重要程度可用优先级来表示，公式越重要，优先级越高。为使问题简化，设 T 中公式优先级各不相同，可按优先级高低严格排序。根据尽量保留优先级高的公式的原则，可得到一个在具有优先关系的知识库上求 $T \circ p$ 的直观算法：

算法 1 求 $T \circ p$

输入：知识库 $T=\{f_1, f_2, \cdots, f_n\}$，其中公式按优先级由高到低排列，即公式的优先关系为 $f_1 > f_2 > \cdots > f_n$，公式 p

输出：更新后的知识库 S(即 $S=T \circ p$)

步骤：

```
S={p}
for i←1 to n do
if f_i 与 S 相容 then S=S∪{f_i} endif
endfor.
```

因为逻辑表达式可满足性问题(SAT 问题)属于 NPC 类[2]，所以判断 S 与 f_i 是否相容(即 $S \cup \{f_i\}$ 中公式是否可以同时满足)，在通常情况下是难解的，除非 P=NP，上述算法不可能在多项式时间完成。

2.3 相容性判断

在通常情况下，判断公式集的相容性是NPC问题，但在实际应用中，常常限制公式为子句形式，针对这种情况，我们用分而治之[3](divide-and-conquer)策略判断公式集的相容性。

设S是子句的集合，且S中不含重言式(同时包含某变元及其否定的子句，是重言式)，S中出现的变元的集合为$U=\{u_1,u_2,\cdots,u_m\}$，设$u\in U$，我们定义如下集合：

S_u是S中包含u的子句的集合

$S_{\sim u}$是S中包含$\sim u$的子句的集合

$S_{\times u}$是S中不含u和$\sim u$的子句的集合

S'_u是将S_u中所有u的出现删掉后的结果

$S'_{\sim u}$是将$S_{\sim u}$中所有$\sim u$的出现删掉后的结果

$S(u)$是S'_u和$S_{\times u}$的并集

$S(\sim u)$是$S'_{\sim u}$和$S_{\times u}$的并集

注意，从一个仅含u(或$\sim u$)的子句中删去u(或$\sim u$)，剩下的子句为空子句(记作□)。空子句不可满足。

例 $S=\{c,\sim a\vee b,\sim b,a\}$，则

$S_b=\{\sim a\vee b\}$　　$S_{\sim b}=\{\sim b\}$

$S_{\times b}=\{c,a\}$　　$S'_b=\{\sim a\}$

$S'_{\sim b}=\{\square\}$　　$S(b)=\{\sim a,c,a\}$

$S(\sim b)=\{\square,c,a\}$

引理[3] 子句集S不可满足(即S中公式不可同时满足)当且仅当对S中任意变元u，$S(u)$和$S(\sim u)$都不可满足。

因此，判断子句集S的相容性(即S是否可满足)，可用分而治之策略，按S中任意变元u将S划分为$S(u)$和$S(\sim u)$，分别判断其相容性，再根据引理确定S的相容性。若$S(u)$和$S(\sim u)$的相容性仍不易判定，则重复上述过程。于是不断对子句集进行划分，直到最终成为空集(空集相容)或含空子句的集合(含空子句的集合不相容)。这是一种递归过程。由于对$S(u)$和$S(\sim u)$相容性的判定是相互独立的，可并行执行。

2.4 算法描述

首先定义子句集对应的二叉搜索树。设子句集为S(S中不含重言式)，其中变元的集合$U=\{u_1,u_2,\cdots,u_m\}$，子句集S对应的二叉搜索树是一棵完全二叉树(图1)，描述了按从u_1到u_m的顺序选取变元对S进行划分的过程，树中每个结点对应一个子句集。由于S不含重言式，子句集必在二叉搜索树最底层或到达最底层之前被划为空集或含空子句的集

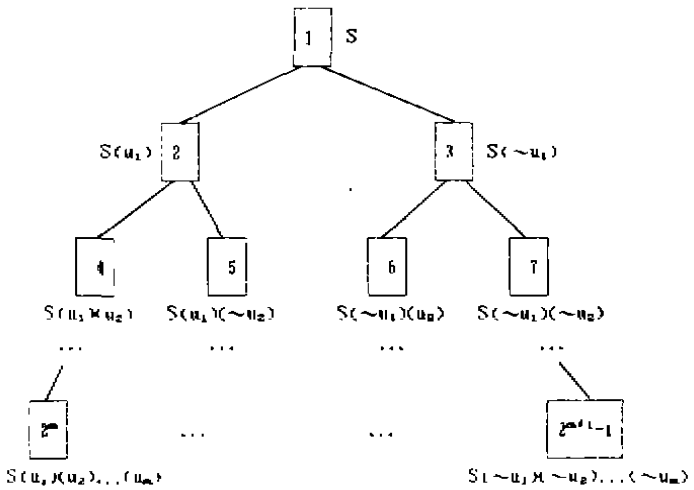

图1

合。于是，判断子句集S的相容性，可根据引理，在S对应的二叉搜索树中，自下而上，由两个子结点的相容性确定其父结点的相容性，不断重复这个过程，直至得出S的相容性。

我们将二叉搜索树中所有结点按从顶层到底层，每层从左到右的顺序编号(如图1)，根据这种编号，可定义一个长度为$2^{m+1}-1$的一维数组A来描述二叉搜索树，使A[i]为第i号结点对应的子句集，即

A[1]=S

A[2]=$S(u_1)$

A[3]=$S(\sim u_1)$

… …

A[i]=R /*R代表i号结点对应的子句集。*/

A[2*i]=$R(u_{i+1})$/*i为小于或等于$\log_2 i$的最大整数。*/

A[2*i+1]=$R(\sim u_{i+1})$

… …

根据判断子句集相容性的分而治之策略，可设计出在具有优先关系的知识库上求T·p的并行算法。算法的基本思想是：从描述公式集S={p}对应二叉搜索树的数组开始，按优先级从高到低的顺序，每加入一个公式f，计算描述新公式集S∪{f}对应二叉搜索树的数组，同时根据引理，在数组所描述的二叉搜索树中，判断新公式集的相容性，若新公式集相容，则S=S∪{f}；否则S保持不变。重复上述过程，直到所有的公式判断完为止，这时得到的公式集S即为T·p。

算法中采用的二叉搜索树与我们前面的定义稍

有不同。在算法中，若相应二叉搜索树的某结点为含空子句的集合，则令该结点为空子句□，而对该结点的后代结点不予考虑，因为它们对判断子句集的相容性没有影响。

算法 2 求 T·p

输入：知识库 $T=\{f_1,f_2,\cdots,f_n\}$，公式按优先级由高到低排列，即公式的优先关系为 $f_1>f_2>\cdots>f_n$，T 中公式和 p 均为子句形式，且均非重言式，设 $T\cup\{p\}$ 中变元的集合为 $U=\{u_1,u_2,\cdots,u_m\}$

输出：更新后的知识库 S(即 S=T·p)

步骤：

step1：S={p}

FILL(p,1) /*建立一个长度为 $2^{m+1}-1$ 的一维数组 A_0，描述{p}所对应的二叉搜索树。*/

step2：for i←1 to n do

(2.1)SIGN(1) /*即执行 $A_i=A_{i-1}$，给长度为 $2^{m+1}-1$ 的一维数组 A_i 赋初值，此时 A_i 描述 S 对应的二叉搜索树。*/

(2.2)SAT(f_i,1) /*判断 $S\cup\{f_i\}$ 的相容性，若相容则返回 Yes；否则，返回 No。并且修改 A_i 的初值，使 A_i 描述 $S\cup\{f_i\}$ 所对应的二叉搜索树。*/

(2.3)若(2.2)返回为 Yes，则 $S=S\cup\{f_i\}$；否则 SIGN(1)。 /*SIGN(1)执行 $A_i=A_{i-1}$。即 $S\cup\{f_i\}$ 不相容时，使 A_i 恢复为描述 S 对应的二叉搜索树。*/

endfor.

过程 FILL(q,j)

输入：子句 q，数字 j

步骤：

step1：如果 $j\leqslant 2^{m+1}-1$，则 $A_0[j]=\{q\}$；否则，返回。

step2：将{q}划分为{q}(u_{l+1})和{q}($\sim u_{l+1}$)，其中 l 为小于或等于 $\log_2 j$ 的最大整数。

step3：并行执行以下两个过程：

FILL({q}(u_{l+1}),2*j)，

FILL({q}($\sim u_{l+1}$),2*j+1。

过程 SAT(q,j)

输入：子句 q，数字 j

步骤：

step1：(1.1)如果 q=□或 $A_{i-1}[j]=$□，则 $A_i[j]=$□；

否则，$A_i[j]=A_{i-1}[j]\cup\{q\}$。

(1.2)检查子句集 $A_i[j]$

· 若是空子句□，则返回 No；

· 若是空集，则返回 Yes；

· 否则，做 step2。

step2：将{q}划分为{q}(u_{l+1})和{q}($\sim u_{l+1}$)，其中 l 为小于或等于 $\log_2 j$ 的最大整数。

step3：并行执行以下两个过程：

SAT({q}(u_{l+1}),2*j)，

SAT({q}($\sim u_{l+1}$),2*j+1)。

step4：检查 step3 返回的值：

· 若两个都为 No，则返回 No；

· 否则，返回 Yes。

过程 SIGN(j)

输入：数字 j

步骤：

step1：如果 $j\leqslant 2^{m+1}-1$，则 $A_i[j]=A_{i-1}[j]$；否则，返回。

step2：并行执行以下两个过程：

SIGN(2*j)，

SIGN(2*j-1)。

算法的正确性可由引理保证。

2.5 复杂性分析

过程 FILL 中，step1 需常数时间，设为 c_1，step2 需对{q}进行划分，即检查 q 中每个文字，看是否有 u_{l+1}或$\sim u_{l+1}$出现，所用时间至多为 c_2m(q 中至多 m 个不同的文字，常数 c_2 表示处理一个文字的时间)。若用 T(j)表示 FILL(q,j)所用的时间，T(k)表示 step3 中两个子过程执行时间的较大值，则有

$$T(j)\leqslant c_1+c_2m+T(k),$$

于是，可递推得

$$\begin{aligned}T(1)&\leqslant c_1+c_2m+T(j_1)\\&\leqslant c_1+c_2m+(c_1+c_2m+T(j_2))\\&\cdots\quad\cdots\\&\leqslant(c_1+c_2+\cdots)+c_2(m+m+\cdots)+T(j_k)\end{aligned}$$

(其中 $2^{m+1}-1<j_k\leqslant 2^{m+2}-1$)

因为 $T(j_k)=c_1$，且从 T(1)到 $T(j_k)$共递推了(m+1)次，所以

$$T(1)\leqslant c_1(m+1)+c_2m(m+1)+c_1$$

因此，FILL(q,1)的时间复杂性为 $O(m^2)$。

过程 SAT 中，设 step1 和 step4 的执行时间分别为常数 c_1 和 c_3，step2 所用时间至多为 c_2m(c_2 为常数)。若用 T(j)表示 SAT(q,j)所用的时间，则

$$\begin{aligned}T(1)&\leqslant c_1+c_2m+c_3+T(j_1)\\&\leqslant c_1+c_2m+c_3+(c_1+c_2m+c_3+T(j_2))\\&\cdots\quad\cdots\\&\leqslant(c_1+c_1+\cdots)+c_2(m+m+\cdots)+(c_3+c_3+\cdots)+T(j_k)\end{aligned}$$

(其中 j_k 满足：在执行 SAT(q_k,j_k)的 step1(1.2)时，子句集为空集或空子句。)

因为 $T(j_k)=c_1$，且从 T(1)到 $T(j_k)$至多递推 m 步，所以

$$T(1)\leqslant c_1m+c_2m^2+c_3m+c_1$$

因此，SAT(q,1)的时间复杂性为 $O(m^2)$。

过程 SIGN 中，设 step1 需常数时间 c_1，同理可

得

$$T(1)\leqslant c_1+T(j_1)$$
$$\leqslant c_1+(c_1+T(j_2))$$
$$\cdots\quad\cdots$$
$$\leqslant(c_1+c_1+\cdots)+T(j_s)$$
(其中 $2^{m+1}-1<j_s\leqslant 2^{m+2}-1$)
$$\leqslant c_1(m+1)+c_1$$

因此,SIGN(1)的时间复杂性为 O(m)。

通过以上对三个过程的分析,知算法 2 中 step1 的时间复杂性为 $O(m^2)$,step2 的时间复杂性为 $O(m^2n)$。因此,算法 2 的时间复杂性为 $O(m^2n)$,其中 m 为 $T\cup\{p\}$中变元的个数,n 为 T 中公式的个数。

2.6 例子

例 $T=\{f_1,f_2,f_3\}$,其中,$f_1=\sim a\vee b$,$f_2=\sim b$,$f_3=a\vee\sim b$,公式的优先关系为 $f_1>f_2>f_3$,$p=a$

设按先 a 后 b 的顺序选取变元对子句集进行划分。算法 2 执行完 step1 后便建立了数组 A_0。A_0 描述的二叉搜索树为(∅表示空集):

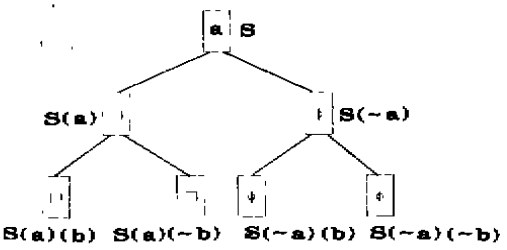

图 2

算法 2 的 step2 循环 3 次,每次循环中,(2.2)结束时 A_i 所描述的二叉搜索树及(2.3)结束时 S 的值分别为(带标记的箭头表示执行过程 SAT 时相应的返回值)图 3 所示。

因此,算法结束时,$T\cdot p=\{a,\sim a\vee b,a\vee\sim b\}$。

三、结束语

知识库是人工智能系统的核心,而知识库的更新是目前人工智能研究的一个热点。知识库的更新具有非单调性,即知识库中一些正确的知识,随着知识的增长,可能成为错误的。目前非单调推理研究已取得一些成果,将一些著名非单调推理系统,如限制理论(circumscription)、缺省逻辑(default logic)、真值维持系统(truth maintenance system)中的一些方法用于知识库的更新,可能得出更好的结果,研究新的知识库更新算法及其并行实现是我们下一步要做的工作。

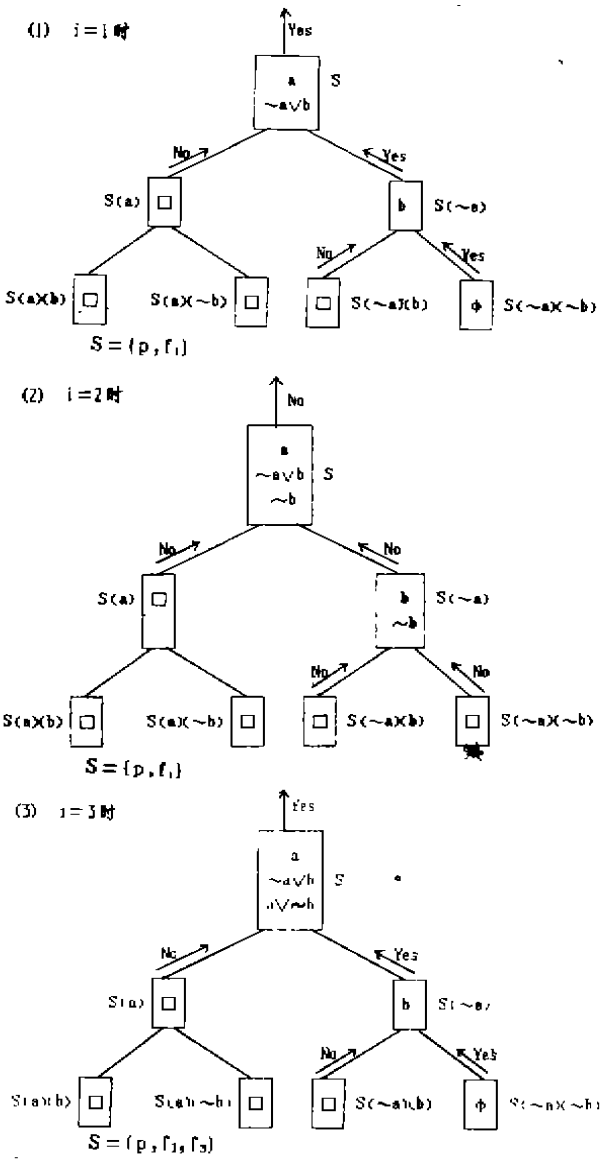

图 3

参考文献

[1] Eiter T, Gottlob G., On the complexity of propositional knowledge base revision, updates, and counterfactuals. Artificial Intellegence, 1992, 57

[2] 马绍汉,算法分析与设计. 山东大学出版社, 1992

[3] 黄智生,信念修改的理论与方法,计算机科学, 1991, 6

[4] Gärdenfors P., Epistemic importance and minimal changes of belief, Australasian J. Philos, 1984, 62

(下转第 31 页)

…,x_{n-2})为 n 阶谓词。

谓词 $P(x_1,\cdots,x_n)$为(n−1)阶谓词,当且仅当其特征函数 $C_P(x_1,\cdots,x_n)$为(n+1)阶函数。

显然有 n 阶函数⊂(n−1)阶函数,n 阶谓词⊂(n−1)阶谓词。并且对任意 n,n 阶函数或 n 阶谓词均是可计算的。进而,可以研究 n 阶函数及 n 阶谓词的演绎推理与归纳推理,在分层递增的基础上研究推理的逐级进化方式。

四、约束满足问题的处理方案

大多数约束满足问题(Constraint Satisfaction Problem,CSP)可以在代数递归逻辑下得到解,其步骤如下:

(Ⅰ)把常规下的 CSP 转化成代数递归逻辑下的 CSP 问题表示,形式如下:

$$\begin{cases} f_1(x_1,\cdots,x_n)=0 \\ f_2(x_1,\cdots,x_n)=0 \\ \quad\vdots \\ f_m(x_1,\cdots,x_n)=0 \end{cases} \tag{1}$$

其中 $f_1,\cdots,f_m$ 均为代数递归函数。

(Ⅱ)化多约束条件为单约束条件,令 $F(x_1,\cdots,x_n)=f_1(x_1,\cdots,x_n)+\cdots+f_m(x_1,\cdots,x_n)$。显然,$F(x_1,\cdots,x_n)=0$ iff $f_1(x_1,\cdots,x_n)=0\wedge\cdots\wedge f_m(x_1,\cdots,x_n)=0$。

(Ⅲ)应用 3.4 节中的算法证明 $F(x_1,\cdots,x_n)\geqslant 1$,即 $1\dot{-}F(x_1,\cdots,x_n)\equiv 0$,其中 $a\dot{-}b$ 定义如下:

$$a\dot{-}b=\begin{cases} a-b & a>b \\ 0 & a\leqslant b \end{cases}$$

若 $F(x_1,\cdots,x_n)\geqslant 1$ 对所有 $x_1,\cdots,x_n$ 成立,则(1)无解;否则(1)有解并转至(Ⅳ)

(Ⅳ)采用深度优先的分解法求解。令 $F(x_1,\cdots,x_n)$定义式为

$$\begin{cases} F(0,x_2,\cdots,x_n)=A(x_2,\cdots,x_n) & (2) \\ F(x_1+1,x_2,\cdots,x_n)\text{为 } P(x_1,\cdots,x_n, \\ \quad F(x_1,\cdots,x_n),t)=0 & (3) \end{cases}$$

的最小自然根

若 $A(x_2,\cdots,x_n)=0$ 有解,则求之并结束;否则求解 $P(x_1,\cdots,x_n,F(x_1,\cdots,x_n),0)=0$ 并结束。

五、结论与进一步工作

基于代数和递归函数理论,作者在本文中总结了近十年来的工作,形成了一套体系,并称之为代数递归逻辑。代数递归逻辑的关键是代数递归运算的形式,它是可反向分解的,这样就使代数递归逻辑区别于传统的组合逻辑,后者的反向分解不唯一,这样导致了反向分解搜索或自顶向下的搜索不可避免地带有不确定性,回溯是不可避免的,因而必然产生组合爆炸。基于代数递归的归纳推理和演绎推理完全使用反向归约或搜索策略,避免了组合爆炸,从而给出了高效的算法。

致谢 在本文工作的研究过程中,得到了洪家荣教授、刘叙华教授、石纯一教授和黄厚宽教授的指点,得到了李忠凯博士和孙吉贵博士的建议与帮助,为此作者衷心感谢。

参考文献

[1] 李爱中,刘叙华,基于组合-分解的机器发现方法,软件学报(待发表)

[2] 刘叙华,基于归结方法的自动推理,科学出版社,1994

[3] 李爱中,模型发现和智能决策支持系统工具研究,哈尔滨工业大学博士学位论文,1991

[4] Kodratoff, Y., Recent Advances in Machine Learning, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 6, No. 4, 1992

[5] Angluin, D., Inductive Inference: Theory and Methods, Computing Surveys, Vol. 15, No. 3, 1983

[6] Michalski, R. S., A Theory and Methodology of Inductive Learning, Artificial Intelligence, Vol. 20, 1983

(上接第 36 页)

[5] Katsuno H., Mendelzon A. O., On the difference between updating a knowledge base and revising it. In: Proceedings KR-91, 1991

[6] Grahne G., Updates and counterfactuals, same to [5]

[7] Nebel B., Belief revision and default reasoning: Syntax-based approaches, same to [5]

[8] Winslett M., Updating Logical Databases, Cambridge University Press, Cambridge, England, 1990

[9] Chen W. T., Liu L. L., A parallel approach for theorem proving in propositional logic. Information Sciences, 1987, 41