形式化说明技术

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，半形式化和形式化 3 类。用自然语言描述需求规格说明，是典型的非形式化方法。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。

所谓形式化方法，是描述系统性质的基于数学的技术，也就是说，如果一种方法有坚实的数学基础，那么它就是形式化的。

4.1 概述

4.1.1 非形式化方法的缺点

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。例如，规格说明书的某一部分可能规定系统必须监控化学反应容器中的温度，而另一部分（可能由另一位系统分析员撰写）却规定只监控在一定范围内的温度。如果这两个相互矛盾的规定写在同一页纸上，自然很容易查出，不幸的是，它们往往出现在相距几十页甚至数百页的两页纸中。

二义性是指读者可以用不同方式理解的陈述。例如，下面的陈述就是具有二义性：

“操作员标识由操作员姓名和密码组成，密码由 6 位数字构成。当操作员登录进系统时它被存放在注册文件中。”

在上面这段陈述中，“它”到底代表“密码”还是“操作员标识”，不同的人往往有不同的理解。

系统规格说明书是很庞大的文档，因此，几乎不可避免地会出现含糊性。例如，我们可能经常在文档中看到类似下面这样的需求：“系统界面应该是对用户友好的。”实际上，这样笼统的陈述并没有给出任何有用的信息。

不完整性可能是在系统规格说明中最常遇到的问题之一。例如，考虑下述的系统功能需求：

“系统每小时从安放在水库中的深度传感器获取一次水库深度数据，这些数值应该保留 6 个月。”

假设在系统规格说明书中还规定了某个命令的功能：“ AVERAGE 命令的功能是，在 PC 机上显示由某个传感器在两个日期之间获取的平均水深。”

如果在规格说明书中对这个命令的功能没有更多的描述，那么，该命令的细节是严重不完整的，例如，对该命令的描述没有告诉我们，如果用户给定的日期是在当前日期的 6 个月之前，那么系统应该做什么。

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。这样的规格说明书使得读者很难了解系统的整体功能结构。

4.1.2 形式化方法的优点

正如上一小节所讲的，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。

在开发大型软件系统的过程中应用数学，能够带来下述的几个优点：

数学最有用的一个性质是，它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。数学特别适合于表示状态，也就是表示“做什么”。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，在这样的规格说明中完全没有含糊性。但是，实际情况并不这么简单，软件系统的复杂性是出了名的，希望用少数几个数学公式来描述它，是根本不可能的。此外，即使应用了形式化方法，完整性也是难于保证的：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。

在软件开发过程中使用数学的另一个优点是，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，当然，程序代码也是一种数学符号（虽然是一种相当繁琐、冗长的数学符号）。

数学作为软件开发工具的最后一个优点是，它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。

4.1.3 应用形式化方法的准则

人们对形式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。编者认为，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。为了更好地发挥这种方法的长处，下面给出应用形式化方法的几条准则，供读者在实际工作中使用。

（1） 应该用适当的表示方法。通常，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。例如， Z 语言并不适于说明并发性。因此，应该仔细选择一种适用开当前项目的形式化说明技术。

（2 ） 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。例如，示例屏幕和自然语言可能还是目前描述用户界面的可视特性的最佳方法。但是，也不能因此就认为完全没必要采用形式化方法。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。事实上，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。

（3 ） 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。

（4 ） 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 ，因此，需要专家指导和培训。

（5 ） 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，而且由于取长补短往往能获得很好的效果。

（6 ） 应该建立详细的文档。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。

（7 ） 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。

（8 ） 不应该盲目依赖形式化方法。这种方法不是包治百病的灵丹妙药，它们只不过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是正确，例如，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，必须用其它方法（例如，评审，测试）来验证软件正确性。

（9 ） 应该测试，测试再测试。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。

（10） 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，而且用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。

4.2 有穷状态机

利用有穷状态机可以准确地描述一个系统，因此它是表达规格说明的一种形式化方法。

4.2.1 概念

下面通过一个简单例子介绍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。这样，在任意时刻转盘都有 6 种可能的运动，即时 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。图 4.1 描绘了保险箱的状态转换情况。有一个初始态，即保险箱锁定状态 。若输入为 1L ，则下一个状态为 A，但是，若输入不是 1L 而是转盘的任何其他移动，则下一个状态为“报警”，报警是两个终态之一（另一个终态是“保险箱解锁”）。如果选择了转盘移动的正确组合，则保险箱状态转换的序列为从保险箱锁定到 A 再到 B，最后到保险箱解锁，即另外一个终态。图 4.1 是一个有穷状态机的状态转态转换图。状态转换并不一定要用图形方式描述，表 4.1 的表格形式也可以表达同样的信息。除了两个终态之外，保险箱的其他状态将根据转盘的转动方式转换到下一个状态。

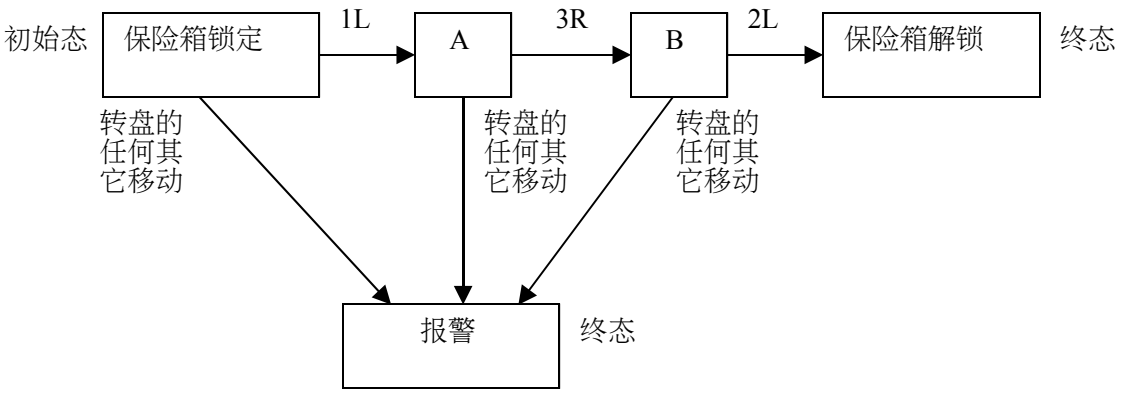


图 4.1 保险箱的状态转换图

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

表 4.1 保险箱的状态转换表

从上面这个简单的例子可以看出，一个有穷状态机包括下述 5 个部分：状态集 J 、输入集K 、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，初始态 S 和终态集 F 。对于保险箱的例子，相应的有穷状态机的各部分如下。

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｝ 。

如果使用更形式化的术语，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 $(J-F)\times K$到J的转换函数；

$S\in J$,是一个初始状态；

$F\subseteq J$，是终态集。

有穷状态机的概念在计算机系统中应用得非常广泛。例如，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] $\Rightarrow$ 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，即在前述的 5 元组中加入 6 个组件——谓词集 P，从而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。转换函数 T 现在是一个从 $(J-F)\times K\times P$ 到 J 的函数。现在的转换规则形式如下：

当前状态 [菜单]+ 事件 [所选择的项]+谓词 $\Rightarrow$ 下个状态。

4.2.2 例子

为了具体说明怎样用有穷状态机技术表达系统的规格说明，现在用这种技术给出大家熟悉的电梯系统的规格说明。首先给出用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

C_1，C_2和C_3在楼层间移动。

C_1：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。

C_2：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。

C_3：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。这个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。此外，每层楼有两个按钮，一人请求向上，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如图 4.2 所示。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（打开）和不发光（关闭）。更精确地说，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

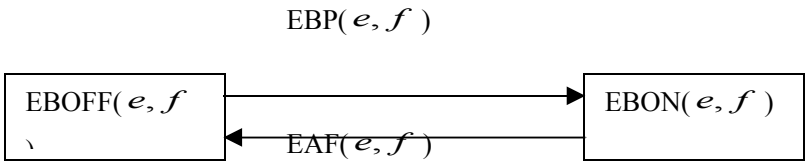


图 4.2 电梯按钮的状态转换图

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。上述描述中包含了两个事件，它们分别是：

EBP(e,f)：电梯按钮 （e,f ）被按下

EAF(e,f)：电梯 e 到达 f 层

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 $V(e,f)$ ，

它的含义如下：$V(e,f)$: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：

$$EBOFF(e,f)+EBP(e,f)+notV(e,f)\Rightarrow EBON(e,f)$$

反之，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。这条转换规则可以形式化地表示为：$EBON:(e,f)+EAF(e,f)\Rightarrow EBOFE(e,f)$

接下来考虑楼层按钮。令 $FB(d,f)$ 表示 f 层请求电梯向 d 方向运动的按钮，楼层按钮 $FB(d,f)$ 的状态转换图如图 4.3 所示。楼层按钮的状态如下

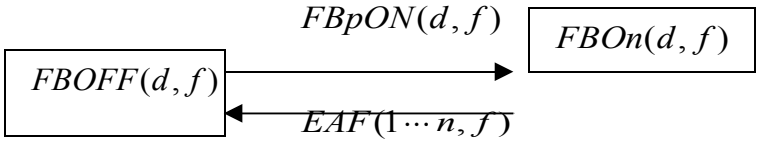


图 4.3 楼层按钮的状态转换图

$FBON(d,f)$：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。这段叙述中包含了以下两个事件。

$FBP(d,f)$：楼层按钮 (d,f) 被子按下

$EAF(1\cdots n,f)$：电梯 1 或…或 n 到达 f 层

其中 1…n 表示或为 1 或为 2… 或为 n

为了定义与这些事件和状态相联系的状态转换规则，同样也需要一个谓词，它是 $S(d,e,f)$，它的定义如下。

$S(d,e,f)$：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 $(d=N)$。

这个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。

使用谓词$S(d,e,f)$,形式化转换规则为 ：

$$FBOFF(d,f)+FBP(d,f)+notS(d,1\cdots n,f)\Rightarrow FBON(d,f)$$

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

其中，$d=UorD$。

也就是说，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。反之，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。

在讨论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：$V(e,f)=S(U,e,f)orS(D,e,f)orS(N,e,f)$

定义电梯按钮和楼层按钮的状态都是很简单、直观的事情。现在转向讨论电梯的状态及其转换规则，就会出现一些复杂的情况。一个电梯状态实质上包含许多子状态（例如 ，电梯减速，停止，开门，在一段时间后自动关门）。

下面定义电梯的 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）

其中 $S(d,e,f)$ 状态已在讨论楼层按钮时定义过，但是，现在的定义更完备些。

图 4.4 是电梯的状态转换图。注意，3 个电梯停止状态 $S(U,e,f)$ 、$S(N,e,f)$ 和 $S(D,e,f)$ 已被组合成一个大的状态，这样做的目的是减少状态总数以简化流图。图 4.4 中包含了下述 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 层时触发传感口器，电梯控制器决定在当前楼层电梯是否停下 RL

：电梯按钮或楼层被按下打开状态，登录需求最后，给出电梯的状态转换规则，为简单起见，这里给出的规则仅发生在关门之时．。

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

第一条规则表明，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。第二条和第三者规则，分别对应于电梯即将下降或者没有待处理的请求的情况

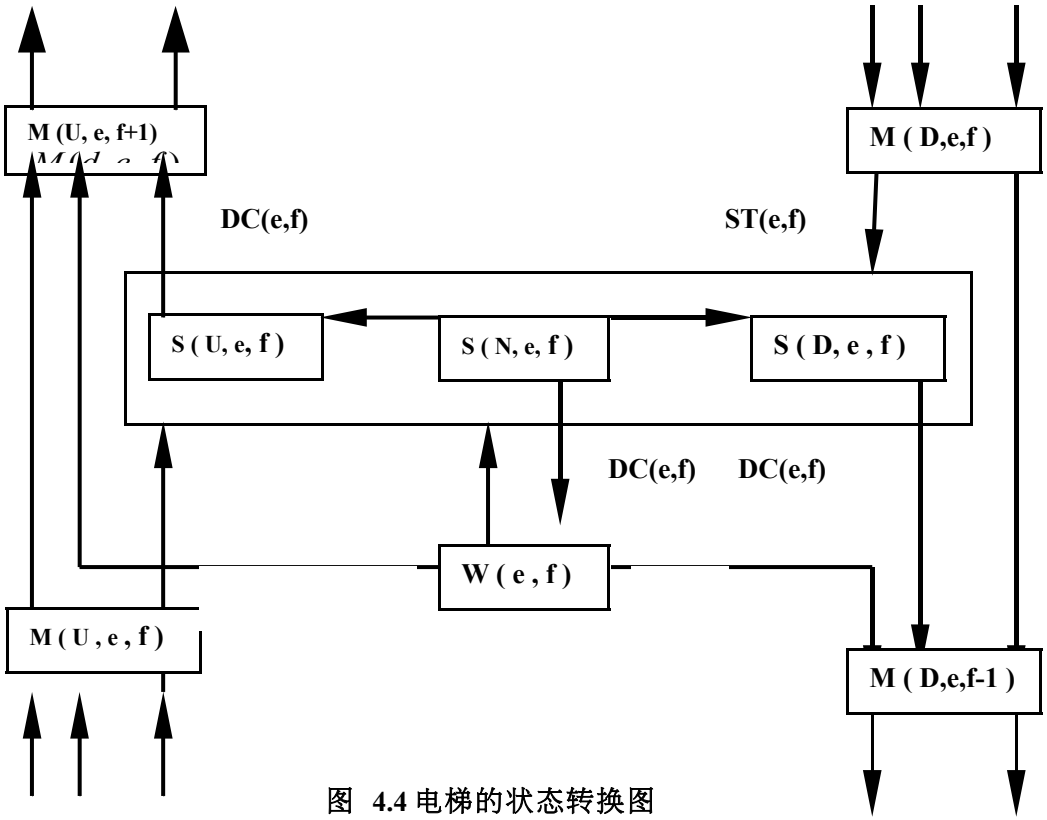


图 4.4 电梯的状态转换图

4.2.3 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 $\Rightarrow$ 下个状态

这种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，事实上，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成源代码。维护可以通过重新转变来实现，也说是说，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，不过，它也有缺点，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，此外，和数据流图方法一样，形式化的有穷状态机方法也没有处理定时需求。下节将介绍的 Petri 网技术，是一种可处理定时的问题形式方法。

4.3 Petri 网

4.3.1 概念

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。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，这种技术的一个很大的优点是它也可以用于设计中。

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颉域， etri 网应用得都比较广泛。特别是已经证明，用Petri网可以有效地描述并发活动。

Petri网包含 4 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。图 4.5 举例说明了Petri网的组成。

其中，

一组位置 P 为 $\{P_1, P_2, P_3, P_4\}$，在图中用圆圈代表位置。

一组转换 T 为 $\{t_1, t_2\}$，在图中用短直线表示转换。

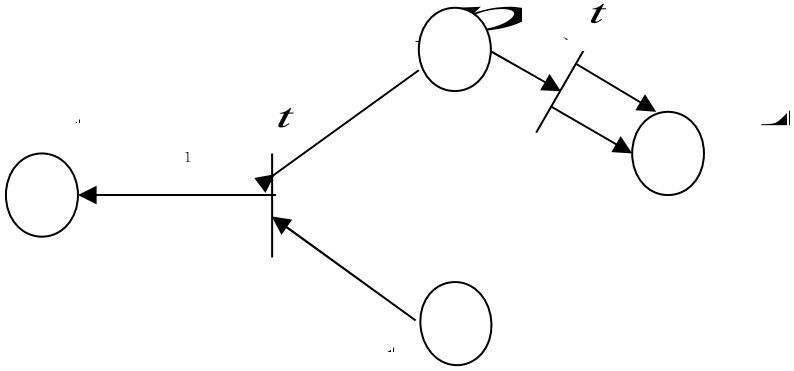


图 4.5 Petri 网的组成

两个用于转换的输入函数，用由位置指向转换的箭头表示，它们是：

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

$I(t_2)=\{P_2\}$

两个用于转换的输出函数，用由转换指向位置的箭头表示，它们是：

$O(t_1)=\{P_1\}$

$O(t_2)=\{P_3, P_3\}$

注意，输出函数 $O(t_2)$ 中有两个 P_3，是因为有两个箭头由 t_2 指向 P_3.

更形式化的 Petri网结构，是一个四元组 $C=(P,T,I,O)$ 。

其中，

$P:\{P_1, \cdots, P_n\}$ 是一个有穷位置集，$n \geq 0$ 。

$T:\{t_1, \cdots, t_m\}$ 是一个有穷转换集，$m \geq 0$，且T和P不相交 。

I

：$T \to P^{\infty}$为输入函数，是由转换到位置无序单位组（$bags$）的映射。

O ：$T \to P^{\infty}$为输入函数，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。

Petri网的标记是在Petri网中权标（ token ）的分配。例如，在图 4.6 中有 4 个权标，其中一个在 P_1 中，两个在 P_2 中，P_3 中没有，还有一个在 P_4 中。上述标记可以用向量 （ 1，2，0，1） 表示。由于 P_2 和 P_4 中有权标，因此 t 1 启动（即被激发）。通常，当每当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 t_1 被激发时， P_2 和 P_4 上各有一个权标被移出，而 P_1 上则增加一个权标，Petri网中权标总数不是固定的，在这个例子中两个权标被移出，而 P_1 上只有能增加一个权标。

在图 4.6 中 P_2 上有权标，因此 t_2 也可以被激发。当 t_2 被激发时，P_2 上将移走一个权标，而 P_3 上新增加两个权标，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子激发。图 4.6 所示，Petri网标记为（1，2，0，1），t_1 和 t_2 都可以 被激发。假设 t_1 被激发了，则结果如图 4.7 所示，标记，标记为（2，1，0，0）。此时，只有 t_2 可以被激发。

如果 t_2 也被激发了，则权标从 P_2 中移出，两个新权标被放在 P_3 上，结果如图 4.8 所示，标记为（2，0，2，0）。

t_2

t_1

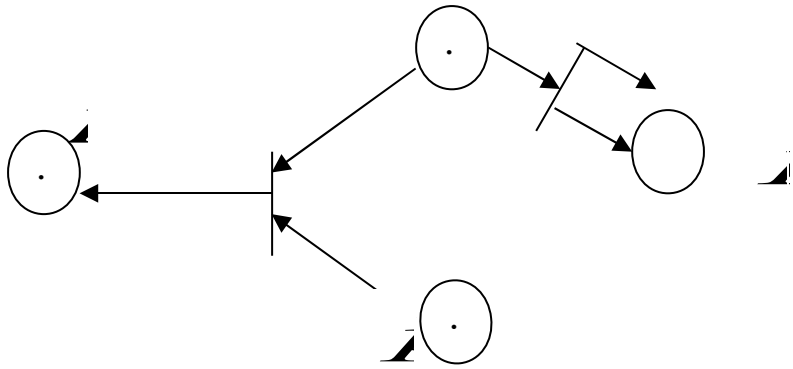


图 4.6 带标记的 Petri 网

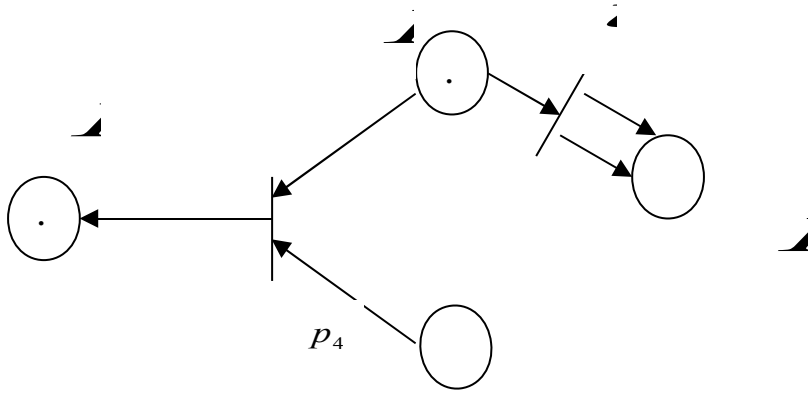


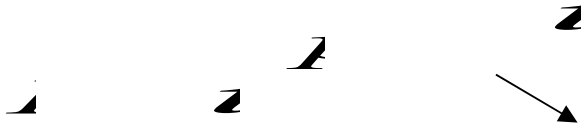
图 4.7 图 4.6 的 Petri 网的转换

更形式化地说，[illegible]由一组位置 P 到一组非负整数的映射：

$$M: P \to \{0,1,2,\cdots\}$$

这样，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。如图 4.9 所示，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。在图 4.9 中，P_3 上有一个权标而 P_2 上没有权标，因此转换 t 1 可以被激发。



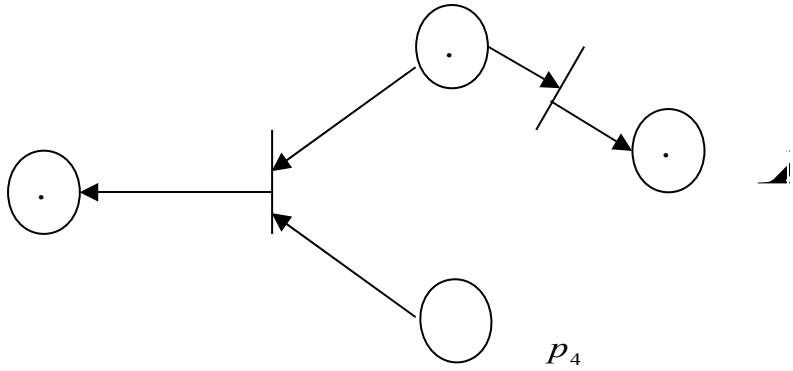


图 4.8 图 4.7 的 Petri 网在转换 t_2 被激发后的情况

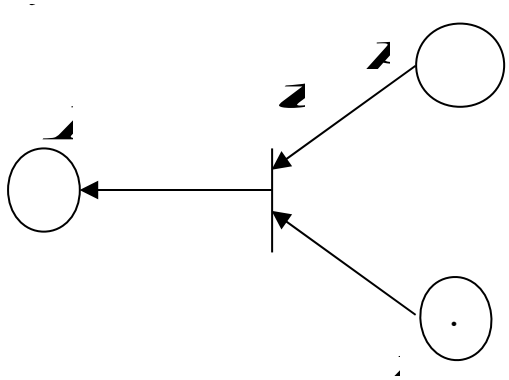


图 4.9 含禁止线的 Petri 网

4.3.2 例子

现在把 Petri网应用于上一节讨论过的电梯问题。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 F_f 代表 $(1 \le f \le m)$，在Petri网中电梯是用一个权标代表的。在位置 F_f 上有权标，表示在楼层 f 上有电梯。

1. 电梯按钮

电梯问题的第一约束条件描述了电梯按钮的行为，现在复述一下这个约束条件。第一条约束 C_1 ：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网中还必须设置其它的位置。电梯中楼层 f 的按钮，在Petri网中用位置 EB_f 表示$(1 \le f \le m)$。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。图 4,10 所示的Petri网准确地描述了电梯按钮的行为规律。首先，假设按钮没有发亮，显然在位置 EB_f 上没有权标，从而在存在禁止线的情况下，转换“ EB_f 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 EB_f 上放置了一个权标，如图 4.10 所示。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 “ EB_f 被按下” 不能再被激发了，因此，位置 EB_f 上的权标数不会多于 1 。

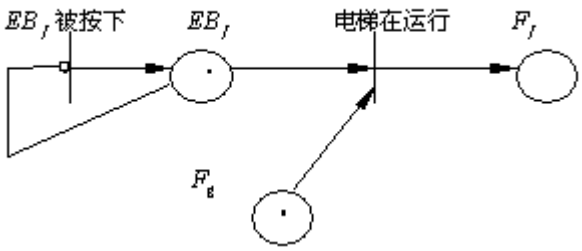


图 4.10 Petri 网表示的电梯按钮

假设电梯由 g 层驶向 f 层，因为电梯在 g 层，如图 4.10 所示，位置 F_g 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 EB_f 和 F_g 上的权标被移走，按钮 EB_f 被关闭，在位置 F_g 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。也就是说，在标准 Petri网中转换是瞬间完成的，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。

2. 楼层按钮

在第二个约束条件中描述了楼层按钮的行为。

第二条约束 C_2 ： 除了第一层与顶层之外，每个楼台层都有两个按钮，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，相应的按钮才什么熄灭。

在Petri网中楼层按钮用位置和 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 （$1<f<m$）.

图4.11所示的情况为电梯由 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。图 4.11所示的Petri网可以保证，当两个按钮都亮了时候，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网网模型，对此本书不做更进一步的介绍。

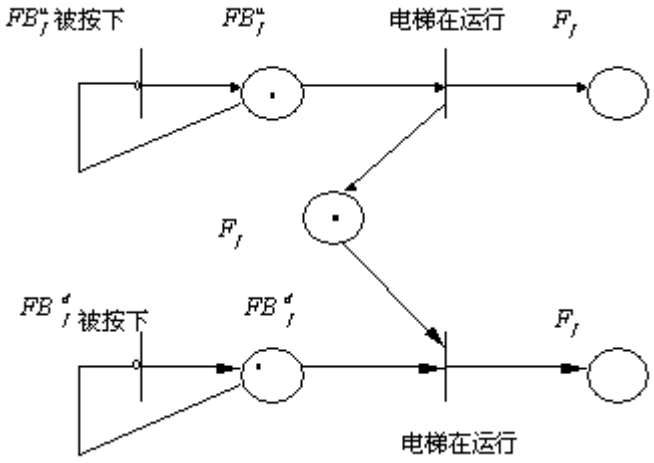


图 4.11 Petri 网表示楼层按钮

最后，考虑

第三条约束 C_2 ： 当电梯没有收到请求时，它将停留在当前楼层并关门。

这条约束很容易实，如图 4.11 所示，当没有请求 (FB^{u}_{f} 和 FB^{d}_{f} 上无权标）时，任何一个转换“电梯在运行”都不能被激发。

4.4 Z语言

在形式化的规格说明语言中， Z 语言赢得了广泛的赞誉。使用 Z 语言需要具备集合论、函数、数理逻辑等方面的知识。即使用户已经掌握了所需要的背景知识， Z 语言也是相当难学的，因为它除了使用常用的集合论和数理逻辑符号之外，还使用一些特殊符号。

4.4.1 简介

本节结合电梯问题的例子，简要地介绍 Z’ 语言。

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。

现在依次介绍这 4 个部分。

1. 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。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]

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。例如，格 S 的格式如图 4．12所示。

在电梯问题中， Button有 4 个子集，即 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）以及 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 ）。图 4．13描述了格Button_State，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（在下面

的讨论中并不需要 floor_buttons 集和elevator_buttons集，把它们放于图 4．13中只是用来说明 Z 格包含的内容）。

图 4.12 Z 格 S 的格式

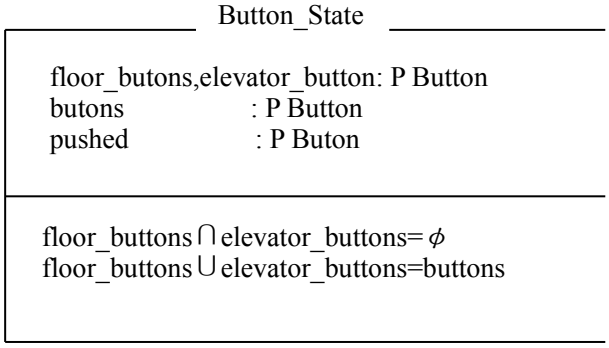


图 4.13 Z 格 Button_State

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。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。

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。图 4．14定义了操作Push_Button(按按钮）。Z 语言的语法规定，当一个格被用在另一个格中时，要在它的前面加上三角形符号△作为前缀，因此，格 Push_Button 的第一行最前面有一个三角形符号作为格Button_State的前缀。操作 Push_Button 有一个输入变量“button?"。问号“?”表示输入变量，而感叹号“!”代表输出变量。

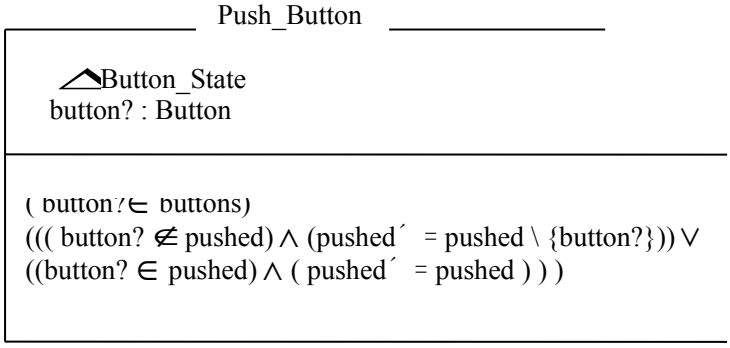


图 4． 14 操作 Push_Button 的 Z 规格说明

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）。

图 4．14中的第一个前置条件规定，“button?"必须是 buttons 的一个元素，而 buttons
是电梯系统中所有按钮的集合。如果第二个前置条件button?∉ pushed 得到满足（即按钮没有开启），则更新pushed按钮集，使之包含刚开启的按钮 "button?" 。在 Z 语言中，当
一个变量的值发生改变时，就用符号 “ ′ ”表示。也就是说，后置条件是当执行完操作
Push_Button 之后，“ button?” 将被加入到pushed集中。无需直接打开按钮，只要使“button?"变成pushed中的一个元素即可。

还有一种可能性是，被按的按钮原先已经打开了。由于 button?∈ pushed ，根据第三个前置条件，将没有任何事情发生，这可以用 pushed ′ = pushed 来表示，即pushed的新状态和旧状态一样。注意，如果没有第三个前置条件，规格说明将不能说明在一个按钮已被按过之后又被按了一次的情况下将发生什么事，因此，结果将是不可预测的。

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，如图 4．15所示（符号“＼”表示集合差运算）。但是，如果按钮
“ button?” 原先没有打开，则pushed集合不发生变化。

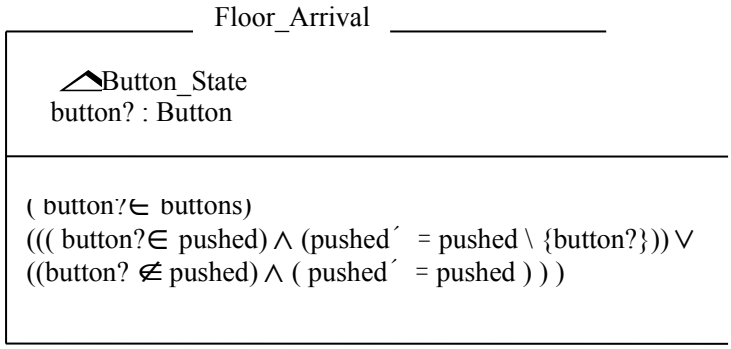


图 4． 15 操作 Floor_Arrival 的 z 规格说明

本节的讨论有所简化，没有区分上行和下行楼层按钮，但是，仍然讲清了使用 Z 语言

说明电梯问题中按钮状态的方法。

4.4.2 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，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。

(2) 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。

(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。

(5) 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。

(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。

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。

4.5 小 结

基于数学的形式化规格说明技术，目前还没有在软件产业界广泛应用，但是，与欠形式化的方法比较起来，它确实有实质性的优点：形式化的规格说明可以用数学方法研究、验证（例如，一个正确的程序可以被证明满足其规格说明，两个规格说明可以被证明是等价的，规格说明中存在的某些形式的不完整性和不一致性可以被自动地检测出来）。此外，形式化的规格说明消除了二义性，而且它鼓励软件开发者在软件工程过程的早期阶段使用更严格的方法，从而可以减少差错。

当然，形式化方法也有缺点：大多数形式化的规格说明主要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，形式化方法比欠形式化方法更难学习，不仅在培训阶段要花大量的投资，而且对某些软件工程师来说，它代表

了一种“文化冲击”。

把形式化方法和欠形式化方法有机地结合起来，使它们取长补短，应该能获得更理想
的效果。本章讲述的应用形式化方法的准则（见 4．1．3 节），对于读者今后在实际工作中
更好地利用形式化方法，可能是有帮助的。

本章简要地介绍了有穷状态机、 Petri 网和 Z 语言等三种典型的形式化方法，使读者
对它们有初步的、概括的了解。当然，要想在实际工作中使用这些方法，还需要进一步研读有关的专著。

习 题 4

1．举例对比形式化方法和欠形式化方法的优缺点。

2．在什么情况下应该使用形式化说明技术？使用形式化说明技术时应遵守哪些准则？

3．一个浮点二进制数的构成是：一个可选的符号（+或一），后跟一个或多个二进制
位，再跟上一个字符 E，再加上另一个可选符号(+或一）及一个或多个二进制位。例如，下列的字符串都是浮点二进制数：

110101E 一 101

— 100111E11101

+1E0

更形式化地，浮点二进制数定义如下：

<floating-point binary>：： =[(sign)] < bitstring)E[<sign>] <bitstring>

<sign> : : =+ | -

<bitstring> : : =<bit> [<bitstring>]

<bit> : : =0 | 1

其中，

符号 ：： = 表示定义为；

符号 […]表示可选项；

符号 a|b 表示 a 或 b 。

假设有这样一个有穷状态机：以一串字符为输入，判断字符串中是否含有合法的浮点
二进制数。试对这个有穷状态机进行规格说明。

4．考虑下述的自动化图书馆流通系统：每本书都有一个条形码，每个借阅人都有一
个带有条形码的卡片。当一个借阅人想借一本书时，图书管理员扫描书上的条形码和借阅人卡片上的条形码，然后在计算机终端上输入 C；当归还一本书时，图书管理员将再做一次扫描，并输入 R 。图书管理员可以把一些书加到 (+) 图书集合中，也可以删除 (-) 它们。借阅人可以在终端上查找到某个作者所有的书（输入“A=”和作者名字），或具有指

定标题的所有书籍（输入“T=”和标题），或属于特定主题范围内的所有图书（输入“S=”
加主题范围）。最后，如果借阅人想借的书已被别人借走，图书管理员将给这本书设置一个预约，以便书归还时把书留给预约的借阅人（输入“H=”加书号）。

试用有穷状态机说明上述的图书流通系统。

5．试用 Petri 网说明第 4 题所述图书馆中一本书的循环过程。在规格说明中应该包
括操作 H 、 C 及 R 。

6．试用 Z 语言对第 4 题所述图书馆图书流通系统做一个完整的规格说明。