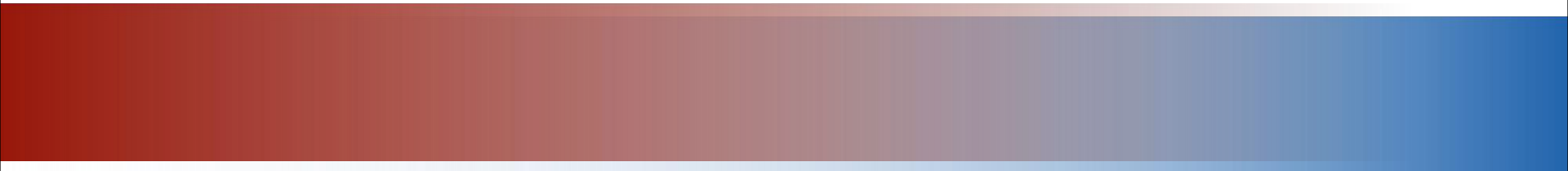


软件工程



第四章 形式化说明技术

❖ Why Formal?

- { 自然语言：矛盾、二义性、含糊
- { 形式化表示：准确、清晰
- { 非形式化：自然语言
- { 半形式化：**DFD**、**E-R**图
- { 形式化：坚实的以数学为基础的表达方法

第四章 形式化说明技术



4.1 概述

❖ 4.1.1 非形式化方法的缺点

- 矛盾：相互冲突的陈述
- 二义性：读者可以用不同的方式理解的陈述
- 含糊
- 不完整
- 抽象层次混乱：非常抽象的陈述中混进了一些关于细节的低层次陈述。

4.1 概述

❖ 4.1.2 形式化方法的优点

- 把数学引入软件开发过程，形成基于数学的形式化方法。
- 数学最有用的一个性质是，能够简洁准确地描述物理现象、对象或动作的结果，因此是理想的建模工具。
- 在软件开发过程中使用数学，可以在不通的软件工程活动之间平滑地过渡。
- 提供了高层确认的手段。

4.1 概述

❖ 4.1.3 应用形式化方法的准则

- 应该选用适当的表示方法
- 应该形式化，但不要过分形式化
- 应该估算成本
- 应该有形式化方法顾问随时提供咨询
- 不应该放弃传统的开发方法
- 应该建立详尽的文档
- 不应该放弃质量标准
- 不应该盲目依赖形式化方法
- 应该测试、测试再测试
- 应该重用

4.2 有穷状态机

❖ 4.2.1 概念

- ❖ 一个保险箱上装了一个复合锁，锁有**3**个位置，分别标记为**1**、**2**、**3**，转盘可向左（**L**）或向右（**R**）转动。
- ❖ 这样，任何时刻转盘都有**6**种可能的运动，即**1L**、**1R**、**2L**、**2R**、**3L**、**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。

4.2 有穷状态机

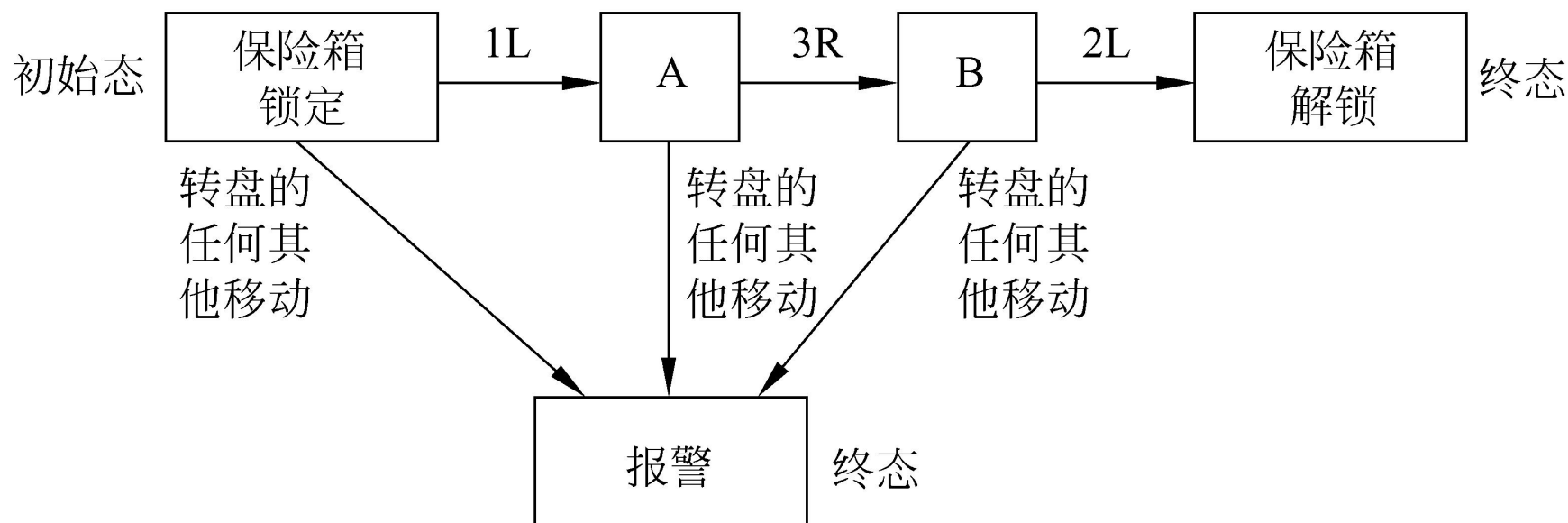


图4.1 保险箱的状态转换图

4.2 有穷状态机

表 4.1 保险箱的状态转换表

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

4.2 有穷状态机

❖ 一个有穷状态机包括**5**个部分：

- 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4.1所示
- 初始态S: 保险箱锁定
- 终态集F: {保险箱解锁, 报警}

4.2 有穷状态机

- ❖ 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** \in **J**，是一个初始状态；
 - **F** \subseteq **J**，是终态集

4.2 有穷状态机

- ❖ 加入谓词集**P**，把有穷状态机扩展为一个**6**元组，其中每个谓词都是系统全局状态**Y**的函数。
- ❖ 则转换函数**T**:

$$(J-F) * K * P \Rightarrow J$$

4.2 有穷状态机

❖ 4.2.2 例子

- 电梯按钮

- $EB(e, f)$: 按下电梯 e 内的按钮并请求到 f 层去

- 状态:

- $EBON(e, f)$: 电梯按钮 (e, f) 打开
 - $EBOFF(e, f)$: 电梯按钮 (e, f) 关闭

- 事件

- $EBP(e, f)$: 电梯按钮 (e, f) 被按下
 - $EAF(e, f)$: 电梯到达 f 层

- 谓词

- $V(e, f)$: 电梯 e 停在 f 层

4.2 有穷状态机

$EBOFF(e,f) + EBP(e,f) + \text{not } V(e,f) \Rightarrow EBON(e,f)$

$EBON(e,f) + EAF(e,f) \Rightarrow EBOFF(e,f)$

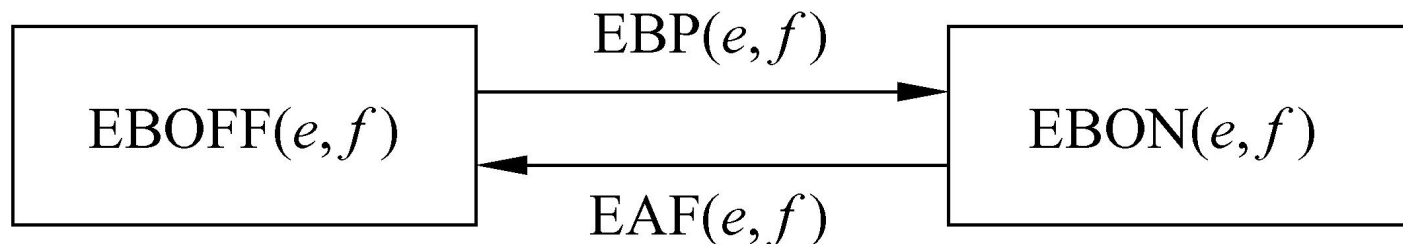


图4.2 电梯按钮的状态转换图

4.2 有穷状态机

- 楼层按钮
 - $FB(d, f)$: f 层请求电梯向 d 方向运动的按钮
- 状态:
 - $FBON(d, f)$: 楼层按钮 (d, f) 打开
 - $FBOFF(d, f)$: 楼层按钮 (d, f) 关闭
- 事件
 - $FBP(d, f)$: 楼层按钮 (d, f) 被按下
 - $EAF(1...n, f)$: 电梯1或...或 n 到达 f 层
- 谓词
 - $S(d, e, f)$: 电梯 e 停在 f 层并且移动方向由 d 确定

4.2 有穷状态机

$\text{FBOFF}(d,f) + \text{FBP}(d,f) + \text{not } S(d,1 \cdots n,f) \Rightarrow \text{FBON}(d,f)$

$\text{FBON}(d,f) + \text{EAF}(1 \cdots n,f) + S(d,1 \cdots n,f) \Rightarrow \text{FBOFF}(d,f)$

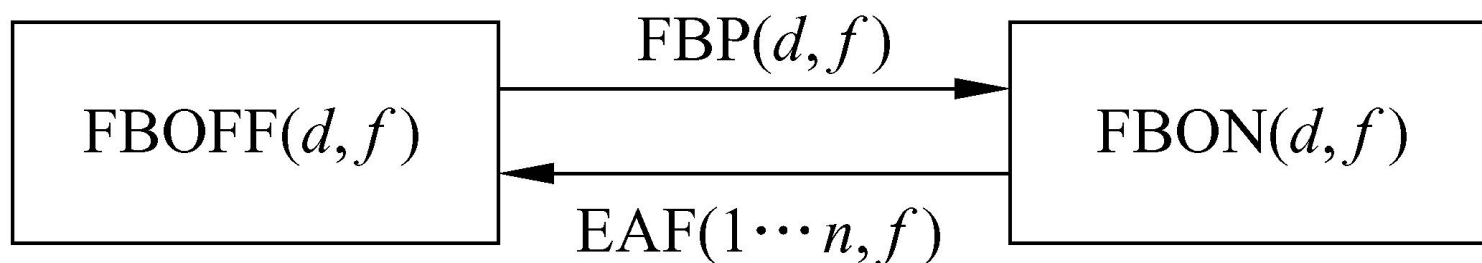


图4.3 楼层按钮的状态转换图

4.2 有穷状态机

- 电梯
- 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)
- 事件
 - $DC(e, f)$: 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

4.2 有穷状态机

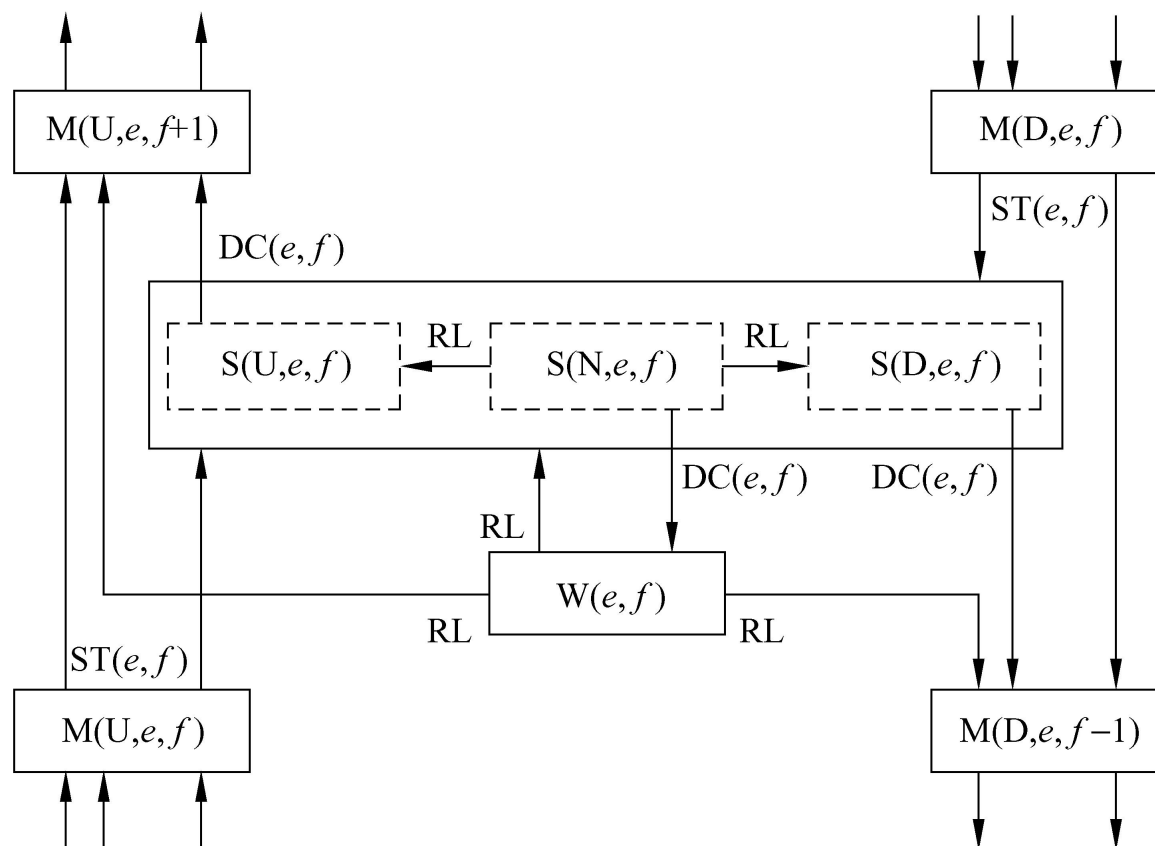


图4.4 电梯的状态转换图

4.2 有穷状态机

❖ 关门之时的规则

$$S(U,e,f)+DC(e,f) \Rightarrow M(U,e,f+1)$$

$$S(D,e,f)+DC(e,f) \Rightarrow M(D,e,f-1)$$

$$S(N,e,f)+DC(e,f) \Rightarrow W(e,f)$$

思考题

❖ 定义状态:

❖ **O (e, f)** : 电梯**e**在**f**层开门, 则如何进入这个状态

$$M(d,e,f)+ST(e,f)+RL \Rightarrow O(e,f)$$

$$W(e,f)+FBP(d,f') \Rightarrow O(e,f) \ (f'=f)$$

❖ 从等待到移动, 如何表示

$$W(e,f)+FBP(d,f') \Rightarrow M(U,e,f+1) \ (f'>f)$$

$$W(e,f)+FBP(d,f') \Rightarrow M(D,e,f-1) \ (f'<f)$$

4.2 有穷状态机

❖ 4.2.3 评价

- 有穷状态机描述规格说明：
 - 当前状态+事件+谓词 \Rightarrow 下个状态
- 优点：
 - 易于书写、易于验证、精确、易于理解
- 缺点：
 - 开发大系统时三元组的数量会迅速增长
 - 没有处理定时的需求

4.3 Petri网

❖ 4.3.1 概述

❖ 并发系统中遇到的一个主要问题是定时问题

- 同步问题
- 竞争条件
- 死锁

❖ **Petri**网是一种用于确定系统中隐含的定时问题的一种有效技术

❖ 可有效的描述并发活动

4.3 Petri网

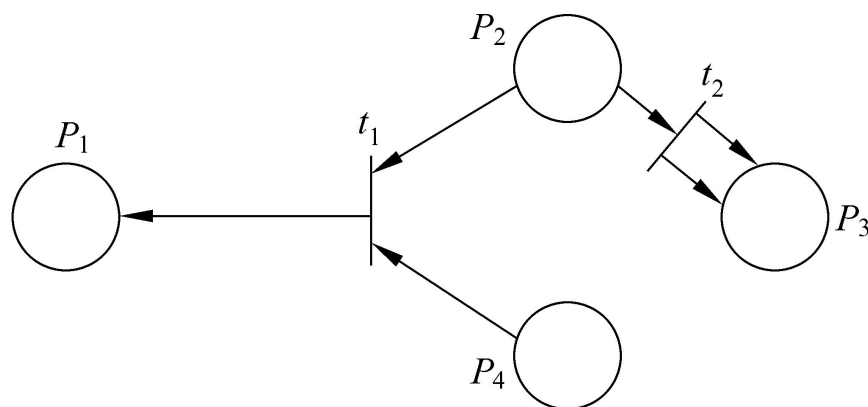


图4.5 Petri网的组成

❖ 包含元素

■ 位置P:

- $\{P_1, P_2, P_3, P_4\}$

■ 转换T:

- $\{t_1, t_2\}$

■ 输入函数I:

- $I(t_1) = \{P_2, P_4\}$
- $I(t_2) = \{P_2\}$

■ 输出函数O

- $O(t_1) = \{P_1\}$
- $O(t_2) = \{P_3, P_3\}$

4.3 Petri网

❖ 更形式化的**Petri**网结构，是一个四元组

- $C = (P, T, I, O)$

- 其中，

- $P = \{P_1, \dots, P_n\}$ 是一个有穷位置集， $n \geq 0$
- $T = \{t_1, \dots, t_m\}$ 是一个有穷转换集， $m \geq 0$ ，且 T 和 P 不相交。
- $I: T \rightarrow P^\infty$ 为输入函数，是由转换到位置无序单位组的映射。
- $O: T \rightarrow P^\infty$ 为输出函数，是由转换到位置无序单位组的映射。

4.3 Petri网

❖ Petri网的标记是在Petri网中权标(token)的分配。

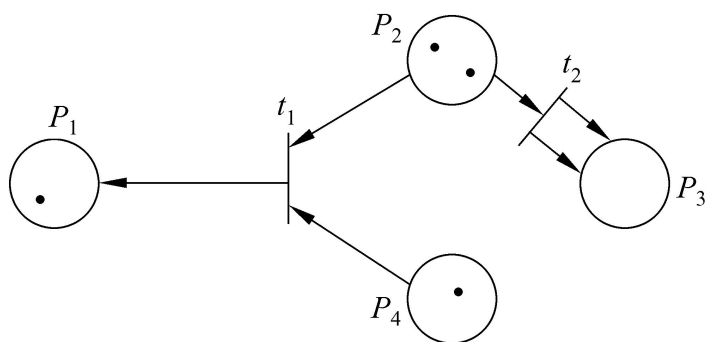


图4.6 带标记的Petri网 (1, 2, 0, 1)

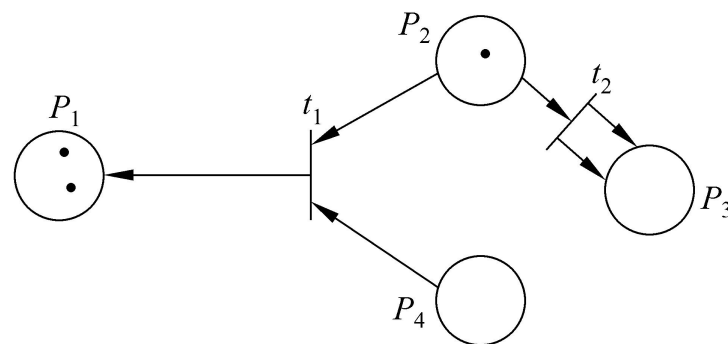


图4.7 图4.6的Petri网在转换t1被激发后的情况 (2, 1, 0, 0)

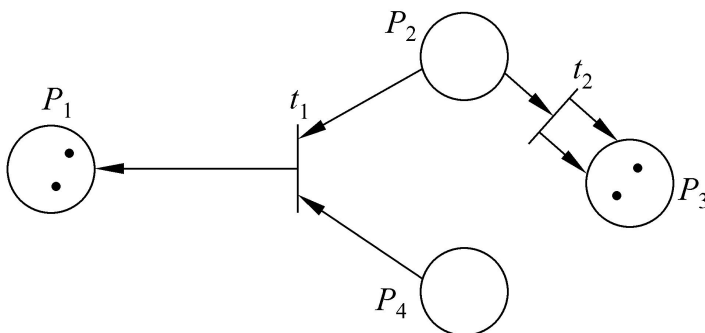


图4.8 图4.7的Petri网在转换t2被激发后的情况 (2, 0, 2, 0)

4.3 Petri网

- ❖ 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令牌，而禁止线上没有令牌的时候，相应的转换才是允许的。

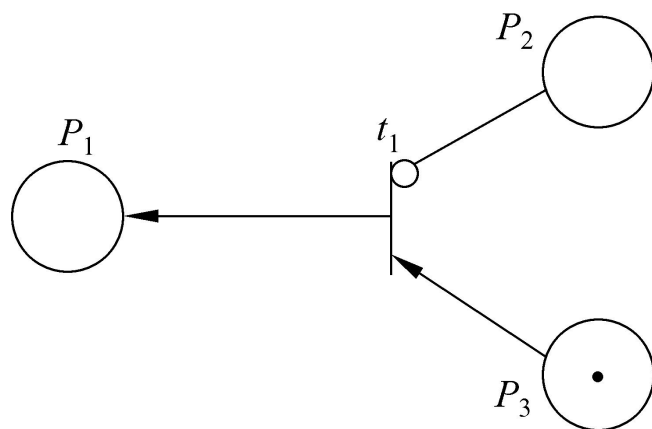


图4.9 含禁止线的**Petri**网

4.3 Petri网

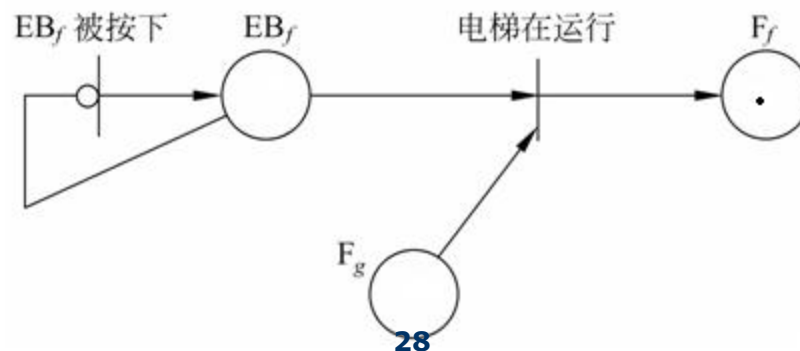
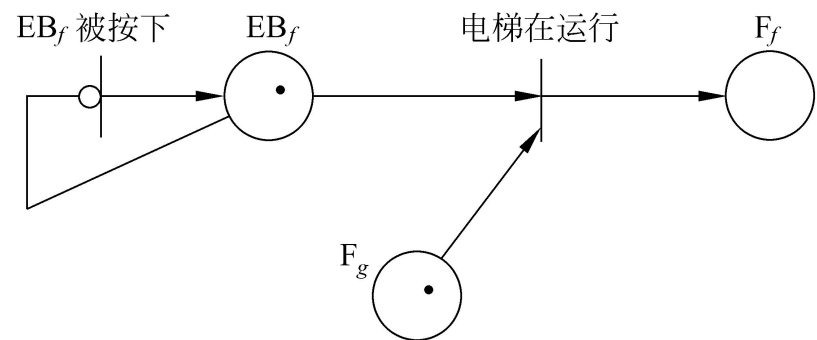
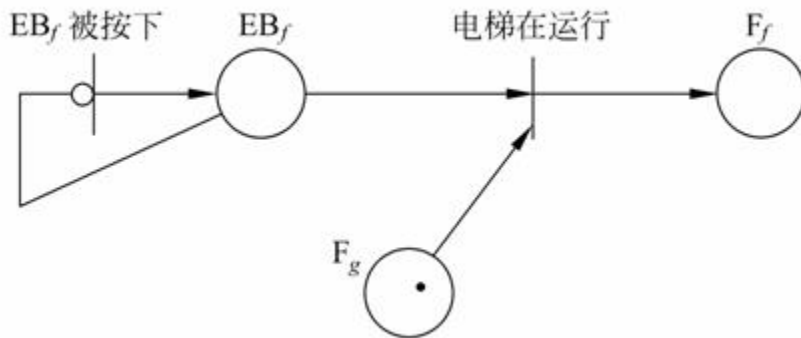
❖ 4.3.2 例子

- ❖ 每个楼层用一个位置 F_f 代表 ($1 \leq f \leq m$)
- ❖ 电梯是用一个权标代表。
- ❖ 在位置 F_f 上有权标，表示在楼层 f 上有电梯。 葛

4.3 Petri网

■ 1. 电梯按钮

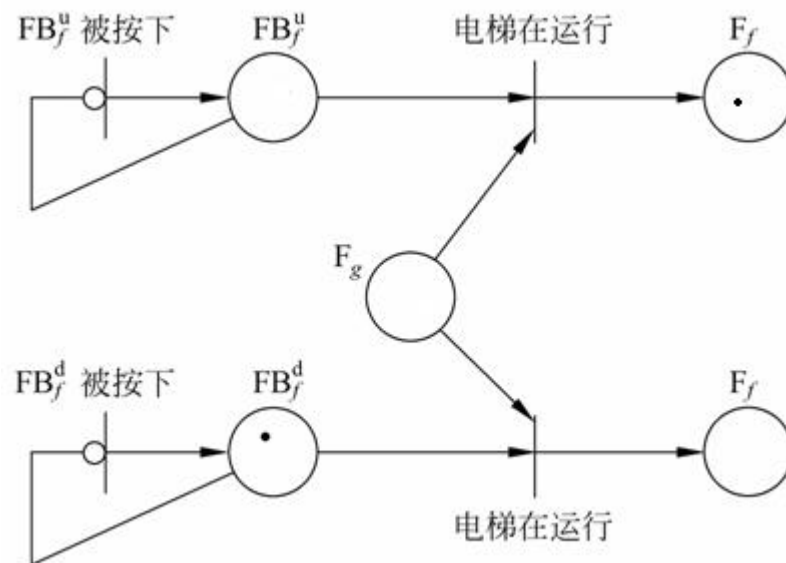
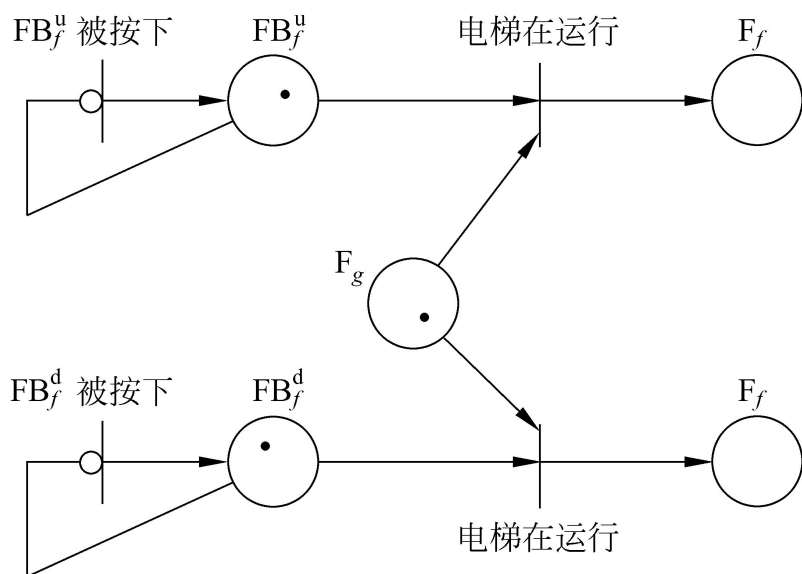
- 电梯中楼层 f 的按钮，在Petri网中用位置 EB_f 表示 ($1 \leq f \leq m$)。在 EB_f 上有一个令牌，就表示电梯内楼层 f 的按钮被按下了。



4.3 Petri网

■ 2.楼层按钮

- 在Petri网中楼层按钮用位置 FB_f^u 和 FB_f^d 表示，分别代表f楼层请求电梯上行和下方的按钮。



4.4 Z语言

❖ 4.4.1 简介

- 用Z语言描述的、最简单的形式化规格说明含有下述4个部分。⌘
 - 给定的集合、数据类型及常数。⌘
 - 状态定义。⌘
 - 初始状态。⌘
 - 操作。⌘

4.4 Z语言

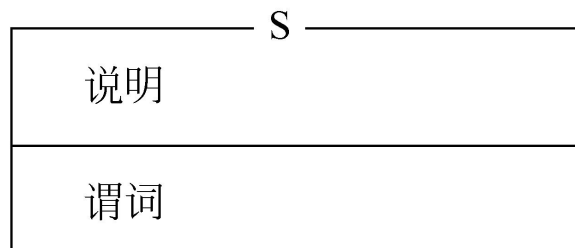
❖ 1. 给定的集合

- 从一系列给定的初始化集合开始。
- 即不需要详细定义的集合，这种集合用带方括号的形式表示。
- {Button}

4.4 Z语言

❖ 2. 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。



4.4 Z语言

- 图4.13描述了格Button_State
- 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

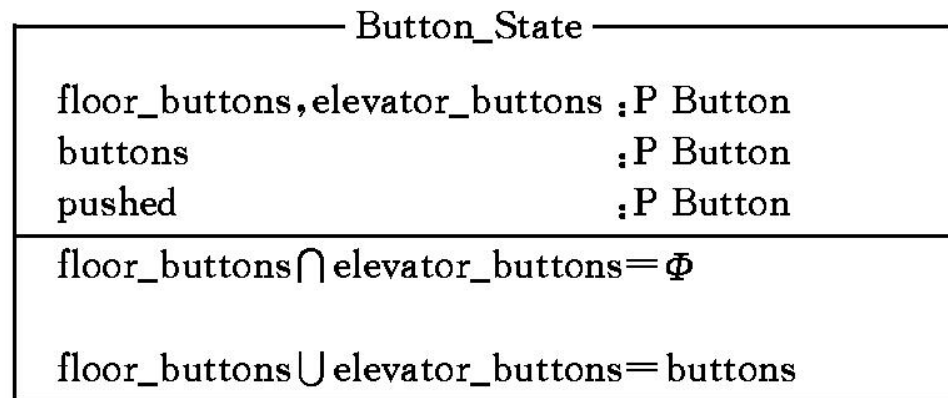


图4.13 Z格Button_State

4.4 Z语言

❖ 3. 初始状态

- 抽象的初始状态是指系统第一次开启时的状态。
- $\text{Button_Init} \triangleq \{ \text{Button_State} \mid \text{pushed} = \emptyset \}$

❖ 4. 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。

Push_Button
$\Delta \text{Button_State}$
$\text{button?} : \text{Button}$
$(\text{button?} \in \text{buttons}) \wedge$ $(((\text{button?} \notin \text{pushed}) \wedge (\text{pushed}' = \text{pushed} \cup \{\text{button?}\})) \vee$ $((\text{button?} \in \text{pushed}) \wedge (\text{pushed}' = \text{pushed})))$

4.4 Z语言

- 假设电梯到达了某楼层，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。

Floor_Arrival	
Δ Button_State	
button?:Button	
$(\text{button?} \in \text{buttons}) \wedge$ $((\text{button?} \in \text{pushed}) \wedge (\text{pushed}' = \text{pushed} \setminus \{\text{button?}\})) \vee$ $((\text{button?} \notin \text{pushed}) \wedge (\text{pushed}' = \text{pushed}))$	

4.4 Z语言

❖ 4.4.2 评价

- Z作为应用得最广泛的形式化语言，在大型项目中的优势更加明显。
- Z语言之所以会获得如此多的成功，主要有以下几个原因：
 - (1) 可以比较容易地发现用Z写的规格说明的错误；
 - (2) 用Z写规格说明时，要求作者十分精确地使用Z说明符。

4.4 Z语言

- (3) 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。
- (4) 虽然完全学会Z语言相当困难，但是，经验表明，只学过中学数学的软件开发人员仍然可以只用比较短的时间就学会编写Z规格说明，当然，这些人还没有能力证明规格说明的结果是否正确。葛
- (5) 使用Z语言可以降低软件开发费用。
- (6) 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。经验证明，这样得到的自然语言规格说明，比直接用自然语言写出的非形式化规格说明更清楚、更正确。葛

4.4 Z语言

- ❖ 使用形式化规格说明是全球的总趋势，过去，主要是欧洲习惯于使用形式化规格说明技术，现在越来越多的美国公司也开始使用形式化规格说明技术。葛

第四章 形式化说明技术



谢谢~!