建立于谓词逻辑上的递归程序及其操作语义

邵志清

(华东化工学院计算机科学系，上海)

SYNTAX AND OPERATIONAL SEMANTICS OF RECURSIVE PROGRAMS BASED ON PREDICATE LOGIC

Shao Zhiqing

(*Department of Computer Science, East China University of Chemical Technology*)

ABSTRACT

In a traditional approach to operational semantics of recursive programs, ω was introduced as an undefined value and then flat partial orders, ω-extensions, transition relation, computation sequences etc. were defined based on a nonstandard interpretation. The aim of this paper is to avoid introducing ω. Instead we introduce directly the syntax and operational semantics of recursive programs based on a basis for predicate logic and its standard interpretation. Hence we refute the claim of Loeckx and Sieber that no operational semantics of recursive programs can build upon predicate logic.

摘 要

对于递归程序的操作语义，常用的刻划方法是引入无定义值ω，再定义函数的ω 延拓和平坦偏序等概念，导入转移关系和计算序列。本文采用优先处理某些项的原则避免引入ω，从而直接根据谓词逻辑的基底的解释引进计算序列，并且保证了其中的转移关系是一个函数。由此我们否定了Loeckx 和Sieber 所宣称的“递归程序的操作语义不能建立于谓词逻辑上”的断言。

1990 年1 月7 日收到，1990 年3 月26 日定稿。

§1. 引言

递归程序在计算机科学中占有重要地位，在对其操作语义的描述方法中，通常的处理步骤是：引入无定义值ω，进而定义平坦偏序和ω延拓等概念，最后用简化和替换导入计算序列，从而刻划递归程序的操作语义[1]。但是这种方法有一个重大缺陷：它不能建立于谓词逻辑上，因为在处理没有定义的情形时引入了无定义值ω。

本文旨在对递归程序的操作语义的上述刻划做出改进，主要思路是：在处理计算序列的项的简化时，我们按优先性的高低而先后进行，从而避免引入无定义值ω。因此我们可以直接在谓词逻辑上定义递归程序及其操作语义，即由谓词逻辑的基底B 决定递归程序的语法，而由B 的解释确定递归程序的语义，从而否定了Loeckx 和Sieber 所宣称的“递归程序的操作语义不能建立于谓词逻辑上”的断言。

本文所用的谓词逻辑的基底及其解释、指派等概念与符号均可在[1] 中找到。以下除非特别声明，B 均指谓词逻辑的某个基底，并约定B 中命题常元只有true 和false。

§2. 递归程序的语法

本节的主要目的是在谓词逻辑的基础上建立递归程序的语法，为此先引入下列辅助符号:

技术性符号: $\Leftarrow$, if, then, else, fi;

一个递归可枚举的无穷的函数变元集$W = \{G, H, \cdots, G_1, \cdots, G', \cdots\}$，其中每个函数变元都有一个正整数作为它的元数。

定义1 (递归程序的项和公式的语法) 设B=(F, P) 是基底，则B 上递归程序的项和公式可定义如下:

(1) 项的定义:

(a) 每个常元$c \in F$ 是项，每个变元$x \in V$ 是项;

(b) 若$t_1, \cdots t_n$ 是项，$f \in F$ 是n 元函数符号，则$f(t_1, \cdots, t_n)$ 是项;

(c) 若$t_1, \cdots, t_n$ 是项，$G \in W$ 是n 元函数变元，则$G(t_1, \cdots, t_n)$ 是项;

(d) 若t_1、t_2 是项，e 是公式，则$if\ e\ then\ t_1\ else\ t_2\ fi$ 是项;

(2) 公式的定义:

(a) true、false 是公式;

(b) 若t_1、t_2 是项，则$(t_1 \equiv t_2)$ 是公式;

(c) 若$t_1, \cdots, t_n$ 是项，$p \in P$ 是n 元谓词符号，则$p(t_1, \cdots, t_n)$ 是公式;

(d) 若w 是公式，则$(\neg w)$ 是公式;

(e) 若w_1、w_2 是公式，则$(w_1 \wedge w_2)$ 是公式。

定义2 (递归程序的语法) 设B 是基底，则B 上的一个递归程序是指下列n 个等式:

$$G_1(x_{11}, \cdots, x_{1s_1}) \Leftarrow t_1$$
$$\vdots$$
$$G_n(x_{n1}, \cdots, x_{ns_n}) \Leftarrow t_n$$

其中有一个固定的$G_k(1 \leq k \leq n)$ 作为主函数变元，而$G_1, \cdots, G_n$ 是n 个不同的函数变元，并且对$i = 1, \cdots, n$，有

(1) $G_i \in W$ 是s_i 元函数变元;

(2) $x_{i1}, \cdots, x_{is_i}$ 是s_i 个不同的变元;

(3) t_i 是变元和函数变元分别来自$\{x_{i1}, \cdots, x_{is_i}\}$ 和$\{G_1, \cdots, G_n\}$ 的项。

§3. 递归程序的操作语义

我们将用计算序列算法来定义递归程序的操作语义，这个序列是通过简化和替换产生的。在给出它们的精确定义前，先对一类特殊的项和公式进行赋值，为了技术上的方便，规定在B 的解释$I = (D, I_0)$ 中，I_0 使得B 中F 的常元与D 中元素具有一一对应(这将在定义7 中用到)，从而F 中常元数目与D 中元素的数目相同[1]。

定义3(不含函数变元的项和公式的语义) 设I 是$B = (F, P)$ 的解释，$\sum$ 是指派集，对应于I，我们定义一个泛函，不妨仍记为I：它把每个不含函数变元的项t 映为一个函数$I(t) : \sum \to D$，而把每个不含函数变元的公式w 映为一个谓词$I(w) : \sum \to Bool$，其定义如下:

(1) $I(t)$ 的定义:

(a) 若$c \in F$ 是常元，则对$\sigma \in \sum$，$I(c)(\sigma) = I_0(c)$;

若$x \in V$ 是变元，则对$\sigma \in \sum$，$I(x)(\sigma) = \sigma(x)$;

(b) 若$t_1, \cdots, t_n$ 是不含函数变元的项，$f \in F$ 是n 元函数符号，则对$\sigma \in \sum$，$I(f(t_1, \cdots, t_n))(\sigma) = I_0(f)(I(t_1)(\sigma), \cdots, I(t_n)(\sigma))$;

(c) 若t_1、t_2 是不含函数变元的项，e 是不含函数变元的公式，则对$\sigma \in \sum$，

$$I(if\ e\ then\ t_1\ else\ t_2\ f_i)(\sigma) = \begin{cases} I(t_1)(\sigma) & 若I(e)(\sigma) = 真 \\ I(t_2)(\sigma) & 若I(e)(\sigma) = 假 \end{cases}$$

(2) $I(w)$ 的定义:

(a) 对$\sigma \in \sum$，$I(true)(\sigma) =$真，$I(false)(\sigma) =$假;

(b) 若t_1、t_2 是不含函数变元的项，则对$\sigma \in \sum$，

$$I((t_1 \equiv t_2))(\sigma) = \begin{cases} 真 & 若I(t_1)(\sigma) = I(t_2)(\sigma) \\ 假 & 若I(t_1)(\sigma) \neq I(t_2)(\sigma) \end{cases}$$

(c) 若$t_1, \cdots, t_n$ 是不含函数变元的项，$p \in P$ 是n 元谓词符号，则对$\sigma \in \sum$，$I(p(t_1, \cdots, t_n))(\sigma) = I_0(p)(I(t_1)(\sigma), \cdots, I(t_n)(\sigma))$;

(d) 若w 是不含函数变元的公式，则对$\sigma \in \sum$，

$$I((\neg w))(\sigma) = \begin{cases} 真 & 若I(w)(\sigma) = 假 \\ 假 & 若I(w)(\sigma) = 真 \end{cases}$$

(e) 若w_1、w_2 是不含函数变元的公式，则对$\sigma \in \sum$，

$$I((w_1 \wedge w_2))(\sigma) = \begin{cases} 真 & 若I(w_1)(\sigma) = 真且I(w_2)(\sigma) = 真 \\ 假 & 若I(w_1)(\sigma) = 假或I(w_2)(\sigma) = 假 \end{cases}$$

定义4 (简化模式) 设I 是B 的解释，则一个(相对于I 的) 简化模式是指一个满足下列条件之一的项或公式的有序对偶(t, u):

(1) t 呈$if\ a\ then\ t_1\ else\ t_2\ f_i$ 型，其中a 为命题常元，t_1、t_2 是项，并且当$a = true$ 时，$u = t_1$，当$a = false$ 时，$u = t_2$;

(2) t 不呈$if\ \cdots\ fi$ 型，而u 的长度短于t 的长度，并且t 和u 在I 下等价，即$I(t)(\sigma) = I(u)(\sigma)$ 对所有指派σ 均成立。

定义5 (简化关系) 设I 是B 的解释，v、v' 是项或公式，若v' 是某个简化模式(t, u) 作用于v 的结果，即将v 中某符号$t^{t_1,\cdots,t_n}_{x_1,\cdots,x_n}$ 换以$u^{t_1,\cdots,t_n}_{x_1,\cdots,x_n}$，其中$x_1,\cdots,x_n$ 是t 中出现的变元，$t_1,\cdots,t_n$ 是项，$t^{t_1,\cdots,t_n}_{x_1,\cdots,x_n}$ 表示将t 中的$x_1,\cdots,x_n$ 分别代以$t_1,\cdots,t_n$，则我们定义$v \to v'$，并称$\to$ 为(相对于I 的) 简化关系。

有关B 上递归程序S 的替换函数φ 的定义可参看[1] 而作，它是一个纯语法概念，今不赘述。

定义6 (转移关系) 设I 是B 的解释，S 是B 上的递归程序，若项t、t' 满足:

(1) t 至少含有一个函数变元;

(2) $\varphi(t) \xrightarrow{*} t'$，并且$t'$ 已被极大简化，其中φ 是S 的替换函数，$\xrightarrow{*}$ 表示简化关系$\to$ 的自反、传递闭包，则我们定义$t \Rightarrow t'$，并称$\Rightarrow$ 为(相对于I 和S 的) 转移关系。

定义7(计算序列) 设I 是B 的解释，S 是B 上的递归程序$G_j(x_{j1},\cdots,x_{js_j}) \Leftarrow t_j$, $j = 1,\cdots,n$，其中$G_k(1 \le k \le n)$ 是主函数变元，σ 是指派，则一个(相对于I、S 和σ 的) 计算序列是一个满足$t_0 = G_k(c_1,\cdots,c_{S_k})$(其中$I_0(c_j) = \sigma(x_j)$，$j = 1,\cdots,S_k$) 并且$t_i \Rightarrow t_{i+1}(i \ge 0)$ 的序列$t_0, t_1, \cdots$。

可以看出，计算序列总是由不含变元的项组成，当它为有穷时，其结束项不含任何函数变元，从而等价于某个常元c，由于它已被极大简化，因此它必然就是c。为了保证下面的语义刻划的合理性，只需证计算序列中的转移关系是一个函数，即其中的简化关系满足Church-Rosser 性质。

定理8 设I 是B 的解释，$\to$ 是相对于I 的简化关系，u, u_1, u_2 是不含变元的项或公式，并且$u \to u_1$，$u \to u_2$，则有项u'，使得$u_1 \to u'$，$u_2 \to u'$。

我们对不含变元的项或公式的结构进行归纳证明:

1. u 为项:

(a) u 为常元，结论显然成立;

(b) u 为$f(t_1,\cdots,t_s)$，其中$t_1,\cdots,t_s$ 是不含变元的项，

(i) 若u_1、u_2 中有一个是不经简化模式作用于某$t_k(1 \le k \le s)$ 而得，则由定义3、定义4 和u 不含变元的事实可证明，u, u_1, u_2 均等价于某个常元c，从而(u_1, c) 和(u_2, c) 均是简化模式，取$u' = c$ 即可;

(ii) 对$i = 1$、2, u_i 是由u 经简化模式作用于v_i 而得，其中$v_i \in \{t_1,\cdots,t_s\}$，分情形讨论:

I. $v_1 = v_2$，则由归纳假设知，若$v_1 \to v_1'$, $v_2 \to v_2'$，那么有项v，使得$v_1' \to v$, $v_2' \to v$，因此，若u' 是将u 中v_1 换以v 后所得结果，则显然有$u_1 \to u'$, $u_2 \to u'$;

II. $v_1 \ne v_2$，设u' 是将u_1 中v_2 换以v_2'(v_2' 是假设中的简化模式作用于v_2 的结果) 后所得的项，则易知$u_1 \to u'$，另一方面，u' 的长度短于u_2 的长度，u' 又是u_2 中v_1 换以v_1'(v_1' 是假设中的简化模式作用于v_1 的结果) 后所得的项，从而$u_2 \to u'$;

(c) u 为$G(t_1,\cdots,t_s)$，其中$t_1,\cdots,t_s$ 是不含变元的项，G 为s 元函数变元，则由定义3 知(b) 中情形(i) 在这里不成立，从而证明完全同于(b) 中情形(ii);

(d) u 为*if* e *then* t_1 *else* t_2 *fi*，其中e 是不含变元的公式，t_1、t_2 是不含变元的项，

(i) 若(b) 中(ii) 不成立，则$u_1 = u_2 \in \{t_1, t_2\}$，因为e 必为命题常元；

(ii) 若(b) 中(ii) 成立，则证明完全类似。

2. u 为公式的情形，限于篇幅，从略。

下面的定义完成了我们在谓词逻辑上对递归程序的操作语义的刻划工作。

定义9 (递归程序的操作语义) 设$I = (D, I_0)$ 是B 的解释，S 是B 上的递归程序，$\sum$ 为指派集，则S 相对于I 的语义函数$M_I(S) : \sum \to D$ 定义为：

$$M_I(S)(\sigma) = \begin{cases} I_0(c) & \text{若相对于}I, S\text{和}\sigma\text{的计算序列有穷且结束项为}c \\ \text{无定义} & \text{若相对于}I, S\text{和}\sigma\text{的计算序列无穷} \end{cases}$$

§4. 结束语

本文在谓词逻辑上建立了递归程序的操作语义，实际上为递归程序提供了一个标准的模型，它消除了原有的(带有ω 的) 非标准模型中转移关系未必是函数的缺陷，从而更能体现计算机执行程序时的确定性。关于这两个模型的等价性问题尚需进一步考察。此外，目前的涉及递归程序的验证方法一般是基于指称语义的，而流程式程序和while 程序的验证可以用归纳断言法，因为它们的操作语义均可建立于谓词逻辑上。因此一个十分自然的问题是：递归程序的验证是否也可用归纳断言法？有关这两个论题的探讨，我们期待在另外的文章中给予解答。

致谢：感谢沈百英教授的指教。

参考文献

[1] Loeckx, J. and Sieber, K., The Foundations of Program Verification, Wiley-Teubner Series in Computer Science, 2nd ed., 1987.

[2] Gordon, M. J. C., Thd Denotational Description of Programming Languages, Springer-Verlag, 1979.

《系统软件研讨会》征文通知

中国计算机学会软件专业委员会系统软件学组将于1992 年第四季度召开第四次学术研讨会，现将有关征文内容及注意事项通知如下：

一、征文内容：　操作系统、编译系统国产化工作的成果与经验；通信软件；计算机语言学，形式语言和语义理论；程序设计逻辑，软件开发形式化技术，软件自动生成技术；微机系统软件的开发及汉化工作；引进系统的消化、分析、开发与创新；大型软件的管理与维护；“八五”系统软件国产化工作展望。

二、征文要求及方法：　应征论文要反映本单位或个人实践或理论研究成果，学术上要有严谨的科学性和一定的实用价值。文章论点要明确，文字精练，数据可靠，图表清晰并附有摘要。会议征集未在全国性会议或国内外刊物上发表过的文章。征文一律不退稿，请作者自留底稿。截稿时间为1992 年2 月底。稿件请寄往：四川成都电子科技大学8010 研究室，陈勇，邮政编码：610054。会议地点：成都。