广义查询的计数算法*

范 明 李连友

(郑州大学计算机科学系,郑州 450052)

摘要 计数算法是最著名的SL递归处理算法之一.对于一类称作计数线性的递归,它具有良好的性能.然而,计数算法要求查询的约束集为单值的,因此很难用作子目标的处理策略.本文提供一种新的广义查询计数算法,它能处理任意的多值约束集.从而本文提供的算法不仅能够用于查询的求值,而且也能用于子目标的处理.算法的正确性和变换后的规则的有效性也在本文简略讨论.

关键词 递归查询,线性递归,广义查询,计数算法.

计数算法是最有效的线性递归查询求值算法之一,其基本思想可以追溯到Henschen和Naqvi的文章[1].Bancilhon等首次使用了计数方法一词,并以同代人规则为例给出了计数算法[2].Ullman定义了计数线性规则,并给出了计数算法的完整描述[3].尽管计数算法只能用于单线性(single linear,简记为SL)递归的特定子类,由于SL是实践中最常见的递归类型之一和计数算法的有效性,计数算法受到到了广泛的关注,并被斯坦福大学的NAIL!系统、MCC的LDL系统用作查询优化处理的主要策略之一.

拓广计数算法,使之可以处理有环数据方面已做了大量工作.Han和Henschen[4]提出了一种层次环合并计数算法,对于同代人规则(下同),其时间复杂度为$O(e^2)$(其中e为边数).Sacca和Zaniolo[5]将计数算法与魔集相结合,提出了魔集计数算法.首先运行计数算法,并同时进行环检测,当检测到环时切换到魔集.对于无环数据,其时间复杂度为O(ne)(其中n为顶点数);对于有环数据,其时间复杂度为$O(e^2)$.Haddad和Naughton[6]使用超图,给出了可以处理有环数据的计数算法,其时间复杂度为O(ne).

试图将计数算法拓广到多线性(multi-linear)递归的尝试都不太成功.正如Ullman所指出的那样,除非规则是可交换的,否则其效率不如魔集[3].

其实,计数算法还有一个限制:查询的初始约束集必须是单值的.拓广计数算法,使之可以处理广义查询是有意义的.首先,具有相同约束模式的查询可以一并处理,从而可以减少对EDB关系的访问次数,降低I/O花费.其次,可以使得计数算法不仅可以用作查询处理,而且也可以用作子目标处理的策略.有些系统,如MCC的LDL系统就是为每个子目标选

* 本文1992-02-09收到,1992-07-24定稿

本文受河南省自然科学基金、河南省教委自然科学基金资助.作者范明,46岁,副教授,主要研究领域为演绎数据库查询优化,程序变换.李连友,58岁,教授,主要研究领域为软件理论,软件方法.

本文通讯联系人:范明,郑州450052,郑州大学计算机科学系

择处理策略.这方面迄今尚无已发表的结果.

范明曾在文献[7]中将右线性、左线性和左－右线性递归算法拓广到具有多值约束查询.其基本思想是将每个约束元组与一个唯一的整数对应起来,并在魔谓词和回答谓词中增加一个整型变元,用于记录它们的元组是由约束的哪个约束元组导出的.类似的思想也可以用来将计数算法拓广到广义查询.

本文的主要工作是拓广计数算法,使之可以处理广义查询,并讨论拓广后的算法的正确性和变换后的规则的有效性.

1　广义查询的计数变换算法

关于约束模式、计数线性等概念见文献[3].设给定的计数线性逻辑程序由一组涉及IDB谓词 p 的规则,和一个查询目标 p^c 组成.和文献[3]一样,我们假定 p 的前 n 个变元是被约束的,后 m 个变元是自由的,其唯一的递归规则经适当的子目标调整之后取如下形式:

$$p(s_1,\cdots,s_n,t_1,\cdots,t_m):-G_1,\cdots,G_k,p(u_1,\cdots,u_n,v_1,\cdots,v_m),H_1,\cdots,H_l \quad (1)$$

这里,$s_1,\cdots,s_n$ 和 $u_1,\cdots,u_n$ 分别是出现在头部和递归子目标 p 的被约束变元中的项;类似地,t_i 和 v_i 占据自由位置.$G_1,\cdots,G_k$ 是向上子目标,而 $H_1,\cdots,H_l$ 是向下子目标.

算法 1. 广义查询的计数变换

对于任意满足定义的计数线性逻辑程序,设查询目标 p^c 的约束集为$\{(x_{11},\cdots,x_{1n}),\cdots,(x_{k1},\cdots,x_{kn})\}$.我们可以按以下步骤构造一组新规则:

(1)创建一个IDB谓词 u_p,称为 p 的"向上"谓词.并创建一个EDB谓词 b_u_p,称为向上谓词的约束谓词,其对应的关系为$\{(1,x_{11},\cdots,x_{1n}),\cdots,(k,x_{k1},\cdots,x_{kn})\}$.然后产生基本规则

$$u\_p(J,X_1,\cdots,X_n,0):-b\_u\_p(J,X_1,\cdots,X_n) \quad (2)$$

(2)由(1)式产生 u_p 的递归规则

$$u\_p(J,u_1,\cdots,u_n,I):-G_1,\cdots,G_k,u\_p(J,s_1,\cdots,s_n,I-1).$$

(3)创建一个IDB谓词 d_p,称为 p 的"向下"谓词.对于每个基本(非递归)规则 $p(w_1,\cdots,w_{n+m}):-E_1,\cdots,E_j$,产生"向下"谓词 d_p 的初始化规则

$$d\_p(J,w_{n+1},\cdots,w_{n+m},I):-u\_p(J,w_1,\cdots,w_n,I),E_1,\cdots,E_j.$$

(4)由(1)式产生 d_p 的递归规则

$$d\_p(J,t_1,\cdots,t_m,I-1):-d\ \ p(J,v_1,\cdots,v_m,I),H_1,\cdots,H_l.$$

(5)用下面规则产生查询的回答

$$p(X_1,\cdots,X_n,Y_1,\cdots,Y_m):-b\_u\_p(J,X_1,\cdots,X_n),d\_p(J,Y_1,\cdots,Y_m,0)$$

算法的输出是至此产生的规则的集合.

为得到查询的回答,只需要自底向上地(例如,用 *Semi－Naive* 算法)求新规则的不动点.现在我们以同代人规则为例,说明以上变换算法的应用.

例 1:设 *par* 和 *person* 是EDB谓词,其中,$par(x,y)$表示 y 是 x 的双亲之一,而EDB谓词 *person* 登记着数据库中出现的每一个人.又设 $sg(x,y)$表示 x 和 y 是同代人,则IDB谓词 sg 可由以下规则定义:

r_1: $sg(X,X)$:− $person(X)$.

r_2: $sg(X,Y)$:− $par(X,Xp),sg(Xp,Yp),par(Y,Yp)$.

设 $W=\{x_1,x_2,\cdots,x_k\}$为人的集合,为找出 W 中每个人的同代人,其查询形如 sg^{bf}(即,sg 的第一个变元是被约束的,而第二个是自由的).对于约束模式 bf,以上规则是计数线性的,根据算法 1 我们有以下改写后的规则:

$u\_sg(J,X,0)$:− $b\_u\_sg(J,X)$.

$u\_sg(J,Xp,I)$:− $par(X,Xp),u\_sg(J,X,I-1)$.

$d\_sg(J,X,I)$:− $u\_sg(J,X,I),person(X)$.

$d\_sg(J,Y,I-1)$:− $par(Y,Yp),d\_sg(J,Yp,I)$.

$sg(X,Y)$:− $b\_u\_g(J,X),d\_sg(J,Y,0)$.

其中,b_u_sg 是 EDB 谓词,其对应的关系为$\{(1,x_1),(2,x_2),\cdots,(k,x_k)\}$.

2 变换算法的正确性

非形式地,本文算法的技巧是:把查询的 k 个约束元组与$\{1,\cdots,k\}$中的整数建立一个 1−1对应关系,在文献[3]的计数变换算法的向上谓词 u_p,和向下谓词 d_p 中增加一个变元 J,它在$\{1,\cdots,k\}$上取值,用于记录 u_p 和 d_p 的元组是由查询的哪个约束元组导出的.借助于这种技术,本文的算法允许查询目标 p^a 的约束集含有任意多个约束元组.当 p^a 的约束集为单值$(x_1,\cdots,x_n)$时,我们可以取消 EDB 谓词 b_u_p,并取消 IDB 谓词 u_p、d_p 中的变元 J,用 $u_p(x_1,\cdots,x_n,0)$取代(2)式,再将第Ⅴ组中唯一规则头部谓词 p 的前 n 个变量 $X_1,\cdots,X_n$ 替换成查询常量 $x_1,\cdots,x_n$,便得到文献[3]的计数算法.

注意,变换后的规则并不一般地与原规则等价.我们说变换算法是正确的,如果变换后的规则能够正确地产生查询的回答.在下面的讨论中,我们称计数算法产生的规则为计数规则,本文算法产生的规则为广义计数规则,并且采用 *Semi−Naive* 算法自底向上地计算变换后的规则.为证明本文算法的正确性,我们需要两个引理:

引理 1. 对于 $1\leqslant j\leqslant k$ 和 $i\geqslant 0$,$u_p(j,c_1,\cdots,c_n,i)$可由广义计数规则推出,当且仅当以$(x_{j1},\cdots,x_{jn})$为初始约束时,$u_p(c_1,\cdots,c_n,i)$可被计数规则推出.

该引理的证明与引理 2 的证明类似,本文从略.我们将详细证明引理 2.

引理 2. 对于 $1\leqslant j\leqslant k$ 和 $i\geqslant 0$,$d_p(j,c_1,\cdots,c_n,i)$可由广义计数规则推出,当且仅当以$(x_{j1},\cdots,x_{jn})$为初始约束时,$d_p(c_1,\cdots,c_n,i)$可被计数规则推出.

证明:充分性.设以$(x_{j1},\cdots,x_{jn})$为初始约束时,$\mu=d_p(c_1,\cdots,c_n,i)$在第 K 轮由计数规则推出.对 $K(\geqslant 1)$进行归纳.

基始:$K=1$.在迭代开始之前,计数规则产生的向下谓词 d_p 的关系中不含元组.μ 必然是由形如

$$r_1: d\_p(w_{n+1},\cdots,w_{n+m},I) :- u\_p(w_1,\cdots,w_n,I),E_1,\cdots,E_j.$$

的初始化规则(不妨就记作 r_1),使用 u_p 的某个元组导出的.而在迭代开始之前 u_p 只含唯一元组 $u\_p(x_{j1},\cdots,x_{jn},0)$,因此 μ 必然是由 r_1 使用 $u_p(x_{j1},\cdots,x_{jn},0)$导出的,并且 μ 形如 $d\_p(c_1,\cdots,c_n,0)$.由本文算法的 EDB 谓词 b_u_p 的构造立明 $u_p(j,x_{j1},\cdots,x_{jn},0)$在

第一轮由基本规则(2)导出.而对应于规则 r_1,广义计数规则中有规则

$$r_1'\colon d\_p(J,w_{n+1},\cdots,w_{n+m},I)\colon -u\_p(J,w_1,\cdots,w_n,I),E_1,\cdots,E_j.$$

从而 $d_p(j,c_1,\cdots,c_n,0)$将由 r_1'使用 $u_p(j,x_{j1},\cdots,x_{jn},0)$导出.

归纳:设 μ 在第 $K\geqslant 2$ 轮由计数规则推出,则 μ 或者由 d_p 的初始化规则,或者由 d_p 的递归规则导出.我们分别加以讨论.

(1)μ 由 d_p 的初始化规则导出.则 μ 必然是由形如 r_1 的初始化规则(不妨就记作 r_1),使用 u_p 的某个在前一轮导出的元组(不妨记作 $\nu=u\_p(b_1,\cdots,b_n,i)$)导出的.由引理 1,$\nu'=u\_p(j,b_1,\cdots,b_n,i)$)由广义计数规则导出.于是 $\mu'=d_p(j,c_1,\cdots,c_n,i)$将由 r_1'使用 ν'导出.

(2)μ 由 d_p 的递归规则导出.则 μ 必然是由递归规则

$$r_2\colon d\_p(t_1,\cdots,t_m,I-1)\colon -d\_p(v_1,\cdots,v_m,I),H_1,\cdots,H_l.$$

使用 d_p 的某个在前一轮导出的元组导出的.由 r_2,该元组形如 $d\_p(a_1,\cdots,a_n,i+1)$.由归纳假设,$d_p(j,a_1,\cdots,a_n,i+1)$由广义计数规则导出.而对应于规则 r_2,广义计数规则中有规则

$$r_2'\colon d\_p(J,t_1,\cdots,t_m,I-1)\colon -d\_p(J,v_1,\cdots,v_m,I),H_1,\cdots,H_l$$

从而 $\mu'=d_p(j,c_1,\cdots,c_n,i)$将由 r_2'使用 $d_p(j,a_1,\cdots,a_n,i+1)$导出.

由归纳法原理,充分性得证.

必要性的证明是类似的.事实上,以上证明是可逆的,本文不再累赘.证毕.

定理 1. 本文的算法是正确的;即广义计数规则能够正确地产生查询的回答.

证明:对比本文的算法和[3]的计数算法产生的第(5)组规则,由引理 2,我们断言:对于查询约束集中的任意约束元组$(x_{j1},\cdots,x_{jn})$,元组 $\mu=p(x_{j1},\cdots,x_{jn},y_1,\cdots,y_m)$可被广义计数规则自底向上地推出,当且仅当以$(x_{j1},\cdots,x_{jn})$为初始约束时,$\mu$ 可被计数规则自底向上地推出.

由于文献[3]已证明计数算法是正确的,从而本文的算法也是正确的.证毕.

3　广义计数规则的有效性

规则变换是在编译时完成的,因此对于变换算法来说,更重要的是变换后规则自底向上计算的有效性.我们将从复杂性和 *I/O* 性能两方面讨论广义计数规则的有效性.像大多数研究者一样,我们分别取导出元组的个数和访问 EDB 关系的次数作为二者的度量标准.首先讨论求广义计数规则不动点的时间和空间复杂度.考虑如下规则

$p(X,Y)\colon -e(X,Y)$.

$p(X,Y)\colon -l(X,Xp),p(Xp,Yp),r(Y,Yp)$.

对于查询 p^{bf},它们是计数线性的.如果允许 X,Y,Xp,Yp 为变元向量的话,任何计数线性的逻辑程序都可以转换成这种形式.设 n 为 EDB 谓词 l 和 r 对应关系定义域并的大小,m 为其对应关系大小之和.Marchetti－Spaccamela 等将 l 和 r 对应关系定义域中的每个值对应于图的一个顶点,将 e,l 和 r 对应关系中的每个元组对应于图的一条边,引入图机制,从而将 u_p 元组的导出对应于在图中沿 l 元组对应的边搜索,将 d_p 元组的导出对应于在图中

沿 e 或 r 元组对应的边搜索. 他们的研究表明:对于简单查询"$p(a,Y)$?"和无环数据,求对应计数规则不动点的最坏情况时间复杂度为 $O(mn)$[8]. 由于计数算法不能直接处理广义查询,对于广义查询"$p(\{a_1,\cdots,a_k\},Y)$?",计数算法只能将它分解成简单查询,分别加以处理. 因此,其最坏情况的时间复杂度为 $O(kmn)$.

另一方面,引理 1 和 2 表明:广义计数规则自底向上计算时,所产生的中间元组(即 u_p、d_p 元组)恰好和计数规则,依次用(a_1),…,(a_k)为初始约束自底向上计算时所产生的中间元组一样多. 而广义计数规则导出的 p 元组都是查询的回答,从而必然被计数规则导出. 因此对于广义查询"$p(\{a_1,\cdots,a_k\},Y)$?",广义计数规则最坏情况的时间复杂度为 $O(kmn)$.

由于广义计数规则求值的空间花费主要是存放导出元组的开销,因此广义计数规则求值的空间花费至多为 $O(kmn)$. 实际上,采用 *Semi−Naive* 算法求广义计数规则的不动点时,我们并不需要保留全部中间元组,而只需要保留迭代的上一轮产生的中间元组和回答元组. 因此,广义计数规则求值的空间实际花费比 $O(kmn)$ 小得多. 采用文献[8]时间复杂度分析的类似方法,可以证明:广义计数规则求值的空间花费至多为 $O(kn)$. 非形式地,求广义计数规则不动点的过程相当于在对应的图中以 $a_1,\cdots,a_k$ 为起点,同时进行广度优先搜索,并且迭代一次相当于以每个 $a_i(1\leqslant i\leqslant k)$为根的广度优先搜索生成树的高度增加 1. 由于以 $a_i(1\leqslant i\leqslant k)$ 为根的搜索生成树的任意一层至多有 n 个结点,因此广义计数规则求值的一次迭代至多产生 kn 个中间元组. 而 p 元组至多有 kn 个,因此广义计数规则求值的空间花费至多为 $O(kn)$.

下面我们讨论广义计数规则的 I/O 性能. 为简单计,我们用例 1 加以说明,但这一讨论不难推广到一般情况. 对于约束模式 bf,文献[3]的计数算法将产生计数规则

$u\_sg(x,0)$.

$u\_sg(Xp,I):-par(X,Xp),u\_sg(X,I-1)$.

$d\_sg(X,I):-u\_sg(X,I),person(X)$.

$d\_sg(Y,I-1):-par(Y,Yp),d\_sg(Yp,I)$.

$sg(x,Y):-d\_sg(Y,0)$.

依次令 $x=x_1,x_2,\cdots,x_k$,求以上规则的不动点,取结果的并,便得到集合 $W=\{x_1,x_2,\cdots,x_k\}$ 中每个人的同代人. 不难明白,若 xi 的祖先链的最大长度为 n_i,则求 x_i 的同代人需要 $2(n_i+1)$次访问 EDB 谓词 *par* 的关系. 整个过程需要 $2(n_1+n_2+\cdots+n_k+k)$次访问 EDB 谓词 *par* 的关系;而求广义计数规则的不动点,只需要访问 EDB 关系 $2(max(n_1,\cdots,n_k)+1)$次. 当 k 较大时,后者显然比前者小得多. 由于数据库中的关系一般都很大,其元组必须大部分驻留外存,减少访问 EDB 关系的次数就意味着降低 I/O 开销,改进 I/O 性能.

综上我们断言:本文的算法是计数算法的不失有效性的拓广,并且对于广义查询,广义计数规则的 I/O 性能优于计数规则.

本文算法的意义还在于:它使得计数算法用作子目标的计算策略成为可能. 尽管对于广义查询我们可以通过依次将查询约束集中的每一个约束代入计数规则,求其不动点,然后再取结果的并得到查询的回答. 但是对于子目标,这种处理方法在实践上是不可行的. 因为在知识库中,约束的传递是通过关系运算实现的. 约束在子目标间的传递是一次一个关系,而不是一次一个元组. 由于子目标的约束难得是单值的,因此文献[3]的计数算法很难用作子

目标的求值策略.

4　结束语

至此,我们已将计数算法推广至广义查询的情况.本文的算法具有两个优点:1.由于本文的算法遵循魔集算法的模式,首先改写规则,然后自底向上地处理新规则,本文的算法易于嵌入业已为魔集算法开发的查询优化处理程序.2.本文的算法不仅能够用于查询的求值,而且也能用于子目标的处理.然而,和文献[3]的计数算法一样,本文的算法也不能用于有环数据.如何与文献[6]的技术相结合,给出一个既遵循魔集算法的模式,又可以处理有环数据的广义查询计数算法,尚需进一步的研究.

参考文献

1　Henschen L J, Naqvi S A. On compiling queries in first—order databases. J. ACM 1984,31(1):47—85.

2　Bancilhon F, Maier D, Sagiv Y *et al*. Magic sets and other strange ways to implement logic programs. In: Proc. of the 5th ACM Sympo. on Principles of Database Systems, ACM Press, 1986:1—15.

3　Ullman J D. Principles of database and knowledge—base systems. Vol. II. Computer Science Press, 1989.

4　Han J, Henschen L J. Handle redundancy in the processing of recursive database queries. In: Proc. of the 1987 ACM SIGMOD Intl. Conf. on Management of Data, ACM Press, 1987:73—81.

5　Sacca D, Zaniolo C. Magic counting methods. In:Proc. of the 1987 ACM SIGMOD Intl. Conf. on Management of Data, ACM Press, 1987:49—59.

6　Haddad R W, Haughton J F. Counting methods for cyclic relations. In: Proc. of the 7th ACM Symp. on Principles of Database Systems, ACM Press, 1988:333—340

7　范明.具有多值约束的线性递归查询的有效计算.计算机学报,1992,15(12):913—919.

8　Marchetti—Spaccamela A, Pelaggi A, Sacca D. Worst—case complexity analysis of methods for logic query implementation. In: Proc. of the ACM Sympo. on Principles of Database Systems, ACM Press,1987:294—301.

A COUNTING ALGORITHM FOR GENERALIZED QUERIES

Fan Ming and Li Lianyou

(*Department of Computer Science, Zhengzhou University, Zhengzhou* 450052)

Abstract　The Counting algorithm is one of the best—known qurey—processing algorithms for SL recursions. It performs well for a kind of recursions which are counting linear. However, since it requires that the binding set of the given query bea singleton, it is very difficult to use it as a subgoal processing strategy. In this paper, a new counting algorithm for generalized queries is presented. The algorithm presented here can handle any multi—value binding sets, so it can be used not only for query evaluating, but also for subgoal processing. The correctness of the algorithm and the efficiency of the transformed rules are also discussed briefly in this paper.

Key words　Recursive query, linear recursion, generalized query, counting algorithm.