

假定TemperatureLight和ActionLight角色所表示的分别是：

与

对于fsm1而言：









对应迁移Tkeep→Theat

对应迁移Theat→Tkeep

对应迁移Theat→Theat

对应迁移Tinit→Theat

对于fsm2而言：









对应迁移Ago→Astop

对应迁移Astop→Ago

对应迁移init→Astop

**原子命题集AP**



关于变量count的定义域的原子命题集：









**状态集**





因此状态集中的每个状态s是一个多元组：

其中每个原子的取值可以是0或者1

**标记函数L**



初始状态是：（1,0,0,1,0,0）

**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



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

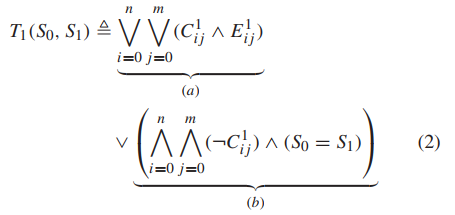
（e）of the textual format

PortName\_isPresent;

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

SR模型的迁移关系的BMC编码思路

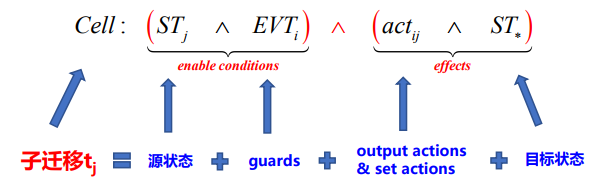
BMC编码思路方法思想主要来源于老师论文中的公式（2）

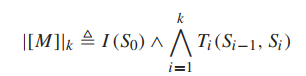


此公式理解为，状态到状态的迁移为两种情况：如果存在cell运行，每次最多只能有一个cell运行，则调用子公式（a），对运行的cell（i，j）来说，当前步既有迁移可行条件也有结果；如果没有cell运行，当前步状态与上一步状态相同，代表当前步不存在迁移可行条件。

根据子式（a），我们可以得出结论在有一个cell运行后，我们要同时考虑**可行条件**和**结果**。

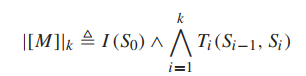
Cell：可行条件 + 结果



**BMC编码公式：**

代表初始状态下编码。是M的迁移关系编码

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量名称 | 取值 | 意义 |
| 状态描述变量 |  | 0,1,2 | STCL=0,1,2分别表示TemperatureLight的状态是：Tinit，Theat，Tkeep |
|  | 0,1,2 | STPL=0,1,2分别表示ActionLight的状态是：init，Astop，Ago |
| 内部参数变量 | *count* | N\* | 表示时间的流失 |
| 端口描述变量 | *Sec* |  | D对应于输入端口处符合条件的任意条件，-1表示无token |
| *Theat* | 0,1,-1 | Theat=0,1分别表示在Theat端口处有token，且token取值为0和1，即表示heat灯的关闭和打开，-1表示无token |
| *Tkeep* | 0,1,-1 | Tkeep=0,1分别表示在Tkeep端口处有token，且token取值为0和1，即表示keep灯的关闭和打开，-1表示无token |
| *Pgo* | 0,1,-1 | Pgo=0,1分别表示在Pgo端口处有token，且token取值为0和1， -1表示无token |
| *Pstop* | 0,1,-1 | Pstop=0,1分别表示在Pstop端口处有token，且token取值为0和1，-1表示无token |
| *Ago* | 0,1,-1 | Theat=0,1分别表示在Theat端口处有token，且token取值为0和1，即表示heat灯的关闭和打开，-1表示无token |
| *Astop* | 0,1,-1 | Theat=0,1分别表示在Theat端口处有token，且token取值为0和1，即表示heat灯的关闭和打开，-1表示无token |

**BMC编码公式：**

初始状态编码：

环境输入端口

状态描述变量



只属于ActionLight角色中的输出端口

通信端口

只属于TemperatureLight的输出端口

迁移关系编码：

tick = i(i>1)时，迁移关系通式：



这里，表示：TemperatureLight角色中所有的状态变量与内部参数变量均保持不变，即：



表示：TemperatureLight角色中的所有输出端口均无token，即：



表示：ActionLight角色中所有的状态变量与内部参数变量均保持不变，即：



表示：TemperatureLight角色中的所有输出端口均无token，即：



对于一个FSMActor角色来说，只有发生迁移和没发生迁移两种情况，若没有迁移发生，角色内所有的状态变量与内部参数均保持不变，并且所有输出端口均无token；若发生迁移，迁移如下

其中，

其中，

其中，



其中，



其中，



其中，



其中，

**手工计算：**

**tick = 0** Sec = -1 ST\_TL = 0 ST\_AL = 0

count = 0 Theat = -1 Tkeep = -1

Pgo = -1 Pstop = -1 Ago = -1 Astop = -1

**tick = 1** Sec = 1 ST\_TL = 1 ST\_AL = 1

count = 0 Theat = 1 Tkeep = 0

Pgo = -1 Pstop = -1 Ago = 0 Astop = 1

**tick = 2** Sec = 2 ST\_TL = 1 ST\_AL = 1

count = 1 Theat = -1 Tkeep = -1

Pgo = -1 Pstop = -1 Ago = -1 Astop = -1

**tick = 3** Sec = 3 ST\_TL = 1 ST\_AL = 1

count = 2 Theat = -1 Tkeep = -1

Pgo = -1 Pstop = -1 Ago = -1 Astop = -1

**tick = 4** Sec = 4 ST\_TL = 2 ST\_AL = 2

count = -1 Theat = 0 Tkeep = 1

Pgo = 1 Pstop = -1 Ago = 1 Astop = 0

**tick = 5** Sec = 5 ST\_TL = 1 ST\_AL = 1

count = 0 Theat = 1 Tkeep = 0

Pgo = 0 Pstop = 1 Ago = 0 Astop = 1

**tick = 6** Sec = 6 ST\_TL = 1 ST\_AL = 1

count = 1 Theat = -1 Tkeep = -1

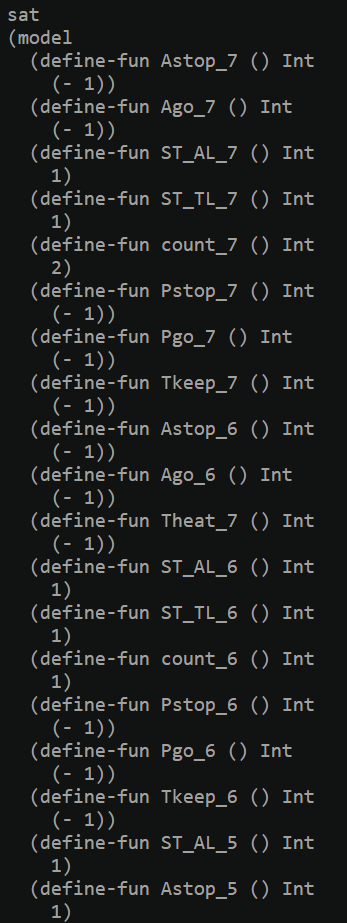
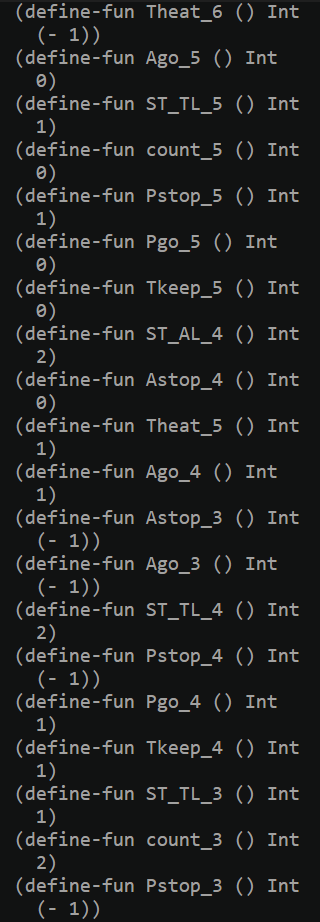
Pgo = -1 Pstop = -1 Ago = -1 Astop = -1

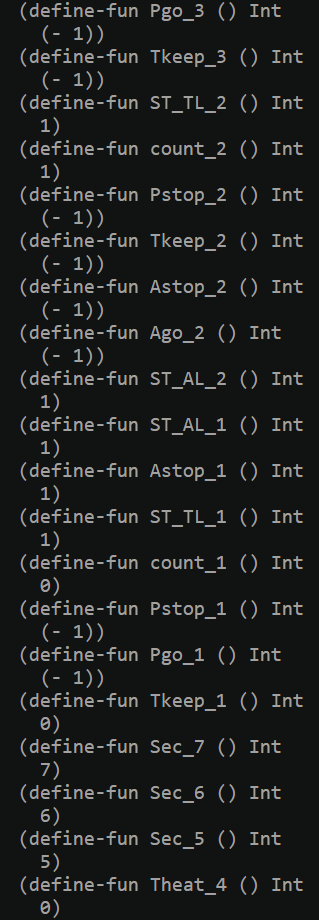
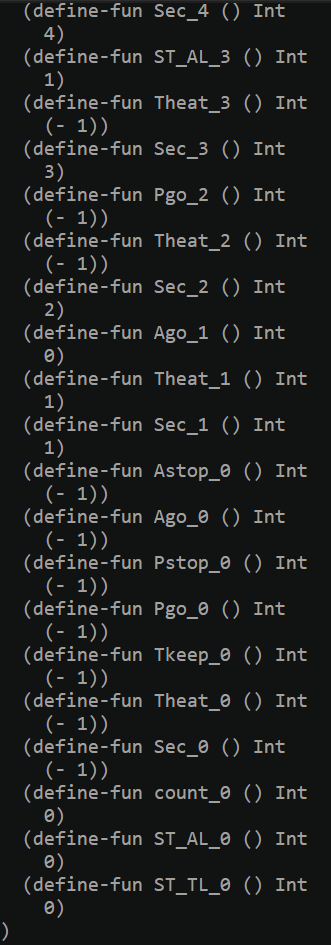
**tick = 7** Sec = 7 ST\_TL = 1 ST\_AL = 1

count = 2 Theat = -1 Tkeep = -1

Pgo = -1 Pstop = -1 Ago = -1 Astop = -1

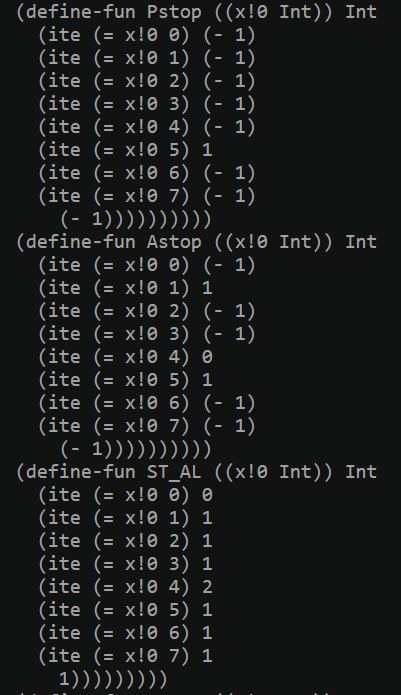
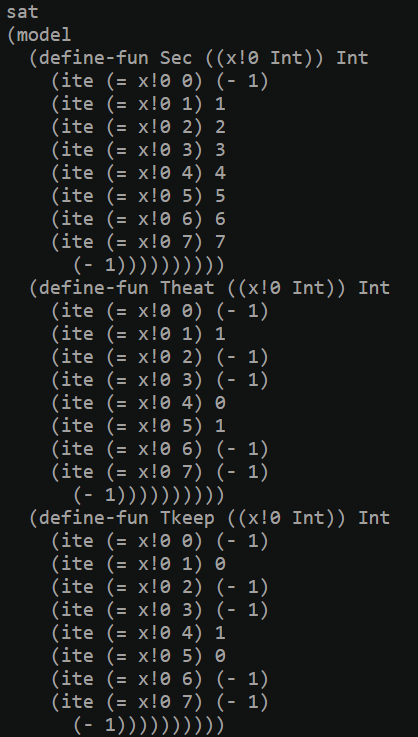
**枚举法结果：**

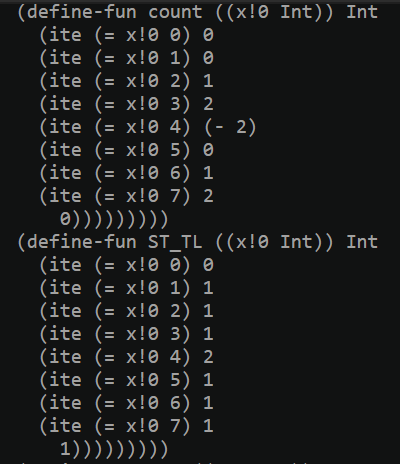
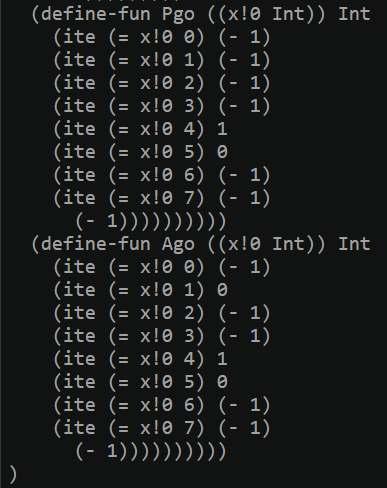
 

**与手动计算相同！**

**函数法结果：**



**与手动计算相同！**

对于不动点的查找，运用迭代的思想。

以师姐的例子为例，先运行FSMActor1，再运行FSMActor2。对两个FSM角色反复迭代运行，直至输出、输入端口不再发生改变。

而不动点的BMC编码，在我看来与正常BMC编码并无区别。列出每一步的迁移式子，在对其进行BMC编译。

FSM,全称Finite State Machine。FSM的输出是由系统的当前输入和当前状态共同决定的。一个FSM包括一系列状态和转换，其中一个状态是初始状态，并且任何数量的状态可能是最终状态。每次迁移需要满足一个警戒条件，并且迁移会产生任意数量的输出行为和设置行为。