# 一类树型知识库的更新算法*

马绍汉 陶雪红

(山东大学计算机科学系 济南 250100)

E-mail: xuehongtao@hotmail.com

**摘要** 知识库的更新意即向知识库中添加新知识,同时为维护相容性而删除旧知识.已有的知识库更新方法在通常情况下都是难解的.该文从限制问题的结构出发,给出了一种当知识库对应的约束图为树时的多项式时间更新算法.在树型约束图中,算法通过一个自底向上的过程,得到更新后的知识库.

**关键词** 人工智能,知识库更新,算法复杂性,约束图,约束可满足问题.

**中图法分类号** TP182

知识库在人工智能研究中占有重要地位.Ginsberg等人介绍了多种知识库更新方法,其中一类是基于模型的方法,一类是基于公式的方法,但所有这些方法在通常情况下都是难解的(intractable)[1].对基于模型的方法,文献[1]中指出,当知识库是Horn公式的集合,新知识是Horn公式且长度有一常数上界时,存在多项式时间算法;对基于公式的方法,我们在文献[2~4]中给出了几类特殊条件下的多项式时间算法.

## 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$T$表示知识库,知识库是命题公式的有限集,用$p$表示要加入的新知识.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."$\circ$"称为更新操作符,常用下标表示所采用的更新方法.我们用符号"~"表示否定.

**定义 1**[1]. Ginsberg更新方法.令$T$是可满足的知识库,$p$是可满足的新知识,令

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

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

$$T\circ_{GB}p=\{T'\cup\{p\}\mid T'\in W(p,T)\}.$$

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

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

因此

$$T\circ_{GB}p=\{\{a,a\wedge b\Rightarrow c,\sim c\},\{b,a\wedge b\Rightarrow c,\sim c\},\{a,b,\sim c\}\}.$$

**定义 2.**[1] WIDTIO(when in doubt throw it out)更新方法.定义

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

用这种方法,在最坏情况下,$T$中的所有公式均被抛弃.如对上例中的$T$和$p$,则$T\circ_{WIDTIO}p=\{\sim c\}$.

本文将重点讨论WIDTIO更新方法.

## 2 算法设计与分析

问题结构与求解复杂性密切相关.直观上看,知识库中各知识之间的联系越密切,加入新知识对判断和消除

---

* 本文研究得到国家自然科学基金和国家863高科技项目基金资助.作者马绍汉,1938年生,教授,博士生导师,主要研究领域为算法复杂性,人工智能.陶雪红,女,1969年生,博士,讲师,主要研究领域为知识库更新.

本文通讯联系人:马绍汉,济南 250100,山东大学计算机科学系

本文1998-05-12收到原稿,1998-12-14收到修改稿

矛盾就越复杂. 为寻找可解算法，我们将对知识间的联系进行限制. 在 Rina Dechter 等人对约束可满足问题(constraint satisfiability problem)和真值维护(truth maintenence)的研究中，限制问题的结构为树型，得到若干有效算法[5,6]. 我们将这种思想用于知识库更新，限制知识库结构为树型，找到一个多项式时间算法.

### 2.1 约束图

我们用图来描述知识库中公式间的关系，将每个公式表示为图中的一个结点，若两公式有一个共享变量，则图中相应两结点间存在一条边，边用共享变量标记. 我们称这种图为约束图，类似于约束可满足问题中的对偶约束图(dual constraint graph)[7].

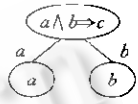

图1

例如：$T=\{a,b,a\wedge b\Rightarrow c\}$，则相应约束图如图 1 所示.

若一个知识库的约束图为若干树的集合，称为树型知识库. 在树型知识库中，任意两公式间至多有 1 个共享变量，至多有两个公式包含同一个变量. 可见，树型知识库各公式间联系较简单. 本文仅讨论知识库的约束图为树型的简单情况.

### 2.2 算法思想

**定义 3.** 删除集. 对公式集 $T$，可找到一个极大相容集 $U$(即不存在另一相容集 $S\subseteq T$ 且 $U\subset S$)，我们将 $T-U$ 称做极小删除集. 这种极小删除集可有多个，将 $T$ 的所有极小删除集的并集称做 $T$ 的删除集.

从公式集 $T$ 中去掉删除集，可使公式集相容，此时公式集中的公式即为出现在所有极大相容集中的公式. 求 $T\circ_{wid}p$ 的算法即从求删除集入手.

算法首先构造 $T\cup\{p\}$ 的约束图，约束图可能有多个连通分支，$p$ 的加入仅可能与包含 $p$ 的分支内的公式发生矛盾，对其他分支无影响，因此，只需考虑包含 $p$ 的分支. 若该分支为树，可采用我们提出的算法. 因为 $p$ 一定要属于更新后的知识库，我们将 $p$ 作为根结点进行特殊处理.

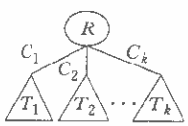

图2

考虑以 $p$ 为根的树中一子树. 设根为公式 $R$，通过共享变量 $C_1,C_2,\ldots,C_k$ 与各子树相连(如图 2 所示).

设对每个子树，当 $C_i$ 为 0 和 1 时，$T_i$ 的删除集均已知. 由于原知识库是可满足的，对任一子树 $T_i$，对 $C_i=0$ 或 $C_i=1$，至少在一种情况下 $T_i$ 可满足，即其删除集为空. 否则，若不论 $C_i$ 取 0 或 1，$T_i$ 都不满足，则说明原知识库不可满足.

对于 $C_1,C_2,\ldots,C_k$ 的一组确定值，整个树可以看做由 $R,T_1,T_2,\ldots,T_k$ 这 $K+1$ 个独立部分组成. $T_1,T_2,\ldots,T_k$ 的删除集已知，$R$ 的删除集也可知(若对于这组值来说，$R$ 为真，则删除集为空；否则为$\{R\}$). 这 $K+1$ 部分删除集的并集即为整个树在 $C_1,C_2,\ldots,C_k$ 为这组值时的删除集. 遍历 $C_1,C_2,\ldots,C_k$ 的所有值，可得若干特定值时的删除集. 求这些集合的极小集，再求并集，即得到整个树的删除集. 因此，由子树的删除集可得整个树的删除集. 算法即根据这种思想设计的.

算法在以 $p$ 为根的树中，自底向上(保证处理某结点时其所有子结点均处理完毕)，逐步求子树的删除集，这个过程一直进行到根结点，至此，得到整个树的删除集，进而得到 $T\circ_{wid}p$.

### 2.3 算法描述

算法中，约束图的每个结点保存一张真值表，其结构见表 1. 其中 $P$ 表示与父结点的共享变量(根结点无此项)；$C_1,C_2,\ldots,C_k$ 表示与 $k$ 个子结点的共享变量(叶结点无此项)；$V$ 表示公式的真假值.

表 1

| $P$ | $C_1$ | $C_2$ | … | $C_k$ | $V$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | … | 0 | 0 |
| … | … | … | … | … | … |
| 1 | 1 | 1 | … | 1 | 1 |

真值表中的每一行称做一项. 一个结点，除与父结点、子结点的共享变量外，还可能有其他变量，设为 $Y_1,Y_2,\ldots,Y_j$，这些变量均不列在真值表中. 对 $P$ 和 $C_1,C_2,\ldots,C_k$ 的一组值来说，若存在 $Y_1,Y_2,\ldots,Y_j$ 的一组值使公式为真，则真值表中 $V$ 值为 1(真)，否则为 0(假). 这说明，在共享变量一定时，即在不影响其他公式时，尽量

使该公式为真.这样,尽量多地保留了原有知识,体现了"极小修改"的原则.

每个结点中还保存一数组 $S$,$S[0]$记录 $P$(与父结点的共享变量)为 0 时以该结点为根的子树的删除集,$S[1]$记录 $P$ 为 1 时以该结点为根的子树的删除集.为表达清晰,我们用 $NODE.S$ 表示结点 $NODE$ 的数组 $S$.

下面给出采用 WIDTIO 方法进行更新的算法.

**算法**. REVISION1.

输入:非空且可满足知识库 $T=\{f_1,f_2,\ldots,f_n\}$,可满足新知识 $p$

输出:$T\circ_{wid}p$

Step1.

(1.1) 构造 $T\cup\{p\}$的约束图.

(1.2) 如果 $p$ 为图中一个孤立的结点,输出 $T\circ_{wid}p=T\cup\{p\}$,算法结束.

(1.3) 如果 $p$ 所在分支不是树,输出"此算法无法更新",算法结束.

(1.4) 在以 $p$ 为根的树中,计算各结点的真值表.

Step2.

在以 $p$ 为根的树中,自底向上(保证处理某结点时,其子结点已处理完毕),对每个结点 $NODE$,做:

(2.1) 如果 $NODE$ 为叶结点:

若 $P=0$ 时 $V=0$,则 $S[0]=\{NODE\}$,$S[1]=\varnothing$;

若 $P=1$ 时 $V=0$,则 $S[1]=\{NODE\}$,$S[0]=\varnothing$.

(2.2) 如果 $NODE$ 为中间结点(各子结点分别记作 $CHILD_1,CHILD_2,\ldots,CHILD_k$,将 $NODE$ 与 $CHILD_i$ 的共享变量记做 $C_i$):

对 $NODE$ 真值表中 $P=0$ 的每一项 $j$,设 $C_1,C_2,\ldots,C_k$ 取值分别为 $u_1,u_2,\ldots,u_k$,

令其对应集合 $D_j=CHILD_1.S[u_1]\cup CHILD_2.S[u_2]\cup\ldots\cup CHILD_k.S[u_k]$,

若 $V=0$,$D_j=D_j\cup\{NODE\}$,

求这些 $P=0$ 的项对应的集合的极小集合,再求这些极小集合的并集,即得 $NODE.S[0]$.

对 $P=1$ 的每一项进行同样的操作,得 $NODE.S[1]$.

(2.3) 如果 $NODE$ 为根结点:/* 根结点(即新知识 $p$)一定要为真,因此,只考虑 $V=1$ 的项. */

对 $V=1$ 的每一项 $j$,设 $C_1,C_2,\ldots,C_k$ 取值分别为 $u_1,u_2,\ldots,u_k$,

令其对应集合 $D_j=CHILD_1.S[u_1]\cup CHILD_2.S[u_2]\cup\ldots\cup CHILD_k.S[u_k]$,

求这些集合的极小集合,将这些极小集合的并集记做 $U$.

Step3.

输出 $T\circ_{wid}p=(T-U)\cup\{p\}$(即从 $T$ 中删除 $U$ 中公式,再加入 $p$).

在(2.2)中,真值表的每一项 $j$ 对应一集合 $D_j$,$D_j$ 是各子树及 $NODE$ 在该取值时删除集的并集.由于子树的两个删除集 $S[0]$和 $S[1]$至多 1 个非空,$NODE$ 的删除集或为空或为$\{NODE\}$,$D_j$ 实际上是选择各子树及 $NODE$ 非空删除集的并集.于是,真值表中每一项对应着对非空删除集的一种选择.因此,求集合的极小集再求并集,可简化为求这种选择的极小集及并集.我们可用 $k+1$ 维数组 $W$ 记录这种选择,$W[1]\sim W[k]$对应子树,若 $CHILD_i.S[u_i]=\varnothing$,令 $W[i]=0$,否则,$W[i]=1$;$W[k+1]$对应 $NODE$,若 $V=1$,则 $W[k+1]=0$;否则,$W[k+1]=1$.于是,数组为 1 的项说明选中子树或 $NODE$ 的非空删除集,为 0 表示没选中.例如,对 $NODE$ 真值表中 $(C_1C_2C_3V)=(1010)$的一项,按算法

$$D=CHILD_1.S[u_1]\cup CHILD_2.S[u_2]\cup CHILD_3.S[u_3]\cup\{NODE\},$$

若 $CHILD_1.S[u_1]<>\varnothing$,$CHILD_2.S[u_2]<>\varnothing$,$CHILD_3.S[u_3]=\varnothing$,则 $D$ 实际上是 $CHILD_1$,$CHILD_2$ 和 $NODE$ 非空删除集的并集.用数组记录这种选择,则 $W=(1101)$.这样,可使真值表中每一项对应一数组,我们仅需对这些数组进行相应处理,最后再由数组对应出删除集.我们将据此对算法进行简化.

我们通过对真值表进行修改,使其起到上述数组 $W$ 的作用.修改前,真值表中关于 $C_1,C_2,\ldots,C_k,V$ 的一项 $(u_1,u_2,\ldots,u_k,v)$,表示变量 $C_1,C_2,\ldots,C_k$ 值为 $u_1,u_2,\ldots,u_k$ 时,公式值为 $v$;修改后,变为该项对应的数组,记

录对非空删除集的一种选择. 另外，每结点设 $k+1$ 维数组 $W$ 记录对修改后真值表处理的最终结果. 每结点还设 $k$ 维数组 $Flag$，若 $CHILD_i.S[1]<>\varnothing$，则 $Flag[i]=1$，否则，$Flag[i]=0$. 这样，若已知选中 $CHILD_i$ 的非空删除集，可由 $Flag[i]$ 判断该非空删除集是 $S[0]$ 还是 $S[1]$.

我们将 REVISION1 的(2.2)和(2.3)进行改进，得到如下算法.

**算法**. REVISION2.

输入：非空且可满足知识库 $T=\{f_1,f_2,\ldots,f_n\}$，可满足新知识 $p$

输出：$T\circ_{wid}p$

Step1.

(1.1) 构造 $T\cup\{p\}$的约束图.

(1.2) 如果 $p$ 为图中的一个孤立的结点，输出 $T\circ_{wid}p=T\cup\{p\}$，算法结束.

(1.3) 如果 $p$ 所在分支不是树，输出"此算法无法更新"，算法结束.

(1.4) 在以 $p$ 为根的树中，计算各结点的真值表.

Step2.

在以 $p$ 为根的树中，自底向上(保证处理某结点时，其子结点已处理完毕)，对每个结点 $NODE$，做：

(2.1) 如果 $NODE$ 为叶结点：

若 $P=0$ 时 $V=0$，则 $S[0]=\{NODE\}$，$S[1]=\varnothing$；

若 $P=1$ 时 $V=0$，则 $S[1]=\{NODE\}$，$S[0]=\varnothing$.

(2.2) 如果 $NODE$ 为中间结点：

(2.2.1) /* 修改真值表 */

将 $NODE$ 真值表中各项的 $V$ 值取反(即 1 置 0，0 置 1).

对于与 $NODE$ 连接的每个子结点(各子结点分别记做 $CHILD_1,CHILD_2,\ldots,CHILD_k$，将 $NODE$ 与 $CHILD_i$ 的共享变量记做 $C_i$)，做：

如果 $CHILD_i.S[0]=CHILD_i.S[1]=\varnothing$，　/* 子树可满足，两个删除集均空 */

将 $NODE$ 真值表中各项的 $C_i$ 值全置 0，

$Flag[i]=0$；

如果只有 $CHILD_i.S[0]<>\varnothing$，　/* $C_i$ 为 0 时子树有非空删除集 */

将 $NODE$ 真值表中各项的 $C_i$ 值取反(即 1 置 0，0 置 1)，

$Flag[i]=0$；

如果只有 $CHILD_i.S[1]<>\varnothing$，　/* $C_i$ 为 1 时子树有非空删除集 */

$Flag[i]=1$.

(2.2.2) /* 求删除集 */

对 $NODE$ 中 $P=0$ 的所有项：

/* ① 类似求极小集合，② 类似求并集，③ 由数组得删除集 */

① 只要存在关于 $C_1,C_2,\ldots,C_k,V$ 的两项 $\alpha,\beta$，并且 $\alpha\wedge\beta=\alpha$，则将 $\beta$ 删除，其中 $\wedge$ 为按位与.

② 对剩余所有项按位或，结果是 $K+1$ 维数组，记做 $W$.

③ $NODE.S[0]=\varnothing$

从 $i=1$ 到 $k$，　/* $W[1]\sim W[k]$对应子树的删除集 */

若 $W[i]=1$，则 $NODE.S[0]=NODE.S[0]\cup CHILI_i.S[Flay[i]]$

若 $W[k+1]=1$　/* $W[k+1]$对应 $NODE$ 的删除集 */

$NODE.S[0]=NODE.S[0]\cup\{NODE\}$

对 $P=1$ 的所有项进行同样的操作，可得 $NODE.S[1]$.

(2.3) 如果 $NODE$ 为根结点：

(2.3.1) 同(2.2.1)

(2.3.2) 删除 $NODE$ 中 $V=1$ 的所有项，对剩余项，做：

/* 即删除真值表修改前 $V=0$ 的项，保证根结点(即新知识 $p$)为真 */

① 只要存在关于 $C_1,C_2,\ldots,C_k,V$ 的两项 $\alpha,\beta$，并且 $\alpha\wedge\beta=\alpha$，则将 $\beta$ 删除，其中 $\wedge$ 为按位与.

② 对剩余所有项按位或，结果是 $K+1$ 维数组，记做 $W$.

③ $U=\varnothing$

从 $i=1$ 到 $k$，　/* $W[1]\sim W[k]$对应子树的删除集 */

若 $W[i]-1$，则 $U=U\cup CHILD_i.S[Flag[i]]$.

Step3.

输出 $T\circ_{wid}p=(T-U)\cup\{p\}$(即从 $T$ 中删除 $U$ 中公式，再加入 $p$).

若 $T\cup\{p\}$可表示为树形结构，在以 $p$ 为根的树中，子结点的左右顺序是无关的，因此该树唯一确定. 该算法采用自底向上的过程，由叶结点开始，不断由子树的删除集得到树的删除集. 叶结点的删除集可根据公式真假直接判断；在由子树的删除集求树的删除集时，遍历了共享变量的所有取值，若树的删除集存在，则一定可找到. 因此，在以 $p$ 为根的树中，我们一定能正确找到删除集，进而得到 $T\circ_{wid}p$. 算法的正确性即由此得到保证.

### 2.4 算法分析

下面我们分析 REVISION2 的时间复杂性. 设 $T$ 中有 $n$ 个公式，$T$ 中的公式和 $p$ 的长度有常数上界 $r$.

找出两公式间的共享变量，需用 $O(r^2)$时间，(1.1)要找出任两公式间的共享变量以确定图的边，需 $O(n^2r^2)$时间；(1.2)需 $O(1)$时间；(1.3)判断是否有圈，可用深度优先搜索算法，对 $e$ 条边的图复杂性为 $O(e)$[8]. 此处任两公式间至多有 $r$ 条边，整个图至多有 $n(n+1)r/2$ 条边，所以，(1.3)用 $O(n^2r)$时间；由于公式长度有常数上界，所以计算一个结点的真值表所用时间有常数上界，则(1.4)需 $O(n)$时间. 因此，Step1 需 $O(n^2)$时间.

(2.1)需 $O(1)$时间. 由于公式长度有常数上界，所以真值表大小有常数上界，(2.2.1)主要修改真值表，需 $O(1)$时间；(2.2.2)的①和②也是对真值表的处理，用 $O(1)$时间；(2.2.2)的③不断向 $NODE.S$ 中加入公式，而 $NODE.S$ 至多可有 $n$ 个公式，所以，(2.2.2)的③至多需 $O(n)$时间. 于是(2.2)可在 $O(n)$时间内完成. 同理，(2.3)也可在 $O(n)$时间内完成. 因此，Step2 循环体用 $O(n)$时间，至多循环 $n+1$ 次共需 $O(n^2)$时间.

$T-U$ 要从 $n$ 个元素的集合中减去至多 $n$ 个元素，用 $O(n^2)$时间，再加入 $p$ 用 $O(1)$时间，因此，Step3 用 $O(n^2)$时间.

综上所述，算法 REVISION2 的时间复杂性为 $O(n^2)$. 于是，可得到如下定理.

**定理**. 对可满足知识库 $T$ 和可满足新知识 $p$，当 $T\cup\{p\}$的约束图为树型，且所有公式长度均有常数上界时，可在 $O(n^2)$时间内求得 $T\circ_{wid}p$.

## 3 结束语

本文给出了一类树型知识库的多项式时间更新算法. 该算法与知识库结构有关，适合知识库中知识间联系较松散的情况，如电子线路和基于模型的医疗诊断系统[6]. 该算法虽是一种简单情况下的算法，但它表明知识库更新的复杂性与知识库的结构密切相关，为我们探索更新问题复杂性来源，并进一步寻找适当限制消除其难解性提供了线索. 另外，在文献[7]中将非树型结构化为树型结构的技术支持下，该算法可找到更广泛的应用领域.

## 参考文献


1 Eiter T, Gottlob G. On the complexity of propositional knowledge base revision, updates, and counterfactuals. Artificial Intelligence, 1992,57(2～3):227～270

2 马绍汉，陶雪红. 知识库更新的研究. 计算机科学，1995,21(3),32～36
(Ma Shao-han, Tao Xue-hong. Research on knowledge base revision. Computer Science, 1995,21(3):32～36)

3 马绍汉，陶雪红，孙伟. 一类命题知识库的更新算法. 计算机研究与发展，1996,33(2):127～131
(Ma Shao-han, Tao Xue-hong, Sun Wei. A knowledge base revision algorithm. Computer Research and Development, 1996,33(2):127～131)

4 陶雪红，孙伟，马绍汉. 命题知识库更新的算法及其复杂性. 软件学报，1996,7(5):300～305
(Tao Xue-hong, Sun Wei, Ma Shao-han. Knowledge base revision algorithm and complexity. Journal of Software, 1996, 7(5):300～305)

5 Dechter Rina, Pearl Judea. Network-based heuristics for constraint-satisfaction problems. Artificial Intelligence, 1988, 34(1):1～38

6 Dechter Rina, Dechter Avi. Structure-driven algorithms for truth maintenance. Artificial Intelligence, 1996,82(1):1～20

7 Dechter Rina, Pearl Judea. Tree clustering for constraint networks. Artificial Intelligence, 1989,38(3):353～366

8 马绍汉. 图算法. 贵阳：贵州人民出版社，1988
(Ma Shao-han. Graph Algorithms. Guiyang: Guizhou People's Press, 1988)

# A Tree-like Knowledge Base Revision Algorithm

MA Shao-han　TAO Xue-hong

(*Department of Computer Science　Shandong University　Ji'nan*　250100)

**Abstract**　Knowledge base revision is to add new knowledge into the knowledge base, and to delete old knowledge if it is necessary for preserving consistency. The recently proposed knowledge base revision methods are all intractable in general case. By restricting the structure of the knowledge base, a polynomial revision algorithm is given in this paper when the corresponding constraint graph of the knowledge base is a tree. In the constraint tree, the authors use a bottom-up process to get the revision knowledge base.

**Key words**　Artificial intelligence, knowledge base revision, algorithmic complexity, constraint graph, constraint satisfiability problem.