

假定CarLight角色所表示的是：

对fsm而言：









对应迁移YellowOff→YellowOn

对应迁移YellowOn→YellowOff

对应迁移Init→YellowOn

***FSMActor\_to\_Kripke:***

**原子命题集：**



**状态集：**

状态集S中的每个状态s是一个多元组：



其中每个原子的取值为0或1

**标记函数：**

L：S→S

对于，其初始状态是，所以关于的原子，除之外，其余都为0。

则初始状态是：1,0,0

**迁移关系R**

对于,若满足下述条件时，有

①状态s满足其中一个关于的原子，即

∃1 i, ，s.t. 在 s中,有

②状态满足其中一个关于的原子，即

∃1 ,，s.t. 在 中,有

③状态s中和中所有变量都各自满足其中一个关于其取值的原子：

,∃1 k,,s.t.

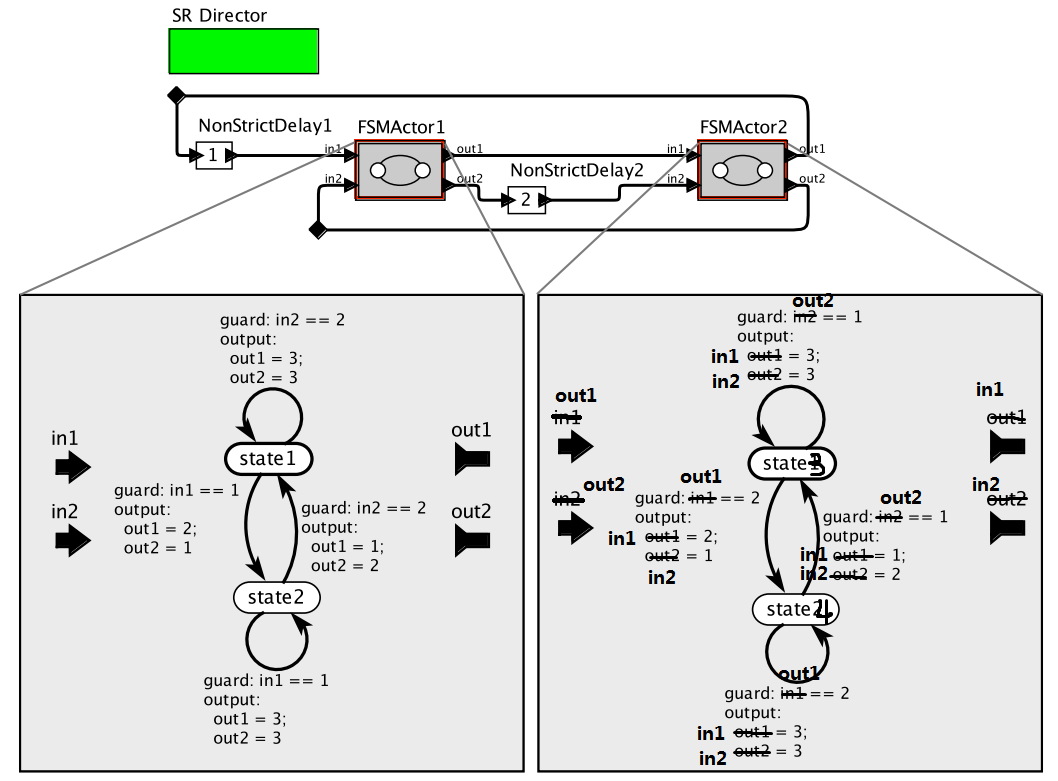
在s中,有；

在中，有,④中存在符合下述条件的迁移，即:  
 (a)



(b) ,





假定和角色所表示的分别是：

与

对于fsm1而言：









对应迁移state2→state1

对应迁移state2→state2

对应迁移state1→state2

对应迁移state1→state1

对于fsm2而言：









对应迁移state2→state1

对应迁移state2→state2

对应迁移state1→state2

对应迁移state1→state1

**原子命题集AP**







**状态集**





因此状态集中的每个状态s是一个多元组：



**标记函数L**



假设初始状态分别是state1，state2，in1=1,out2=2

初始状态是：1,0,1,0

**获取迁移关系R**

前提：相连接的两个端口名字相同

注：这里是in1,in2,out1,out2

对于若满足下述条件，有

①状态s满足其中一个关于和的原子，即

，s.t.在s中，有

②状态满足其中一个关于和的原子，即

，s.t.在s中，有

③对于状态s和都满足：状态机1和状态机2中的所有参数取值符合要求，即（这个例子里没有参数。。。。）



s.t.在s中，有在中，有

④和中存在符合下述条件的迁移，即



（a）

（b）



（c）并且

（d）并且

（e）of the textual format

PortName\_isPresent;

iff , a subexpression of , of the textual format PortName=IntegerValue;

状态迁移图没能想出来。。

2015\_2.2-

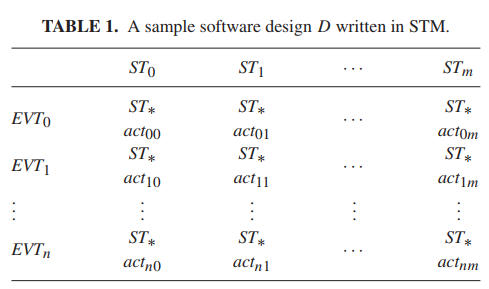


表1中用STM编写的示例设计D，表示系统D中的事件，表示系统D可以处于的状态（默认初始状态就是）。表示系统处于事件的状态下时将会执行的动作，并且会将状态切换到 (中的某些状态)。表示赋能条件，而表示执行（，）索引单元格的效果。

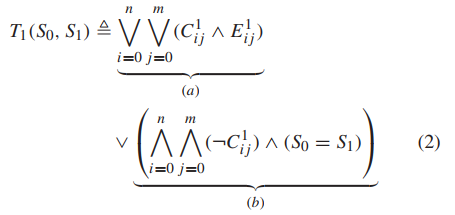
即使D系统中多个单元格被激活，其中只有某一单元格会被执行。系统D在同一时间点只能处于唯一的状态。

|  |
| --- |
| M\_IDLE M\_ACTIVE M\_WAIT\_BILL\_TAKEN |
| xMachinePrepare  xUserOperation  RequestProcessed  UserBillobatated |

MEM例子的表格

对于单元格(0,0)，赋能条件被定义为，其中ST\_mainInterface是一个额外的变量，用于表示STM MainInterface的当前活动状态。效果被定义为

时间步骤1下的迁移可以表示为：

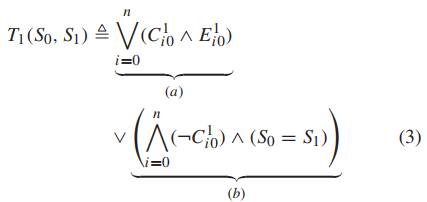


有单元格被激活执行时使用子式（a），没有单元格被激活时使用子式（b）。上标1表示当前时间步骤的状态变量(当前一步为1，前一步为0)。

例子MEM中单元格（0,0）的执行公式即为。

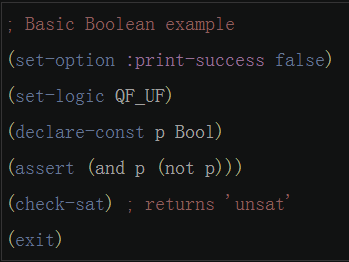
公式（2）的设计思路是，在给定的时间步骤中，所有的单元格（m\*n）都有被激活、被执行的可能。在这种思路下存在的问题就显而易见，在实际问题执行的过程中，公式（2）会变得无比巨大（m\*n）。

而回到问题本身，在时间步骤1时，只有作为初始状态的才有执行的资格。那么公式就会被大大化简（m\*n→n），此时我们只需要考虑状态下的n种事件即可，化简后的公式（3）为：



这只是时间步骤1下的化简，这种化简方式应用在其他时间步骤的困难是：不能确定系统D在某一时间步骤时所处的状态。为了解决这个问题我们可以通过在构造之前将SESE作为一个预处理来动态地解决这个问题，即利用SESE遍历M的状态空间，在每个时间步长中记住准确的迁移信息，在构造时只对这些迁移进行编码。就是可以通过SESE计算出每一时间步骤下可执行的单元格，这同样可以减小公式的大小。

SMT-LIB例子



注释：Basic Boolean example

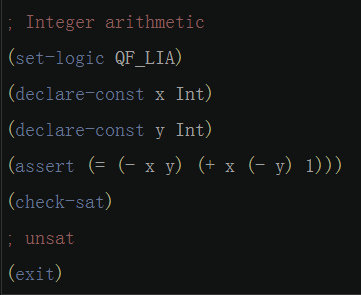
将:print-success选项设置为false时，求解器在返回成功的情况下不会产生响应

设置逻辑QF\_UF

定义Boolean常量p

添加公式

（check-sat）响应结果为unsat表明求解器已经建立，但其中没有模型



注释：Integer arithmetic

设置逻辑QF\_LIA

定义整型常量x

定义整型常量y

添加公式

（check-sat）响应结果为unsat表明求解器已经建立，但其中没有模型