

# 状态图

首先介绍状态图的基本概念，并具体讲解状态图的几个重要元素，然后介绍如何使用Rose创建状态图，最后通过一个示例详细讲解使用Rose创建状态图的步骤。

# 1 状态图的基本概念

状态图用于描述模型元素的实例  
(如对象或交互) 的行为。

## 1.1 状态图的定义

### 1. 状态机

状态机是一种记录下给定时刻状态的设备，它可以根据各种不同的输入对每个给定的变化而改变其状态或引发一个动作，如计算机、各种客户端软件、Web上的各种交互页面都是状态机。

**状态机由对象的各个状态和连接这些状态的转换组成，是展示状态和状态转换的图**

- 对象的**生命周期**——从开始创建到最终消亡的完整过程
- 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对这些事件的响应
- 状态机**依附于一个类**：描述该类的实例对接收到的事件的响应，将对象孤立地从系统中抽象出来进行观察，将来自外部的影响都抽象为事件。**可依附于用例、操作**等，描述其动态执行过程。
- 状态机将实体和外部世界相互分离，所以是一个对象、协作、或用例的**局部视图**，**适合对局部、细节进行建模**

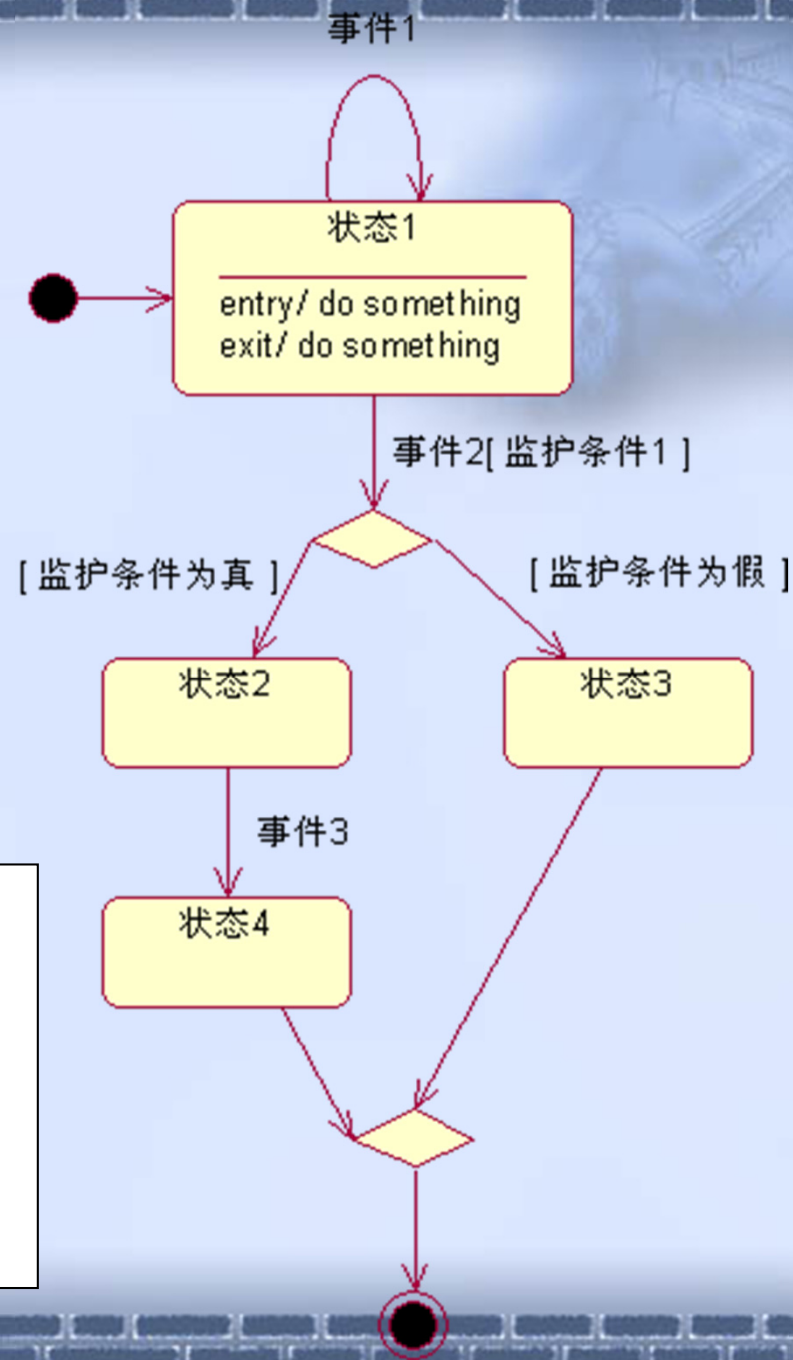


## 2. 状态图

一个状态图 (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投影，这也就意味着状态图包括状态机的所有特征。

状态的转换由事件触发。

**组成：**元素状态、转换、初始状态、终止状态、判定等



状态图适合于描述跨越多个用例的单个对象的行为，而不适合描述多个对象之间的行为协作。状态图描述从状态到状态的控制流，适合于对系统的动态行为建模。在UML中，对系统动态行为建模时，除了使用状态图，还可以使用序列图、协作图和活动图，但这四种图存在着以下重要差别：

- 序列图和协作图用于对共同完成某些对象群体进行建模。
- 状态图和活动图用于对单个对象（可以是类、用例或整个系统的实例）的生命周期建模。

## (1) 状态

状态用于对实体在其生命周期中的各种状况进行建模，一个实体总是在有限的一段时间内保持一个状态。

**状态的描述应该包括状态名、入口、出口动作、内部转换和嵌套状态**

**入口动作、出口动作：一个状态可以没有入口和出口动作（指进入和退出一个状态时所执行的边界动作）。**

**内部转换：不导致状态改变的转换。**

**嵌套状态：状态分为简单状态和组成状态**

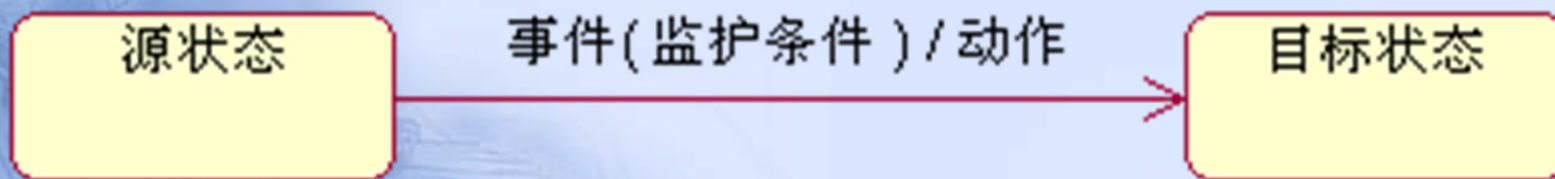
转换上可以标注与此转换相关的事件、监护条件、动作，若未标注触发转换的事件，则表示此转换自动进行

## (2) 转换

在UML的状态建模机制中，转换用带箭头的直线表示，一端连接源状态，箭头指向目标状态。

**事件触发器：**引起源状态转换的事件。事件不是持续发生，只发生在时间的一点上，对象接收到事件，激活转换并使监护条件得到满足

**监护条件：**接收到触发事件触发转换时求布尔表达式值。如果为真，激活转换；如果为假，不激活转换，接收到的触发事件丢失



**动作：**一个可执行的原子计算



### (3) 初始状态

每个状态图都应该有一个初始状态，它代表状态图的起始位置。

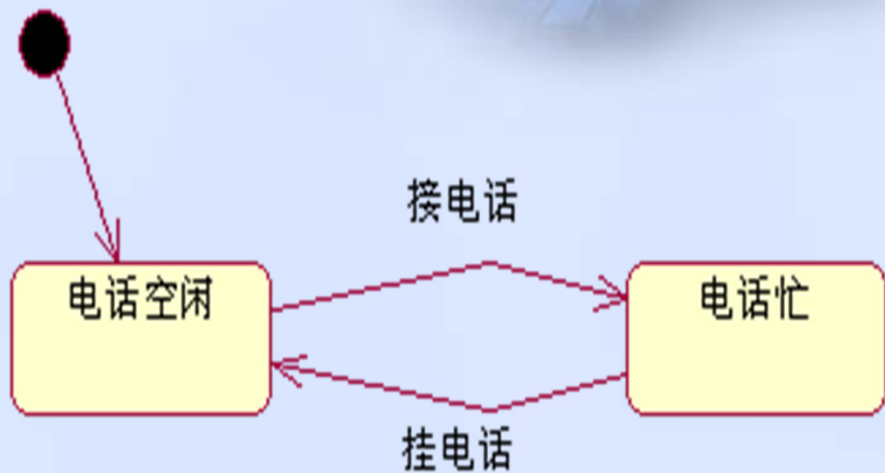
- 初始状态是一个**伪状态**（和普通状态有连接的假状态）
- 对象不能保持在初始状态，必须有无事件触发器的转换
- **初始状态上的转换无监护条件**
- 初始状态只能作为转换的源，不能作为转换的目标
- **一个状态图只能有一个初始状态**

## (4) 终止状态

终止状态是一个状态图的终点，**一个状态图可以拥有一个或者多个终止状态。**

**对象可以保持在终止状态  
但终止状态不能有任何形式的  
触发转换**

**一些特殊的状态图，  
可以没有终止状态**



## (5) 判定

活动图和状态图中都有需要根据给定条件进行判断，然后根据不同的判断结果进行不同的转换的情况。

## 1.2 状态图的作用

状态图清晰地描述了状态之间的转换顺序，通过状态的转换顺序可以清晰看出事件的执行顺序。

清晰的事件顺序有利于程序员在开发程序时避免出现事件错序的情况。

状态图清晰地描述了状态转换时所必须触发的事件、监护条件和动作等影响转换的因素。状态图通过判定可以更好地描述工作流因为不同的条件发生的分支。



## 2 状态图的组成

### 2.1 状态

状态是状态图的重要组成部分，它描述了一个类对象生命周期中的一个时间段。

#### 1. 状态名

状态名可以把一个状态和其他状态区分开来。在实际使用中，状态名通常是直观、易懂、能充分表达语义的名词短语，其中每个单词的首字母要大写。

#### 2. 内部活动

状态可以包含表达式的内部活动。当状态进入时活动在进入动作完成后就开始。

### 3. 内部转换

状态可能包含一系列的内部转换，内部转换因为只有源状态而没有目标状态，所以内部转换的结果并不改变状态本身。

### 4. 入口和出口动作

状态可能具有入口和出口动作。这些动作的目的是封装这个状态，这样就可以不必知道状态的内部状态而在外部使用它。

### 5. 历史状态

组成状态可能包含历史状态 (History State)，历史状态本身是个伪状态，用来说明组成状态曾经有的子状态。

- 一般情况下，当状态机通过转换进入组成状态嵌套的子状态时，被嵌套的子状态要从子初始状态进行。但是如果一个被继承的转换引起从复合状态的自动退出，状态会记住当强制性退出发生的时候处于活动的状态。这种情况下就可以直接进入上次离开组成状态时的最后一个子状态，而不必从它的子初始状态开始执行。
- 历史状态可以有来自外部状态或者初始状态的转换，也可以有一个没有监护条件的出发完成转换；转换的目标是默认的历史状态。如果状态区域从来没有进入或者已经退出，到历史状态的转换会到达默认的历史状态。



- 历史状态代表上次离开组成状态时的最后一个活动子状态，可分为浅历史状态和深历史状态：浅历史状态保存并激活与历史状态在同一个嵌套层次上的状态；深历史状态保存在最后一个引起封装组成状态退出的显式转换之前处于活动的所有状态。它可能包含嵌套在组成状态里的任何深度的状态。要记忆深状态，转换必须直接从深状态中转出。
- 浅历史状态，只记住直接嵌套的状态机的历史，使用一个含有字母H的小圆圈表示；深历史状态，会在任何深度上记住最深的嵌套状态，使用内部含有H\*的小圆圈表示，如图所示。

H

浅历史状态

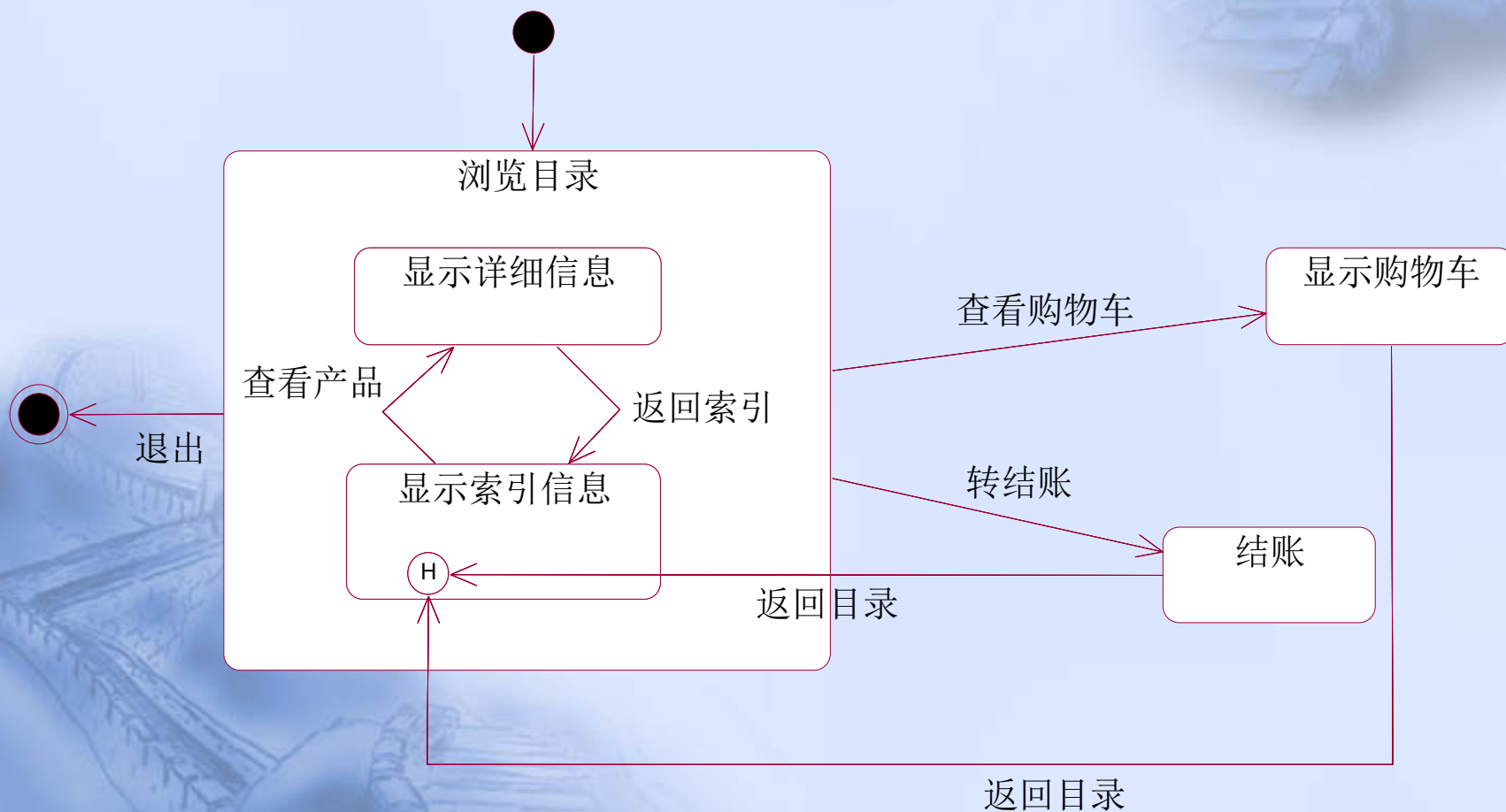
H\*

深历史状态



- 如果一个转换先从深状态转换到一个浅状态，并由浅状态转出组成状态，记忆的将是浅状态的转换。如果组成状态进入终态，则它将丢弃所有保存的历史状态。一个组成状态最多只有一种历史状态，每个状态可能有它自己的默认历史状态。

- 如图所示，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，转到购物车或结账区之后，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。



历史状态虽然有它的优点，但是它过于复杂，而且不是一种好的实现机制，尤其是深历史状态更容易出问题。在建模的过程中应该尽量避免历史机制，使用更易于实现的机制。

## 2.2 转换

转换用于表示一个状态机的两个状态之间的一种关系，即一个在某初始状态的对象通过执行指定的动作并符合一定的条件下进入第二种状态。在这个状态的变化中，转换被称作激发。在激发之前的状态叫做**源状态**，在激发之后的状态叫做**目标状态**。简单转换只有一个源状态和一个目标状态。复杂转换有不只一个源状态和有不只一个目标状态。一个转换通常由**源状态、目标状态、事件触发器、监护条件和动作组成**。在转换中，这5部分信息并不一定都同时存在，有一些可能会缺少。

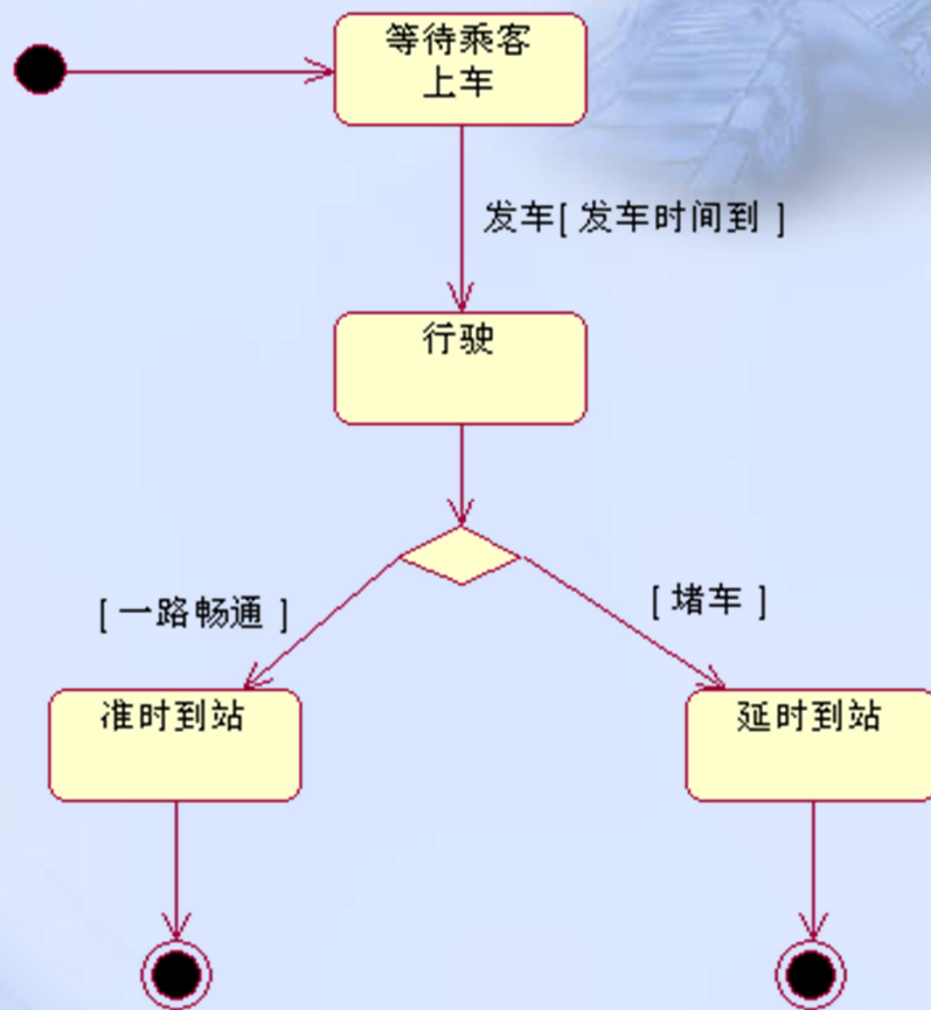
语法形式：

**转换名：事件名 参数列表 监护条件/动作列表**



## 1. 外部转换

外部转换是一种改变状态的转换，也是最普通、最常见的一种转换。



## 2. 内部转换

**内部转换只有源状态，没有目标状态，不会激发入口和出口动作，因此内部转换激发的结果不改变本来的状态。**

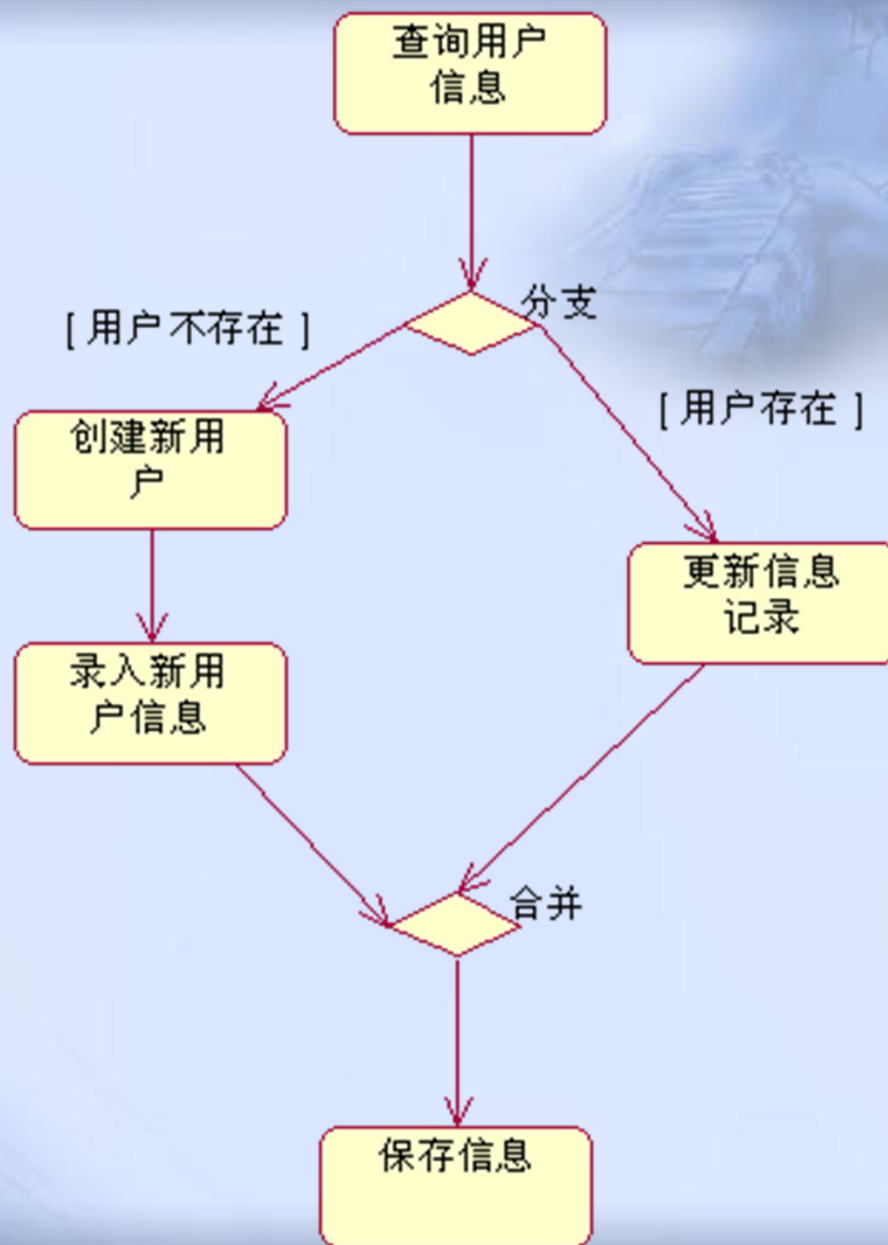
## 3. 完成转换

**完成转换没有明确标明触发器事件的转换是由状态中活动的完成引起的。**

## 4. 复合转换

### 复合转换

(Complex Transition) 由简单转换组成，这些简单转换通过分支和合并组合起来，因此复合转换可以具有多个源状态和多个目标状态。



## 5. 监护条件

转换可能具有一个监护条件，监护条件是一个布尔表达式，它是触发转换必须满足的条件。





## 6. 触发器事件

触发器事件就是能够引起状态转换的事件。如果此事件有参数，则这些参数可以被转换所用，也可以被监护条件和动作的表达式所用。

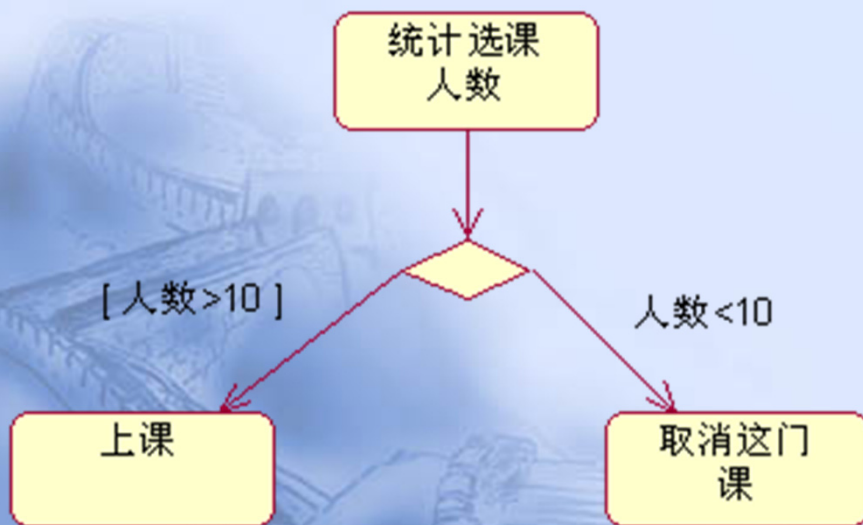
## 7. 动作

动作 (Action) 通常是一个简短的计算处理过程或一组可执行语句。动作也可以是一个动作序列，即一系列简单的动作。动作是原子型的，所以动作是不可中断的，动作和动作序列的执行不会被同时发生的其他动作影响或终止。动作的执行时间非常短，所以动作的执行过程不能再插入其他事件。如果在动作的执行期间接收到事件，那么这些事件都会被保存，直到动作结束，这时事件一般已经获得值。

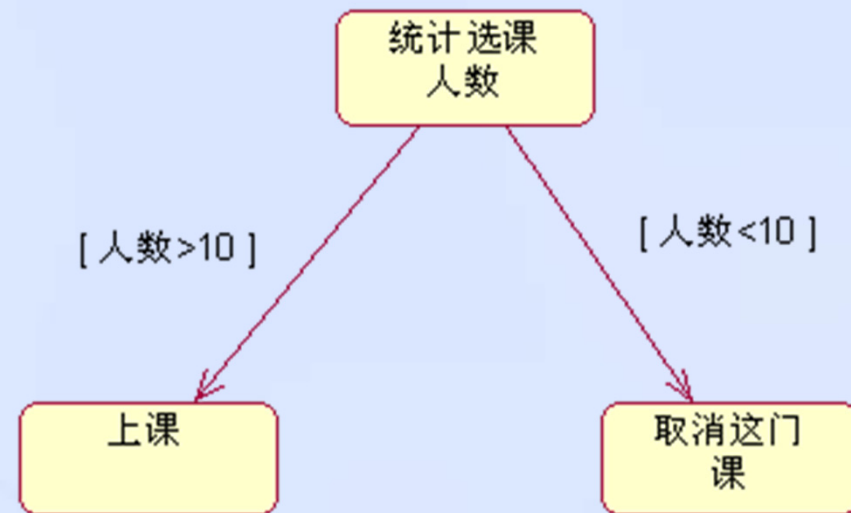
## UML建模语言

### 2.3 判定

判定用来表示一个事件依据不同的监护条件有不同的影响。在实际建模的过程中，如果遇到需要使用判定的情况，通常用监护条件来覆盖每种可能，使得一个事件的发生能保证触发一个转换。通常情况下判定有一个转入和两个转出，根据监护条件的真假可以触发不同的分支转换，如图所示。使用判定仅仅是一种表示上的方便，不会影响转换的语义，。



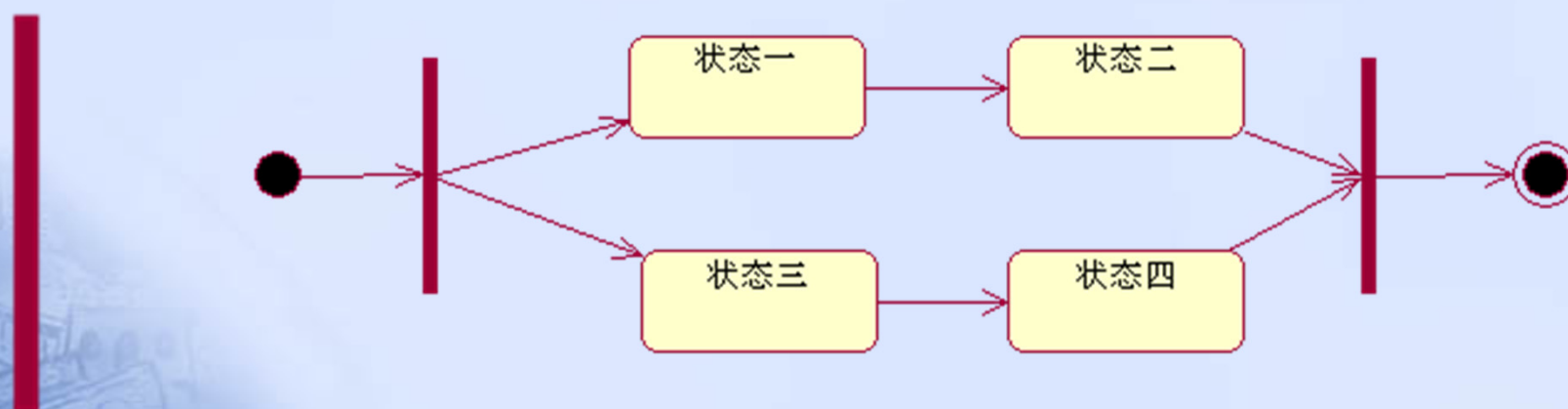
判定示例



无判定示例

## 2.4 同步

同步是为了说明并发工作流的分支与汇合。状态图和活动图中都可能用到同步。



同步

同步示例

- 需要注意同步与判定的区别。同步和判定都会造成工作流的分支，初学者很容易将两者混淆。它们的区别是：判定是根据监护条件使工作流分支，监护条件的取值最终只会触发一个分支的执行，如有分支**A**和分支**B**，假设监护条件为真时执行分支**A**，那么分支**B**就不可能被执行。反之则执行分支**B**，分支**A**就不可能被执行。而同步的不同分支是并发执行，并不会因为一个分支的执行造成其他分支的中断。



## 2.5 事件

一个事件的发生能触发状态的转换，事件和转换总是相伴出现。事件既可以是内部事件，又可以是外部事件，可以是同步的，也可以是异步的。内部事件是指在系统内部对象之间传送的事件，例如，异常就是一个内部事件。外部事件是指在系统和它的参与者之间传送的事件，例如，在指定文本框中输入内容就是一个外部事件。

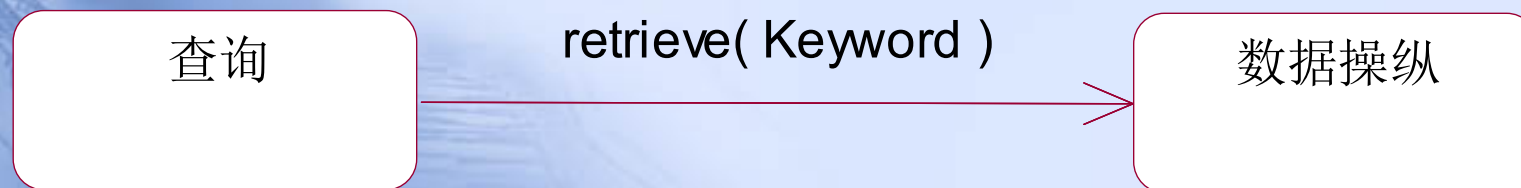
事件可以分成几种，主要包括：信号事件、调用事件、改变事件、时间事件、延迟事件等。

## 1. 信号事件 (Signal Event)

- 信号是作为两个对象之间的通信媒介的命名的实体，信号的接收是信号接受对象的一个事件。发送对象明确地创建并初始化一个信号实例并把它发送到一个或一组对象。最基本的信号是异步单路通信，发送者不会等待接收者如何处理信号而是独立地做它自己的工作。在双路通信模型中，要用到多路信号，即至少要在每个方向上有一个信号。发送者和接受者可以是同一个对象。
- 信号可以在类图中被声明为类元，并用构造型《**signal**》表示，信号的参数被声明为属性。同类元一样，信号间可以有泛化关系，信号可以是其他信号的子信号，它们继承父信号的参数，并且可以触发依赖于父信号的转换。
- 信号事件和调用事件的表示格式是一样的。

## 2. 调用事件（Call Event）

- 调用事件表示调用者对操作的请求，调用事件至少涉及两个及以上的对象，一个对象请求调用另一个对象的操作。
- 调用事件一般为同步调用，也可以是异步调用。如果调用者需等待操作的完成，则是同步调用，否则是异步调用。
- 调用事件的定义格式为：
- 事件名（参数列表）
- 参数的格式为：
- 参数名：类型表达式
- 如图所示，转换上标出了一个调用事件，其名称为“**retrieve**”，带有参数“**Keyword**”。当在状态“查询”中发生调用事件“**retrieve**”时，则触发状态转换到“数据操纵”，要求执行操作“**retrieve(Keyword)**”，并且等待该操作的完成。



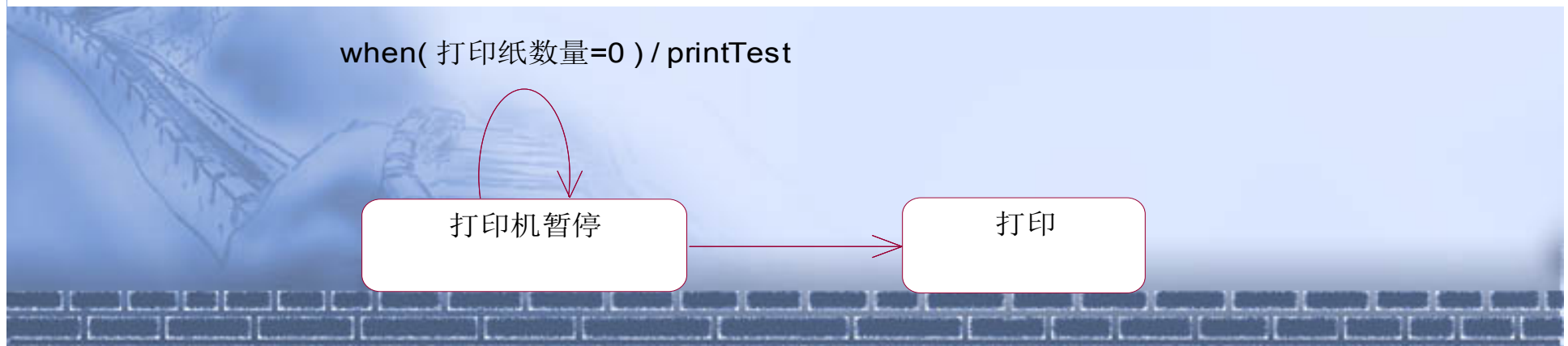
### 3. 改变事件 (Change Event)

改变事件指的是依赖与特定属性值的布尔表达式所表示的条件满足时，事件发生改变。改变事件用关键字when来标记，包含由一个布尔表达式指定的条件，事件没有参数。这种事件隐含一个对条件的连续的测试。当布尔表达式的值从假变到真时，事件就发生。要想事件再次发生，必须先将值变成假，否则，事件不会再发生。建模人员可以使用诸如when(time=8:00)的表达式来标记一个绝对的时间，也可以用如when(number<100)之类的表达式来对其进行连续测试。

改变事件的定义格式为：

when(布尔表达式)/动作

如图所示，“打印机暂停”状态有一个自转换，其上标出了改变事件的条件是“打印纸数量=0”，动作是“printTest”。

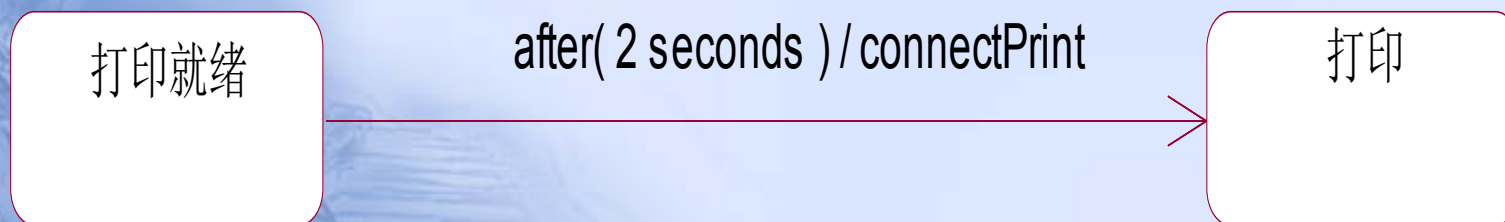




- 要小心使用改变事件，因为它表示了一种具有事件持续性的并且可能是涉及全局的计算过程。它使修改系统潜在值和最终效果的活动之间的因果关系变得模糊。可能要花费很大的代价测试改变事件，因为原则上改变时间是持续不断的。因此，改变事件往往用于当一个具有更明确表达式的通信形式显得不自然时。
- 注意改变事件与监护条件的区别：监护条件仅只在引起转换的触发器事件触发时或者事件接受者对事件进行处理时被赋值一次。如果为假，那么转换不激发并且事件被遗失，条件也不会再被赋值。而改变事件隐含连续计算，因此可以对改变事件连续赋值，直到条件为真激发转换。

#### 4. 时间事件（Time Event）

- 时间事件是经过一定的时间或者到达某个绝对时间后发生的事件，用关键字**after**来标识，包含时间表达式，后跟动作。如果没有特别说明，表达式的开始时间是进入当前状态的时间。
- 时间事件的定义格式为：
- **after(时间表达式)/动作**
- 如图所示，在“打印就绪”状态和“打印”状态之间的转换上列出了一个时间事件“**after(2 seconds)/connectPrint**”，说明若在“打印就绪”状态的时间达2秒钟就执行动作“**connectPrint**”，连接打印机，转换到“打印”状态。



## 5. 延迟事件（Deferred Event）

- 延迟事件是在本状态不处理、推迟或排队等到另外一个状态才处理的事件，用关键字**defer**来标识。
- 延迟事件的定义格式为：
- 延迟事件/**defer**
- 通常，在一个状态的生存期出现的事件，若不被立即响应，就会被丢失。这些未立即触发转换的事件，可以放入一个内部的延迟事件队列，直到它被需要或被撤销为止。如果一个转换依赖于一个事件，而该事件已在内部的事件队列中，则立即触发该转换。如果存在多个转换，则在内部的延迟事件队列中的首个事件将优先触发相应的转换。
- 例如图中的延迟事件是“**Print/defer**”，在当前状态下不执行打印，而将打印事件放进队列中排队，要求延迟到后面的状态中再执行。



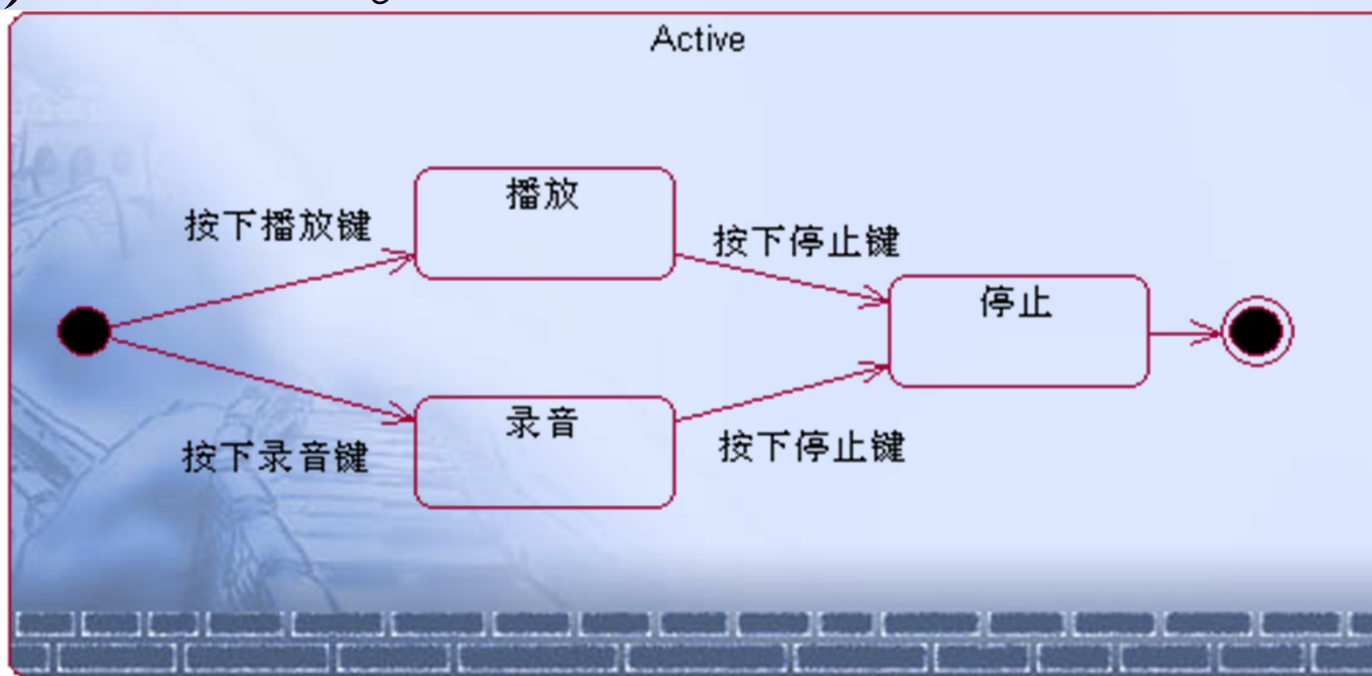


### 3 组成状态

**组成状态 (Composite State) 是内部嵌套有子状态的状态。**

#### 1. 顺序组成状态

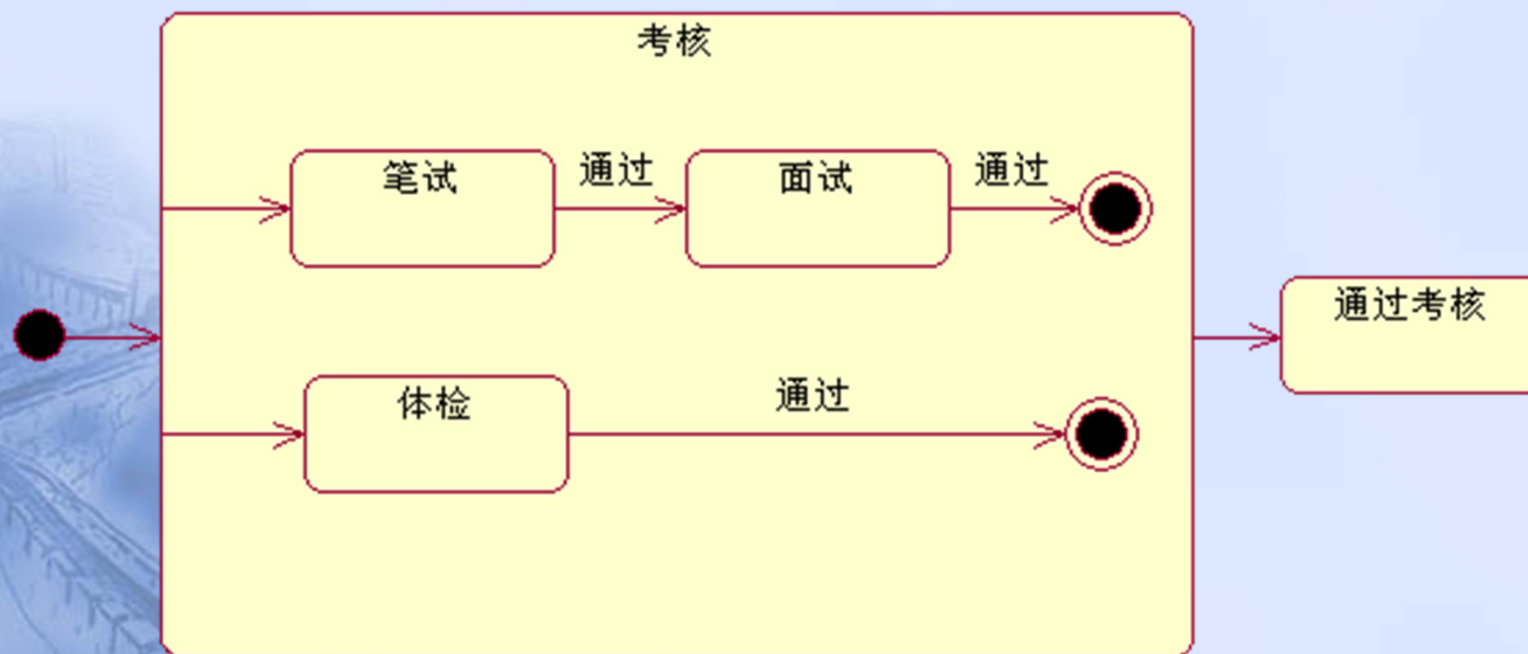
**一个顺序组成状态最多可以有一个初始状态和一个终态，同时也最多可以有一个浅 (Shallow) 历史状态和一个深 (Deep) 历史状态。**





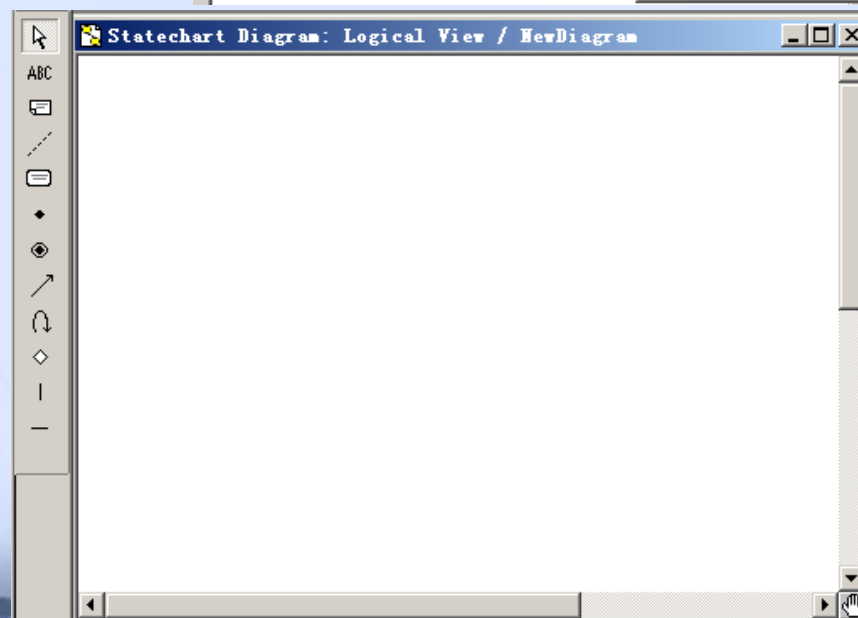
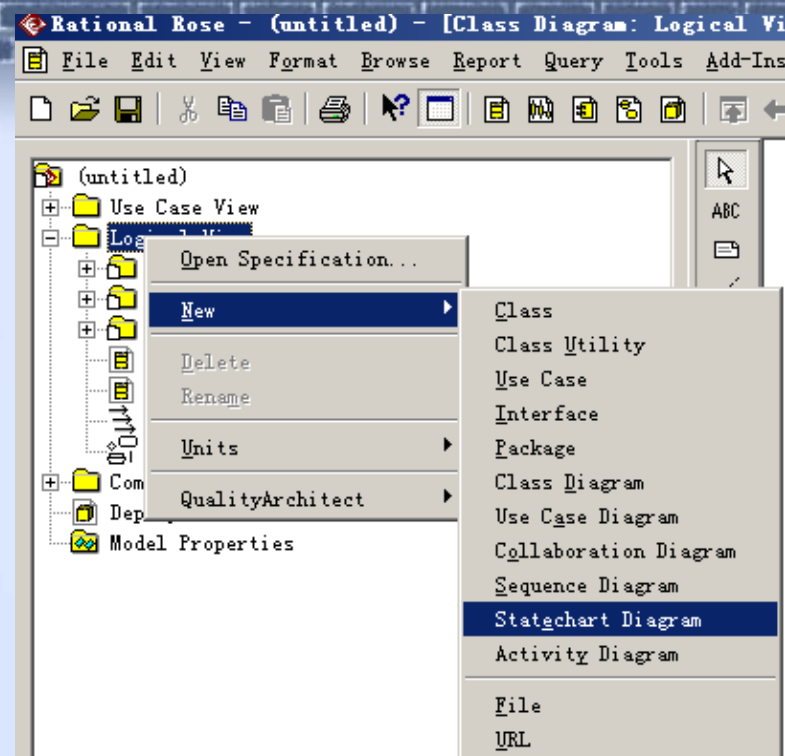
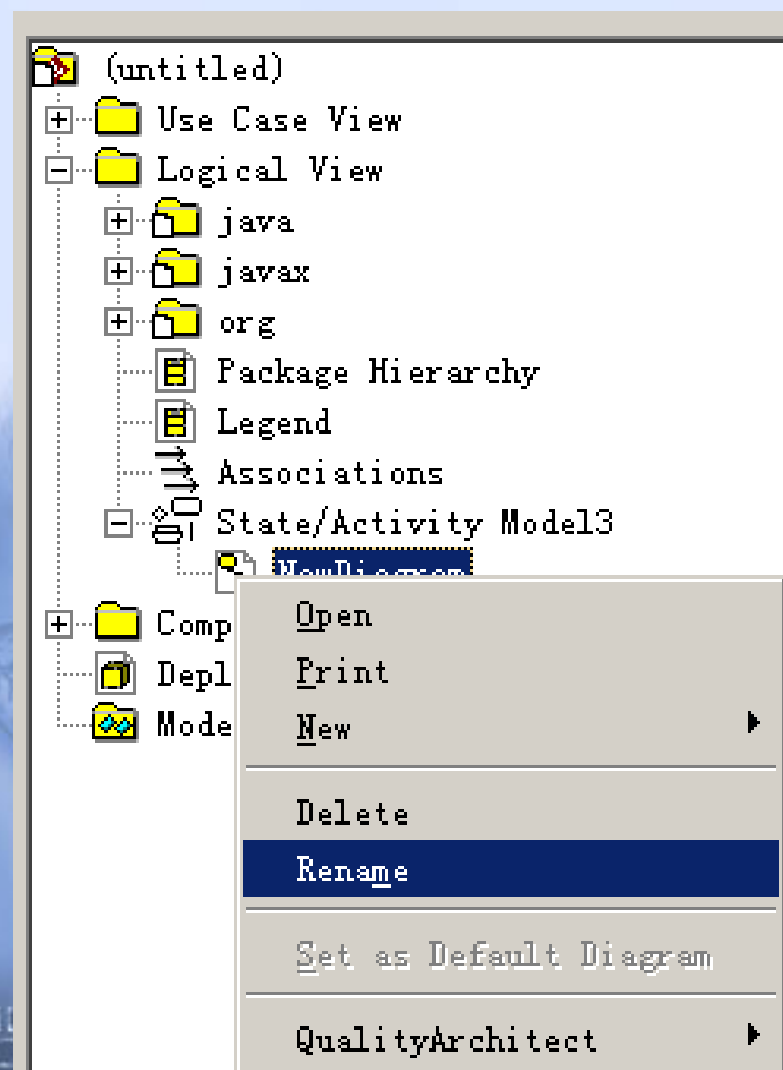
## 2. 并发组成状态

在一个组成状态中，可能有两个或者多个并发的子状态机，称这样的组成状态为并发组成状态。



## 4 状态图的创建概述

### 4.1 创建状态图



## 4.2 创建初始和终止状态

初始状态和终止状态是状态图中的两个特殊状态。初始状态代表着状态图的起点，终止状态代表着状态图的终点。



## 4.3 创建状态

**创建状态的步骤可以分为：创建新状态、修改新状态名称、增加入口和出口动作、增加活动。**



## 1. 创建新状态

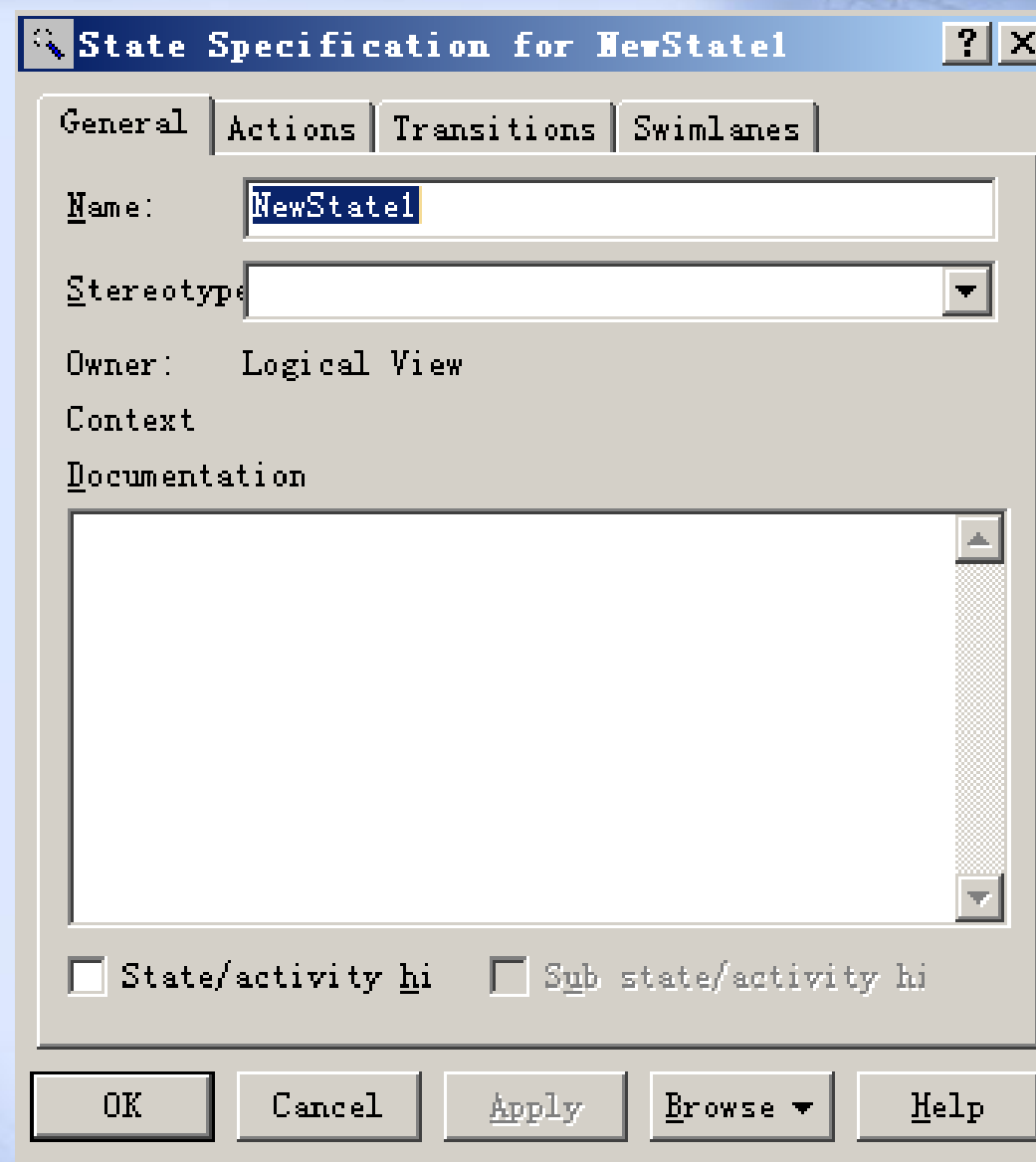
单击状态图工具栏中的图标，然后在绘制区域单击鼠标左键。



NewState1

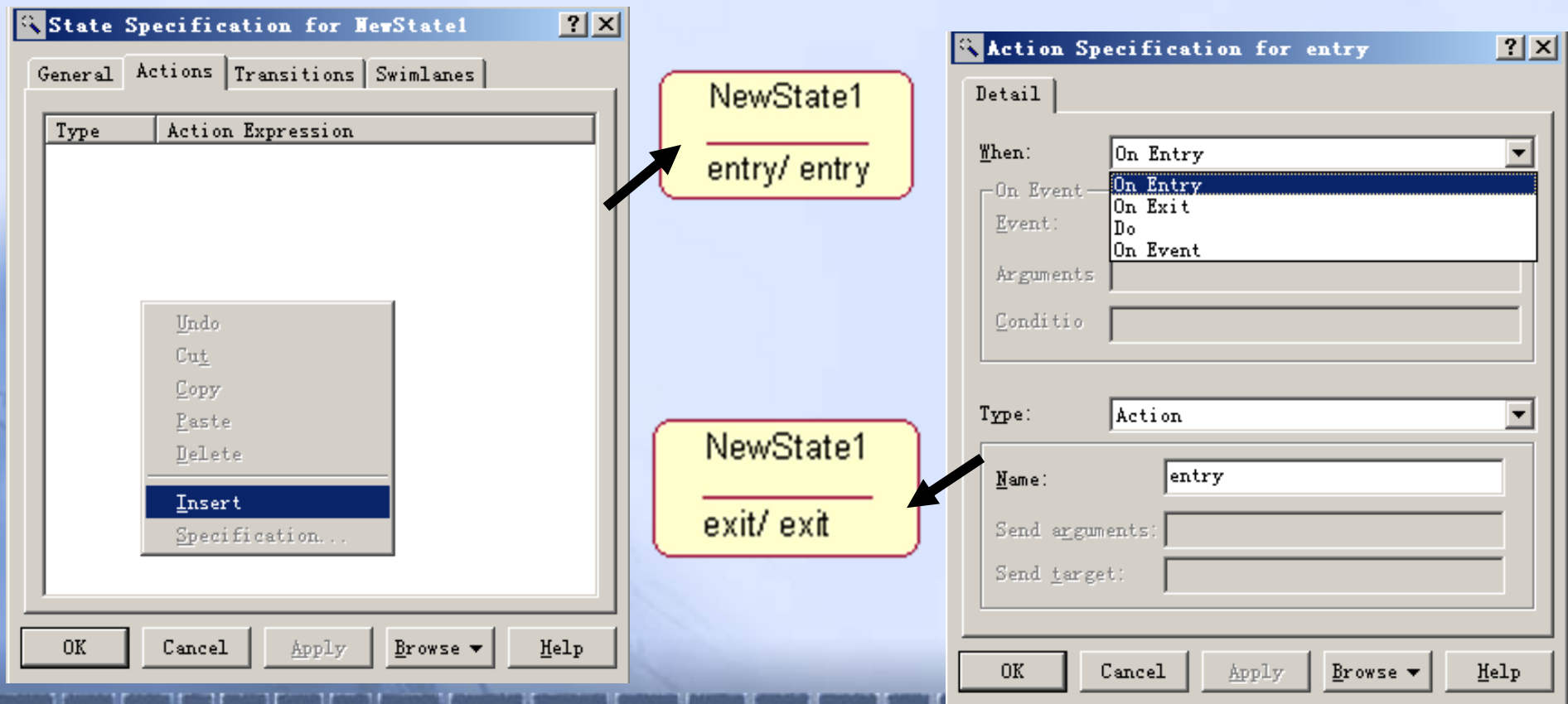
## 2. 修改新状态名称

创建新的状态后可以修改状态的属性信息。双击状态图标，在弹出对话框的General选项卡里进行名称Name和文档说明Documentation等属性的设置。



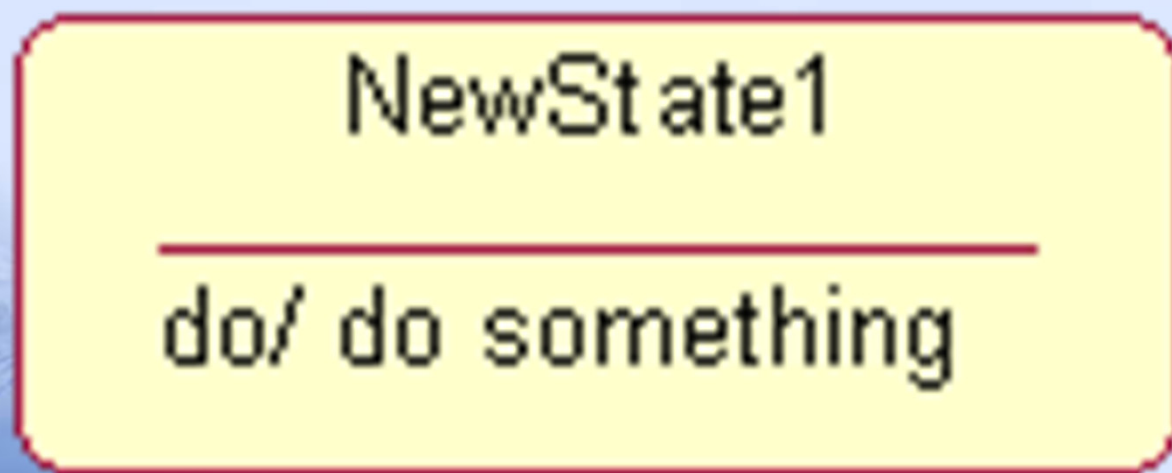
### 3. 增加入口和出口动作

在状态属性设置对话框中打开Actions选项卡，在空白处单击鼠标右键，在弹出的快捷菜单中选择Insert命令，双击出现的动作类型Entry，在弹出对话框的When下拉列表中选择On Entry选项，在Name文本框中添加动作的名称。



#### 4. 增加活动

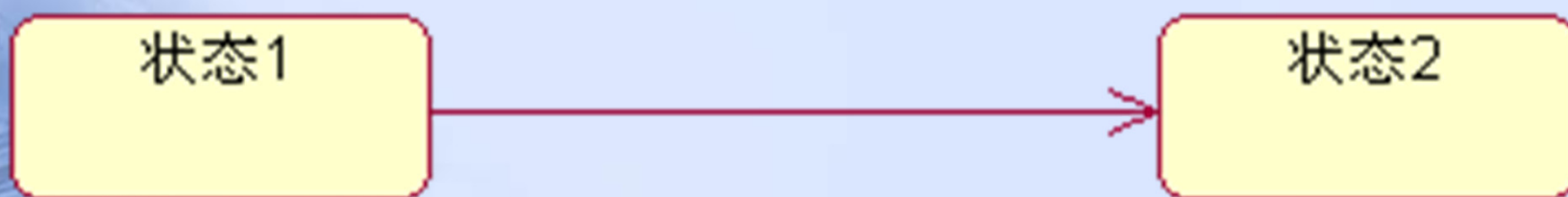
增加活动与增加入口动作和出口动作类似，区别是在When下拉列表中选择Do选项。





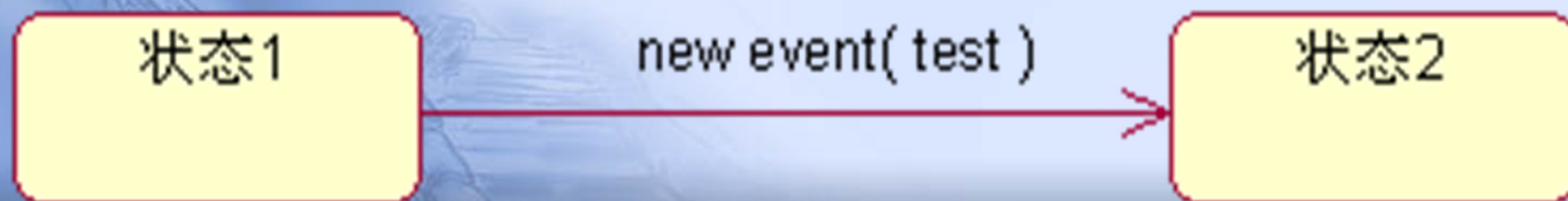
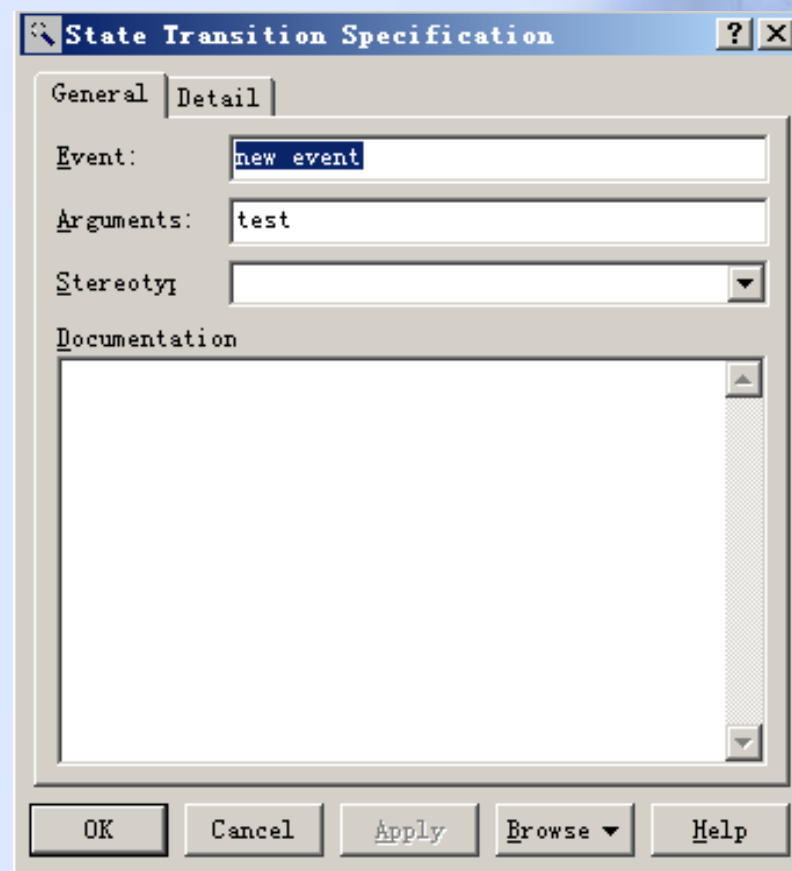
## 4.4 创建状态之间的转换

转换是两个状态之间的一种关系，代表了一种状态到另一种状态的过渡，在UML中转换用一条带箭头的直线表示。



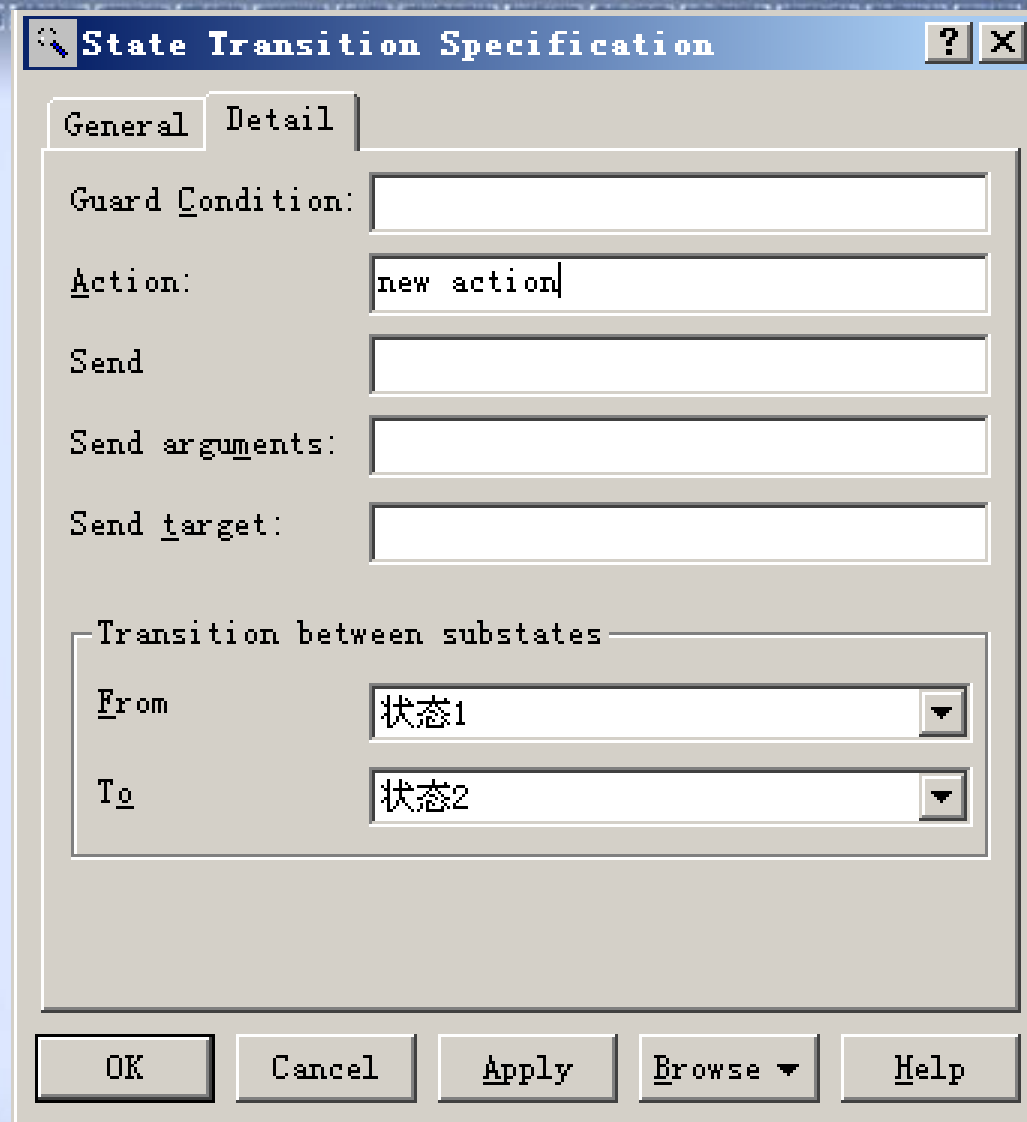
## 4.5 创建事件

双击转换图标，在弹出对话框的General选项卡里增加事件即可。在Event文本框中添加触发转换的事件，在Arguments文本框中添加事件的参数。

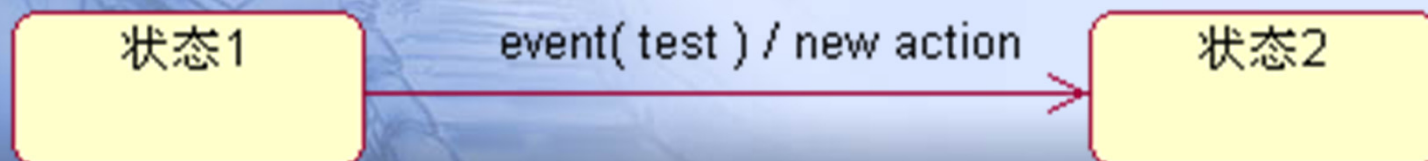


## 4.6 创建动作

双击转换的图标，在弹出的对话框中打开Detail选项卡，在Action文本框中添加要发生的动作。

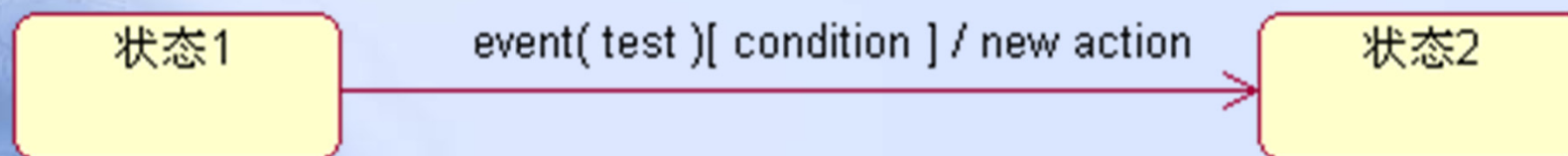


The image shows a 'State Transition Specification' dialog box with two tabs: 'General' and 'Detail'. The 'Detail' tab is selected. It contains several input fields: 'Guard Condition:', 'Action:' (containing 'new action'), 'Send', 'Send arguments:', and 'Send target:'. Below these is a section titled 'Transition between substates' with 'From' and 'To' dropdown menus, both containing '状态1' and '状态2' respectively. At the bottom are buttons for 'OK', 'Cancel', 'Apply', 'Browse', and 'Help'.



## 4.7 创建监护条件

双击转换的图标，在弹出的对话框中打开Detail选项卡，在Guard Condition文本框中添加监护条件。





## 5 状态图的创建示例

创建一个状态图的步骤如下：

标识出建模实体。

标识出实体的各种状态。

创建相关事件并创建状态图。

## 5.1 标识建模实体

**一般来说，不需要给所有的类都创建状态图，只有具有重要动态行为的类才需要。**

## 5.2 标识实体的各种状态

对于一个学生账号来说，它的状态主要包括以下几种：

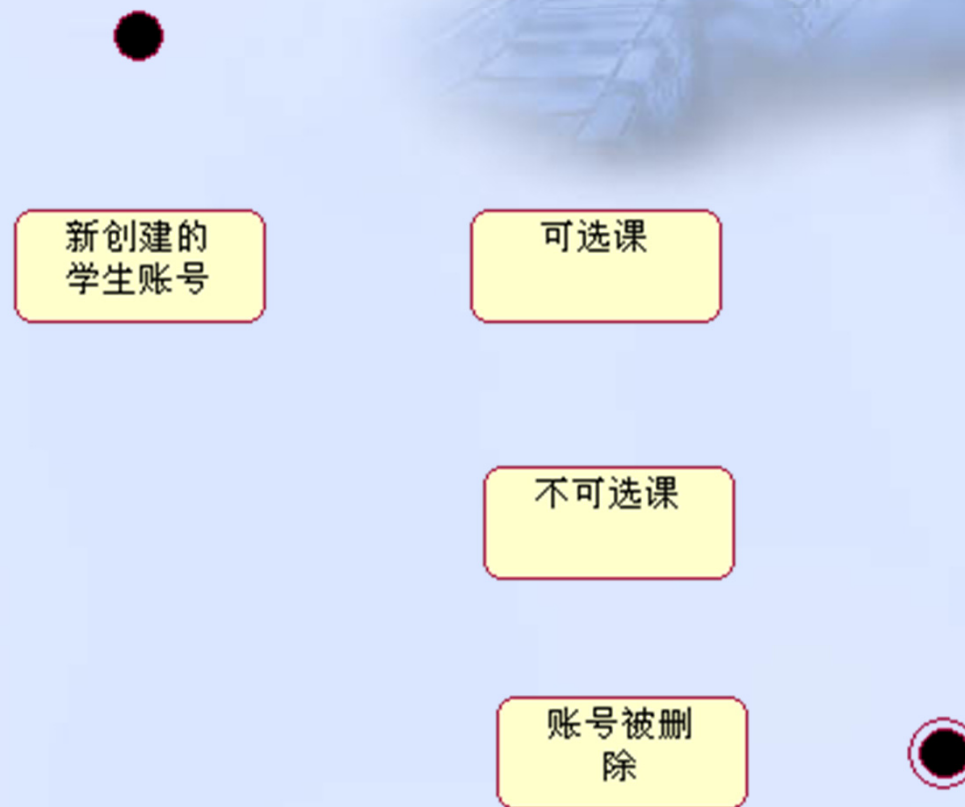
初始状态。

终止状态。

可选课状态。

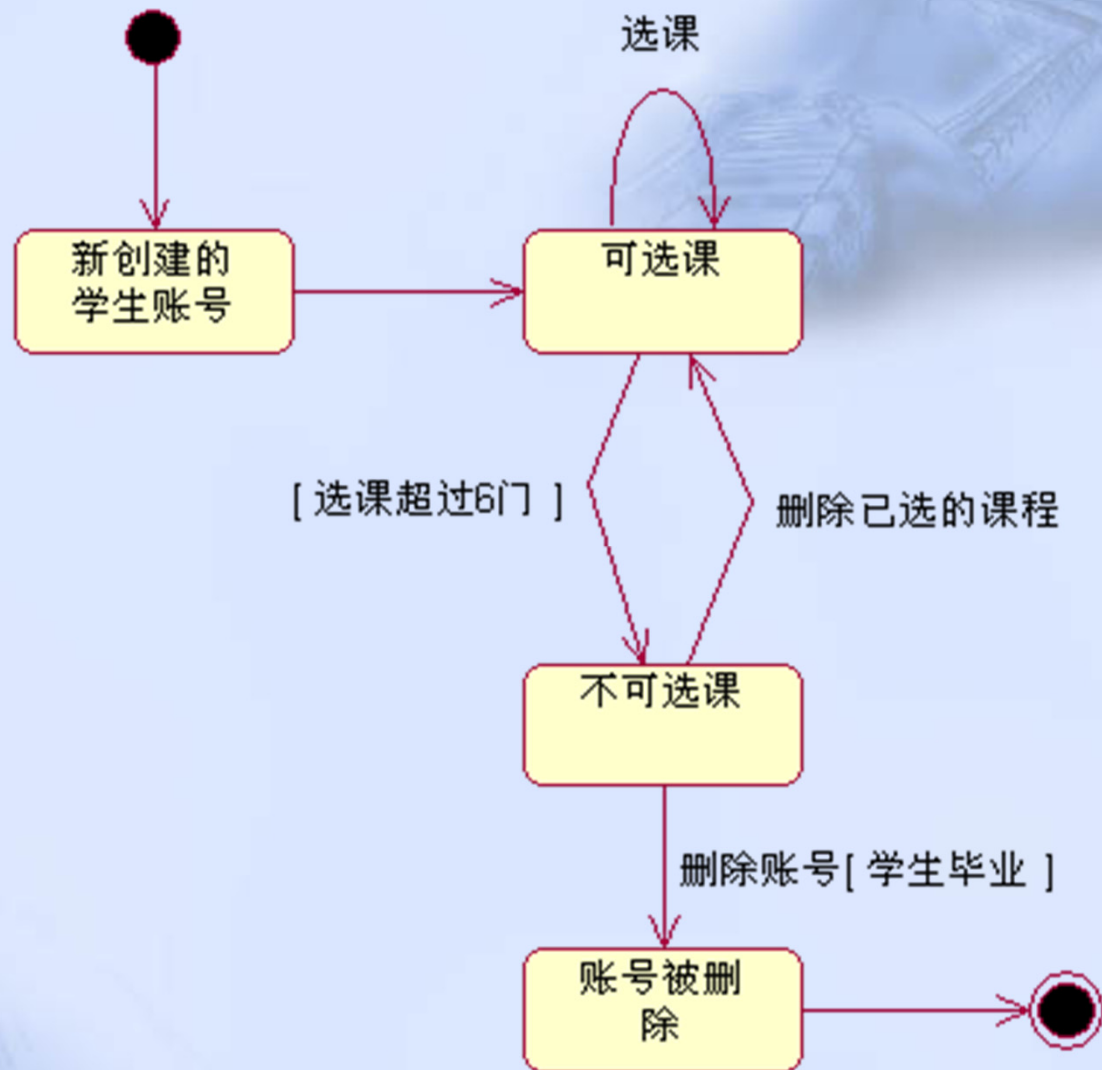
不可选课状态。

账号被删除状态。



## 5.3 标识相关事件并创建状态图

当确定了需要建模的实体并找出了实体的初始状态、终止状态以及其他相关状态后，就可以着手创建状态图。





## 6 本章小结

本章首先介绍了状态图的概念和作用，讲解了状态图的重要组成元素：状态、转换、初始状态、终止状态和判定。接着又介绍了如何通过Rational Rose创建状态图和状态图的各个元素，并创建它们之间的关系。最后通过实例具体讲解了如何创建状态图。