状态图

状态机

- 状态机**Statechart**
  - 用于描述一个对象在其生存期间的动态行为，表示对象响应事件所经历的状态序列以及伴随的动作。
  - 状态机是用于描述状态与状态转换的。从对象的初始状态起，开始响应事件并执行某些动作，这些事件引起状态的转换；对象在新状态下又开始响应事件和执行动作，如此连续直到终结状态。
  - **状态机由状态、转换、事件、活动和动作5部分组成。**

状态机

1. 状态

状态指的是对象在其生命周期中的一种状况，处于某个特定状态中的对象必然会满足某些条件、执行某些动作或者等待某些事件。一个状态的生命周期是一个有限的时间阶段。

2. 动作

动作指的是状态机中可以执行的那些原子操作。所谓原子操作，指的是它们在运行的过程中不能被其他消息所中断，必须一直执行下去，最终导致状态的变更或者返回一个值。

状态机

3. 事件

事件指的是发生在时间和空间上的对状态机有意义的事情。事件通常会引起状态的变迁，促使状态机从一种状态切换到另一种状态，如信号、对象的创建和销毁等。

4. 活动

活动指的是状态机中进行的非原子操作。

5. 转换

转换指的是两个不同状态之间的一种关系，表明对象将在第一种状态中执行一定的动作，并且在满足某个特定条件下由某个事件触发进入第二个状态。

状态机

- 在UML中，状态机由对象的各种状态和连接这些状态的转换组成，是展现状态与状态转换的图。在面向对象的软件系统中，一个对象必然会经历一个从开始创建到最终消亡的完整过程，这个过程称为对象的生命周期。对象在其生命期内必然会接受消息来改变自身或者发送消息来影响其他对象。
- 状态机用于说明对象在其生命周期中响应事件所经历的状态序列以及其对这些事件的响应。
- 在状态机的语境中，一个事件就是一次激发的产生，每一个激发都可以触发一个状态转换。

状态机

- 状态机常用于对模型元素的动态行为进行建模，即对系统行为中受事件驱动的方面进行建模。不过状态机总是一个对象、协作或用例的局部视图。由于它考虑问题时，将实体与外部世界相互分离，所以适合对局部细节进行建模。
- 一个状态机依附于一个类，并且描述该类的实例（即对象）对接收到的事件的响应。状态机还可以依附于用例、操作等，用于描述它们的动态执行过程。在依附于某一个类的状态机中，总是将对象孤立地从系统中抽象出来进行观察，而来自外部的影响则都抽象为事件。

状态图与状态机

- 在日常生活中，事物状态的变化是无时不在的。例如，使用银行的ATM机，当一部ATM机没有人使用时它处于闲置状态，当插入银行卡进行存、取款操作时，ATM机处于工作状态。使用完毕退出银行卡后，ATM机又回到了闲置状态。使用状态图就可以描述ATM机整个状态变化的过程。
- 状态图(Statechart Diagram)描述一个实体基于事件反应的动态行为，显示了实体如何根据当前所处的状态对不同的事件做出反应的。
- 创建UML状态图是为了以下的研究目的：研究类、角色、子系统、或组件的复杂行为。

状态图与状态机

- 状态图(Statechart Diagram)本质上就是一个状态机，或者是状态机的特殊情况，它基本上是一个状态机中的元素的一个投影，这也就意味着状态图包括状态机的所有特征。
- 状态图用于对系统的动态方面建模，适合描述跨越多个用例的对象在其生命周期中的各种状态及其状态之间的转换。这些对象可以是类、接口、构件或者节点。状态图常用于对反应型对象建模，反应型对象在接收到一个事件之前通常处于空闲状态，当这个对象对当前事件作出反应后又处于空闲状态，等待下一个事件发生。

活动图和状态图的区别

- 状态图和活动图是状态机的两种表现形式，利用状态机可以精确地描述对象的行为。
- 活动图着重表现从一个活动到另一个活动的控制流，是内部处理驱动的流程。一般是一个结束后，自动转入下一个活动。
- 状态图着重描述从一个状态到另一个状态的流程，主要由外部事件的参与。
- 活动图是一种特殊的状态图，如果在一个状态图中大多数状态是表示操作的活动，而转移是由状态中动作的完成来触发，即全部或绝大多数的事件是由内部产生的动作完成的，则该状态图是活动图。

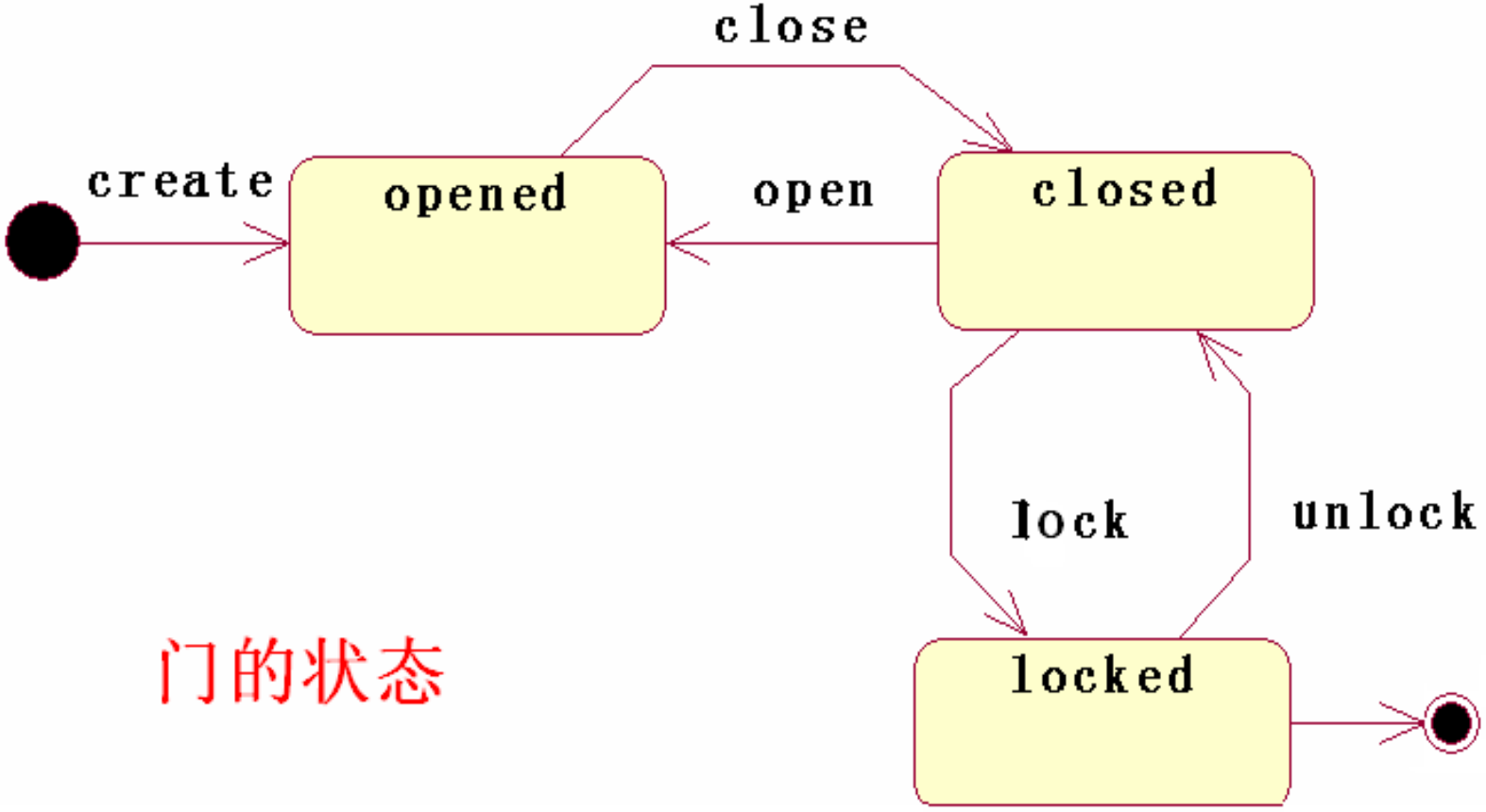
状态图（Statechart Diagram）

· 状态图主要用于描述一个对象在其生存期间的动态行为，表现一个对象所经历的状态序列，引起状态转移的事件(event)，以及因状态转移而伴随的动作(action)。

· 一个状态机可以被一个状态图描述，对于一个比较复杂的状态机，也可用多张状态图来表示。

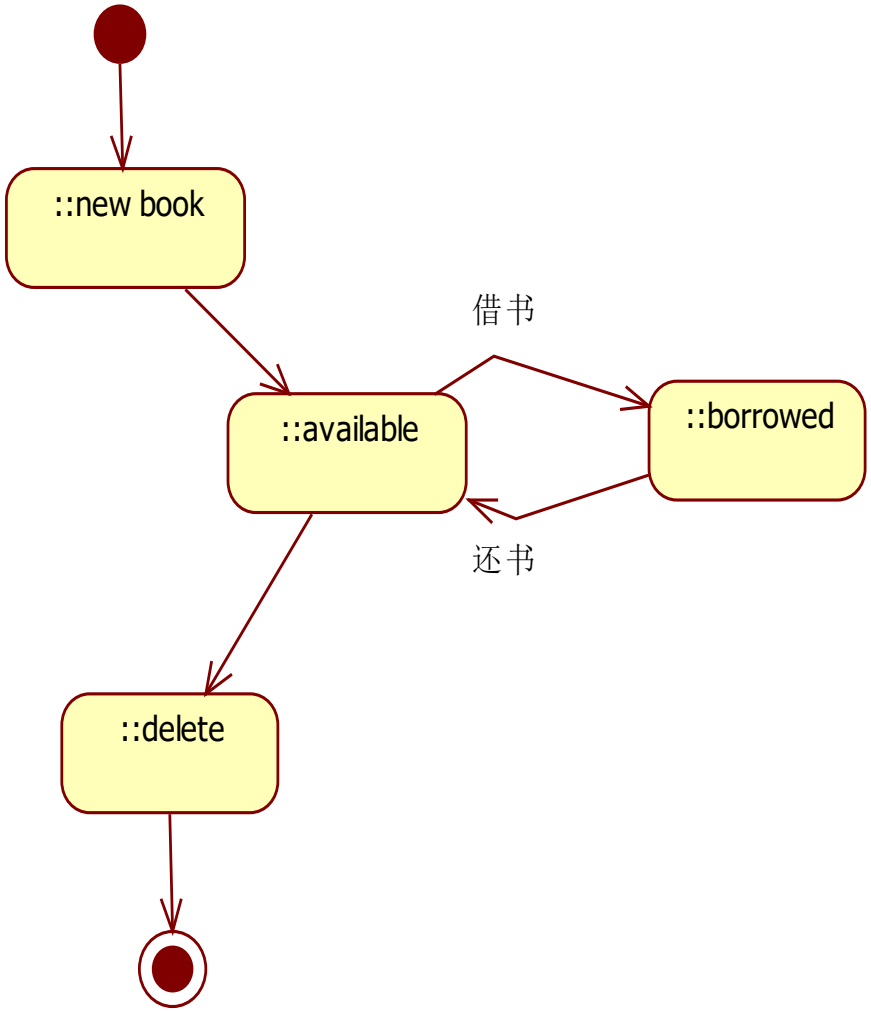
· 状态图表现从一个状态到另一个状态的控制流。

一个门的状态图



- 状态图是为系统的动态行为建模，是系统分析的常用工具，因为系统中对象状态的变化比较容易发现和理解。

图书馆书籍的状态图



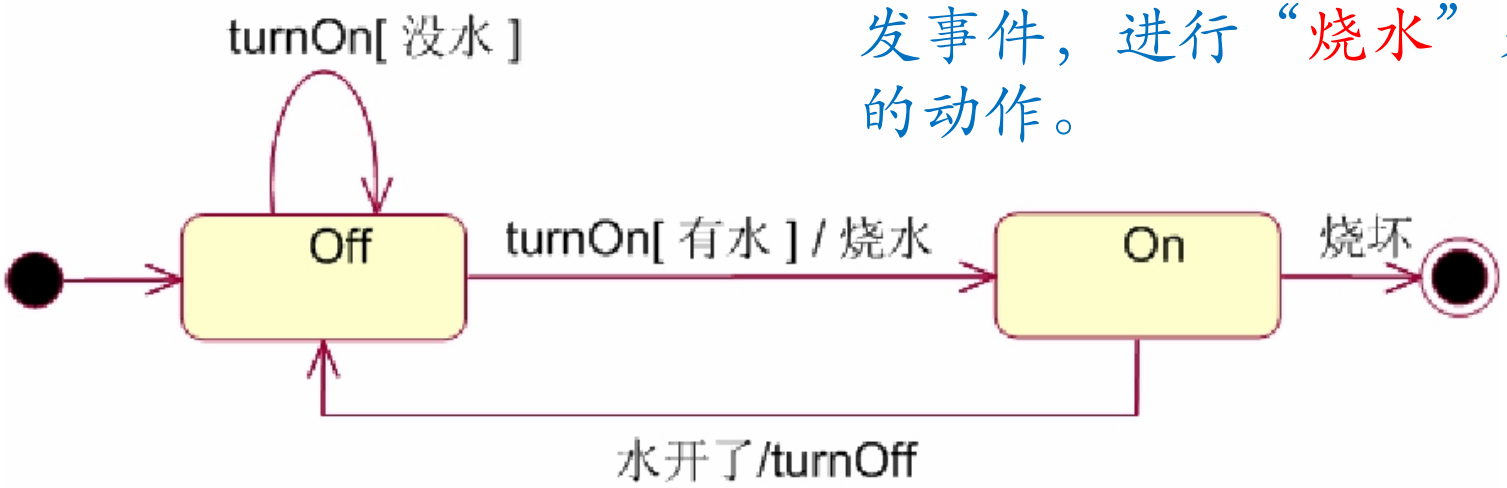
一本图书对象从它的起始点开始，首先是“新书”状态（new book），然后是“可以借阅”（available）的状态，如果有读者将书借走，则该书的状态为“已借出”状态（borrowed），如果图书被归还图书馆，图书的状态又变为“可以借阅”状态。图书馆如果放弃该图书对象的收藏，则图书对象处于“删除”状态（delete），最后到达“终止”状态。

状态图的组成

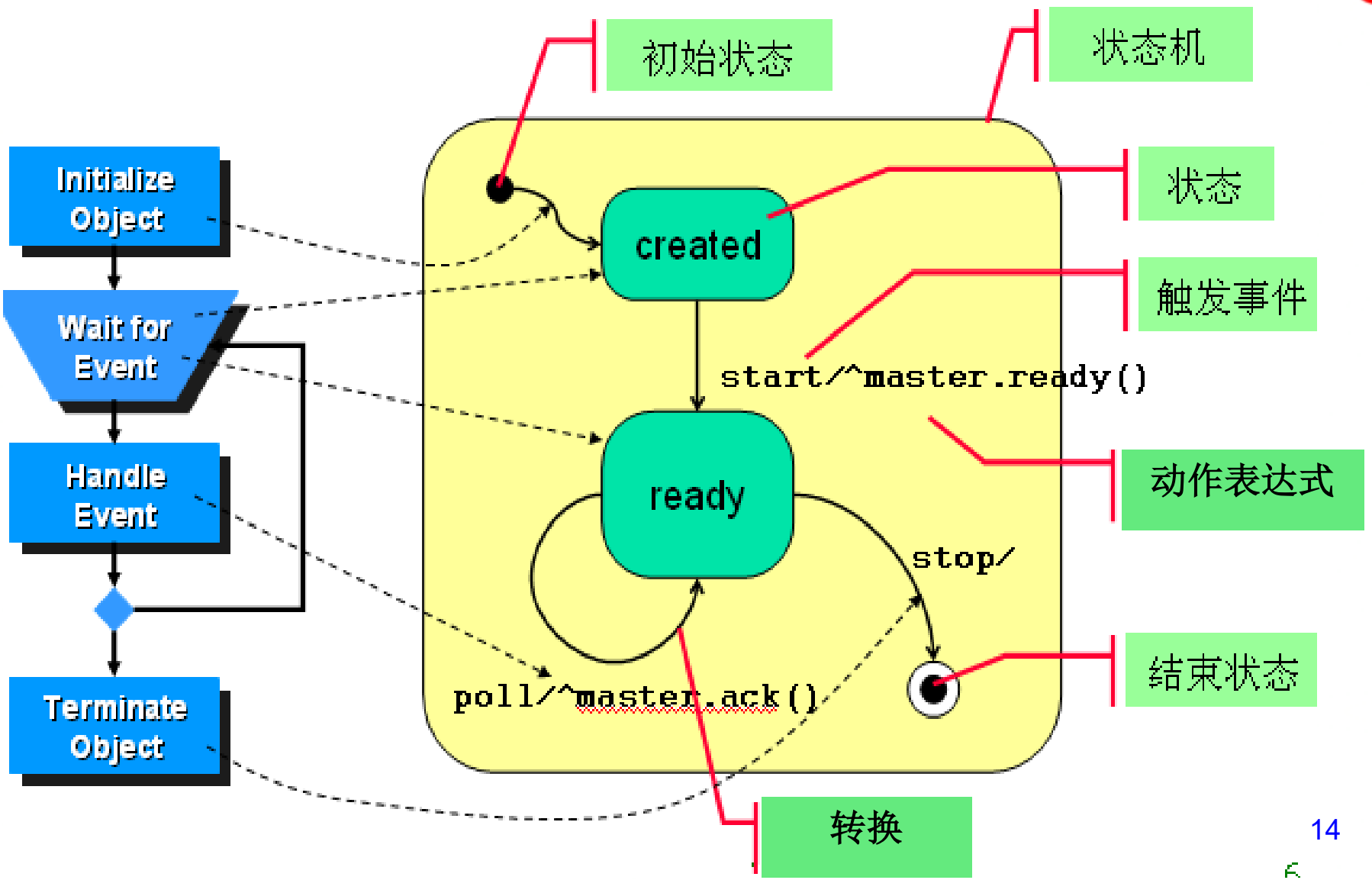
(1) 状态图中的事物

- 状态 **state**
- 转换/迁移 **transition**
- 事件、动作

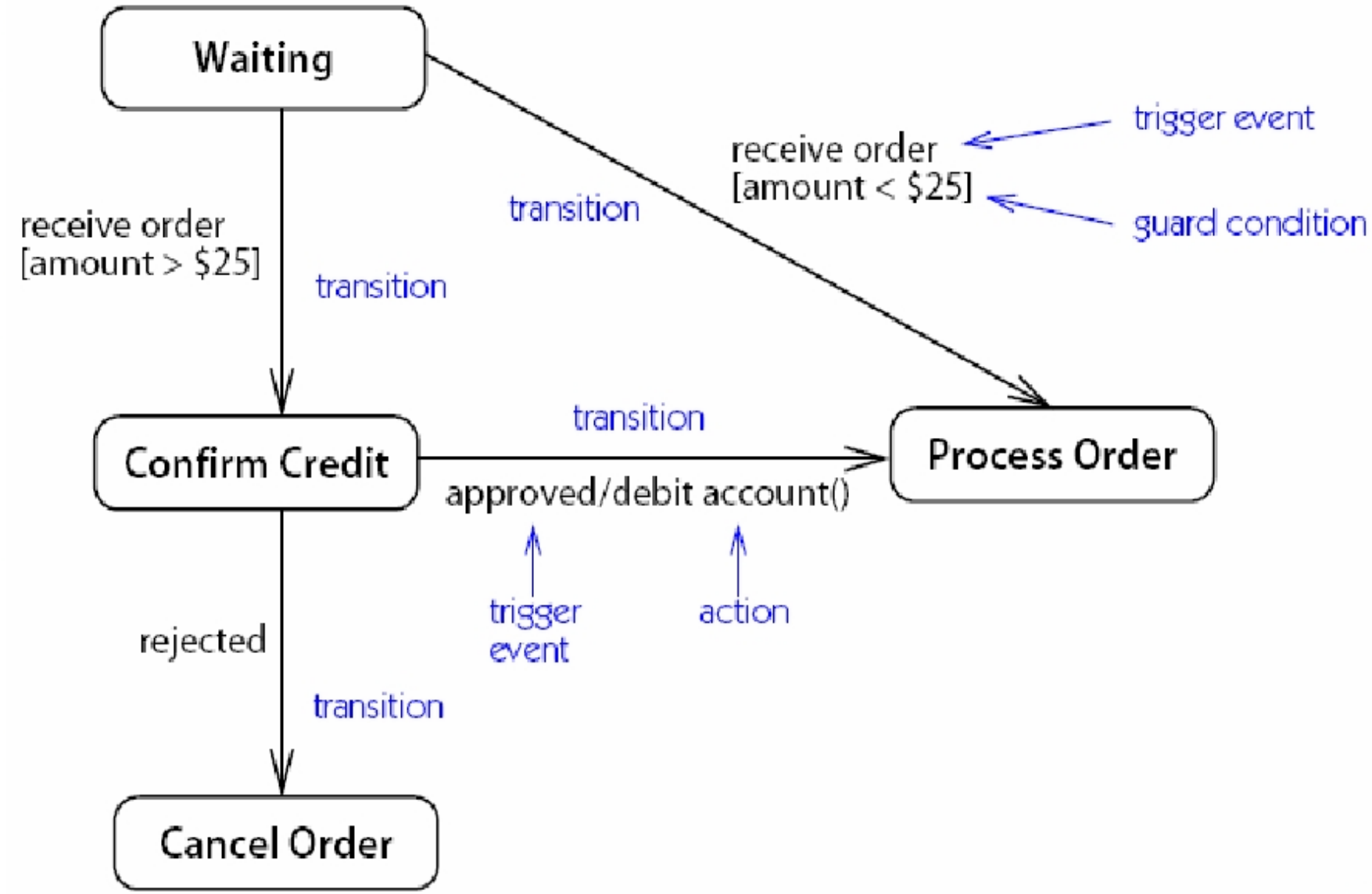
在用电磁炉烧开水的过程中，水的状态由源状态“Off”（不沸腾）转换为目标状态的“On”（沸腾）时，水壶中“有水”就是其监护条件，开启电源开关“turnOn”是其触发事件，进行“烧水”是状态转换的动作。



状态图的组成


初始状态
状态机
状态
触发事件
动作表达式
结束状态
转换
Initialize Object
Wait for Event
Handle Event
Terminate Object
created
ready
start/^master.ready()
stop/
poll/^master.ack()

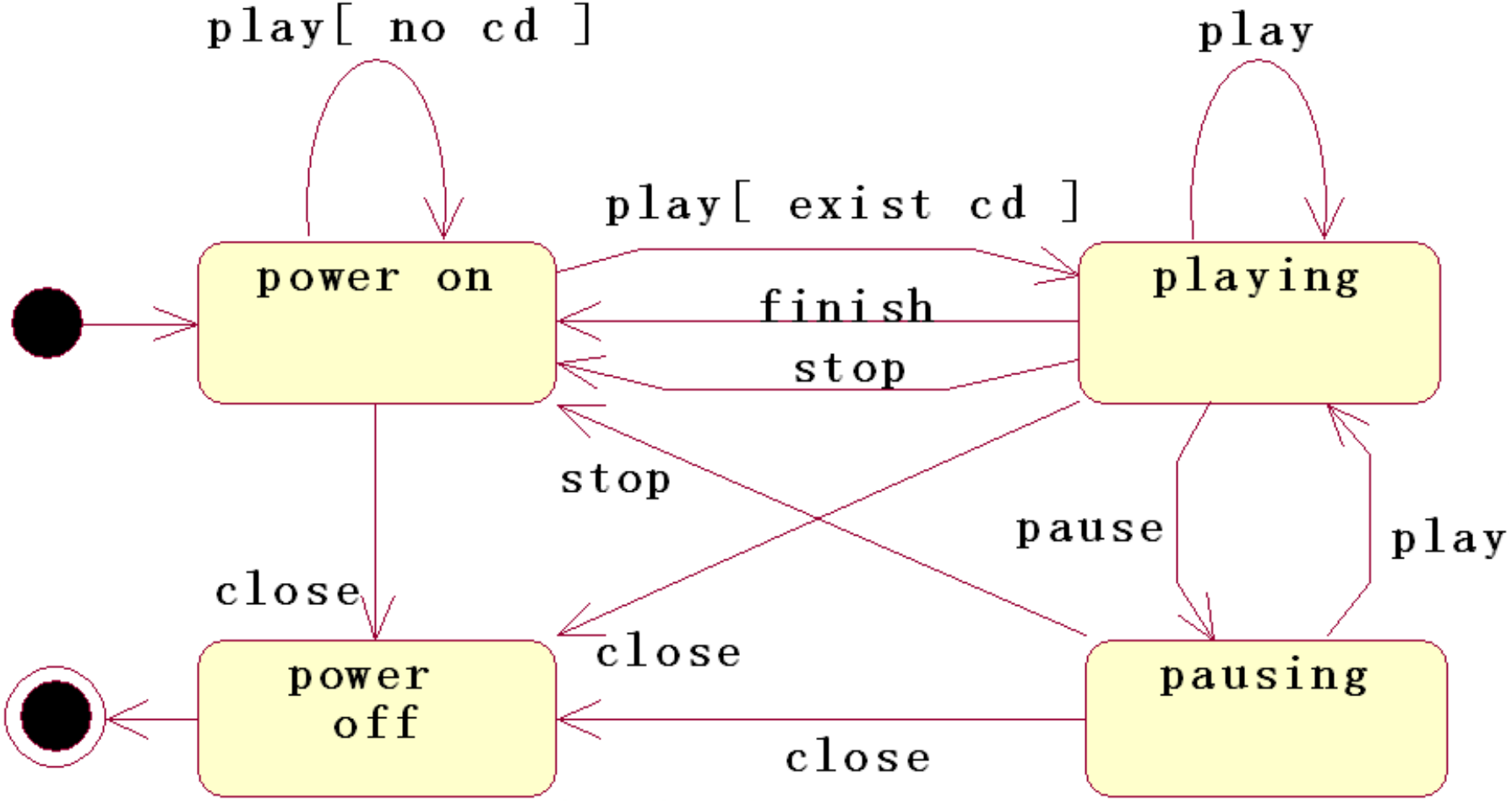
状态图的组成（刷POS机）



状态

- 状态是指在对象的生命期中满足某些条件、执行某些活动或等待某些事件时的一个条件或状况。下面一些例子形象地说明了对象和状态。
  - （1）支票（对象）已付（状态）。
  - （2）汽车（对象）已启动（状态）。
  - （3）小王（对象）睡着了（状态）。
  - （4）小红（对象）未婚（状态）。
  - 对象在任何时候都会处于某种状态中，所有对象都有状态
  - 对象所处的状态决定了它如何响应所检测到的事件或所接收的消息。
  - 清醒—（被批评）—生气　醉酒—（被批评）—无反应
  - 通常，事件使对象从一个状态转向另一个状态（即状态的转换）

- CD Player

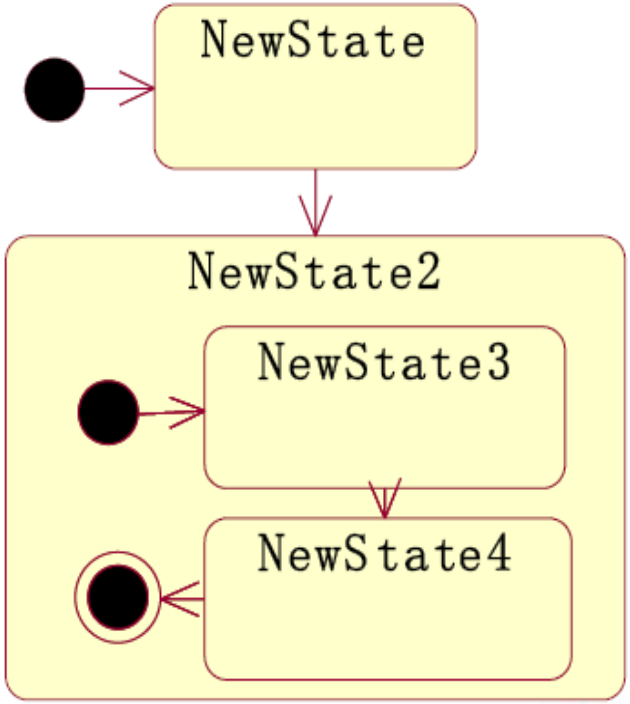

play[no cd]
play
play[exist cd]
power on
playing
finish
stop
stop
pause
play
close
close
power
off
pausing
close

状态

- 状态的类型：
  - 初态
  - 终态
  - 中间状态
  - 组合状态
  - 历史状态

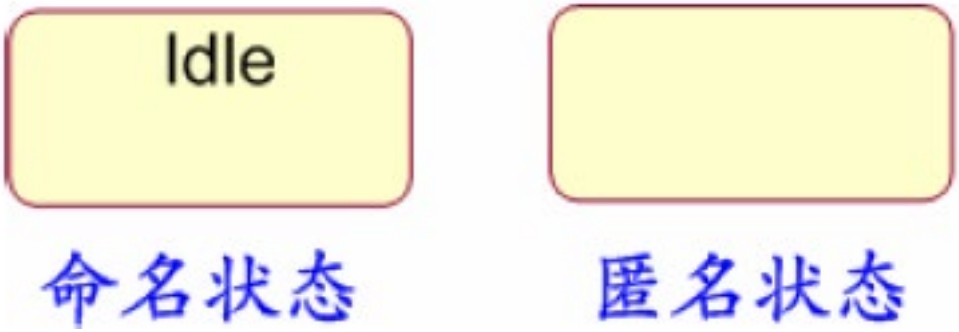
1. 初态和终态

- 初态 （start state）
  - 显示状态图中状态机执行的开始
  - 一个状态机只能有一个初态，因为每一个执行都是从相同的地方开始的。如果一个状态机用多张状态图描述，则多张图用一个初态。
  - 嵌套状态中可以使用新的初态。
- 终态 （end state）
  - 表示最后的或者终端状态；
  - 终态数目可以不确定，
    一个或多个也可以没有。



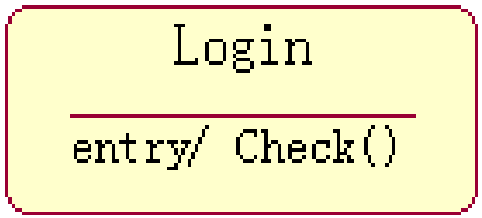
2. 中间状态

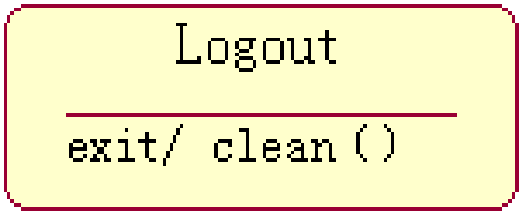
- 中间状态的组成（除初态终态外，最常见的状态）
  - 状态名（name）
    - 是可以把该状态和其他状态区分开的字符串;状态也可能是匿名的，即没有名称。



2. 中间状态

- 中间状态的组成（除初态终态外，最常见的状态）
  - 入口动作（entry action）用来指定进入状态时发生的动作。
  - 出口动作（exit action）用来指定离开该状态时发生的动作。





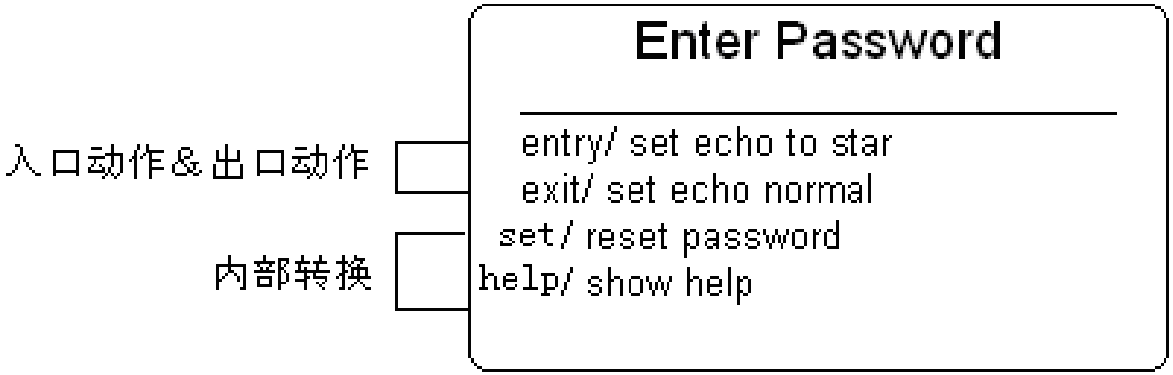
注意：由于入口动作和出口动作是隐式地激活，因此它们既没有参数也没有守卫条件。

进入/退出动作（entry/exit action）

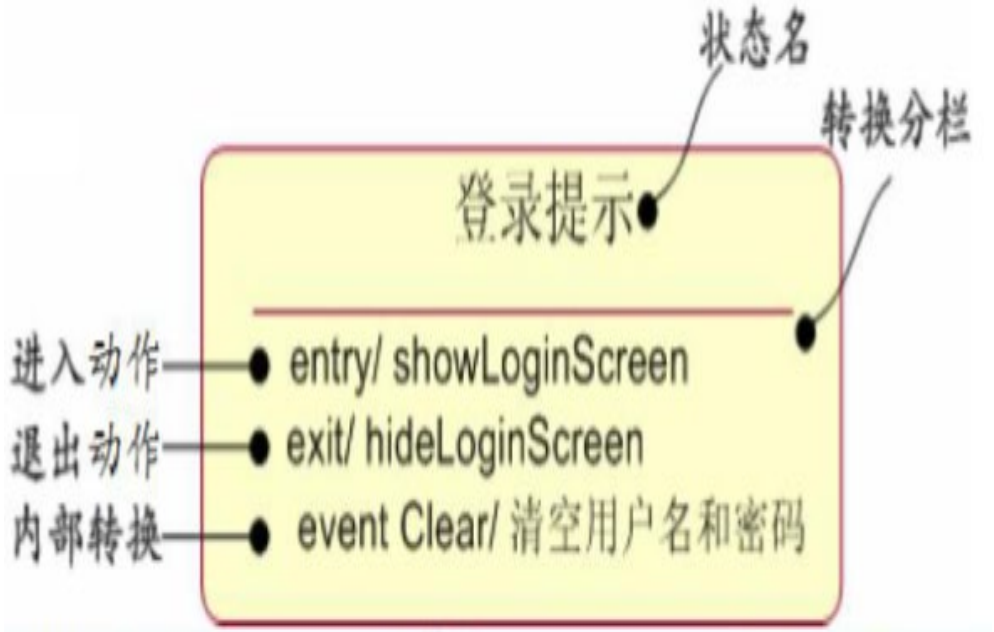
- **Action**
  - 可执行的原子计算。
  - 不可中断，其执行时间可忽略不计。
- **两种特殊动作：**
  - 进入动作（entry action）
    - ▶进入状态时执行的动作；Entry / setMode(onTrack)
  - 退出动作（exit action）
    - ▶退出状态时执行的动作；Exit / setMode(offTrack)

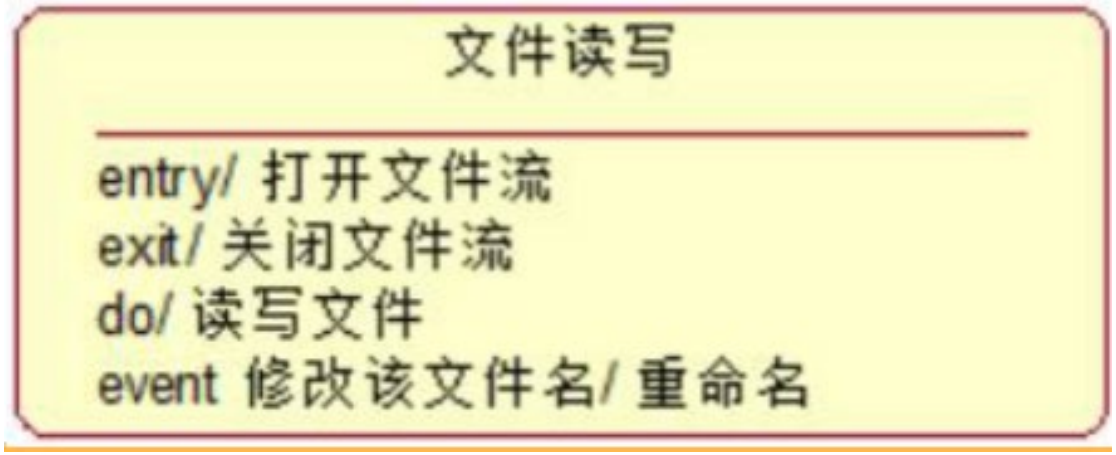
2. 中间状态

- 中间状态的组成（除初态终态外，最常见的状态）
  - 内部转换（internal transition）
    - 不导致状态改变的转换，不会执行 entry 和 exit 动作。
    - 内部转移不会改变对象的状态，内部转移在入口动作执行完毕后开始执行。
    - 用来处理一些不离开该状态的事件。



2. 中间状态


状态名
转换分栏
登录提示
进入动作
entry/ showLoginScreen
退出动作
exit/ hideLoginScreen
内部转换
event Clear/ 清空用户名和密码


文件读写
entry/ 打开文件流
exit/ 关闭文件流
do/ 读写文件
event 修改该文件名/ 重命名

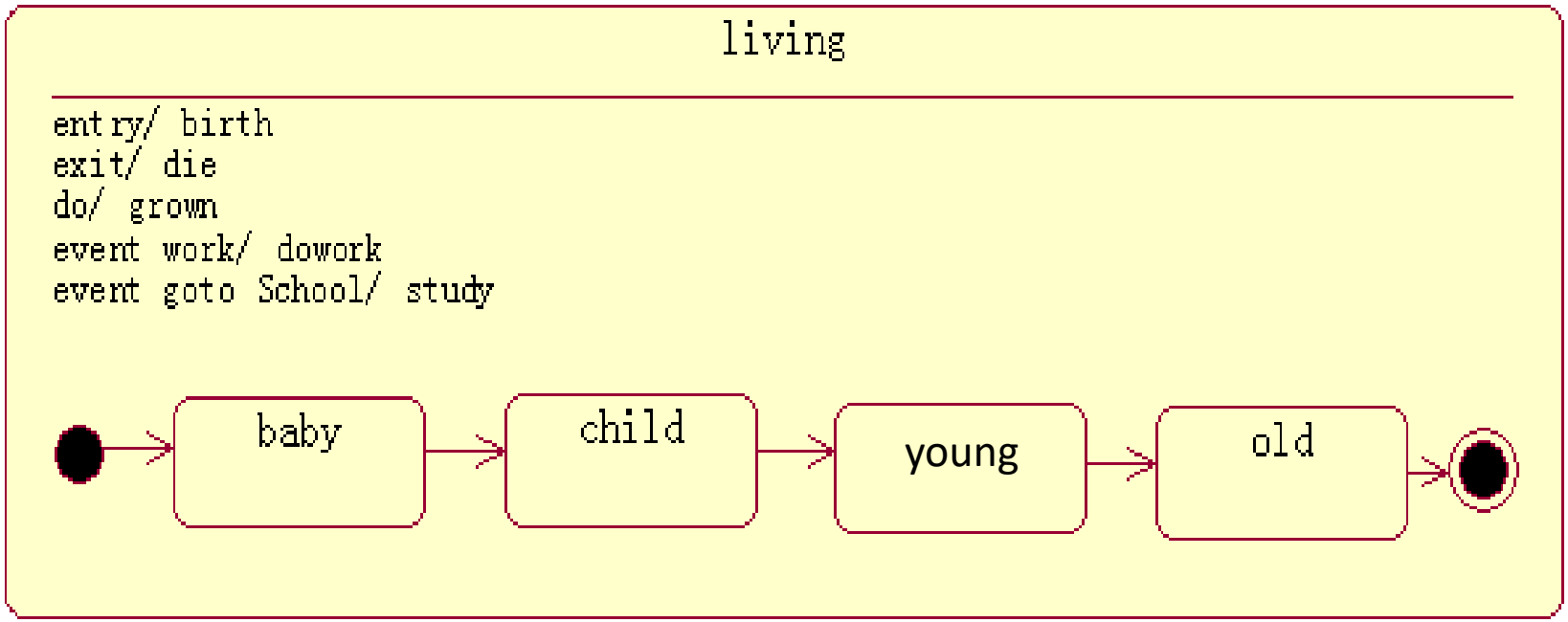
2. 中间状态

- 延迟事件
  - 是一种特殊的事件，它是指该事件不会触发状态的转换，当对象处于该状态时事件不会丢失，但会被延迟执行。
  - 例如，当E-mail程序中正在发送第一封邮件时，用户下达发送第二封邮件指令就会被延迟，但第一封邮件发送完成后，这封邮件就会被发送。这种事件就属于延迟事件
  - 延迟事件（deferred event）
    - 是指在该状态下暂不处理，但将推迟到该对象的另一个状态下排队处理的事件
  - 语法形式为　　延迟事件/defer

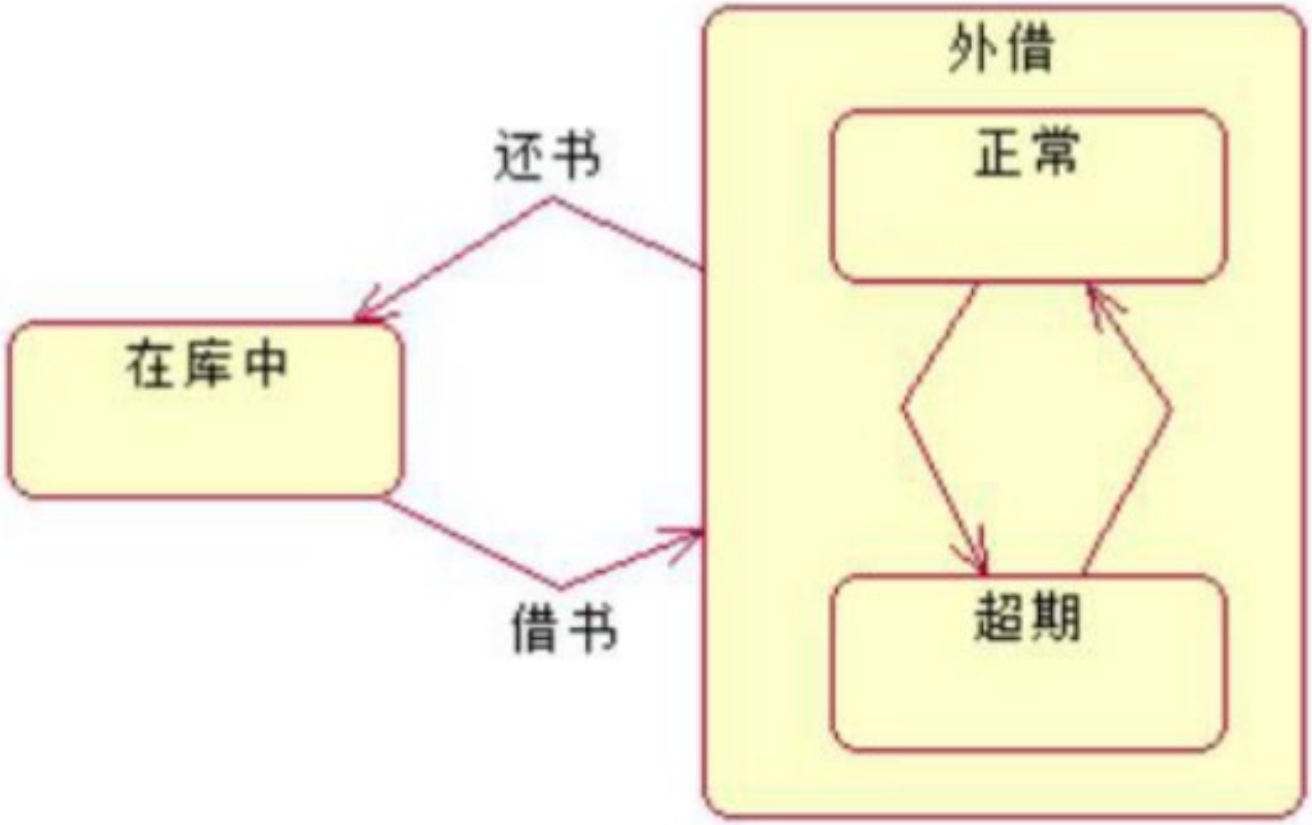
2. 中间状态

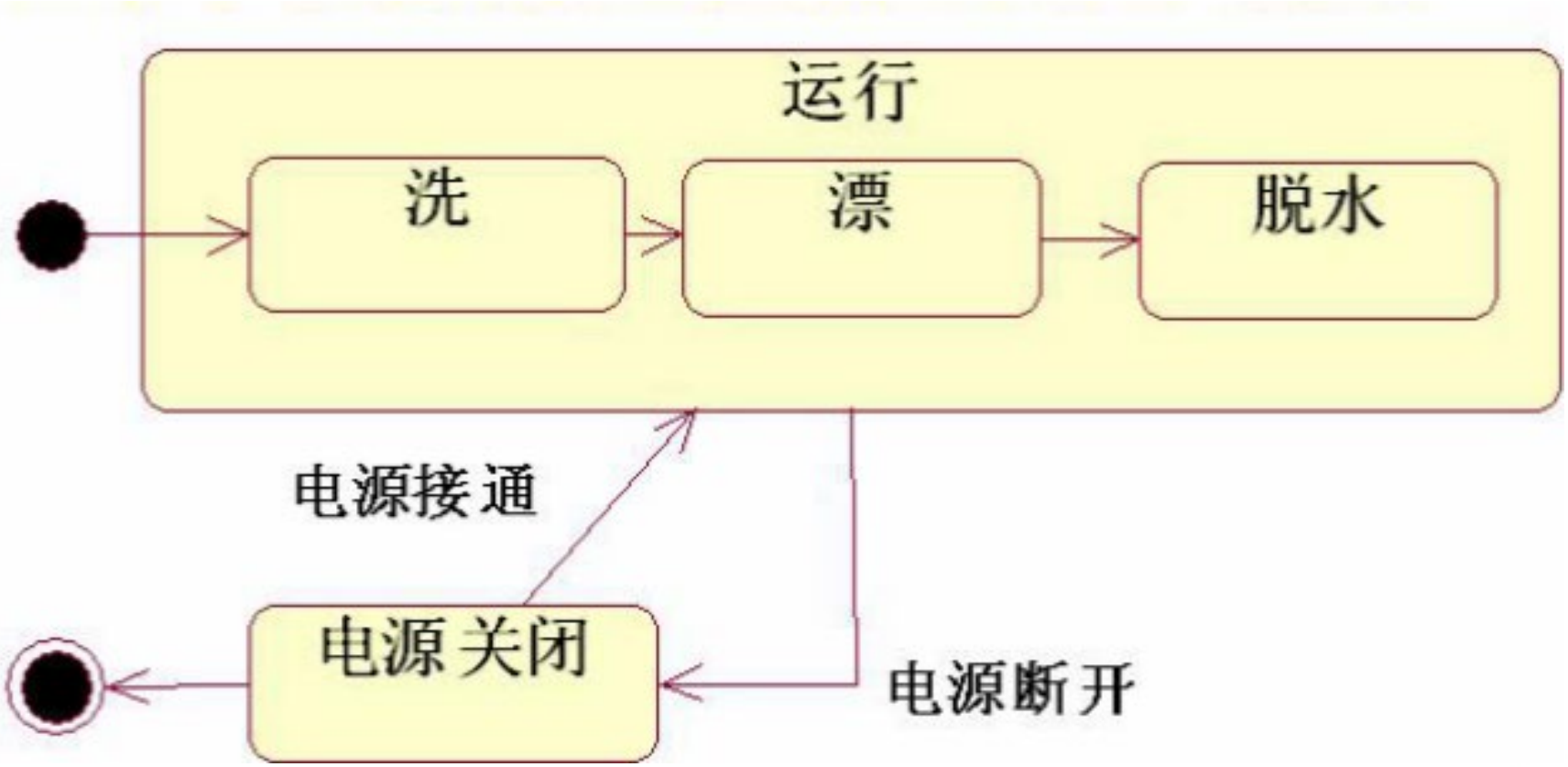
- 子状态（substate）：嵌套在另外一个状态中的状态

例如：空调：停止、运行状态，运行状态中可嵌套制冷、制热、除湿等子状态

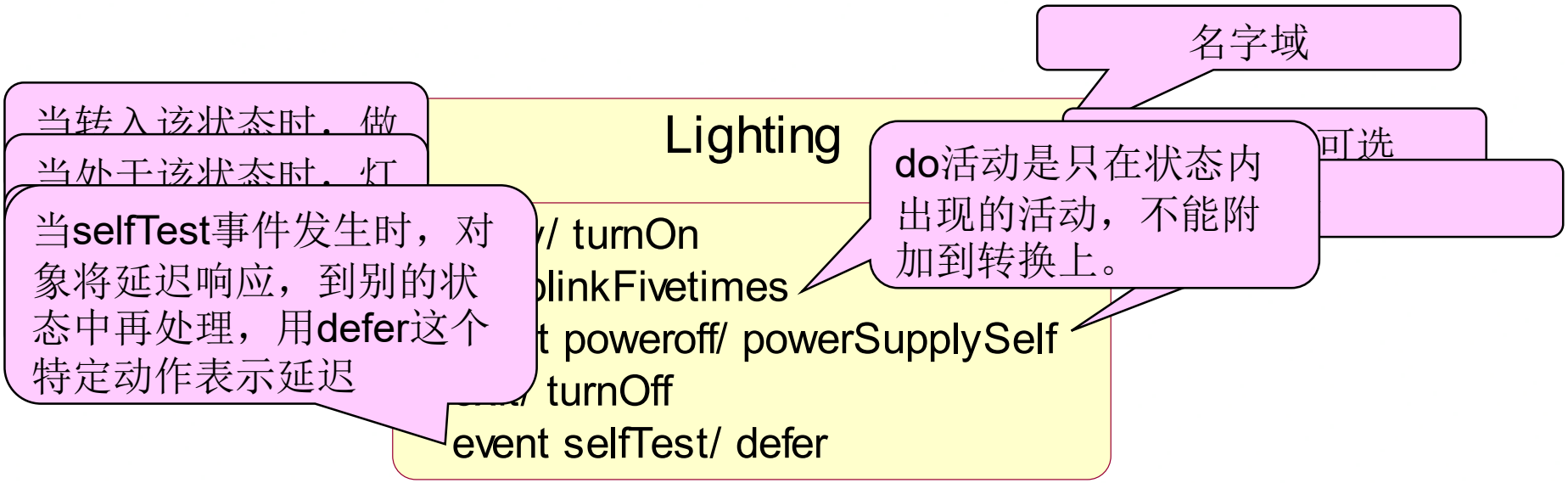


嵌套在另外一个状态中的状态



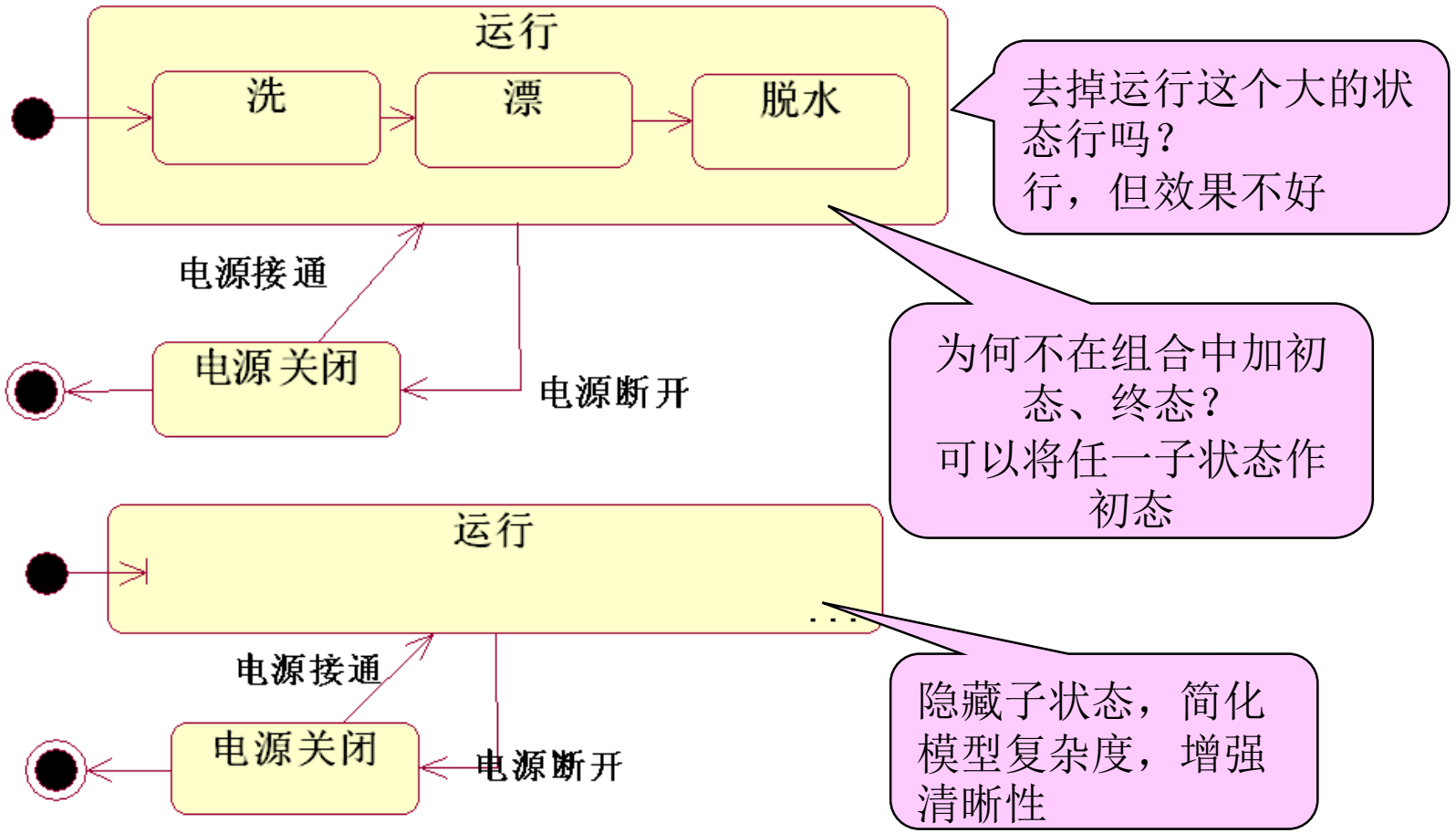

运行
洗
漂
脱水
电源接通
电源关闭
电源断开

中间状态举例


名字域
当转入该状态时，做
当处于该状态时，灯
Lighting
可选
当selfTest事件发生时，对象将延迟响应，到别的状态中再处理，用defer这个特定动作表示延迟
/ turnOn
linkFivetimes
t poweroff/ powerSupplySelf
/ turnOff
event selfTest/ defer
do活动是只在状态内出现的活动，不能附加到转换上。

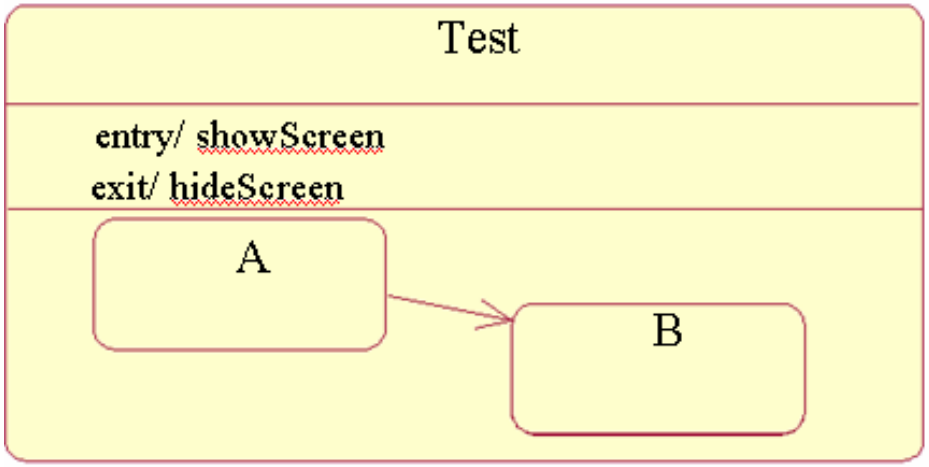
3. 复合状态

- 含有子状态的状态被称为组合或嵌套状态

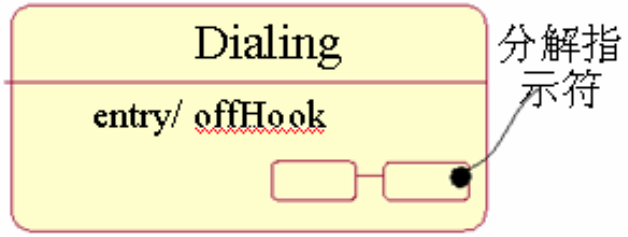


3. 复合状态

- 含有子状态的状态被称为复合状态。
- 包括顺序和并发的复合状态
- 两种表示方法：



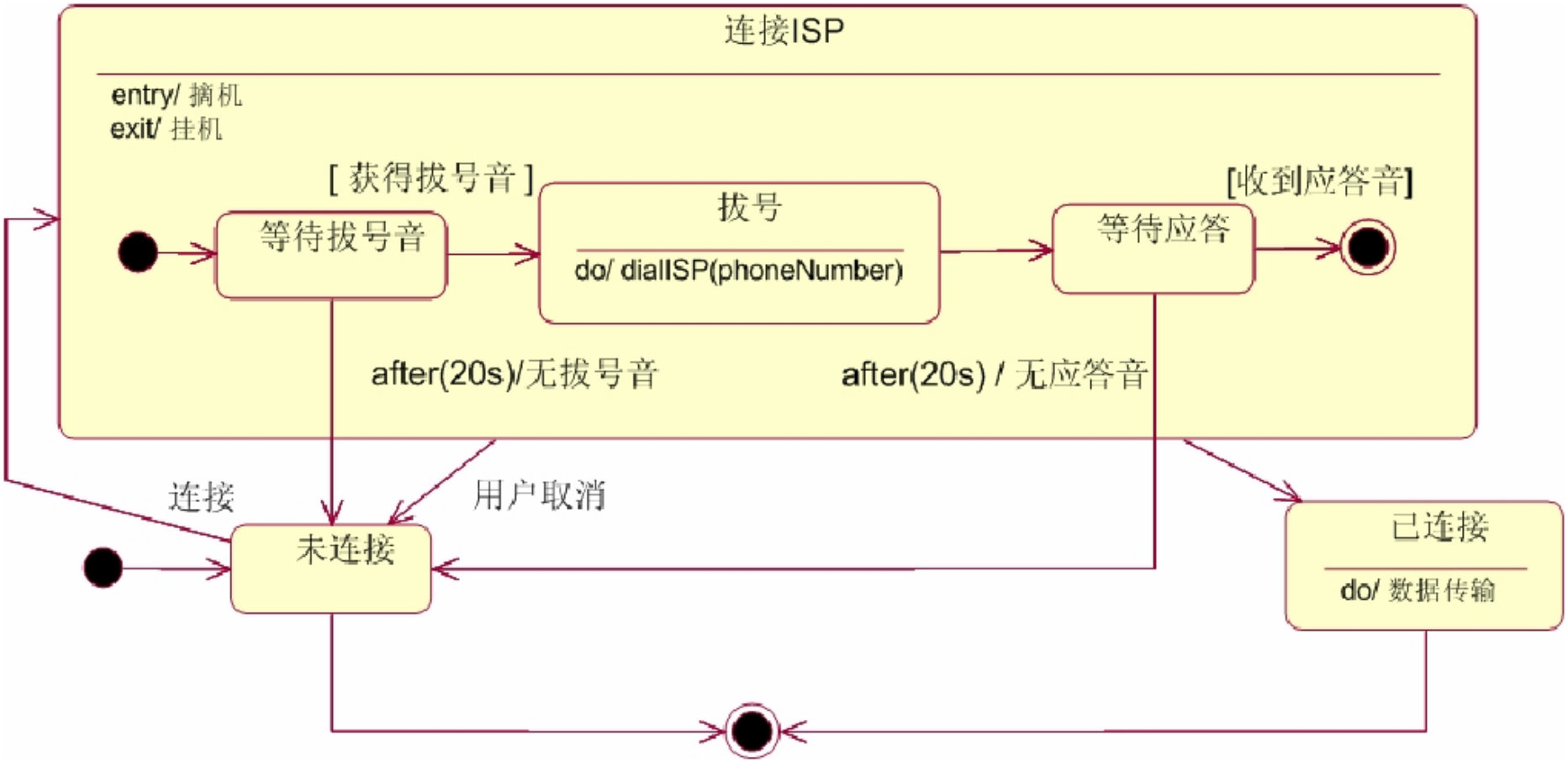
嵌套区域表示法



分解指示符法

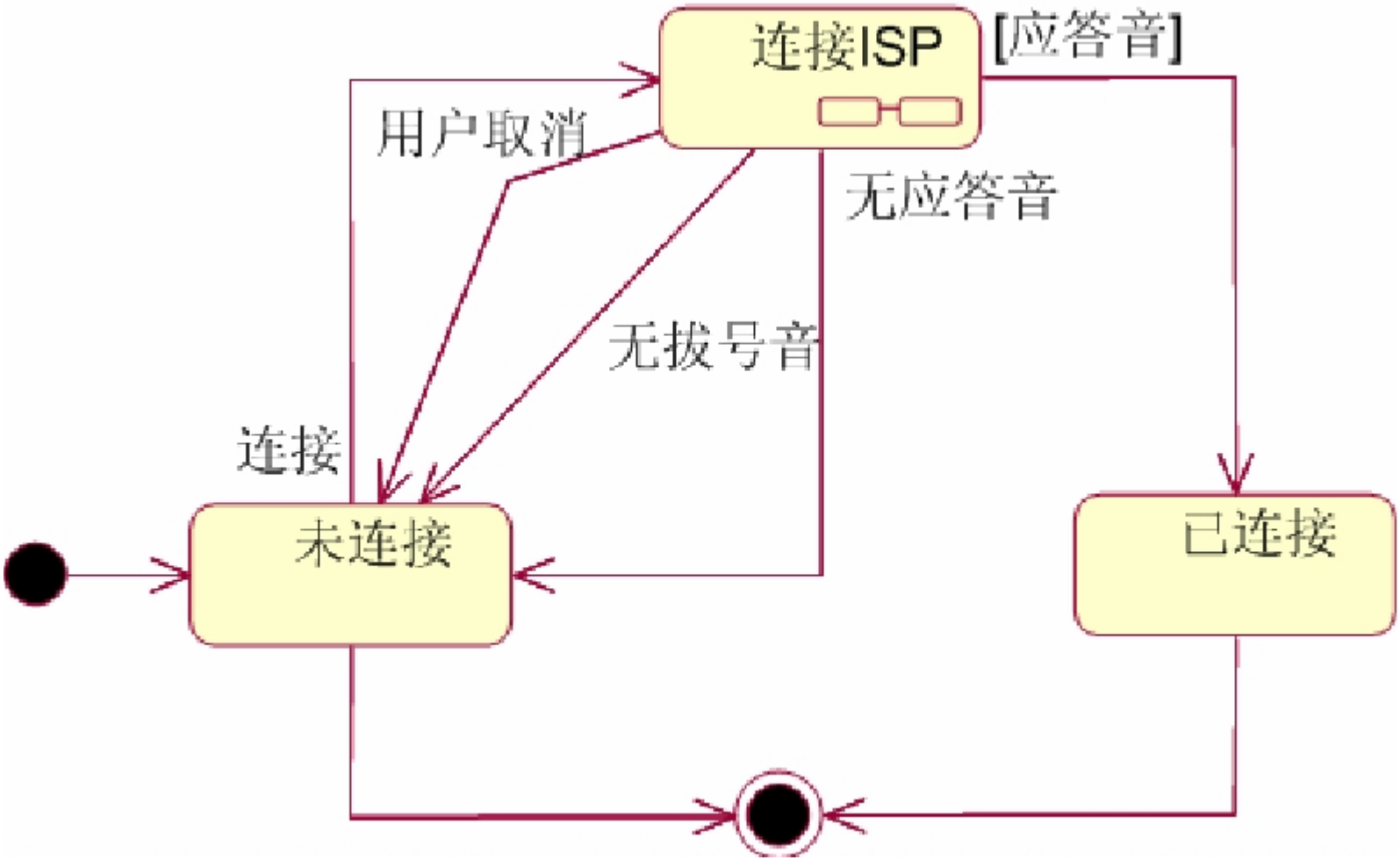
3. 复合状态

- 顺序复合状态图（表示方法1）



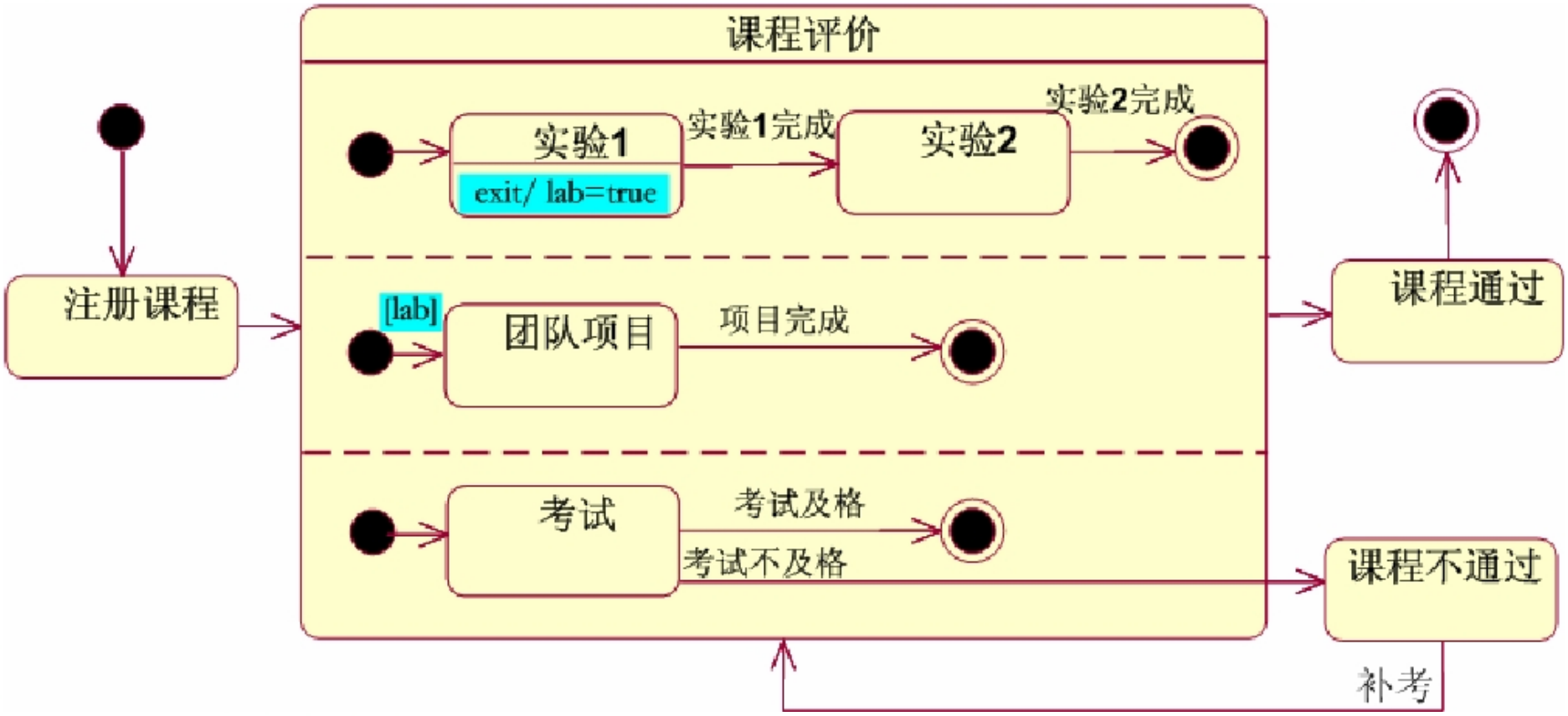
3. 复合状态

顺序复合状态图（表示方法2）



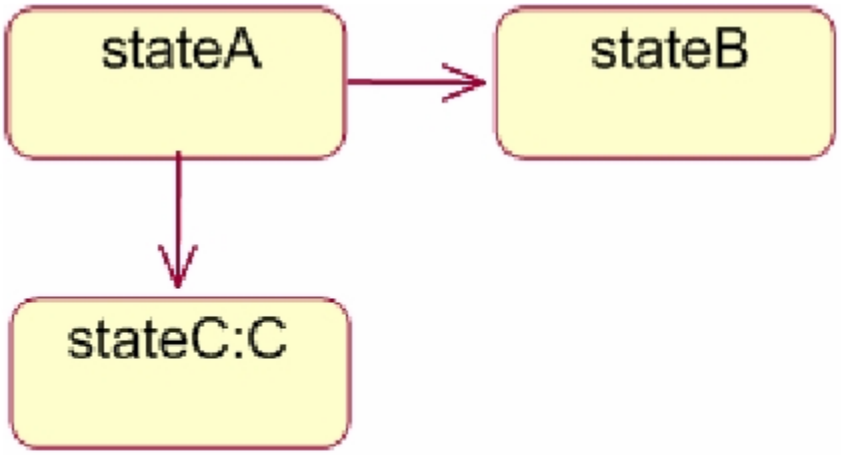
3. 复合状态

- 并发复合状态图

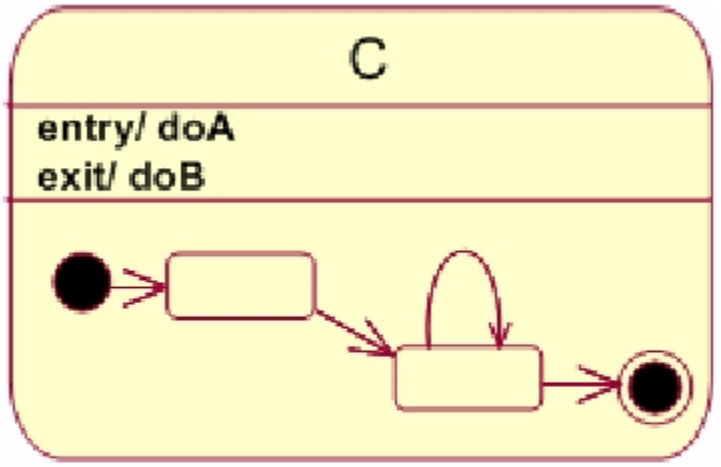


4. 子状态

- 将子状态单独定义，并对其进行命名（通常以大写字母开头），然后在需要使用的地方来引用它



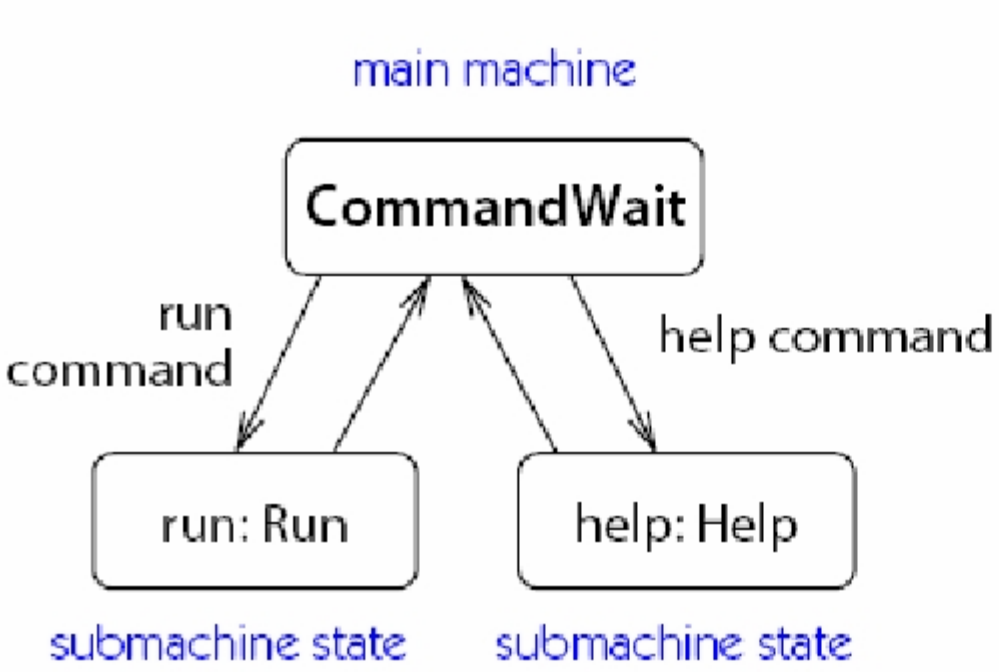
状态机图

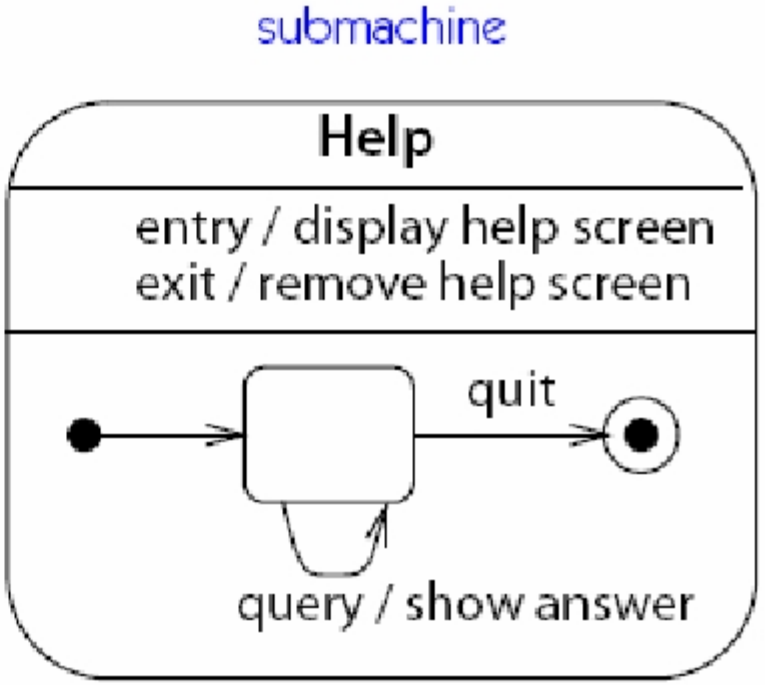


子状态机

4. 子状态

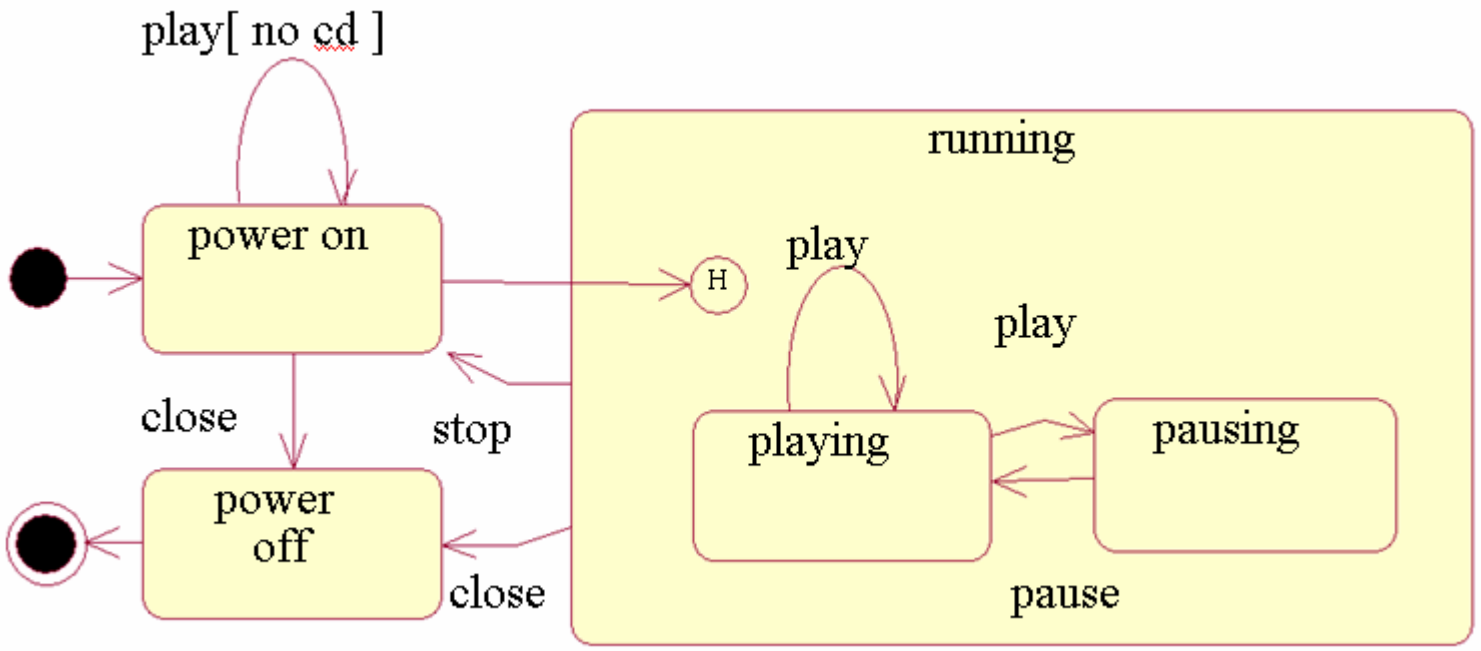
- 例：


main machine
CommandWait
run
command
help command
run: Run
help: Help
submachine state
submachine state

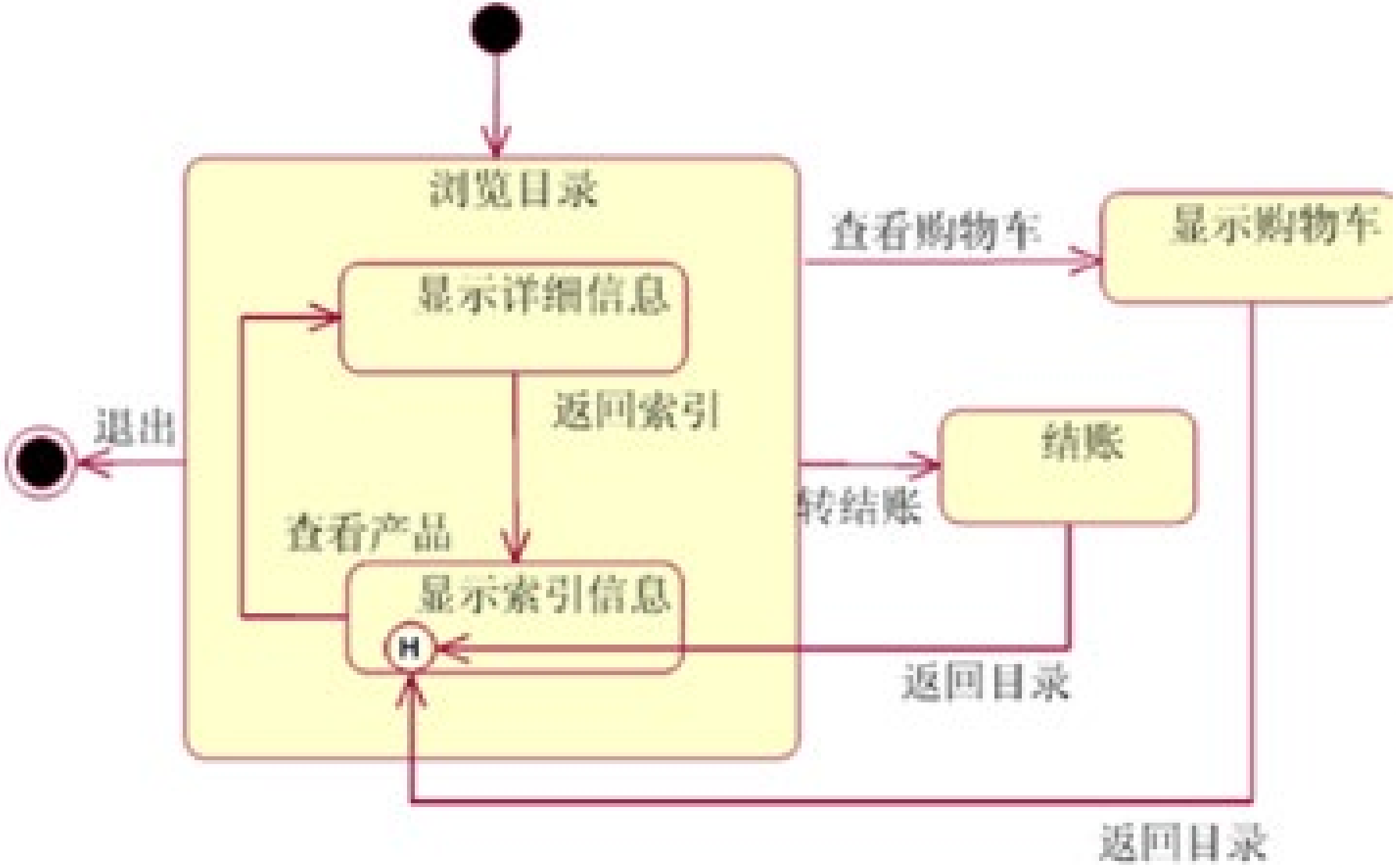

submachine
Help
entry / display help screen
exit / remove help screen
quit
query / show answer

5. 历史状态

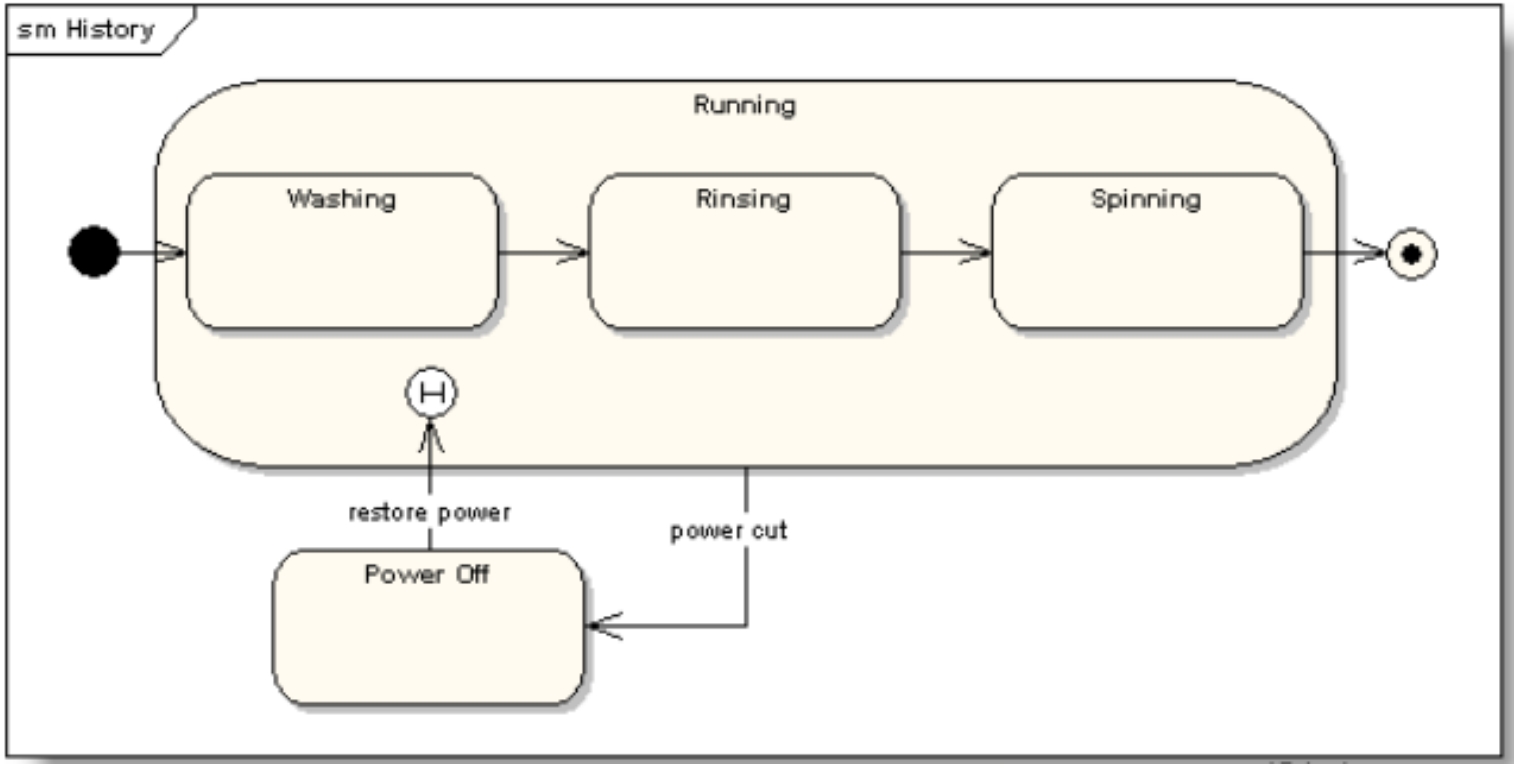
- 历史状态
  - 一种伪状态。可以存储退出组合状态时所处的子状态，则返回组合状态时可以直接回到相应的子状态。它用一个包含字母“H”的小圆圈表示。



- 例如：
  - 当从状态“结账”和“显示购物车”返回子状态“显示索引信息”时，将进入的是离开时的历史状态。
  - 也就是说，转到购物或结账区之后，再回到“浏览目录”的页面时，其中的内容是不变的，仍然保留原来的信息。

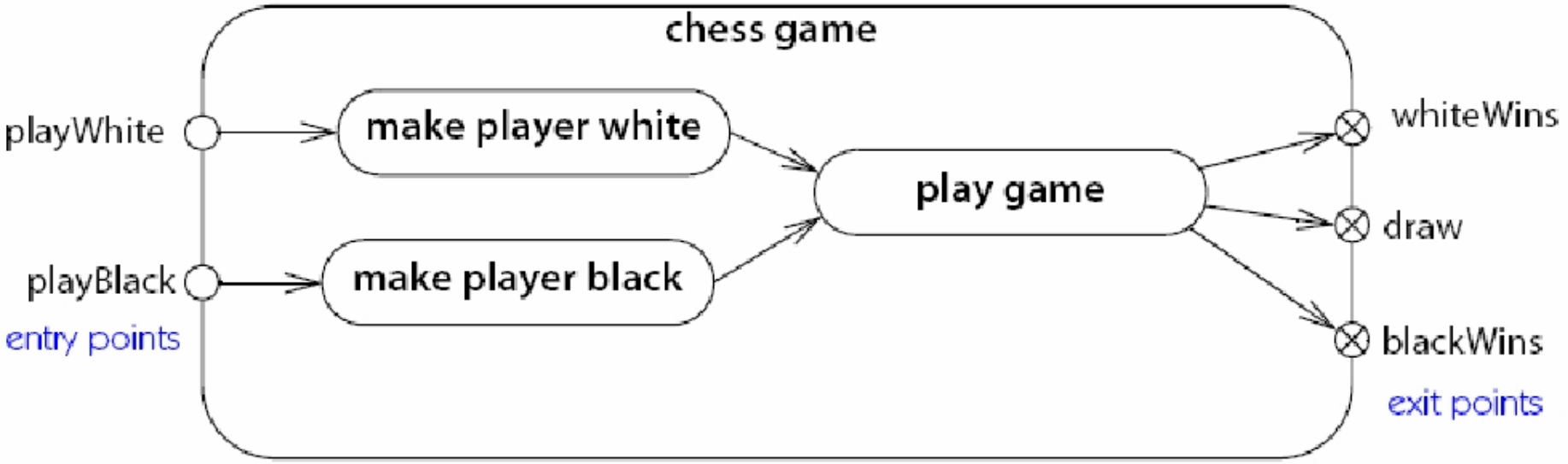

浏览目录
显示详细信息
返回索引
退出
查看产品
显示索引信息
H
查看购物车
显示购物车
结账
转结账
返回目录
返回目录

- 在下面的状态图中，正常的状态顺序是:【Washing】->【Rinsing】->【Spinning】。
- 如果是从状态【Rinsing】突然停电（Power Cut）退出,洗衣机停止工作进入状态【Power Off】，当电力恢复时直接进入状态【Running】。

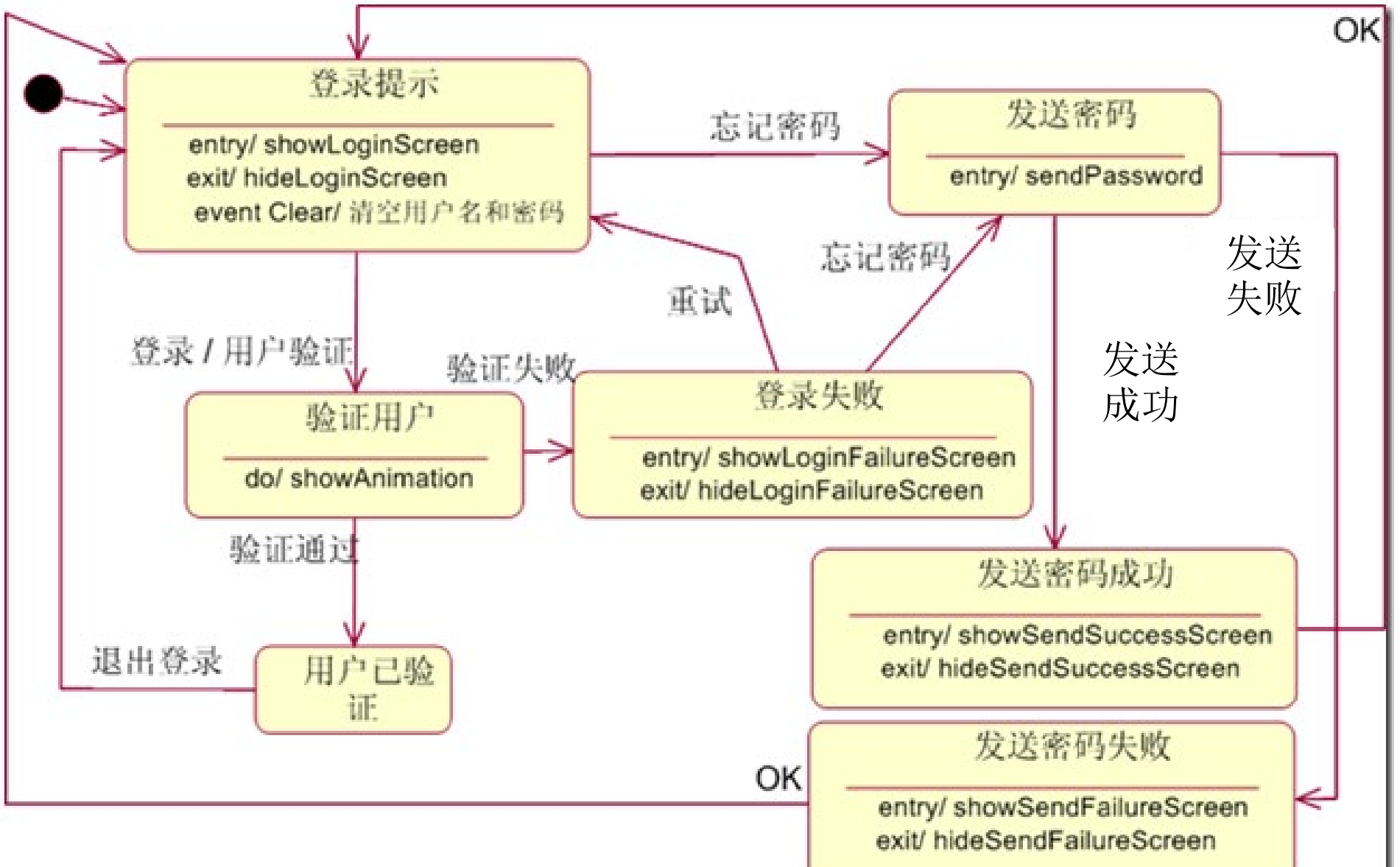


6. 入口点和出口点

- 入口点
  - 状态内的一个外部可见的伪状态，外部转换可以将它作为目标，从而被有效连接到指定状态上。
- 出口点
  - 状态内的一个外部可见的伪状态，外部转换可以将它作为源，代表状态内的一个终点。

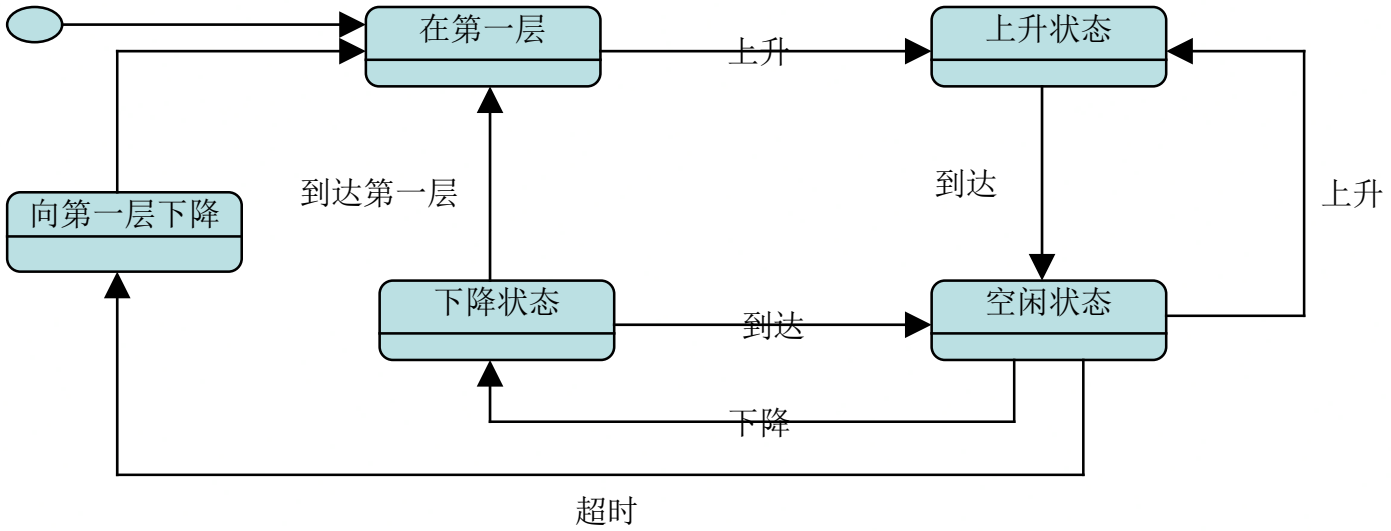


阅读具有复杂转换的状态图



练习1

- 分析下面的状态图，回答问题



(1) 以下那些图形元素是对状态的描述？

(a) 超时　　(b) 到达　　(c) 在第一层

(2) 空闲状态超时后转移到____状态

(a) 向第一层下降　　(b) 上升状态　　(c) 终态

答案

(1) (c)
(2) (a)

转换

- 转换是两个状态间的一种关系，表示当一个特定事件发生或者某些条件得到满足时，一个源状态下的对象在完成一定的动作后将发生状态转变，转向另一目标状态。
- 每个转换只允许有一个事件触发，一个事件只允许有一个动作
- 是由一种状态到另一种状态的迁移。这种转移由被建模实体内部事件或外部事件触发。对一个类来说，转移通常是调用了一个可以引起状态发生重要变化的操作的结果。

- 转换
  - 两个状态之间的一种关系，表示对象在第一个状态中执行一定的动作，并在某个特定事件发生而且满足某个条件时进入第二个状态。每个转换只允许一个事件，一个事件只允许一个动作。
  - 如果箭头上不带任何事件名，叫无触发转移（完成转移），当与源状态相关的活动完成时就会自动触发。

转换

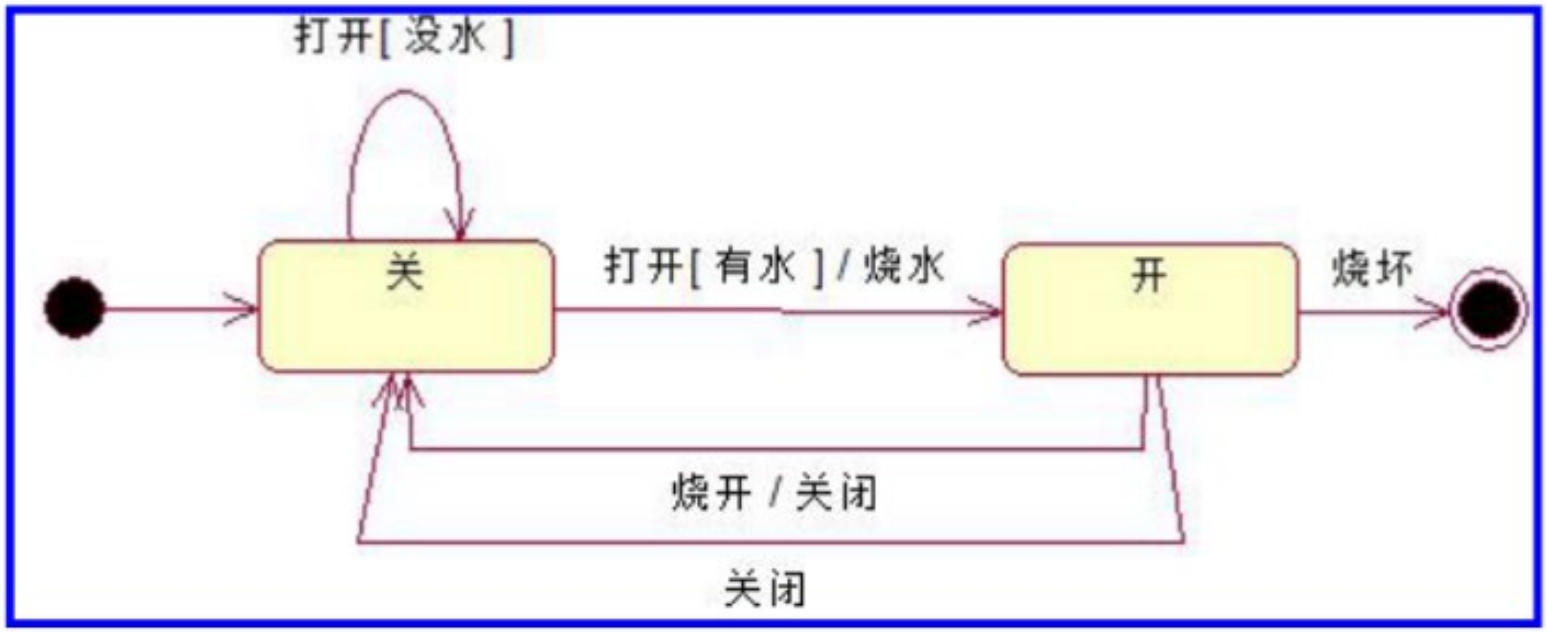
- 转换的五要素
  - 源状态
  - 目标状态
  - 触发事件
  - 监护条件
  - 动作

格式：

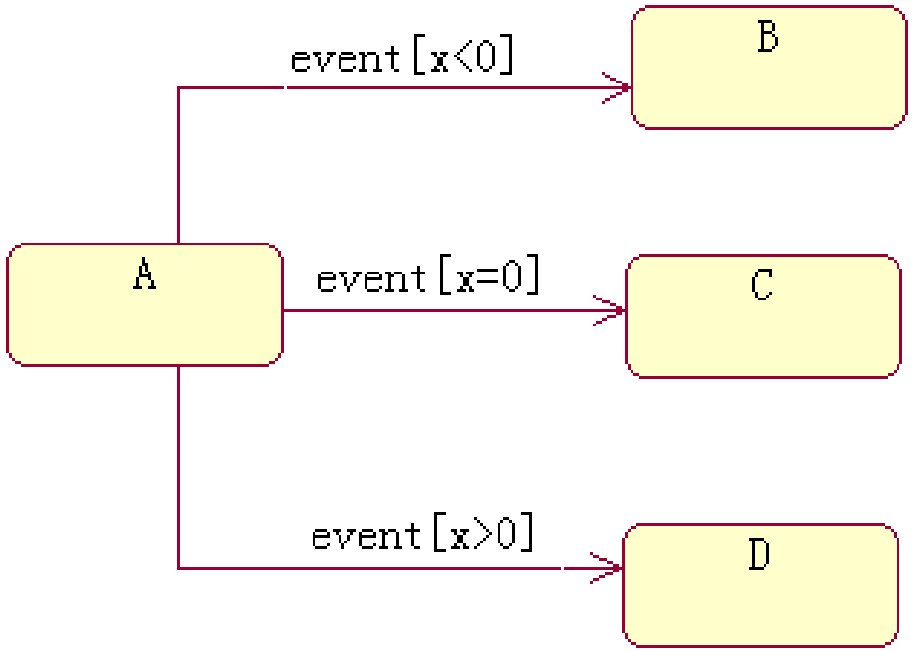
- 事件 (参数) [条件] / 动作
- **Help[len(PSW)<>6] / verifyPSW.help()**

- 监护条件：用方括号括起来的布尔表达式，它放在事件的后面。
- 如果监护条件和事件放在一起使用，则当且仅当事件发生且监护条件为真时，转移发生；如果只有监护条件，只要监护条件为真就发生转移。

- 打开[有水]/烧水：
  - 打开：触发事件
  - 有水：监护条件
  - 烧水：动作



- 从相同的状态出来的、事件相同的几个转移之间的条件应该是互斥的。

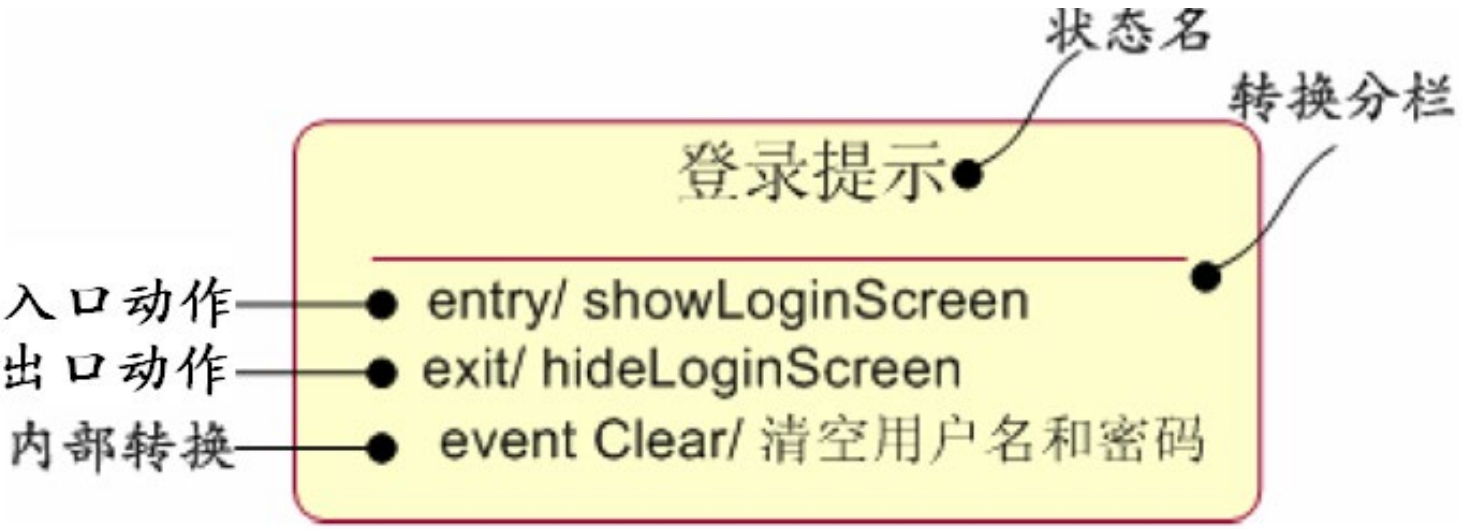


- 动作（Action)
  - 可执行的行为。
  - 不可中断，其执行时间可忽略不计
  - 当转换被引发时，它对应的动作被执行。它一般是一个简短的处理过程。
- 两种特殊动作（只要进入或离开该状态一定会执行该动作）
  - 入口动作（**entry**）
    - 进入某一状态时执行的动作
  - 出口动作（**exit**）
    - 退出某一状态时执行的动作

动作可以与状态相关，也可以与转移相关：

1. 如果动作与状态相关，则对象在进入一个状态时将触发此动作，而不管是从哪个状态转入这个状态的。

2. 如果动作与转移相关，当对象在不同的状态转移时，将触发相应的动作。



转换的种类

1 外部转换

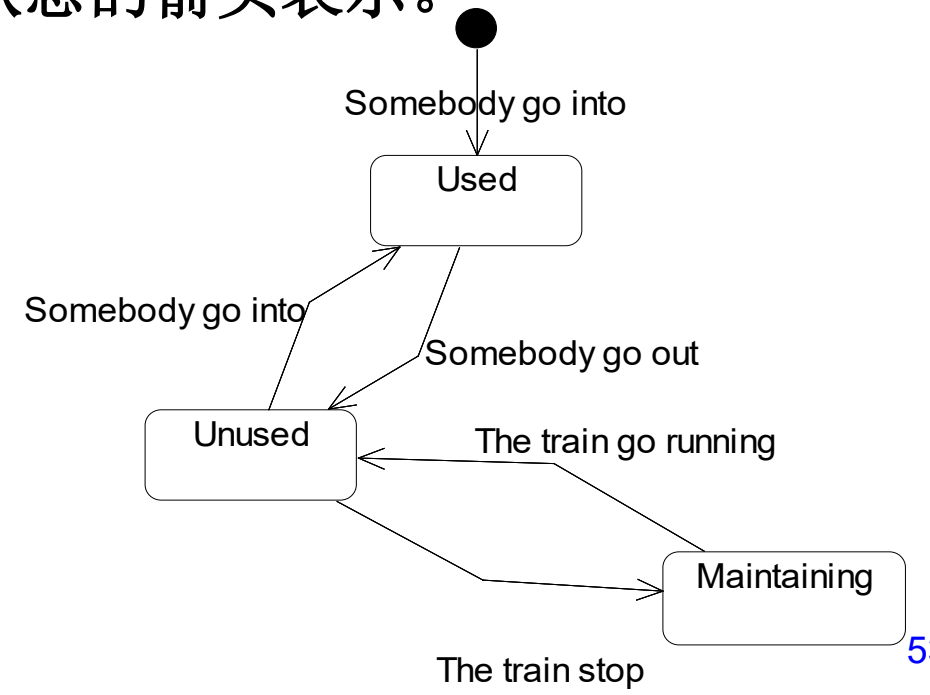
2 内部转换

3 自转换

4 复合转换

外部转换

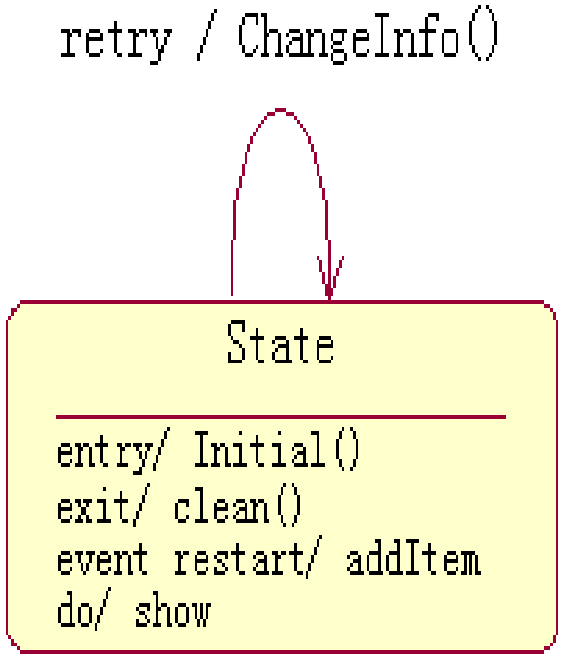
- 外部转换是一种改变对象状态的转换，是最常见的一种转换。
- 对事件做出响应，引起状态转换或自身转换，同时执行一个特定的动作，如果离开或进入状态，将引起入口动作和出口动作的执行
- 外部转换用从源状态到目标状态的箭头表示。



语法：事件(参数)[监护条件]/动作

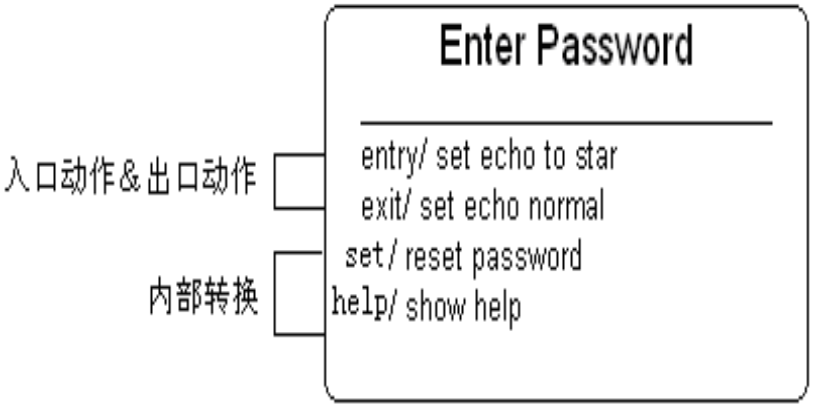
自转换

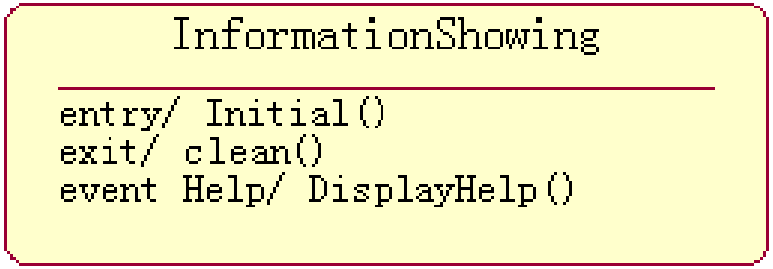
- 当事件发生时，导致状态中断，使对象退出当前状态，然后又重新回到该状态。
- 自转换在作用时首先将当前状态下的活动终止，然后执行该状态的出口动作，接着执行引起转移事件的相关动作，紧接着返回该状态，开始执行该状态的入口动作和其他操作。



内部转换

- 内部转换有一个源状态但是没有目标状态，它转换后的状态仍旧是它本身。
- 对事件做出响应，并执行一个特定的动作，但不引起状态的改变或不引起入口动作或出口动作的执行
- 内部转换的激发规则和外部转换的激发规则相同。
- 如果一个内部转换带有动作，动作也要被执行，但是由于没有状态改变发生，因此不需要执行入口和出口动作。





语法：事件(参数)[监护条件]/动作

内部转换和自转换（完成转换）不同：

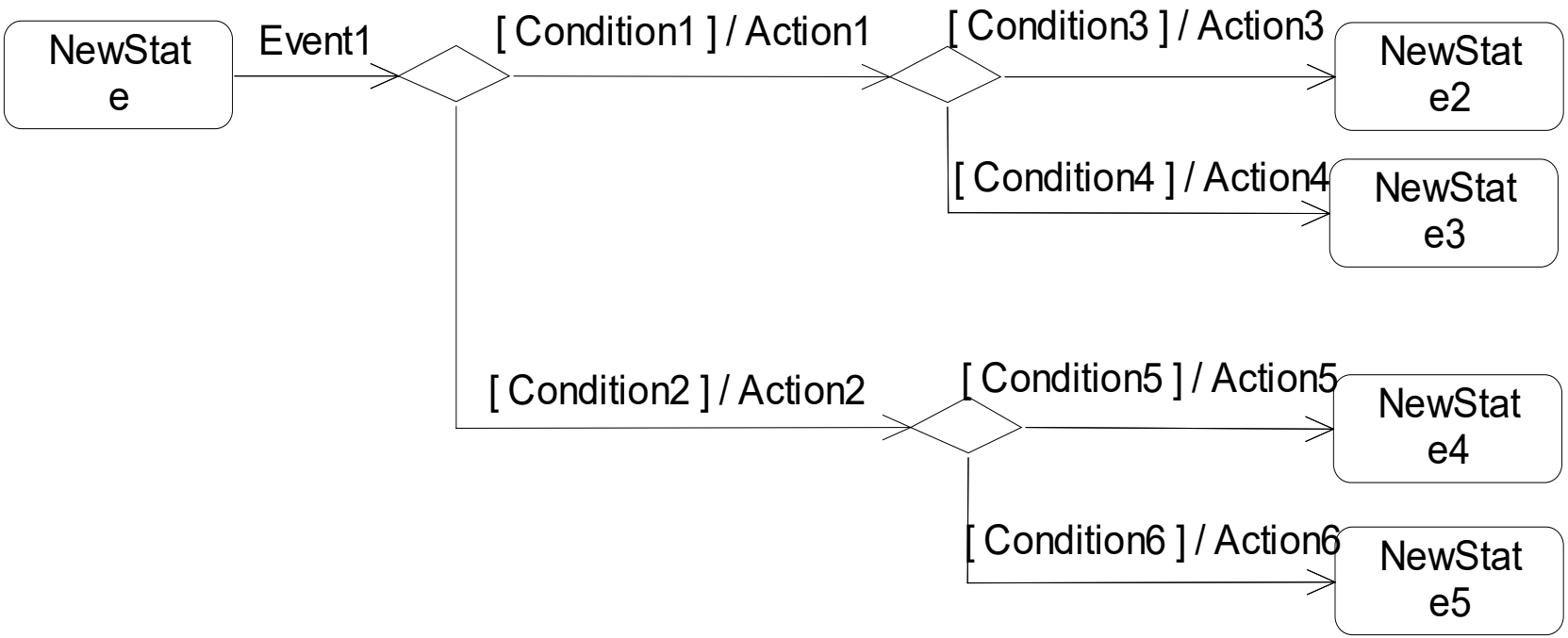
① 自转换是离开本状态后重新进入该状态，它会激发状态的入口动作和出口动作的执行。

② 内部转换自始至终都不离开本状态，所以没有出口或入口事件，也就不执行入口和出口动作。

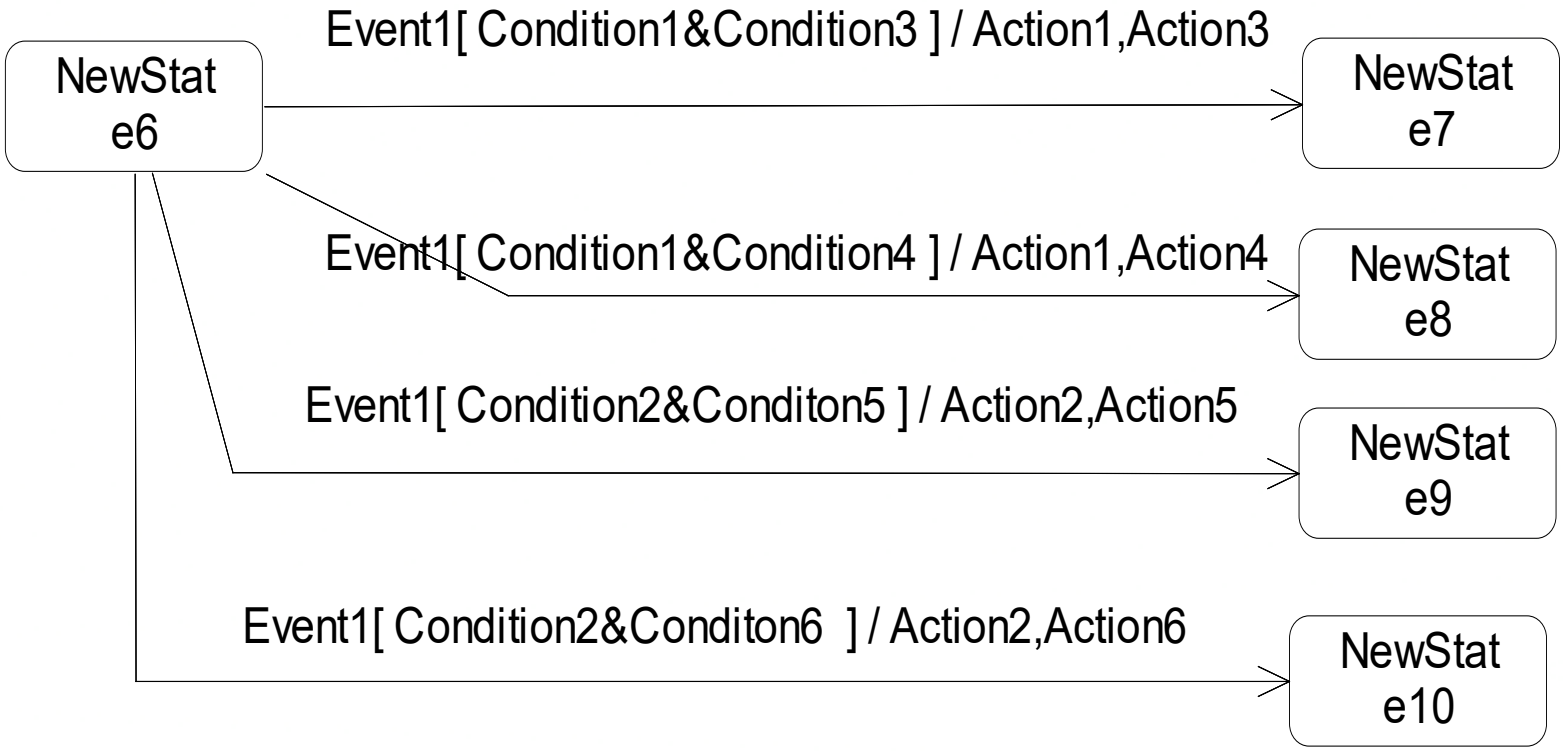
复合转换

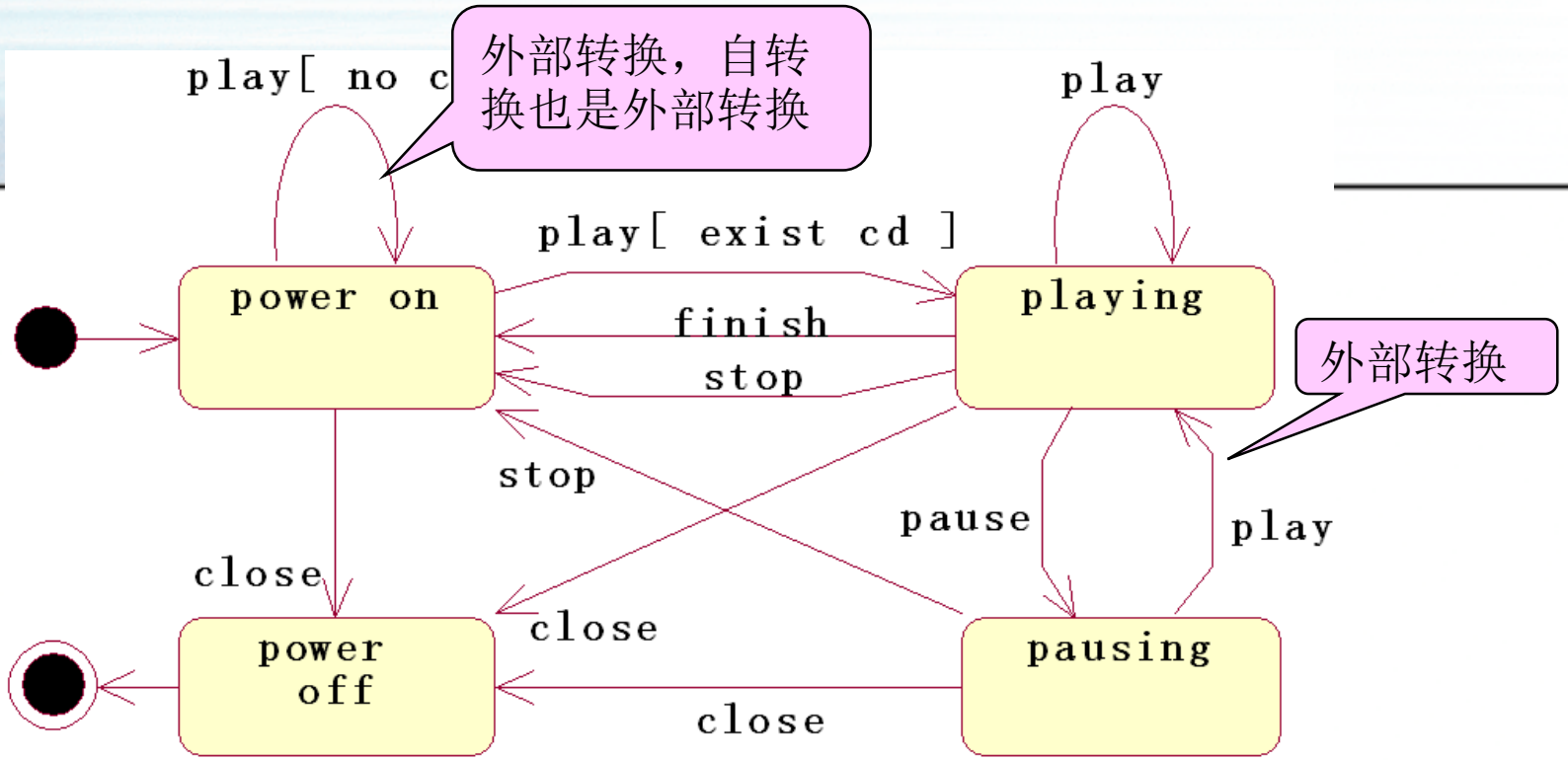
- 复合转换由简单转换组成，这些简单转换通过分支判定、分叉或汇合组合在一起。
- 除了两个分支的判定，还有多条件的分支判定。
- 多条件的分支判定又分为链式的和非链式的分支。

- 链式分支


NewStat
e
Event1
[Condition1] / Action1
[Condition3] / Action3
NewStat
e2
[Condition4] / Action4
NewStat
e3
[Condition2] / Action2
[Condition5] / Action5
NewStat
e4
[Condition6] / Action6
NewStat
e5

- 非链式分支：


NewStat
e6
Event1[Condition1&Condition3] / Action1,Action3
NewStat
e7
Event1[Condition1&Condition4] / Action1,Action4
NewStat
e8
Event1[Condition2&Conditon5] / Action2,Action5
NewStat
e9
Event1[Condition2&Conditon6] / Action2,Action6
NewStat
e10



名字域

Lighting

可选

当转入该状态时，做

当处于该状态时，灯

/ turnOn

linkFivetimes

t poweroff/ powerSupplySelf

/ turnOff

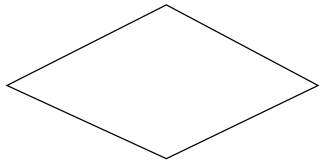
event selfTest/ defer

do活动是只在状态内出现的活动，不能附加到转换上。

当selfTest事件发生时，对象将延迟响应，到别的状态中再处理，用defer这个特定动作表示延迟

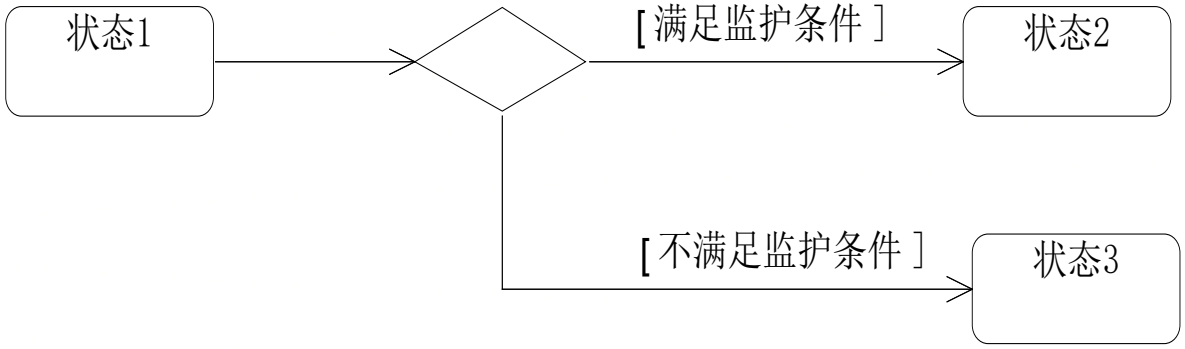
判定（决策点）

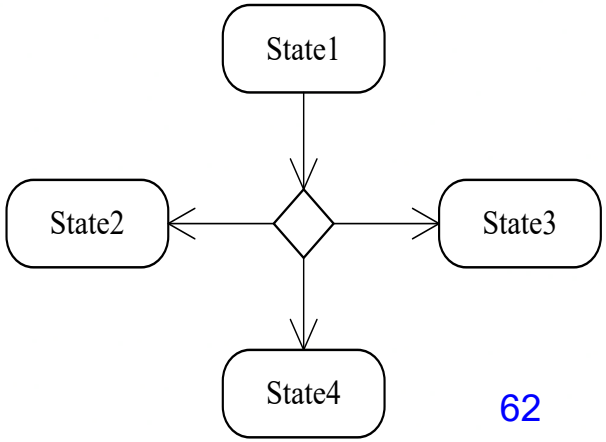
- 判定在状态图中的位置：工作流在此处按监护条件的取值而发生分支。
- 判定用空心小菱形表示。



判定

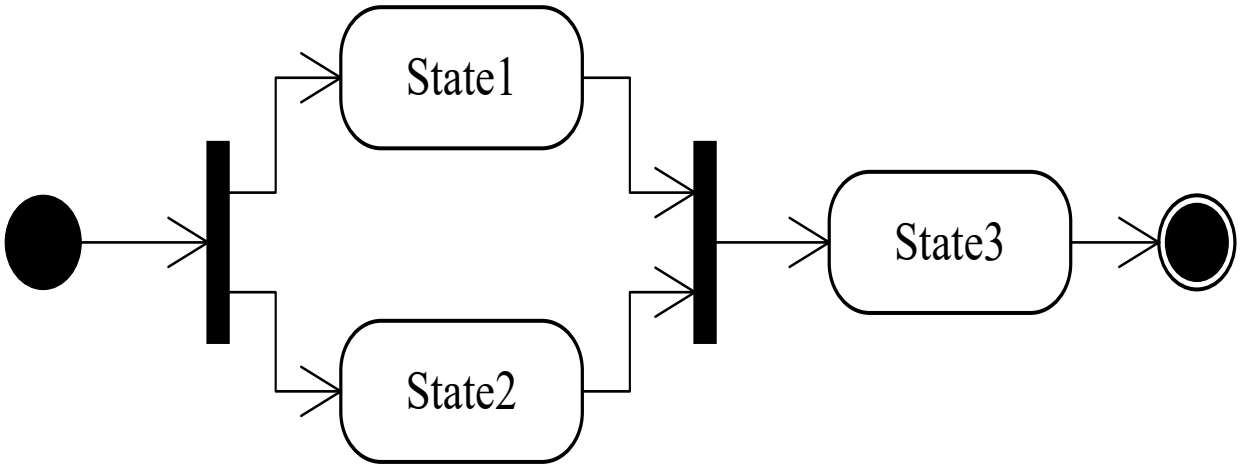
- 它在建模状态机图时提供了方便，因为它通过在中心位置分组转移到各自的方向，从而提高了状态机图的可视性。
- 因为监护条件为布尔表达式，所以通常条件下的判定只有一个入转换和两个出转换。
- 根据监护条件的真假可以触发不同的分支转换。





同步状态

- 使用同步条可以显示并发转移。
- 并发转移表示一个同步将一个控制划分为并发的线程。状态图中使用到同步条是为了说明某些状态在哪里需要跟上或者等待其他状态。状态图中同步条是一条黑色的粗线，图显示了使用了同步条的状态机图。

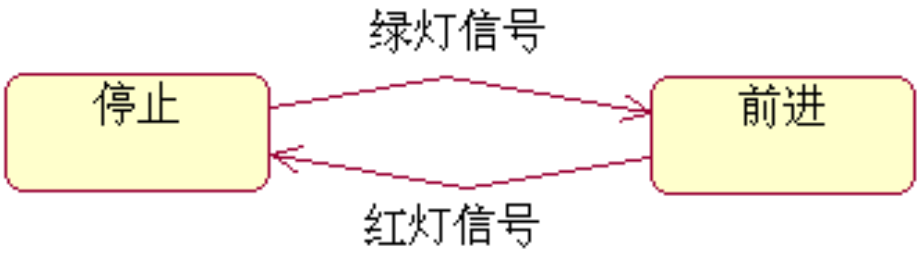


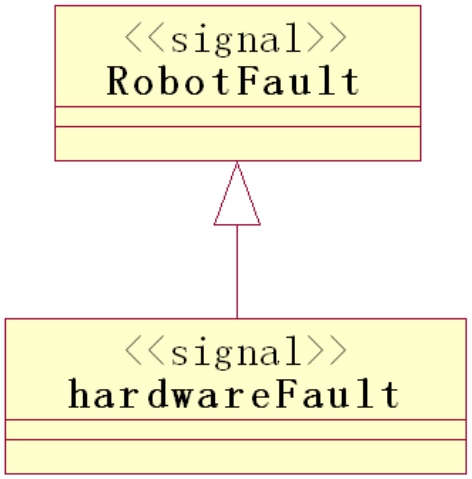
事件

- **Event**
  - 是对一个时间和空间上占有一定位置的有意义的事情的规格说明。
  - 事件触发状态的转移
- 四类主要事件
  - 信号事件
  - 调用事件
  - 变化事件
  - 时间事件

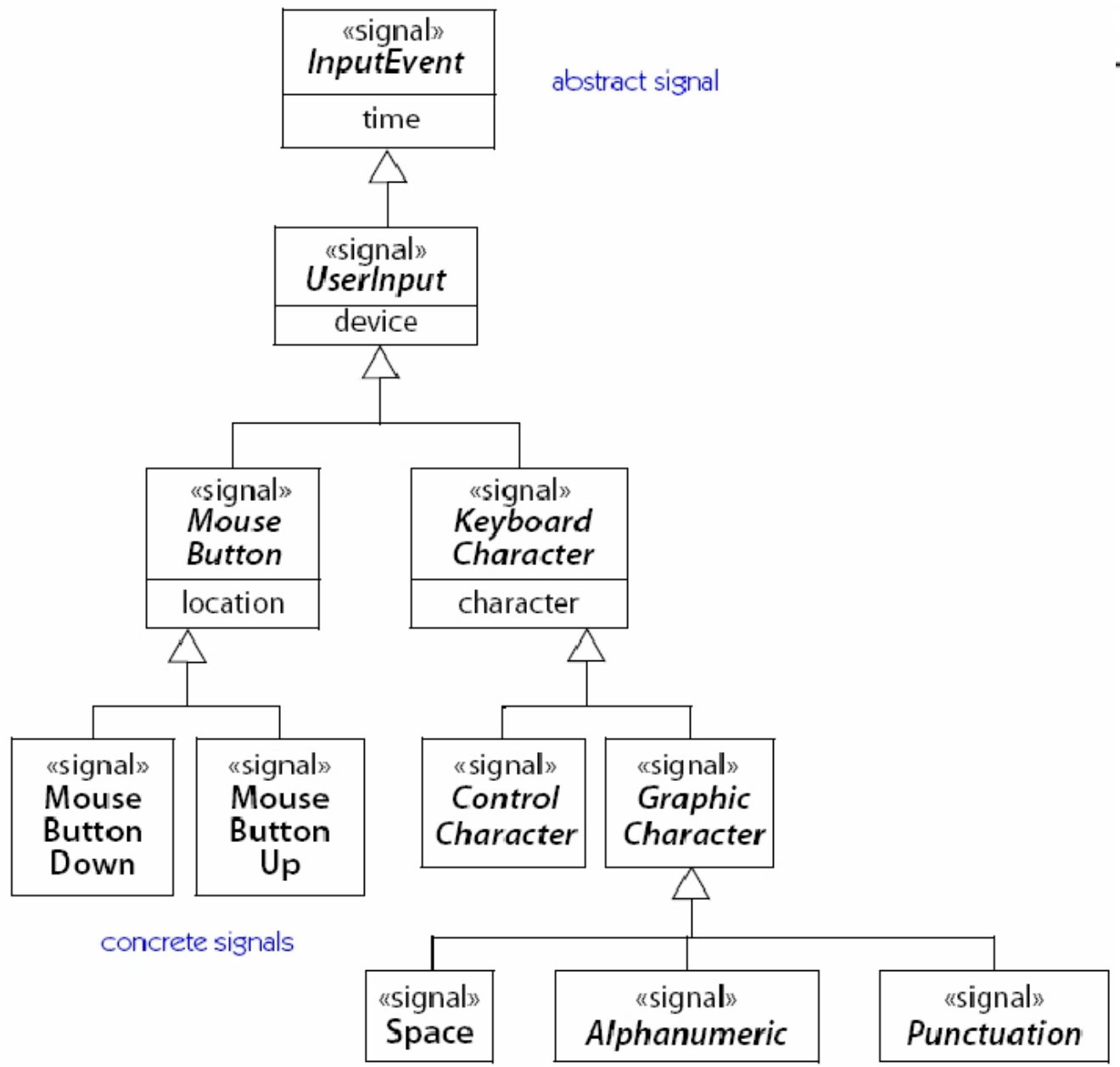
1.信号事件

- 信号（signer）事件
  - 所谓信号，是指由一个对象异步地发送、并由另外一个对象接收的一个已命名的对象。
  - 信号事件表示对象接收到某个信号。
  - 信号可以作为状态中一个状态转换的动作而被发送。
  - 信号间可以有泛化，信号可以是其他信号的子信号，它们继承父信号的属性，并可以触发包含信号类型的转换。



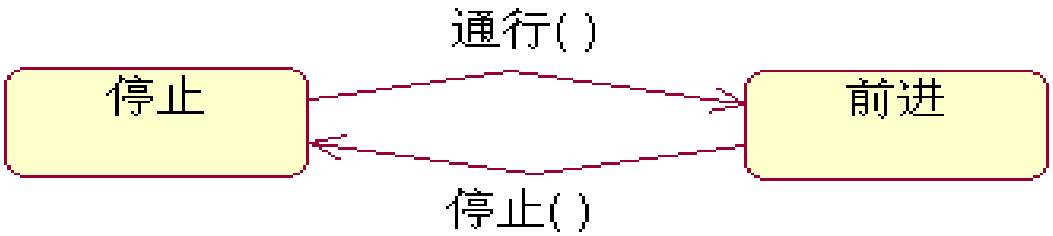


信号事件


«signal»
InputEvent
time
abstract signal
«signal»
UserInput
device
«signal»
Mouse
Button
location
«signal»
Keyboard
Character
character
«signal»
Mouse
Button
Down
«signal»
Mouse
Button
Up
«signal»
Control
Character
«signal»
Graphic
Character
concrete signals
«signal»
Space
«signal»
Alphanumeric
«signal»
Punctuation

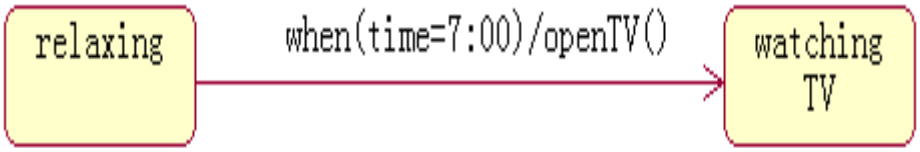
2. 调用call事件

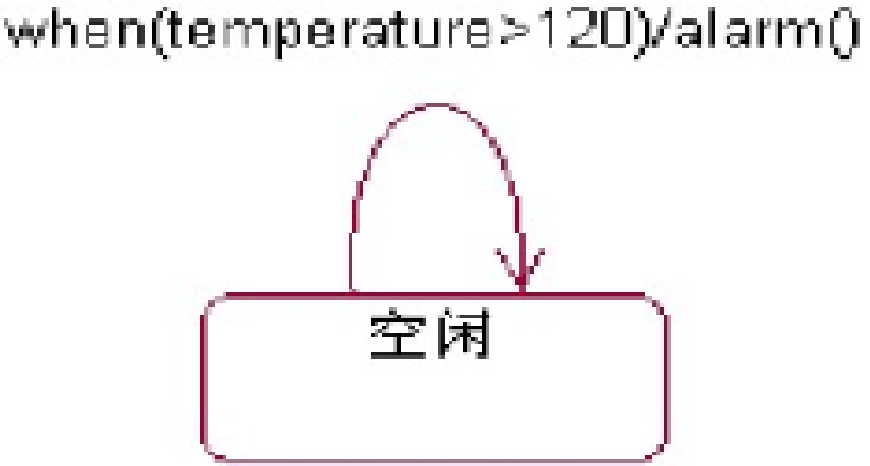
- 表示一个操作的调度。请求调用另一个对象的操作
- 当对象调用另一对象的操作时，控制就从发送者传送到接收者，该事件触发转换，完成操作后，接收者转换到一个新的状态，控制返还给发送者。
- 信号是一个异步事件，而调用事件一般是同步的。



3. 变化change事件

- 如果一个布尔表达式中的变量发生变化，使得该表达式的值相应变化，从而满足条件，则这种事件叫变化事件。
- 用关键字**When**，后面跟布尔表达式
- **When(temperature>120) /alerm()**
- 变化事件的意图是要频繁测试表达式，只要表达式由假变为真，事件就会发生。



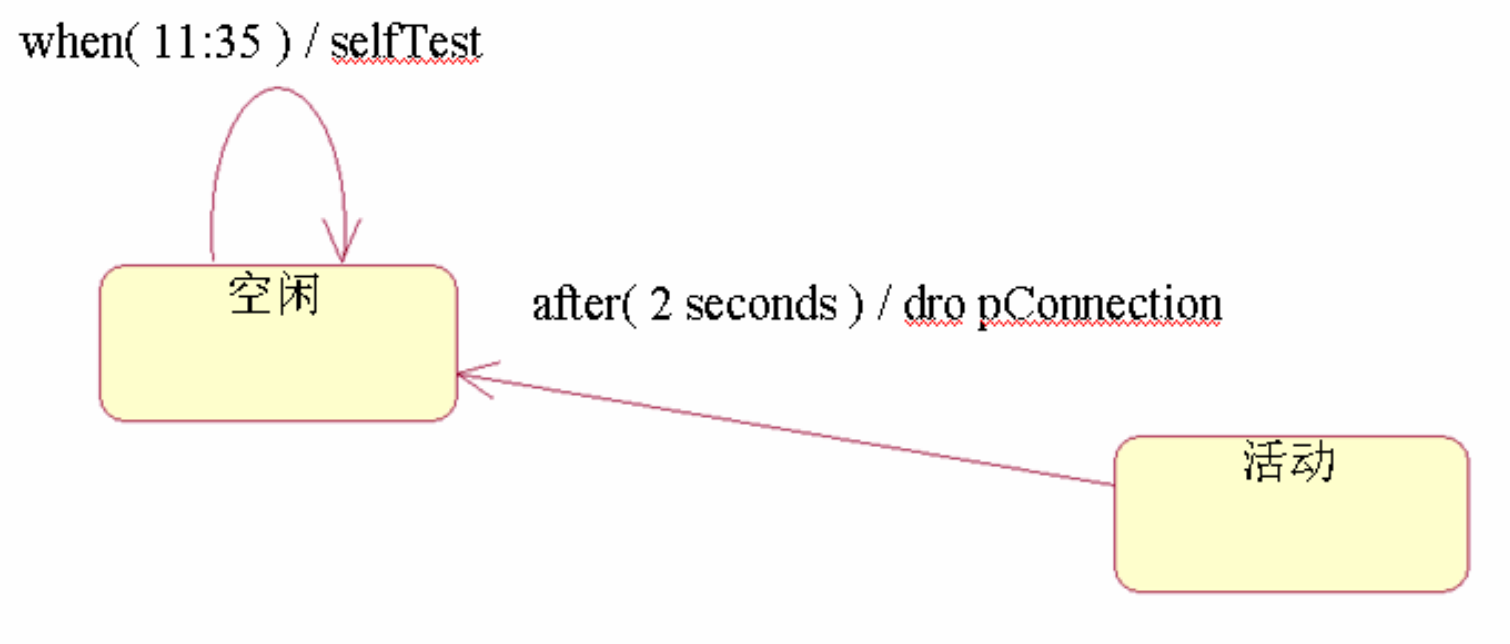


注意：变化事件与监护条件的不同。

- 监护条件是转移说明的一部分，只有所相关的事件出现后计算一次这个条件，如果值是**false**，则不进行转移，以后也不再重新计算这个监护条件，除非事件又重新出现。
- 变化事件表示的是一个要被不断测试的事件。

4.时间事件

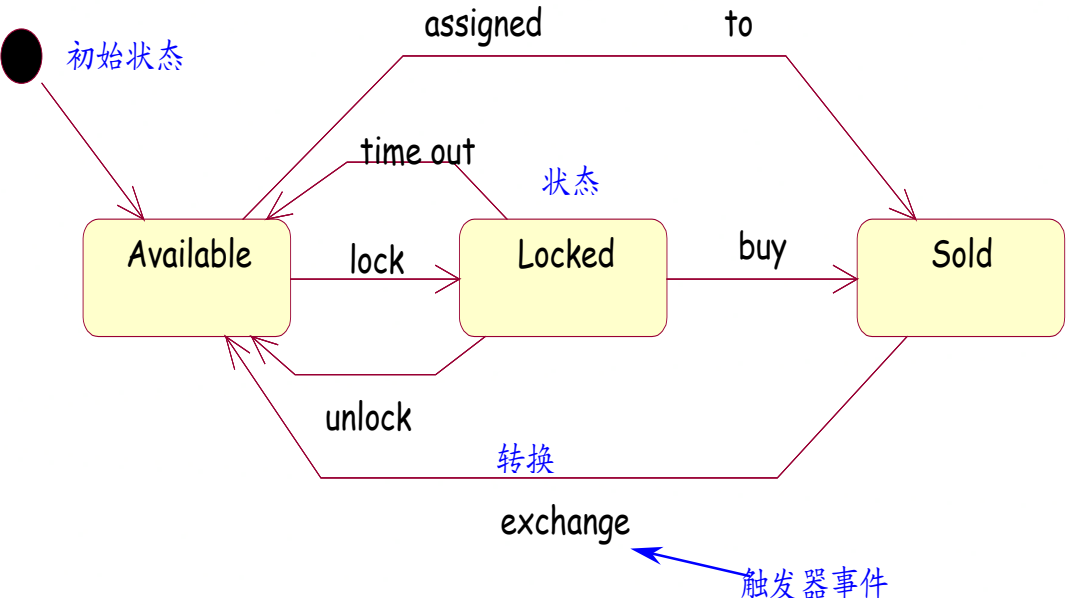
- 时间（time）事件
  - 满足某一时间表达式的情况的出现，例如到达某一时间
  - 或经过了某一时间段。用关键字After或When表示。



状态图的例子-1

(1) 状态图中包含以下状态

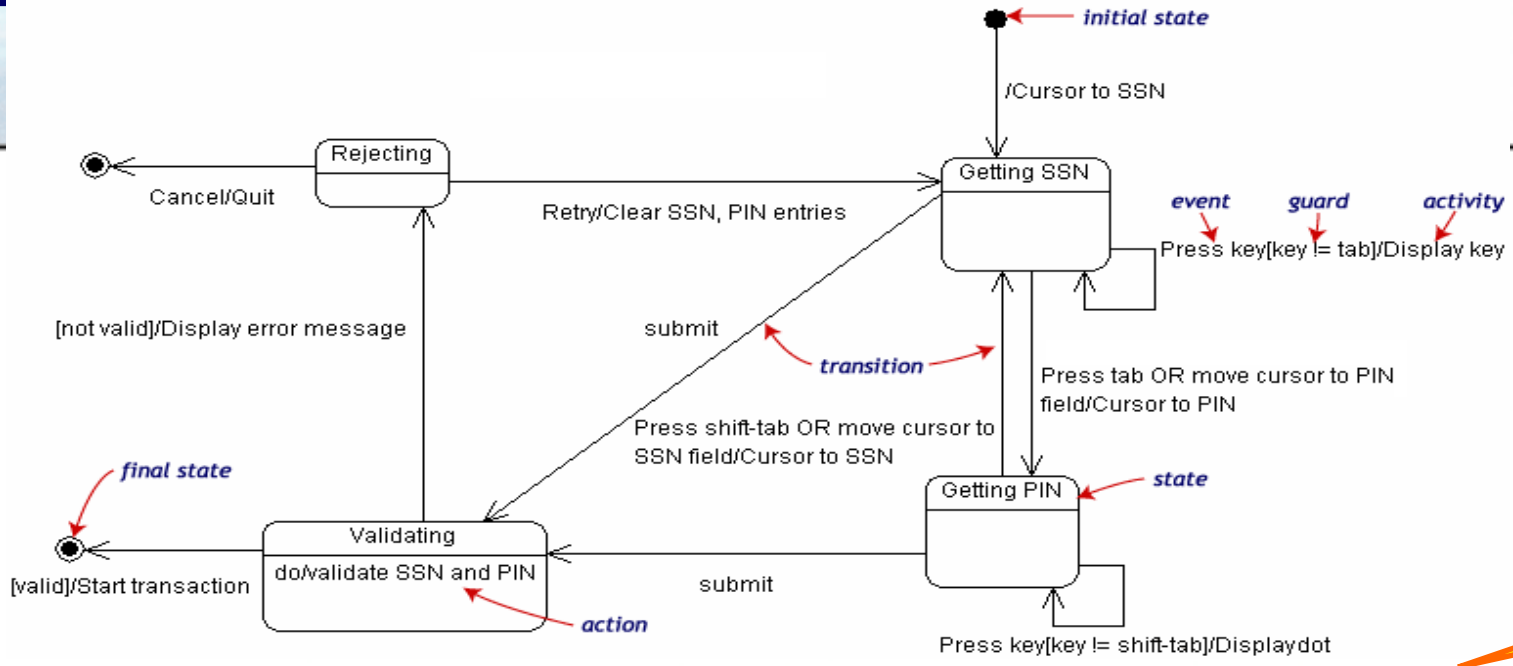
- 初始状态
- **Available状态**
- **Locked状态**
- **Sold状态**



(2) 状态间的转移

- **初始状态→Available状态**
- **票被预订(lock)：Available→Locked**
- **预定后付款(buy)：Locked→Sold**
- **预定解除(unlock)：Locked→Available**
- **预定过期(time out)：Locked→Available**
- **直接购买(assigned to)：Available→Sold**
- **换其它票(exchang)，该票重新有效：Sold→Available**

(2)网上银行登陆系统



状态转移的过程

登陆要求提交个人社会保险号(SSN)和密码(PIN)经验证有效后登陆成功。

登陆过程包括以下状态:

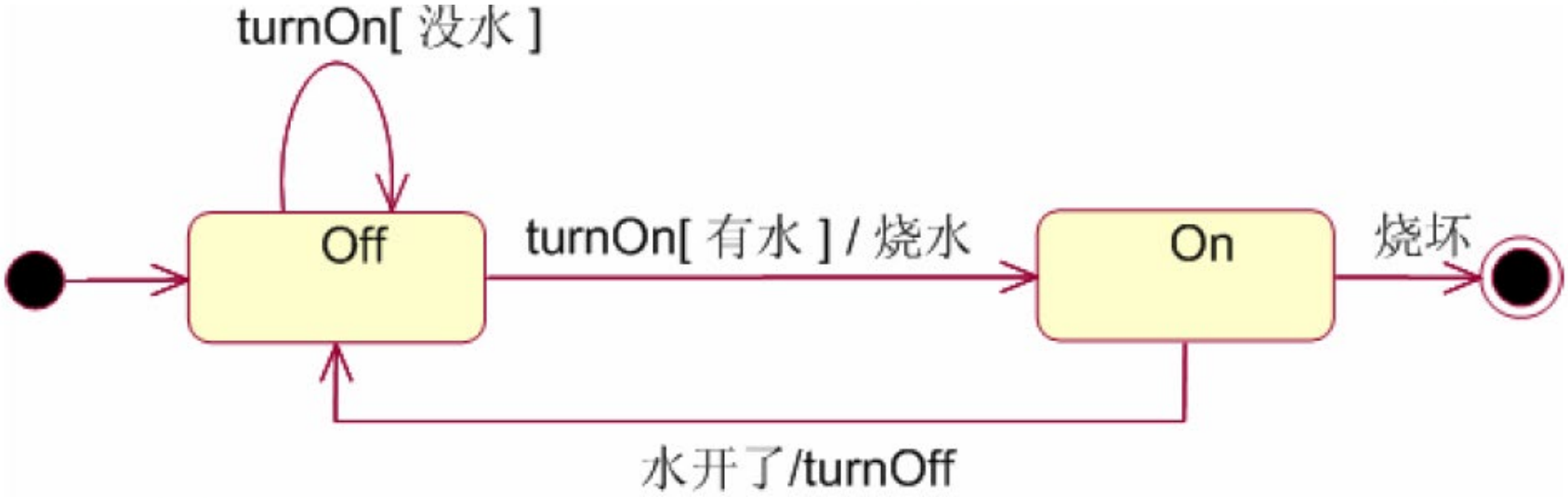
※ *初态(Initial state)*
※ *获取社会保险号状态(Getting SSN)*
※ *获取密码状态(Getting PIN)*
※ *验证状态(Validating)*
※ *拒绝状态(Rejecting)*
※ *终态(Final state)*

出发状态	动作	到达状态
Initial state	移动鼠标到 **SSN**	Getting SSN
Getting SSN	键入非**tab**键，显示键入内容	Getting SSN
	键入**tab**键，或移动鼠标到**BIN**	Getting PIN
	提交	Validating
Getting PIN	键入非**shift-tab**键，显示 “ * ”	Getting PIN
	键入**shift-tab**键，或移动鼠标到**SSN**	Getting SSN
	提交	Validating
Validating	验证提交信息有效，状态转移	Final state
	验证提交信息无效，显示错误信息	Rejecting
Rejecting	退出	Final state
	重试，清除无效的**SSN**，**PIN**	Getting SSN

有两个不同的终态

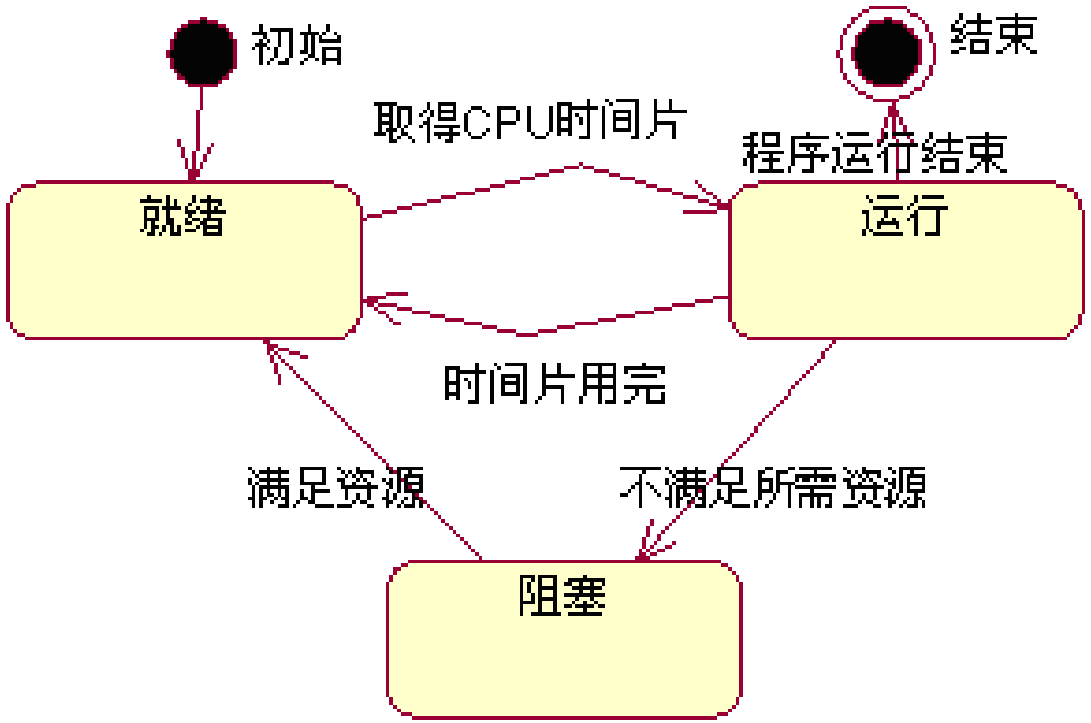
练习2

- 电水壶：on和off两个状态，初态off。trunOn事件发生时，判断水壶是否有水，若没有水，则仍处于off状态，若有水，则turnOn事件引起烧水活动，使状态从off转入on，水开，则从on转入off状态，烧坏则转换到终态。



练习3

- 进程
  - 就绪 运行 阻塞三个状态
  - 初态就绪; 程序运行结束后终态
  - 就绪状态获得CPU时间片转为运行态；运行态时间片用完转为就绪态；运行态不满足所需资源转为阻塞态，阻塞态若资源满足则回到就绪态



状态图的作用

- 用途
  - 对对象生命周期建模：
    - 主要描述对象能够响应的事件、对这些事件的影响以及过去对当前行为的影响
  - 对反应型对象建模：
    - 这个对象可能处于的稳定状态、从一个状态到另一个状态之间的转换所需的触发事件，以及每个状态改变时发生的动作
  - 状态图既可以用来表示一个业务领域的知识，也可以用来描述设计阶段对象的状态变迁

状态图的作用

- 业务阶段：通常使用状态图来说明业务角色或业务实体可能的状态——导致状态转换的事件和状态转换引起的操作
- 建模阶段：状态机对模型元素的动态行为进行建模，就是对系统行为中受事件驱动的方面进行建模
- 实体类描述阶段：状态机用于描述实体类对象的整个生命周期内的状态变迁以获得对这个实体的理解，同时获得系统和实体对象相互影响的关系

状态图的作用

- 设计实现：对于类对象所有可能的状态，状态图都显示它可能接收的消息、将执行的操作和在此之后类的对象所处的状态
- 注：
  - 状态机主要用于描述对象的状态变化以确定何种行为改变了对象状态，以及对象状态变化对系统的影响
  - 状态图通常只用于描述单个对象的行为，如果要描述对象间的交互，最好采用时序图或协作图

状态图建模步骤

- 寻找主要的状态
- 确定状态之间的转换
- 细化状态内的活动与转换
- 用组合/嵌套状态来展开细节

详细步骤

1)）确定状态机的上下文，它可以是一个类、子系统或整个系统。

2）选择初始状态和终结状态。

3）发现对象的各种状态。状态属性，行为（确定事件和动作）

4）确定状态可能发生的转移。注意分出从一个状态可能转移到哪些状态，对象的哪些行为可引起状态的转移并找出触发状态转移的事件。

5）把必要的动作加到状态或转移上。

6）复合状态、子状态、历史状态等概念组织和简化一个复杂的状态机。

7）分析状态的并发和同步情况。

8）绘制状态图。

9）确认每一个状态在某个时间组合之下都是可到达的。确认没有一个死状态，对象不能从该状态转移出来。

学生学习过程-案例分析1

- 状态分析：
  - 学前
  - 在校
  - 退学
  - 毕业
  - 休学

学生学习过程-案例分析1

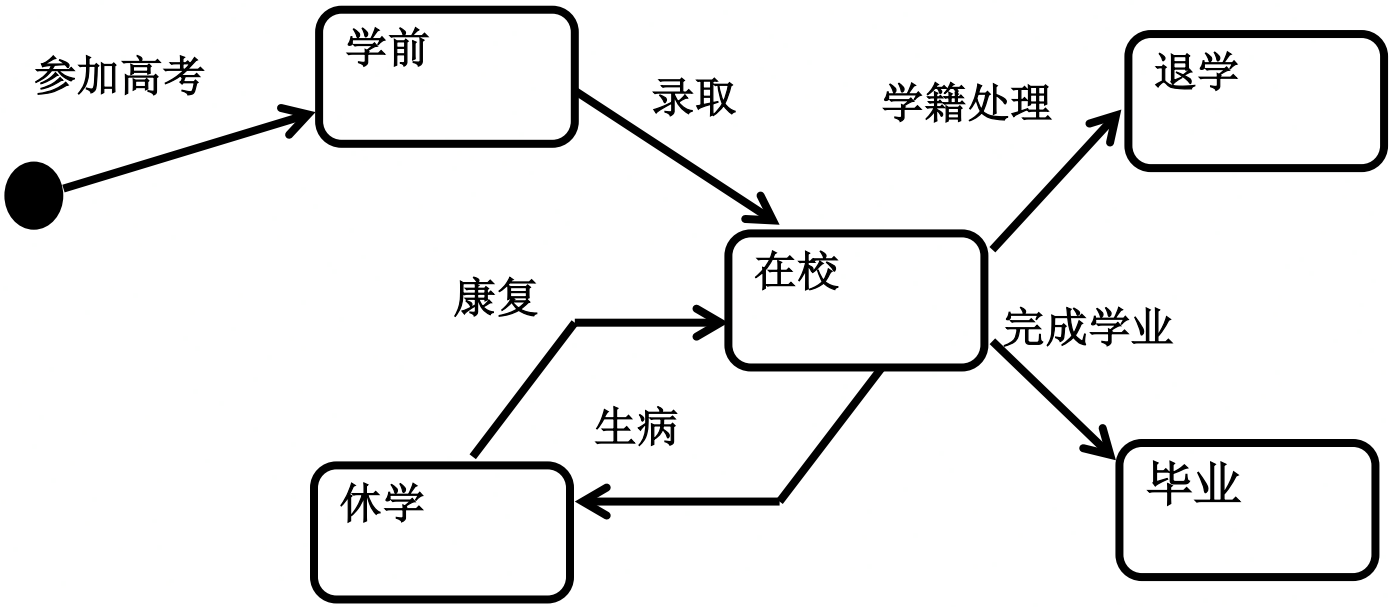
- 首先分析事件：
  - 参加高考
  - 录取
  - 生病
  - 康复
  - 完成学业
  - 学籍处理

学生学习过程-案例分析1

- 接下来分析转换事件：
  - 参加高考：高考后进入“学前”状态
  - 录取：学生被录取后变成“在校”的学生
  - 生病：学生长时间生病则进入“休学”状态
  - 康复：学生再回到学校转为“在校”状态
  - 完成学业：学生毕业进入“毕业”状态
  - 学籍处理：学生被学籍处理，进入“退学”状态

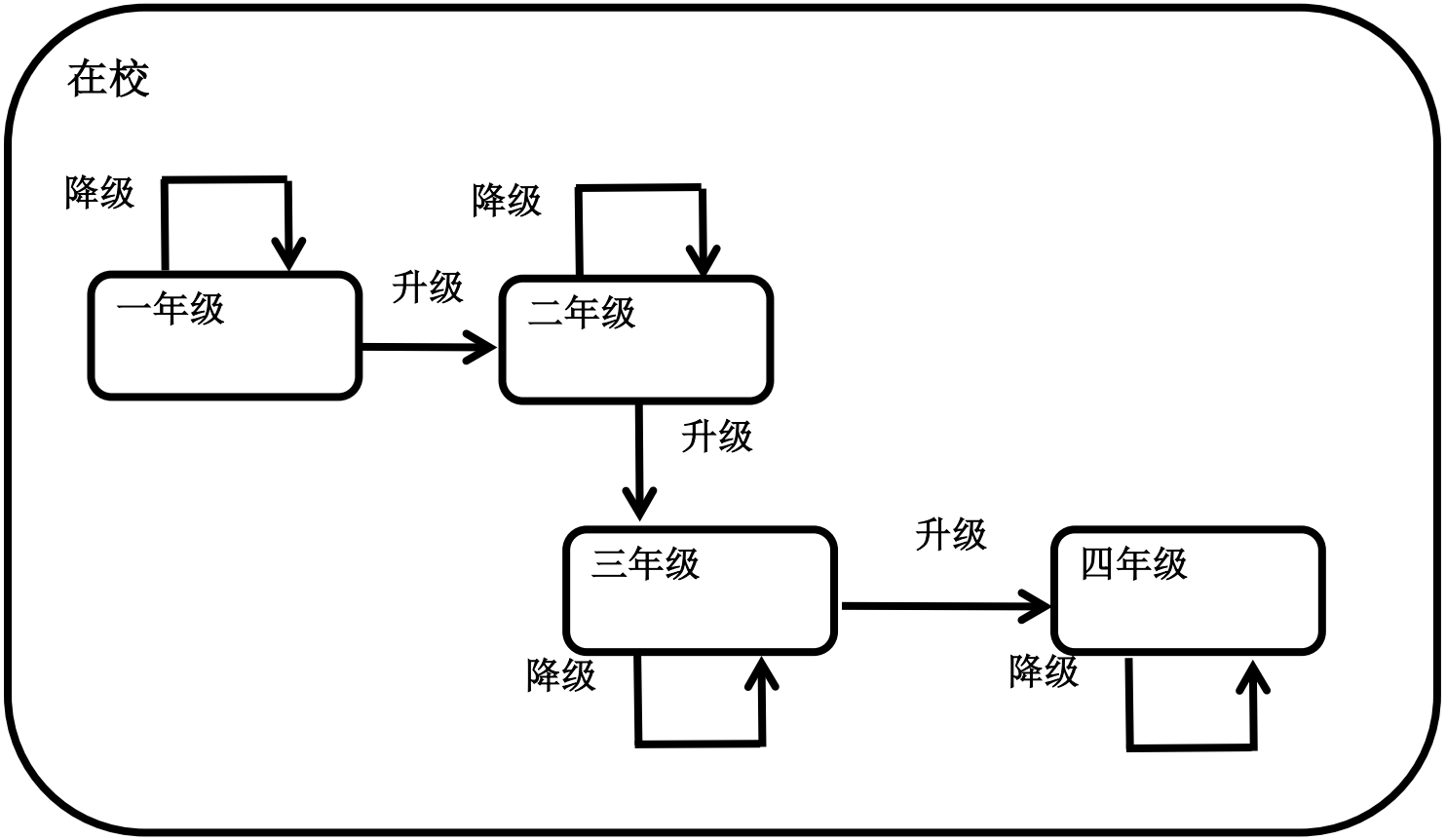
学生学习过程-案例分析1

- 状态图：



学生学习过程-案例分析1

- 可以对每个状态进行详细分析
- 例如前例中的状态“在校”可以细化为



航班机票预订-案例2

- 下面以一个航班机票预订的例子来说明状态机图的建立过程。

航班机票预订-案例2

- 寻找主要状态
  - 在绘制状态机图时，第一步就是寻找出主要的状态。对于航班机票预订系统而言，我们把飞机票看作一个整体，我们来看飞机票有哪几种状态，以及有哪些事件触发机票状态的变化。
  - 确定状态
    - 飞机票有以下4种状态：无预订、部分预订、预订完、预订关闭。
    - 在刚确定飞行计划时，显然是没有任何预订的，并且在顾客预订机票之前都将处于这种“无预订”状态。对于订座而言，显然有“部分预订”和“预订完”两种状态。
    - 当航班快要起飞时，显然要“预订关闭”。

航班机票预订-案例2

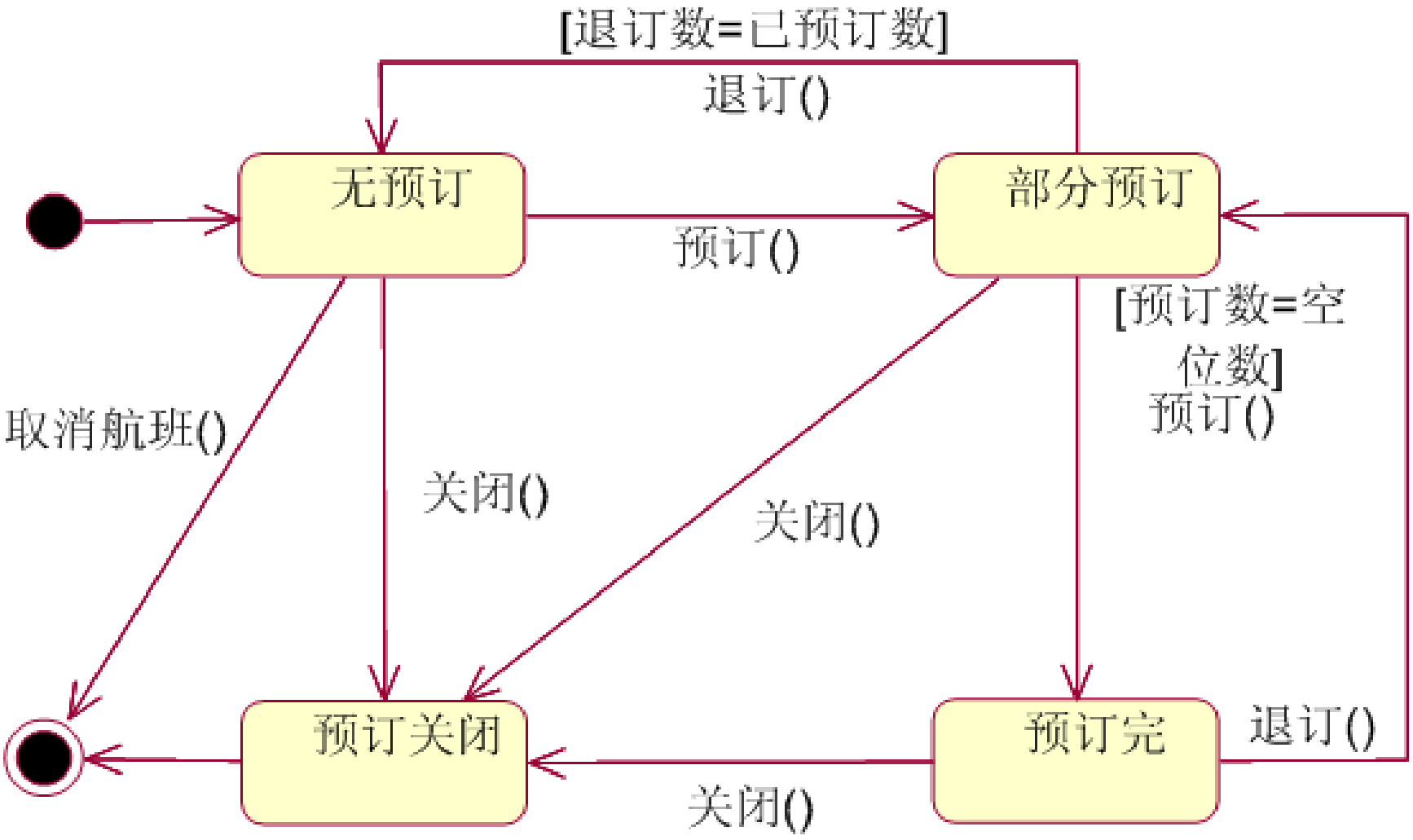
- 寻找外部事件
  - 无论机票处于哪种状态，可能有的外部事件有：
  - 预订()：顾客预订机票。
  - 退订()：顾客退订机票。
  - 关闭()：机票管理员关闭订票系统。
  - 取消航班()：飞机调度人员取消飞行计划。
- 确定状态间的转换
  - 分析状态之间的转换(这里指外部转换)。即，确定当机票处于这一状态时，哪些外部事件能真正改变机票状态，哪些事件对本状态不起作用。可以采用表格的方式来进行分析。

航班机票预订-案例2

- 刚确定航班时，机票无人订，机票处在“无预订”状态；当有顾客预订机票时，机票处在“部分预订”状态；当有人退订时，如果退订时“退订数等于已预订数”，那么退订后状态将回到“无预订”状态。
- 在部分预订状态时，如果再发生预订，而且“预订数=空位数”，那么将订完所有的位置，因此将进入“预订完”状态。
- 当机票处在“预订完”状态时，只要有人退订，就必将转为“部分预订”状态。

航班机票预订-案例2

源目标	无预订	部分预订	预订完	预订关闭
无预订		预订()	不直接转换	关闭()
部分预订	退订()事件发生后，使预订人=0		预订(),无空座	关闭()
预订完	不直接转换	退订()		关闭()
预订关闭	无转换	无转换	无转换	

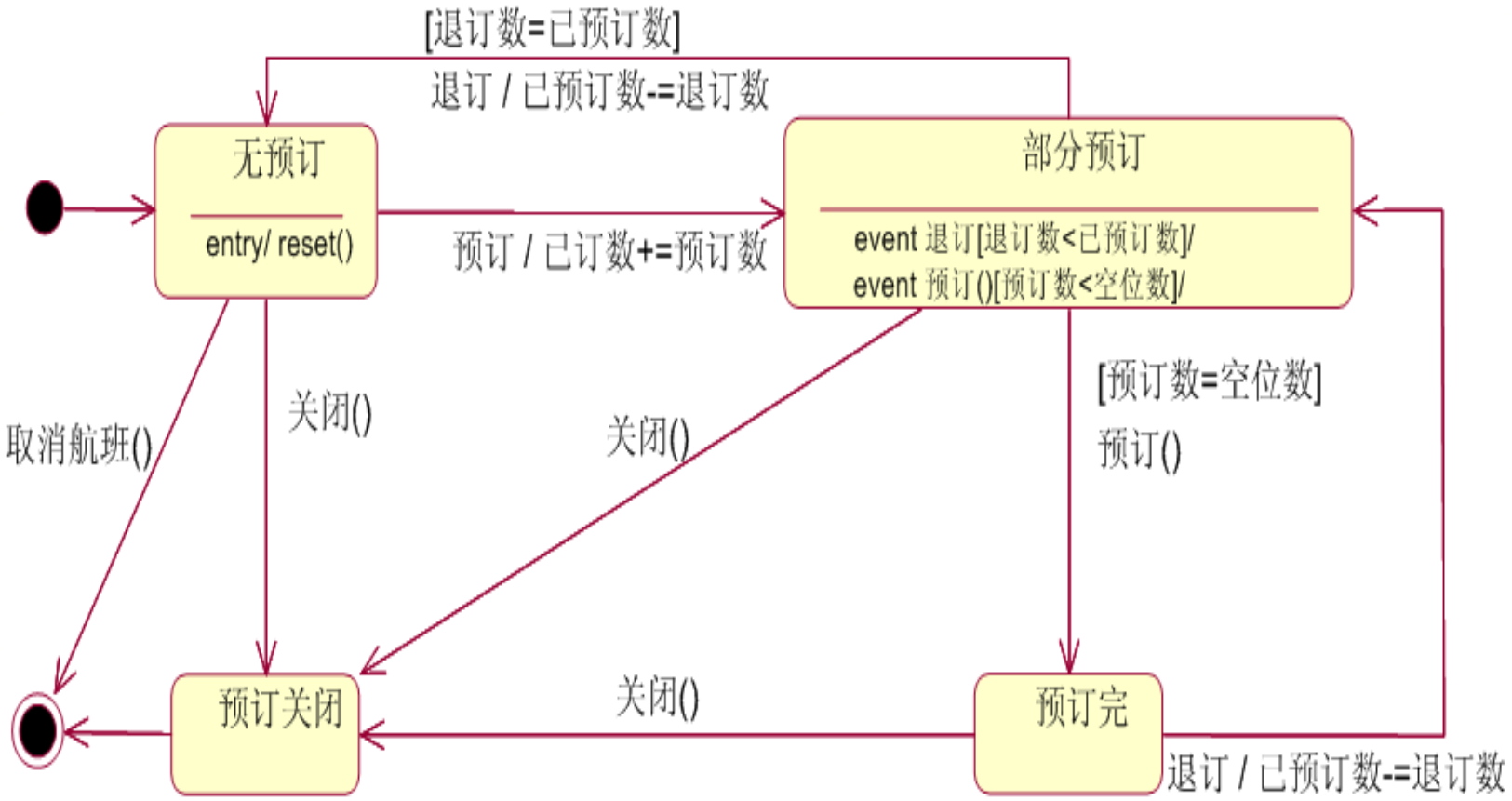

[退订数=已预订数]
退订()
无预订
部分预订
预订()
[预订数=空位数]
预订()
取消航班()
关闭()
关闭()
预订关闭
预订完
退订()
关闭()

航班机票预订-案例2

- 详细描述每个状态和转换
  - 前面已经确定了各个状态之间的外部转换，为了详细描述状态，我们给状态添加内部转换、外部转换时的进入和退出动作，以及相关的活动等。例如，在这个例子中，还存在一些内部转换和活动：
    - 机票处在“部分预订”状态时，当发生退订事件时，如果退订数小于预订数，那么状态不变；同样的道理，当发生预订事件时，如果预订数小于空位数，那么状态也是不变的。
    - 从初态到“无预订”状态时，我们要对机票数、预定数和空座位数进行初始化活动。
    - 当预订事件和取订事件发生时，都应该更新预订数和空位数的值。但由于座位总数是已知的，因此只要更新预订数就可以了。

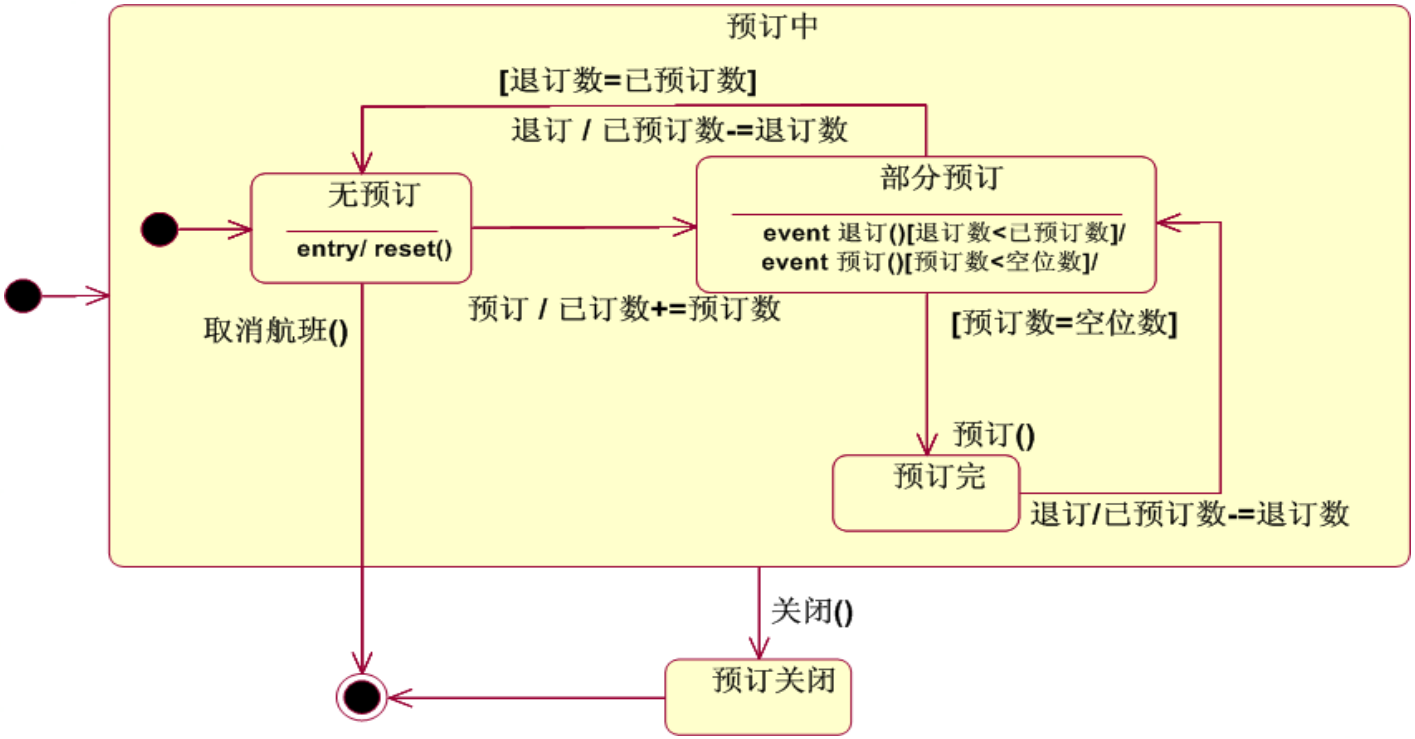
航班机票预订-案例2

- 详细描述后的状态图



航班机票预订-案例2

- 把简单状态图转换为复合状态图
  - 可以将无预定、部分预订、预订完三个状态归结为“预订”状态，这样就可以采用一个复合状态，即“预订”状态来表示该图。



复合状态表示机票预订系统

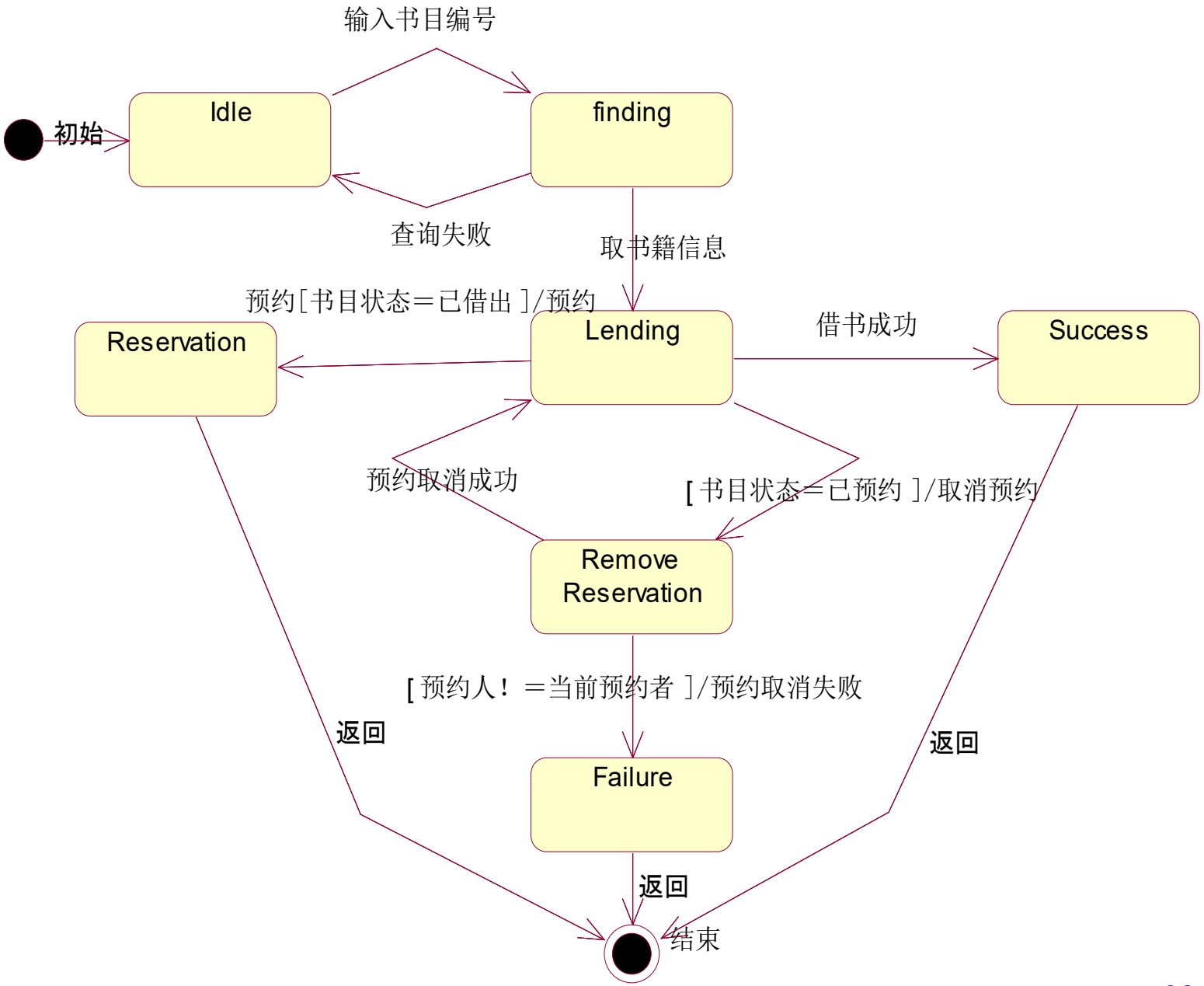
借书状态图-案例3

- 借书业务在系统的业务建模中是一个用例，而这种用例是一个应对型对象。为便于理解该业务的控制流程和确保业务处理的正确性。从前面章节对该业务描述可知，借书业务是由
  - 借书空闲(idle)
  - 书目查询（finding）
  - 借书（Lending）
  - 预约（reservation）
  - 取消预约（remove reservation）
  - 借书成功（Success）
  - 失败（Failure）7种状态组成。

任务解决-主要事件

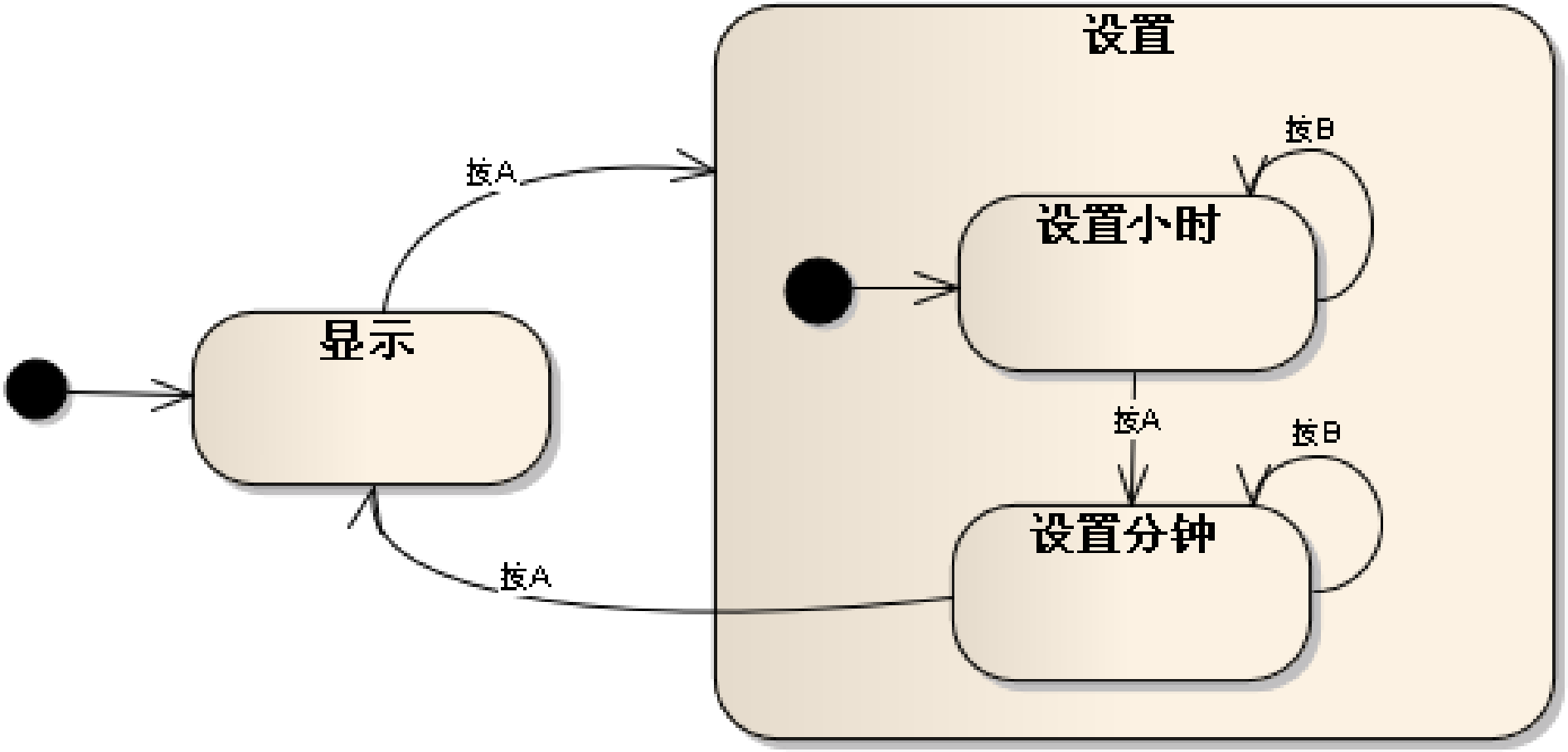
1. 从空闲状态到书目查询状态是由书目编号录入引发的；
2. 同样查询失败也会引发查询状态转换到借书业务的空闲状态；
3. 查询成功的事件会激发从查询状态到借书状态；
4. 当所查到的书在库时则借阅成功转发成功显示状态；
5. 当所查到的书已预约时则激发取消预约事件；
6. 当所查到的书已借出则激发预约事件转入预约状态；
7. 在取消预约时如预约取消成功则转入借书状态；
8. 如预约成功则转入信息显示状态

借书状态图


输入书目编号
初始
Idle
finding
查询失败
取书籍信息
预约[书目状态＝已借出]/预约
Reservation
Lending
借书成功
Success
预约取消成功
[书目状态＝已预约]/取消预约
Remove
Reservation
[预约人！＝当前预约者]/预约取消失败
返回
返回
Failure
返回
结束

案例4

- 简单的数字手表表面上有一个显示屏和两个设置按钮A和B，有两种操作模式：显示时间和设定时间。在显示时间模式下，手表会显示小时和分钟，小时和分钟由闪烁的冒号分隔。设定时间模式有两种子模式：设定小时和设定分钟。按钮A选择模式，每次按下此按钮时，模式会连续前进：设定小时、设定分钟等。在子模式内，每次只要按下按钮B，就会拨快小时或分钟。绘制一个数字手表的状态机图。


设置
按B
按A
设置小时
显示
按A
按B
设置分钟
按A