概率拟 Hoare 逻辑

吴新星[1] 胡国胜[1] 陈仪香[2]
(上海电子信息职业技术学院计算机应用系 上海 201411)[1]
(华东师范大学教育部软硬件协同设计技术与应用工程研究中心 上海 200062)[2]

摘 要 基于 Hoare 逻辑,给出了概率拟 Hoare 逻辑,用于刻画程序执行的正确度,定量地描述理论与程序(或软件)实际执行之间的差距,反映理论被程序实现的程度,从而量化理论上正确的程序在实际执行时出错的可能性,以及解释正确度很高的两个程序串行复合之后的整体正确度可能并不高等问题。

关键词 Hoare 逻辑,Hoare 三元组,正确度,概率测度
中图法分类号 TP3-0 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.4.036

Probabilistic Quasi-Hoare Logic

WU Xin-xing[1] HU Guo-sheng[1] CHEN Yi-xiang[2]
(Department of Computer, Shanghai Technical Institute of Electronics & Information, Shanghai 201411, China)[1]
(Soft/Hardware Co-design Engineering Research Center MoE, East China Normal University, Shanghai 200062, China)[2]

Abstract A Hoare logic-baesed probabilistic quasi-Hoare logic for program correctness assessment was presented. Probabilistic quasi-Hoare logic describes the difference between the theory idea and implementation, and reflects the degree of which the theory idea is implemented by the program in practice. Thus, it can formally explain the reasons of errors of programs which are theoretically correct and the low correctness degree of the sequential composition of two programs (components) which are both high in correctness degree, etc.

Keywords Hoare logic, Hoare triple, Correctness degree, Probability measure

1 引言

R. W. Floyd 于 1967 年发表的"给程序赋义(Assigning Meanings to Programs)"[1]提出了如何描述人们所关心的程序含义及如何去论证一个程序是否具有某种含义的数学方法——论证程序流程图的规则[2]。基于此工作,C. A. R. Hoare 修改并扩充了这些规则,在 1969 年的"计算机程序设计的公理基础(An Axiomatic Basis for Computer Programming)"一文中提出了一种形式化处理程序的方法——Hoare 逻辑[3-5],它不仅能够证明程序的性质,而且还能用于解释程序结构的含义。这是计算机科学家第一次采用公理系统来定义一类程序设计语言的语义,Hoare 的这种"以公理或是规则来表示程序结构的含义,并说明如何证明程序性质"[6]的方法被称为公理语义。

Hoare 逻辑是一种可以对程序正确性进行证明的形式化方法。如可以通过 Hoare 逻辑证明下面的程序 $x:=x+1$ 在理论上是正确的。

$$\{x\geqslant 0\}x:=x+1\{x\geqslant 1\} \tag{1}$$

但是,我们说实际程序的实现还需要借助于一定的硬件设备或物理条件,而在程序的执行过程中这些相关设备或条件未必能保证完全可靠。因此,如果将上面的程序(1)放到实际环境中执行,就有可能会发生出错。如,将 $x:=x+1$ 在计算机上执行,运行 $x:=32767+1$(x 的数据类型为 signed short int),我们发现由于数据溢出等原因,实际的执行结果可能为 -32767,是错误的。又如:

$$\{0\leqslant x\leqslant 1000\}x:=x^2\{0\leqslant x\leqslant 1000000\} \tag{2}$$

表 1 的程序 Square.c 100%程度上实现了上面的(2)吗?事实上,我们说一个程序的运行依赖于环境,很可能程序的运行在有些环境下是正确的,但在其它的环境就未必正确。根据不同的实际运行环境,如在 16 位、32 位或是 64 位计算机上,程序 Square.c 被实现的程度可能就会不一样。

表 1 程序 Square.c

```
int main(void)
    {
        int x;
        printf("Enter the number x \n");
        scanf("\%d",\&x);
        x=x*x;
        printf("\%d",x);
        return 0;
    }
```

再如下面的例子:

例 1 在 32 位计算机上的程序满足

到稿日期:2015-03-13 返修日期:2015-07-03

吴新星(1981—),男,博士,助理研究员,CCF 会员,主要研究方向为形式化方法、随机过程,E-mail:xinxingwu@yeah.net;胡国胜(1965—),男,博士,教授,主要研究方向为机器学习、负荷预测;陈仪香(1961—),男,博士,教授,主要研究方向为物联网、实时协同规范语言设计、程序语义模型、软件可信度量与评估理论。

$\{0\leqslant x\leqslant 192\}x:=x^2\{0\leqslant x\leqslant 32767\}$ (3)

将上面的(3)移植到64位计算机上,实现情况会怎么样?

另,如果使用C语言在计算机(数据类型是long double)上计算25!时,实际往往可能会因数据溢出等原因而出现问题,导致结果不正确。

出现这些情况的根本原因是Hoare逻辑是不考虑环境的。因此我们说,对具体的实际执行而言,Hoare逻辑是理想化的,它描述程序正确性的方法本质上是绝对的,在Hoare逻辑框架下,程序本身只有两种属性:正确或者错误。

在Hoare逻辑理论上证明是正确的程序,实际执行时却可能会出错;Hoare逻辑没有考虑环境等因素。鉴于此,基于经典Hoare逻辑,我们先研究、讨论如下的问题:

(1)若所给的条件比最弱前置还要弱

程序的执行局限于一定的物理设备和条件等,而这些相关的物理设备和条件未必完全可信。注意到这种情况,我们考虑如下的理论问题:

假定经典的Hoare三元组$\{A\}c\{B\}$,其中,A是$c\{B\}$的最弱前置,且有$A\Rightarrow A'$。如果以A'替换$\{A\}c\{B\}$中的A,那么程序就会出现虽然当前状态满足A,但是c未必能正确地执行的情况。此时,被修改后的"三元组"$\{A'\}c\{B\}$的语义如何来描述?

(2)若所给的条件比最强后置还要强

从程序的使用者(或用户)角度出发,程序的执行结果不满足其使用者的要求(出现这种现象有两种原因:一是程序执行的结果出错了;二是虽然程序执行没有出错,但是程序的这种功能是使用者所不需要的或是不要求的)。针对这种情况,有如下的理论问题:

假定经典的Hoare三元组$\{A\}c\{B\}$,B是$\{A\}c$的最强后置,有$B'\Rightarrow B$。如果以B'替换$\{A\}c\{B\}$中的B,那么程序c即使是成功执行并且终止,返回状态也可能不满足B'。此时,被修改后的"三元组"$\{A\}c\{B'\}$的语义如何来描述?

基于前面的问题和讨论,本文主要基于Hoare逻辑提出了一种概率拟Hoare逻辑,用于刻画程序执行的正确度,将经典Hoare逻辑框架下程序属性的两种取值状态(0和1)推广到整个[0,1]区间。具体的工作如下:

· 首先,给出α正确蕴含、α正确包含和程序诱导分布等定义和命题,并进一步地给出概率拟Hoare三元组、程序正确度规则等定义。

· 然后,在Hoare逻辑的基础上,针对IMP语言[6]给出一种概率拟Hoare逻辑用于刻画程序的量化性质——执行正确度,从而描述程序实际执行与理论之间的差距,反映理论被实际程序实现的程度,进而为"程序往往在理论上是正确的,实际执行却有可能会出错"和"程序单独存在时,正确执行概率很高,但一旦做了顺序复合后正确执行概率并不高"(即局部的正确执行未必导致全局正确执行)等现象给出理论依据。

在本文的讨论中,若非特别说明,有如下假设。

(i)Σ表示可数状态集。$Assn$表示扩充的布尔表达式集。Com表示命令集。Loc表示存储单元的集合。$\bar{N}$表示正整数、负整数和零的集合。$Index$表示任意指标集。I_c是程序c的关系语义或说明。$\{A\}c\{B\}$表示部分正确性断言。$[A]c[B]$表示完全正确性断言。

(ii)称$Prob$是$A(\subseteq\Sigma)$上的概率测度,实际上是指$Prob$是可测空间$(A,\sigma(A))$上的概率测度,而所构成的三元组$(A,\sigma(A),Prob)$称为概率测度空间。在不引起混淆的情况下,A上的概率测度$Prob_A$(或$Prob_{S_A}$)简记为$Prob$。其中,$\sigma(A)$表示由状态空间A的一切子集所生成的σ-代数。

(iii)如果$Prob$是$A(\subseteq\Sigma)$上的一个概率测度,那么有$\forall\sigma\in A, Prob(\{\sigma\})>0$。

(iv)程序运行环境为Intel(R)Core(TM)2 Duo CPU E8400@3.00GHz,2.99GHz,2.00GB,Windows XP SP3,采用Win-TC编译器/Eclipse。

(v)另外,由于考虑到程序实际运行中数据溢出等物理环境问题,我们认为指称函数是概率意义下成立的。为了区别文献[6]中的符号,分别记$\mathscr{A}$、$\mathscr{B}$和$\mathscr{C}$为$\mathscr{A}_{env}$、$\mathscr{B}_{env}$和$\mathscr{C}_{env}$。

2 预备知识

本节将给出α正确蕴含、α正确包含、程序诱导分布以及程序正确度α规则等的定义和命题。

下面首先给出一些后面讨论要用到的约定。

设$A\in Assn$,所有满足A的状态的集合记为

$S_A=\{\sigma\in\Sigma\mid\sigma\vdash A\}$[1)]

定义1(状态)[7] 一个状态是一个以Loc作为定义域、集合$\bar{N}$作为值域的函数,即$X\in Loc$,$\sigma(X)$表示在状态σ下存储单元X的求值结果。

定义2(程序说明)[7] 程序的关系语义或说明(用I_c表示)是状态空间Σ上的二元关系$\langle\sigma,\sigma'\rangle\in I_c$,当且仅当存在一个可执行的程序$c$,它以初始状态$\sigma$开始,以终结状态$\sigma'$结束。

定义3(定义域)[7] 给定一个关系$R\subset\Sigma\times\Sigma$,$R$的定义域表示为$Dom.R$,即

$Dom.R=\{x\mid x\in\Sigma\wedge\exists x'(x'\in\Sigma\wedge\langle x,x'\rangle\in R)\}$

定义4(值域) 给定一个关系$R\subset\Sigma\times\Sigma$,$R$的值域表示为$Range.R$,即

$Range.R=\{x'\mid x'\in\Sigma\wedge\exists x(x\in\Sigma\wedge\langle x,x'\rangle\in R)\}$

定义5(定义域限制)[7] 给定一个关系$R\subset\Sigma\times\Sigma$和一个集合$Y\subset\Sigma$,关系$R$的定义域由$Y$限制,记为$Y\upharpoonright R$,即

$Y\upharpoonright R=\{\langle x,x'\rangle\mid x\in Y\wedge\langle x,x'\rangle\in R\}$

定义6(值域限制)[7] 给定一个关系$R\subset\Sigma\times\Sigma$和一个集合$Z\subset\Sigma$,关系$R$的值域由$Z$限制,记为$R\downharpoonright Z$,即

$R\downharpoonright Z=\{\langle x,x'\rangle\mid x'\in Z\wedge\langle x,x'\rangle\in R\}$

定义7(σ-代数)[8] 称集合Ω的子集类$\mathfrak{A}$为Ω中的σ-代数,如果它满足下列条件:

(a)$\Omega\in\mathfrak{A}$;

(b)若$A\in\mathfrak{A}$,则$A^c\in\mathfrak{A}$;

(c)若$A_n\in\mathfrak{A}$,$n=1,2,\cdots$,则$\bigcup_{n=1}^{+\infty}A_n\in\mathfrak{A}$。

定义8[8,9] 设$\mathfrak{A}$为Ω的子集类,$\emptyset\in\mathfrak{A}$。集函数$\mu:\mathfrak{A}\to\bar{R}$,如果

(a)非负性:$\forall A\in\mathfrak{A},\mu(A)\geqslant\mu(\emptyset)=0$;

(b)σ-可加性:设$A_n(n=1,2,\cdots)$两两不交(即$A_i\cap A_j=$

1) "$\vdash$"代表满足或成立的意思,如"$\sigma\vdash A$"表示状态σ满足布尔表达式A

$\emptyset, i\neq j$)，且$\sum_{n=1}^{+\infty}A_n\in\mathfrak{A}$，则

$$\mu(\sum_{n=1}^{+\infty}A_n)=\sum_{n=1}^{+\infty}\mu(A_n)$$

则称μ为测度。如果$\forall A\in\mathfrak{A}$，有$\mu(A)<+\infty$，则称μ为有限测度；如果存在一列$A_i\in\mathfrak{A}, n\geqslant 1$，使得$\bigcup_{n=1}^{+\infty}A_i=\Omega$且$\mu(A_n)<+\infty, n\geqslant 1$，则称$\mu$为$\sigma$-有限测度。具有性质$\mu(\Omega)=1$的测度称为正规测度，亦称概率测度，一般用$Prob$表示。

注 1[9]　从上面测度的定义可知，测度的自然定义域应当是σ-代数。

定义 9[10]　设$\mathfrak{A}$为Ω上一σ-代数，称序偶$(\Omega,\mathfrak{A})$为可测空间，$\mathfrak{A}$中的元素称为$\mathfrak{A}$-可测集。设μ为可测空间$(\Omega,\mathfrak{A})$上的测度，则称三元组$(\Omega,\mathfrak{A},\mu)$为测度空间。若$\mu$是有限测度，则称$(\Omega,\mathfrak{A},\mu)$为有限测度空间；若$\mu$是概率测度，则称$(\Omega,\mathfrak{A},\mu)$为概率测度空间。

定义 10[10]　设$(\Omega,\mathfrak{A})$为可测空间，E是Ω的一个子集，f是定义在E上的有限实函数。如果对于一切实数c，集合$E(c\leqslant f)$都是$E(c\leqslant f)$上的可测集(即$E(c\leqslant f)\in\mathfrak{A}$)，那么则称$f$是$E$上关于$(\Omega,\mathfrak{A})$的可测函数，简称是$E$上可测函数。

定义 11[10]　设$(\Omega,\mathfrak{A},\mu)$为一测度空间。若$A\in\mathfrak{A}$，且$\mu(A)=0$，则称$A$为$\mu$-零测集。

定理 1(积分变换定理)[9]　设f是测度空间$(\Omega,\mathfrak{A},\mu)$到可测空间$(\Pi,\mathfrak{A}')$上的可测映射，$\mu_f$为由$f$导出的测度。又设$g$是$(\Pi,\mathfrak{A}')$到$(R,\mathscr{B}(R))$的可测函数，则$\forall B\in\mathfrak{A}'$，

$$\int_{f^{-1}(B)}g(f)\mathrm{d}\mu=\int_B g\,\mathrm{d}\mu_f$$

上式的意义是若一方有意义，则另一方也有意义，且两者相等。

命题 1[11]　$\mathfrak{A}$上的任何测度μ的原子是一个使得$\mu(\{x\})>0$的单元。σ-有限测度的原子个数是可数的。

在文献[12,13]等中，讨论了布尔表达式(公式)真度的概念。在这里，也引入类似的概念。

定义 12(α正确蕴含)　我们说A_1以正确度α蕴含A_2，如果有

$$\int_{S_{A_1}}(\|A_1\wedge A_2\|)(\omega)Prob_{A_1}(\mathrm{d}\omega)=\alpha \tag{4}$$

记为$A_1\overset{\alpha}{\infty}A_2$。其中，$A_1, A_2\in Assn$，$Prob_{A_1}$是$S_{A_1}$上的概率测度或分布。若$A_1\wedge A_2$为真，那么$\|A_1\wedge A_2\|=1$，否则$\|A_1\wedge A_2\|=0$。

注 2：(i)在式(4)中，若$A_1\equiv$false，则$\alpha=0$；(ii)在式(4)中，若$A_1\equiv$true，称α为A_2的正确度(类似于文献[13]的真度定义)；(iii)在式(4)中，$\alpha=1$蕴含$A_1\Rightarrow A_2$，但是，注意到false$\Rightarrow A_2$，因此，若$A_1\Rightarrow A_2$，未必会有$A_1\overset{1}{\infty}A_2$；(iv)$A_1\overset{\alpha}{\infty}A_2$表示$A_1$以大于等于$\alpha$的正确度蕴含$A_2$，于是有$\int_{S_{A_1}}(\|A_1\wedge A_2\|)(\omega)Prob_{A_1}(\mathrm{d}\omega)\geqslant\alpha$；(v)$A_1\overset{\alpha}{\infty}A_2$表示$A_1\overset{\alpha}{\infty}A_2$且$A_2\overset{\alpha}{\infty}A_1$。

定义 13(α正确包含)　我们说S_{A_1}以正确度α包含于S_{A_2}，如果有

$$\int_{S_{A_1}}\chi(S_{A_1}\cap S_{A_2})(\omega)Prob_{A_1}(\mathrm{d}\omega)=\alpha \tag{5}$$

记为$S_{A_1}\sqsubseteq_\alpha S_{A_2}$。其中，$A_1, A_2\in Assn$，$Prob_{A_1}$是$S_{A_1}$上的概率测度或分布。

注 3：(i)在式(5)中，若$S_{A_1}=\emptyset$，则$\alpha=0$；(ii)在式(5)中，$\alpha=1$蕴含$S_{A_1}\subseteq S_{A_2}$。但是，注意到$\emptyset\subseteq S_{A_2}$，因此，若$S_{A_1}\subseteq S_{A_2}$，未必会有$S_{A_1}\sqsubseteq_1 S_{A_2}$；(iii)若$\int_{S_{A_1}}\chi(S_{A_1}\cap S_{A_2})(\omega)Prob_{A_1}(\mathrm{d}\omega)\geqslant\alpha$，则记为$S_{A_1}\sqsubseteq_{\alpha\leqslant}S_{A_2}$。

命题 2(程序诱导分布(或测度))　$[A]c[B]$是Hoare三元组。B是$[A]c$的最强后置，$Prob$[1]是S_A上的概率测度，$Prob_c$是由c导出的S_B上的概率测度。则有$\forall F\in\sigma(S_B)$，

$$Prob_c(F)=Prob(Dom.(I_c\downarrow F))$$

其中，$\sigma(S_B)$表示由S_B的一切子集所成的σ-代数。

证明：首先，由假设条件显然有

$$Dom.(I_c\downarrow S_B)=S_A$$

又注意到程序c可以看成是一个从测度空间$(S_A,\sigma(S_A),Prob)$到可测空间$(S_B,\sigma(S_B))$的可测映射。于是，由积分变换定理可知结论成立。

命题 3　$A_1\overset{\alpha\leqslant}{\infty}A_2\Leftrightarrow S_{A_1}\sqsubseteq_{\alpha\leqslant}S_{A_2}$。

证明：由定义 12、定义 13、注 2 和注 3 可知

$$A_1\overset{\alpha\leqslant}{\infty}A_2\Leftrightarrow Prob_{A_1}(S_{A_1}\cap S_{A_2})\geqslant\alpha \tag{6}$$

及

$$S_{A_1}\sqsubseteq_{\alpha\leqslant}S_{A_2}\Leftrightarrow Prob_{A_1}(S_{A_1}\cap S_{A_2})\geqslant\alpha \tag{7}$$

由式(6)和式(7)易知结论成立。

3　概率拟 Hoare 三元组

在这一节中将给出一种概率拟 Hoare 三元组，用于量化程序执行的正确性，刻画程序的正确度(理论被实际程序实现的程度)；在本节的最后给出相关的例子对我们提出的概率拟 Hoare 三元组进行说明。

下面给出概率拟 Hoare 三元组的定义。这里主要介绍了概率拟 Hoare 三元组及其 6 种特殊情况，我们统称这些为概率拟 Hoare 三元组。

定义 14(概率拟 Hoare 三元组)　设$A', B'\in Assn, c\in Com.$ $\{A'\}_{[\alpha_1,\alpha_2]}(c)_{[\beta_1,\beta_2]}\{B'\}$为概率拟 Hoare 三元组。其中，$0\leqslant\alpha_1\leqslant\alpha_2\leqslant 1, 0\leqslant\beta_1\leqslant\beta_2\leqslant 1$，$A'$是程序$c$的概率拟前置条件，$B'$是程序$c$的概率拟后置条件。其意思是说：

若程序c的当前状态满足A'，那么程序段c以$\alpha\in[\alpha_1,\alpha_2]$的概率执行并能正常终止，且返回值以$\beta\in[\beta_1,\beta_2]$的概率满足$B'$[2]。

注 4：特别地，

(i)$[\alpha_1,\alpha_2]$1-拟 Hoare 三元组

$$\{A'\}_{[\alpha_1,\alpha_2]}(c)_1\{B'\}$$

其中，$0\leqslant\alpha_1\leqslant\alpha_2\leqslant 1$，$A'$是程序$c$的$[\alpha_1,\alpha_2]$-拟前置条件，$B'$是程序$c$的 1-拟后置条件。其意思是说：

程序c的当前状态满足A'，那么程序段c以$\alpha\in[\alpha_1,\alpha_2]$的概率执行并能正常终止，且返回值以 1 的概率满足B'。

---

1) $Prob$关于c独立

2) 为了避免与c下标的混淆，我们将$\{A'\}_{[\alpha_1,\alpha_2]}c_{[\beta_1,\beta_2]}\{B'\}$写成$\{A'\}_{[\alpha_1,\alpha_2]}(c)_{[\beta_1,\beta_2]}\{B'\}$

(ii)$1[\beta_1,\beta_2]$-拟 Hoare 三元组

$\{A'\}_1(c)_{[\beta_1,\beta_2]}\{B'\}$

其中,A'是程序 c 的 1-拟前置条件,B'是程序 c 的$[\beta_1,\beta_2]$-拟后置条件。其意思是说:

程序 c 的当前状态满足A',那么程序段 c 以 1 的概率执行并能正常终止,且返回值以$\beta\in[\beta_1,\beta_2]$的概率满足 B'。

(iii)$\alpha\beta$-拟 Hoare 三元组

$\{A'\}_\alpha(c)_\beta\{B'\}$

其中,A'是程序 c 的 α-拟前置条件,B'是程序 c 的 β-拟后置条件。其意思是说:

程序 A 的当前状态满足A',那么程序段 c 以 α 的概率执行并能正常终止,且返回值以 β 的概率满足 B'。

(iv)$\alpha 1$-拟 Hoare 三元组

$\{A'\}_\alpha(c)_1\{B'\}$

其中,A'是程序 c 的 α-拟前置条件,B'是程序 c 的 1-拟后置条件。其意思是说:

程序 c 的当前状态满足A',那么程序段 c 以 α 的概率执行并能正常终止,且返回值以 1 的概率满足 B'。

(v)1β-拟 Hoare 三元组

$\{A'\}_1(c)_\beta\{B'\}$

其中,A'是程序 c 的 1-拟前置条件,B'是程序 c 的 β-拟后置条件。其意思是说:

程序 c 的当前状态满足A'。那么程序段 c 以 1 的概率执行并能正常终止,且返回值以 β 的概率满足 B'。

(vi)$\{A'\}_{\alpha\leqslant}(c)_{\leqslant\beta}(B')$表示程序 c 的当前状态满足A',那么程序段 c 以大于等于 α 的概率执行并能正常终止,且返回值以小于等于 β 的概率满足 B'。

定义 15(程序正确度(或实现度)α 规则) 概率拟 Hoare 三元组正确度 α 规则有如下说明:

$$I[\{A'\}_{[\alpha_1,\alpha_2]}(c)_{[\beta_1,\beta_2]}\{B'\}]=(S_{A'}\sqsubseteq_{[\alpha_1,\alpha_2]}Dom.\ I_c)\wedge(Range.(S_{A'}\uparrow I_c)\sqsubseteq_{[\beta_1,\beta_2]}S_{B'}) \tag{8}$$

其中,$\alpha=\tau_1\times\tau_2$,$\tau_1\in[\alpha_1,\alpha_2]$,$\tau_2\in[\beta_1,\beta_2]$,$0\leqslant\alpha_1\leqslant\alpha_2\leqslant1$,$0\leqslant\beta_1\leqslant\beta_2\leqslant1$。

注 5:这里,我们只对注 4(i)和(vi)的正确度 α 规则作如下说明:

(1) $I[\{A'\}_{[\alpha_1,\alpha_2]}(c)_1\{B'\}]=(S_{A'}\sqsubseteq_{[\alpha_1,\alpha_2]}Dom.\ I_c)\wedge(Range.(S_{A'}\uparrow I_c)\sqsubseteq_1 R_{B'})$,其中,$\alpha\in[\alpha_1,\alpha_2]$。

(2) $I[\{A'\}_{\alpha\leqslant}(c)_{\leqslant\beta}\{B'\}]=(S_{A'}\sqsubseteq_{\alpha\leqslant}Dom.\ I_c)\wedge(Range.(S_{A'}\uparrow I_c)\sqsubseteq_{\leqslant\beta}R_{B'})$,其中,$\alpha=\tau_1\times\tau_2$,$\tau_1\in[\alpha,1]$,$\tau_2\in[0,\beta]$。

注 6:下面对定义 15 程序正确度(或称实现度)α 规则的合理性进行解释。

设 $Prob_c$ 是由 c 导出的 $Range.(S_{A'}\uparrow I_c)$上的概率测度,$Prob_{Dom.\ I_c\cap S_{A'}}$是$Dom.\ I_c\cap S_{A'}$上的概率测度,$Prob_{S_{A'}}$是 $S_{A'}$ 上的概率测度。

由命题 2 可以知道,$\forall F\sqsubseteq Range.(S_{A'}\uparrow I_c)$

$$Prob_{Dom.\ I_c\cap S_{A'}}(Dom.(I_c\downarrow F))=Prob_c(F) \tag{9}$$

再由图 1 可以知道:

$$Prob_{Dom.\ I_c\cap S_{A'}}(Dom.(I_c\downarrow F))=\frac{Prob_{S_{A'}}(Dom.(I_c\downarrow F))}{Prob_{S_{A'}}(Dom.\ I_c\cap S_{A'})} \tag{10}$$

$$Prob_c((Range.(S_{A'}\uparrow I_c))\cap S_{B'}=\tau_2 \tag{11}$$

$$Prob_{S_{A'}}(Dom.\ I_c\cap S_{A'})=\tau_1 \tag{12}$$

取 $F=(Range.(S_{A'}\uparrow I_c))\cap S_{B'}$,结合式(9)—式(12)便有

$$\begin{aligned}&Prob_{S_{A'}}(Dom.(I_c\downarrow((Range.(S_{A'}\uparrow I_c))\cap S_{B'})))\\&=Prob_{S_{A'}}(Dom.(I_c\downarrow F))\\&=Prob_{S_{A'}}(Dom.\ I_c\cap S_{A'})\times Prob_{Dom.\ I_c\cap S_{A'}}(Dom.(I_c\downarrow F))\\&=Prob_{S_{A'}}(Dom.\ I_c\cap S_{A'})\times Prob_c(F)\\&=\tau_1\times\tau_2\end{aligned}$$

从而说明了在定义 15 中所给程序正确度(或实现度)α 规则的合理性。

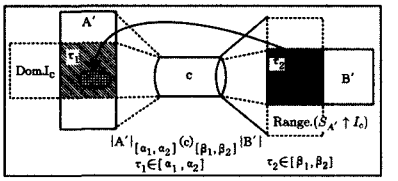

图 1 程序正确度(或称实现度)α 规则的合理性分析

下面给出相关例子对本节提出的概率拟 Hoare 三元组进行说明。为计算简单,此处假设定义 14 中所涉及的概率分布 $Prob$ 为均匀分布。

例 2 拟前置条件$\{x|x\in\{0,1,2,3,4\}\}$,拟后置条件$\{x|x\in\{1,\frac{1}{2}\}\}$,程序如表 2 所列。

表 2 程序 MathDivision.java

```
public class MathDivision {
    public int division(int x){
        int result;
        result=1/x;
        return result;
    }
}
```

注意到,对于整数除法计算,被除数为 0 时,程序不能正常执行(抛出异常),所以程序 MathDivision.java 的当前状态满足$\{x|x\in\{0,1,2,3,4\}\}$,程序以$\frac{4}{5}$的概率执行并能正常终止,且返回值以$\frac{1}{2}$的概率满足$\{x|x\in\{1,\frac{1}{2}\}\}$。因此,可以用本节所提出的概率拟 Hoare 三元组描述如下:

$$\{x|x\in\{0,1,2,3,4\}\}_{\frac{4}{5}}(x:=\frac{1}{x})_{\frac{1}{2}}\{x|x\in\{1,\frac{1}{2}\}\}$$

例 3 拟前置条件$\{x|x\in\{\text{“1M”},2,3,4\}\}$,拟后置条件$\{x|x\in\{4,9,16\}\}$,程序如表 3 所列。

表 3 程序 MathSquare.java

```
public class MathSquare {
    public int division(String x){
        int result;
        int y= Integer.valueOf(x)
            result=y * y;
            return result;
    }
}
```

注意到,当输入为“1M”时,程序不能正常执行(抛出异常),所以程序 MathSquare.java 的当前状态满足$\{x|x\in\{\text{“1M”},2,3,4\}\}$,程序以$\frac{3}{4}$的概率执行并能正常终止,且返回

值以 1 的概率满足$\{x|x\in\{4,9,16\}\}$。因此，可以用本节所提出的概率拟 Hoare 三元组描述如下：

$\{x|x\in\{\text{“1M”},2,3,4\}\}_{\frac{3}{4}}(x:=x*x)_{\frac{1}{2}}\{x|x\in\{4,9,16\}\}$

4 概率拟 Hoare 规则

在第 3 节中给出了概率拟 Hoare 三元组的定义，并且给出了程序正确度 α 规则。在这一节中将针对 IMP 语言，基于 Hoare 逻辑给出概率拟 Hoare 规则。这里，概率拟 Hoare 规则按 $\alpha\beta$-拟 Hoare 三元组给出；并且将借助所给的 $\alpha\beta$-拟 Hoare 规则说明和讨论：为何两个具有高正确度的程序串行之后整体正确度可能并不高，以及如何构建具有高正确度的串行程序等等问题。

• 拟 Skip 规则

我们说 Skip 不改变状态，在它执行之前的状态下断言为真，那么在它执行之后的断言显然仍为真。故有$\{A\}_1 Skip_1\{A\}$。

• 拟赋值规则

赋值语句执行的结果是将程序中变量 X 的值变成执行该语句前表达式 a 的值。注意到式(1)，由于局限于一定的硬件设备和物理条件等，有$\{A[a/X]\}_a X:=\alpha_\beta\{A\}$，$\alpha,\beta\in[0,1]$。

• 拟顺序规则

顺序语句 $c_0;c_1$ 的执行是说使用执行该语句前程序变量的值，先执行 c_0，然后使用执行 c_0 正确终止后的程序变量的结果值再执行 c_1，c_1 的执行后的结果值就是整个顺序语句终止后的结果值。并且，我们注意到在实际的顺序语句或是构件执行过程中，很可能是 $\exists\sigma'\in Range.(I_{c0})$，但是 $\sigma'\notin Dom.(I_{c1})$，故有

$$\frac{\{A\}_{\alpha_1}(c_0)_{\beta_1}\{D_1\}\{D_2\}_{\alpha_2}\{c_1\}_{\beta_2}\{B\}D_1\overset{\alpha_0}{\propto}D_2}{\{A\}_a(c_0;c_1)_\beta\{B\}}$$

图 2 是拟顺序规则的前提图解。

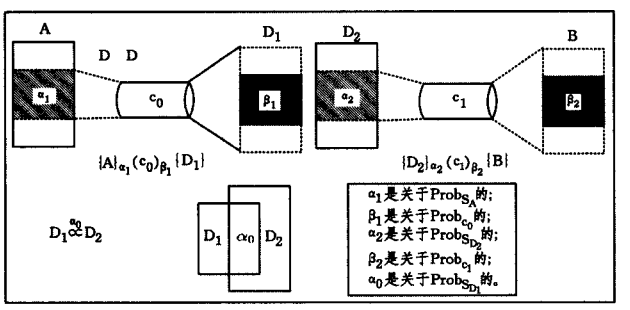

图 2　拟顺序规则前提：$\{A\}_{\alpha_1}(c_0)_{\beta_1}\{D_1\}\{D_2\}_{\alpha_2}\{c_1\}_{\beta_2}\{B\}D_1\overset{\alpha_0}{\propto}D_2$

分析：

根据图 3，此时 $c_0;c_1$ 的输入依赖于 c_0 的输入。

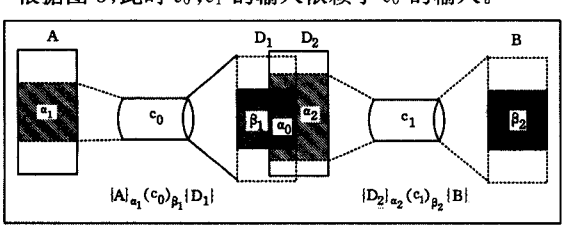

图 3　拟顺序规则分析

当 $Range.(S_A\uparrow I_{c_0})\overset{\alpha_0\leqslant}{\propto}Dom.I_{c_1}\cap S_{D_2}$ 时，有 $c_0;c_1$ 的当前状态满足 A。那么程序段 $c_0;c_1$ 以 $\alpha\in[\alpha_1\wedge\alpha_0,\alpha_1]$ 的概率执行并能正常终止，且返回值以小于等于 $\beta\in[\alpha_0\cdot\alpha_2,1]$ 的概率满足 B。

当 $Range.(S_A\uparrow I_{c_0})\overset{\alpha_0>}{\propto}Dom.I_{c_1}\cap S_{D_2}$ 时，有 $c_0;c_1$ 的当前状态满足 A。那么程序段 $c_0;c_1$ 以 $\alpha\in[0,\alpha_1]$ 的概率执行并能正常终止，且返回值以小于等于 $\beta\in[0,1]$ 的概率满足 B。

注 7：特别地，由拟顺序规则可以得到

$$\frac{\{A\}_1(c_0)_1\{D_1\}\{D_2\}_1(c_1)_1\{B\}D_1\overset{\alpha_0}{\propto}D_2}{\{A\}_{[\alpha_0,1]}(c_0;c_1)_{[\alpha_0,1]}\{B\}}\tag{13}$$

上面的式(13)揭示了：

• 一个程序(构件)单独存在时，它的正确性很高(甚至完全正确)，但是一旦做了复合，组成了一个软件或系统时，这个软件或系统的正确性可能并不高；

• 如何构建具有高正确度的串行程序：由式(13)可知，要使串行进程的正确性尽可能高，就要使得 α_0 尽可能地大。

• 拟选择规则

$$\frac{\{A\wedge b\}_{\alpha_1}(c_0)_{\beta_1}\{B\}\{A\wedge\neg b\}_{\alpha_2}(c_1)_{\beta_2}\{B\}}{\{A\}_{[0,(\alpha_1\vee\alpha_2)]}(\text{if } b \text{ then } c_0 \text{ else } c_1)_{[(\beta_1\wedge\beta_2),1]}\{B\}}$$

• 拟循环规则

$$\frac{\{A\wedge b\}_{\alpha_1}(c)_{\beta_1}\{A\}}{\{A\}_{[0,\alpha_1]}(\text{while } b \text{ do } c)_{\beta_1}\{A\wedge\neg b\}}$$

注 8：b 从(拟)前置条件到程序中布尔条件的过程中，由于涉及程序实际的执行，因此可能会遇到内存溢出等问题，而使程序正确性降低。

• 拟推理规则 1

$$\frac{\{A\}_{\alpha_1}(c)_{\beta_1}\{B'\}B'\overset{\alpha_0}{\propto}B}{\{A\}_{\alpha_1}(c)_{[0,1]}\{B\}}$$

注 9：(i)$\dfrac{\{A\}_1(c)_1\{B'\}B'\overset{\alpha_0}{\propto}B}{\{A\}_1(c)_{[0,1]}\{B\}}$

(ii)$\dfrac{\{A\}_1(c)_1\{B'\}B'\overset{\alpha_0}{\propto}B,B'\text{是}\{A\}c\text{的最强后置}}{\{A\}_1(c)_{\alpha_0}\{B\}}$

• 拟推理规则 2

$$\frac{A\overset{\alpha_0}{\propto}A'\{A'\}_{\alpha_1}(c)_{\beta_1}\{B\}}{\{A\}_{[0,1]}(c)_{[0,1]}\{B\}}$$

注 10：(i)$\dfrac{A\overset{\alpha_0}{\propto}A'\{A'\}_1(c)_1\{B\}}{\{A\}_{[\alpha_0,1]}(c)_{[0,1]}\{B\}}$

(ii)$\dfrac{A\overset{\alpha_0}{\propto}A'\{A'\}_1(c)_1\{B\},A'\text{是}c\{B\}\text{的最弱前置}}{\{A\}_{[\alpha_0,1]}(c)_{[0,1]}\{B\}}$

如果将 A'、B 看作是需求，(i)、(ii)说明了程序 c 虽然不能完全实现需求，但是程序 c 可能还有一些需求之外的功能。

结束语　注意到 Hoare 逻辑理论上证明是正确的程序，实际执行时却可能会出错等，且为了对此现象进行刻画，本文基于经典 Hoare 逻辑提出了一种概率拟 Hoare 逻辑，以用于量化程序的正确执行情况，度量程序实际执行与理论之间的差距，反映理论被实际程序实现的程度。在接下来的工作中，我们会进一步考虑如何将本文所提出的概率拟 Hoare 逻辑理论应用到软件工程的实际中，并基于该理论开发相应的工具。

(下转第 191 页)

sues and their research progress in multicore software [J]. ACTA Electronica Sinica, 2010, 38(9): 2140-2146(in Chinese)
杨际祥,谭国真,王荣生. 多核软件的几个关键问题及其研究进展[J]. 电子学报, 2010, 38(9): 2140-2146

[2] Karl-Filip F. Wool-A work stealing library [J]. ACM SIGARCH Computer Architecture News, 2009, 36(5): 93-100

[3] Vikranth B, Rajeev W, Raghavendra R C, et al. Topology Aware Task stealing for On-Chip NUMA Multi-Core Processors[J]. Procedia Computer Science, 2013, 18(1): 379-388

[4] Kim W, Voss M. Multicore desktop programming with Intel Threading Building Blocks [J]. IEEE Software, 2011, 28(1): 23-31

[5] Koutny T. Experience with Lamport Clock Ordered Events with Intel Threading Building Blocks in a Glucose-Level Prediction Software[C]// Proceedings of International Work-Conference on Bioinformatics and Biomedical Engineering (IWBBIO'14). Washington: IEEE CS Press, 2014: 515-526

[6] Navarro A, Asenjo R, Corbera F, et al. A case study of different task implementations for multioutput stages in non-trivial parallel pipeline applications [J]. Parallel Computing, 2014, 48(8): 374-393

[7] Leiserson C E. The Cilk++ concurrency platform[J]. The Journal of Supercomputing, 2010, 51(3): 244-257

[8] Acar U A, Charguéraud A, Rainey M. Scheduling parallel programs by work stealing with private deques [J]. ACM SIGPLAN Notices, 2013, 48(8): 219-228

[9] Singler J, Sanders P, Putze F. MCSTL: the multi-core standard template library[C]// Proceedings of 13th International Euro-Par Conference (Euro-Par'07). Heidelberg: Springer-Verlag, 2007: 682-694

[10] OpenMP home page. The OpenMP API specification for parallel programming [OL]. (2015-2-3) http://www.openmp.org

[11] Lea D. A Java fork / Join framework [C]// Proceedings of the ACM 2000 conference on Java Grande. New York: ACM Press, 2000: 36-43

[12] Tardieu O, Wang H C, Lin H B. A work-stealing scheduler for X10's task parallelism with suspension [J]. ACM SIGPLAN Notices, 2012, 47(8): 267-276

[13] Leijen D, Schulte W, Burckhardt S. The design of a task parallel library[C]// Proceeding of the 24th ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA'09). New York: ACM Press, 2009: 227-241

[14] Qiu J, Scott B, Seung-Hee B, et al. Performance of Windows Multicore Systems on Threading and MPI[C]// Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid'10). Washington: IEEE CS Press, 2010: 814-819

[15] Mackie R I. Dynamic analysis of structures on multicore computers—Achieving efficiency through object oriented design[J]. Advances in Engineering Software, 2013, 66(12): 3-9

(上接第 181 页)

参 考 文 献

[1] Floyd R W. Assigning Meanings to Programs[C]// Schwartz J T, ed. Proceedings of Symposium on Applied Mathematics. 1967: 19-32

[2] Zhou Chao-chen. Introduction to Formal Semantics[M]. Changsha: Hunan Science and Technology Press, 1985(in Chinese)
周巢尘. 形式语义学引论[M]. 长沙:湖南科学技术出版社, 1985

[3] Hoare C A R. An Axiomatic Basis for Computer Programming [J]. Communications of The ACM, 1969, 12(10): 576-580, 583

[4] Apt K R. Ten Years of Hoare's Logic: A Survey Part-I[J]. ACM Transactions on Programming Languages and Systems, 1981, 3(4): 431-483

[5] Jones C B, Roscoe A W, Wood K R, et al. Reflections on the Work of C. A. R. Hoare[M]. Springer-Verlag, 2010

[6] Winskel G. The Formal Semantics of Programming Languages: An Introduction[M]. MIT Press, 1993

[7] Wang Zhi-jian, Fei Yu-kui, Lou Yuan-qing. Software Component Technology and Its Application[M]. Beijing: Science Press, 2005 (in Chinese)
王志坚,费玉奎,娄渊清. 软件构件技术及其应用[M]. 北京:科学出版社, 2005

[8] Yan Shi-jian, Wang Jun-xiang, Liu Xiu-ying. A Foundation Course for Probability Theory (Second Edition)[M]. Beijing: Science Press, 2009(in Chinese)
严士健,王隽骧,刘秀英. 概率论基础(第二版)[M]. 北京:科学出版社, 2009

[9] Ding Wan-ding. A Summary Measure Theory[M]. Hefei: Anhui People's Publishing House, 2005(in Chinese)
丁万鼎. 测度论概要[M]. 合肥:安徽人民出版社, 2005

[10] Yan Jia-an. Measure Theory Handout (Second Edition)[M]. Beijing: Science Press, 2004(in Chinese)
严加安. 测度论讲义(第二版)[M]. 北京:科学出版社, 2004

[11] Chung K L. A Course in Probability Theory (Third Edition) [M]. Academic Press, 2001

[12] Hailperin T. Probability Logic[J]. Notre Dame Journal of Formal Logic, 1984, 25(3): 198-212

[13] Wang Guo-jun, Wang Wei. Logical Metric Spaces [J]. Acta Mathematica Sinica, 2001, 44(1): 159-168(in Chinese)
王国俊,王伟. 逻辑度量空间[J]. 数学学报, 2001, 44(1): 159-168

[14] Wu Xin-xing, Hu Guo-sheng. Trustworthiness Measurements of Real-time Web Services[C]// 2014 International Conference on E-Commerce, E-Business and E-Service (EEE 2014). 2014

[15] Wu Xin-xing, Hu Guo-sheng, Chen Yi-xiang. Studies on Measurements of Component Approximate Matching[J]. Computer Science, 2014, 41(5): 190-195(in Chinese)
吴新星,胡国胜,陈仪香. 构件近似匹配的度量研究[J]. 计算机科学, 2014, 41(5): 190-195

[16] Wu Xin-xing, Hu Guo-sheng, Chen Yi-xiang. Quantification and Conformance of Web Service Degraded Substitution[J]. Computer Science, 2015, 42(2): 81-85, 94(in Chinese)
吴新星,胡国胜,陈仪香. Web 服务降级替换的一致性问题及量化研究[J]. 计算机科学, 2015, 42(2): 81-85, 94