第四章 形式化说明技术

4.1 概述
4.2 有穷状态机
4.3 Petri网
4.4 Z语言
4.5 小结

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
  - 非形式化是指用自然语言描述软件需求（如系统规格说明书）。因此，可能存在矛盾性、二义性、含糊性、不完整性、抽象层次混乱等问题。
- 4.1.2 形式化方法的优点
  - 形式化方法是指在软件工程中引入数学方法和模型。利用数序的简洁性、严谨性、科学性，克服非形式化方法的缺点。
- 4.1.3 应用形式化方法的准则
  - “软件需求”是软件产品的高层、概念模型，而“形式化方法”是严谨、科学的数学方法，因此，在形式化方法的应用方面应考虑适度性、实用性、实用性。

- 常用的形式化方法
  - 较严格的形式化方法（语法和语义都严谨）：有穷状态机、Petri网、Z语言……
  - 半形式化方法（语法和语义都不太严谨）：系统流程图、数据流图、数据字典、ER图、数据库范式、状态转换图、层次方框图、Warnier图、IPO图、IPO表……

4.1 概述——非形式化方法的缺点

矛盾性:

- 在需求规格说明书中对同一问题前后存在不同的描述。

二义性

- 对同一问题的描述存在不同的理解。

含糊性

- 对某一问题的描述不清晰、不可理解、不知如何实现、不具可操作性。

不完整性

- 对某一问题的描述不完整。只说明了局部，没有说明整体；或者只说明了概要，未说明细节。因此不具可操作性。

抽象层次混乱

- 在不同层次的抽象模型中内容混乱，如在高层模型中混有底层细节，造成读者不能理解系统的整体功能和下级功能。

4.1 概述——非形式化方法的优点

- 可严谨地描述软件需求中的问题
  - 可简介、准确描述物理现象、对象或动作的结果问题；适用于描述详细的需求规格；可用数学方法验证需求。
- 可在软件工程不同阶段平滑过渡
  - 从需求、到设计、到实现都基于同一系统模型，平滑过渡。
- 可提供高层确认手段
  - 可用数学方法证明软件工程各阶段的正确性（可回溯性），如“设计”符合“规格说明”、“编码实现”符合“设计”。

- 形式化方法的适用性问题
  - 形式化方法能较好地解决需求的“二义性”、“含糊性”问题。但不能解决需求的矛盾性、完整性等问题，这些问题涉及工程管理。

4.1 概述——应用形式化方法的准则

- 应该选用适当的规格说明表示方法
- 应该采用形式化，但不要过分形式化
- 应该估算推行形式化的成本
- 应该引入形式化方法的顾问与咨询
- 应该结合传统的、证明有效的开发方法
- 应该在采用形式化的同时，建立详尽的文档
- 应该坚持质量保障活动
- 应该不总是盲目依赖形式化方法
- 应该重视测试
- 应该重视重用

第四章 形式化说明技术

4.1 概述

4.2 有穷状态机

4.3 Petri网

4.4 Z语言

4.5 小结

4.2 有穷状态机

- 4.2.1 有穷状态机概念
- 4.2.2 有穷状态机例子
- 4.2.3 有穷状态机方法评价

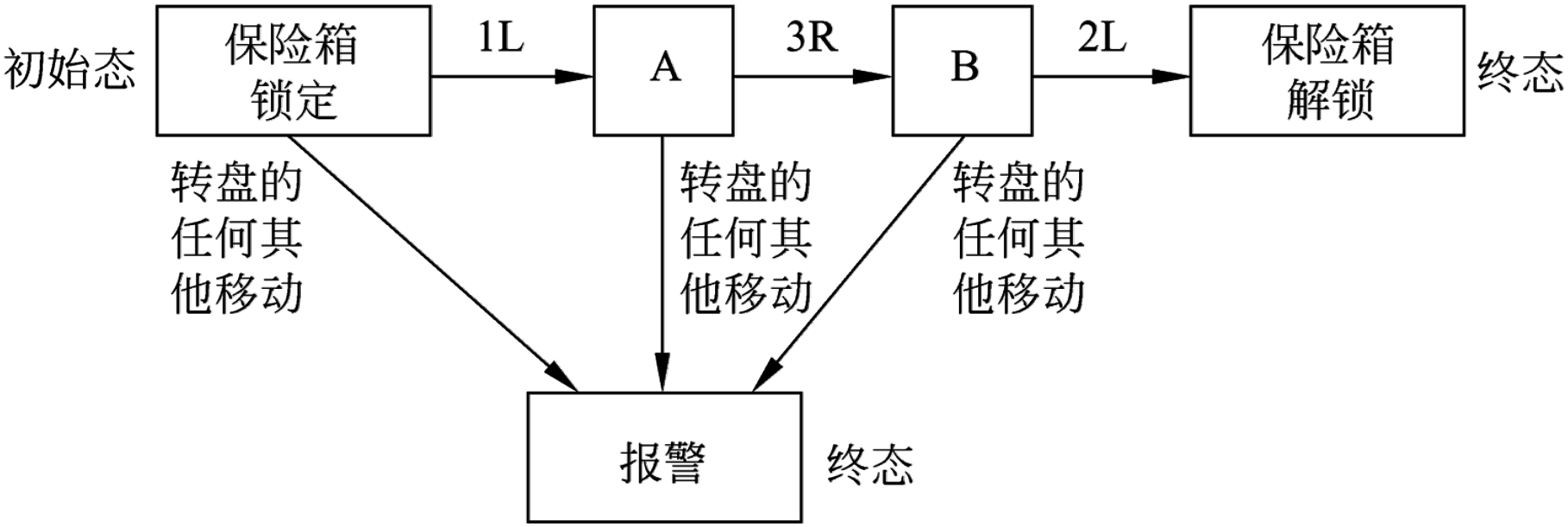
4.2 有穷状态机

- 4.2.1 有穷状态机概念
- 4.2.2 有穷状态机例子
- 4.2.3 有穷状态机方法评价

4.2.1 有穷状态机的概念——引例

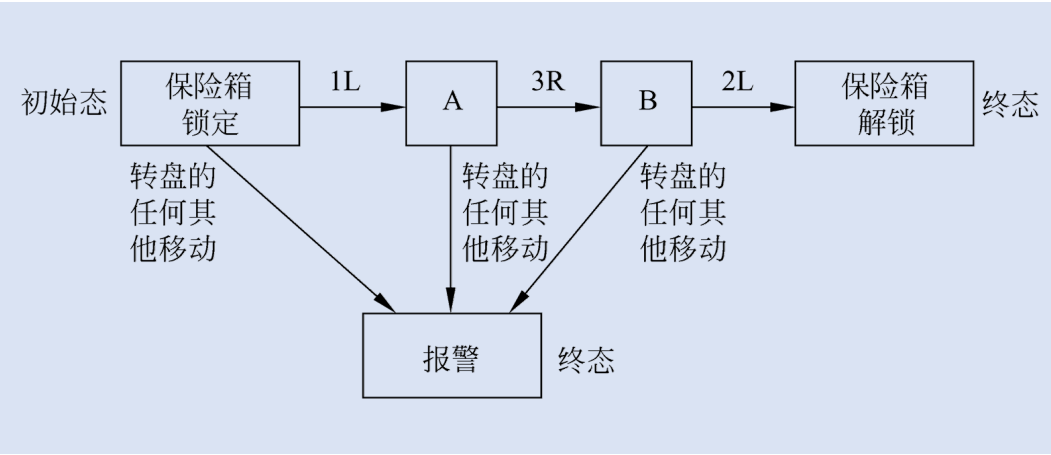
- 通过简单例子引入有穷状态机的基本概念：
  - ✓一个保险箱上装了一个复合锁，锁有三个位置，分别标记为 1 、 2 、 3 。
  - ✓转盘可向左 (L) 或向右 (R) 转动。
  - ✓因此，在任意时刻转盘都有 6 种可能的运动，即 1L 、1R 、2L 、2R 、3L 和 3R 。
  - ✓保险箱的组合密码是 1L 、 3R 、 2L ，转盘的任何其他运动都将引起报警。

4.2.2 有穷状态机的概念——引例



保险箱的状态转换图　组合密码 1L 、 3R 、 2L

4.2.1 有穷状态机的概



保险箱状态转换表

当前状态 / 次态	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

4.2.1 有穷状态机的概念——形式化表示

- 一个有穷状态机包括下述 5 个部分:
  - 状态集 J： 有所有可能的状态构成的有穷集合;
  - 输入集 K: 引发状态变换的可能的外部输入（或操作）;
  - 转换函数T: 由当前状态和当前 输入变换到下一个状态 (次态) 的函数（或规则）;
  - 初始态 S ∈ J， 状态机的初始状态;
  - 终态集 F ∪ J， 状态机的终止状态集;

4.2.1 有穷状态机的概念——形式化表示

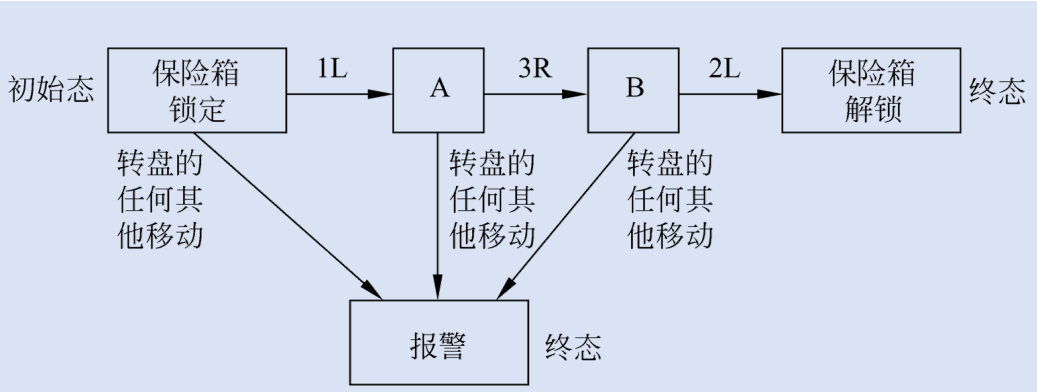
- 可以表示为一个 5 元组 (J，K，T，S，F)，其中：
  - J 是一个有穷的非空状态集；
  - K 是一个有穷的非空输入集；
  - T 是一个从 (J-F) × K 到 J 的转换函数；
  - S ∈ J，是一个初始状态；
  - F ⊆ J，是终态集。

4.2.1 有穷状态机的概念——形式化表示

保险箱的例子：

- 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：见“状态转换表”
- 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	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



4.2.1 有穷状态机的概念——增加谓词集

- 一个有穷状态机可以增加谓词集，表示为一个 6 元组 (J, K, P, 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；
- P 是一个有穷的非空谓词集(条件函数集合)；
- T 是一个从 (J-F)×K×P 到 J 的转换函数；
- S ∈ J，是一个初始状态；
- F⊆ J，是终态集。

4.2 有穷状态机

- 4.2.1 有穷状态机概念
- 4.2.2 有穷状态机例子
- 4.2.3 有穷状态机方法评价

4.2.2 有穷状态机的例子——电梯控制系统

- 自然语言描述的对电梯系统的需求
- 电梯系统有穷状态机-按钮集
- 电梯按钮（电梯内）的状态转换
- 楼层按钮（电梯外）的状态转换

4.2.2 有穷状态机的例子——电梯控制系统

自然语言描述的对电梯系统的需求：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（电梯外）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。

4.2.2 有穷状态机的例子——电梯控制系统

现在使用一个扩展的有穷状态机对本产品进行规格说明。这个问题中有两个按钮集：

1. 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。

2. 此外，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。

电梯按钮的状态转换

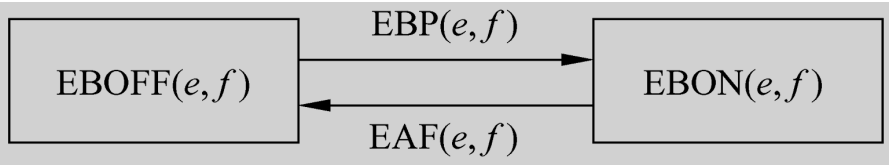
令 EB(e,f) 表示按下电梯 e 内的按钮并请求到 f 层去。 EB(e,f) 有两个状态， 分别是按钮发光和不发光 。两个状态是:

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

如果电梯按钮 (e,f) 发光且电梯到达 f 层， 该按钮将熄灭。相反如果按钮熄灭， 则按下它时，按钮将发光。 两个事件是:

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

电梯按钮 谓词



为了定义与这些事件和状态相联系的状态转换规则， 电梯按钮需要一个谓词 V(e,f)：

V(e, f)： 电梯 e 停在 f 层

V(e, f) = S(U, e, f) or S(D, e, f) or S(N, e, f)　U (向上) , D(向下) , N(待定)

1. 如果电梯按钮 (e,f) 处于关闭状态〔当前状态〕，
2. 而且 电梯按钮 (e,f) 被按下〔事件〕，
3. 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〕。

状态转换规则描述为：

- EBOFF(e,f)+EBP(e,f)+not V(e,f) => EBON(e,f)

1. 反之，如果电梯到达 f 层，
2. 而且电梯按钮是打开的，它就会熄灭。

转换规则表示为：

- EBON(e,f)+EAF(e,f) => EBOFF(e,f)

楼层按钮的状态转换

令 FB(d,f) 表示 f 层请求电梯向 d 方向运动的按钮。

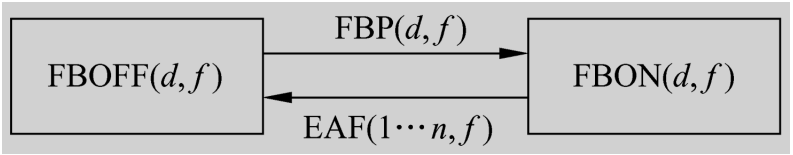
楼层按钮的状态如下：

- FBON(d,f)： 楼层按钮 (d,f) 打开
- FBOFF(d,f)： 楼层按钮 (d,f) 关闭

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。反之，如果楼层按钮原来是关闭的，被按下后该按钮将打开。楼层按钮的两个事件：

- 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 。

楼层按钮 谓词



为了定义与这些事件和状态相联系的状态转换规则，楼层按钮需要一个谓词 S(d,e,f)：

S(d,e,f)： 电梯 e 停在 f 层并且移动方向由 d 确定为向上 (d=U) 或向下 (d=D) 或待定 (d=N) 。

FBOFF(d, f) + FBP(d, f) + not S(d, 1 ··· n, f) => FBON(d, f)

1. 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，
2. 现在该按钮被按下，
3. 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。

FBON(d, f) + EAF(1 ··· n, f) + S(d,1 ··· n, f) => FBOFF(d, f)；其中， d=U or D

1. 反之，如果楼层按钮已经打开，
2. 且至少有一部电 梯到达 f 层，
3. 并且该部电梯将朝 d 方向运动，则楼层按钮将关闭。

电梯的状态、事件

电梯的 3 个状态:

- 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)

电梯的 3 个可触发状态发生改变的事件:

- 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

电梯的状态转换规则

(这里给出的规则仅发生在关门时)

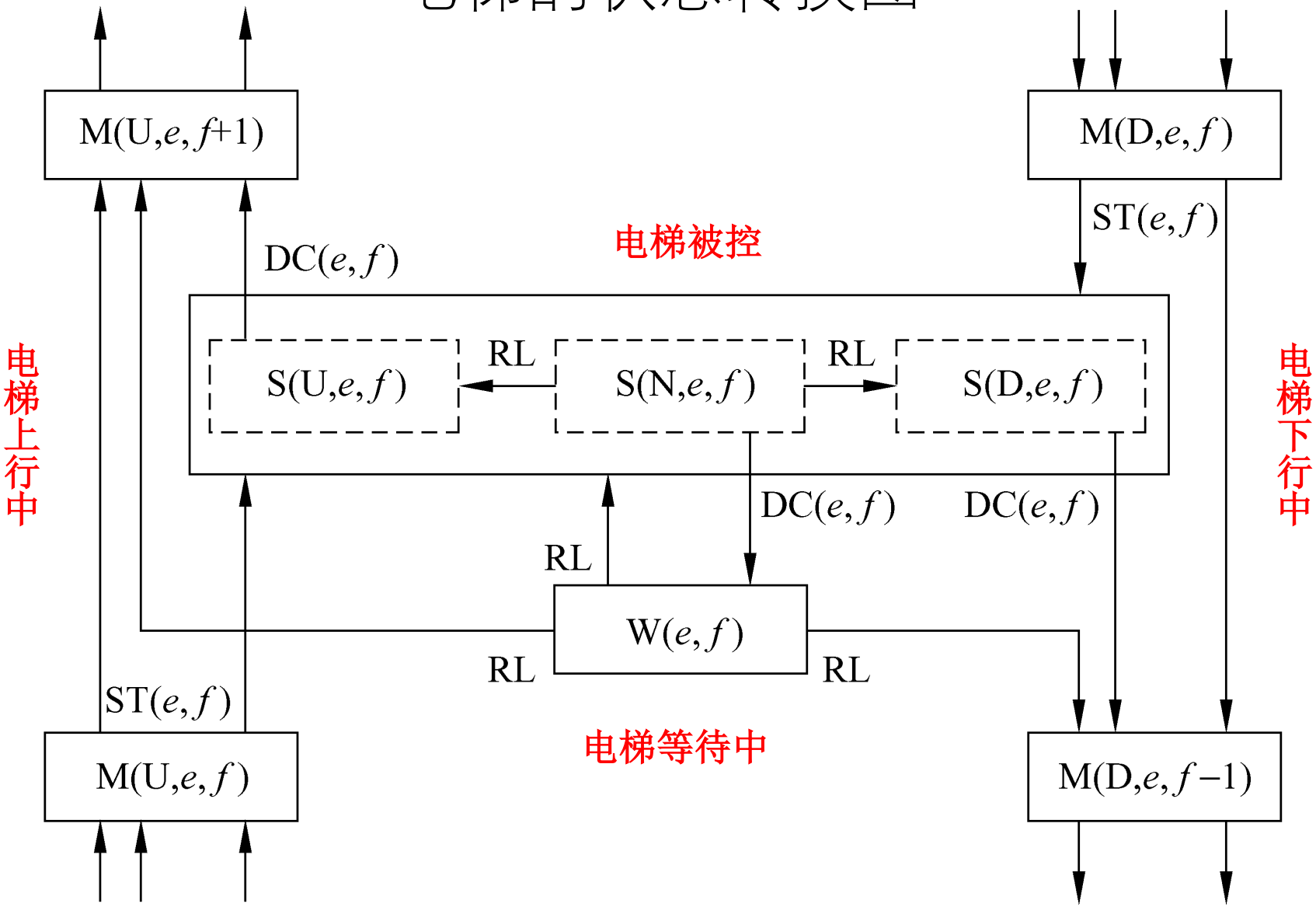
S(U,e,f)+DC(e,f)=>M(U,e,f+1):

1.如果电梯 e 停在 f 层准备向上(U)移动，
2.且门已经关闭，则电梯将向上一楼层移动。

1.如果电梯 e 停在 f 层准备向下(D)移动，
2.且门已经关闭，则电梯将向下一楼层移动。

1.如果电梯 e 停在 f 层没有移动请求(N)，
2.且门已经关闭，则电梯等待移动。

电梯的状态转换图


M(U,e,f+1)
M(D,e,f)
ST(e,f)
DC(e,f)
电梯被控
电梯上行中
电梯下行中
S(U,e,f)
RL
S(N,e,f)
RL
S(D,e,f)
DC(e,f)
DC(e,f)
RL
W(e,f)
RL
RL
ST(e,f)
电梯等待中
M(U,e,f)
M(D,e,f−1)

4.2 有穷状态机

- 4.2.1 有穷状态机概念
- 4.2.2 有穷状态机例子
- 4.2.3 有穷状态机方法评价

4.2.3 有穷状态机——方法评价

1. 采用一种简单的有穷状态机方法格式来描述规格说明:

- 当前状态 + 事件 + 谓词 => 下个状态
- 这种形式的规格说明易于书写、易于验证。

2. 有穷状态机可以比较容易地把它转变成设计或程序代码:

- 可开发一个 CASE 工具把一个有穷状态机规格说明直接转变为源代码。（如 Nunni FSM Generator）
- 维护可以通过重新转变来实现，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。

4.2.3 有穷状态机——方法评价

3. 有穷状态机方法与数据流图技术比较

- 有穷状态机描述需求比数据流图技术更精确;
- 与数据流图一 样易于理解（稍难一点）。

4. 有穷状态机存在的缺点:

- 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，有穷状态机方法也没有处理定时需求。

第四章 形式化说明技术

4.3 Petri网

4.3.1 Petri网概念

4.3.2 Petri网例子

4.3.3 Petri网的评价

4.3 Petri网

4.3.1 Petri网概念

4.3.2 Petri网例子

4.3.3 Petri网的评价

4.3.1 Petri网的概念

- 软件系统的定时问题
- Petri网及其用途
- Petri网的构成
- Petri网的形式化表示

4.3.1 Petri网的概念——软件系统的定时问题

- 并发系统中遇到的一个主要问题是定时问题。这个问题可以表现为多种形式，如同步问题、竞争条件以及死锁问题。

- 定时出现问题通常是由不好的设计或有错误的实现引起的，而这样的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。

4.3.1 Petri网的概念——Petri网及其用途

- 用于确定系统中隐含的定时问题的一种有效技术是 Petri 网，这种技术的一个很大的优点是它可以有效地描述并发活动。
- Petri网是对离散并行系统的数学表示。Petri网既有严格的数学表述方式，也有直观的图形表达方式。
- Petri网适合于描述异步的、并发的自动化系统和计算机系统模型。Petri 网在计算机科学得到了广泛的应用，例如，在系统性能评价、操作系统和软件工程等领域，Petri 网应用得都比较广泛。

4.3.1 Petri网的概念——Petri网的构成

- Petri 网包含 4 种元素：
  - 一组位置 P (Place):圆形
  - 一组转换 T (Transition):线性/矩形
  - 输入函数 I (Input)
  - 输出函数 O (Output)

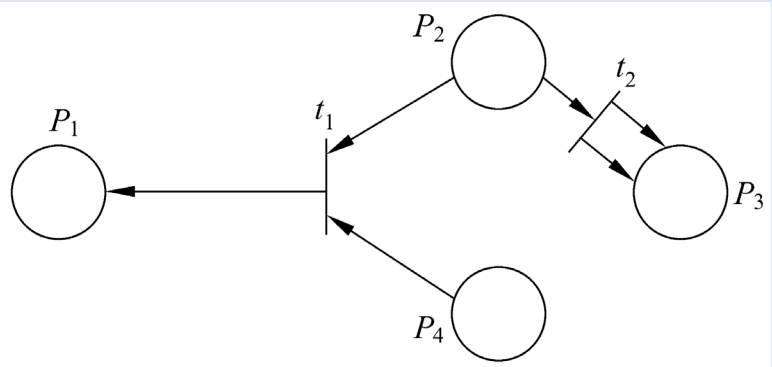
- 其他元素
  - 连接Connection:箭头
  - 权标Token(令牌):圆点

P:｛P1，P2，P3，P4｝

T:｛ t1，t2｝

I: I (t1)=｛P2, P4｝, I(t2)=｛P2｝

O: O(t1)=｛P1｝ O(t2)=｛P3，P3｝



4.3.1 Petri网的概念——Petri网的形式化表示

一个Petri网是一个四元组 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： $T \rightarrow P^{\infty}$ 为输入函数，是由转换到位置无序单位组(bags) 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。

4.3.1 Petri网的概念——Petri网的标记

权标(token ， 令牌)

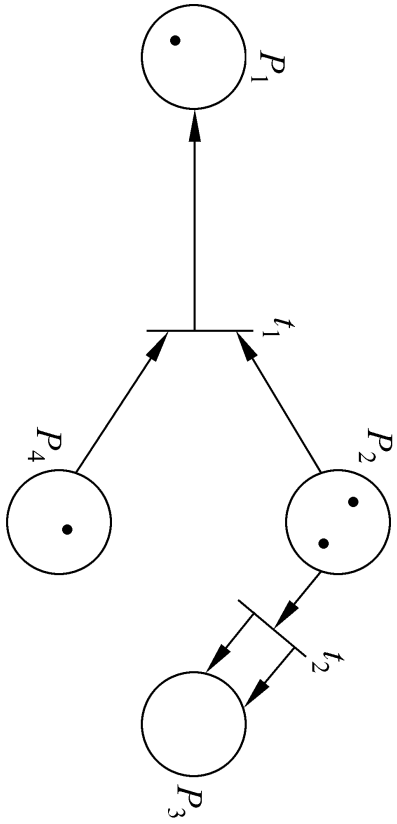
- 位置P中的权标（用圆点表示），表示该位置持有的可激发与其连接的后继转换 t 的条件。

标记 （M，Mark）

- 标记是在Petri网中权标(token)的分配。Petri 网标记 M 是由一组位置 P 到一组非负整数的映射， 即 M： P → { 0 ， 1 ， 2 ， … }
- 其中的数字表示每个位置中当前的权标数量。

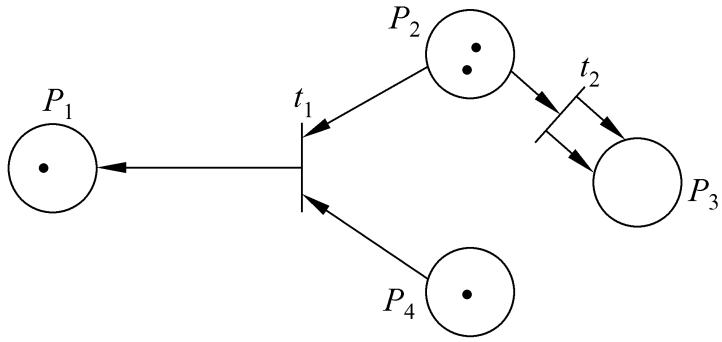
转换的激发

- 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所直接连接的后续转换 t 的线数时，就允许激发该转换 t 。此时， t 所直接连接的前序位置减少一个权标（用掉1个）； t 所直接连接的后续位置增加一个权标（获得1个）。
- Petri 网具有非确定性，也就是说，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。



4.3.1 Petri网的概念——例：带权标的Petri网

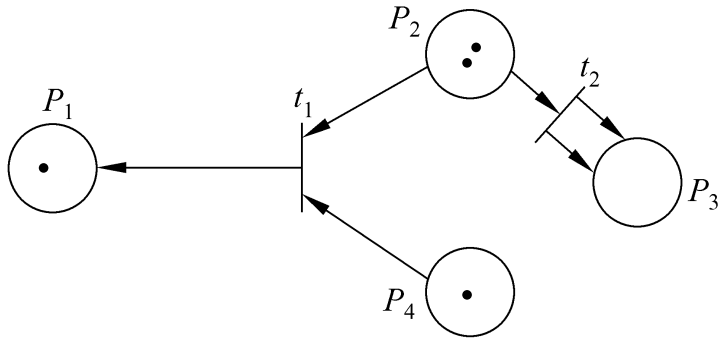
- 图1中有 4 个权标， P1，P2，P3，P4 中的标记可以用向量 (1, 2, 0,1）表示。



带标记的Petri网: （1，2，0，1）

4.3.1 Petri网的概念——例：带权标的Petri网

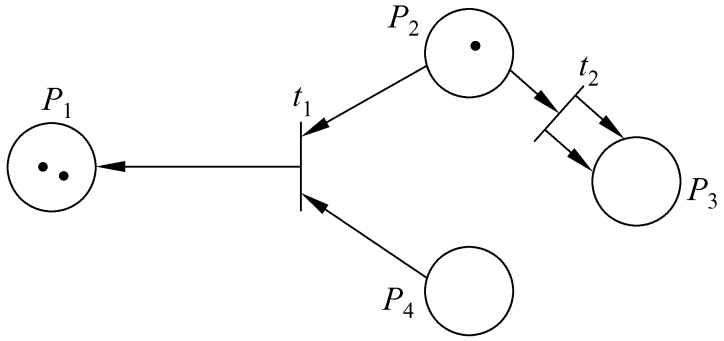
- P2 和 P4 中有权标，因此 t1 被激发。
- 当 t1 被激发时，P2 和 P4 上各有一个权标被移出，而 P1上则增加一个权标。



t1被激发：（2，1，0，0）

4.3.1 Petri网的概念——例：带权标的Petri网

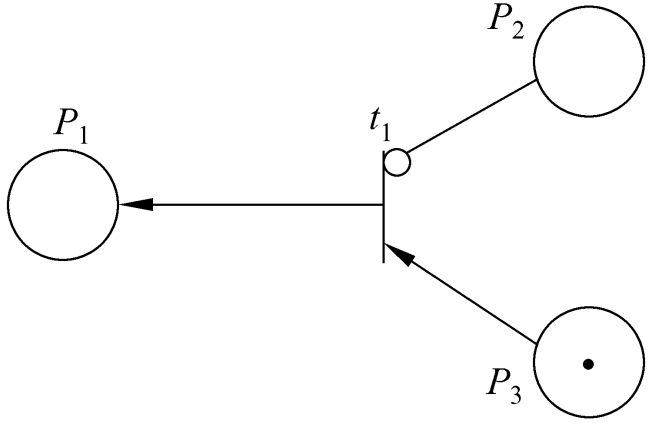
- 图中 P2 上有权标，因此 t2 也可以被激发。
- 当 t2 被激发时， P2 上将移走一个权标，而 P3 上新增加两个权标。



t2被激发：（2，0，2，0）

4.3.1 Petri网的概念——含禁止线的Petri网

- 禁止线是以使用“圆点” 标记的输入线。表示禁止线上（图中P2）没有权标时，后续的转换（图中t1）才可激活。
- 箭头线P3上有权标，而禁止线P2上没有权标，所以转换t1可以激活。



含禁止线的Petri网

4.3.1 Petri网概念——更形式化的Petri网 (增加标记M)

一个Petri网是一个五元组 C = (P，T，I，O，M)

- 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：$T \rightarrow P^{\infty}$，为输入函数，是由转换到位置无序单位组(bags) 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
- M：P→｛0，1，2，…｝，是由一组位置P到一组非负整数的映射

4.3 Petri网

4.3.1 Petri网概念

4.3.2 Petri网例子

4.3.3 Petri网的评价

4.3.2 Petri网的例子——电梯系统控制

Petri网应用于电梯问题

- 每个楼层用一个位置F_f代表($1 \leqslant f \leqslant m$);
- 电梯用一个权标代表。
- 在位置F_f上有权标，表示在楼层f上有电梯。

可能的几种约束条件

- 电梯按钮（条件C1，电梯内有按钮操作）
- 楼层按钮（条件C2，楼层中有按钮操作）
- 电梯静止（条件C3，没有任何操作请求）

4.3.2 Petri网的例子——电梯按钮（约束条件C1）

约束条件C1，描述了“电梯按钮”的行为

- 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。

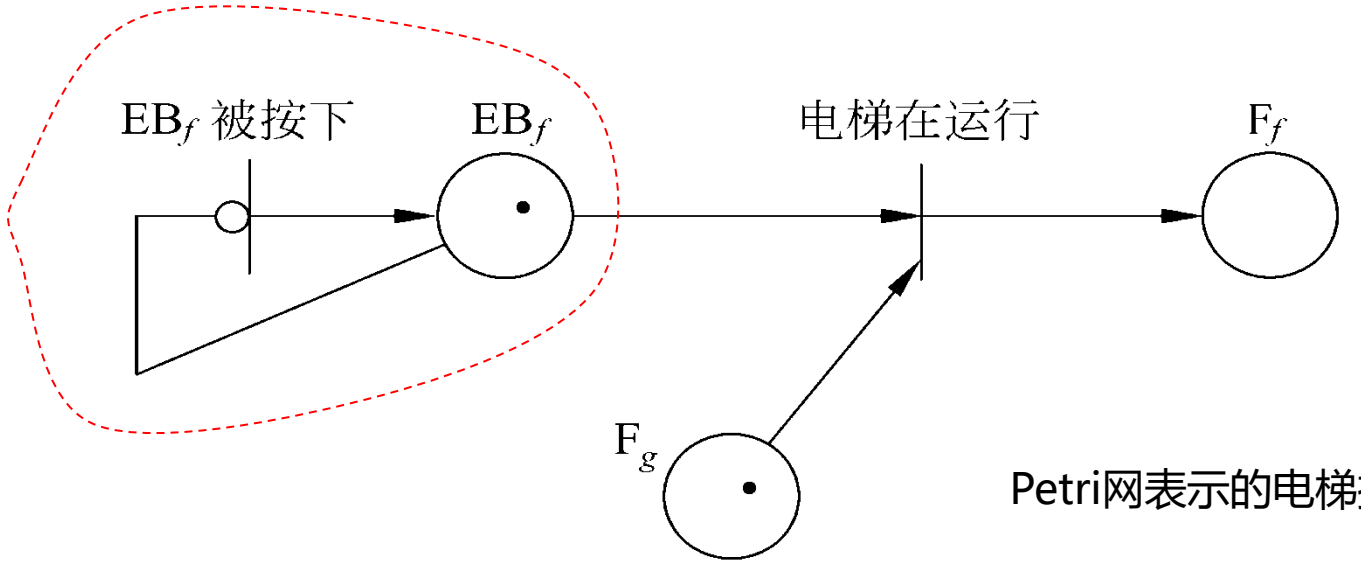
电梯按钮位置

- 电梯中楼层 f 的按钮，在Petri网中用位置EB_f表示($1 \leqslant f \leqslant m$)
- 在EB_f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。(采用禁止线)

4.3.2 Petri网的例子——电梯按钮的Petri网

首先，假设按钮没有发亮，现在按下按钮，则转换被激发并在EBf上放置了一个权标。

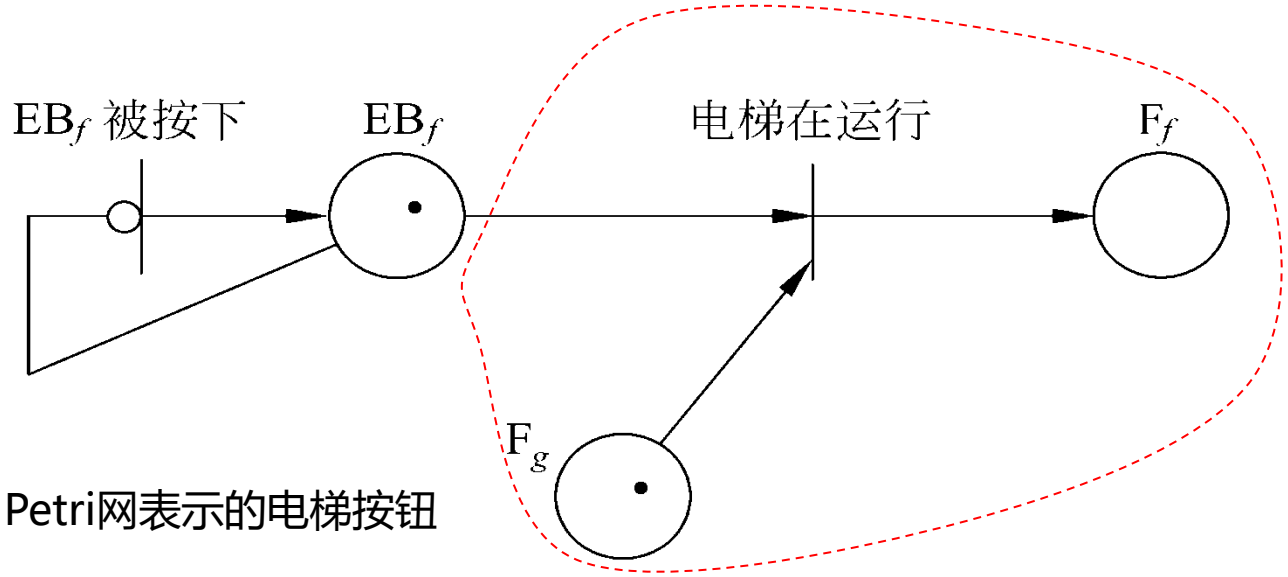
以后不论再按下多少次按钮，禁止线与现有权标的组合都决定了转换“EBf被按下”不能再被激发。



Petri网表示的电梯按钮

4.3.2 Petri网的例子——电梯按钮的Petri网

- 电梯由 g 层驶向 f 层。
- 实际运行中，电梯由 g 层移到 f 层是需要时间的，为处理这个情况，Petri网模型中必须加入时限。



Petri网表示的电梯按钮

4.3.2 Petri网的例子—— 楼层按钮（约束条件C2）

约束条件C2，描述了“楼层按钮”的行为

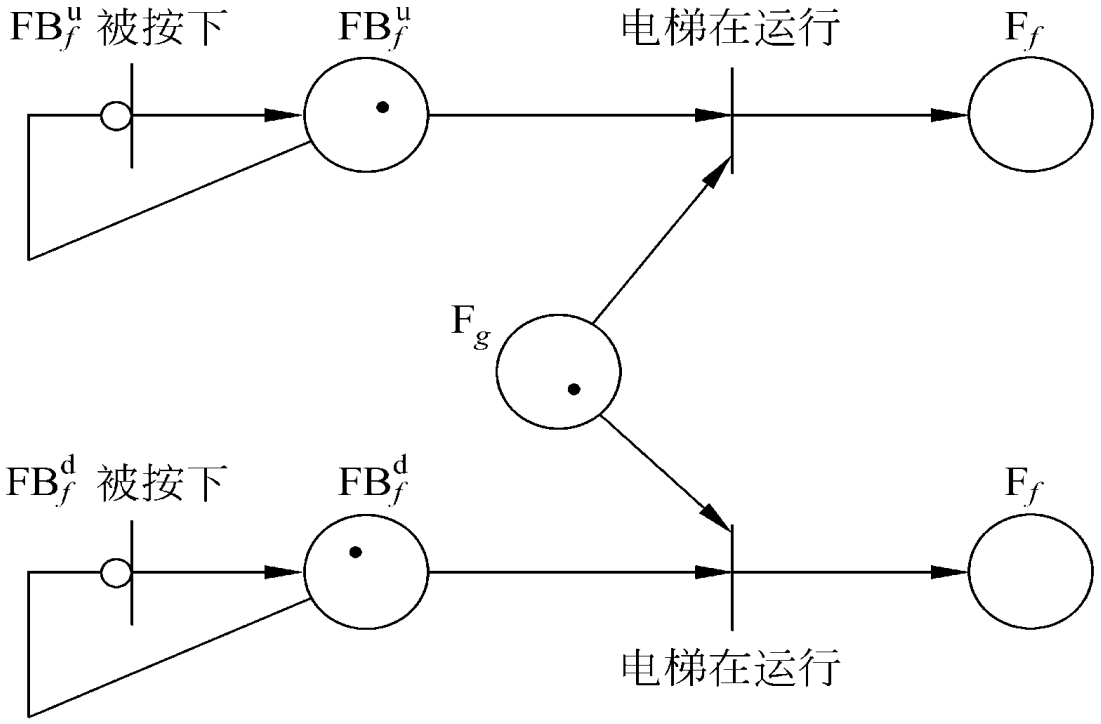
- 除了第一层与顶层之外，每个楼层都有两个按钮，一个要求电梯上行，另一个要求电梯下行。
- 这些按钮在按下时发亮，当电梯到达该层并将向指定方向移动时，相应的按钮才会熄灭。

楼层按钮位置

- 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行的按钮。
- 底层的按钮为 FB_1^u，最高层的按钮为 FB_m^d，中间每一层有两个按钮 FB_f^u 和 FB_f^d $(1 < f < m)$。

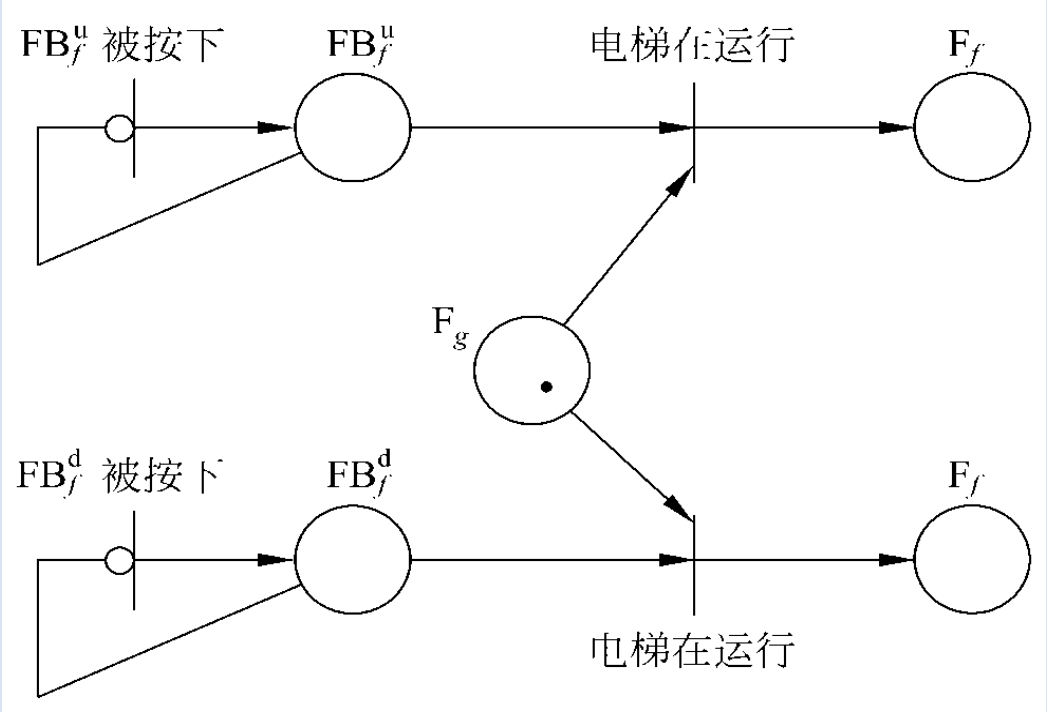
4.3.2 Petri网的例子—— 楼层按钮的Petri网

- 电梯由g层驶向f层。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。
- 但是要保证按钮熄灭正确，则需要更复杂的Petri网模型。



4.3.2 Petri网的例子—— 电梯静止（约束条件C3）

- 最后，考虑第三条约束C3：
  - 当电梯没有收到请求时，它将停留在当前楼层并关门。



如图所示，当没有请求(FB_f^u和FB_f^d上没有权标）时，任何一个转换“电梯在运行”都不能被激发，而是停留在当前层（Fg）。

4.3.3 Petri网方法评价

- Petri 网是对离散并行系统的数学表示。
- Petri 网适合于描述异步的、并发的计算机系统模型。
- Petri 网既有严格的数学表述方式，也有直观的图形表达方式。

第四章 形式化说明技术

4.4 Z语言

4.4.1 Z语言简介

4.4.2 Z语言例子

4.4.3 Z语言评价

4.4 Z语言

4.4.1 Z语言简介

4.4.2 Z语言例子

4.4.3 Z语言评价

4.4.1 Z语言简介

- Z语言是目前使用最广泛的一种形式化描述语言，将事物的状态和行为用数学符号形式化表达的语言，是软件编码之前的规格说明语言。
- Z语言是一种以一阶谓词演算为主要理论基础的规约语言，是一种功能性语言。
- 在Z中有两种语言：数学语言和模式语言。数学语言用来描述系统的各种特征：对象及其之间的关系。模式语言是一种半图形化的语言，它用来构造、组织形式化说明的描述、整理、封装信息块并对其命名以便可以重用这些信息块。
- 通常，形式化说明的可读性都不太好，但由于Z采用半图形化的模式语言，能用一种比较直观、有条理的方式来表达形式化说明，改善了可读性。

4.4 Z语言

4.4.1 Z语言简介

4.4.2 Z语言例子

4.4.3 Z语言评价

4.4.2 Z语言例子——需求规格说明的构成

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：

1. 给定的集合、数据类型及常数
2. 状态定义
3. 初始状态
4. 操作

4.4.2 Z语言例子——构成1：给定的集合

- 给定的集合

  •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，这种集合用带方括号的形式表示。

- 电梯控制系统的“给定的集合”

  •电梯问题中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〕

4.4.2 Z语言例子——构成2：状态定义

状态定义——Z格

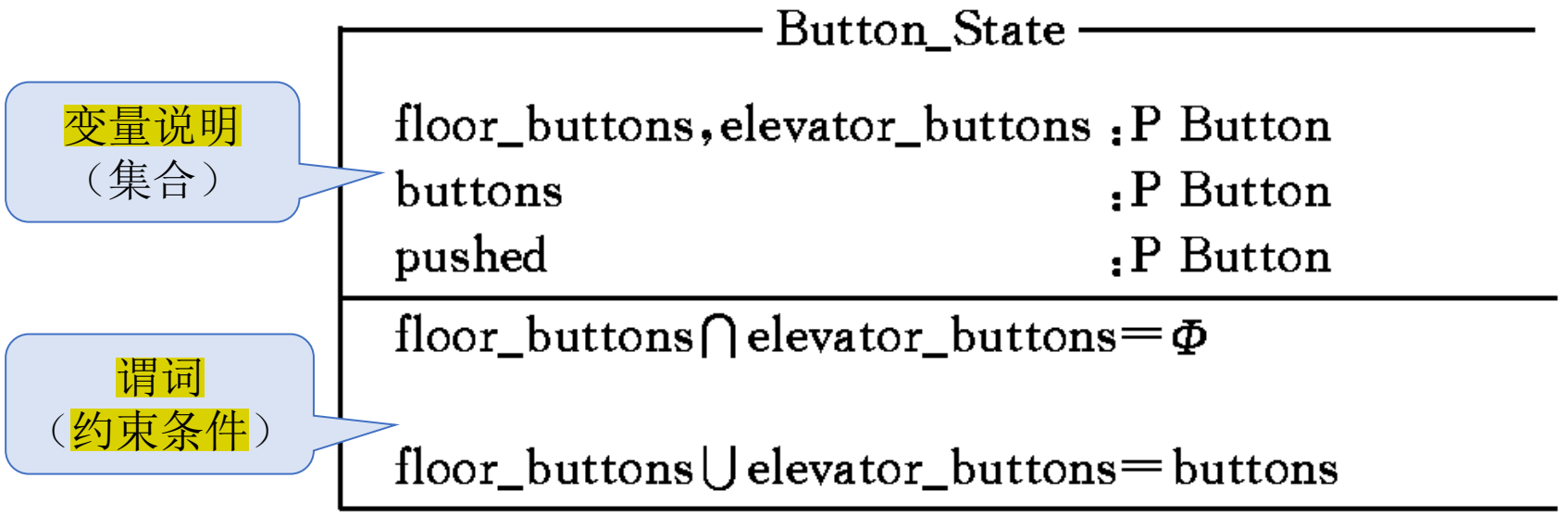
- 一个Z规格说明由若干个“格(schema)”组成，Z格的格式图所示。

Z格由以下几个部分构成：

- 格的名称（如：“S”）；
- 一组说明：给出Z格中可能用到的变量；
- 一组谓词：限定变量取值范围的。

4.4.2 Z语言例子——构成2：状态定义

电梯控制系统中的Z格：Button_State



4.4.2 Z语言例子——构成3：初始状态

- 初始状态：指系统第一次开启时的状态。
- 电梯控制系统的抽象初始状态为：
  - Button_Init $\widehat{=}$〔Button_State | pushed=Φ〕
  - 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。

4.4.2 Z语言例子——构成4：操作

- Z语言的操作：包括操作涉及到的变量、操作的前置条件以及操作的后置条件。

  操作的说明部分：定义了该操作涉及到的输入、输出变量。

  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。

- 运算符&表达式

  △格名——在本格中引用的格；

  变量名？、变量名！——输入变量、输出变量；

  逻辑运算符（∧、∨、…）；

  集合运算符（∈、∪、∩、/、…）；

  变量名′ ——变化后的变量。

Push_Button

△Button_State

button?:Button

(button? ∈buttons) ∧
(((button? ∉pushed) ∧ (pushed′=pushed∪{button?})) ∨
((button? ∈pushed) ∧ (pushed′=pushed)))

4.4.2 Z语言例子——构成4：操作

电梯控制系统的操作：Push_Button

- 图中定义了“按下按钮”操作 Push_Button
- 操作的说明部分
  - 该操作中引用了外部Z格△ Button_State；
  - 该操作的输入变量为Button？。
- 操作的谓词部分
  - 第一个前置条件：输入变量 button？∈ buttons（所有按钮的集合）；
  - 第二个前置条件：输入变量 button？不属于pushed，则在pushed集合中增加该按钮；
  - 第三个前置条件：输入变量 button？属于pushed，则pushed保持不变。

Push_Button

$\Delta Button\_State$

$button? : Button$

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

4.4.2 Z语言例子——构成4 操作

电梯控制系统的操作：Floor_Arrival

- 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。也就是说，如果“button?”属于pushed集，则将它移出该集合。

Floor_Arrival

ΔButton_State

button? : Button

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

4.4 Z语言

4.4.1 Z语言简介

4.4.2 Z语言例子

4.4.3 Z语言评价

4.4.3 Z语言评价

使用形式化规格说明是全球的总趋势。是应用得最广泛的形式化语言，在大型项目中Z语言的优势更加明显：

- (1) 易于发现规格说明中的错误。
- (2) 可精确地描述规格说明。
- (3) 方便需求说明的正确性验证。
- (4) 具有一定的可掌握性。
- (5) 可降低软件开发费用。
- (6) 易于正确地转换成自然语言描述。

第四章 形式化说明技术

4.4.5 小结

- 形式化规格说明方法的优点
  - 形式化的规格说明可以用数学方法研究、验证。能够消除二义性，鼓励软件开发者在软件工程过程的早期阶段使用更，从而可以减少差错。
- 形式化规格说明方法的缺点
  - 大多数规格说明方法关注于系统的功能和数据，而问题的时序、控制和行为等方面的需求却更难于表示。此外，比欠形式化方法更难学习。
- 形式化方法和欠形式化方法结合