拓广的左线性递归变换算法及其正确性*

范 明

(郑州大学计算机科学系,郑州 450052)

摘要 本文给出拓广的左线性递归变换算法并证明其正确性.拓广的左线性递归中可以包含一个或多个IDB谓词,它是左线性递归的一般化.和左线性递归计算算法一样,本文提供的算法遵循魔集的模式:首先改写规则,然后用半扑质的自底向上算法计算新规则.算法的有效性也在本文作简略讨论.

关键词 演绎数据库,查询处理,线性递归,左线性递归.

有效地处理由逻辑规则表示的查询,是演绎数据库(知识库)研究的主要课题之一.已经有了一些通用的算法,[1]其中最有影响的是魔集算法.[2,3]作为一种通用算法,魔集算法是相当好的,并且几乎总是比SLD一归结有效.[4]然而,对一些简单、常见的递归查询,还有更好的方法.

Ullman等人提出的右线性、左线性和左一右线性递归变换[4,5]就是针对一些常见的递归类的专用算法.由于实践中许多递归都属于这些类型,并且变换后的规则的自底向上处理相当有效,因此这三种类型的递归变换已被斯坦福大学的NAIL! 系统、[6]MCC的LDL系统[7]用作查询优化处理的首选策略.但是,由于只考虑了仅含一个IDB谓词的情况,这就大大限制了该方法的应用范围.

本文的主要工作是:拓广左线性递归的定义,修改[4,5]的规则改写算法,使之可以处理含任意多个IDB谓词的更加一般化的左线性递归,并证明和讨论修改后的变换算法的正确性和有效性.

1 基本概念

为便于叙述,我们有以下定义:

定义1. 仅由事实定义的谓词称为EDB谓词;由规则定义的谓词称为IDB谓词.EDB谓词和IDB谓词对应的关系分别称为EDB关系和IDB关系.如果一个目标的谓词是EDB(IDB)谓词,则称它为EDB(IDB)目标.

* 本文1991年3月15日收到,1991年6月25日定稿

本文的工作得到了河南省优秀中青年教师奖励基金的资助.作者范明,45岁,副教授,主要研究领域为演绎数据库查询优化处理,逻辑程序设计,程序变换.

本文通讯联系人:范明,郑州 450052,郑州大学计算机科学系

定义 2. 如果规则 r 的体中含有递归谓词(直接或间接由自身定义的谓词),则称 r 为递归的;否则称 r 为非递归的. 若 r 的体中恰含一个递归谓词,则称 r 为线性递归的. 如果 r 头部的每个变量都在 r 的体中的 EDB 或 IDB 子目标中出现,则称 r 是安全的. 若 r 的头部谓词为 p,则称 r 为 p 的规则,或 p—规则.

定义 3. 查询就是一个待满足的目标. 目标是一个谓词实例,其中一些变元不含变量(仅含常量),称为被约束的;另一些含变量,称为自由的. 若某目标的谓词 p 含 n 个变元,则由 b 和 f 组成的、长度为 n 的字符串称为该目标的约束模式;其中 b 表示对应的变元是被约束的,而 f 表示对应的变元是自由的. 约束模式通常用希腊字母 $\alpha,\beta,\cdots$ 等表示. 在不需要指明目标中常量的具体值时,一个目标可以简单地记作 p_α;其中,p 是该目标的谓词,而 α 是其约束模式.

定义 4. 谓词 p 的规则极小集归纳地定义为:(1)p—规则在 p 的规则极小集中;(2)若 r 在 p 的规则极小集中,q 是 r 体中的一个 IDB 子目标,则 q—规则在 p 的规则极小集中;(3)p 的规则极小集仅含由(1)、(2)推出的规则.

定义 5. 逻辑程序由一组规则和一个被约束的查询目标组成. 这组规则是定义查询谓词的规则极小集. 如果一个逻辑程序中至少含有一个递归规则,则称它为递归程序;简称递归(recursion).

其它概念参见[4]. 本文讨论线性递归程序,并假定所有的规则都是安全的.

2　左线性递归的拓广

2.1　拓广的左线性递归的定义

设给定的逻辑程序,或递归中的 IDB 谓词为 $p_1,\cdots,p_l$,查询目标为 p^a,其中 $p\in\{p_1,\cdots,p_l\}$. 进一步,我们假定所有规则的 IDB 子目标(如果有的话)已移至体的左端. 该递归称作拓广的左线性递归,如果下列条件成立:

1. 在给定查询的规则/目标图中,所有的 IDB 谓词都不以两种不同的约束模式出现在目标结点中. 记 p_i 的唯一约束模式为 α_i,显然,若 $p_i=p$,则 $\alpha_i=\alpha$.

2. 所有的规则或者是基本规则(不含 IDB 子目标),或者仅含一个 IDB 子目标.

3. 若规则 r 以 IDB 谓词 p_i 为头部谓词,以谓词 p_j 为 IDB 子目标谓词,则

(a)根据侧向信息传递,若 p_i 的修饰为 α_i,则 p_j 的修饰为 α_j.

(b)p_i 中 α_i 指明的被约束变元和 p_j 中 α_j 指明的被约束变元都是相同的变量,并且这些变量不在 r 中的 IDB 谓词的两个不同的约束变元中出现.

注意:由上述定义,若假定每个 IDB 谓词的被约束变元都在所有自由变元的前面,则每个含 IDB 子目标的规则都具有以下形式

$$p_i(X_1,\cdots,X_m,t_1,\cdots,t_{n_i}) :- p_j(X_1,\cdots,X_m,s_1,\cdots,s_{n_j}),G_1,\cdots,G_K.$$

其中 $X_1,\cdots,X_m$ 是变量,根据 α_i 或 α_j,它们在 p_i 和 p_j 中是被约束的;$t_1,\cdots,t_{n_i}$ 和 $s_1,\cdots,s_{n_j}$ 是任意项,根据 α_i 或 α_j 它们是自由的;而 $G_1,\cdots,G_K$ 是 EDB 子目标. 不失一般性,以后我们假定每个含 IDB 子目标的规则都取这种形式. 对于基本规则,定义没有任何限制.

在上述定义中,如果限定给定的递归中只含一个 IDB 谓词并去掉条件(1),则我们得到

左线性递归的定义.下面的定理保证了上述定义是对左线性递归定义的拓广.

定理 2.1. 每个左线性递归都是拓广的左线性递归,但其逆不真.

证明:对于任意只含一个 IDB 谓词 p 的左线性递归,条件(2)和(3)成立.而在此情况下,条件(3)蕴涵着唯一的 IDB 谓词 p 具有唯一约束模式,从而该递归也是拓广的左线性递归.其逆不真由下例表明.

例 1:设 father 和 mother 是 EDB 谓词,其中,father(x,y)表示 y 是 x 的父亲,mother(x,y)表示 y 是 x 的母亲.又设 anc1(x,y)表示 y 是 x 的男性祖先之一,anc2(x,y)表示 y 是 x 的女性祖先之一,则 IDB 谓词 anc1 和 anc2 可由以下规则定义:

r_1: anc1(X,Y):—father(X,Y).

r_2: anc1(X,Y):—anc1(X,Z),father(Z,Y).

r_3: anc1(X,Y):—anc2(X,Z),father(Z,Y).

r_4: anc2(X,Y):—mother(X,Y).

r_5: anc2(X,Y):—anc2(X,Z),mother(Z,Y).

r_6: anc2(X,Y):—anc1(X,Z),mother(Z,Y).

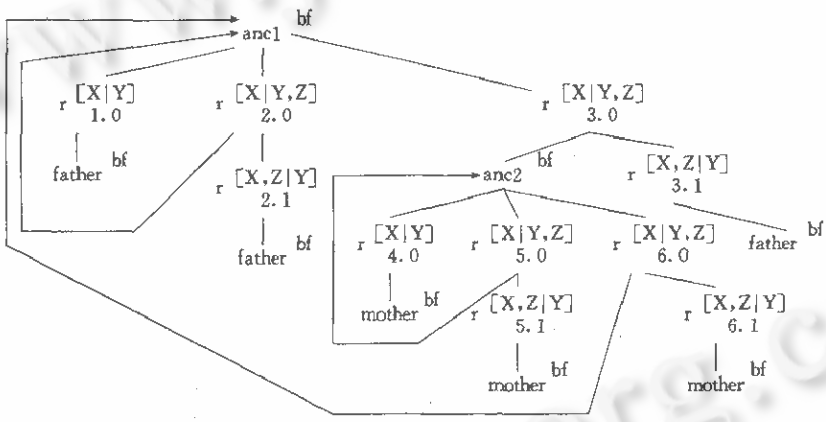

图1 例1的规则/目标图

为找出某人 x_0 的所有男性祖先,可用查询 anc1(x_0,Y).该查询和规则 $r_1,\cdots,r_6$ 构成的递归包含了两个相互递归的 IDB 谓词,因而不是右线性的.但由它的规则/目标图(图 1)可以看出,以 IDB 谓词 anc1 和 anc2 为谓词的目标结点都具有唯一约束模式,并不难验证条件(2)、(3)亦成立,从而它是拓广的左线性递归.

2.2 变换算法

现在,我们给出拓广的左线性递归变换算法的形式描述.设任意给定的拓广的左线性递归包含 IDB 谓词 $p_1,\cdots,p_l$,查询目标为 p^a,其中 $p\in\{p_1,\cdots,p_l\}$.首先,对于每个 IDB 谓词 p_i,我们创建一个新谓词 a_p_i,称为 p_i 的回答谓词.然后,按以下步骤产生一组新规则:

1. 对于每个含 IDB 子目标的规则

$p_i(X_1,\cdots,X_m,t_1,\cdots,t_{n_i}):-p_i(X_1,\cdots,X_m,s_1,\cdots,s_{n_j}),G_1,\cdots,G_K.$

构造规则:

$a\_p_i(\sigma(t_1),\cdots,\sigma(t_{n_i}))-a\_p_j(\sigma(s_1),\cdots,\sigma(s_{n_j})),\sigma(G_1),\cdots,\sigma(G_K).$

这里,σ 是一个置换,它将每个 X_i 用 x_i 替换,而保持其它符号不变;其中,$1\leqslant i\leqslant m$,而

$x_1,\cdots,x_m$ 是来自查询的常量.

Ⅰ. 对于每个基本规则

$p_i(t_1,\cdots,t_m,s_1,\cdots,s_{n_i}):-G_1,\cdots,G_K.$

将项 $t_1,\cdots,t_m$ 与查询常量 $x_1,\cdots,x_m$ 一致置换，得到 τ. 扩充 τ，令它在不同于 $t_1,\cdots,t_m$ 中的符号上为恒等的. 然后构造规则

$a\_p_i(\tau(s_1),\cdots,\tau(s_{n_i})):-\tau(G_1),\cdots,\tau(G_K).$

Ⅱ. 对查询 p_α 的回答可由以下规则导出：

$p(x_1,\cdots,x_m,Y_1,\cdots,Y_n):-a\_p(Y_1,\cdots,Y_n).$

其中 $x_1,\cdots,x_m$ 是查询中的常量.

为得到对查询的回答，只需用自底向上的算法，如半朴质算法，计算新规则.

若限定递归中只含一个 IDB 谓词，以上算法和左线性递归变换算法是一致的.

例 2：作为例子，考虑对例 1 的程序进行变换. 对于该例的简单形式，第Ⅰ组规则中的置换 σ 是不必要的，因为被约束变元 X 仅在 r_2,r_3,r_5 和 r_6 的头部及 IDB 子目标的第一个变元处出现. 对于这四个规则，我们有以下新规则：

a_anc1(Y):-a_anc1(Z),father(Z,Y).

a_anc1(Y):-a_anc2(Z),father(Z,Y).

a_anc2(Y):-a_anc2(Z),mother(Z,Y).

a_anc2(Y):-a_anc1(Z),mother(Z,Y).

对于第Ⅱ组规则，一致置换 τ 是 $\tau(X)=x_0$，而保持其它符号不变. 由 r_1 和 r_3，我们有以下第Ⅱ组规则

a_anc1(Y):-father(x_0,Y).

a_anc2(Y):-mother(x_0,Y).

查询的回答由以下规则(第Ⅱ组)导出

anc1(x_0,Y):-a_anc1(Y).

2.3 变换算法的正确性

我们说变换算法是正确的，意指对于任意给定的拓广的左线性递归，变换前的程序和变换后的规则将对原来的查询产生相同的回答. 在下面的讨论中，我们假定对原来的程序(旧规则)和变换后的程序(新规则)均采用自底向上的朴质计算算法计算. 我们的目标是证明.

定理 2.2. 拓广的左线性递归的变换算法是正确的.

为证明该定理，我们先证明一个引理.

引理 2.1. 设 $x_1,\cdots,x_m$ 为查询常量. 对于任意 IDB 谓词 p_i，$p_i(x_1,\cdots,x_m,y_1,\cdots,y_{n_i})$ 可以由旧规则导出的充分必要条件是 $a\_p_i(y_1,\cdots,y_{n_i})$ 可以由新规则导出.

证明：必要性. 显然，在朴质计算开始之前，任何 p_i 对应的关系均不含元组. 设 $p_i(x_1,\cdots,x_m,y_1,\cdots,y_{n_i})$ 在第 $K(\geqslant 1)$ 轮由旧规则导出，今对 $K\geqslant 1$ 进行归纳.

基始：$K=1$. 此时 $p_i(x_1,\cdots,x_m,y_1,\cdots,y_{n_i})$ 必然是由形如

r_1: $p_i(t_1,\cdots,t_m,s_1,\cdots,s_{n_i}):-G_1,\cdots,G_K.$

的规则导出的. 由变换算法，我们有第Ⅱ组的规则

r'_1: $a\_p_i(\tau(s_1),\cdots,\tau(s_{n_i})):-\tau(G_1),\cdots,\tau(G_K).$

其中，τ 是 $t_1,\cdots,t_m$ 与 $x_1,\cdots,x_m$ 的一致置换. 非形式地，r'_1 是由 r_1 将其中的 $t_1,\cdots,t_m$ 的出现

分别用查询常量 $x_1,\cdots,x_m$ 替换，然后去掉 p_i 的前 m 个变元(它们是查询常量)，并将 p_i 改为 a_p_i 而得到的. 即然 $p_i(x_1,\cdots,x_m,y_1,\cdots,y_{n_i})$可由 r_1 导出，则它可由

$$p_i(x_1,\cdots,x_m,\tau(s_1),\cdots,\tau(s_{n_i})):-\tau(G_1),\cdots,\tau(G_K).$$

导出. 从而，$a_p_i(y_1,\cdots,y_{n_i})$可以由 r'_1，即由新规则导出.

归纳：K>1. 由于在 K>1 轮，不含 IDB 子目标的规则不可能再产生新的元组，故 $p_i(x_1,\cdots,x_m,y_1,\cdots,y_{n_i})$必然是由形如

$$r_2: p_i(X_1,\cdots,X_m,t_1,\cdots,t_{n_i}):-p_j(X_1,\cdots,X_m,s_1,\cdots,s_{n_j}),G_1,\cdots G_K.$$

的规则，使用 p_j 的一个在第 K 轮之前业已导出的元组(不妨设为 μ)导出的. 显然，μ 必取 $p_i(x_1,\cdots,x_m,w_1,\cdots,w_{n_j})$这种形式. 由归纳假设，$a_p_i(w_1,\cdots,w_{n_j})$可由新规则导出. 另一方面，由变换算法，对应于 r_2 我们有第 I 组新规则

$$r'_2: a\_p_i(\sigma(t_1),\cdots,\sigma(t_{n_i}))-a\_p_j(\sigma(s_1),\cdots,\sigma(s_{n_j})),\sigma(G_1),\cdots,\sigma(G_K).$$

其中，σ 是一个置换，它将 X_i 的出现用 x_i 替换. 既然 $p_i(x_1,\cdots,x_m,y_1,\cdots,y_{n_i})$由 r_2 用 μ 导出，则它必然能由

$$p_i(x_1,\cdots,x_m,\sigma(t_1),\cdots,\sigma(t_{n_i}))-p_j(x_1,\cdots,x_m,\sigma(s_1),\cdots,\sigma(s_{n_j})),\sigma(G_1),\cdots,\sigma(G_K).$$

用 μ 导出. 而 r'_2 可由上面的规则去掉 p_i 和 p_j 的前 m 个变元(它们是查询常量)，并将 p_i 改为 a_p_i，p_j 改为 a_p_j 得到. 因而 $a_p_i(y_1,\cdots,y_{n_i})$必然能由 r'_2 利用 $a_p_j(w_1,\cdots,w_{n_j})$导出. 由归纳法原理，必要性得证.

充分性的证明是类似的. 事实上，以上证明都是可逆的，本文不在赘述.

在引理 2.1 中，取 p_i 为查询谓词 p，注意到第Ⅲ组规则，立明定理 2.2 成立，变换算法的正确性得证.

2.4 有效性

事实上，本文算法产生的规则是否可以有效地计算的关键在于第Ⅰ、Ⅱ组规则是否可以有效地计算. 而这组规则可以由魔集的第Ⅳ组规则按以下步骤化简得到：① 将辅助谓词的规则体代入化简，删除其中的辅助谓词；② 用 $x_1,\cdots,x_m$ 替换规则中那些在 m_p_i 的变元中出现的变量，并删除 m_p_i；③ 删除 p_i 的前 m 个变元，并用 a_p_i 替换 p_i.

[4]已证明这些化简不影响效率，并且还可以利用合取查询的优化技术进一步优化化简后规则的计算. 因此，采用自底向上的方法计算时，本文算法产生的第Ⅰ、Ⅱ组规则的计算花费不多于魔集的第Ⅳ组规则的计算花费. 唯一的其它规则是第Ⅲ组的唯一规则，它将 a_p 的元组转换成 p 的元组，这基本上是拷贝回答元组. 如果我们在产生 a_p 的元组时附上查询常量，第Ⅲ组的唯一规则完全可以省略，这种拷贝也完全可以避免. 因此，我们断言：在最坏情况下，本文算法产生的规则至少象魔集规则一样有效.

一般地，即使对于未拓广的左线性递归，也不能保证变换后的规则比魔集规则更加有效.[4]但是，(拓广的)左线性递归变换比魔集变换简单，变换所产生的规则数目和 IDB 谓词数目都远少于对应的魔集变换(例如，对于例 1，本文的变换产生了 7 个规则，包含 3 个 IDB 谓词；而魔集变换将产生 21 个规则，包含 14 个 IDB 谓词)，并且产生的规则至少象魔集规则一样有效，因此(拓广的)左线性递归变换仍比魔集变换更为可取.

结束语：我们已证明，本文的结果是对左线性递归的拓广. 右线性递归的拓广问题也已有了结果. 对于同时含有左线性和右线性的递归变换，其理论基础是可交换引理，该引理一

般不能用于含多个 IDB 谓词的递归. 把可交换引理和/或左－右线性递归变换拓广至含多个 IDB 谓词的情况，不仅在理论上，而且在应用上都是有意义的. 该问题正在研究中，目前尚无一般结果.

致谢　笔者感谢加拿大西蒙. 弗雷泽大学计算机系 J. Han 博士. 在我访问加拿大期间，曾获得他的多方面帮助，并和他进行过一些有益的讨论.

参考文献

1 Bancilhon F. Ramakrishnan R. An amateur's introduction to recursive query processing strategies. In Proc. of the 1986 ACM SIGMOD Intl. Conf. on Management of Data, Washinghton:1986, ACM Press, 1986:16－52.

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 SIGACT－SIGMOD Sympo. on Principles of Database systems, Cambridge: 1986, ACM Press, 1986:1－15.

3 Beery C, Ramakrishnan R. On the power of magic. In Proc. of the 6th ACM SIGACT－SIGMOD Sympo. on Principles of Database systems, ACM Press, 1987:269－283.

4 Ullman J D. Principles of database and knowledge－Base systems, Vol. II, Computer Science Press, 1989.

5 Naughton J F, Ramakrishnan R, Sagiv Y *et al*. Efficient evaluation of right－, left－, and multi－linear rules. In Proc. of the 1989 ACM SIGMOD Intl. Conf. on Management of Data, ACM Press, 1989:235－242.

6 Morris K, Ullman J D, Van Gelder A. Design overview of the NAIL! system. In Proc. of the Third Intl. Conf. on Logic Programming, ACM Press1986: 554－568.

7 Naqvi S A, Tsur S. A logic language for data and knowledge bases. Computer Science Press, 1988.

8 范明. 拓广的右线性递归变换算法及其正确性. 计算机学报，1992;15(12):906－912.

TRANSFORMATION FOR GENERALIZED LEFT－LINEAR RECURSIONS: ALGORITHM AND ITS CORRECTNESS

Fan Ming

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

Abstract　This paper presents an algorithm which transform a generalized left－linear recursion into more efficient rules, and shows the correctness of the given algorithm. A generalized left－linear recursion contains one or more IDB predicates and its definition is essentially generalized from that of left－linear recursions. As the algorithm for the evaluation of left－linear recursions, the algorithm we present here follows the magic－sets paradigm of a rewriting phase followed by semi－naive bottom－up evaluation. The efficinecy of the transformed rules is also discussed in this paper.

Key words　Deductive database, query processing, linear recursion, left－linear recursion.