关于 Peterson—Fischer 二进程算法的断言式证明

苏运霖
(暨南大学计算机科学系,广州 510632)

AN ASSERTIONAL PROOF FOR PETERSON—FISCHER 2—MUTUAL ALGORITHM

Su Yunlin
(*Department of Computer and Science, Jinan Unversity, Guangzhou* 510632)

Abstract This paper purports to present an assertional proof for Peterson—Fischer 2—mutual exclusion algorithm. According to Nancy A. Lynch of MIT[1], this was an open problem. Hence the significance of this paper is to fill the gap.

摘要 本文旨在为 Peterson—Fischer 二进程互斥算法提供一个断言式证明,根据麻省理工学院 Nancy A. Lynch 教授的论述[1],这是一个未解决的问题.因此本文的意义在于填补这一空缺.

§1. 引 言

Peterson—Fischer 二进程互斥算法是两位作者为解决二进程的互斥执行所提出的一个著名算法[2].它不仅具有满足互斥和进展的特性,而且还满足和 Bakery 算法一样的容错条件,即允许出错的进程重新开始,并使它们的变量复原.然而这些性质还没有以断言式的方法加以证明.因此麻省理工学院的 Nancy A. Lynch 教授把它作为一个未解决的问题提出.

下边把 Peterson—Fischer 二进程算法给出如下:

Peterson—Fischer 二进程互斥算法:

共享变量.

.q:其下标为 0 和 1 且取值{nil, T=1, F=0}的数组,开始时 q[0]和 q[1]皆为 nil,q[i]由 Pi 来写而可由任何进程来读,i=0,1,Pi 即进程.

记号: opp(i)= ~i, * * i 的对手 * *

Pi 的代码.

```
q[i]←if q[opp(i)]=nil then T else i ⊕ q[opp(i)]
q[i]←if q[opp(i)]=nil then q[i] else i ⊕ q[opp(i)]
wait until q[opp(i)]=nil or (i ⊕ (q[opp(i)]≠q[i]))
 * * critical region * *
q[i]←nil
```

图 1 Peterson—Fischer 二进程互斥算法

本文 1990 年 12 月 13 日,1991 年 4 月 22 日收到.作者苏运霖,教授,主要研究领域为智能科学,人工智能,分布式算法,Petri 网等.

© 中国科学院软件研究所 http://www.jos.org.cn

此算法的运作相当简单——仅有进程 P0 和 P1 两者竞争. 在 P0 进入试验区 $\mathscr{T}$ 时,它置 q[0]←q[1],这满足 P1 的等候条件. 相应地,P1 置 q[i]←q[0],这满足 P0 的等候条件,这两个条件显然不能同时满足,当另一进程处于剩余区域 $\mathscr{R}$ 时,一个进程才可满足它的等候条件.

为了看出为什么要进行两次检测,来看出对手是在试验区 $\mathscr{T}$ 还是在临界区 $\mathscr{C}$,考虑仅用一个检测的下列瞬间情况:

	P0 的步骤	P1 的步骤
时间 ↓	读 q[1]=nil	
		读 q[0]=nil
	置 q[0]=$\mathscr{T}$	
	读 q[1]=nil	
	进入 $\mathscr{C}$	
		置 q[1]=T
		读 q[0]=T
		检查 1⊕(T≠T)是否满足
		进入 $\mathscr{C}$

由此可以看出,如果没有第二个检测,就不可能保证两个进程的真正互斥,而只有有了第二个检测,才可使两个进程互斥地进入临界区.

然而上述的说明不是一个断言式的证明. Nancy A. Lynch 因而要求一个断言式的证明.

§2. 算法的完整描述

上边给出的算法,实际上仅是这一算法的片断,它只给出一个进程的代码,为了进行证明的需要,我们需要给出算法的完整描述.

以下是此算法的完整形式:

完整的 Peterson－Fischer 二进程互斥算法.

共享变量.

. q[0],q[1]: 从{nil,T=1,F=0}取值,且初始值皆为 nil,q[0]和 q[1]分别由进程 P0 和 P1 写入,而可为两个进程所读. 我们定义对于{nil,T=1,F=0}的⊕运算如下:

⊕	nil	F	T
nil	nil	F	T
F	F	F	T
T	T	T	F

算法的代码:

P0:

$$q[0]=\begin{cases}T & \text{if } q[1]=nil\\ 0\oplus q[1] & \text{else}\end{cases} \quad (1)$$

$$q[0]=\begin{cases}q[0] & \text{if } q[1]=nil\\ 0\oplus q[1] & \text{else}\end{cases} \quad (2)$$

wait until q[1]=nil or (0 ⊕ (q[1]≠q[0]))

** 进入临界区 **

q[0]←nil

** 剩余区域 **

P1:

$$q[1]=\begin{cases}T & \text{if } q[0]=nil\\ 1\oplus q[0] & \text{else}\end{cases} \quad (3)$$

$$q[1]=\begin{cases}q[1] & \text{if } q[0]=nil\\ 1\oplus q[0] & \text{else}\end{cases} \quad (4)$$

wait until q[0]=nil or (1 ⊕(q[0]≠q[1]))

** 进入临界区 **

q[1]←nil

** 剩余区域 **

图 2 Peterson－Fischer 二进程互斥算法的完整描述

© 中国科学院软件研究所 http://www.jos.org.cn

§3. Peterson — Fischer 二进程互斥算法的断言式证明

现在我们系统地给出 Peterson — Fischer 二进程互斥算法的断言式证明.

定理 1：

□ ~[(q[1]=nil or(0 ⊕ (q[1]≠q[0]))) and (q[0]=nil or (1 ⊕ (q[0]≠q[1])))](5)

这一表达式的含义为：无论任何时候，P0 和 P1 的等候条件不可能同时成立. 因而它们满足互斥条件.

证明：注意(5)式可予展开且有下列的式子成立：

不带□和~的(5)式

=(q[1]=nil) and (q(0)=nil)　　(6)

or (q[1]=nil) and (1 ⊕ (q(0)≠q[1]))　　(7)

or (q[0]=nil) and (0 ⊕ (q(1)≠q[0]))　　(8)

or (0 ⊕ (q[1]≠q[0])) and (1 ⊕ (q(0)≠q[1]))　　(9)

为了获得我们的结论，我们需要来研究此算法的执行. 在执行(1)和(2)之后，P0 的等候条件被检测以确定它可否进入临界区. 类似地，在执行了(3)和(4)之后，P1 的等候条件被检测以确定它可否进入临界区. 为了下边叙述的方便，我们引进记号≺，它表示"先于"关系. 由于(1)的执行先于(2)的执行，我们有(1)≺(2)，且(1)≺(2)≺P0 的等候条件的检测，类似地，(3)≺(4) ≺P1 的等候条件的检测.

由于 P0 和 P1 是并发的，因而(1)(2)和(3)(4)的执行是并发的. 于是总共可以有三种模式六个组合序列顺序. 它们是：

(a) (1) (2) (3) (4)
　　(3) (4) (1) (2)

(b) (1) (3) (2) (4)
　　(3) (1) (4) (2)

(c) (1) (3) (4) (2)
　　(3) (1) (2) (4)

由于对于每个模式 P0 和 P1 的对称性. 我们仅需分析每一种模式的一个序列. 我们将考虑下列三个序列：

(1) (2) (3) (4)

(1) (3) (2) (4)

(1) (3) (4) (2)

此即意为：

(1)≺(2)≺(3)≺(4)

(1)≺(3)≺(2)≺(4)

(1)≺(3)≺(4)≺(2)

现在我们需要证明的是，对于任何执行序列，(5)都成立. 这即是，在任何序列中(6)、(7)、(8)和 (9) 为假. 事实上，我们有

(1)→ {[(q[1]=nil)→(q[0]=T)]
　or[(q[1]=F)→(q[0]=F)]
　or[(q[1]=T)→(q[0]=T)]}
　{[(q[1]=nil)→(q[0]=T)]
　or [(q[1]=F)→(q[0]=F)]
　or[(q[1]=T)→(q[0]=T)]}
　→ ~[(q[1]=nil) and (q[0]=nil)]

所以(1) →(6).

类似地，我们可以推导

(2) →(6) ， (3) →(6) ， (4) →(6)

综合类似的推导，我们可以有下列的表

序列 \ 项	(6)	(7)	(8)	(9)
(1)(2)(3)(4)	FFFF	FFFF	FFFF	FFFF
(1)(3)(2)(4)	FFFF	FFFF	FFFF	FFFF
(1)(3)(4)(2)	FFFF	FFFF	FFFF	FFFF

■

现在，我们以 T 表示项的集合，以 S 表示序列的集合. 假设 t 是 T 中的一项，而 s 是 S 中的一个序列，并令 t(s)表示对于序列 s，t 所取的相应值序列，由上边的表的结果，我们有：

定理 2：

□[∀ t,s ((t∈T ∧ s∈S)→~t(s))]

其中~t(s)表示整个值序列为假.

这个谓词的含义即：对于无论什么样的序列，(5)式总为假.

定理 3：

□[((q[1]=nil) or (0 ⊕ (q[1]≠q[0])))

© 中国科学院软件研究所 http://www.jos.org.cn

or ((q[0]=nil) or (1 ⊕ (q[0]≠q[1])))] (10)

这一表达式的含义是:总有一个等候条件为真.即无论什么时候,两者皆为假的时刻不存在.

证明:为了证明这一点,我们有下面的推导:

(q[1]=nil)→(q[0]=T) (由(1)式上半部)

(q[0]=T)∧(q[1]=nil)

→(q[1]≠q[0]) (由上式)

因此:

0 ⊕ (q[1]≠q[0])=T (由上式)

所以[(q[1]=nil) or (0 ⊕ (q[1]≠q[0])))

or((q[0]=nil)or (1 ⊕ (q[0]

≠q[1])))]=T

类似的:

(q[0]=nil)→(q[1]=T) (由(3)式上半部)

(q[1]=T)∧(q[0]=nil)

→(q[0]≠q[1]) (由上式)

然而,由此

(1⊕(q[0]≠q[1]))=F

因此P1等候条件的第二项为假,但是头一项为真.如果q[1]≠nil,有下列两种可能性:

q[1]=F 或 q[1]=T

今分别论述之:

a)如果q[1]=F,则

(q[1]=F)→(0 ⊕ q[1])=F

所以(q[0]=F)∧(q[1]=F)

所以 1 ⊕ (q[0]≠q[1])=F

所以(q[0]=nil) or (1 ⊕ (q[0]

≠q[1]))=T

即P1的等候条件成立.

b)如果q[1]=T,我们将有

(q[1]=T)→(0 ⊕ q[1])

由(2) q[0]=0 ⊕ q[1]=T

所以(0 ⊕ (q[1]≠q[0]))=F

这就意味着P0的等候条件为假,但却有

1 ⊕ (q[0]≠q[1])=T

即P1的等候条件为真.

由于q[0]和q[1]必取nil,T=1和F=0三者当中的一个值,而它们分别导致P0和P1等候条件的成立.

下表表示出进程P0和P1的等候条件交替为真的情况.

序列	等候条件成立			
	q[0]	q[1]	立即	最后
1 2 3 4	T	nil	P0	
3 4 1 2	nil	T	P1	
1 3 2 4	F	T		P0
3 1 4 2	F	F		P1
1 3 4 2	F	F		P1
3 1 2 4	T	F		P0

上表中,立即指的是在(1)(2)执行之后(当(1)(2)≺(3)(4)时),对于等候条件的测试立即获得为真的结果;或者在(3)(4)执行之后(当(3)(4)≺(1)(2)时),也有类似的情况.而"最后"是指在整个序列完成之后,两个等候条件都受检测所得的结果.

定理 4:在Peterson—Fischer二进程互斥算法中,

□∃ t1∃ t2{[t1→((q1=nil)or(0 ⊕ (q[1]≠q[0])))]∧[t2→((q[0]=nil)or(1 ⊕ (q[0]≠q[1])))]∧(t1≠t2)}

这一式子的含义是和上边的结论类似的,仅仅是以不同的方式来表达而已.它指出,存在使P0或P1获得进展的时刻t1和t2,而且这两个时刻是不同的.在这点上,本定理不同于以前的定理,它指出了进展性而不仅仅是互斥.

证明:我们有

((2)≺(3)) or ((2)≺(4)) → P0等候条件

((4)≺(1)) or ((4)≺(2)) → P1等候条件

此即t1=((2)≺(3)) or ((2)≺(4))

t2=((4)≺(1)) or ((4)≺(2))

显然有t1≠t2,且定理4的论断得证. ■

我们引进记号C(Pi)表示Pi进入临界区𝒞.于是我们有以下几个序列

(1)(2)(3)(4)C(P0)

(3)(4)(1)(2)C(P1)

(1)(3)(2)(4)C(P0)

© 中国科学院软件研究所 http://www.jos.org.cn

(3)(1)(4)(2)C(P1)

(1)(3)(4)(2)C(P1)

(3)(1)(2)(4)C(P0)

因此算法的可达执行序列是这些序列的随机的连接，这意味着无单纯的由(1)(2)加临界区或(3)(4)加临界区的执行序列，以 $\mathscr{R}$ 来表示所有可达执行序列的集合. 定理 4 的论断于是可以叙述成为

定理 5：在 Peterson—Fischer 二进程互斥算法中

$$\sim\exists\, s\{s\in\mathscr{R}|s=\{(1)(2)\mathscr{C}(P0)\}^+\}\wedge\sim\exists\, s\{s\in\mathscr{R}|s\in\{(3)(4)\mathscr{C}(P1)\}^+\}$$

其中的＋表示不包括空序列在内的极大传递闭包.

定理 6：在 Peterson—Fischer 二进程互斥算法中

$$\Box\exists\, t1,t2[(t1\rightarrow(q[0]\leftarrow nil))\wedge(t2\rightarrow(q[1]\leftarrow nil))]$$

此定理断言，两个进程必然要在某一时刻恢复初态(不论它出故障与否)，换而言之，一个出故障的进程不会使不出故障的进程在它的等候条件处受阻.

证明：由于每一进程总有机会进入临界区，而它又不可能在临界区中无限制地停留. 所以总有一个时刻，进程会离开临界区. 一旦它离开，它也就被置为初始状态，即 q[0] 或 q[1]成为 nil. 这一定理的形式证明，是顺序地列出算法的执行序列，即用时态逻辑指出，总有某一时刻(无论出故障与否)，q[0]或 q[1]被置为初态，故定理得证.

参考文献

[1] Nancy A. Lynch and Kenneth J. Goldman, Distributed Algorithms Lecture Notes for 6.852, MIT/LCS/RSS 5 Laboratory for Computer Science MIT, 1989.

[2] G. Peterson and M. Fischer, Economical Solutions for the Critical Section Problem in a Distributed System. In Proceeding of 9th ACM Symposium on Theory of Computing, 91—97, May 1977.

第五届全国青年计算机会议

5th National Conference for Young Computer Scientists

征　文　通　知

第五届全国青年计算机会议(NCYCS'94)定于 1994 年 9 月在历史名城西安召开。热诚欢迎全国从事计算机科学技术及工程应用研究的青年工作者踊跃投稿。会议将邀请著名专家作综述或专题报告，组织有关计算机学科前沿课题及其发展方向的专题讨论。由专业出版社出版会议论文集。评选出的优秀论文将直接推荐到 1995 年国际青年计算机大会。与此同时，还将举办计算机研究与应用的新成果、新产品大型展示会。现将征文的有关事项通知如下：

主　　办：中国计算机学会

承　　办：西北工业大学

征文对象：论文第一作者的年龄不大于 40 岁

征文范围：并行与分布式处理　网络与通讯　软件工程　器件与 VLSI 技术　数据库系统　人工智能与知识工程　CAD/CAM/CAI　计算机科学理论　计算机安全与保密　图形与图象处理　计算机工程与工艺　多媒体技术　计算机应用　文字信息处理

说　　明：(1)应征论文应未在其他学术刊物或学术会议上正式发表过，(2)论文一式三份，注明所属领域及研究(资助)背景，(3)论文尽量用计算机打印(A4 纸)，正文不超过 6000 字，(4)要求 250 字左右的中文摘要并附关键词，(5)论文请自行留底，来稿一律不退，(6)请写清作者详细通讯地址和邮政编码以便联系。

论文截止日期：1994 年 1 月 20 日

论文投寄地址：710072 西北工业大学计算机系 NCYCS'94 筹委会　赵政文

© 中国科学院软件研究所 http://www.jos.org.cn