# DATALOG 程序的组及其应用*

王云明 施伯乐

(复旦大学计算机科学系 上海 200433)

**摘要** 为了提高DATALOG程序自底向上计值的效率,本文提出了把DATALOG规则分成若干个组,对这些组可以给出一个拓扑序列,而对同一个组内的各个规则可以给出一个激发序列,以此来有效地控制DATALOG程序自底向上计值时对规则的调度,从而提高计算效率.这一技术可以应用于传统的各种自底向上计算方法,本文给出了半朴质(Semi-Naive)算法和良基模型(Well-Founded)的交替不动点算法(Alternating FixPoint)的改进.改进后的算法比原算法在时间和空间效率上都有较大的提高.

**关键词** DATALOG程序,增广依赖图,组,拓扑序列,激发序列.

**中图法分类号** TP311.13

对非递归的DATALOG程序计值时,我们把各个谓词根据它们的依赖关系排成拓扑序列,逐一计算各个谓词.这样,在计算某谓词时,它所依赖的所有谓词都已计算完成,从而每条规则只需被计算一次.但是对递归的DATALOG程序,不论是朴质算法(Naive)还是半朴质(Semi-Naive)算法[1],都不考虑规则的次序,反复地计算各条规则直到稳定,这显然会影响计算效率.我们期望能象非递归程序一样,对递归程序的规则也给出一个合理的次序再进行计算,使得对某条规则进行计算时,它所依赖的谓词都已最大程度地完成了计算,以提高计算效率.如例1.

例1:设有DATALOG程序如图1.

从直观上,我们可以按如下次序进行计算:

1. 计算$r1$;
2. 计算$r3$;
3. 用半朴质算法反复依次计算$r5$、$r2$、$r4$;
4. 计算$r6$;

$r1$ $q(X,Y):-e_1(X,Y).$
$r2$ $q(X,Y):-p(X,Y,Z).$
$r3$ $p(X,Y,Z):-e_2(X,Y,Z).$
$r4$ $p(X,Y,Z):-r(X,Y),q(Y,Z).$
$r5$ $r(Y,Z):-p(X,Y,Z).$
$r6$ $s(X,Y):-q(X,Z),r(Z,W),e_3(W,Y).$

图1 DATALOG程序$P_1$

注意在第3步中,我们先做规则$r5$、$r2$再做$r4$,这比先做$r4$再做$r5$、$r2$具有更高的效率.并且在第3步结束时,谓词$q$和$r$都已经完成了计算,因此规则$r6$只被计算一次.而用

* 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.作者王云明,1972年生,硕士生,主要研究领域为知识库,数据库.施伯乐,1935年生,教授,博士导师,主要研究领域为数据库,知识库,面向对象数据库.

本文通讯联系人:王云明,上海200433,复旦大学计算机科学系

本文1996-11-12收到修改稿

半朴质算法计算时，每一次只要$q(X,Z)$或$r(Z,W)$中增加若干元组就要激发$r6$的计算，并且会反复地做联结$\Delta q(X,Z)\bowtie r(Z,W)\bowtie e_4(W,Y)$或$q(X,Z)\bowtie\Delta r(Z,W)\bowtie e_1(W,Y)$.□

为找出一个良好的计算规则次序，并减少激发规则次数，我们提出了组的概念. 它把DATALOG程序的规则分成若干个组，对这些组可给出一个拓扑序列，对同一个组内的各个规则也可给出激发序列. 这一排序技术可和传统的各种自底向上的计算方法结合起来，能普遍提高计算效率. 组的应用并不仅在于算法的优化，还可望在其它理论研究中得到应用.

## 1 组的定义和计算方法

### 1.1 组的定义

**定义** 1.1. 增广依赖图

设有DATALOG程序$P$，它的$IDB$谓词集合为$I=\{p_i\mid 1\leqslant i\leqslant n\}$，规则集合为$R=\{r_j\mid 1\leqslant j\leqslant m\}$，记$V=I\cup R$，程序$P$的增广依赖图$ADG(P)$是一有向图$\langle V,E+,E-\rangle$，其中正边集$E+$和负边集$E-$分别为：

$E+=\{(r_i,p_j)\mid p_j$是规则$r_i$的头部谓词$\}\cup\{(p_j,r_i)\mid p_j$是规则$r_i$的正子目标谓词$\}$

$E-=\{(p_j,r_i)\mid p_j$是规则$r_i$的负子目标谓词$\}$

在不区分$E+$和$E-$时，记$E=E+\cup E-$，增广依赖图简记作$\langle V,E\rangle$.

**定义** 1.2. 有向图的组及拓扑序列

设有向图$G=\langle V,E\rangle$，$V_i$、$V_j$是两个顶点，若它们同时出现在某个环中，则称$V_i$和$V_j$同组；显然同组关系是等价关系，它把$G$中的顶点分成若干个组. 若存在一条有向边从组$g_i$中的一个顶点到组$g_j$中的某个顶点，则称组$g_i$优先于$g_j$，可以证明：组之间的优先关系是不会构成环的，因此这些组根据优先关系可以有一个拓扑序列.[2]

**定义** 1.3. 程序的组及拓扑序列

设DATALOG程序$P$的增广依赖图为$ADG(P)$，$\langle g_1,g_2,\ldots,g_n\rangle$为$ADG(P)$的组的一个拓扑序列，在$\langle g_1,g_2,\ldots,g_n\rangle$中去掉只包含谓词结点的组，并在余下的组中去掉谓词结点，得到$\langle g_1',g_2',\ldots,g_m'\rangle$；显然程序$P$的规则集$R$是各个$g_i'$的不交并，称$g_i'$为DATALOG程序$P$的组，$\langle g_1',g_2',\ldots,g_m'\rangle$为组的一个拓扑序列. 并记$P_k$为程序$\bigcup_{1\leqslant i\leqslant k}g_i'$.

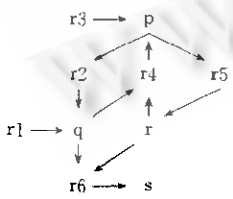


图2 $P_1$的增广依赖图

例2：对于例1中的程序$P_1$，它的增广依赖图$ADG(P_1)$如图2所示. $ADG(P_1)$的组及其它的一个拓扑序列为$\langle\{r1\},\{r3\},\{p,r5,r,r2,q,r4\},\{r6\},\{s\}\rangle$. 程序$P_1$的组及相应的拓扑序列为$\langle\{r1\},\{r3\},\{r5,r2,r4\},\{r6\}\rangle$.

**定义** 1.4. 激发序列

设DATALOG程序$P$的增广依赖图$ADG(P)=\langle V,E\rangle$，若$(V_i,V_j)\in E$，则称$V_j$为$V_i$的上级，称$V_i$为$V_j$的下级. 把$ADG(P)$中的每一个顶点看作对象，它接受下级发给它的消息并做一定的操作，具体的协议如下：

1. 初始激发：$ADG(P)$中所有没有入边的规则结点打印自己，并向上级发送一消息.
2. 规则对象的激发：若规则结点已收到它所有下级发来的消息，则打印自己，并向上级发送一消息.

3. 谓词对象的激发:若谓词结点收到了某个下级发来的消息,则它处于待激发状态.当没有规则结点可以激发时,在所有待激发的谓词结点中选取一个收到消息最多的结点,向它的所有上级发送一个消息.

4. 附加激发:若没有结点可以激发且 $ADG(P)$ 中还有结点未被激发,则在所有未激发的谓词结点中选取在组的拓扑序列中出现最早的一个结点,向它的所有上级发送一个消息.

5. 每个结点只被激发一次(即激发之后不再活动).

根据这一协议,$ADG(P)$ 将打印出一个规则的序列,把某个组 $g_i$ 中的各个规则按照这一序列中出现的次序排列起来,称作 $g_i$ 中规则的激发序列.

例 3:对于例 2 中的 $ADG(P)$,它打印的一个可能的规则序列为〈$r1,r3,r5,r2,r6,r4$〉,组{$r5,r2,r4$}相应的激发序列为〈$r5,r2,r4$〉.

### 1.2 组的计算方法

为了保证组的概念得以广泛的应用,必须要有一个高效的算法来计算组及其拓扑序列.本文给出了计算有向图 $G$ 的组及其拓扑序列的算法,根据它可以方便地得到 DATALOG 程序 $P$ 的组及其拓扑序列.算法 1.1 先把 $G$ 中所有的顶点单独成组,并让组的优先关系为空,在对 $G$ 的深度优先遍历过程中逐步合并各组,并且构造优先关系.最后根据优先关系给出组的拓扑序列.

**算法 1.1.** 计算有向图的组及其拓扑序列

输入:有向图 $G$

输出:$G$ 的组及其拓扑序列

方法:

1. 对 $G$ 的所有顶点单独成组,并让组的优先关系为空
2. 找一个顶点,设为 $n_0$
3. 从 $n_0$ 出发用深度优先方法遍历,每经历一条有向边时删去该边,每访问到一个结点就把它放入栈顶.假设当前栈中的结点为 $n_0,n_1,\ldots,n_k$,最近访问的结点为 $n_k$,若 $n_k$ 与 $n_i(0\leqslant i\leqslant k-1)$ 在同一个组中,则合并 $n_i,n_{i+1},\ldots,n_{k-1}$ 所在的组为一个大组,取消大组内部的优先关系,否则,令 $n_{k-1}$ 所在的组优先于 $n_k$ 所在的组
4. 若还有结点未被访问过,不妨仍设为 $n_0$,返回 3
5. 根据优先关系算出组的拓扑序列
6. 输出

算法 1.1 的正确性证明略,它的时间复杂性为 $O(n^2)$,其中 $n$ 为图 $G$ 的顶点数.

## 2 自底向上算法的改进

### 2.1 半朴质算法(SN 算法)的改进

对于不带否定的 DATALOG 程序,用半朴质算法 SN(semi-naive)[1]来计算它的最小不动点是目前认为比较高效的算法.本节将给出半朴质算法的改进.限于篇幅,我们不再叙述原来的算法,而直接给出改进后的半朴质算法(ISN 算法).

**算法 2.1.** ISN 算法

输入:不带否定的 DATALOG 程序 $P$

输出:$P$ 的最小模型

方法:

1. 计算 $P$ 的组及其拓扑序列
2. 根据定义 1.4 算出各组内规则的激发序列
3. 按照组的拓扑序列依次对每一个组中的规则进行计值,若组中规则不递归,则用关系代数求解,否则用半朴质算法反复对各规则按激发序列计值直到稳定
4. 输出

算法比较:显而易见,对于非递归的程序,ISN 算法与原来的算法是一致的.

对于递归程序,ISN 算法只存放某一个组内的谓词的增量,而 SN 算法则同时存放了程序中所有谓词的增量,因此,ISN 算法比 SN 算法需要更少的空间代价.

另外,ISN 算法在计算组 $g_k$ 时,$P_{k-1}$的最小模型已经计算完成,从而为 $g_k$ 的计算一次性提供了最大的关系,所以有:(1) ISN 算法对规则的计算次数少于 SN 算法;(2) ISN 算法在联结上所花的代价少于 SN 算法.

例 4:设有 DATALOG 程序 $P_2$ 如下:

$P_2$ $p_i(X,Y):-e_i(X,Y).$ $i=1,2,\ldots,n$

$p_i(X,Y):-e_i(X,Z),p_i(Z,Y).$ $i=1,2,\ldots,n$

(*) $q(X,Y):-p_1(X,Z_1),p_2(Z_1,Z_2),\ldots,p_n(Z_{n-1},Y).$

我们用元组的数目来衡量关系的大小.假设每个 EDB 关系 $e_i(X,Y)$的大小为 $e$ 个元组,$p_i$ 谓词的每一次迭代增加 $\Delta$ 个元组,共要迭代 $N$ 次,即最终 $p_i$ 关系的大小为 $e+\Delta N$.再假设在每一次做关系联结时的有效率为 $\alpha$,即一个大小为 $N_1$ 的关系和一个大小为 $N_2$ 的关系做联结时,将得到一个大小为 $N_1 * N_2 * \alpha$ 的关系.定义联结工作的代价为二个关系大小的乘积.则有:(1) 程序 $P_2$ 用 ISN 算法求解时,规则(*)只计算一次,而用 SN 算法则计算 $N+1$ 次;(2) 程序 $P_2$ 用 ISN 算法求解时,联结代价为 $ISN_n=(e+\Delta N)^n\alpha^{n-2}$,

而用 $SN$ 算法时联结的总代价为

$$SN_n=[e^n+n\Delta((e+\Delta)^{n-1}+(e+2\Delta)^{n-1}+\ldots+(e+N\Delta)^{n-1})]\alpha^{n-2}$$

可以证明,对所有的 $n\geqslant 2$ 有 $ISN_n<SN_n$,且 $SN_n-ISN_n$ 随 $n$ 指数增长.特别地,

当 $n=2$ 时,$SN_2-ISN_2=\Delta^2N$

当 $n=3$ 时,$SN_3-ISN_3=3e\Delta^2N\alpha+(3N^2+N)\Delta^3\alpha/2$

当 $n=4$ 时,$SN_4-ISN_4=[6e^2\Delta^2N+6\Delta^3N^2e+2\Delta^3eN+\Delta^4N^2+2\Delta^4N^3]\alpha^2$

由此可见,二者之差随着 EDB 关系大小、迭代次数和每次迭代的增量而急剧增加,因此,对大型数据库及多子目标的程序进行计算时,用 ISN 算法的改进效果是非常明显的.

本例中只是对递归程序的一个"基本块"进行分析的.在实际应用中的程序往往是逐层累积的"基本块"组成的,这样,ISN 算法的优越性将得到更好的体现.

### 2.2 良基模型 AFP 算法的改进

对于带否定的 DATALOG 程序,良基模型(Well-Founded Model)[3]和断系模型(Tie-Breaking Model)[4]是被广泛接受的语义,它们的计算方法非常相似,都是先构造程序的基底图,然后逐步删去可断定为真或为假的基原子结点,最后得到所要求的模型.这也是一种自底向上的计算方法,我们可以用组的技术依次求解 $P_k$ 的良基模型.在求 $P_k$ 的良基模型时可以利用 $P_{k-1}$的良基模型和组 $g_i$ 的基底图来计算,特别地,当组 $g_i$ 中的规则不递归或不带否定时还可以用关系代数来求解 $P_k$ 的良基模型.这样在时间和空间上都具有更高的效率.限于篇幅,本文不再赘述这种计算方法的改进.

文献[5]提出了用交替不动点算法 AFP(alternating fix point)来计算良基模型,它也是一种自底向上的计算方法,下面将用组的技术来改进 AFP 算法.

#### 2.2.1 良基模型的定义

设有 DATALOG 程序 $P$,把规则中的各个变量都用常量替换后称作为基规则.把基原子和基原子的非统称为基文字.设 $I$ 是基文字的集合,记$\sim I=\{\sim l\mid l\in I\}$;记 $T_P(I)$是基原

子的集合,一个基原子 $a$ 属于 $T_P(I)$ 当且仅当存在一条以 $a$ 为头的基规则,它的每一个子目标都在 $I$ 中;记 $U(I)$ 是基原子的集合,称它是一个非基集,如果:对于 $U(I)$ 中的每一个基原子,任何一条以它为头的基规则都满足:(1) 存在某个子目标属于 $\sim I$;或者(2) 存在某个正子目标属于 $U(I)$.

把最大的非基集记为 $U_P(I)$. 又记 $W_P(I)=T_P(I)\cup\sim U_P(I)$,可以证明 $W_P(I)$ 是一个单调函数,因此它有一个最小不动点,把这一最小不动点称作程序 $P$ 的良基模型.

2.2.2　交替不动点算法(AFP 算法)

设 $I$ 是基原子的集合,把 DATALOG 程序 $P$ 中的否定谓词看作是一个新的谓词,它们的关系为 $\sim(H_P-I)$,其中 $H_P$ 是程序 $P$ 的 Herbrand 基. 这样,$P$ 就被看作不带否定的程序,它的最小模型记为 $STABLE(I,P)$.[6]可以证明 $STABLE^2(I,P)=STABLE(STABLE(I,P),P)$ 是单调递增的,故而 $STABLE^2$ 函数具有最小不动点. AFP 算法就是用它的最小不动点来求 $P$ 的良基模型.[5]

**算法 2.2.** AFP 算法

输入:带否定的 DATALOG 程序 $P$
输出:$P$ 的良基模型
方法:1. $M0=EDB$ 基原子集
2. $M1=STABLE^2(M0,P)$
3. WHILE $M0\neq M1$ {$M0=M1$;$M1=STABLE^2(M0,P)$}
4. $M1=STABLE(M0,P)$
5. $P$ 的良基模型为 $M0\cup\sim(H_P-M1)$;输出

2.2.3　IAFP 算法

可以有两种方法来改进 AFP 算法,分别称为 IAFP1 和 IAFP2.

IAFP1 **算法.** 前面指出 $STABLE(I,P)$ 是把 $P$ 看作不带否定的程序来计算的,因此可以用 ISN 算法来做 $STABLE(I,P)$,其余和 AFP 算法一样.

IAFP2 **算法.**

输入:带否定的 DATALOG 程序 $P$
输出:$P$ 的良基模型
方法:1. 计算 $P$ 的组及其拓扑序列;$M0=EDB$ 基原子集
2. 按照组的拓扑序列依次对每一个组 $g_i$ 做
a. 若组中规则不递归或不带否定,则:
以 $M0$ 作为已知关系,用关系代数求解 $g_i$ 中的规则得到新的 $M0$
b. 否则:
$M1=STABLE^2(M0,g_i)$
WHILE $M0\neq M1$ {$M0=M1$;$M1=STABLE^2(M0,g_i)$}
3. $M1=STABLE(M0,P)$ /* 用 ISN 算法 */
4. $P$ 的良基模型为 $M0\cup\sim(H_P-M1)$;输出

2.2.4　算法比较

由于 ISN 算法比 SN 算法效率高,算法 IAFP1 用 ISN 来求 STABLE,并且多次调用 STABLE,因此优化效果得到了充分的体现.

IAFP2 在时间效率上优于 AFP,这体现在:(1) 在语句 a 中用关系代数求解,因此这些规则只被计算一次,而 AFP 算法则在每做一次 STABLE 时就要计算这些规则;(2) IAFP2 算法在计算 $g_i$ 时 $STABLE^2(I,P_{i-1})$ 的最小不动点已计算完成,因此 IAFP2 中计算 $STABLE^2(I,g_i)$ 的最小不动点时的迭代次数必少于 AFP 的迭代次数.

## 3 总 结

本文提出了组的概念,定义了组的拓扑序列以及组内规则的激发序列,利用它们可以有效地控制DATALOG程序自底向上计值时对规则的调度,提高计算效率.算法1.1说明计算组的效率是高的,并且这一技术可以和传统的各种自底向上计算方法结合起来,因此具有较广的应用.本文提出了改进的半朴质(Semi-Naive)算法和改进的交替不动点(Alternating FixPoint)算法.改进后的算法比原来的算法在时间和空间效率上都有较大的提高.另外,组的应用并不仅仅在于对算法的优化,还可以用来推广模块分层程序(Modularly Stratified Program)[7]或在其它的理论研究中得到应用.

## 参考文献


1 Ullman J D. Principles of database and knowledge base system. Rockville: Computer Science Press, 1989.

2 施伯乐·蔡子经等.数据结构.上海:复旦大学出版社,1988.

3 Gelder Van, Ross K A *et al*. Unfounded sets and well-founded semantics for general logic programs. In: Proc. of ACM Symp. on Principles on Database System, 1988. 620~650.

4 Diego San Papadimitriou C. Tie-breaking semantics and structural totality. In: Proc. of ACM Symp. on Principles on Database System, 1992. 16~22.

5 Van Gelder A. The alternating fix point of logic programs with negation. In: Proc. of ACM Symp. on Principles on Database System, 1989. 1~10.

6 Gelfond M, Lifschitz V. The stable model semantics for logic programming. In: Proc. of ACM Symp. on Principles on Database System, 1988. 1070~1080.

7 Ross K A. Modularly stratification and magic sets for datalog programs with negation. In: Proc. of ACM Symp. on Principles on Database System, 1990. 161~171.


# THE GROUP OF DATALOG PROGRAMS AND ITS APPLICATIONS


WANG Yunming SHI Baile

(*Department of Computer Science Fudan University Shanghai* 200433)



**Abstract** In order to gain more efficiency when calculating a Datalog program bottom up, a suggestion is proposed to divide the set of Datalog rules into the several groups. A reasonable sequence is provided for those groups, and so for rules in the same group as well. Consequently, according to those sequences scheduling rules will result in more efficiency when calculating a Datalog program bottom-up. This ordering technique can be applied to any traditional bottom-up algorithms, hence the wide application of group. The authors put forward some improved versions of certain traditional algorithms, and they are more efficient than the previous versions in space and time. It is also pointed out that the conception of group has more applications other than improving bottom-up calculations.



**Key words** Datalog program, augmented dependency graph, group, topological sequence, triggered sequence.

**Class number** TP311.13