实验一

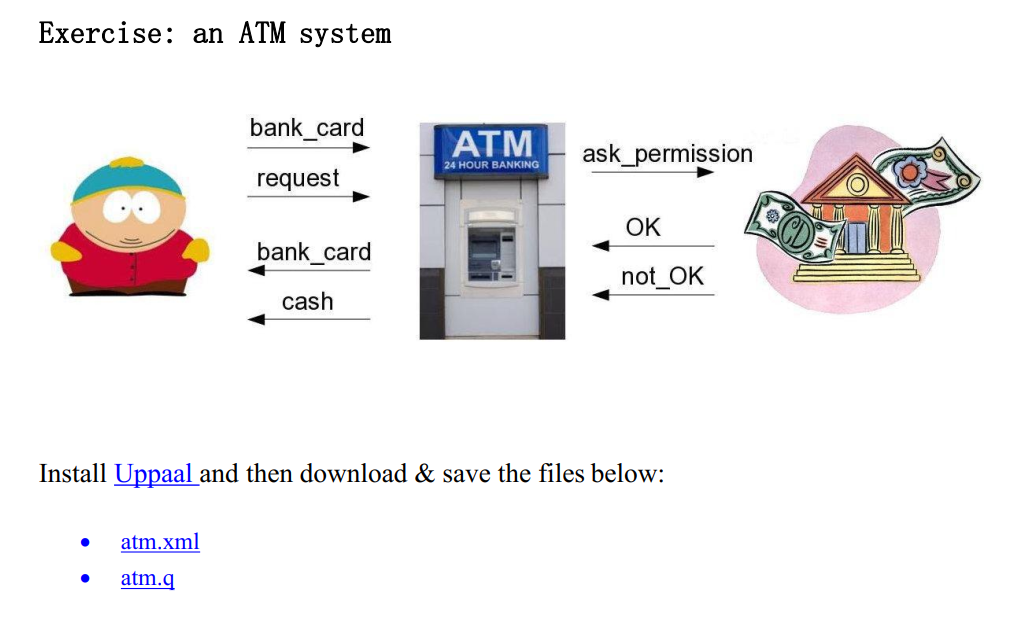
使用Uppaal进行建模和分析

下面是使用模型检查器Uppaal进行建模和分析的练习。

Uppaal是免费下载的，非常易于安装和使用。可以做练习短时间后由没有任何先验知识的高中或本科生说明什么是状态图，以及演示基本内容的演示Uppaal的功能。 （说明应涵盖Uppaal的符号？和！作为输入并在通道上输出动作（死锁的概念），说明如何编辑状态图，并显示模拟器和验证程序，包括验证程序如何产生您可以在模拟器中逐步执行的反例。）

模型检查

模型检查是一种可用于设计和分析的技术动态系统。模型检查器是一种快速运行的计算机程序并巧妙地搜索系统的所有可能状态以寻找解决问题。实际上，您正在搜寻潜在问题。模型检查器不适用于实际系统，但可以用于系统模型-由此得名。 Uppaal使用状态图作为模型。除了图描述我们要分析的系统，我们还需要描述一些期望的系统属性。这些属性称为查询。



使用File：Open System在Uppaal：atm.xlm中打开这些文件，并使用File：Import Queries打开atm.q。

文件atm.xml提供了一个ATM（自动柜员机或现金点），客户和银行。为简单起见，银行只有一台ATM和一名客户。那里名为Eric的客户在他的电子钱包中携带Eric.cash\_in\_pocket欧元。自动取款机具有ATM.in\_till欧元兑换。银行是银行的后端系统，并跟踪Eric的

Bank.balance中的余额。

埃里克（Eric）与自动取款机（ATM）互动，从机器中提取现金。 ATM反过来进行通信

在银行，以确保银行正确跟踪Eric银行的余额帐户。

最初，Eric口袋里没有现金。您可以通过单击左侧的系统在Uppaal中进行检查Eric，然后单击“声明”。埃里克（Eric）的银行帐户中有80欧元的余额，而且ATM的收款机里有200欧元。文件atm.q表示系统的两个属性：即Eric始终拥有80欧元（因为该模型不包括他花钱的可能性），并且该系统不应该陷入僵局。

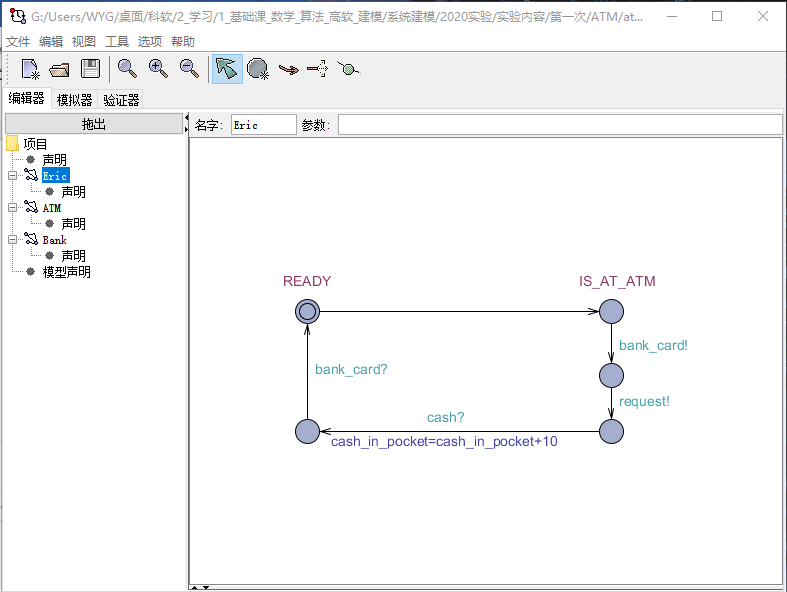
问题1：使用验证程序检查属性。尝试改善模型，直到两者属性是正确的。对于第一个物业，您必须更换银行。对于第二个属性您必须更改Eric。 （如果ATM不支付现金但确实退款，Eric应该怎么做？他的银行卡？）

提示：在菜单选项：诊断跟踪下，选择选项最短以使验证程序在模拟器中生成反例。

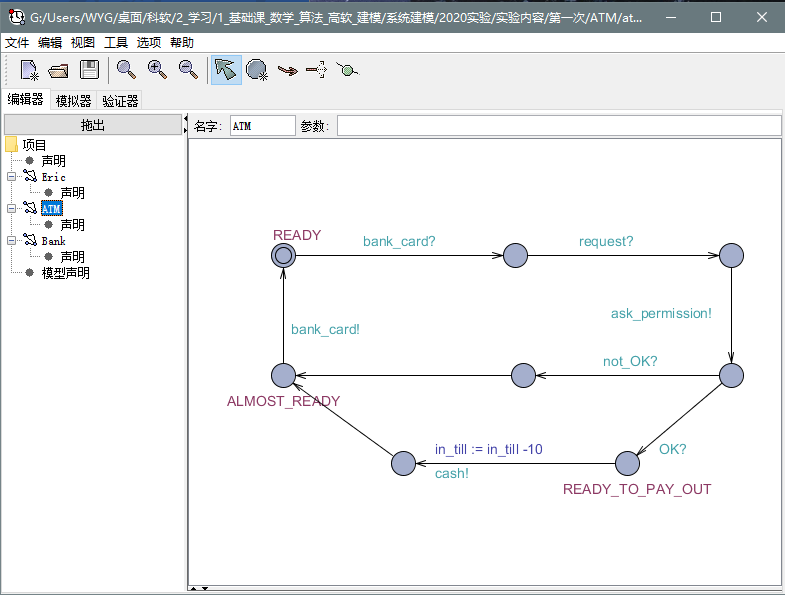
问题2：更改自动取款机，使其最初只有30欧元，而不是200欧元。自动柜员机模型中仍然存在一些不切实际的问题：直到模型中可能发生某些事情，这在现实中是不可能的。尝试

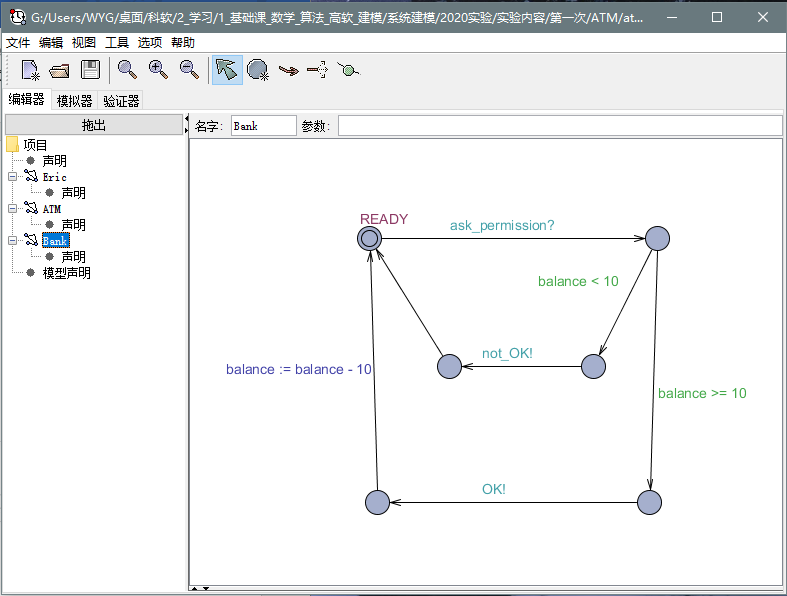
弄清楚这是什么。您可以通过模拟模型一段时间或让Uppaal做到这一点做一个随机模拟。 （提示：埃里克（Eric）想购物，需要大量现金。）更改自动柜员机的型号以改善此情况。完成此操作后，将一个查询添加到验证器以检查是否已避免了该问题。查询应表示对ATM的明显的现实检查。当然，适应模型仍然不应该陷入僵局！

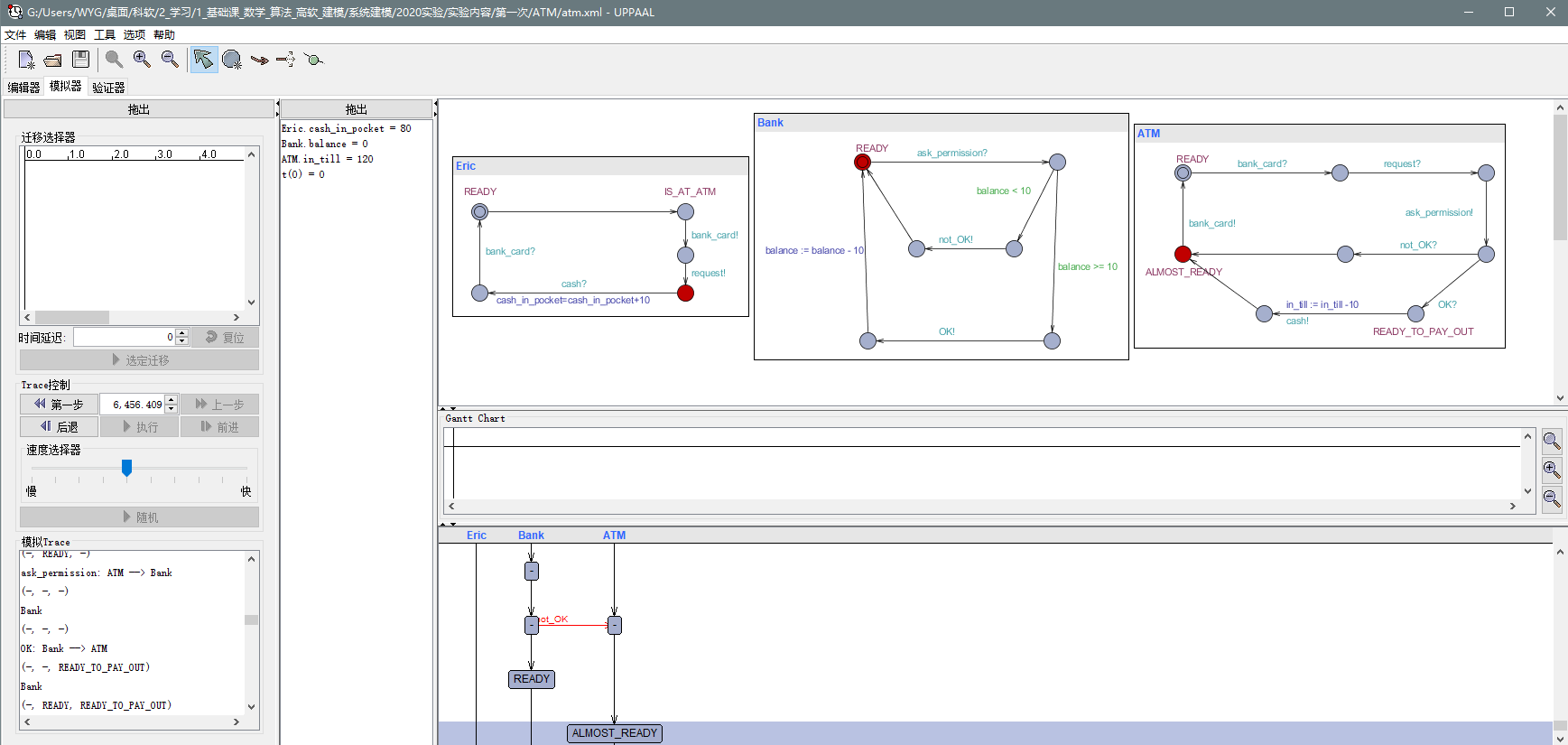
原始xml：



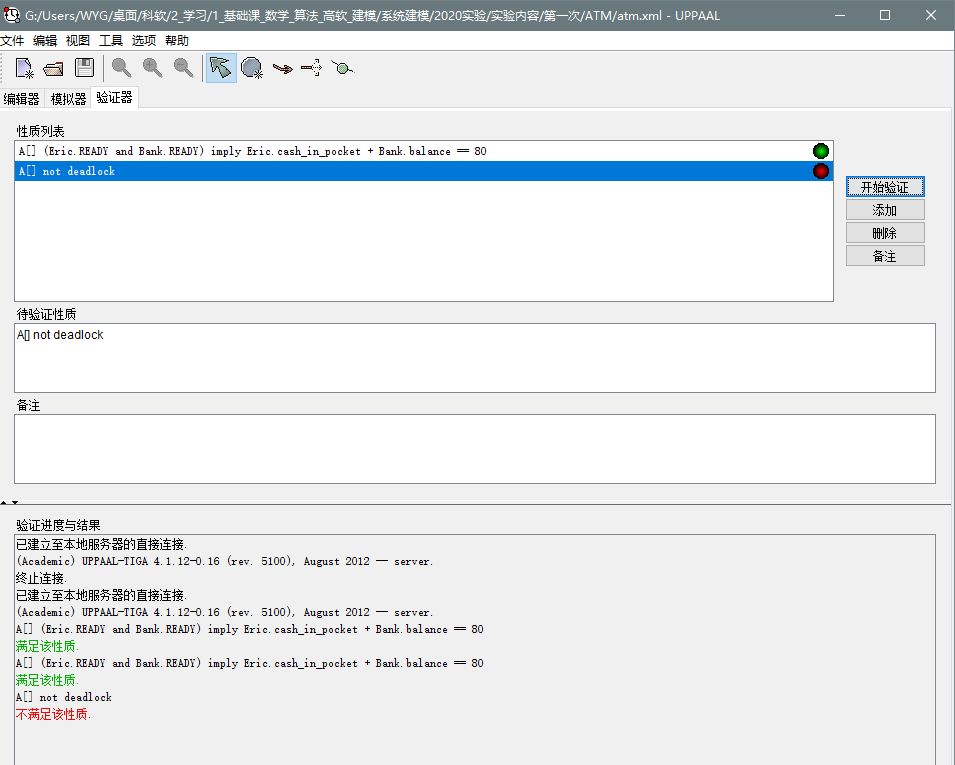
可以看出了eric的状态图中缺少银行钱不够的信号状态，可能产生死锁，后面进一步修改。



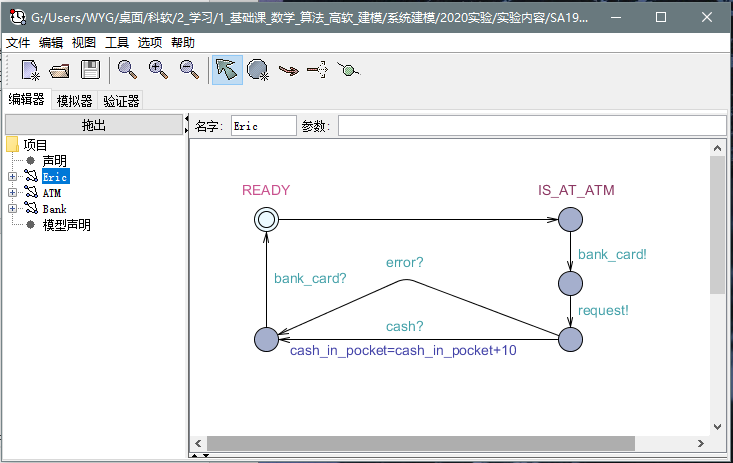




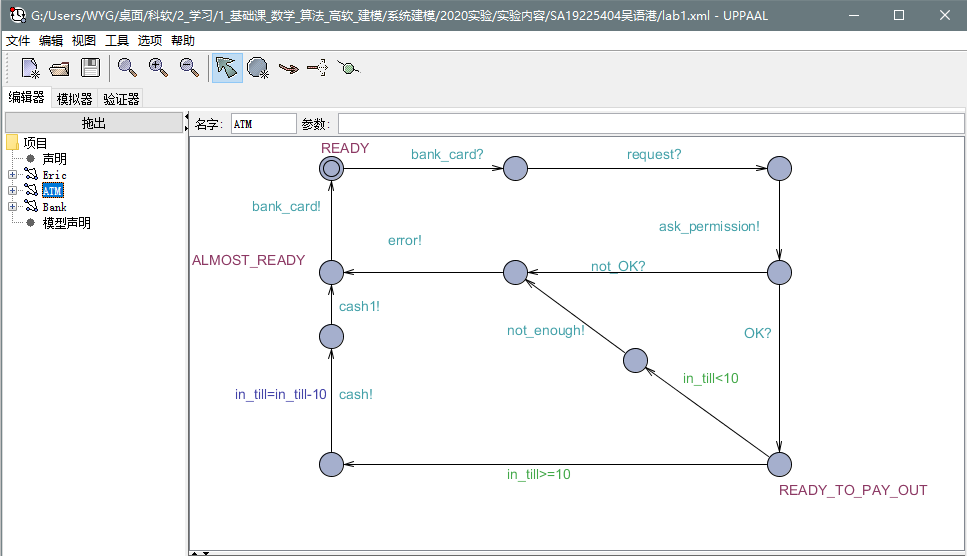
ATM的死锁问题：要理解实验一的原模型是Eric每次取10块钱，在收到取钱请求后，Bank会把Eric要取钱的数目与Eric账户的余额进行比较，如果超支，则会返回not\_Ok!信号，然后Bank回到起点状态。同时ATM会收到这个信号进行转换状态（退卡），然后Atm回到起点状态。但是Eric将会一直停留在等待cash信号的这个状态，无法回到起点状态，所以造成了死锁。

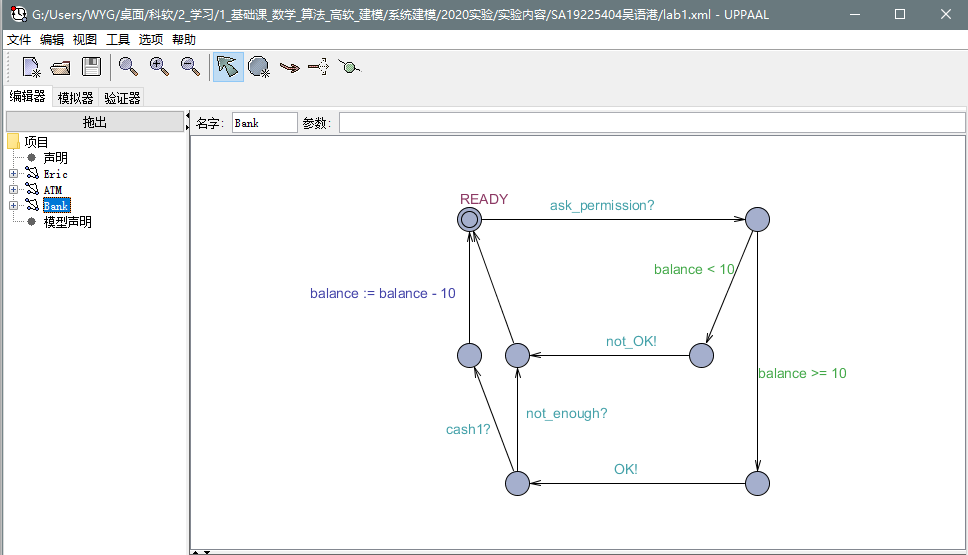


改进

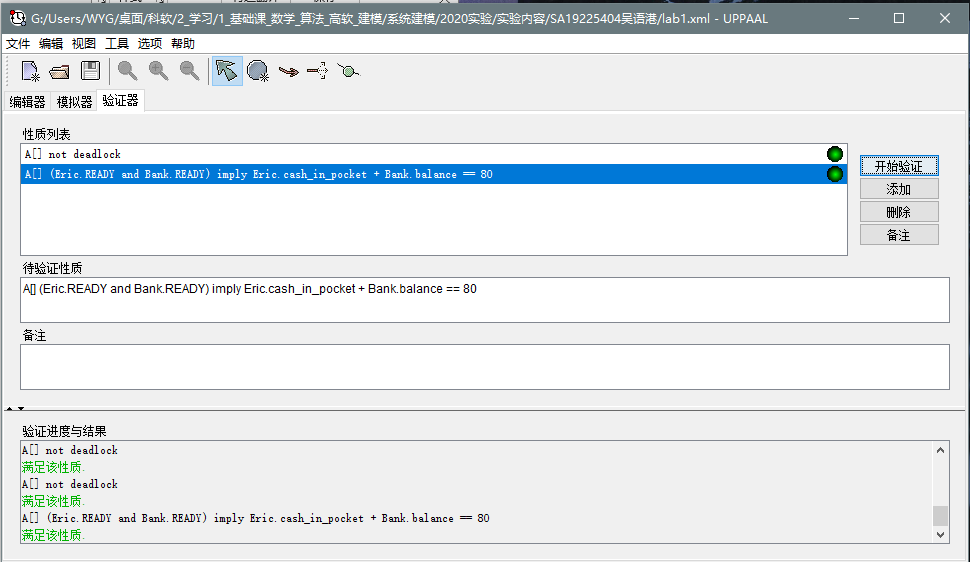


Eric加入了error信号又来防止eric状态死锁

atm在in\_till：=in\_till-10的前面增加了atm机剩余现金够不够的判断



Bank状态图也添加了够不够的支路，和not\_OK汇集一路



这样就解决了这个死锁的问题了

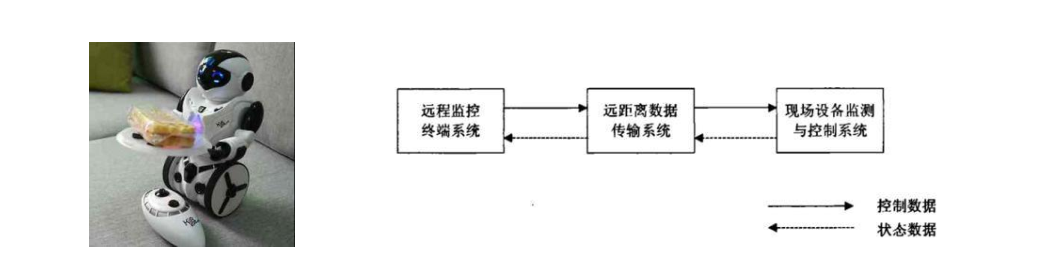
实验二

Telerobot控制系统》

—建模与分析

我们设计了Telerobot Control（TC）系统的简单版本。 TC系统是远程控制的机器人。 它由两个主要部分组成：传感器手套和远程RobotArm。 同时，它也包括这两个参数之间的网络连接。 感应手套将进行准确的测量，然后将传感器数据发送到机械臂，并使用传感器手套执行相同的操作。 TC系统在危险环境中非常有用，因为操作员可以在院子外解散炸弹或炸药。处理危险化学品。 为了发挥作用，机器人必须对机器人的手部动作做出响应

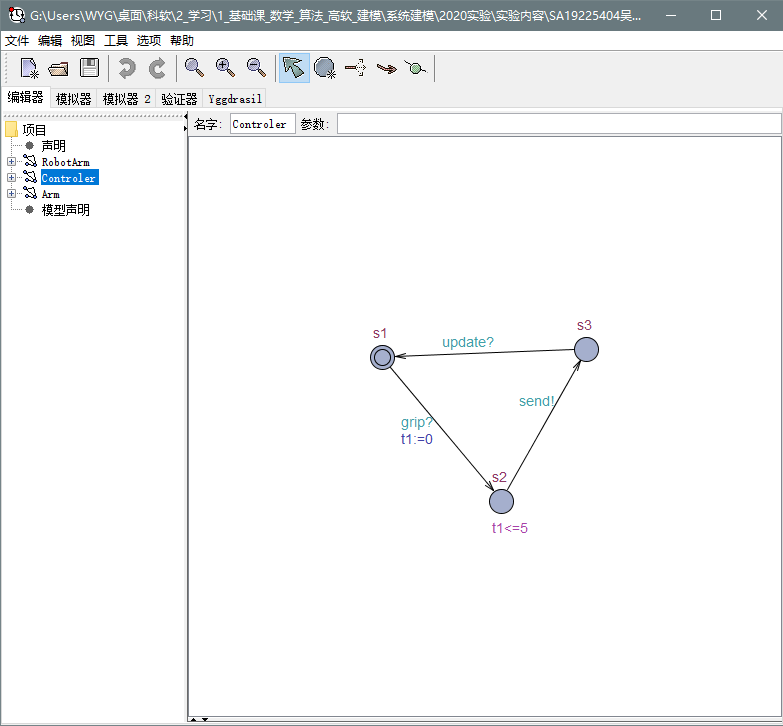
传感器手套足够快地利用用户固有的电机控制系统。 如果机器人响应太慢，用户将反应过度，从而形成不稳定的控制环。 同样，控制如果机器人缓慢提供触觉或视觉反馈，循环可能会变得不稳定



TC系统分为三个部分，可以看作三个组成部分。

手臂组件具有抓握功能，可从传感器获取手套的力量，并且将抓握命令发送给控制器。

另一个组件是名为Controler的站，其中包含以下操作为执行相应的行为而触发的事件：Controler组件将接受手臂发出的抓地力命令，“发送”动作会将控制指令发送到远程机器人 另一个操作是“更新”，它将在收到更新命令。



另一个组件是远程机器人：该组件将使用以下命令接收send命令网络协议栈； 然后“更新”操作会将更新命令发送回给控制器。是RobotArm and Arm的组件设计吗？S是sse.ustc.edu.cn（中科大软件学院）《系统建模与分析》TeleRobot控制系统的时间和功能属性指定如下：

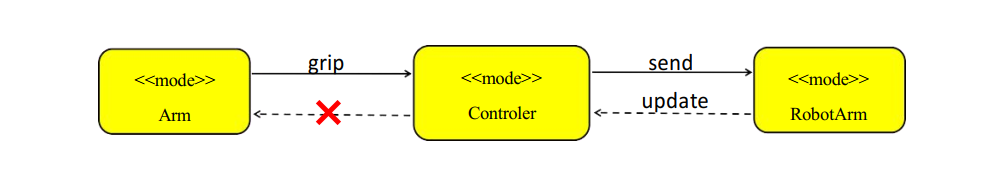
R1：抓地力周期为10ms（Arm）；

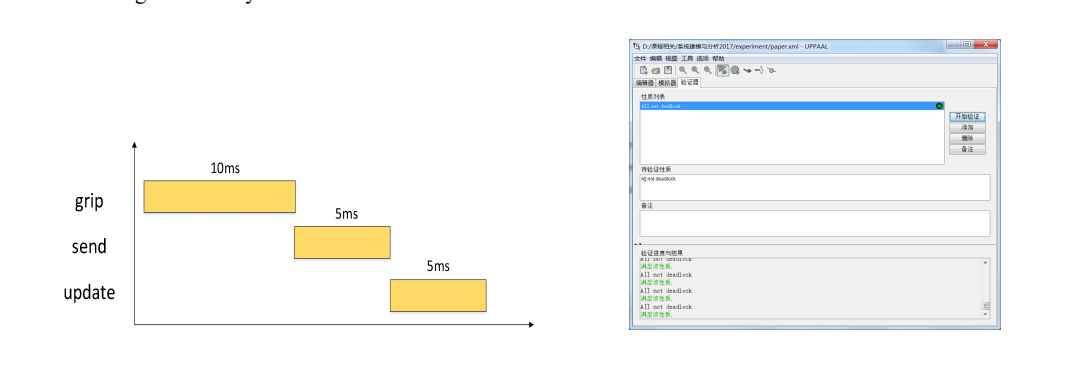
R2：发送后总是紧握（控制器）；

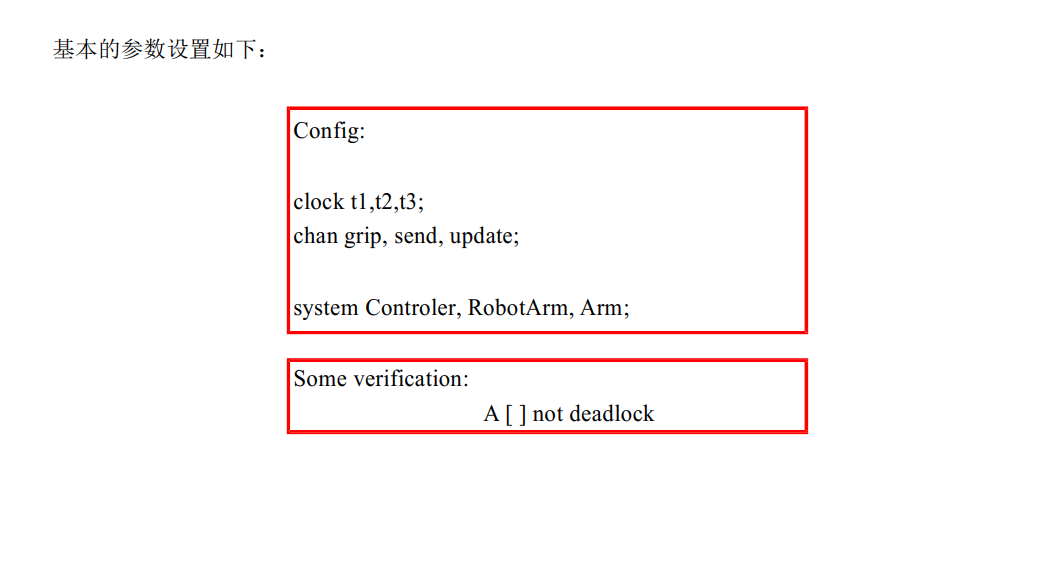
R3：更新的行为总是跟着发送（控制）；

R4：RobotArm将接收发送命令（RobotArm）；

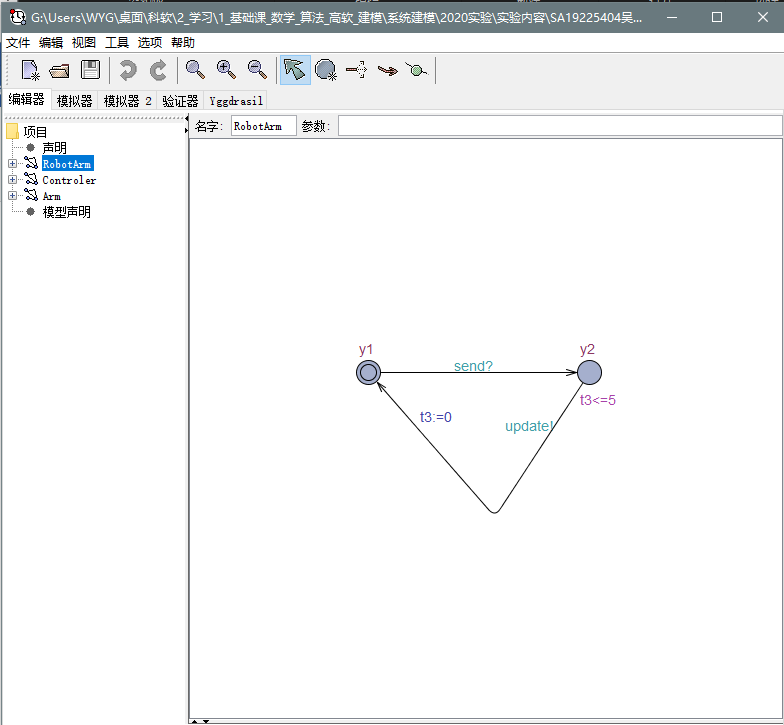
在RobotArm收到发送命令（RobotArm）后，R6：update总是在5毫秒内发送。



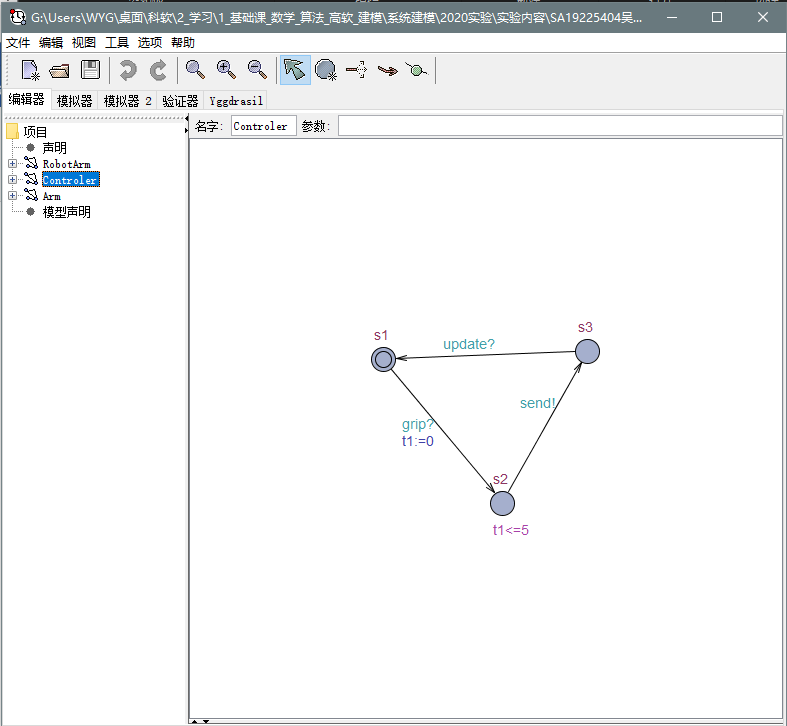
TC系统的时间预算：  




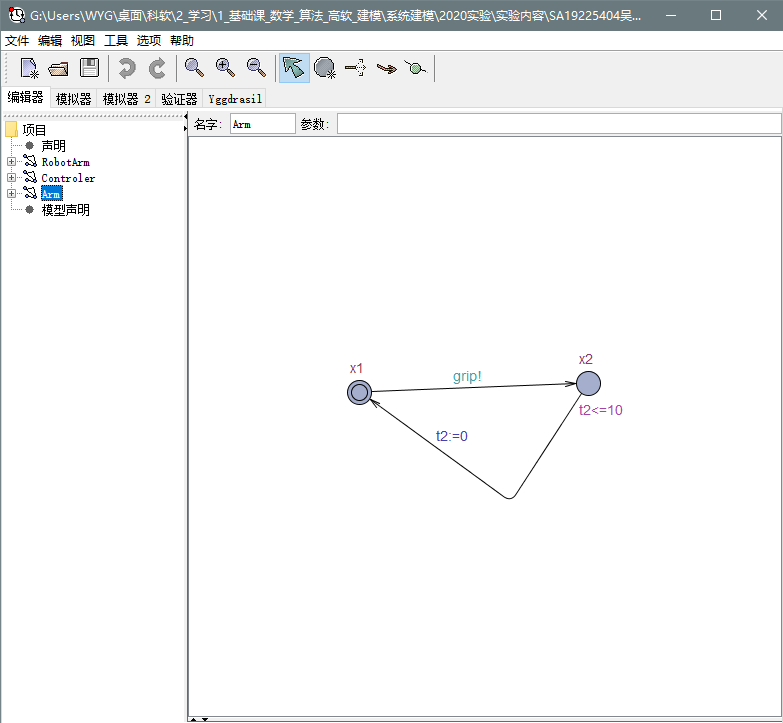
实操：



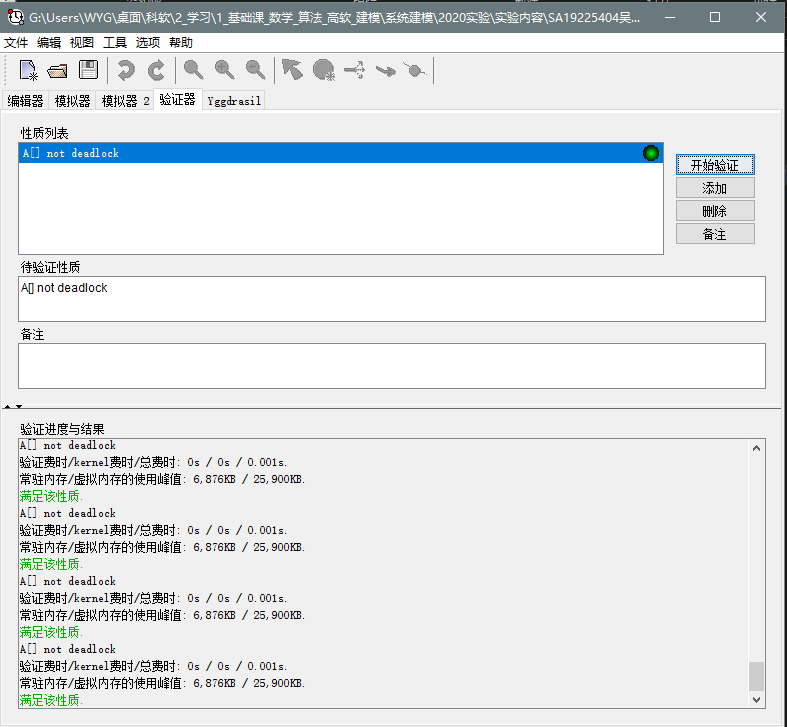
Update 5ms



Send 5ms



Grip 10ms



实验三

实验目的：

理解UPPAAL中的位置不变量（invariants）和约束（guards）；

理解UPPAAL中的验证查询语言（query language）;

图中

双圈点表示Initial表示初始状态

圈内C点表示committed在这个状态点不会逗留

蓝色是update 更新

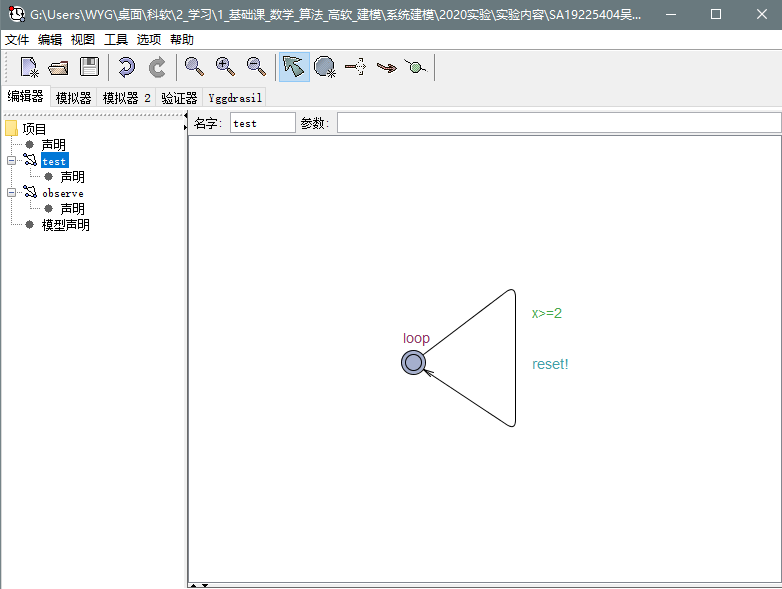
紫色是invariant 位置不变量

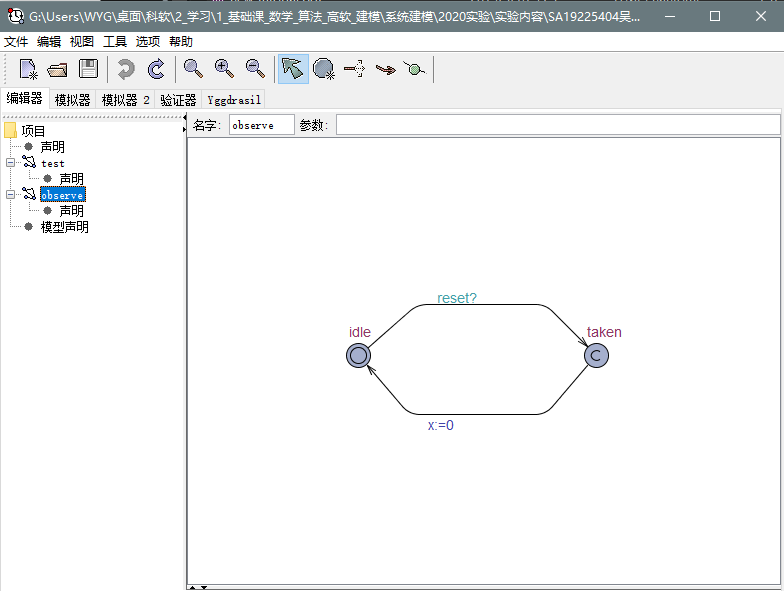
暗红是location 状态名称

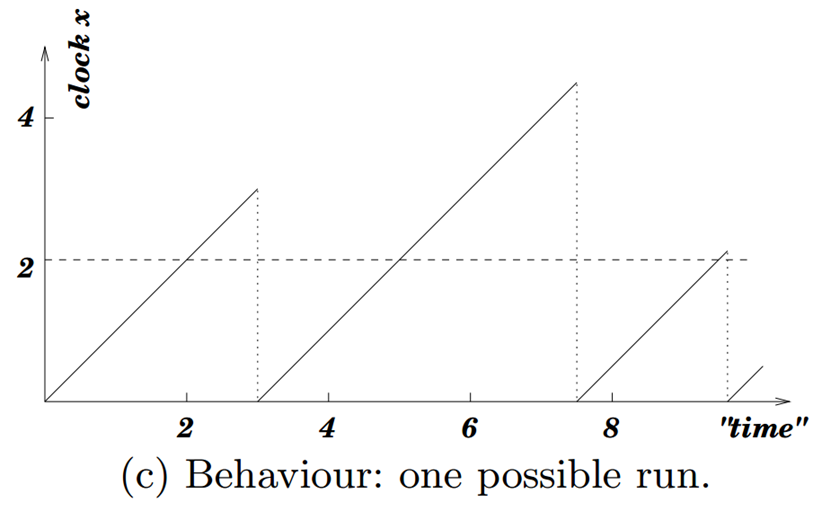
蓝绿是synchronisation发射接收同步信号

绿色是guard

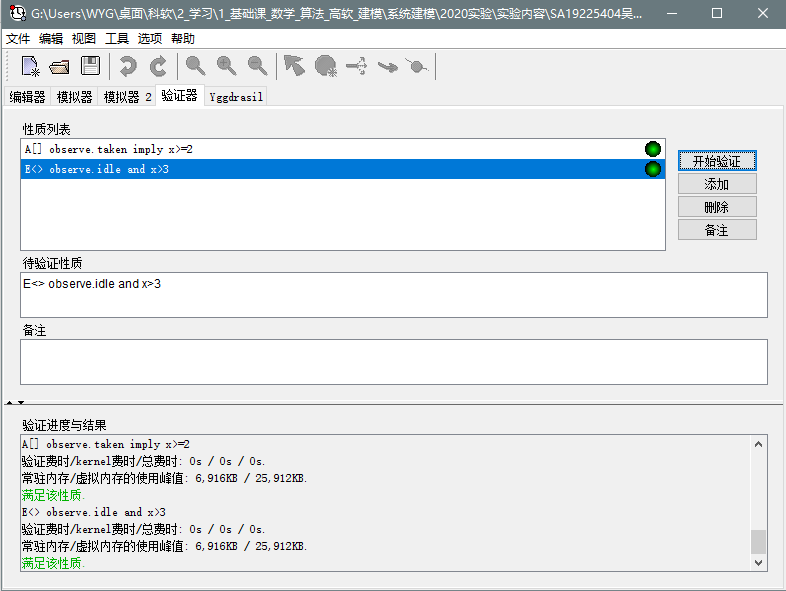
位置不变量不存在时：





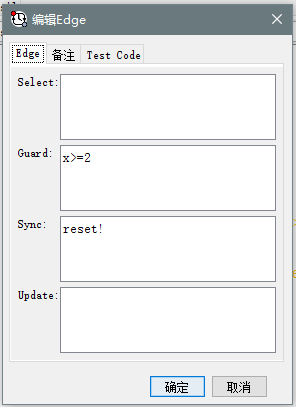
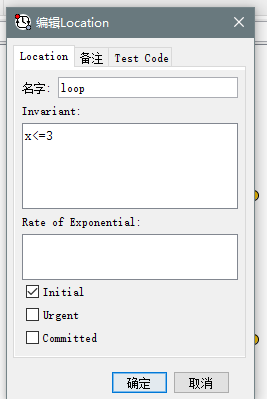


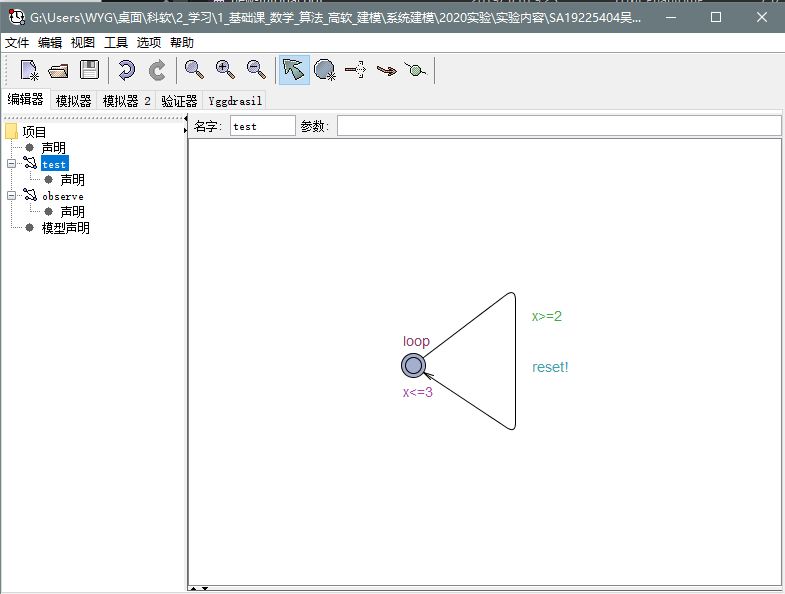
验证查询语言

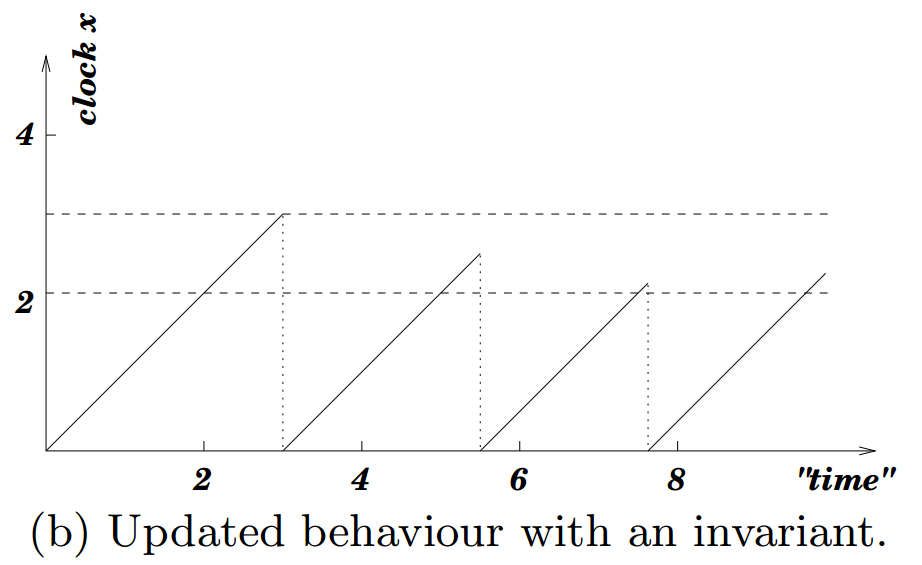


两个性质均满足

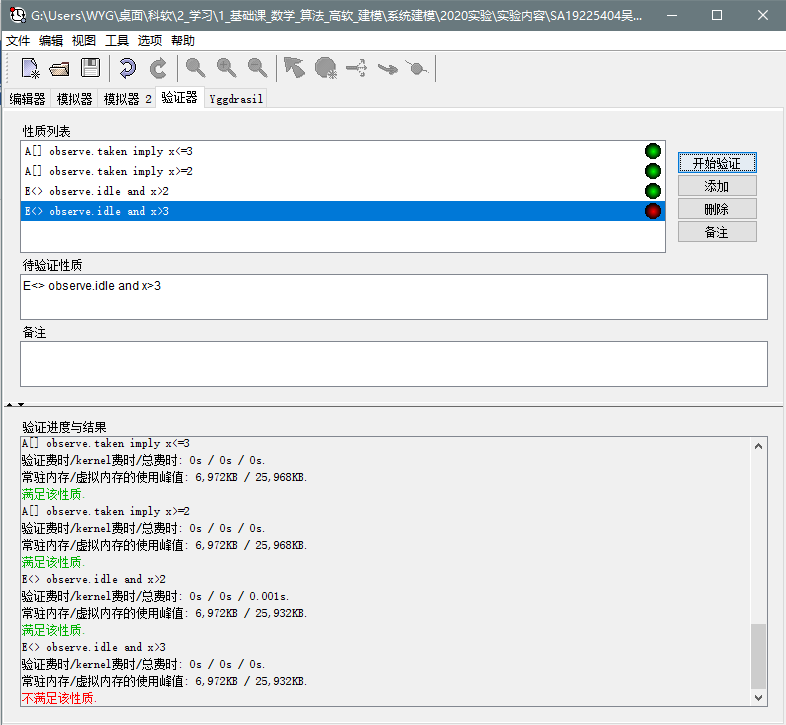
添加位置不变量之后



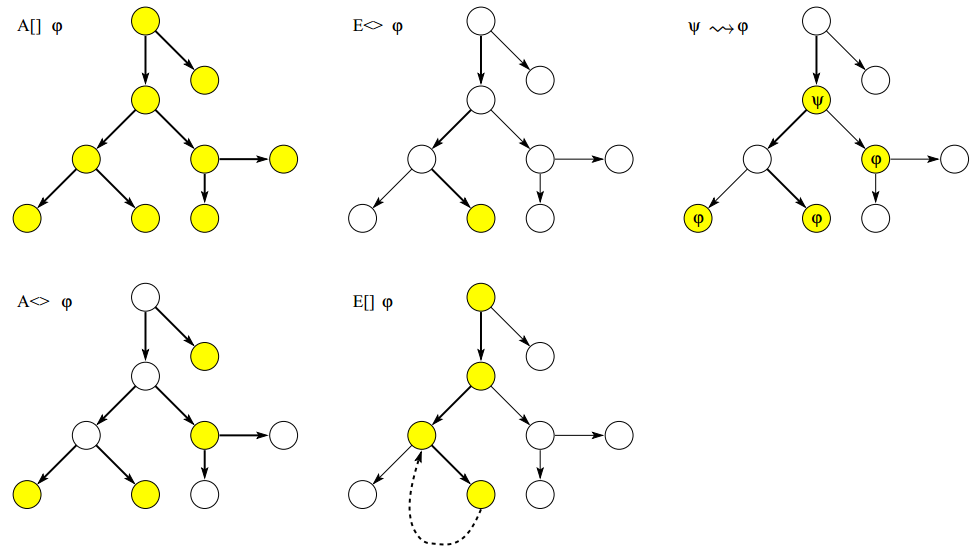


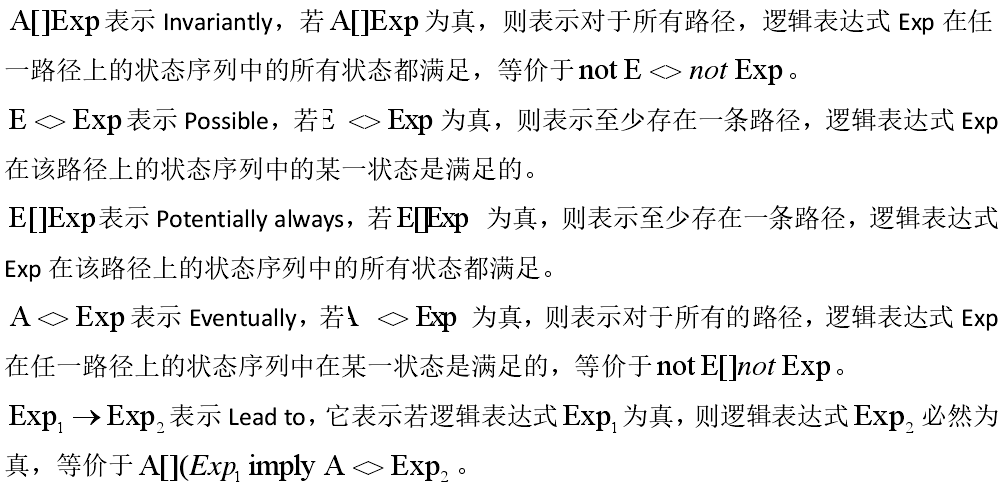


验证查询语言修改



验证查询语言学习 [ ] 每次 <> 某一次 A所有路径E某一路径





请针对“new-tutorial.pdf”中的fig.5，fig.6，fig.7进行建模实现，并结合具体的验证结果，理解位置不变量的含义。

首先要明白，系统每次运行时都有可能走不同的路径，并且这些路径运行的次数不止一次，会反复运行，反复循环

A[] Observer.idle imply x<=3

可以理解为：对于Observer系统下的所有路径，当该路径到达idle状态的时候，时钟x的值如果都是小于或者等于3的话，那么该式子为真

A<>Observer.idle and x>2

可以理解为：对于Observer系统下的所有路径，当所有的这些路径到达idle状态的时候，对于每个路径时钟x的值至少有一次大于2的话，那么该式子为真

E[] Observer.idle and x>2

可以理解为：对于Observer系统下的某一条路径，对于该条特定路径而言，在每次该路径到达idle状态的时候，时钟x的值一定是大于2的，那么该式子为真

E<>Obs.idle and x>3

可以理解为：对于Obs系统下的某一条路径，当该路径到达idle状态的时候，时钟x的值至少有一次大于3的话，那么该式子为真