并行程序的流程图分析法*

马 军　马绍汉

(山东大学计算机系,济南 250100)

摘要 本文给出一种对 Ada 并行程序的流程图分析法. 该方法不仅给出在并行程序中,并发多任务之间相互通信的一种可视化的表示,同时可检查出并行程序中存在的死锁,并能启发如何消除死锁. 并行流程图在检查死锁的方便性、消除死锁的启发性、实际应用的可行性和对并行算法思想的描述性等方面明显优于已知的 Petri 网分析法.

关键词 程序正确性验证,并行程序设计,并行程序静态分析.

并行程序由于其多进程的并发执行,执行语句顺序的非确定性及发生死锁不易检查等特性,比串行程序更难理解和验证其正确性,而且目前的并行程序设计语言,尚不能很好地对并行程序进行 debug,因此对并行程序给出一种可视化的描述与分析方法,已成为国际上对并行程序设计的重要研究课题之一[1-6]. 目前国际上主要讨论用 Petri 网对并行程序进行分析,但因 Petri 网是基于对各种瞬时动作的描述来完成对整个并行程序的图描述,对一稍微复杂的程序,其 Petri 网将十分庞大复杂,因此在实际应用上受到了很大限制. 本文基于 Petri 网的原理,给出了一种适合于 Ada 并行程序的、实用的流程图分析法.

1 Ada 多任务并发程序的流程图表示

有关 Petri 网和 Ada 语言的详细知识请参见文献[7,8]. 在 Ada 中,进程被称为任务. 图 1 给出自动售油机的 Ada 并行程序. Customer、Operator 和 Pump 是三个并发任务,共同完成以下功能:Customer 收到顾客的预付款后通知 Operator,Operator 接到通知后,通知 Pump 起动油泵,Customer 命令 Pump 进行注油和关闭油泵. 当 Pump 收到关闭命令后,关闭油泵并通知 Operator,Operator 接到通知后,通知 Customer 为顾客付余款. 图 1 似乎给出了一正确的实现,然而,该程序中存在一个在语句级很难查觉到的死锁. 下面以该程序为例,开始我们的讨论.

```
task body Customer is
  begin
    loop
      Operator. Prepay
```

* 本文1994－04－14收到,1994－08－08定稿

本课题受到山东省自然科学基金资助. 作者马军,1956年生,副教授,主要研究领域为算法分析与设计,并行算法,人工智能. 马绍汉,1938年生,教授,主要研究领域为算法分析与设计,并行算法,人工智能.

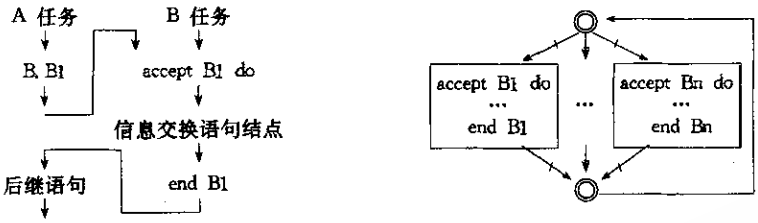
本文通讯联系人:马军,济南250100,山东大学计算机系

```
          Pump. Start
          Pump. Finish
          accept Change
        end loop
      end customer

    task body Pump is
      begin
        loop
          accept Activate
          accept Start
          accept Finish do
          Operator. Charge
        end Finish
      end loop
end Pump

    task body Operator is
      begin
        loop
          select
            accept Prepay do
              Pump. Activate
            end Prepay
          or
              accept Charge do
                Customer. Change
              end Charge
          end select
        end loop
      end Operator
```

图1 自动售油机的 Ada 程序

串行程序的流程图是程序的有向图 G=〈V,A〉表示法,其中 V 为全体语句集合,若 S2 是 S1的后继语句,则〈S1,S2〉∈A. 对并行语言则要给出对通信(包括同步)的表示. 在 Ada 语言中,任务间的通信是通过会合语句实现的. 若 A 任务要求与 B 任务的入口 B1(entry B1)会合时,用语句 B. B1实现. 语句 B. B1隐含了3个操作,(1)A 向 B 发出会合要求,然后等待 B 执行到相应接受语句 accept B1 do(同步操作为 accept B1);(2)A 与 B 共同完成 accept B1 do 与 end B1之间信息交换;(3)在 end B1语句后,两并行任务分离,重新并发执行.(对同步操作,只有3.);对于 B 任务中的 accept B1 do(同步操作为 accept B1)语句,该语句也隐含着3个操作:(1)等待到某一并行任务 A 执行到语句 B. B1;(2)与 A 共同完成 accept B1 do 与 end B1之间的信息传递操作;(3)在 end B1语句后,两并行任务分离重新并发执行(对同步操作,只有3.). 另外,loop select…or…end select end loop 语句是由多个并行 accept 语句组成的复合控制语句. 图2(a)、(b)给出对上述任务间相互通信的流程图表示法. 其他串行语句的表示法同传统的一样.

并行流程图中给出并发多任务之间通信及动态执行的可视化表示. 由于增加了通信边,可能对一结点 v∈V,v 有多条入边,约定,v 被执行的充要条件为,对所有入边〈w,v〉∈A,w 已被执行. 其充分性同 Petri 网的发火[7]的条件一样,表现并发任务之间的同步;其必要性则进一步说明,我们把程序在并行流程图上的执行看成是由信息驱动的. 另外,在并行流程图



(a) 会合语句的表示 (b) Loop selec…控制语句，任一时刻只能响应一个会合要求

图2 Ada通信语句的流程图

中，accept 语句被分为两个以上的结点，即 accept B1 do 结点、中间交换信息的若干结点和 end B1 do 结点；特别要注意的是，提出会合语句 B. B1的结点 v，它与书写顺序的后继语句结点 v′没有了边，而是通过一条经过 B 任务的 accept B1 do 结点和 end B1结点的有向路径到达 v′. 因此，v′能被执行的必要条件为始于 v 的信息流一定最终经过 end B1结点到达 v′.

根据上述画法，在图3中我们给出了对图1程序的并行流程图表示. 并用双实线标出的始于 Customer 的 Pump. Finish 语句结点的实际信息路径，当信息流到达结点 Accept Charge 时，显然该信息流未经过 Pump 的 end Finish 结点. 故不可能由 end Finish 结点传送信息到 Accept Charge，所以 Accept Charge 不可能被执行，从而程序陷入死锁.

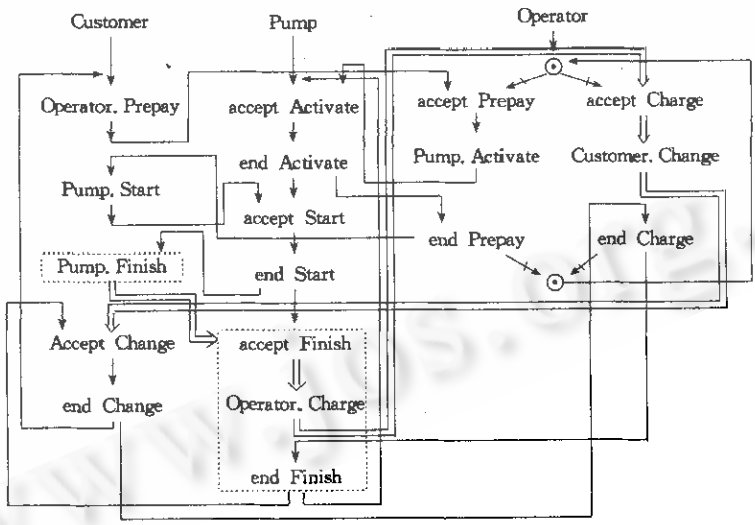


图3 自动售油机程序的并行流程图

定义1. 在并行程序流程图 G 中，设结点 v 表示某一任务要求与 B 任务的入口 B1会合语句，v′为 v 的后继语句，若始于 v 的实际信息流最终总可经过代表语句 end B1的结点到达 v′，则称 v 为安全的；否则，若 v 不是安全的，并且实际始于 v 的信息流所途经的中间会合语句结点均为安全的，则称 v 所代表的语句为死锁语句，相应 B 任务中的 accept B1 do 语句被称为死锁源语句.

依定义1，在图3中 Customer 任务中的 Pump. Finish 语句为死锁语句；Pump 任务中的

accept Finish…end Finish 为死锁源语句,死锁语句意指该语句引起 Ada 并行程序中的死锁;死锁源语句意指该语句是造成死锁的根源. 实际上,在并行程序设计中,所有的死锁,反映在以语句为结点的并行流程图上,均表现为存在死锁语句,故我们有定理1.

定理1. 一 Ada 并行程序中不存在死锁的充要条件为,在其并行流程图中,不存在死锁语句.

以下给出消除 Ada 并行程序中的死锁算法:

1. 检查并行流程图中是否有死锁语句;若没有时,成功返回;

2. 找出死锁源语句,在满足程序功能的前提下,修改死锁源语句所在的任务代码并根据修改后的语句代码修改并行流程图,转1.

3. 若不能修改,停止. 重新设计算法,划分任务及设计任务之间的通信.

根据以上算法,我们修改任务 Pump 代码如图4. 读者不妨自己修改图3的流程图,并验正修改后的并行流程图不再存在死锁语句,而且能正确地完成自动售油机功能.

```
task body Pump is
  begin
    loop
      accept Activate
      accept Start
      accept Finish
      Operator.Charge
    end loop
  end Pump
```

图4 消除死锁后的 Pump 任务代码

2　结束语

与使用 Petri 网相比,使用并行流程图对 Ada 并行程序进行分析有三大优点:(1)检查死锁简单,能启发消除死锁. 对采用面向对象的方法,设计出的并行程序,基本上能在消除死锁算法的前二步完成. 而在 Petri 网中通过检查网络中是否存在有向回路的方法检查死锁,在网络比较复杂的情况下,并非易事,对如何修改程序启发性也比较差;(2)实用性强. 一般情况下,用一定尺寸的纸,总可以清晰地、有规律地画出并行程序的流程图. 而 Petri 网则不然,有时即使画的开,由于边数太多,绘图规律性差,使得很难阅读分析;(3)Petri 网的结点为抽象的符号,并行算法的思想很难反映出来,而且对瞬时操作的理解不一样,Petri 网常因人而异. 而在并行流程图中,图的结点为语句,结合信息流,并行程序的可视性强,容易解释清楚并行算法的思想及程序的动态并行执行的过程.

我们对并行程序设计中常遇到的精典问题,如允许任意多读任务同时读数据,只允许一个写任务写的 CRWR 共享内存 Ada 管理程序,5个哲学家就餐的 Ada 并行程序[8]等均应用并行流程图给出直观的描述,检查出不正确程序中的死锁. 对由并行程序的不确定性引起的对同一输入,多次运行其输出结果不一样的现象(如哲学家就餐顺序)也能作出解释. 同理,我们也可以给出基于 Occam,Star－Mod,SR 语句的并行流程图分析法,我们认为,并行流程图分析法将成为并行程序的强有力的分析方法.

致谢　本文作者十分感谢审稿老师对原稿提出的建设性的修改意见.

参考文献

1 Murata T. Petri nets: properties, analysis and applications. Proc. IEEE, 1989,77(4):541—580.

2 Helmbold D, Luckham D. Debugging Ada tasking programs. IEEE Softw., 1985,2(2):45—57.

3 Shatz S M, Cheng W K. A Petri net framework for automated static analysis of Ada tasking behavior. J. Syst. Softw., 1988,8:343—359.

4 Murata T, Shenker B, Shatz S M. Detection of Ada static deadlocks using Petri net invariants. IEEE Trans. Softw. Eng., 1989,15(3):314—326.

5 Tiusanen M, Murata T. Graph models for static analysis of ada tasking programs. Report of Info. Processing Research in Japan, 1992,92(59):141—150.

6 Murata T. Static analysis of concurrent programs by Petri nets. Info. of Processing of Japan, 1993,34(6):701—709.

7 Peterson J L. Petri net theory and the modelling of systems. New Jersey, Prentice Hall, 1981.

8 Habermamm A N, Perry D E. Ada for experienced programmers. Massachusetts, Addison Wesley, 1983.

THE STATIC ANALYSIS FOR CONCURRENT PROGRAMS BY FLOWCHART

Ma Jun Ma Shaohan

(*Department of Computer Science, Shandong University, Jinan* 250100)

Abstract In this paper, a kind of flowchart is given for audio—visual analysis to Ada parallel programs, the flowchart can not only give the audio—visual description for the communications among tasks, but also find all deadlocks and help people delete deadlocks. As a static analysis methord for parallel programs, flowchart is better than Petri nets in the convenience of finding deadlocks, helping to delete deadlocks, practical feasibility in use and the description for the ideas of parallel algorithms.

Key words Program correctness examination, parallel programming, static analysis of concurrent programs.