使用选择函数的亚蕴涵询问求值*

姜云飞 智桂兰

(吉林大学计算机科学系 长春 130023)

摘要 本文提出了使用选择函数的亚蕴涵询问求值方法以及依据此方法建立的询问求值算法IVAL.这种方法比Bossu和Siegal的算法VAL简便,效率高.IVAL和VAL同时在SUN-4上用C-Prolog语言实现.本文给出了IVAL的理论依据,并把两种算法的执行效果作了比较.

关键词 智能数据库,亚蕴涵,选择函数.

中图法分类号 TP18

闭世界假定[1]是人们在使用一阶数据库进行日常推理时经常使用的一种约定,其主要思想是"把失败当否定"[2,3]:如果从一阶数据库推不出基本事实 A,则假定 $\sim A$ 成立.闭世界假定对数据库来说是十分重要的,有了闭世界假定,我们可以不必存储大量的反面事实,把数据库的规模减到最低限度.

设 P 是一阶数据库,q 是一阶逻辑公式,所谓询问求值就是问从 P 出发是否可以推出 q,即 $P\vdash q$ 是否成立.这个问题是数据库理论的关键.在关系数据库中,这个问题可用检索技术来解决.对于一阶数据库,因为 $P\vdash q$ 问题是不可判定的,这个问题解决起来要困难得多.在闭世界假定下的询问求值当然就更困难,我们只能希望对于一阶逻辑的某些特殊的类解决这一问题.

Clark提出了Horn数据库在闭世界假定下的询问求值算法.[2]某些相容的非Horn数据库在闭世界假定下可能会变得不相容.为解决这个问题,Minker[4]提出了广义闭世界假定的概念,Gelfond等人[4]提出了处理这类问题的算法.为使一阶数据库在闭世界假定下的询问求值技术进一步实用化,Bossu和Siegel[5]提出了亚蕴涵的概念,并根据亚蕴涵建立了询问求值算法VAL.

1 亚蕴涵,饱和归结与限制

亚蕴涵是文献[5]中为实现闭世界假定下的询问求值所引进的重要概念,是Bossu等人提出的算法VAL的理论基础.下面我们简要介绍Bossu等人的研究工作.

设 F 是一个解释集合,P,Q 是公式集,我们用 $P\models_F Q$ 表示 P 在 F 中的每一个模都满足 Q,用 $dis(P)$ 表示 P 的判别模集.

定义1.1. 设 P,Q 是公式集,如果存在 $dis(P)$ 的一个闭弱集 F 使 $P\models_F Q$,则称 P 亚蕴涵 Q,记为 $P]=Q$.

关于判别模集与闭弱集的概念请参见文献[5].

设 P 是一阶数据库,q 是询问,则在亚蕴涵意义下的询问求值即是问是否有 $P]=q$,具体地说,如果用VAL表示求值结果,则有

$$VAL(q,p)=\begin{cases}1 & \text{当 } P]=q \text{ 时,此时对询问回答 YES}\\ 0 & \text{当 } P]=\sim q \text{ 时,此时对询问回答 NO}\\ 1/2 & \text{当 } P]\neq q \text{ 时,此时对询问回答 Indefinite}\end{cases}$$

定义1.2. 设 C 是子句集,如果 C 中任意两个子句的首文字归结式或者是恒真公式,或者被 C 中的某一字句所包含,则 C 称作是对首文字归结是饱和的.

给定满足某些条件的子句集 S(例如 g-子句集[6]),可以指定文字的某种排序,然后不断地把首子文字归结式加入该集合中,同时删去恒真公式和被其它子句所包含的公式,在有限步内最终得到饱和集,这个过程叫饱和归结.因为饱

---

* 本文研究得到国家自然科学基金和国家教委博士点基金资助.作者姜云飞,1945年生,教授,博士导师,主要研究领域为人工智能,非单调推理,智能诊断与智能规划.智桂兰,女,1969年生,讲师,主要研究领域为人工智能,自动推理.现工作单位是大连铁道学院计算机系.

本文通讯联系人:姜云飞,广州 510275,中山大学计算机软件所

本文1996-10-20收到原稿,1997-01-16收到修改稿

和归结要考虑所有的归结式，同时还要检验这些归结式是否为恒真，是否为其它子句所包含，所以是一件相当费时的工作.

定义 1.3. 设 P 是 g-有限子句集，Q 是 g-有限公式集，Q 在 P 中的特征公式是形式为 $f=c''\vee\sim g'$ 的公式，满足：

(1) $q'=c'\vee\sim g'$ 是 $ins[Q,P]$ 中的元素.

(2) c'' 是 $P\cup\{c'\}$ 的特征子句，但不是 P 的特征子句. Q 对 P 的特征公式集记为 $Carf[Q,P]$.

特征公式集在询问求值中的关键作用由下面的求值定理给出.

定理 1.4. $VAL[Q,P]=VAL[Carf[Q,P],P]$.

根据定理 1.4，要得到询问求值的结果，只要计算出 $Carf[Q,P]$ 即可. 文献[5]提出了使用饱和归结求特征子句集的方法，根据定义 1.3，每求一次 $Carf[Q,P]$，至少需要作两次饱和归结，一次用于产生 P 的特征子句集，另一次用于产生 $P\cup\{c'\}$ 的特征子句集，然后把二者加以比较，得到 $Carf[Q,P]$. 如果 $Carf[Q,P]$ 非空，我们还要递归地计算 $\sim Carf[Q,P]$ 对 P 的特征字句集，还要再次使用饱和归结，复杂的问题要递归地使用多次，这是造成 VAL 效率低的重要原因.

并行限制(Parallel Circumscription)是 McCarthy[6] 的限制的推广. Liftchitz[7,8] 深入研究了并行限制，获得了一些重要结果. 文献[9]对 Liftchitz 的工作做了很好的总结，本文采用文献[9]的记法.

2　IVAL 的理论基础

本节证明几个定理，这些定理是下一节的算法 IVAL 的基础.

定理 2.1. 设 $\triangle$ 是全称定量的一阶数据库，q 是一阶逻辑公式，则

$$CIRC(\triangle;p)\wedge UNA\models q \text{ 当且仅当 } \triangle]=q$$

其中 UNA 代表唯一名假定(Unique Name Assumption)(证明略).

定理 2.1 告诉我们，亚单调询问求值问题可以转换为限制数据库下的普通询问求值问题. 根据文献[9]的结果，如果一阶数据库关于它的所有谓词是可排序的，则可直接引用限制结果，于是 $CIRC(\triangle;p)$ 变成了普通的一阶数据库，从而亚单调询问求值问题变成了普通的数据库下的普通询问求值. 为了把一个数据库分裂成关于谓词可排序的数据库，我们还需要证明如下定理.

定理 2.2. 相容的有限 g-Horn 子句集 P 有唯一的极小模(证明详见附录).

在下面的定理中，我们用 $mdis(P)$ 表示数据库的极小判别模集.

定理 2.3. 设 P 是相容的有限 g-子句集，H 是 P 中的 Horn 子句构成的集合，C 是 P 的特征子句集，则

$$mdis(P)=mdis(H\cup C)\text{(证明详见附录)}$$

定理 2.3 把对于数据库 P 的询问求值转换为对于数据库 $H\cup C$ 的询问求值. 如果特征子句集 C 全部由单元字句组成，则 $H\cup C$ 为 Horn 数据库，如果 Horn 数据库不含循环前提(例如，$p\rightarrow g,q\rightarrow r,r\rightarrow p$)，一般都可以改写成谓词可排序的，如果 C 特征子句集不是全由单元子句组成，则需要进一步把 C 分裂，我们利用下面的选择过程实现这种分裂.

定义 2.4. 设 C 是正基子句集，S 和 S' 是从 C 的每一个子句中选出一个正单文字得到单元子句集合 $S(C)$ 和 $S'(C)$ 的函数，如果 $S(C)$ 是选出的集合的极小集合(即对一切函数 S'，若 $S'(C)\subseteq S(C)$，则 $S'(C)=S(C)$)，则称 S 为选择函数，$S(C)$ 为选择集合.

例 2.5：设子句集 C 由下列子句组成

$$C=\{le(jean,eng)\ le(jean,ger),le(jean,lat)\ le(jean,gre),le(jean,eng)\ le(jean,lat)\}$$

则 $S_1=\{le(jean,eng),le(jean,lat)\}$ 是 C 的一个选择集合. 为简便起见，我们省略了子句中的析取号"∨"，以后也经常使用这种省略.

设 P 是有限 g-子句集，则 P 的特征子句集 C 是有限的，于是 C 的所有选择集合为有限个，设这些选择集合为 $S_1,\ldots,S_n$，则可以把 $H\cup C$ 分裂成 $H\cup S_1,H\cup S_2,\ldots,H\cup S_n$，每一个 $H\cup S_i$ 都是 Horn 子句集.

初看起来，在构造选择集合时似乎会产生严重的组合问题，增加计算的复杂性，但实际上因为特征子句集的各子句中有许多相同的特征文字，所有的能构造出的选择集合并不多. 例如，从例 2.5 的子句集里每一个子句取一个元素有 $2^3=8$ 种不同的选取方法，但 C 的选择集合只有 3 个，除 S_1 外，另外两个是：

$$S_2=\{le(jean,eng),le(jean,gre)\}\qquad S_3=\{le(jean,ger),le(jean,lat)\}.$$

注意，$\{le(jean,ger),le(jean,lat),le(jean,eng)\}$ 等集合不是最小集合，因而不是选择集合.

下面的定理为数据库的分裂提供了理论依据.

定理 2.6. 设 P 是有限 g-子句集,H 是 P 的 Horn 子句集,C 是 P 的特征子句集,$S_1,\ldots,S_n$ 是 C 的选择集合,则

$$mdis(H\cup C)=\bigcup mdis(H\cup S_i)$$(证明略)

3 IVAL 算法和例

根据上一节的定理,我们建立了下述 IVAL 算法,设 P 是有限 g-子句集,q 是对数据库 P 的询问,结果为 ANS.

Procedure $IVAL(q,P)$

1. $H\leftarrow Horn(P)$,$C\leftarrow char(P)$; Horn 函数计算出 P 的 Horn 子集 H,Char 函数计算出 P 的特征子句集.
2. if $C=$Nil, then ANS$\leftarrow HVAL(q,P)$过程结束; HVAL 是 IVAL 的子过程,对 Horn 数据库进行询问求值.
3. SC$\leftarrow Sele(C)$; 函数 Sele 计算出特征子句集 C 的所有选择集合,存储在表 SC 中.
4. $S_1\leftarrow First$(SC); S_1 是 SC 中第一个选择集合.
5. SC$\leftarrow Tail$(SC); 从 SC 中删去第一个元素.
6. ANS$\leftarrow HVAL(Q,H\cup S_1)$; 设置 ANS 的初值
7. LOOP: if SC=Nil, 回答 ANS; 如果 SC 为空,回答 ANS
8. $SM\leftarrow First$(SC); 从表 SC 中取出一个选择集
9. SC$\leftarrow Tail$(SC); 从 SC 中删去一个元素
10. ANS$_1\leftarrow HVAL(q,H\cup SM)$; 对 Horn 数据库 $H\cup SM$ 调用子过程 HVAL 进行询问求值.
11. if ANS=Fail or ANS$_1$=Fail then ANS$\leftarrow$Fail,SC$\leftarrow$Nil
else if ANS$_1$ ANS then ANS$\leftarrow$1/2, SC$\leftarrow$Nil;
如果 ANS 或 ANS$_1$ 等于 Fail,置答案为 Fail,立即结束过程,如果 ANS$\neq$ANS$_1$,置答案为 1/2,也立即结束过程.
12. Goto LOOP.

HVAL 是 IVAL 的子过程,对 Horn 数据库进行询问求值.

Procedure $HVAL(q,H)$

1. $SH\leftarrow Simp(H)$; 函数 Simp 对 H 进行化简,删除 H 中的被包含子句
2. $OH\leftarrow Order$(SH); 函数 Order 对 SH 进行排序,排序的结果是 OH,如果 SH 是不可排序的,OH 为 Fail.
3. if $OH\leftarrow$Fail then 回答 Fail,过程结束; 如果 OH 为 Fail,则过程回答 Fail 然后结束.
4. $CH\leftarrow Circ(OH)$; 函数 Circ 是计算并行限制结果的函数,其结果 CH 为通常的一阶数据库.
5. 回答 $Deduce(q,CH)$; Deduce 为在通常一阶数据库下对 q 进行询问求值的过程,询问的结果是 1(表示真)和 0(表示假)

例:[6]设数据库 P 由下列子句构成.

(1) $le(x,eng)le(x,ger)\sim s(x)$,
(2) $le(x,lat)le(x,gre)\sim 3rd(x)$
$\sim s(x)$,
(3) $\sim le(x,gre)\sim le(x,ger)$,
(4) $l(x)\sim lm(x),l(x)\sim ld(x)$,
(5) $s(toto),s(jean),s(aul)$,
(6) $3rd(toto),3rd(jean)$,
(7) $lm(eng),lm(ger)$,
(8) $ld(lat),ld(gre)$,
(9) $le(paul,eng),le(paul,ger)$,
$\sim le(toto,eng)$

询问 $q_1=x(3rd(x)\rightarrow le(x,lat))$.

利用饱和归结,求出 P 的特征子句集 C 为

(10) $le(jean,eng)le(jean,ger)$,
(11) $le(toto,ger)$,
(12) $le(paul,eng)$,
(13) $le(jean,lat)le(jean,gre)$,
(14) $le(paul,ger)$,
(15) $le(toto,lat)$,
(16) $le(jean,eng)le(jean,lat)$.

设 C 中的单元子句组成的集合为 CU,则 CU 由单元子句(11)~(15)组成.根据选择集合的定义知 C 的选择集合为 $CU\cup S_1,CU\cup S_2,CU\cup S_3$.其中 S_1,S_2,S_3 如例 2.5 所示.于是算法 IVAL 把询问求值 $IVAL(q,P)$分裂成 $H_i=H\cup CU\cup S_i(i=1,2,3)$三个 Horn 子句集上的询问求值问题,其中 H 是 P 的 Horn 子句,由子句(3)~(9)组成.接下去 IVAL调用 $HVAL(q,H_i)(i=1,2,3)$分别对 q 求值.我们先考虑 $HVAL(q,H_1)$,根据 HVAL 过程的步骤,先把 H_1 化简,删除被包含子句(12),(14).设 $A_1(x)=le(x,eng)$,$A_2(x)=le(x,ger)$,$A_3(x)=le(x,lat)$,$A_4(x)=le(x,gre)$,利用排序函数 Order,可把 H_1 改写成 OH_1,由以下公式组成,

(1)' $N(x)=(\sim A_4(x)\vee\sim A_2(x))\wedge\sim A_1(toto)$ 由(3),(9)
(2)' $x=toto\vee x=jean\vee x=paul\rightarrow s(x)$ 由(5)
(3)' $x=toto\vee x=jean\rightarrow 3rd(x)$ 由(6)
(4)' $x=eng\vee x=ger\rightarrow lm(x)$ 由(7)
(5)' $x=lat\vee x=gre\rightarrow ld(x)$ 由(8)

(6)′ $lm(x) \vee ld(x) \rightarrow l(x)$　　由(4)　　(8)′ $x=paul \vee x=toto \rightarrow A_2(x)$　　由(9),(11)

(7)′ $x=paul \vee x=jean \rightarrow A_1(x)$　　由(9),S_1　　(9)′ $x=jean \vee x=toto \rightarrow A_3(x)$　　由(15),S_1

对 OH_1 施行限制运算 $Circ$ 时，可以直接利用关于限制的定理，得到 OH_1 的限制结果 CH_1，把(1)′～(9)′式中的蕴涵号改成等价号即得 CH_1，CH_1 为一阶公式. 询问 $q=\forall\, x(3rd(x) \rightarrow le(x,lat)) = \forall\, x(3rd(x) \rightarrow A_3(x))$，根据限制结果，使 $3rd(x)$ 为真的客体只有 *toto* 和 *jean*，而这两个客体都使 $A_3(x)$ 为真，所以 $HVAL(q_1,H_1)=1$，利用类似的求值过程，可得 $HVAL(q_1,H_2)=0$，所以 $IVAL(q_1,P)=1/2$.

设 $q_2=le(toto,gre)$，采用上述过程，可求出 $IVAL(q_2,P)=0$.

同样，设 $q_1=\forall\, x(lm(x) \rightarrow le(paul,x))$，可计算出 $IVAL(q_3,P)=1$，设 $q_4=\forall\, x(S(x) \rightarrow le(x,eng))$，可计算出 $IVAL(q_4,P)=0$.

总之，利用IVAL计算的询问求值结果与利用VAL计算的结果是完全相同的.

4　VAL与IVAL的比较

因为IVAL只使用一次饱和归结，所以询问求值的速度要快得多，但询问求值的结果是完全相同的. 我们在SUN-4工作站上用C-Prolog同时实现了VAL和IVAL，下面的表1列出了VAL和IVAL在执行时间上的对比，表中 P_1,q_1,q_3,q_4 如上例所述，P_2，q_5 和 q_6 分别为

$$P_2=\{w(a),w(b),s(a),s(b),s(c),abs(a) \vee abs(c),abs(b) \vee abs(c)\}$$

$$q_5=\forall\, x(s(x) \rightarrow abs(x)),\quad q_6=\forall\, x(abs(x) \rightarrow s(x))$$

表1　VAL和IVAL在执行时间上的对比　　单位：s

算法 \ 询问 时间	$val(q_1,P_1)$	$val(q_3,P_1)$	$val(q_4,P_1)$	$vla(a_5,P_2)$	$vla(q_6,P_2)$
VAL	1.55	0.68	2.05	0.0455	0.1783
IVAL	0.3667	0.3833	0.3333	0.0333	0.0357
结果	1/2	1	0	0	1

从执行结果也可以看出饱和归结对效率的影响. 当使用VAL计算询问求值时，$val(q_3,P_1)$ 只使用两次饱和归结，所以用的时间较少，$val(q_1,P_1)$ 和 $val(q_4,P_1)$ 用了3次饱和归结，所以用的时间就比较多，而 $val(q_4,P_1)$ 在完成第2次饱和归结后产生两个特征子句，第3次饱和归结的过程比较复杂，用的时间就更长一些，但IVAL只使用一次饱和归结，所以对同一数据库的不同询问求值的运行时间差不多.

5　结　论

本文研究了亚蕴涵与限制的关系，提出了基于限制的亚蕴涵询问求值算法IVAL. 在IVAL中我们使用了选择函数的概念，减少了VAL中费时较多的饱和归结的次数，所以效率比较高. 本文描述了在SUN-4上执行VAL和的IVAL过程，并把两种算法的执行效果作了比较.

参考文献

1 Reiter R. On closed world data base. In: Gallaire H, Minker J Eds. Logic and Data Base. New York: Plennum Press, 1978. 55～76

2 Clark K L. Negation as failure. In: Gallaire H, Minker J Eds. Logic and Data Base. New York: Plennum Press, 1978. 292～322

3 Gelgond M, Przymsinska. Negation as failure: careful closure procedure. Artificial Intelligence, 1986,30:273～287

4 Minker J. On indefinite database and the closed world assumption. In: Proceedings of the 6th Conference on Automated Deduction, Lecture Note in Computer Science. 138 Berlin, Springer, 1982. 292～308

5 Bossu G, Seigel P. Saturation nonmonotonic reasoning and closed world assumption. Artificial Intelligence, 1985,25:13～63

6 Mcarthy J. Circumscription——a form of nonmonotonic reasoning. Artificial Intelligence, 1980,13:27～39

7 Liftchitz V. Computing circumscription. In: Proceedings of the 9th International Joint Conference on Artificial Intelligence. Los Angeles, CA, 1985. 121～127

8 Lifschitz V. On the atisfiability of circumscription. Artificial Intelligence, 1986,28:17～27

9 Genesereth M R, Nilsson N J. Logic foundation of artificial intelligence. Morgan Kaufmann, Publishers, Inc. 1989. 115～157

附录:几个定理的证明

定理 2.2 的证明:设解释集合 $I=\{l_i|l_i$ 是正基文字且 $P=l_i\}$,因 P 是有限的基子句集,所以 I 是有限集合,设 I 有 n 个元素.以下证明 I 是 P 的模.若 I 不是 P 的模,则 P 有子句 C_0 在解释 I 下为假,因 P 是 Horn 子句集,C_0 具有 $l_0 \vee \sim l_1' \vee \sim l_2' \vee \ldots\ldots \vee \sim l_m'$ 的形式,C_0 中也可以不含正文字 l_0,此时 C_0 全由负文字构成.

因为 C_0 在 I 下为假,所以 $I[l_0]=0, I[l_i'] \quad 1\ (i=1,\ldots,m)$,所以 l_i' 是解释集合 I 中的元素,即 $P \models l_i'$,根据归结方法的完备性知由 P 出发利用归结方法可得出正单元子句 l_i',在 P 中使用归结方法可得如图 1 所示的归结过程.

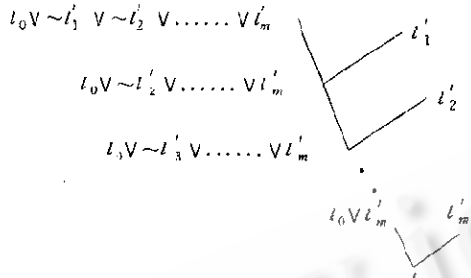

图1

如果 C_0 中不含 l_0,说明从 P 出发可推出空子句,与 P 的相容性矛盾.如果 C_0 中含有正文字 l_0,则 $P \models l_0$,因此 $l_0 \in I, I[l_0]=1$,与 C_0 在 I 下为假矛盾,所以 I 是 P 的模.

因为 $P \models l_i (i=1,2,\ldots,n)$,所以 $P \models l_1 \wedge l_2 \wedge \ldots \wedge l_n$.这说明 P 的任何模都使 l_i 为真,所以 I 是 P 的极小模. □

定理 2.3 的证明:对出现在 C 中的特征文字个数 n 使用数学归纳法.

当 $n=0$ 时,P 的特征子句集为空集,所以对 P 中的任何 Horn 非子句的正文字 l,$P \not\models l$,所以 P 的最小判别模由 P 的 Horn 子句集 H 决定,根据定理 2.2,H 有唯一的极小模,所以 $mdis(P)=mdis(H \cup C)=mdis(H)$.

假设当 C 中的特征文字个数小于 n 时定理成立,以下证明当 C 中的特征文字个数等于 n 时定理成立.

设 P 的特征句子集 C 的特征文字个数为 n,以下分两种情况考虑:

(1) 如果特征子句集的子句全由正单文字子句组成,任取 $I_1 \in mdis(H \cup C)$,则 I_1 具有如下形式:$I_1=I \vee \{l_i|l_i \in C\}$,其中 I 是 H 的一个极小判别模,l_i 是 C 中的正单文字子句,以下证明 I_1 是 P 的极小判别模.若 I_1 不是 P 的极小判别模,则有 P 的模 $J_1=J \cup \{l_j|l_j$ 是 C 中的某些元素$\}$,满足 $J_1 < I_1 (J_1 \leqslant I_1, J_1 \neq I_1)$,于是存在某一正文字 $l \in I_1$ 使 $J[l]=0$,若 $l \{l_i|l_i \in C\}$,则因 J 是 H 的模,与 I 是 H 的极小模矛盾,若 $l \in \{l_i|l_i \in C\}$,则因存在 P 的模 J_1 使 l 为假,与 l 是特征子句矛盾.于是 $mdis(H \cup C) mdis(P)$.

任取 P 的极小判别模 I_2,则因 I_2 是 P 的模,可知 I_2 满足 H,I_2 满足 C,并且 $\{l_i|l_i \in C\} I_2$,又因 I_2 是极小模,所以 I_2 是 $H \cup C$ 的极小判别模,即 $mdis(P) mdis(H \cup C)$.

综上所述,$mdis(H \cup C)=mdis(P)$.

(2) P 的特征子句集 C 不是全由正单元子句组成,则 C 中包含一正的非单元子句 $l_1 \vee l_2 \vee \ldots\ldots \vee l_k$.

设 $C(P)$ 表示子句集 P 的特征子句集,则因 $P \cup \{\sim l_i\}$ 的特征子句集相当于在 P 的特征子句集中删去 l_i 的所有出现,所以 $P \cup \{\sim l_i\}$ 的特征子句集中出现的特征文字的个数为 $n-1$,根据归纳假设知

$$mdis(P \cup \{\sim l_i\}) = mdis(H \cup C(P \cup \{\sim l_i\}))$$

于是

$$mdis(P) = \bigcup_i mdis(P \cup \{\sim l_i\}) = \bigcup_i mdis(H \cup C(P \cup \{\sim l_i\})) = mdis(H \cup C)$$

□

Query Evaluation for Sub-Implication Using Selection Function

JIANG Yun-fei ZHI Gui-lan

(*Department of Computer Science Jilin University Changchun* 130023)

Abstract A method for query evaluation for sub-implication using selection function is presented. An algorithm IVAL is developed based on the method. IVAL is more simple and efficient than the algorithm VAL presented by Bossu and Siegel. Both VAL and IVAL are implemented on SUN-4 in C-Prolog, and the efficiency of the two algorithms is compared.

Key words Intelligent database, sub-implication, selection function.

Class number TP18