适用于分布式程序设计的图示化模型*

方 林 谢 立

(南京大学计算机软件新技术国家重点实验室 南京 210093)

摘要 VMDP(visual mode for distributed programming)模型是一个基于图标的面向分布式程序设计的图示化模型.它用图标表示分布式系统中的对象及其控制,例如,用车辆、路口和红绿灯等图标分别表示进程、资源和进程控制,从而使分布式系统中进程之间的同步、异步、互斥、共享等协调关系变得十分直观和易于理解.VMDP模型可以广泛应用到分布式系统模型、网络程序设计、并行程序设计和分布式程序设计等领域.

关键词 分布式系统,图示语言,程序设计模型,分布式程序设计.

中图法分类号 TP311

分布式程序是若干独立运行的进程的集合,这些进程之间通过相互协作完成一个共同目标.分布式程序必须解决进程间的同步、异步和互斥等问题,以达到资源共享、减少消息传递数量的目的,避免死锁、饿死等现象的发生.由于分布式程序具有这些与顺序程序和集中式程序截然不同的特殊性质,用传统文本表示的分布式程序往往是非常复杂和难以理解的.

为了解决这个问题,人们提出了CODE[1]和HeNCE等图示化工具.这些工具主要用来在异构环境下编写分布式和并行程序.其特点是用结点表示一个程序段,用连线表示结点之间的控制流程或数据流程.这类模型过于简单,很难描述进程之间的复杂协调关系.基于这种考虑,本文提出了一个图示化的分布式程序设计模型VMDP(visual mode for distributed programming).它用图标语言[2,3]表示分布式程序中的对象(如进程、资源等)和控制(如同步、异步和互斥等),使进程之间复杂的同步关系、通信关系和依赖关系等变得明显和易于理解.VMDP中的图标是对现实生活中交通系统的对象的抽象,它们非常直观.例如,红绿灯表示同步或异步控制,路口表示一个需协调使用的分布式资源等.VMDP中的图标是结构化的,可用基本图标构造复合图标.VMDP还定义了简单的图标操作,使用户可以方便地操纵和使用图标.最后,本文对VMDP的一些应用实例进行了分析.

1 VMDP模型

VMDP模型包括图标和图标操作两部分.前者表示分布式系统中的对象和控制,分为基本图标和复合图标两大类.复合图标是对基本图标施以图标操作而构成的图标.

1.1 基本图标

图1显示了VMDP中的基本图标.其中红绿灯用RG表示,有红灯和绿灯两种状态.车辆用VC表示,代表一个进程.VC只能沿着道路RD排成一条线行驶(即不能有两辆以上的汽车在同一条道路上并排行驶),并受到红绿灯的指挥.红灯停,表示进程处于挂起状态.车辆在红绿灯处等候.1次最多有1辆汽车在红绿灯处等候,后继车辆将在该车后排队.绿灯行,表示进程处于活跃状态.VMDP用红绿灯对车辆的控制来模拟分布式系统对进程的控制.道路上的箭头指明车辆行驶的方向.路口CR是道路与道路的交叉点,表示分布式系统中的1个资源或系统对进程进行协调的1个点.为简化讨论,我们约定车辆在通过路口时是直线行驶,并不转弯.

---

* 本文研究得到国家863高科技项目基金资助.作者方林,1970年生,博士,副教授,主要研究领域为图示化分布式程序设计环境,分布计算.谢立,1942年生,教授,博士生导师,主要研究领域为分布计算,并行处理.

本文通讯联系人:方林,上海 200135,上海海运学院计算机系

本文1997-12-18收到原稿,1998-03-17收到修改稿

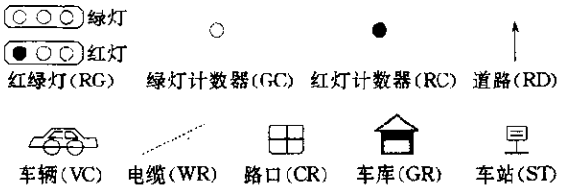

图1 VMDP的图标

图1中红灯计数器RC和绿灯计数器GC用来对红绿灯进行控制,当有车辆经过1个RC或GC时,该计数器将自动加1.RC和GC的初始值为0,并通过电缆WR同至少1个红绿灯连接在一起.1个RG必定同1个RC和1个GC连接在一起,并且当RC的计数大于GC的计数时,发出红灯信号;反之,发出绿灯信号.如果计数相等,RG处于初始状态.

车辆到达1个车库GR表示对应的进程应结束本次调用或转入下一个程序流程.车站ST是没有红绿灯的道路交叉点.指向ST和从ST发出的道路可以有任意多条,到达ST的车辆将等待,直到从ST发出的道路中有1条可通行(见第1.2节)为止.如果有多条可通行道路则任选一条通过.例如,图8中消费者进入车站后面临两条道路,分别通向两个路口.若两个路口都为红灯,那么消费者将在车站中等待,直到其中1个路口为绿灯为止.

图2显示了VMDP模型在异步读写问题中的应用,图右是对应的图标操作(见第1.2节).代表读进程的车辆将从右边进入路口,代表写进程的车辆将从下边进入路口.这里,路口表示被读写的资源.rg_1,rg_2和rg_3都被初始化为绿灯,表示两个方向来的车辆谁先到,谁先通过.这意味着资源是被读进程和写进程竞争使用的.rc_1和rc_2使两边来的车辆不能同时进入路口,表示读、写进程只能互斥地访问资源.而gc_1和gc_2将在车辆经过时增加红绿灯的绿灯计数,以恢复红绿灯的初始状态.rc_3,gc_3和rg_3将保证一次只能有一辆车辆从下方进入路口,即一次只能有一个写进程访问资源.VMDP就是用这种直观、简洁的红绿灯规则表示读写进程之间的复杂关系.它的简单和易于理解的特点是非常明显的.

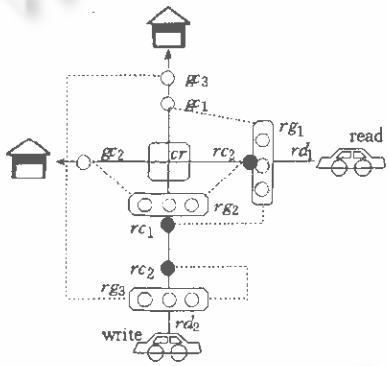

图2 异步读写问题

$cr=CR()$;
$rc_1=RC()$;$rc_2=RC()$;$rc_3=RC()$;
$gc_1=GC()$;$gc_2=GC()$;$gc_3=GC()$;
$rg_1=RG(G,rc_1,gc_1)$;
$rg_2=RG(G,rc_2,gc_2)$;
$rg_3=RG(G,rc_3,gc_3)$;
$rd_1=RD(rg_1,rc_2,cr,gc_2,GR())$;
$rd_2=RD(rg_3,rc_3,rc_1,rg_2,cr,gc_1,gc_3,GR())$;
read$=VC(rd_1)$;
wirte$=VC(rd_2)$;

1.2 图标操作

为了把各种基本图标组合在一起,从而构成一个复合图标,VMDP提供了一种称为图标申明的图标操作,其形式是〈图标变量〉=〈图标类型〉(〈属性表〉).其中图标变量是一个代表所申明图标的标识符,等号右边称为一个图标申明.不同类型图标的申明方式是不一样的,下面分别说明.

(1) CR,ST,RC,GC,GR的申明.这些图标都没有属性,令属性表为空即可.例如,申明一个路口用$v=CR()$表示.

(2) RG的申明.红绿灯的属性是初始状态、与之相关联的红灯计数器和绿灯计数器.用$v=RG$(〈初始状态〉,〈红灯计数器〉,〈绿灯计数器〉)申明.这里,初始状态用R表示红灯状态,用G表示绿灯状态.〈红(绿)灯计数器〉是一个红(绿)灯计数器变量或申明.

(3) 申明RD.申明一条道路只需在属性表中指明这条道路上依次经过的图标即可:$v=RD$(〈图标1〉,〈图标2〉,...).这里,〈图标i〉是一个图标变量或图标申明.

(4) 申明VC.申明一辆车辆要指明它所行驶的道路:$v=RD$(〈道路〉),这里,〈道路〉是一个道路变量或申明.例如,图2左是运用图右所示申明的结果(申明与申明之间用分号隔开).

VMDP规定:(1) 如果一条道路通向一个车站,那么它必须是该道路所到达的最后一个图标;(2) 当一个车

站发出多条道路时,车辆到达该车站后将等待,直到该车站可通行为止.一个车站被定义为可通行,只要它发出的道路中有 1 条所通向的第 1 个红绿灯发出绿灯信号或(在不经过红绿灯的情况下)通向的车站可通行为止.

2　进程之间关系的表达

进程之间的关系主要有异步、同步、交替和等待 4 种.下面以 VMDP 模型为工具对这 4 种关系分别加以描述.

如图 3 所示,从右方和下方开来的车辆不能同时进入路口.图 2 所示的路口是这种路口的一个应用.这种路口用来表示要互斥使用的资源,访问这种资源的进程之间具有异步关系.图 3 中右下所示的图标是上述表示异步关系路口的简化表示.这样,图 2 所示路口可以表示为图 4.

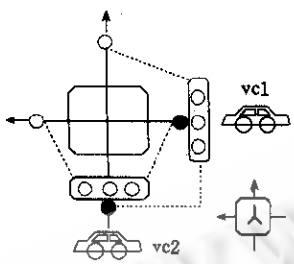

图3　异步

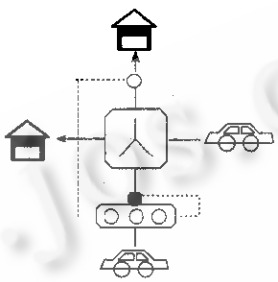

图4　异步读写问题简化解

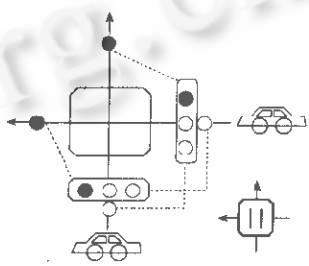

图5　同步

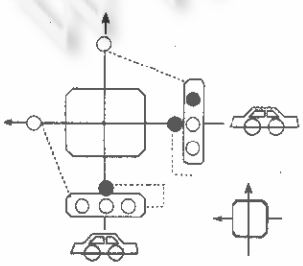

图6　交替

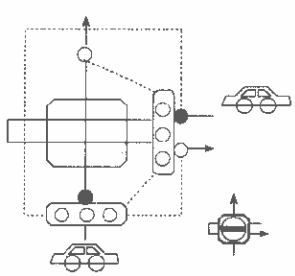

图7　等待

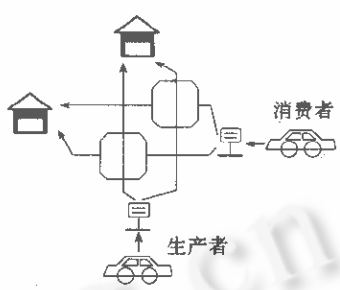

图8　双缓冲生产者—消费者问题

图 5 中,路口用来表示同步资源,描述进程之间的同步关系(右下图标是这种路口的简化表示).右、下两边试图进入路口的车辆经过绿灯计数器,从而令对方红绿灯发出绿灯信号,通过路口之后,将经过红灯计数器,使红绿灯的红灯计数和绿灯计数相等从而转为初始状态,发出红灯信号.这表明只有当右、下两边都有车辆时,它们才能同时通过路口.这是同步控制的情形.图 9 是这种路口的一个应用.

图 6 中有两个红绿灯分别被初始化为红灯和绿灯.从右、下两边进入路口的车辆首先把本方向上的红绿灯置为红灯,通过路口后再将另一方向的红绿灯置为绿灯.这意味着右、下两边车辆只能交替通过路口,并且第 1 辆通过的是从下边进入的车辆(右下图标是这种路口的简化表示).图 8 是这种路口的一个应用.图 7 显示了一种特殊的异步控制方式——等待.从路口右、下两边开来的车辆以竞争的方式通过路口,这一点与图 3 异步路口相同.不同的是,如果有车辆已经从右边进入路口,那么从下方开来的车辆将等待直到这辆车从原路返回再次通过路口为止.当有进程需要同时占有多个资源时这种路口非常有用.图 10 是这种路口的一个应用.

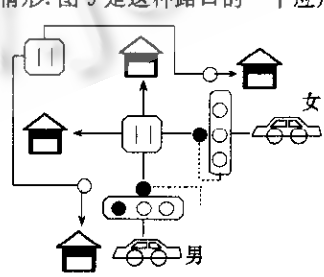

图9　舞会问题

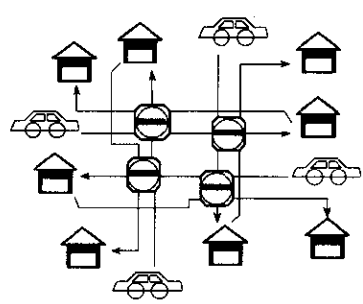

图10　哲学家就餐问题

以上 4 种关系并不能涵盖进程之间的全部关系.但我们可以利用表示这 4 种关系的路口,通过图标操作加

以组合从而表达复杂的关系.下面通过实例加以说明.

3 实例分析

实例 1:多缓冲的生产者-消费者问题

图 8 是双缓冲生产者-消费者问题的解.其中,两个路口表示两个缓冲,生产者将首先访问其中的一个,随后消费者访问该缓冲.对于每个缓冲,生产者和消费者总是交替访问它.图 8 清楚、直观地表明了生产者和消费者之间的协调.

实例 2:舞会问题

每个参加舞会的人都要寻找舞伴,找到后得到对方姓名.这个问题分为 3 步解决.首先,男女舞者必须配对进入舞场,然后在登记本上写上自己姓名以便对方查阅,最后从登记本上获得对方的姓名.在这个较复杂的例子中,我们将再次看到 VMDP 模型的强大功能.

图 9 右下方同步路口表示大门,大门将保证男女舞者成对进入舞场.另一个路口表示记录姓名的登记本.车辆将依次通过两个车库.第 1 个车库是男/女进程程序运行的一个中转站.男女舞者应在此把自己的姓名写到登记本上,随后进入第 2 个同步路口获得对方姓名.最后将通过一个绿灯计数器,从而允许第 2 对舞者进入大门.

实例 3:哲学家就餐问题

图 10 是 4 个哲学家问题的解.这是由 4 个如图 7 所示的等待路口组成的一个较复杂的图形.其中每个路口代表一把调羹,每辆汽车代表一个哲学家.车辆到达第 1 个车库表示该哲学家已拿到两把调羹,可以进餐.到达第 2 个车库表示进餐完毕,并已把调羹放回到桌面上.图 10 形象地表明每把调羹是如何被左右两个哲学家竞争使用的.图 10 还表明,当 4 辆车同时启动时,如果速度相同(即进程运行速度相同),它们都能顺利通过第 1 个路口,但在各自的第 2 个路口处被红灯挡住.这是死锁的形象化表示.解决的方法是从中任取一辆汽车,调换它所经过的两个路口的次序即可.

4 结束语

VMDP 使用交通系统的对象,如车辆、道路和红绿灯作为基本元素,描述分布式系统中各进程之间复杂的协调关系.VMDP 的图标和图标操作都是非常简单和直观的.在以上实例中,我们看到了 VMDP 模型的强大功能.

VMDP 模型中的红绿灯、路口等图标常被用来表达两个或两类进程之间的协调关系,我们很容易把它们的功能加以扩展(如通过一个路口的道路允许有多条),从而方便地描述 3 个或 3 类以上进程之间的协调关系.这在本质上没有增加 VMDP 模型的功能,此处不再赘述.

参考文献

1 Browne J C, Azam M, Sobek S. CODE: a unified approach to parallel programming. IEEE Software, 1989,6(4):10~18
2 Clarisse O, Chang S K. An icon manager in lisp. In: Proceedings of 1985 IEEE Workshop on Languages for Automations. IEEE Press, 1985. 116~131
3 Korfhage R R, Korfhage M A. Criteria for iconic languages. In: Chang S K *et al* eds. Visual Languages. New York: Plenum Publishing Corporation, 1986. 207~231

A Visual Model for Distributed Programming

FANG Lin　XIE Li

(*State Key Laboratory for Novel Software Technology　Nanjing University　Nanjing*　210093)

Abstract　VMDP is a visual model for distributed programming based on icons, which is used to illuminate the objects and the controls in a distributed system. With VMDP, a process is represented by a vehicle, a resource by a crossroad, and the coordination among processes by the traffic light. The coordination relations among processes and controls such as synchronization and mutual exclusion are understandable and readable in VMDP. VMDP devotes itself to building distributed system model, network programming, and parallel/distributed programming as well.

Key words　Distributed system, visual language, programming model, distributed programming.