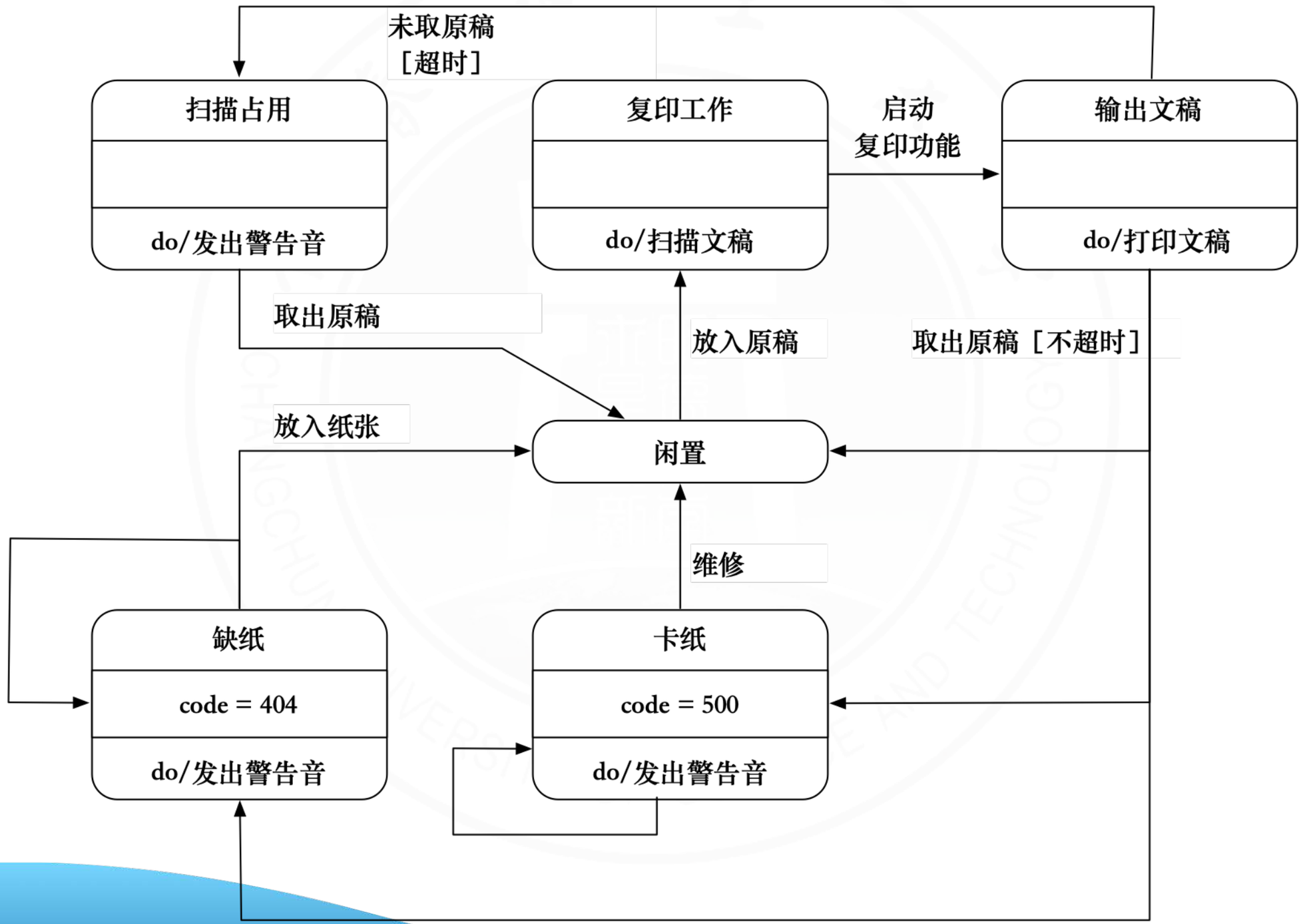
第四章

形式化说明技术

习题

- 复印机的工作过程大致如下
  - 未接到复印命令时处于闲置状态
  - 一旦接到复印命令则进入复印状态，完成一个复印命令规定的工作后又回到闲置状态，等待下一个复印命令
  - 如果执行复印命令时发现没纸，则进入缺纸状态，发出警告，等待装纸，装满纸后进入闲置状态，准备接收复印命令
  - 如果复印时发生卡纸故障，则进入卡纸状态，发出警告等待维修人员来排除故障，故障排除后回到闲置状态
- 请用状态转换图描绘复印机的行为


未取原稿
[超时]
扫描占用
do/发出警告音
复印工作
do/扫描文稿
启动
复印功能
输出文稿
do/打印文稿
取出原稿
放入原稿
取出原稿［不超时］
放入纸张
闲置
维修
缺纸
code = 404
do/发出警告音
卡纸
code = 500
do/发出警告音

软件工程中的形式化方法

- 按照形式化的程度，可以把软件工程使用的方法划分成
  - 非形式化
    - 用自然语言描述需求规格说明，是典型的非形式化方法
  - 半形式化
    - 用数据流图或实体-联系图建立模型，是典型的半形式化方法
  - 形式化
- 什么是形式化方法
  - 描述系统性质的基于数学的技术
  - 即如果一种方法有坚实的数学基础，那么它就是形式化的

非形式化方法的问题

- 用自然语言书写的系统规格说明书，可能存在
  - 矛盾、二义性、含糊性、不完整性及抽象层次混乱等问题
    - 所谓矛盾是指一组相互冲突的陈述
    - 二义性是指读者可以用不同方式理解的陈述
- 系统规格说明书是很庞大的文档
  - 几乎不可避免地会出现含糊性
  - 不完整性可能是在系统规格说明中最常遇到的问题之一
  - 抽象层次混乱是指在非常抽象的陈述中混进了一些关于细节的低层次陈述
    - 使得读者很难了解系统的整体功能结构

形式化方法的优点

- 人在理解用自然语言描述的规格说明时，容易产生二义性
- 为了克服非形式化方法的缺点，人们把数学引入软件开发过程，创造了基于数学的形式化方法
- 优点
  - 在开发大型软件系统的过程中应用数学，能够简洁准确地描述物理现象、对象或动作的结果
    - 因此数学是理想的建模工具
    - 数学特别适合于表示状态，也就是表示“做什么”
  - 可以在不同的软件工程活动之间平滑地过渡
    - 不仅功能规格说明，而且系统设计也可以用数学表达，当程序代码也是一种数学符号
      - 虽然程序代码是一种相当繁琐、冗长的数学符号
  - 数学提供了高层确认的手段
    - 可以使用数学方法证明，设计符合规格说明，程序代码正确地实现了设计结果
- 需求规格说明书主要描述应用系统在运行前和运行后的状态
  - 因此，数学比自然语言更适于描述详细的需求
- 在理想情况下，分析员可以写出系统的数学规格说明，它准确到几乎没有二义性，而且可以用数学方法来验证，以发现存在的矛盾和不完整性，在规格说明中完全没有含糊性
  - 但是实际情况中，试图用少数几个数学公式来描述复杂的软件系统是根本不可能的
  - 即使应用了形式化方法，也很难保证完整性
    - 由于沟通不够，可能遗漏了客户的一些需求
    - 规格说明的撰写者可能有意省略了系统的某些特征，以便设计者在选择实现方法时有一定自由度
    - 要设想出使用一个大型复杂系统的每一个可能的情景，通常是做不到的

应用形式化方法的准则

- 人们对形式化方法的看法并不一致
  - 形式化方法对某些软件工程师很有吸引力
    - 其拥护者甚至宣称这种方法可以引发软件开发方法的革命
  - 另一些人则对把数学引入软件开发过程持怀疑甚至反对的态度
- 对形式化方法也应该“一分为二”
  - 既不要过分夸大它的优点也不要一概排斥

应用形式化方法的准则

- 应该选用适当的表示方法
  - 一种规格说明技术只能用自然的方式说明某一类概念，如果用这种技术描述其不适于描述的概念，则不仅工作量大而且描述方式也很复杂
- 应该形式化，但不要过分形式化
  - 目前的形式化技术还不适于描述系统的每个方面
  - 形式化规格说明技术要求我们非常准确地描述事物，因此有助于防止含糊和误解
  - 如果用形式化方法仔细说明系统中易出错的或关键的部分，则只用适中的工作量就能获得较大回报
- 应该估算成本
  - 为了使用形式化方法，通常需要事先进行大量的培训
  - 最好预先估算所需的成本并编入预算
- 应该有形式化方法顾问随时提供咨询
  - 绝大多数软件工程师对形式化方法中使用的数学和逻辑并不很熟悉，而且没受过使用形式化方法的专业训练，需要专家指导和培训
- 不应该放弃传统的开发方法
  - 把形式化方法和结构化方法或面向对象方法集成起来是可能的，而且取长补短往往能获得很好的效果
- 应该建立详尽的文档
  - 建议使用自然语言注释形式化的规格说明书，以帮助用户和维护人员理解系统

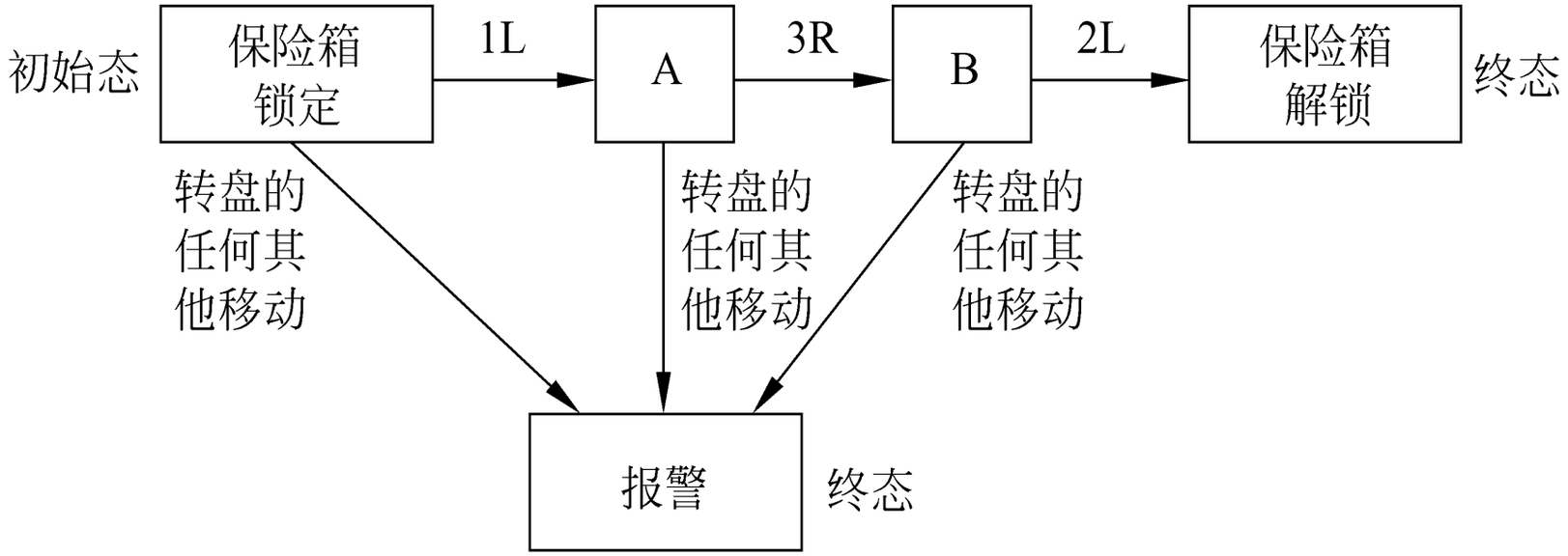
应用形式化方法的准则

- 不应该放弃质量标准
  - 形式化方法并不能保证软件的正确性，它们只不过是有助于开发出高质量软件的一种手段
  - 除了使用形式化说明技术外，在系统开发过程中仍然必须一如既往地实施其他质量保证活动
- 不应该盲目依赖形式化方法
  - 形式化方法只是众多工具中的一种
  - 形式化方法并不能保证开发出的软件绝对正确
    - e.g., 无法用形式化方法证明从非形式化需求到形式化规格说明的转换是正确的，因此，必须用其他方法(例如，评审、测试)来验证软件正确性
- 应该反复测试
  - 形式化方法不仅不能保证软件系统绝对正确，也不能证明系统性能或其他质量指标符合需要，因此，软件测试的重要性并没有降低
- 应该重用
  - 即使采用了形式化方法，软件重用仍然是降低软件成本和提高软件质量的惟一合理的方法
  - 形式化方法说明的软件构件具有清晰定义的功能和接口，使得它们有更好的可重用性

有穷状态机

实例

- 一个保险箱上装了一个复合锁
- 锁有三个位置，分别标记为1、2、3，转盘可向左(L)或向右(R)转动
  - 在任意时刻转盘都有6种可能的运动
    - 即1L、1R、2L、2R、3L和3R
  - 保险箱的组合密码是1L、3R、2L，转盘的任何其他运动都将引起报警



有穷状态机的组成部分

- 一个有穷状态机包括下述5个部分
  - (1)状态集 J、(2)输入集K、(3)由当前状态和当前输入确定下一个状态(次态)的转换函数T、(4)初始态S和(5)终态集F
- 对于保险箱实例
  - 状态集J : {保险箱锁定，A，B，保险箱解锁，报警}
  - 输入集K :{1L，1R，2L，2R，3L，3R}
  - 转换函数T : 如表所示
  - 初始态S : 保险箱锁定
  - 终态集F : {保险箱解锁，报警}

当前状态 / 次态 / 转盘动作	保险箱锁定	A	B
1L	A	报警	报警
1R	报警	报警	报警
2L	报警	报警	保险箱解锁
2R	报警	报警	报警
3L	报警	报警	报警
3R	报警	B	报警

有穷状态机的一般化表示

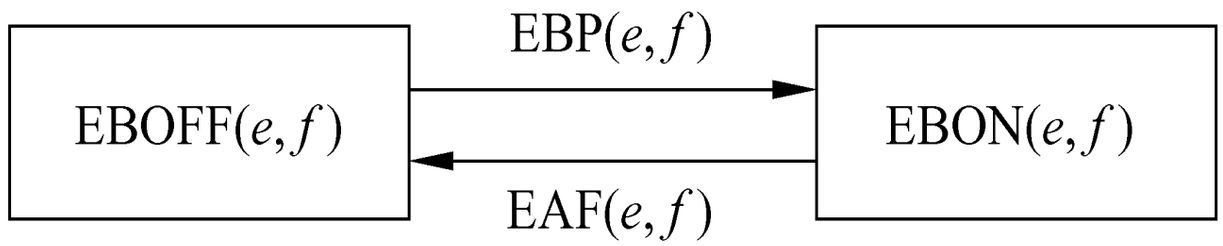
- 一个有穷状态机可以表示为一个5元组(J，K，T，S，F)，其中
  - J是一个有穷的非空状态集
  - K是一个有穷的非空输入集
  - T是一个从(J-F)×K到J的转换函数
  - S∈J，是一个初始状态
  - F 包含于 J，是终态集
- 另外一个实例
  - 每个菜单驱动的用户界面都是一个有穷状态机的实现
  - 一个菜单的显示和一个状态相对应，键盘输入或用鼠标选择一个图标是使系统进入其他状态的一个事件
  - 状态的每个转换都具有下面的形式
    - 当前状态〔菜单〕+ 事件〔所选择的项〕➢ 下个状态
- 为了对一个系统进行规格说明，通常都需要对有穷状态机做一个很有用的扩展
  - 在前述的5元组中加入第6个组件：谓词集P
  - 其中每个谓词都是系统全局状态Y的函数
  - 转换函数T现在是一个从(J-F)×K×P到J的函数
  - 转换规则形式为
    - 当前状态〔菜单〕+事件〔所选择的项〕+谓词 ➢ 下个状态

实例

- 在一幢m层的大厦中需要一套控制n部电梯的产品，要求这n部电梯按照约束条件C1，C2和C3在楼层间移动
  - C1
    - 每部电梯内有m个按钮，每个按钮代表一个楼层
    - 当按下一个按钮时该按钮指示灯亮，同时电梯驶向相应的楼层，到达按钮指定的楼层时指示灯熄灭
  - C2
    - 除了大厦的最低层和最高层之外，每层楼都有两个按钮分别请求电梯上行和下行
    - 这两个按钮之一被按下时相应的指示灯亮，当电梯到达此楼层时灯熄灭，电梯向要求的方向移动
  - C3
    - 当对电梯没有请求时，它关门并停在当前楼层
- 现在使用一个扩展的有穷状态机对本产品进行规格说明
  - 问题中有两个按钮集
    - n部电梯中的每一部都有m个按钮，一个按钮对应一个楼层
      - 因为这m×n个按钮都在电梯中，所以称它们为电梯按钮
    - 每层楼有两个按钮，一个请求向上，另一个请求向下，这些按钮称为楼层按钮

实例 - 电梯按钮

- 令EB(e,f)表示按下电梯e内的按钮并请求到f层去
- EB(e,f)有两个状态
  - 按钮发光(打开) ➢ EBON(e,f)：电梯按钮(e,f)打开
  - 按钮不发光(关闭) ➢ EBOFF(e,f)：电梯按钮(e,f)关闭
- 规则
  - 如果电梯按钮(e,f)发光且电梯到达f层，该按钮将熄灭
  - 如果电梯按钮熄灭，则按下它时，按钮将发光
- 电梯按钮事件
  - EBP(e,f)：电梯按钮(e,f)被按下 ➢ Elevator Button is Pressed
  - EAF(e,f)：电梯e到达f层 ➢ Elevator Arrival at Floor

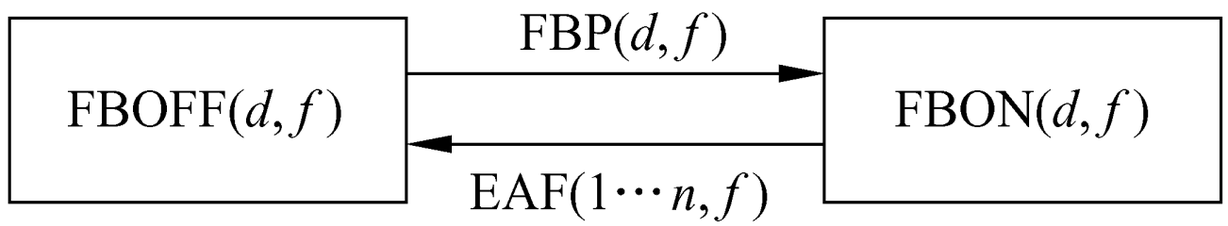


实例 - 电梯按钮

- 为了定义与这些事件和状态相联系的状态转换规则，需要一个谓词V(e,f)，其含义如下：
  - V(e,f)：电梯e停在f层
- 如果电梯按钮(e,f)处于关闭状态〔当前状态〕，而且电梯按钮(e,f)被按下〔事件〕，而且电梯e不在f层〔谓词〕，则该电梯按钮打开发光〔下个状态〕
  - 状态转换规则的形式化描述如下
    - EBOFF(e,f)+EBP(e,f)+not V(e,f) ➢ EBON(e,f)
- 如果电梯到达f层，而且电梯按钮是打开的，于是它就会熄灭
  - 转换规则表示为：
    - EBON(e,f)+EAF(e,f) ➢ EBOFF(e,f)

实例 - 楼层按钮

- 令FB(d,f)表示f层请求电梯向d方向运动的按钮
- FB(d,f)有两个状态
  - FBON(d,f)：楼层按钮(d,f)打开
  - FBOFF(d,f)：楼层按钮(d,f)关闭
- 规则
  - 如果楼层按钮已经打开，而且一部电梯到达f层，则按钮关闭
  - 如果楼层按钮原来是关闭的，被按下后该按钮将打开
- 楼层按钮事件
  - FBP(d,f)：楼层按钮(d,f)被按下 ➢ Floor Button is Pressed
  - EAF(1…n,f)：电梯1或…或n到达f层 ➢ Elevator Arrival at Floor
    - 其中1…n表示或为1或为2…或为n



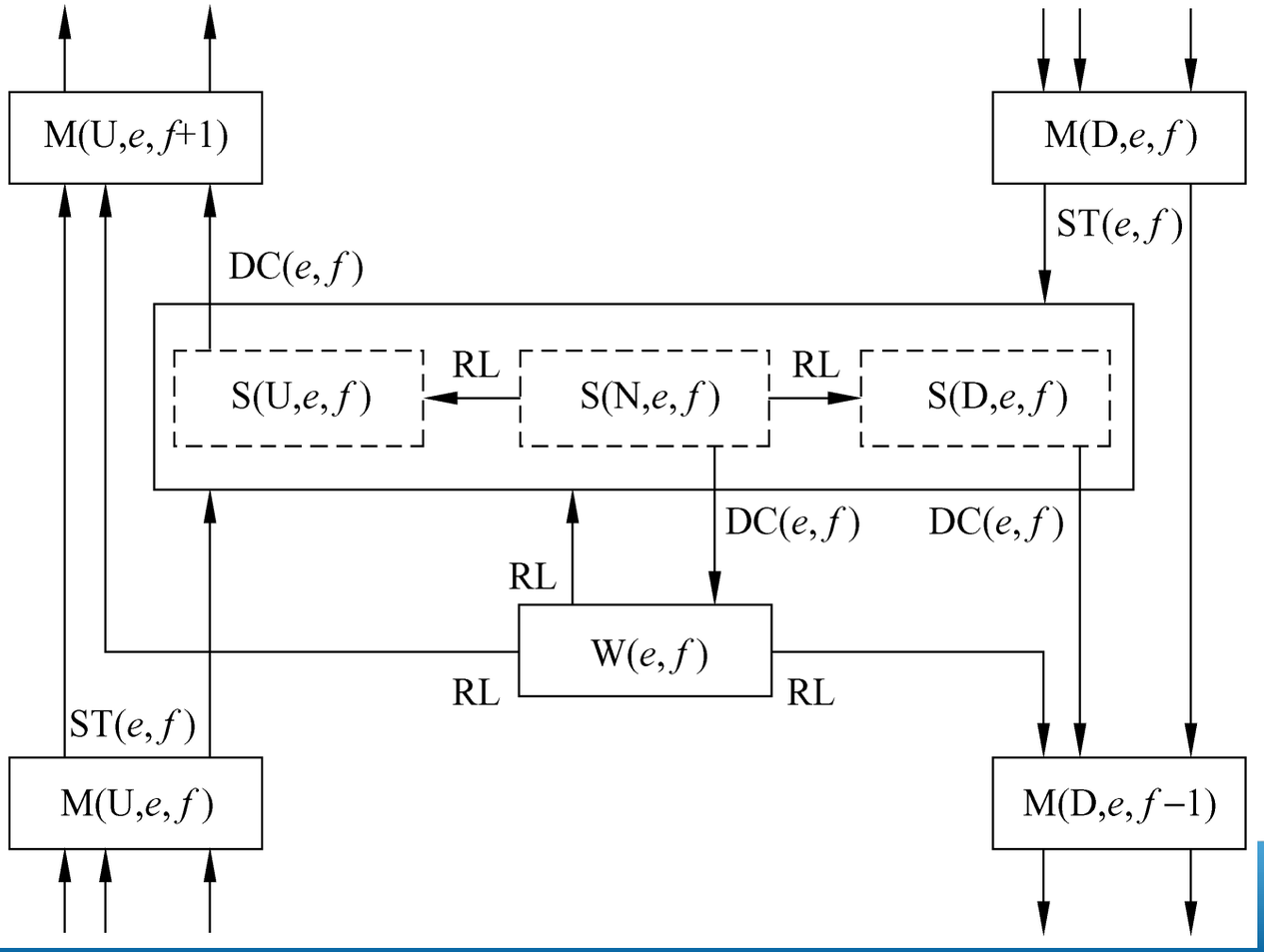
实例 - 楼层按钮

- 定义一个谓词S(d,e,f)，其含义如下：
  - S(d,e,f)：电梯e停在f层并且移动方向由d确定
    - 移动方向d可取为向上(d=U)或向下(d=D)或待定(d=N)
    - 上述谓词实际上是一个状态，形式化方法允许把事件和状态作为谓词对待
- 状态转换规则的形式化描述如下
  - FBOFF(d,f)+FBP(d,f)+not S(d,1…n,f) ➢ FBON(d,f)
  - FBON(d,f)+EAF(1…n,f)+S(d,1…n,f) ➢ FBOFF(d,f)
  - 其中，d=U / D
- 如果在f层请求电梯向d方向运动的楼层按钮处于关闭状态，现在该按钮被按下，并且当时没有正停在f层准备向d方向移动的电梯，则该楼层按钮打开
- 如果楼层按钮已经打开，且至少有一部电梯到达f层，该部电梯将朝d方向运动，则按钮将关闭

实例 - 按钮

- 电梯按钮状态转换规则时定义的谓词V(e,f)，可以用谓词S(d,e,f)重新定义如下
  - V(e,f)=S(U,e,f)or S(D,e,f)or S(N,e,f)
    - 即电梯的三个停止状态
- 现在转向讨论电梯的状态及其转换规则
  - 一个电梯状态包含许多子状态
  - M(d,e,f)：电梯e正沿d方向移动，即将到达的是第f层
  - S(d,e,f)：电梯e停在f层，将朝d方向移动(尚未关门)
  - W(e,f)：电梯e在f层等待(已关门)

- DC(e,f)：电梯e在楼层f关上门
- ST(e,f)：电梯e靠近f层时触发传感器，电梯控制器决定在当前楼层电梯是否停下
- RL：电梯按钮或楼层按钮被按下进入打开状态，登录需求



实例

- 电梯的状态转换
  - 为简单起见，这里给出的规则仅发生在关门之时
  - S(U,e,f)+DC(e,f) ➢ M(U,e,f+1)
    - 如果电梯e停在f层准备向上移动，且门已经关闭，则电梯将向上一楼层移动
  - S(D,e,f)+DC(e,f) ➢ M(D,e,f-1)
  - S(N,e,f)+DC(e,f) ➢ W(e,f)
    - 分别对应于电梯即将下降或者没有待处理的请求的情况

对有穷状态机的评价

- 有穷状态机方法采用了一种简单的格式来描述规格说明
  - 当前状态+事件+谓词 ➢ 下个状态
- 有穷状态机表达形式的规格说明易于书写、易于验证，而且可以比较容易地把它转变成设计或程序代码
  - 可以开发一个CASE工具把一个有穷状态机规格说明直接转变为源代码
  - 维护可以通过重新转变来实现
    - 即如果需要一个新的状态或事件
    - 首先修改规格说明
    - 然后直接由新的规格说明生成新版本的产品
- 有穷状态机方法比数据流图技术更精确，而且和它一样易于理解
- 缺点
  - 在开发一个大系统时三元组(即状态、事件、谓词)的数量会迅速增长
  - 和数据流图方法一样，形式化的有穷状态机方法尚未处理定时需求

PETRI

PETRI的基本作用

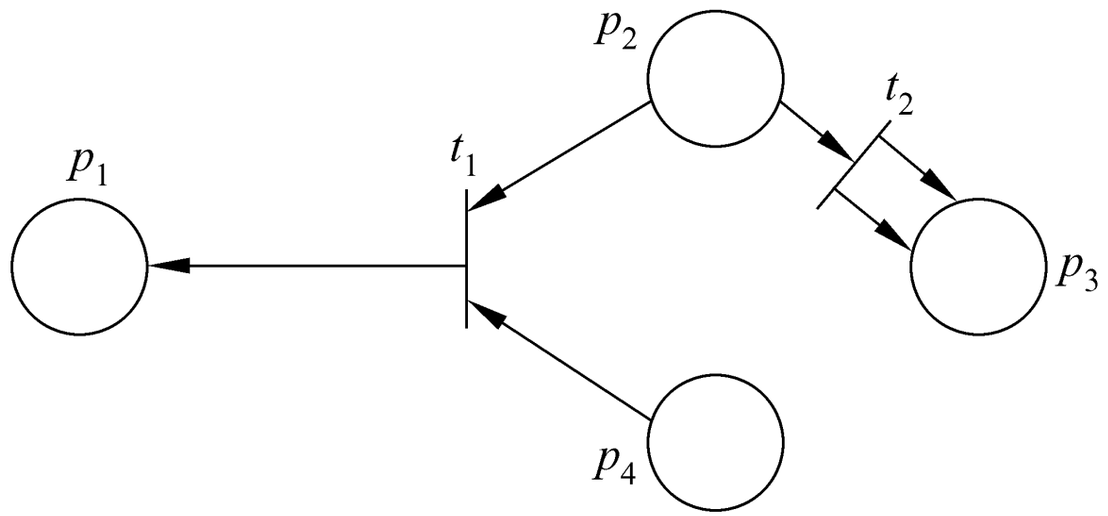
- 并发系统中遇到的一个主要问题是定时问题
  - 该问题可表现为多种形式
    - 如同步问题、竞争条件以及死锁问题
  - 定时问题通常是由不好的设计或有错误的实现引起的
    - 进一步来说，设计或实现通常又是由不好的规格说明造成的
    - 如果规格说明不恰当，则有导致不完善的设计或实现的危险
- 用于确定系统中隐含的定时问题的一种有效技术是Petri网

软件开发模型

- Petri网是由Carl Adam Petri发明的
- 最初只有自动化专家对Petri网感兴趣，后来Petri网在计算机科学中也得到广泛的应用
  - 例如，在性能评价、操作系统和软件工程等领域
- 已经证明，用Petri网可以有效地描述并发活动
- Petri网包含四种元素
  - 一组位置P
  - 一组转换T
  - 输入函数I以及
  - 输出函数O

PETRI网构造

- 一组位置P为｛P1，P2，P3，P4｝
  - 在图中用圆圈代表位置
- 一组转换T为｛t1，t2｝
  - 在图中用短直线表示转换
- 两个用于转换的输入函数
  - 用由位置指向转换的箭头表示，包括：
    - I(t1)=｛P2，P4｝
    - I(t2)=｛P2｝



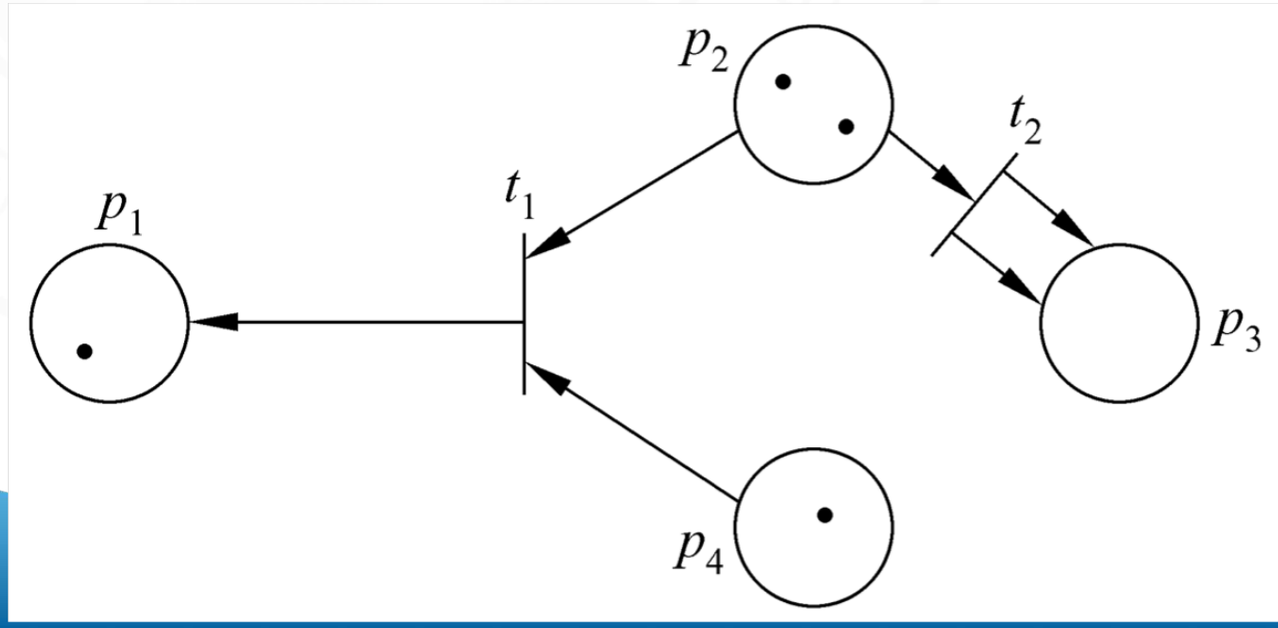
- 两个用于转换的输出函数
  - 用由转换指向位置的箭头表示
  - O(t1)=｛P1｝
  - O(t2)=｛P3，P3｝
    - 输出函数O(t2)中有两个P3，是因为有两个箭头由t2指向P3

形式化的PETRI网

- 一个四元组C = (P, T, I, O)
  - P = {P1，…，Pn}是一个有穷位置集，n≥0
  - T = {t1，…，tm}是一个有穷转换集，m≥0，且T和P不相交
  - I：T→P∞为输入函数，是由转换到位置无序单位组(bags)的映射
  - O：T→P∞为输出函数，是由转换到位置无序单位组的映射
    - 一个无序单位组或多重组是允许一个元素有多个实例的广义集

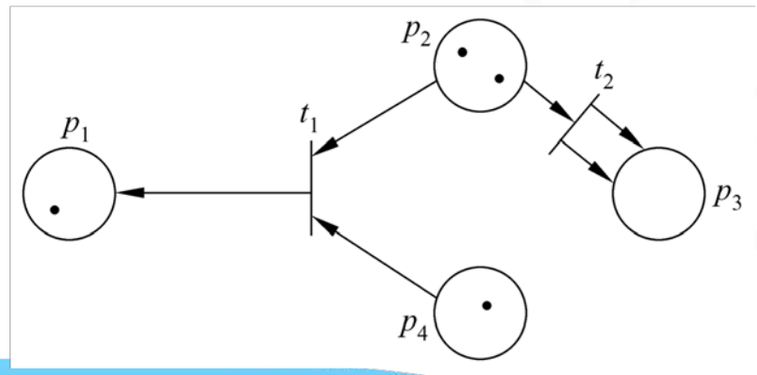
形式化的PETRI网

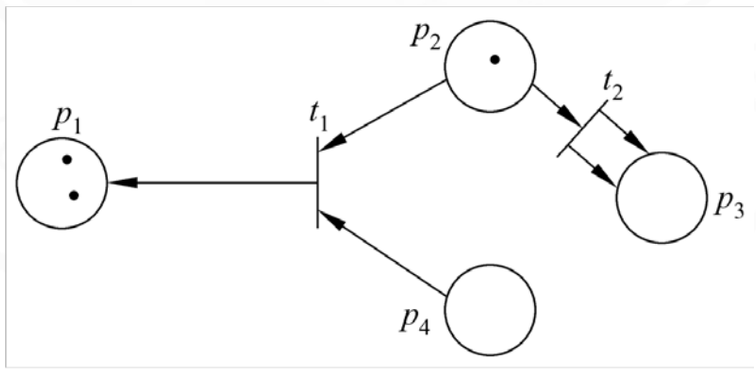
- Petri网的标记是在Petri网中权标(token)的分配
  - 例如，示例图中有4个权标
    - 其中一个在P1中，两个在P2中，P3中没有，另外一个在P4中
    - 上述标记可以用向量(1，2，0，1)表示
    - 由于P2和P4中有权标，因此t1启动(即被激发)
    - 通常，当每个输入位置所拥有的权标数大于等于从该位置到转换的线数时，就允许转换
      - 当t1被激发时，P2和P4上各有一个权标被移出，而P1上则增加一个权标
  - Petri网中权标总数不是固定的，在上述示例中两个权标被移出，而P1上只能增加一个权标

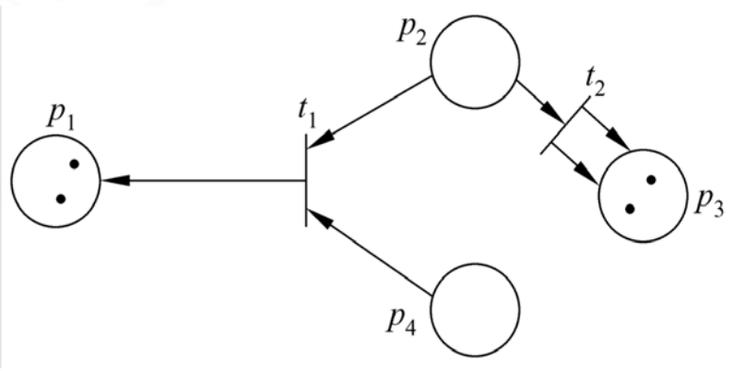


形式化的PETRI网

- Petri网的标记是在Petri网中权标(token)的分配
  - P2上有权标，因此t2也可以被激发
  - 当t2被激发时，P2上将移走一个权标，而P3上新增加两个权标
- Petri网具有非确定性，即，如果数个转换都达到了激发条件，则其中任意一个都可以被激发
  - 图中Petri网的标记为(1，2，0，1)，t1和t2都可以被激发
    - 假设t1被激发了，则结果如下中图所示，标记为(2，1，0，0)
    - 此时，只有t2可以被激发
    - 如果t2也被激发了，则权标从P2中移出，两个新权标被放在P3上，结果标记为(2，0，2，0)

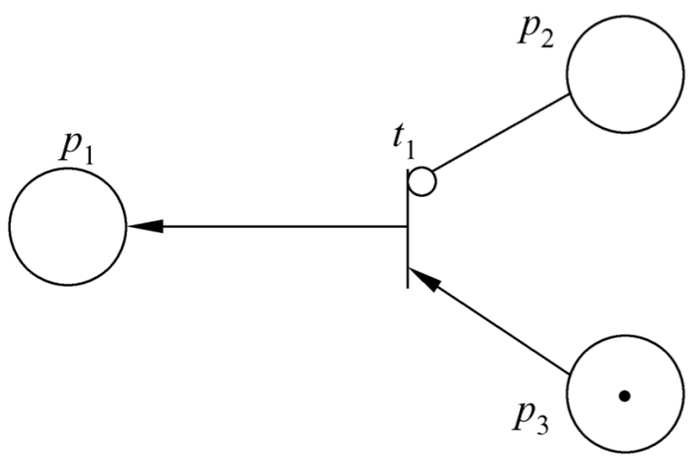






形式化的PETRI网

- 更形式化地说，Petri网C=(P，T，I，O)中的标记M，是由一组位置P到一组非负整数的映射：
  - M：P→｛0，1，2，…｝
  - 则带有标记的Petri网成为一个五元组(P，T，I，O，M)。
- 对Petri网的一个重要扩充是加入禁止线
  - 禁止线是用一个小圆圈而不是用箭头标记的输入线
  - 通常，当每个输入线上至少有一个权标，而禁止线上没有权标的时候，相应的转换才是允许的
  - 示例图中，P3上有一个权标而P2上没有权标，因此转换t1可以被激发



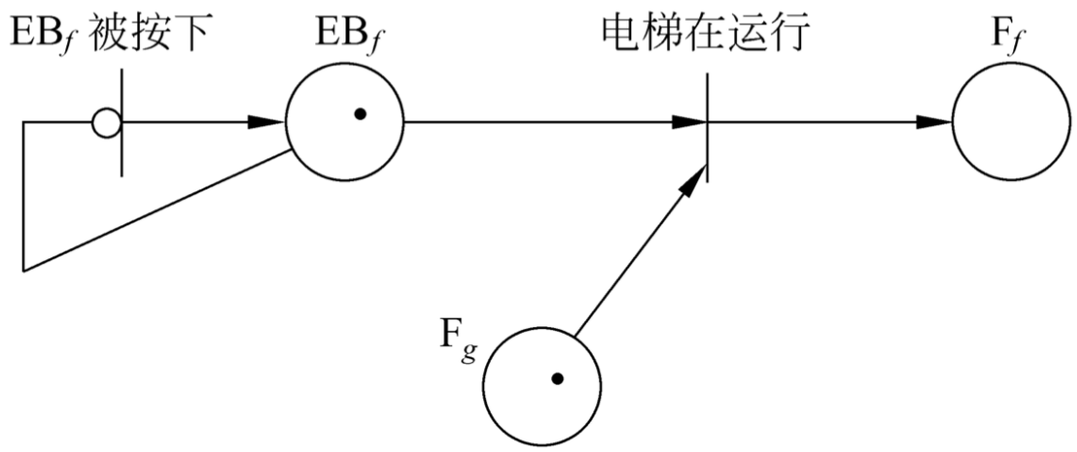
实例

- 当用Petri网表示电梯系统的规格说明时，每个楼层用一个位置Ff代表(1≤f≤m)，在Petri网中电梯是用一个权标代表的。在位置Ff上有权标，表示在楼层f上有电梯

- 在一幢m层的大厦中需要一套控制n部电梯的产品，要求这n部电梯按照约束条件C1，C2和C3在楼层间移动
  - C1
    - 每部电梯内有m个按钮，每个按钮代表一个楼层
    - 当按下一个按钮时该按钮指示灯亮，同时电梯驶向相应的楼层，到达按钮指定的楼层时指示灯熄灭
  - C2
    - 除了大厦的最低层和最高层之外，每层楼都有两个按钮分别请求电梯上行和下行
    - 这两个按钮之一被按下时相应的指示灯亮，当电梯到达此楼层时灯熄灭，电梯向要求的方向移动
  - C3
    - 当对电梯没有请求时，它关门并停在当前楼层

实例 - 电梯按钮

- 第一条约束C1
  - 每部电梯有m个按钮，每层对应一个按钮
  - 当按下一个按钮时该按钮指示灯亮，指示电梯移往相应的楼层
  - 当电梯到达指定的楼层时，按钮将熄灭
- 为了用Petri网表达电梯按钮的规格说明，在Petri网中还必须设置其他的位置
  - 电梯中楼层f的按钮，在Petri网中用位置EBf表示(1≤f≤m)
  - 在EBf上有一个权标，就表示电梯内楼层f的按钮被按下了
  - 电梯按钮只有在第一次被按下时才会由暗变亮，以后再按它则只会被忽略
- 描述
  - 假设按钮没有发亮，显然在位置EBf上没有权标，从而在存在禁止线的情况下，转换“EBf被按下”是允许发生的
  - 假设现在按下按钮，则转换被激发并在EBf上放置了一个权标，之后不论再按下多少次按钮，禁止线与现有权标的组合都决定了转换“EBf被按下”不能再被激发了，因此，位置EBf上的权标数不会多于1

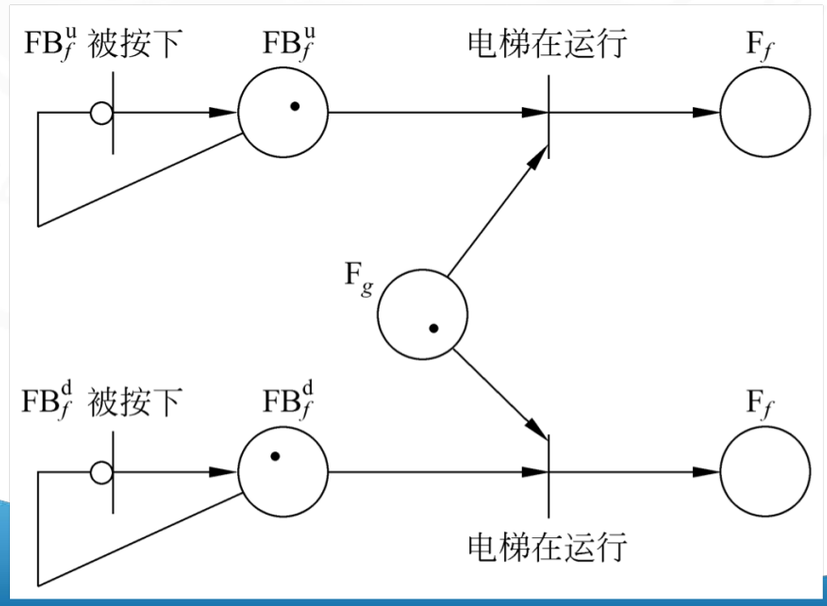


实例 - 电梯按钮

- 假设电梯由g层驶向f层，因为电梯在g层，位置Fg上有一个权标
  - 由于每条输入线上各有一个权标，转换“电梯在运行”被激发，从而EBf和Fg上的权标被移走，按钮EBf被关闭，在位置Ff上出现一个新权标，即转换的激发使电梯由g层驶到f层
- 事实上，电梯由g层移到f层是需要时间的，为处理这个情况及其他类似的问题(例如，由于物理上的原因按钮被按下后不能马上发亮)，Petri网模型中必须加入时限
  - 即，在标准Petri网中转换是瞬时完成的，而在现实情况下就需要时间控制Petri网，以使转换与非零时间相联系

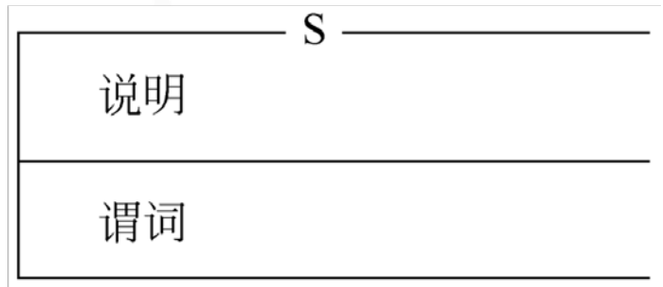
实例 - 楼层按钮

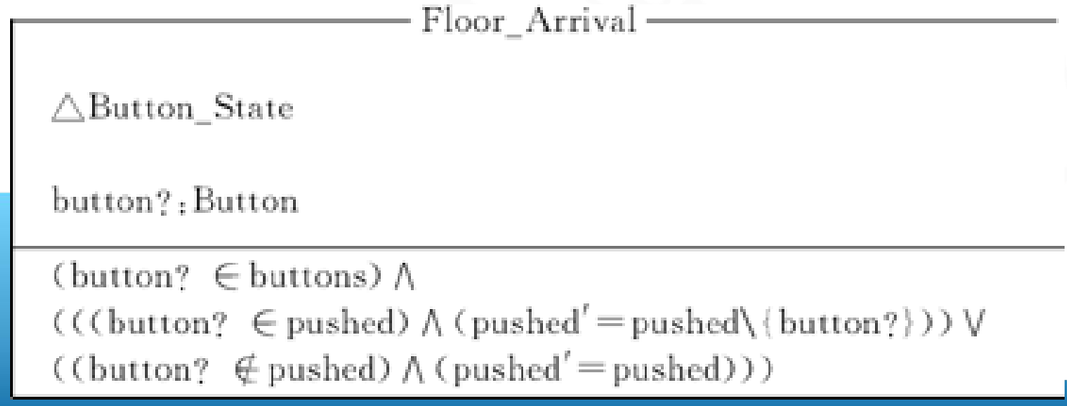
- 第二条约束C2
  - 除了第一层与顶层之外，每个楼层都有两个按钮，一个要求电梯上行，另一个要求电梯下行；这些按钮在按下时发亮，当电梯到达该层并将向指定方向移动时，相应的按钮才会熄灭
- 在 Petri网中楼层按钮用位置FBf^u和FBf^d表示，分别代表f楼层请求电梯上行和下行的按钮 / 底层的
  - 按钮为$FB1_u$，最高层的按钮为FBm_d，中间每一层有两个按钮FBf_u 和FBf_d $(1 < f < m)$
  - 当电梯由g层驶向f层，根据电梯乘客的要求，某一个楼层按钮亮或两个楼层按钮都亮
    - 如果两个按钮都亮了，则只有一个按钮熄灭
    - 如图的Petri网可以保证，当两个按钮都亮了的时候，只有一个按钮熄灭
      - 但是要保证按钮熄灭正确，则需要更复杂的Petri网模型
- 第三条约束C3
  - 当没有请求(FBf_u和FBf_d上无权标）时，任何一个转换“电梯在运行”都不能被激发

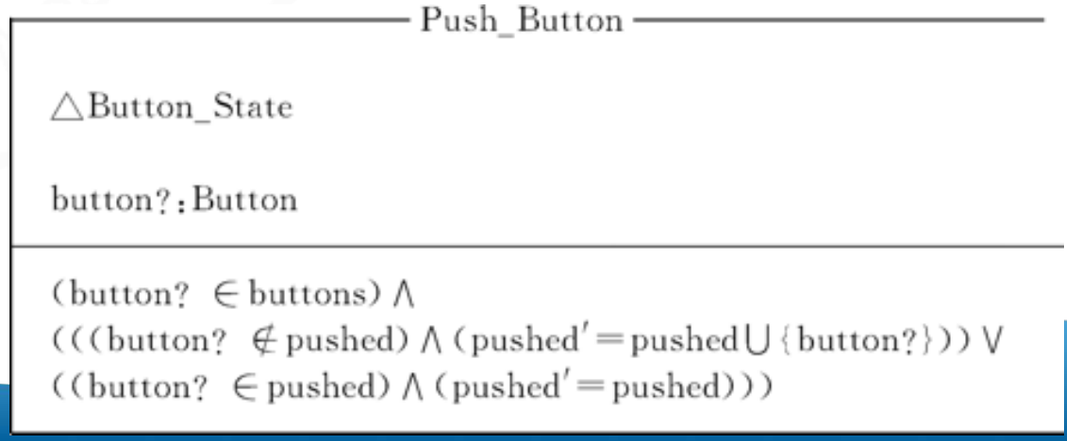


Z语言

- 用Z语言描述的、最简单的形式化规格说明含有4个部分
  - 给定的集合、数据类型及常数
    - 一个Z规格说明从一系列给定的初始化集合开始
    - 所谓初始化集合就是不需要详细定义的集合，该集合用带方括号的形式表示。
      - 对于电梯问题，给定的初始化集合称为Button，即所有按钮的集合，因此，Z规格说明开始于：〔Button〕
  - 状态定义
    - 一个Z规格说明由若干个“格(schema)”组成，每个格含有一组变量说明和一系列限定变量取值范围的谓词
      - 例如，格S的格式见示意图
  - 初始状态
    - 抽象的初始状态是指系统第一次开启时的状态
    - 对于电梯问题来说，抽象的初始状态为：
      - Button_Init≙〔Button_State | pushed=Φ〕
      - 上式表示，当系统首次开启时pushed集为空，即所有按钮都处于关闭状态
  - 操作
    - 如果一个原来处于关闭状态的按钮被按下，则该按钮开启，这个按钮就被添加到pushed集中
    - 如果电梯到达了某楼层，如果相应的楼层按钮已经打开，则此时它会关闭；同样，如果相应的电梯按钮已经打开，则此时它也会关闭。也就是说，如果“button?”属于pushed集，则将它移出该集合







实例

- 在电梯问题中，Button有4个子集
  - floor_buttons(楼层按钮的集合)
  - elevator_buttons(电梯按钮的集合)
  - buttons(电梯问题中所有按钮的集合)
  - pushed(所有被按的按钮的集合，即所有处于打开状态的按钮的集合)

Button_State

floor_buttons, elevator_buttons : P Button
buttons : P Button
pushed : P Button

floor_buttons ∩ elevator_buttons = Φ

floor_buttons ∪ elevator_buttons = buttons

- Bu
  - 其中，符号P表示幂集(即给定集的所有子集)
- 约束条件声明
  - floor_buttons集与elevator_buttons集不相交
  - 而且它们共同组成buttons集

实例

- 操作的谓词部分，包含了一组调用操作的前置条件，以及操作完全结束后的后置条件
  - 如果前置条件成立，则操作执行完成后可得到后置条件
  - 但是，如果在前置条件不成立的情况下调用该操作，则不能得到指定的结果(因此结果无法预测)。
    - 如果电梯到达了某楼层，如果相应的楼层按钮已经打开，则此时它会关闭
    - 如果相应的电梯按钮已经打开，则此时它也会关闭
      - 即，如果“button?”属于pushed集，则将它移出该集合

Floor_Arrival

△Button_State

button? : Button

$$
\begin{aligned}
&(button? \in buttons) \land \\
&(((button? \in pushed) \land (pushed' = pushed \setminus \{button?\})) \lor \\
&((button? \notin pushed) \land (pushed' = pushed))
\end{aligned}
$$

Push_Button

△Button_State

button? : Button

$$
\begin{aligned}
&(button? \in buttons) \land \\
&(((button? \notin pushed) \land (pushed' = pushed \cup \{button?\})) \lor \\
&((button? \in pushed) \land (pushed' = pushed)))
\end{aligned}
$$

Z语言的评价

- 已经在许多软件开发项目中成功地运用了z语言，
  - 目前，z也许是应用得最广泛的形式化语言，尤其是在大型项目中z语言的优势更加明显
- z语言获得使用许可
  - 可以比较容易地发现用z写的规格说明的错误，特别是在自己审查规格说明，及根据形式化的规格说明来审查设计与代码时，情况更是如此
  - 用z写规格说明时，要求作者十分精确地使用z说明符
    - 由于对精确性的要求很高，从而和非形式化规格说明相比，减少了模糊性、不一致性和遗漏
  - z是一种形式化语言，在需要时开发者可以严格地验证规格说明的正确性
  - 完全学会z语言相当困难
  - 使用z语言可以降低软件开发费用
    - 虽然用z写规格说明所需用的时间比使用非形式化技术要多，但开发过程所需要的总时间却减少了
  - 虽然用户无法理解用z写的规格说明，但是，可以依据z规格说明用自然语言重写规格说明