重叠规则与歧义性*

陆朝俊　孙永强　林　凯

（上海交通大学计算机系　上海　200030）

摘要　重写系统是一种一般的计算模型.重写系统的归约策略的范式化性质对于实际应用重写系统进行计算具有决定意义,而重叠规则导致的歧义性是使归约过程复杂化的重要原因.本文对重写系统的歧义性进行了初步研究,并对一类常见的歧义问题作了具体分析,同时提出了解决办法.

关 键 词　项重写系统,归约策略,重叠规则,歧义性.

项重写系统是由重写规则组成的系统,重写规则是表示一个项规律性地转换(重写)到另一个项的规则.由于计算过程总可以看作计算对象的语法表示按一定的规则不断进行转换的过程,故重写系统是一种非常一般的计算模型.

重写系统具有多方面的应用.最常见的是,给定一个方程式集合(方程式理论),可以生成一个重写系统,并利用这个重写系统来进行有关的计算、推理.与方程式理论相比,重写系统具有明显的操作意义,因此所得重写系统可以看作是为方程式理论建立了操作语义.由于常用方程式描述程序的功能,即给出程序(或抽象数据类型)的代数规范,因此重写系统实际上可以用作程序规范的(抽象)解释器,从而支持"可直接执行的规范"(或"规范的直接实现")的思想.

在为方程式理论生成重写系统时,该重写系统往往包含重叠的重写规则,这种重叠反映了执行重写(归约)时的歧义性,它直接影响归约策略的范式化性质,即归约策略作用于计算对象时,能否终止于该对象的范式的问题.目前对于不包含重叠规则的正交重写系统的研究比较成熟,范式化策略问题已被解决,但对有重叠规则的重写系统来说,包括范式化策略在内的各种问题都研究甚少,其原因是重叠规则的存在导致归约过程极度复杂.

本文对重写系统的歧义性进行一般的分析,并对一类常见的歧义性提出了解决方法.

1　重写系统简介

重写系统具有非常简单的语法和语义,它是建立在项和项集上的重写关系这 2 个基本

* 本文研究得到国家 863 高科技项目基金资助.作者陆朝俊,1964 年生,讲师,主要研究领域为新型程序语言.孙永强,1931 年生,教授,博士导师,主要研究领域为计算理论,新型程序语言.林凯,1962 年生,副教授,主要研究领域为计算理论,新型程序语言.

本文通讯联系人:陆朝俊,上海 200030,上海交通大学计算机系

本文 1995－05－02 收到修改稿

成分之上的.

给定函数符号集合 $\mathscr{F}$ 和变量符号集合 $\mathscr{V}$,它们生成的项集 $\mathbf{T}(\mathscr{F},\mathscr{V})$是由变量和如下形式的项组成:$f(M_1,\ldots,M_n)$,其中 $f\in\mathscr{F}$ 是 n 元函数符号,$M_i\in\mathbf{T}(\mathscr{F},\mathscr{V})$. 为了精确描述项的结构及其子项的位置,常用"出现"的概念. 称一个自然数序列 $u\in\mathbf{N}^*$ 为出现,空序列记为 Λ,用 $\cdot$ 表示序列的毗联,则有如下定义:

定义 1.1. 令 $M\in\mathbf{T}(\mathscr{F},\mathscr{V})$,$M$ 的出现集合 $\mathscr{O}(M)$及 M 在 $u\in\mathscr{O}(M)$处的子项 M/u 定义为

(1)若 $M\in\mathscr{V}$,则 $\mathscr{O}(M)=\{\Lambda\}$,$M/\Lambda=M$;

(2)若 $M\equiv f(M_1,\ldots,M_n)$,则 $\mathscr{O}(M)=\{\Lambda\}\cup\{i\cdot u\mid u\in\mathscr{O}(M_i),1\leqslant i\leqslant n\}$

$$M/\Lambda=M,M/i\cdot u=M_i/u$$

在出现之间定义前缀关系$\leqslant$:$u\leqslant v$ *iff* $\exists\ w(v=u\cdot w)$,并记为 $v/u=w$. 又若 $u\leqslant v$ 和 $v\leqslant u$ 都不成立,则记为 $u\mid v$.

重写过程就是对子项进行代换的过程.

定义 1.2. 令 $M,N\in\mathbf{T}(\mathscr{F},\mathscr{V})$,$u\in\mathscr{O}(M)$,则子项代换 $M[u\leftarrow N]$定义为

$$M[\Lambda\leqslant f\leftarrow N]=M,$$

$$f(M_1,\ldots,M_n)[i\cdot u\leftarrow N]=f(M_1,\ldots,M_i[u\leftarrow N],\ldots,M_n)$$

子项代换的特例是变量代换,即从变量集合到项集的映射 θ,θ 扩张到项集之上即有:

$$\theta(f(M_1,\ldots,M_n))=f(\theta(M_1),\ldots,\theta(M_n))$$

$\theta(M)$称为 M 的实例. 对项 M,N,若有 θ 使得 $\theta(M)\equiv\theta(N)$,则称 M 和 N 可合一,记为 $M\triangledown N$,并称 θ 是 M 和 N 的合一子. 若 $N\equiv\theta(M)$,则称 M 比 N 一般,记为 $M\lhd N$. 众所周知,若 $M\triangledown N$,则通过合一算法可找出它们的最一般公共实例,并得到称为最一般合一子(*mgu*)的变量代换 θ.

定义 1.3. 令 $M,N\in\mathbf{T}(\mathscr{F},\mathscr{V})$,$u\in\mathscr{O}(N)$,若 $M\triangledown N/u$,则称 M 与 N 在出现 u 处匹配.

显然若 $M\in\mathscr{V}$,则 M 与任何项匹配. 当然我们更关心非变量项之间的匹配.

定义 1.4. $\mathbf{T}(\mathscr{F},\mathscr{V})$上的重写规则是项的有序对$\langle M,N\rangle$,且满足下面 2 个条件:

(1)$M\notin\mathscr{V}$;

(2)N 中只能包含 M 中出现过的变量.

重写规则$\langle M,N\rangle$常记作 $M\rightarrow N$.

定义 1.5. $\mathbf{T}(\mathscr{F},\mathscr{V})$上重写规则的一个集合 $\mathscr{R}$ 称为一个项重写系统,简称重写系统.

重写系统 $\mathscr{R}$ 确定了一个重写关系$\rightarrow$:

(1)对任何$\langle M,N\rangle\in\mathscr{R}$,$\theta(M)\rightarrow\theta(N)$;

(2)若 $M_k\rightarrow N_k$,则 $f(M_1,\ldots,M_k,\ldots,M_n)\rightarrow f(M_1,\ldots,N_k,\ldots,M_n)$

定义 1.6. 对规则$\langle M,N\rangle$的实例$\langle M_1,N_1\rangle$,M_1 称为可归约式,简记为$\langle M,N\rangle-redex$或 *redex*.

归约项 M 中的一个 *redex* 的过程就是用相应规则右部去替换该 *redex* 的过程,称为一个归约步:$M\rightarrow N$. 可以定义$\rightarrow$的各种闭包及归约序列等概念. 如果一个项中不含 *redex*,则称之为范式. 归约策略就是确定在每一归约步中选择哪一个(或一些)*redex* 进行重写的策

略. 一个归约策略是范式化的,当且仅当对任何有范式的项,该策略一定能把这个项归约到它的范式.

目前对重写系统的研究一般局限于正交重写系统,即满足下列条件的重写系统:

(1)若$\langle M,N\rangle\in\mathscr{R}$,则 M 中没有相同变量;

(2)$\mathscr{R}$ 中不含重叠规则.

关于重叠规则我们在下一节中讨论. 这里要指出的是,在实际应用中经常遇到非正交重写系统,而目前对这类重写系统的性质尚知之甚少.

2 重叠规则与歧义性

我们希望对包含重叠规则的重写系统进行研究,为此必须先对重叠规则及其对归约过程的影响进行探讨.

在一个项中可能同时存在多个 *redex*,于是这个项就有多种归约途径. 这些 *redex* 的相互位置关系对归约过程有重要影响.

定义 2.1. 重写系统 $\mathscr{R}$ 具有歧义性,当且仅当 $\mathscr{R}$ 中存在 2 条不同的规则$\langle\alpha_1,\beta_1\rangle$和$\langle\alpha_2,\beta_2\rangle$,使得 $\alpha_1\triangledown\alpha_2$. 这时也称$\langle\alpha_1,\beta_1\rangle$和$\langle\alpha_2,\beta_2\rangle$间有歧义,记为$\langle\alpha_1,\beta_1\rangle\infty\langle\alpha_2,\beta_2\rangle$.

由此定义看出,对某个 *redex* M 可能同时存在多条可应用规则,这就导致在归约 M 时有歧义. 注意歧义性完全不同于在一个项中选取不同的 *redex* 进行归约的问题.

一般说来,不能笼统地从重写系统中完全排斥歧义性,关键是要能对其进行有效的处理. 我们来对歧义现象作进一步的分析.

首先,一条重写规则当然不会与它自身有歧义,这已在定义中表明了. 但在后面讨论的重叠现象中,则与此相反.

令$\langle\alpha_1,\beta_1\rangle\infty\langle\alpha_2,\beta_2\rangle$,$\theta$ 是 α_1 和 α_2 的 mgu,

(1)若 $\theta(\beta_1)\equiv\theta(\beta_2)$,则称这种歧义性为弱歧义性;

(2)若 $\alpha_1\lhd\alpha_2$,则称$\langle\alpha_1,\beta_1\rangle$比$\langle\alpha_2,\beta_2\rangle$一般,也记作$\langle\alpha_1,\beta_1\rangle\lhd\langle\alpha_2,\beta_2\rangle$. 这时考虑规则右部的情况,若 $\theta(\beta_1)\equiv\beta_2$,则为弱歧义性的特例;

(3)若$\langle\alpha_1,\beta_1\rangle\lhd\langle\alpha_2,\beta_2\rangle$和$\langle\alpha_2,\beta_2\rangle\lhd\langle\alpha_1,\beta_1\rangle$都不成立,则称$\langle\alpha_1,\beta_1\rangle$和$\langle\alpha_2,\beta_2\rangle$是根重叠的. 这时同样存在弱歧义性特例.

以上概念可通过例子来说明.

例 1:(i)$\{f(A)\to B,f(x)\to C\}$中,$f(x)\to C$ 比 $f(A)\to B$ 一般,而$\{f(A)\to B,f(x)\to B\}$则为弱歧义性的一种;

(ii)$\{f(A,x)\to B,f(x,C)\to D\}$中的 2 条规则是根重叠的,而$\{f(A,x)\to B,f(x,C)\to B\}$又是弱歧义性的一种.

为使重写计算是确定性的,应该排除(2)类歧义性. 因为在实际应用中,我们总是希望用最一般的规则来描述程序的功能,而不需要再加入蛇足式的特例规则,尤其是当特例规则与一般规则有冲突的时候. 但(3)类歧义性则是常见的,例如 Parallel Or 规则:

$\mathrm{or}(\mathrm{true},x)\to\mathrm{true}$

$\mathrm{or}(x,\mathrm{true})\to\mathrm{true}$

or(false,false)→false

就包含根重叠规则.

除上述歧义现象之外,还有一种更复杂更一般的重叠现象.

定义 2.2. 设$\langle\alpha_1,\beta_1\rangle,\langle\alpha_2,\beta_2\rangle\in\mathscr{R}$,$u\in\mathscr{O}(\alpha_1)$,$u\neq\Lambda$且$\alpha_1/u\notin\mathscr{V}$,若$\alpha_1/u\nabla\alpha_2$,则称$\langle\alpha_2,\beta_2\rangle$重叠于$\langle\alpha_1,\beta_1\rangle$之上.这种重叠确定了一个临界对$[P,Q]$

$$P\equiv\theta(\alpha_1)[u\leftarrow\theta(\beta_2)],$$

$$Q\equiv\theta(\beta_1)$$

其中θ是α_1/u和α_2的mgu.

例 2:$\{f(g(x))\to d(x),g(h(y))\to e(y)\}$是重叠的,并确定临界对$[f(e(x)),d(h(x))]$.

显然,临界对反映了同一个项按 2 种不同途径进行归约所得的不同结果,这个概念可以推广到前述根重叠的情形.

要说明的是,重叠甚至可以发生在同一条规则之间,如下例所示.

例 3:规则$f(f(x))\to g(x)$是自身重叠的.从项$f(f(f(x)))$可得临界对

$$[f(g(x)),g(f(x))]$$

例 4:2 条规则之间可以有多种歧义现象.如

$$f(f(x))\to\cdots\cdots$$

$$f(f(g(x))\to\cdots\cdots$$

之间既有"一般性"差别,又是重叠的.$f(f(x))\to\cdots\cdots$还与自身重叠.

至此,我们得出项中各$redex$之间的位置关系有以下几种情形.设$M\in\mathbf{T}(\mathscr{F},\mathscr{V})$,$u,v\in\mathscr{O}(M)$,$R_1\equiv M/u$和$R_2\equiv M/v$是$redex$,

(1)若$u|v$,则称R_1和R_2是平行的;

(2)若$u\leqslant v$,则称R_1和R_2是嵌套的.这时又分 2 种情形.令R_1是$\langle\alpha_1,\beta_1\rangle-redex$,$R_2$是$\langle\alpha_2,\beta_2\rangle-redex$,

(2.1)若$\alpha_1/(v/u)\in\mathscr{V}$,则称$R_1$和$R_2$是(变量)代换型嵌套的;

(2.2)若$\langle\alpha_1,\beta_1\rangle,\langle\alpha_2,\beta_2\rangle$是(在$v/u\in\mathscr{O}(\alpha_1)$处)重叠的,则称$R_1$和$R_2$是重叠型嵌套的;

(3)若$v\leqslant u$,这与(2)类似,不赘述.

重叠规则的存在导致了归约过程的复杂化.例如,设$R_1[R_2]$是重叠型嵌套的$redex$,若归约外层的R_1,则内层的R_2将不复存在(专门的术语叫作没有"后裔",在此不详细讨论);同样,归约R_2则使得R_1没有后裔.这种情形在正交重写系统中是没有的.我们认为,可以通过对临界对施加限制条件,从而能够处理重叠对归约造成的影响.[3]

3　一类歧义问题

递归函数是程序设计中的重要形式.我们将讨论有关递归函数实现中的歧义性问题.

考虑典型的阶乘函数的程序(用重写规则的形式):

$$f(0)\to 1 \tag{1}$$

$$f(n)\to n*(f(n-1)) \tag{2}$$

我们所希望的实现中，对 $f(2)$ 的计算应该是这样的：

$f(2)\rightarrow 2*(f(2-1))$ 规则(2)

$\rightarrow 2*(f(1))$ 基本算术运算

$\rightarrow 2*(1*(f(1-1)))$ 规则(2)

$\rightarrow 2*(1*(f(0)))$ 基本算术运算

$\rightarrow 2*(1*1)$ 规则(1)

$\rightarrow 2*1$ 基本算术运算

$\rightarrow 2$ 基本算术运算

在这个计算过程中有 2 个问题：

(1)重写过程中有一些隐含更优先的基本算术运算. 如从 $2*(f(2-1))$ 到 $2*(f(1))$ 的归约步中，隐含着先计算(2−1)并替换成 1. 但从重写系统的角度看，对 $2*(f(2-1))$ 完全可以应用规则(2)，从而得到 $2*((2-1)*(f((2-1)-1)))$，对此还可以继续应用规则(2). 经过一些重写和算术化简之后，就可能得到 $2*(1*(0*f(-1)))$. 这个项可能归约到 0，也可能归约不终止.

(2)对项 $f(0)$ 来说，它既与规则(1)匹配，也可以与规则(2)匹配，因而可能重写成 $0*(f(0-1))$，导致与上述相同的情况.

在函数式语言的实现中，由于通常采用了“自顶向下，自左向右”的模式匹配策略，即程序(规则)的书写顺序影响着匹配规则的选择，因而上述问题得以避免. 但这种策略实际上是过程式的(读者可以与用过程式语言书写的阶乘程序进行比较，从而看出它们在控制流程上的共同性)，与作用式语言的宗旨相背. 在重写系统的语义下，并不存在这种“预定的”歧义规则的选取原则，归约策略并不考虑重写的书写顺序之类.

上述问题一般地存在于任何基于规则的系统之中，例如逻辑式语言. PROLOG 语言中的 *cut* 机制实际上就具有排除歧义规则的作用，在此我们不作进一步探讨.

解决上述问题有 2 种方法. 其一，消除重写规则的歧义性. 仍以上例来说明：将变量 n 改用构造子形式即可. 这符合我们对递归函数的直观理解：0 和 n 是分别处理的，而非特例与一般的关系. 这样，阶乘函数可写成：

$$f(0)\rightarrow 1 \tag{3}$$

$$f(S^{(n)}(0))\rightarrow n*(f(S^{(n-1)}(0))) \tag{4}$$

这里函数符号 S 代表“后继”，是一个构造子. 要注意的是，规则(4)实际上应理解为一系列(无穷多条)规则：

$$f(S(0))\rightarrow 1*f(0)$$

$$f(S(S(0)))\rightarrow 2*f(S(0))$$

$$\cdots\cdots$$

这当然不难借助某些实现技术来做到. 可以看到，规则(3)、(4)之间没有歧义性，对 $f(S(S(0)))$ 的计算就会终止了.

另一种解决办法是允许歧义规则的存在，而另外引进使歧义“合流”的条件(或规则). 例如，$f(0)$ 可分别按规则(1)、(2)归约到 1 和 $0*f(0-1)$，由于 $f(0-1)$ 是没有意义的，我们可以增加规则 $0*f(0-x))\rightarrow 1$ 之类，当然这必须不与其他规则冲突. 这个问题还需进行更深入的研究，可以设想，我们必须对由歧义或重叠引起的临界对施加某些条件.[3]

参考文献

1 Huet G. Confluent reductions: abstract properties and applications to term rewriting systems. JACM, 1980, 27 (4): 797~821.

2 Toyama Y. Strong sequentiality of left-linear overlapping term rewriting systems. In: Proc. 7th IEEE Symp. on Logic in Computer Science, 1992. 274~284.

3 陆朝俊. 重写系统研究[博士论文]. 上海交通大学, 1994.

OVERLAPPING RULES AND AMBIGUITIES

Lu Chaojun Sun Yongqiang Lin Kai

(*Department of Computer Science Shanghai Jiaotong University Shanghai* 200030)

Abstract Term rewriting systems are a general model of computation. The normalization property of a reduction strategy for TRS's plays an important role in the application of TRS's, and overlapping rules and other types of ambiguities complicates reduction strategies greatly. This paper deals with these ambiguities in general and proposes an approach to a general class of problems involving ambiguities.

Key words Term rewriting systems, reduction strategy, overlapping rules, ambiguities.