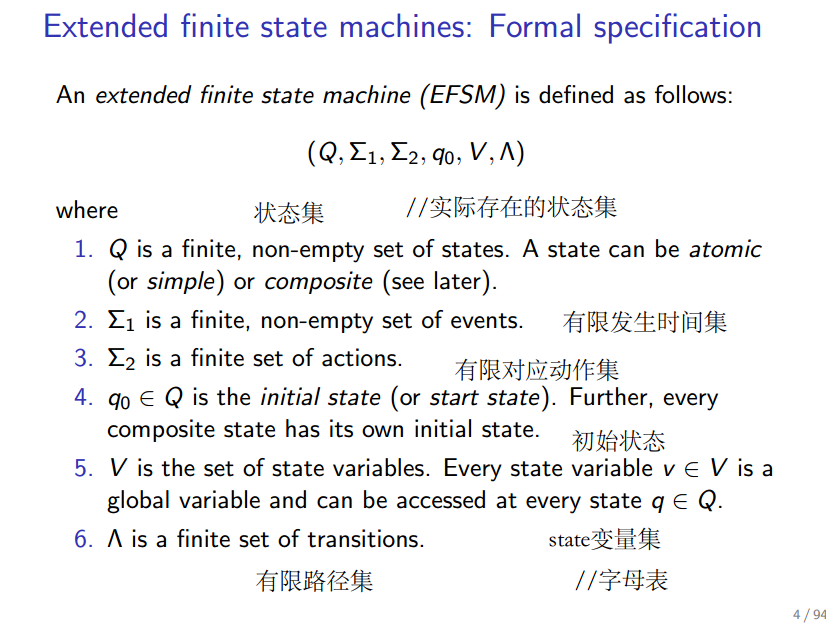
用UML specification//规范 来表示 Extended finite state machines，这样的machine叫做UML state machines

Event-driven systems

这个系统的行为是由event事件决定的(用户操作，传感器收到信号，时间信号，其他系统的信息…)

我们把这类系统叫做event-driven system(reactive system)

一旦受到了一个event，这类系统就会做出对应的反应动作



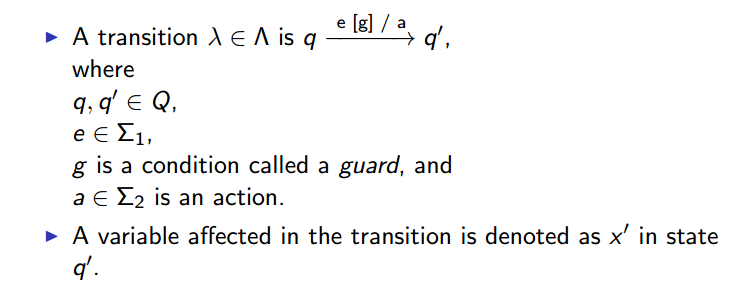
总结，EFSM圈里的就是q0,q1,q2,q3，他们组成了Q

V:我们假设Q有三个状态1，2-9,10

那么V是我们需要的一个全球变量，用来在每个不同的state设立对应guard or action, 我们假设V={size=1,2,3,4,5,6,7,8,9,10}

V是全局变量，不管你在哪个状态，都可以用V，比如guard，任意transition,你都可以看if v=1,if v=2, if v=3

比如action,v’=v+1



guard可以理解成if ,if in the guard g,event发生了，那么q到q’，并做出对应action a

e是发生的事件属于alphabet1，g是一个叫做guard的状态，a是对应的动作，在状态q’中被改变的变量x叫做x’

一个transition是两个状态之间的路径，整个系统从source state改变到target state

在这个转变过程中，有可能有一个动作

EFSM的state diagram: efsm的绘图形式

最简单的EFSM只包含state, event, transition

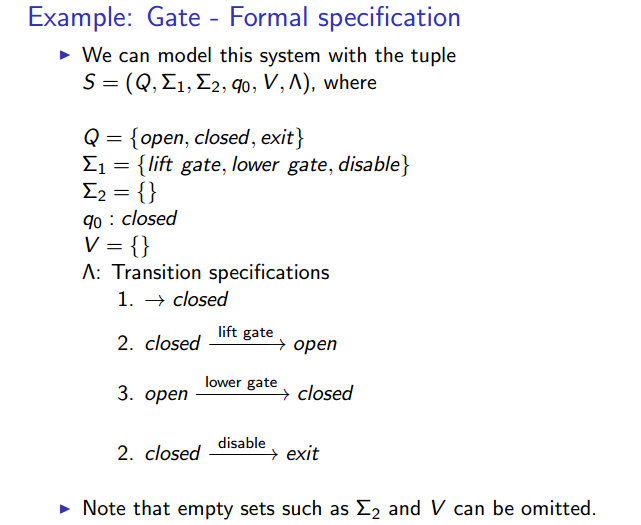
例如：gate

假设有一个大门,可以打开或关闭

当一个车靠近大门的时候，一个传感器sensor会产生一个信号，lift gate抬高闸门，让gate打开

当离开这个gate的时候，另一个传感器会产生信号lower gate，让gate下降

当gate关闭的时候，我们可以发送合适的信息让system终止



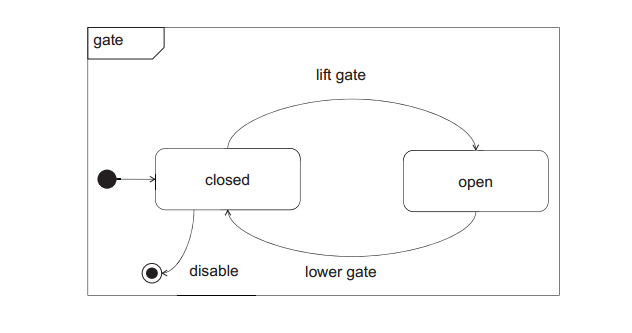
Q指的是整个系统可以进入的状态//不能包含动词，打开显然不是状态

event是对这个系统发生的事，这个门系统被发生了升高，降低等事

q0就是初始状态，

空集可以被省略

最简单的这个系统是没有action的，closed状态经过lift gate event进入open



state写在圆角长方形里虚拟

小黑点代表pseudostate（就是原来的start现在在箭头左侧加了一个初始虚拟状态） 它会自动transition到初始状态的closed

一共有四种情况会导致transition的发生

1.call event，系统外的一个请求激活了一个操作//车子进出激活了系统升降，升降本身是event

2.change event:如果transition的值变成true//假设closed到open之间有个boolean值，如果true，transition

3.signal event：内部或外部有一个特殊的计时器，时间事件用的是关键词at//比如说，at九点我们进行这个transition

4.time event：transition之前的source state如果被激活了一段指定时间，guard会被评估然后如果guard为true，那么就会transition,如果没有guard，那么就直接transition，一个time event用的关键词是after

Guard:

guard提供了transition能发生的基础条件

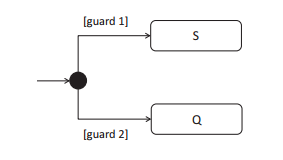
guard由系统评估

psedostate：

psedostate用来连接transition或者指向transition。

initial: psedostate也用来标记起始点，这个起始点会连向这个系统的默认起始状态

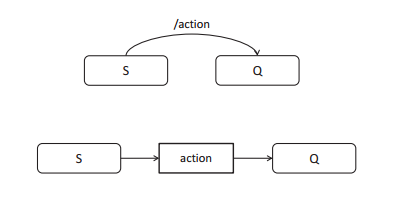
如果有多个起始状态，我们应该用guard来决定到底使用哪个默认起始状态。



actions：

transition可以用action连接起来

在event后面用斜杠表示，或者用一个纯粹方框来表示，action由system来表示



例子：bounded buffer//有限缓冲区

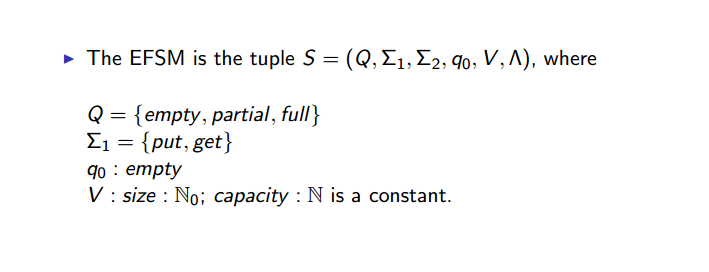
假设有一个bounded buffer，他的容积大于1

他有三种状态，空，有东西，满

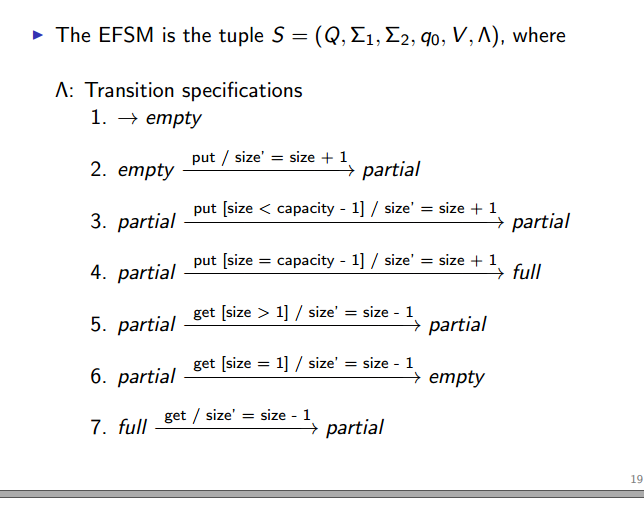
如果buffer不是满，可以往里面加东西

如果不是空，可以从里面拿东西

我们用event put 与get对应这些个操作

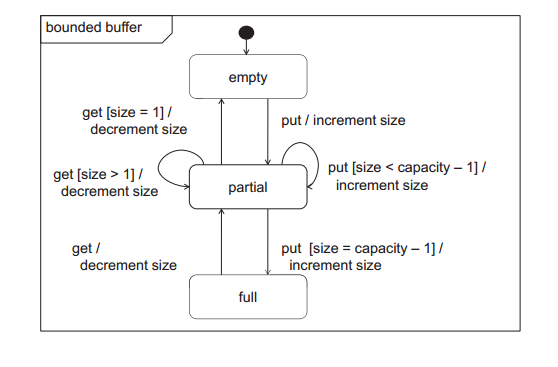


这个系统本身的状态是Q，empty partial full, 动作是put get，//怎么判断，首先，是动词，经过这个动词，Q之间状态会发生改变，V就是size，也就是alphabet。N0是最大容量，常数



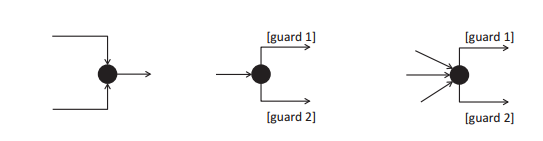
action是event发生以后，system对应的变化

方框里是guard



Pseudostates之junction

junction：可以是1 to many, many to 1, many to many



1 to many:用来把多个transition收敛成一个transition//merge

2. one to many:通过不同的guard，把一个transition转换成了多个trainsition//有点像case

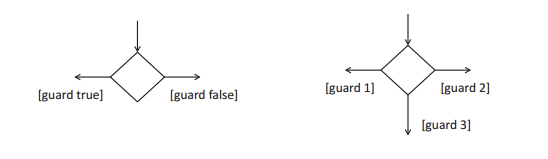
3.multiple，收敛又guard

Pseudostates之choice，

Choice：只接受一个incoming transition，生成两个或多个transition,每一个transition有一个guard，只有一个是true

如果没有一个guard是对的，那么这个choice是不合格的，所以我们需要一个transition来表示均非情况

如果说guard的判断都是对一个变量进行的，那么我们把这个变量放在菱形框里



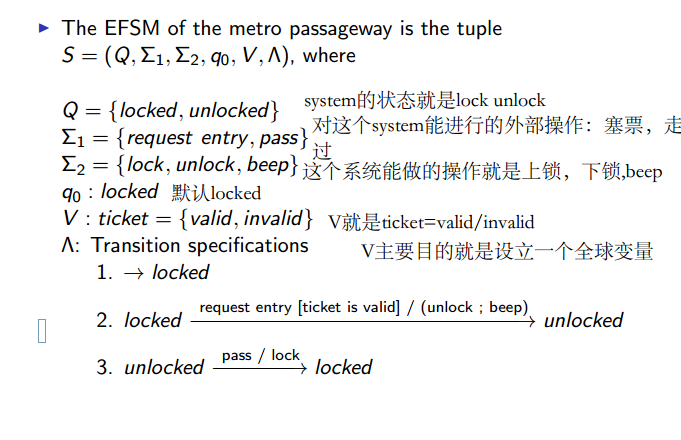
例如metro passageway-descrption

假设有一个的地铁通道//检票的东西，只有两个状态，可通行，不可通行

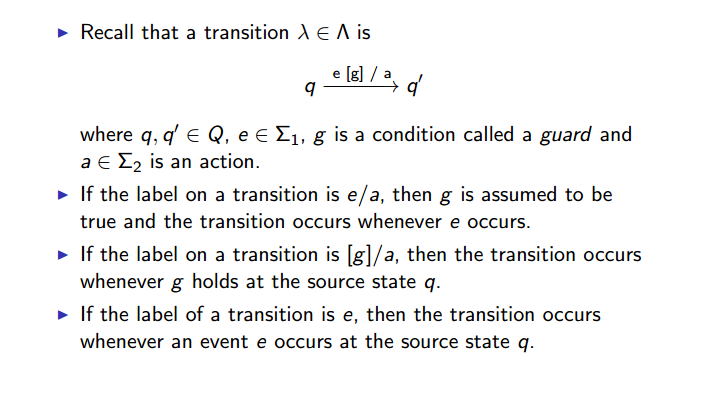
如果不可提通行，旅客输入一张票

这会让地铁通道由state不可通行到可通行，同一时间，这个通道会进行两个对应操作1.让大门unlock2.他会发出响声，提醒有人通过

当一个人通过，传感器会检测到，并产生一个transition，回到不可通行 状态，并且同时进行一个lock action



现在我们来回顾一下



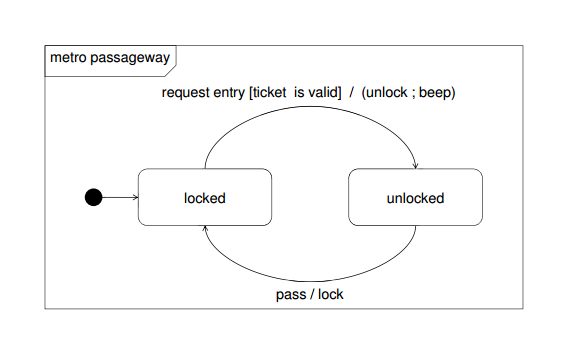
g就是guard，if

a就是整个system可以自主进行的操作

如果有e/a，那么我们就默认guard是true

如果有[g]/a，那么满足这个guard，就会transition

如果只有e，那么外力对他作出动作e，transition



hierarchically nested state//分层nested状态

基于UML的EFSM引入了概念:hierarchically nested states。

换句话说，一个state本身可以被设置成一个EFSM并且包含他自己的state叫做substate (或nested state)

包含其他state的state叫做composite state//复合state，不包含的叫做simple state

我们因此有了新关系superstate/substate

direct substate：如果一个substate不被另外的state所包含

transitively nested substate:与direct substate相对，被两个或两个以上的state包含

substate继承了他们superstate的transition

如果一个系统在substate中，那么他们也在superstate中

例子：

safe保险箱

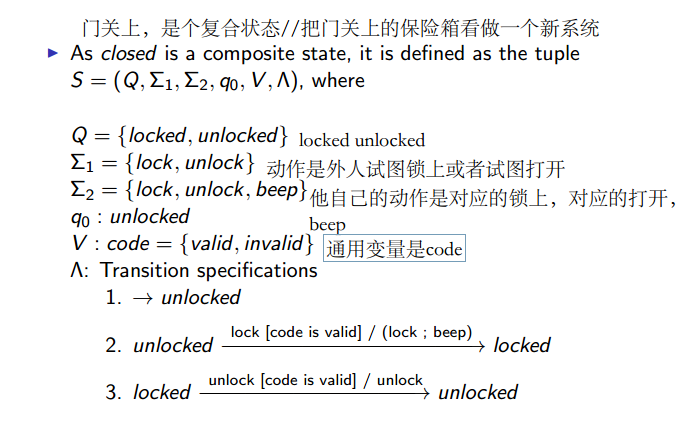
如果存在一个safe。他的门可以是open或者关上

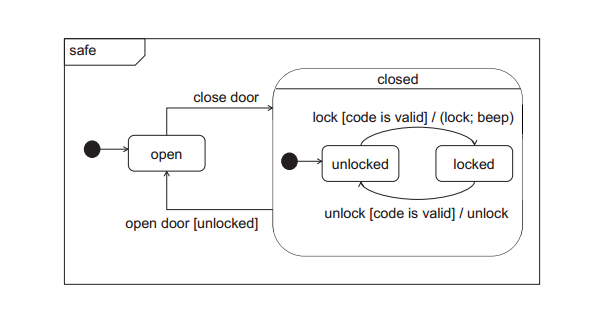
当门关上的时候，这个safe可以是unlocked 状态，locked状态

当这个safe是unlocked状态并且门关上的时候，用户可以输入密码让这个safe自己lock

lock的时候，safe会响一声

一旦safe locked了，person可以输入密码。让这个safe unlock

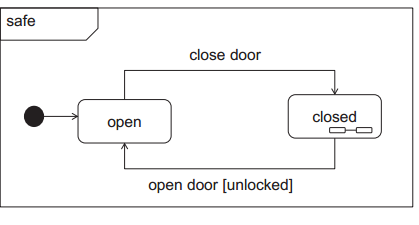




我们可以看见closed状态自己也被看做一个EFSM，他有自己的state与starting state

unlocked 与locked是closed的direct substate

composite state可以表示称simple state，但是要加一个icon复合图标，用来表示他的内部结构没有在当前图示中被画出来



state behavior

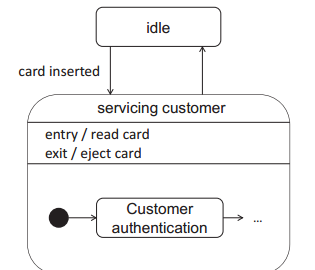
一个state可以有他自己的behavior

entry behavior：进入这个state的时候，进行的behavvior

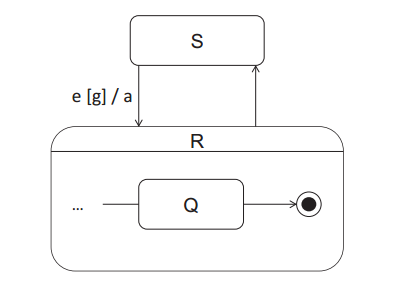
exit behavior

do behavior：在这个state的时候

比如说进入客服，会read，离开，弹出卡片



有离开state的substate，当进入到离开state，直接离开整个substate并沿着transition进入另外一个state



例子：alarm

假设有两个状态，active/idle

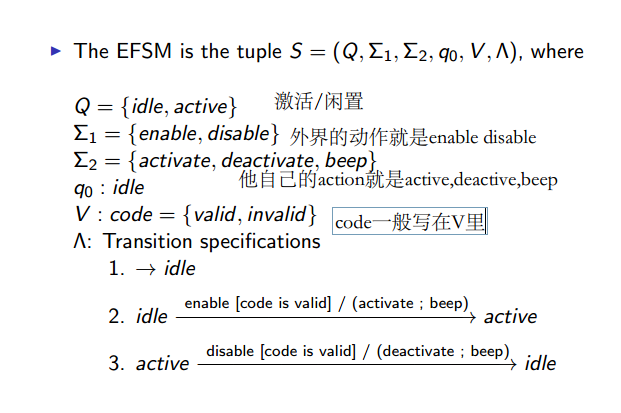
当idle时，输入密码，可以激活

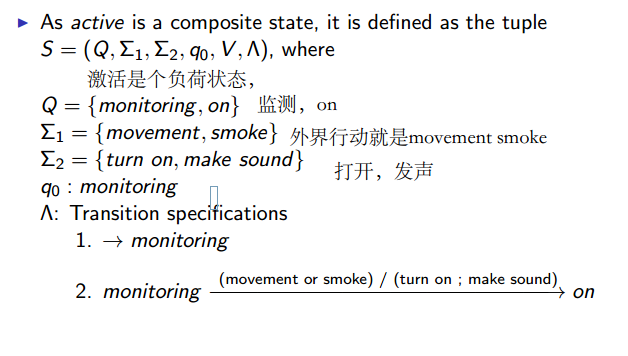
从idle到激活的这个transition会生成两个动作，activation/BEEP

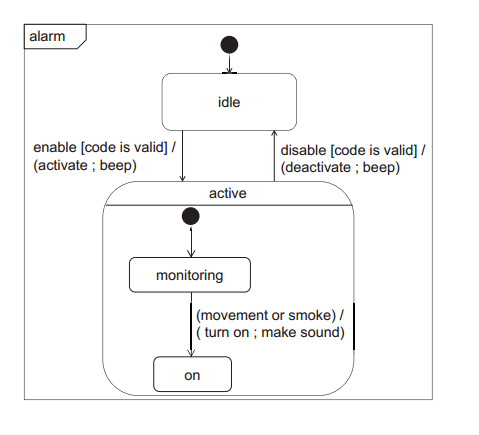
在active时，系统可以分成两个子状态

1.monintoring （没发生情况，但是保持监测） 2.on

在monitoring的时候，如果系统的传感器检测到movement/smoke,系统会进入到on state，同时进行两个操作,turning on.发出声音



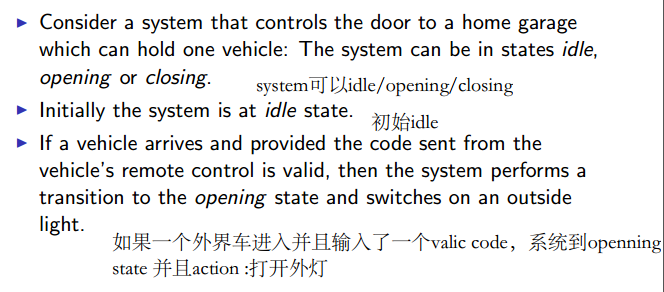


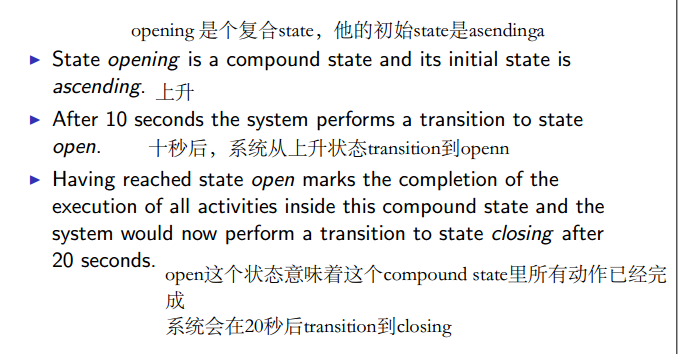
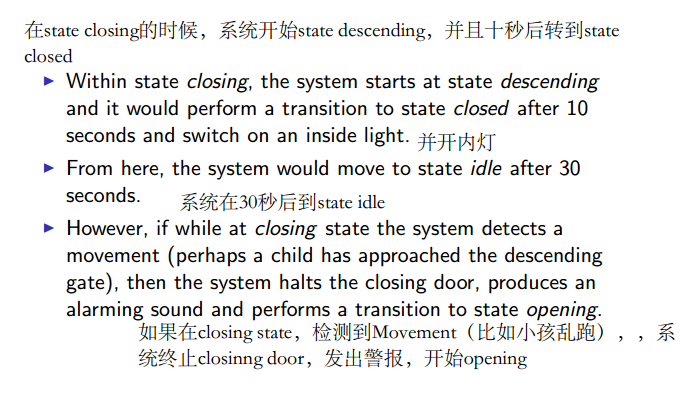


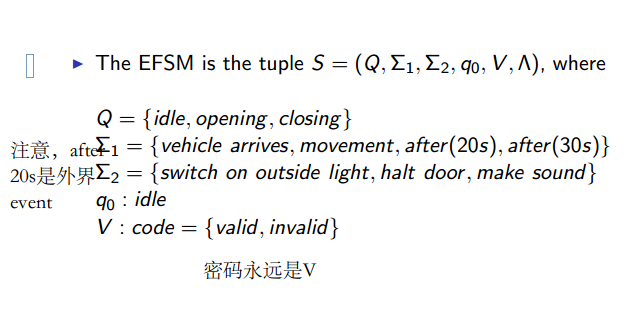
注意：无论在monitoring 或者on状态，只要输入了密码，都会idle，因为substate继承了主state的transition

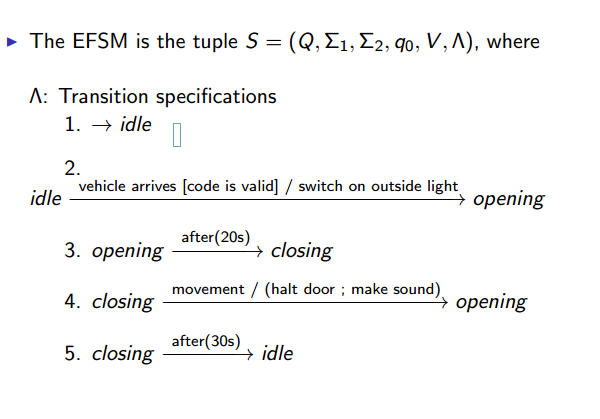
transition继承关系化简了图片，如果没有composite state，我们必须化两个transition:monitoring->idle/on->idle

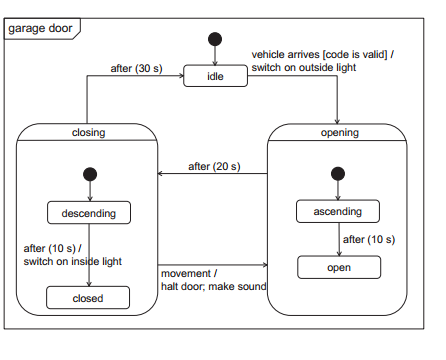
example:Garage door汽车库门

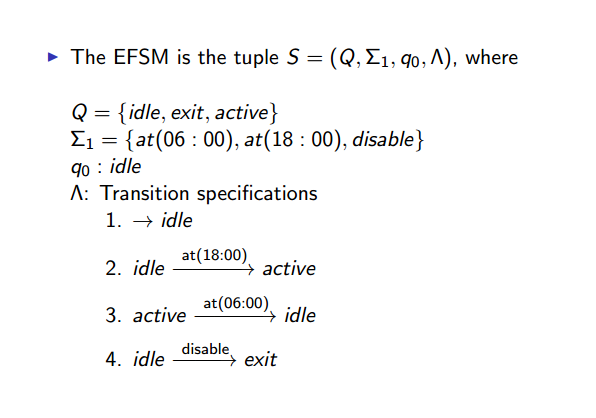


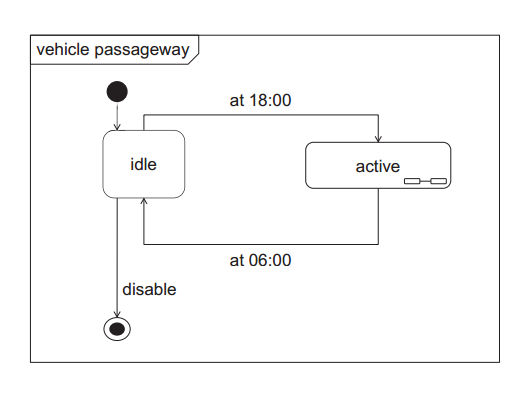




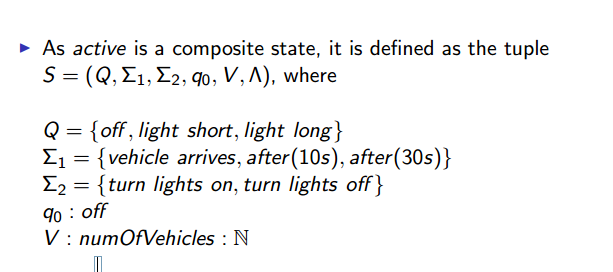




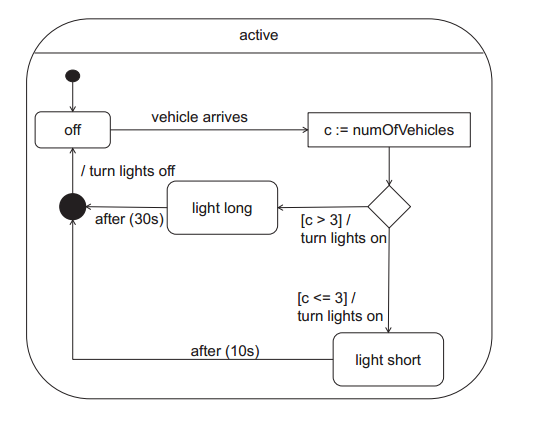
at after都是第一类，外界event



注意idle和disable是两个不同的状态，idle是state之一，而diisable是死去

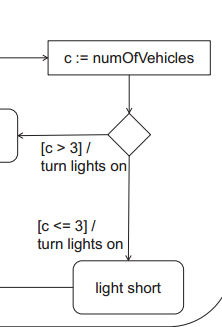


数量永远是V

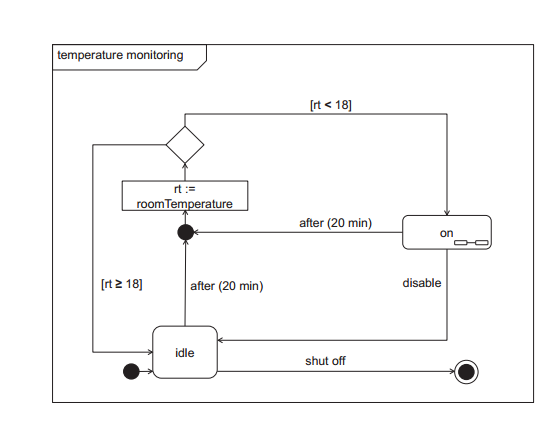


两个transition合并，就要用persedue， 判断不同的if (case)，就要用菱形

这里的非圆角方框意思是action，如果说我们要给一个等待菱形的变量初步赋值，就用这个结构



‘

’