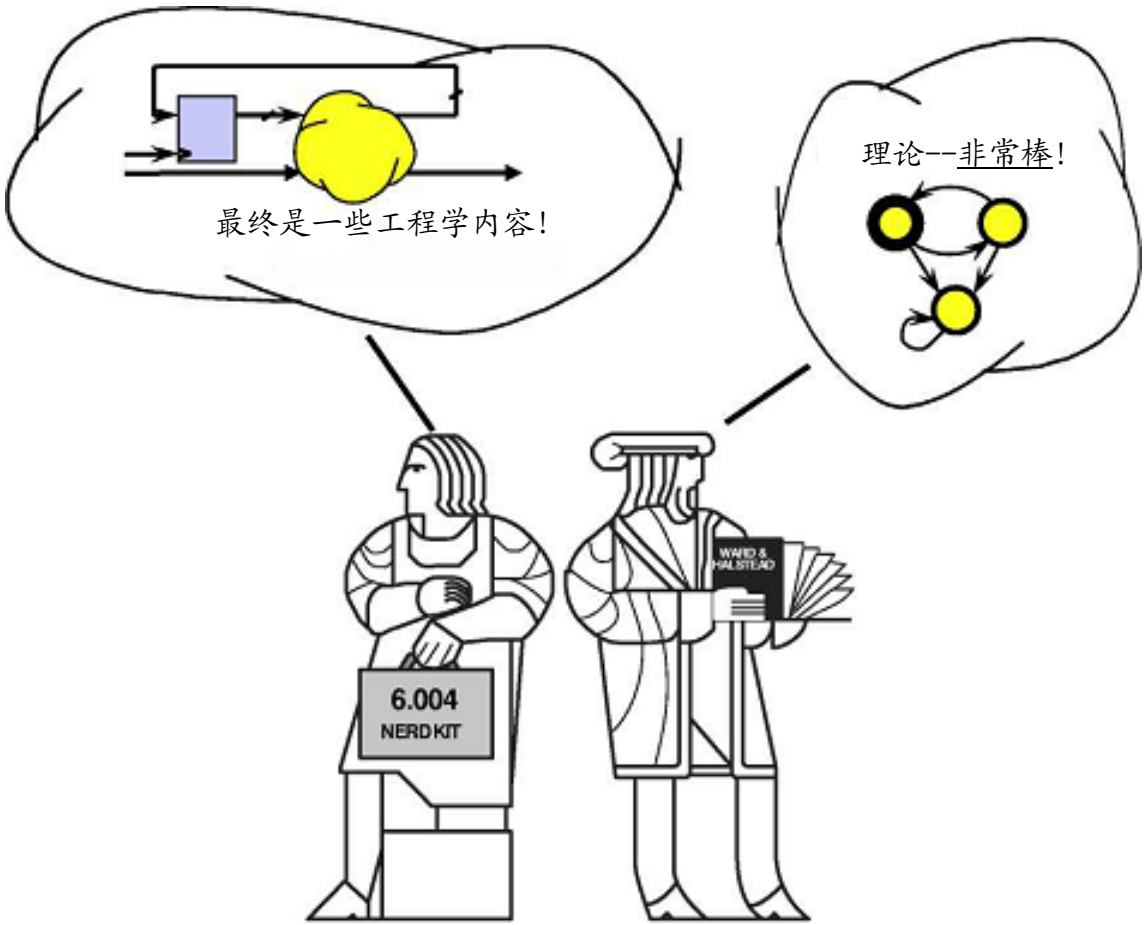
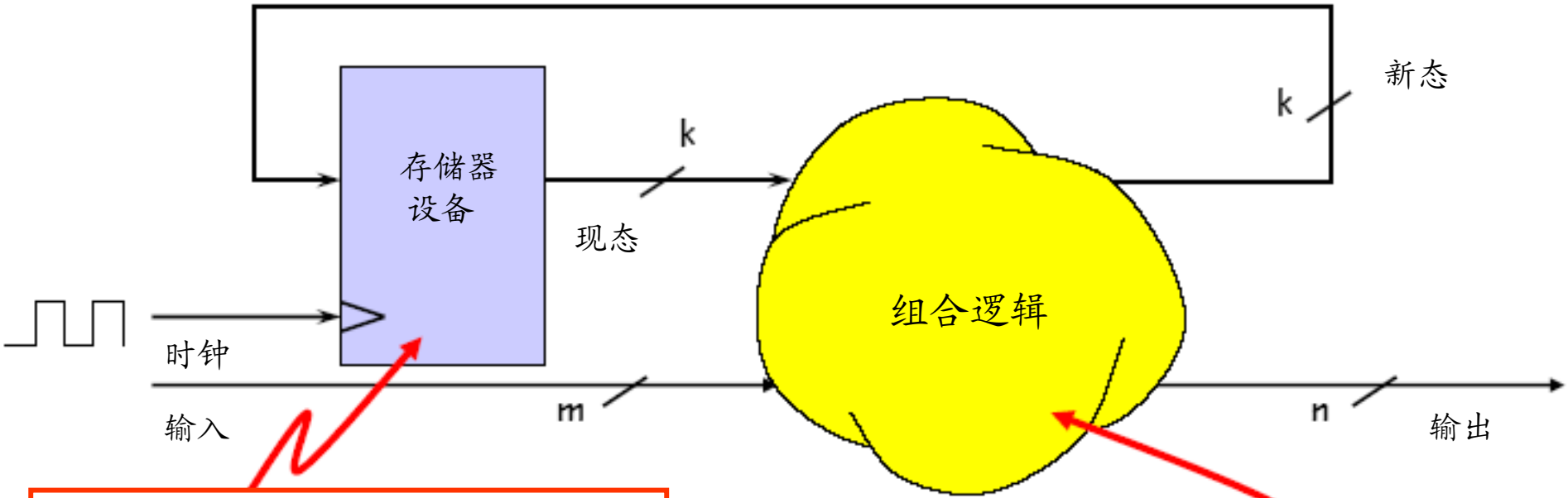
（同步）有限状态机



实验2预定在今晚进行

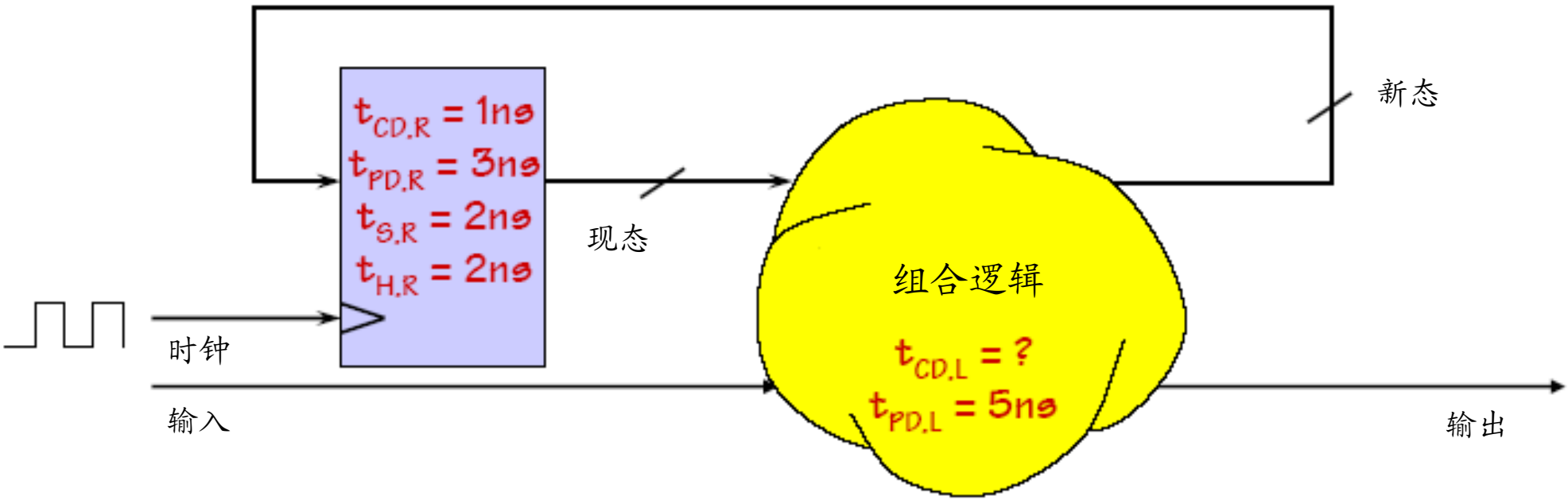
我们的新计算机



- 工程周期;
- 只有遵守动态规则时，才能正常工作;
- 记住：2^k种不同组合总共需要k位。

- 非循环图;
- 遵守静态规则;
- 通过一个由2^{k+m}行和k+n输出列组成的
  真值表，可以穷尽列举其全部逻辑。

必须注意时序假定条件



问题:

- 该逻辑中针对T_{CD}的约束条件是什么?
- 最小时钟周期?
- 输入端的设置和保持时间是多少?

$t_{CD,R}$ (1 ns) + $t_{CD,L}$(?) > $t_{H,R}$(2 ns)

$t_{CD,L}$ > 1 ns

$t_{CLK} > t_{PD,R}+t_{PD,L}+ t_{S,R} > 10nS$

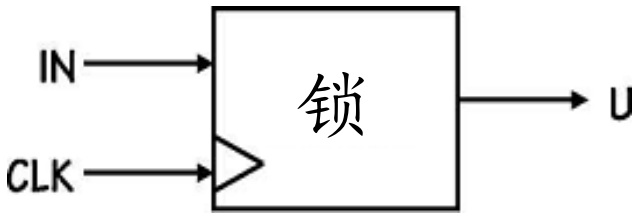
$t_S = t_{PD,L} + t_{S,R} = 7$ nS

$t_H = t_{H,R} - t_{CD,L} = 1$ nS

我们知道它有多快...但是，那又怎么样?

一个简单的时序电路...

让我们来构造一个二进制数字*组合锁*:

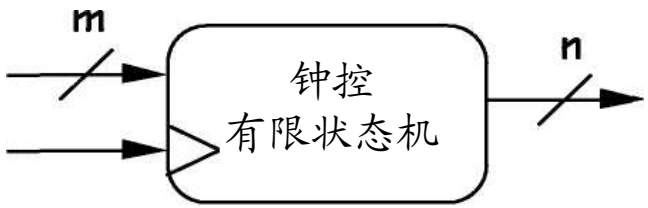


我们需要几个寄存器?

规范:

- 一个输入(“0”或“1”);
- 一个输出(解锁信号);
- 当且仅当为1时解锁: 该“组合”的最后四个输入为: 0110。

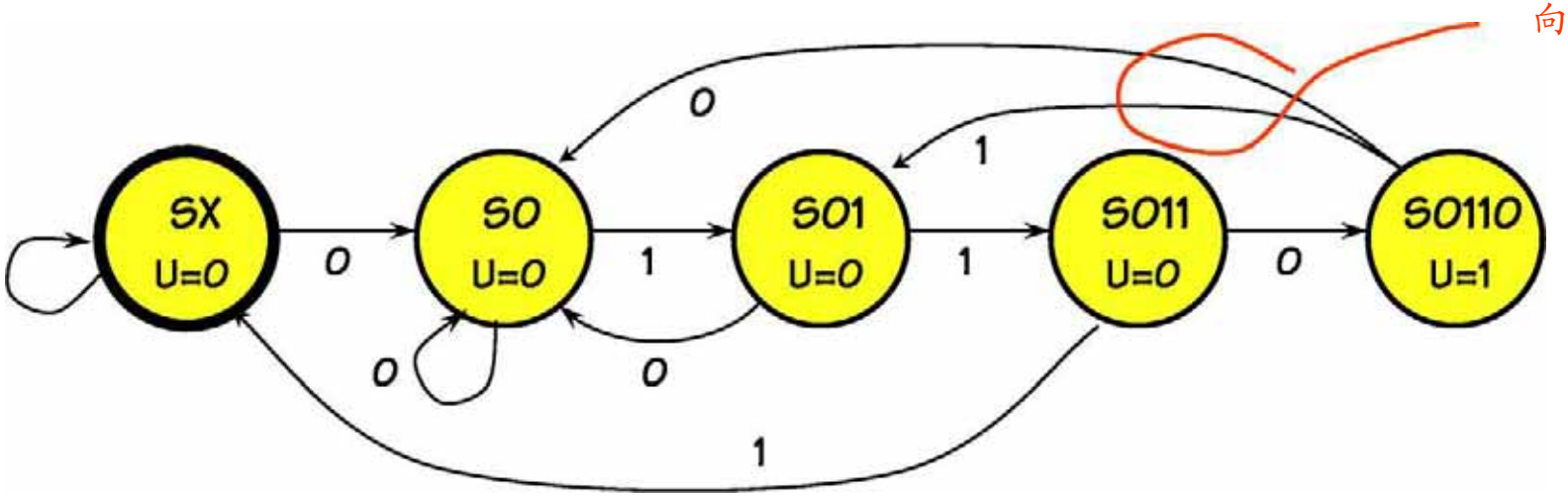
特殊抽象：有限状态机



- 一个有限状态机具有：
  - k个状态：$S_1 \ldots S_k$（其中的某一个为“初始”状态）；
  - m个输入：$I_1 \ldots I_m$；
  - n个输出：$O_1 \ldots O_n$；
  - 针对每个状态s和输入I，有一个转换规则s‘（s，I）；
  - 针对每个状态s，有一个输出规则Out（s）。

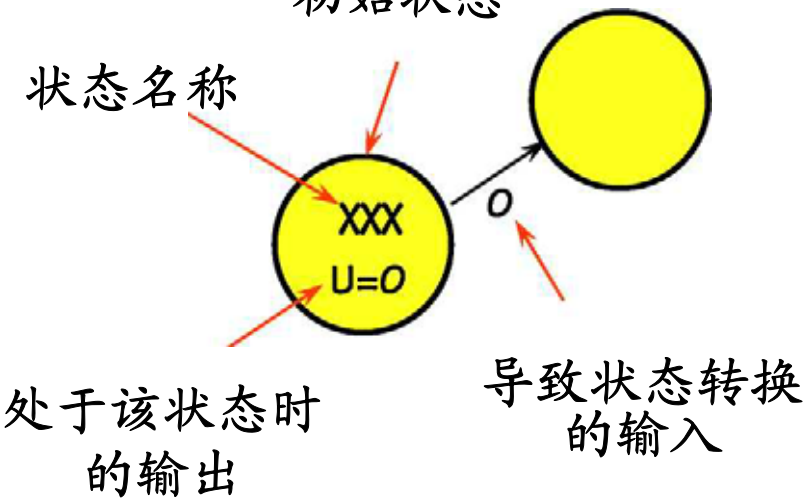
状态转换图

为什么这些要转向S0和S01?

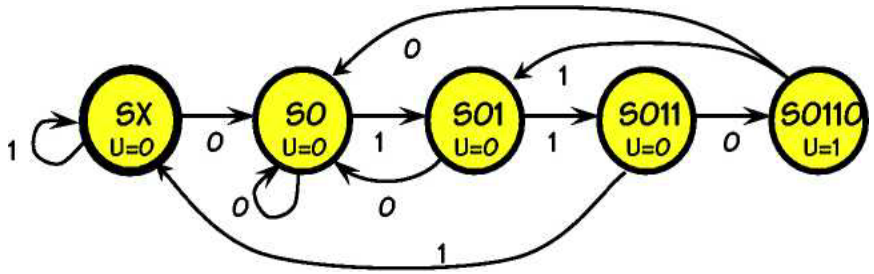


设计我们的锁...

- 需要一个初始状态，称之为SX;
- 在适当的项序列中，每一步都必须有一个独立的状态;
- 必须处理其他（错误的）项。



还有另外一种规范



输入	现态		次态	解锁	
0	SX	000	S0	001	0
1	SX	000	SX	000	0
0	S0	001	S0	001	0
1	S0	001	S01	011	0
0	S01	011	S0	001	0
1	S01	011	S011	010	0
0	S011	010	S0110	100	0
1	S011	010	SX	000	0
0	S0110	100	S0	001	
1	S0110	100	S01	011	

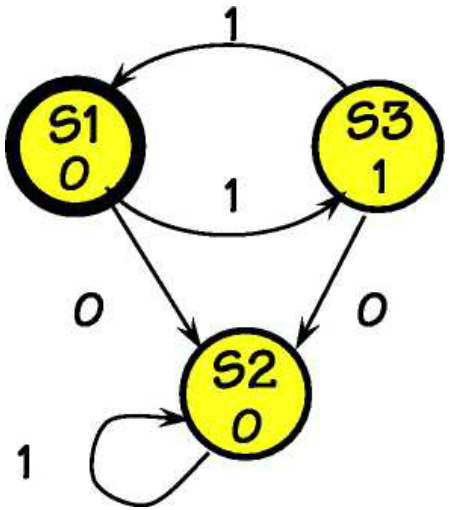
可以为状态任意分配代码，但是，如果你仔细选择分配方案，就可以大大简化逻辑需求。

所有状态转换图都可以用真值表来进行描述...

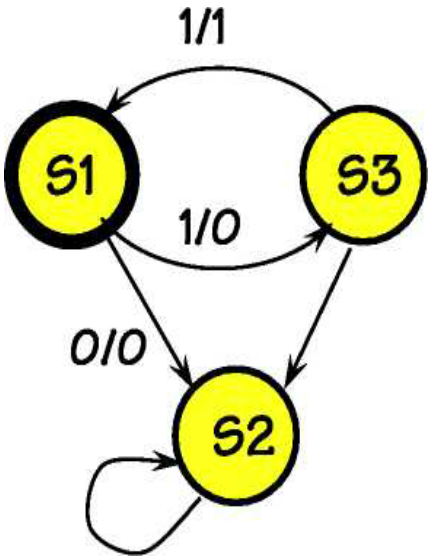
为每个状态分配一个二进制编码（某种技术中的一位）。

可以采用我们学过的用于组合逻辑的化简方法来简化真值表。

有效的状态图



摩尔机:
输出依赖于状态

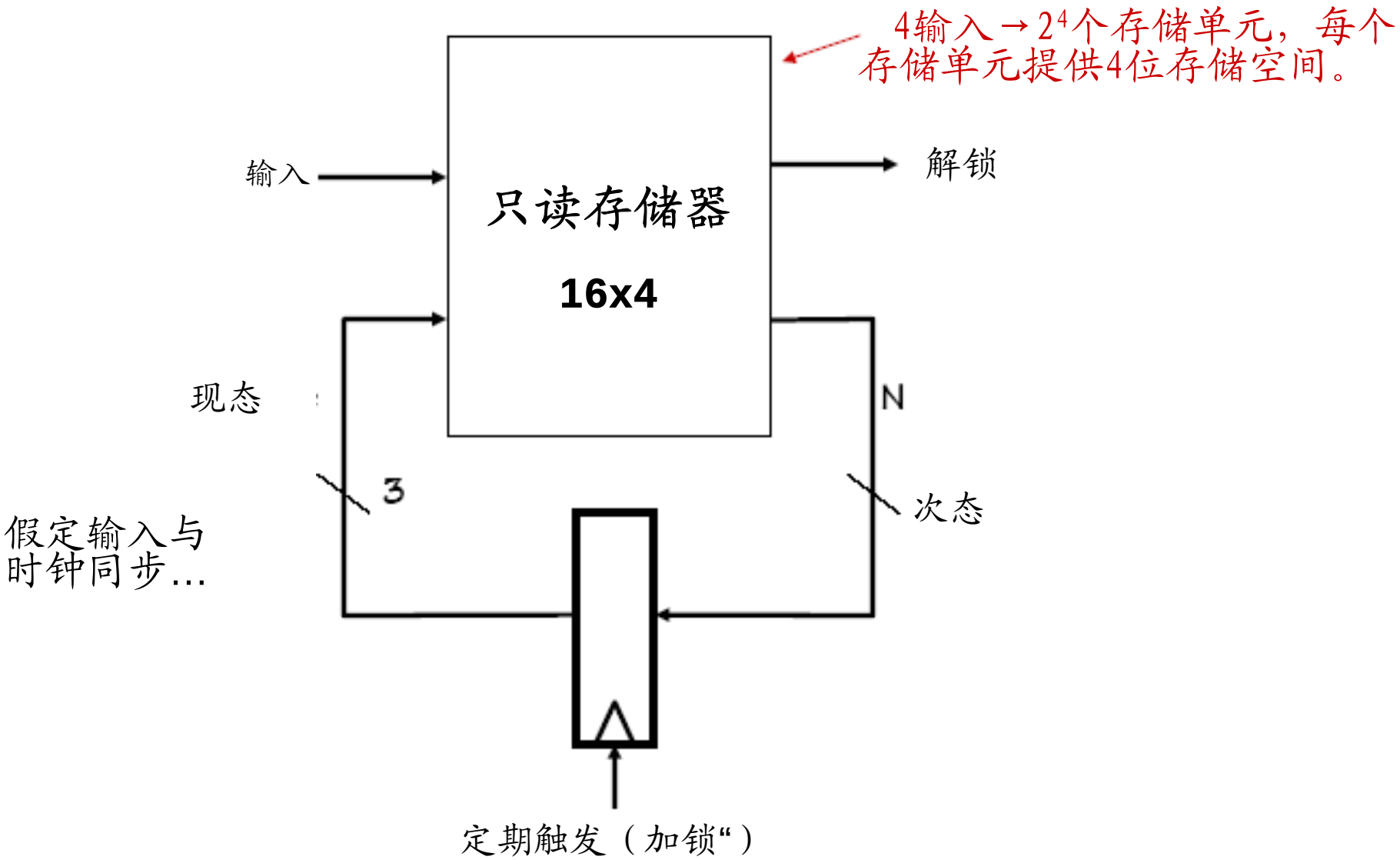


米莉机:
输出依赖于转换

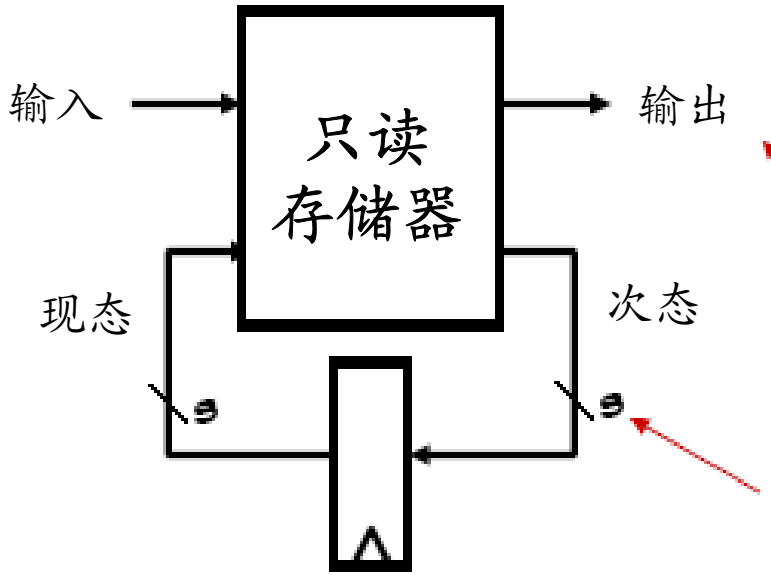
- 离开某种状态的弧必须是:
- （1）互斥
  ——对于给定的输入值，不能有两种选择。

- （2）完全选择
  ——*每种状态都必须指定对于每一种可能的输入组合会发生什么情况。*
  “什么也没有发生”意味着返回到该状态自身。

现在将它放置在硬件中

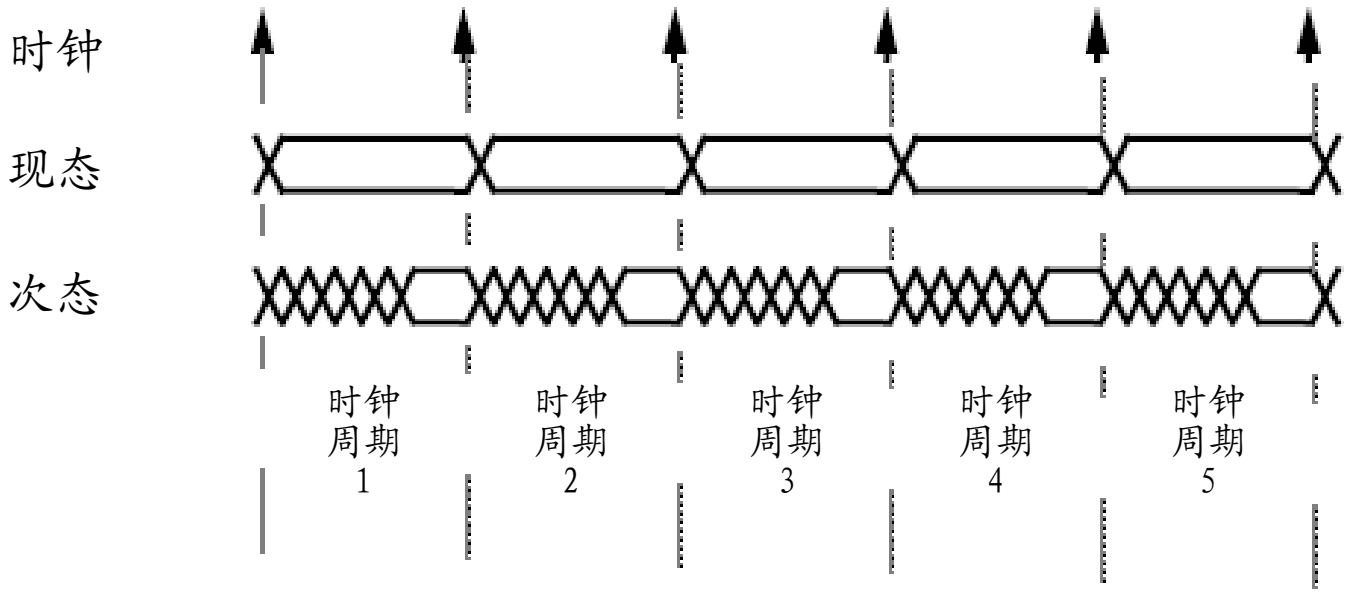

4输入→2^4个存储单元，每个存储单元提供4位存储空间。
输入
只读存储器
16x4
解锁
现态
N
3
次态
假定输入与时钟同步...
定期触发（加锁“）

离散状态，时间



两种设计选择:
（1）输出仅依赖于状态（摩尔机）；
（2）输出依赖于输入+状态（米莉机）。

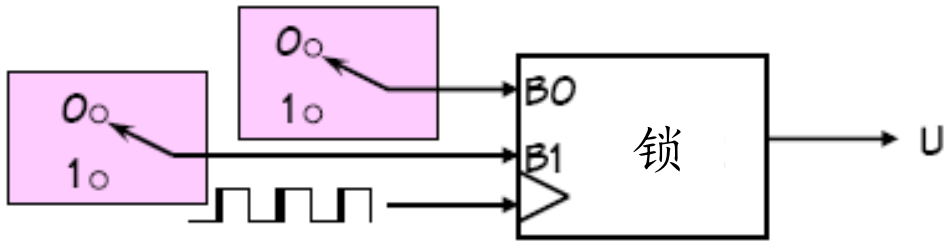
s个状态位→2^s种可能的状态

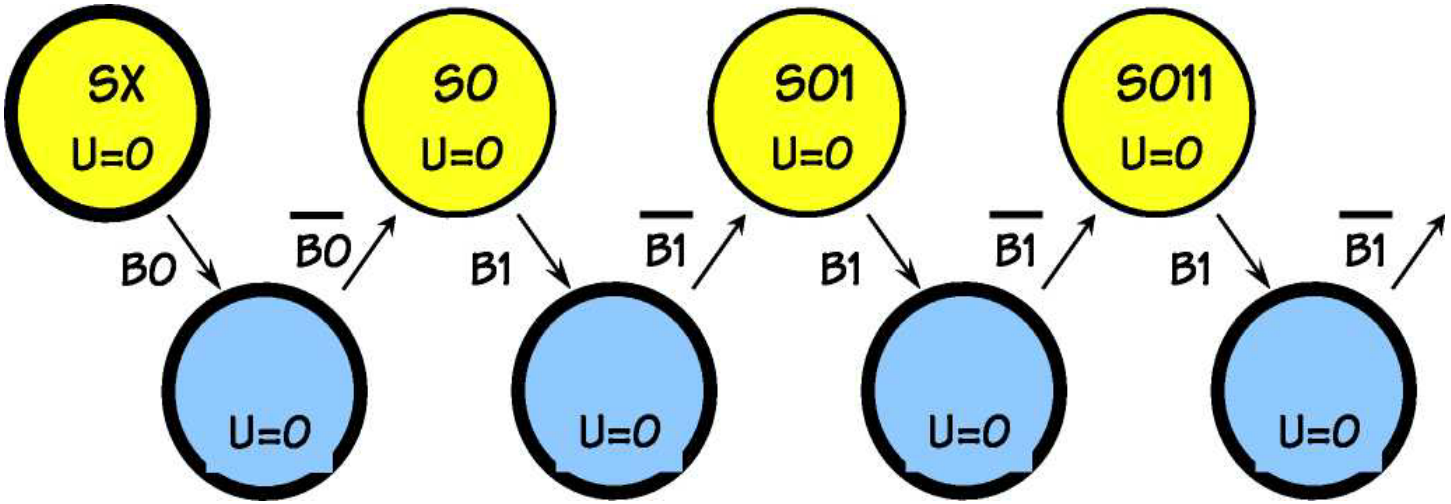


异步输入-I

在我们给出的例子中，假定每个输入对应着一次时钟转换。如果“按钮器”没有察觉到时钟信号或是与时钟不同步，将会出现什么现象？

如果每个按钮输入是一个异步的0/1电平，会发生什么情况？怎样做才能防止单个按钮生成多个转换？



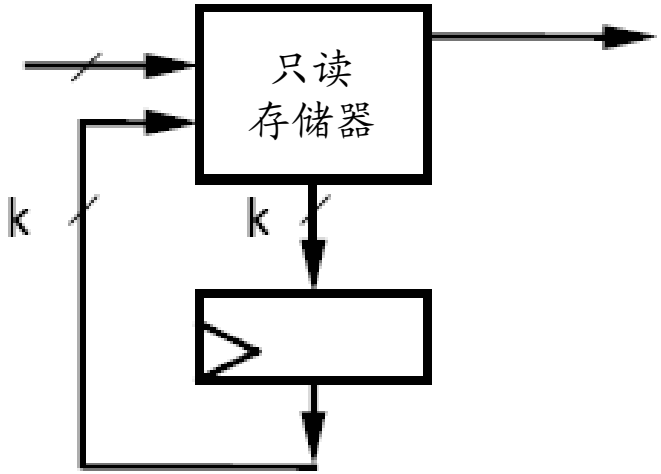


但是，动态规则又会怎样呢？

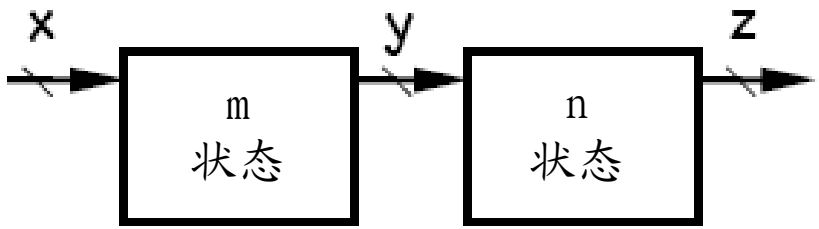
使用 *中间*状态来同步按钮！

有限状态机的团队游戏

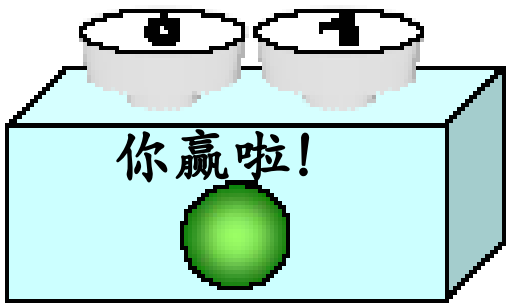
1. 你对状态数有多少了解?



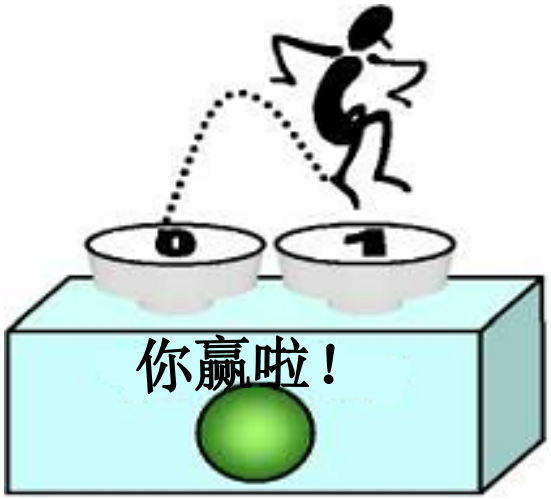
2. 相同问题:



3. 此处给出的是一个有限状态机,
   你能发现其规则吗?

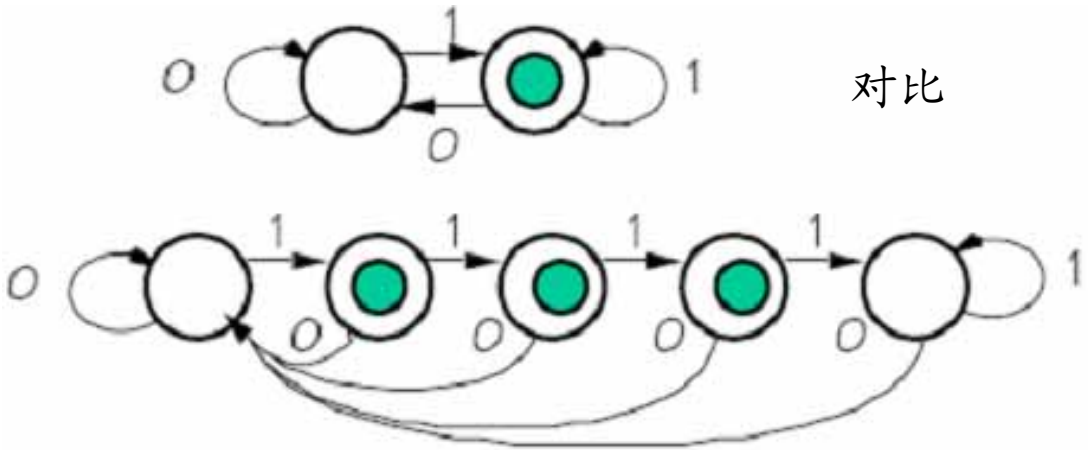


我的转换图是什么？



0=关闭，
1=打开？

”1111“=令人惊奇！



对比

- 如果你对有限状态机一无所知的话，你当然看不懂这个图形！
- 如果你限制了状态数，将会发现它有以下行为：

K状态有限状态机：可以用<k步来到达每一种（可获得的）状态。

但是...状态可能是等价的！

等价有限状态机

这两个状态图是否不一样?

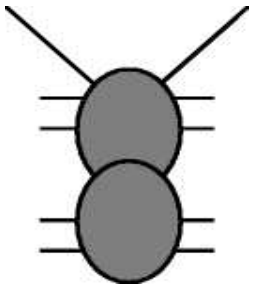
实际上根本没有区别！从外部来看，它们是不可区分的，因此是可以互换的。

> 当且仅当每一个输入序列都产生相同的输出序列时，两个有限状态机是等价的。

工程目标:

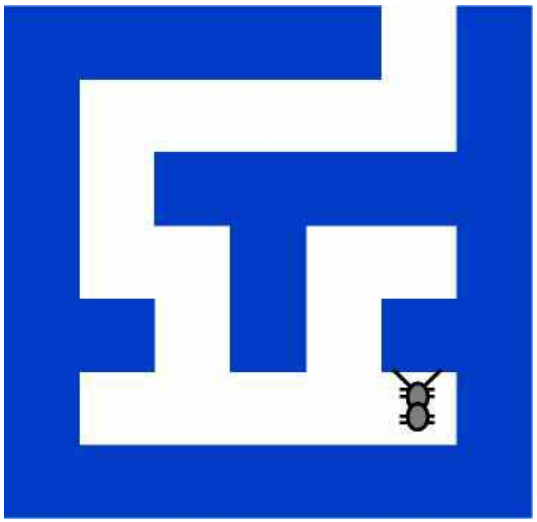
- 有一个能够工作的有限状态机...
- 想要最简单的（因此也是最便宜的）有限状态机。

让我们来创建一个蚂蚁



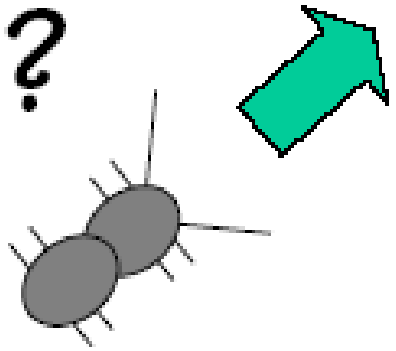
- 传感器：若接触到某个东西，则将触角L和R的值置为1。
- 制动器：向前走F步，左转（TL）和右转（TR）10度。

目标：使蚂蚁足够聪明，以便能够走出如下所示的迷宫：

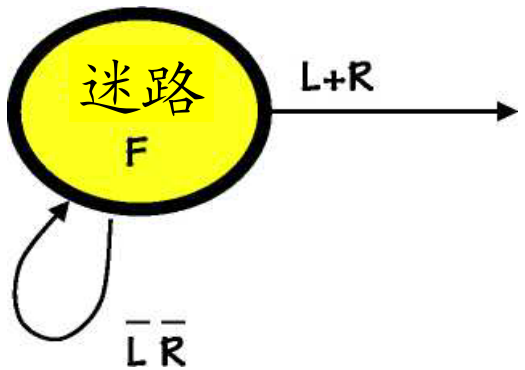


策略：右触角向墙。

迷路

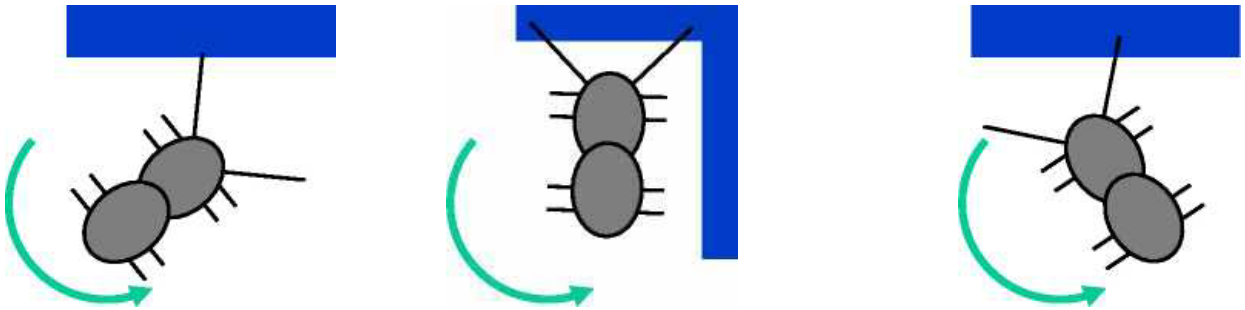


动作：一直朝向前走，直到碰到某个东西为止。

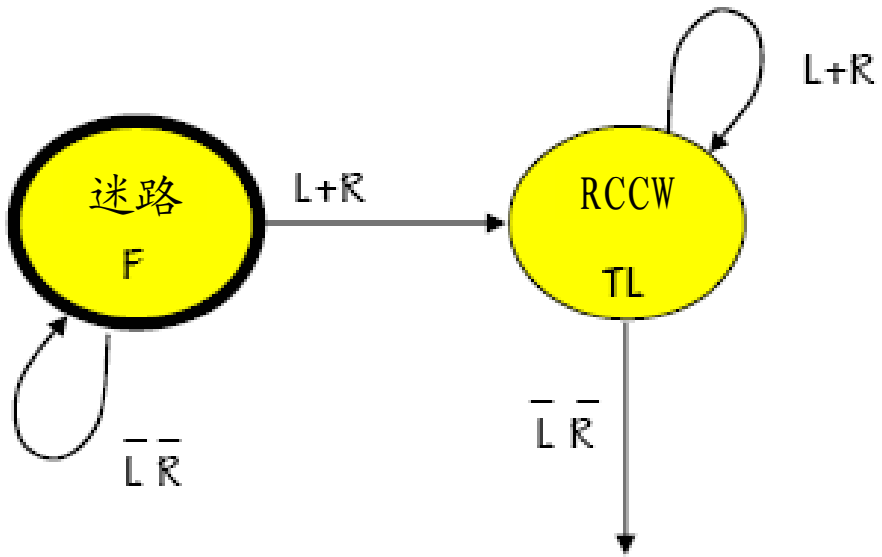


“迷路”是初始状态

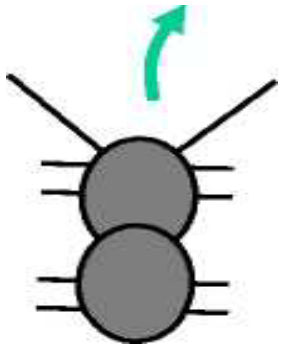
触碰后发出巨响！



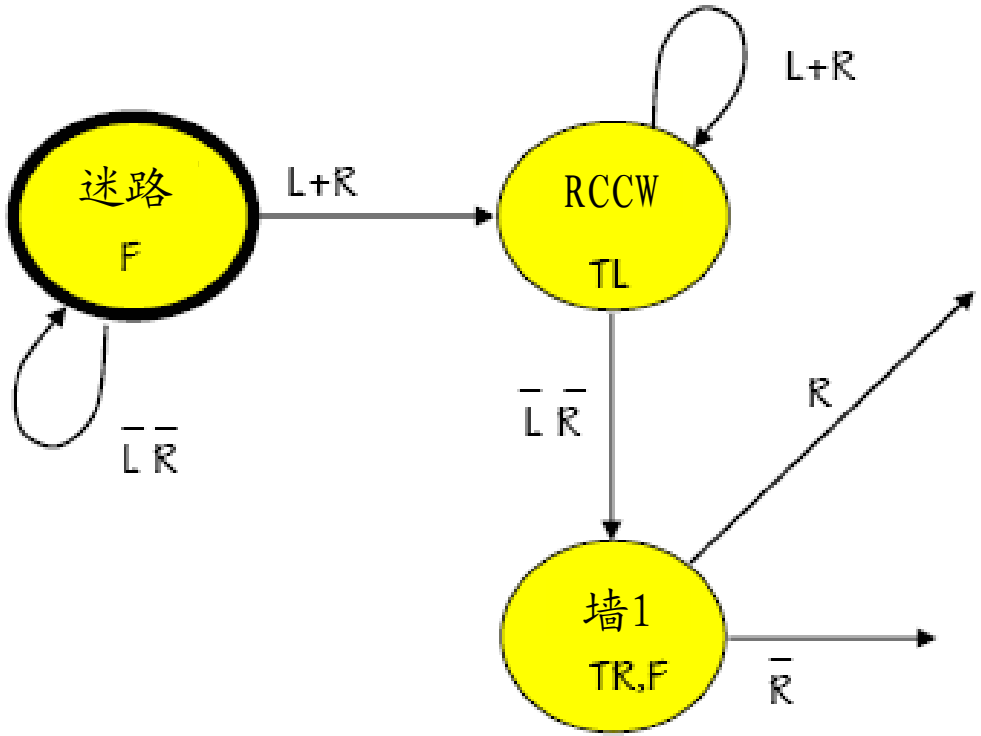
动作：向左转（CCW），直到不再触碰到任何东西时为止。



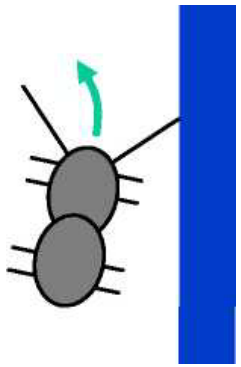
向右一点儿...



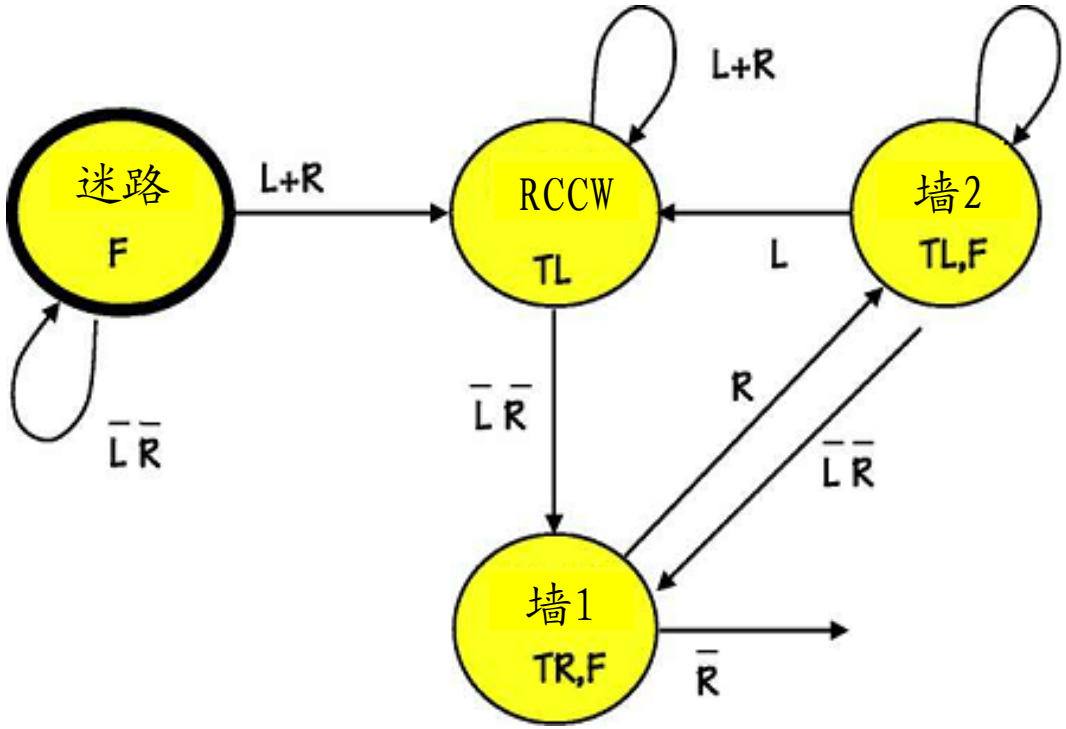
动作：向前并且右转一点，寻找墙。



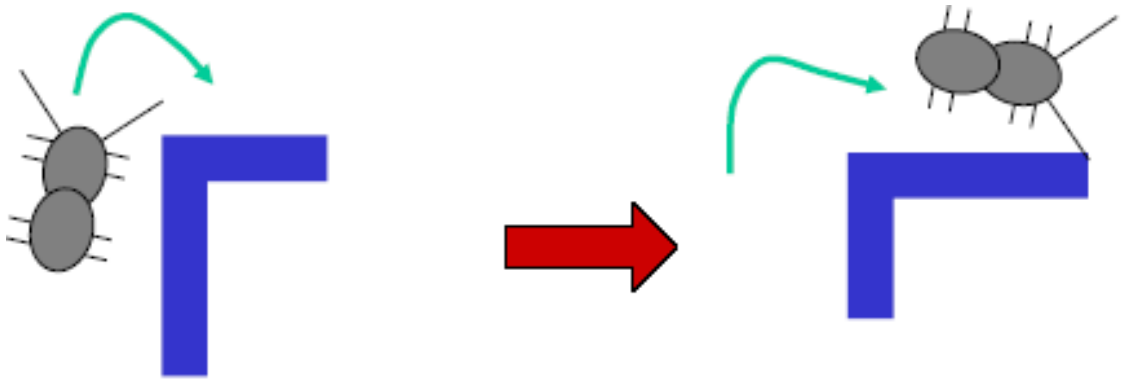
然后向左转一点儿



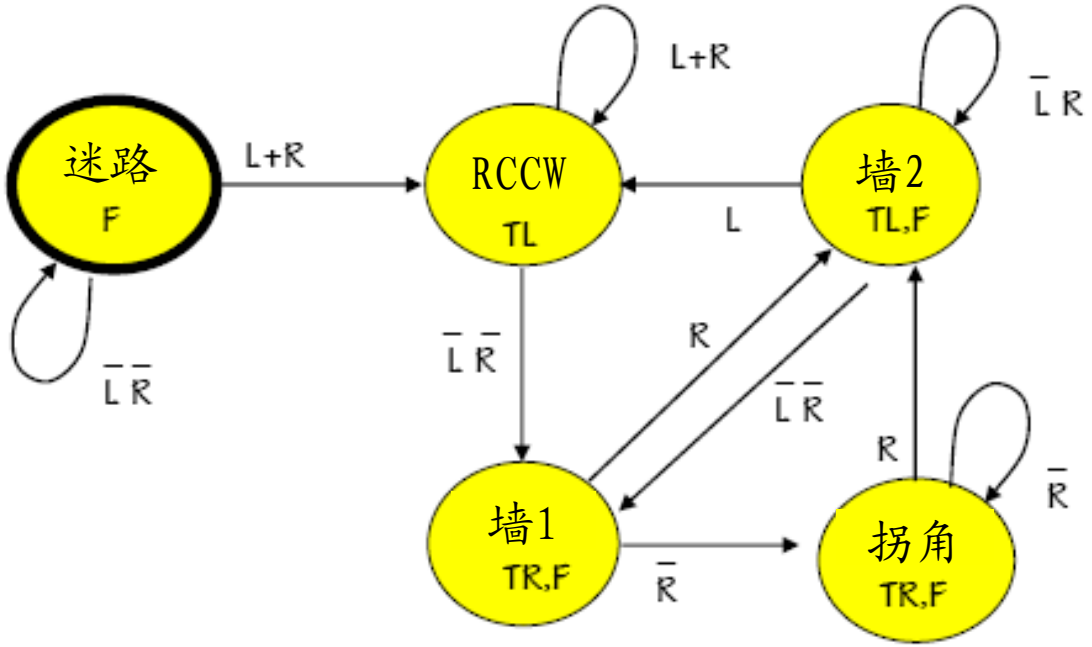
动作：向前并且左转一点，直到没有触碰到墙时为止。



处理拐脚



动作：向前并且右转，直到接触到垂直的墙时为止。



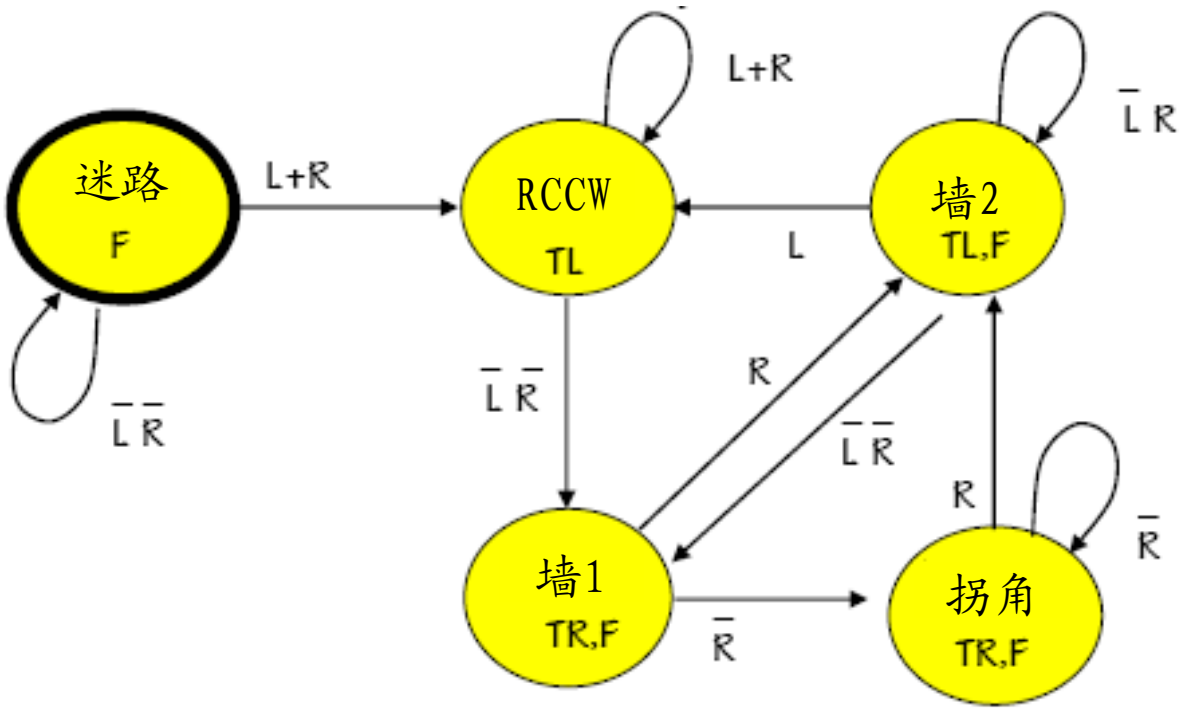
等价状态简化图

观察：当下列条件成立时，$S_i \equiv S_j$：

1. 两个状态的输出相同；并且
2. 每个输入→等价状态。

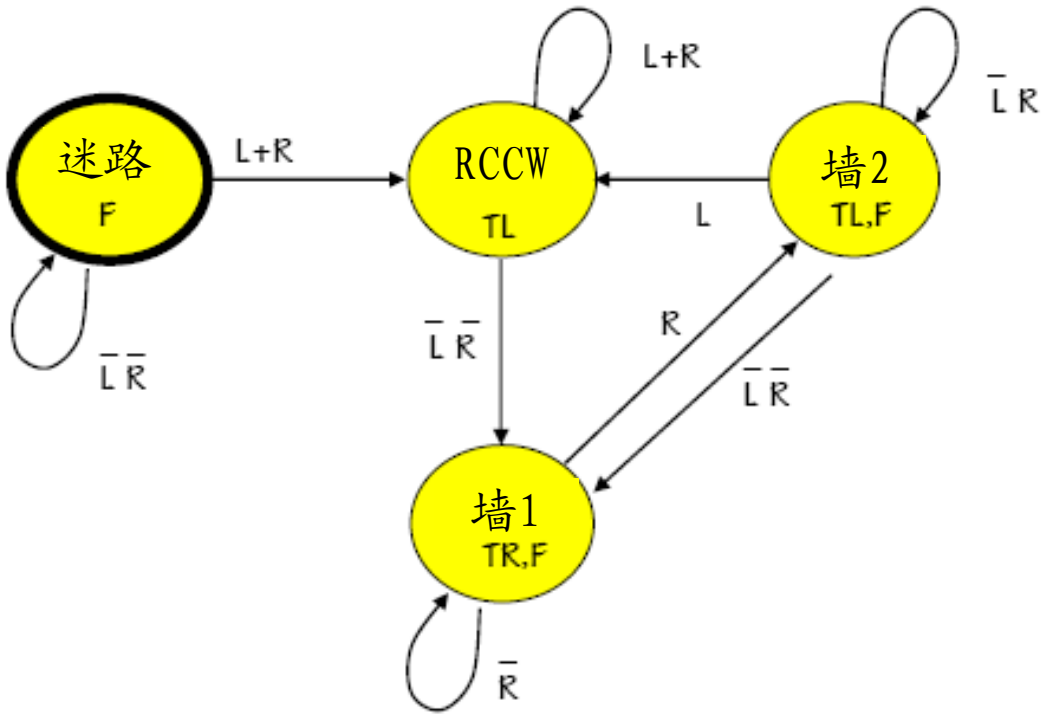
简化策略：

找到成对的等价状态，然后将其合并。



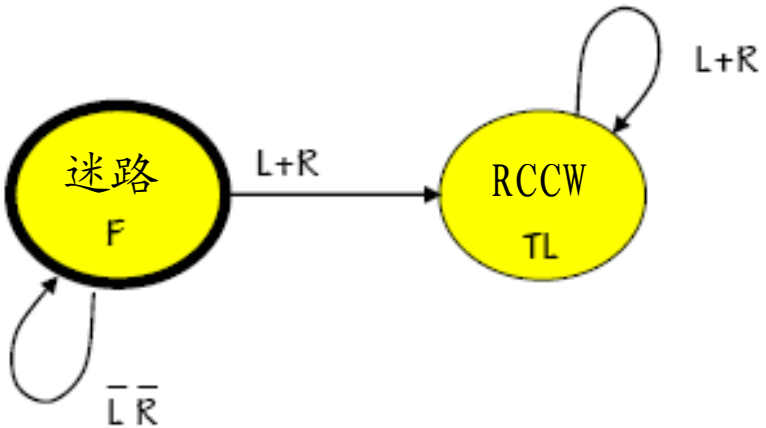
进一步简化步骤

将墙1和拐角这两种等价状态合并成一个新的组合状态。



其行为表现与前面的（5状态）有限状态机完全一致，但只需要一半的只读存储器就能够实现它！

构造转换表



S	L	R	S′	TR	TL	F
00	0	0	00	0	0	1
00	1	-	01	0	0	1
00	0	1	01	0	0	1
01	1	-	01	0	1	0
01	0	1	01	0	1	0

实现细节

	S	L	R	S′	TR	TL	F
迷路	00	0	0	00	0	0	1
	00	1	-	01	0	0	1
	00	0	1	01	0	0	1
RCCW	01	1	-	01	0	1	0
	01	0	1	01	0	1	0
	01	0	0	10	0	1	0
墙1	10	-	0	10	1	0	1
	10	-	1	11	1	0	1
墙2	11	1	-	01	0	1	1
	11	0	0	10	0	1	1
	11	0	1	11	0	1	1

完全转换表

S1′

LR \ S_1S_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	1
10	0	0	0	1

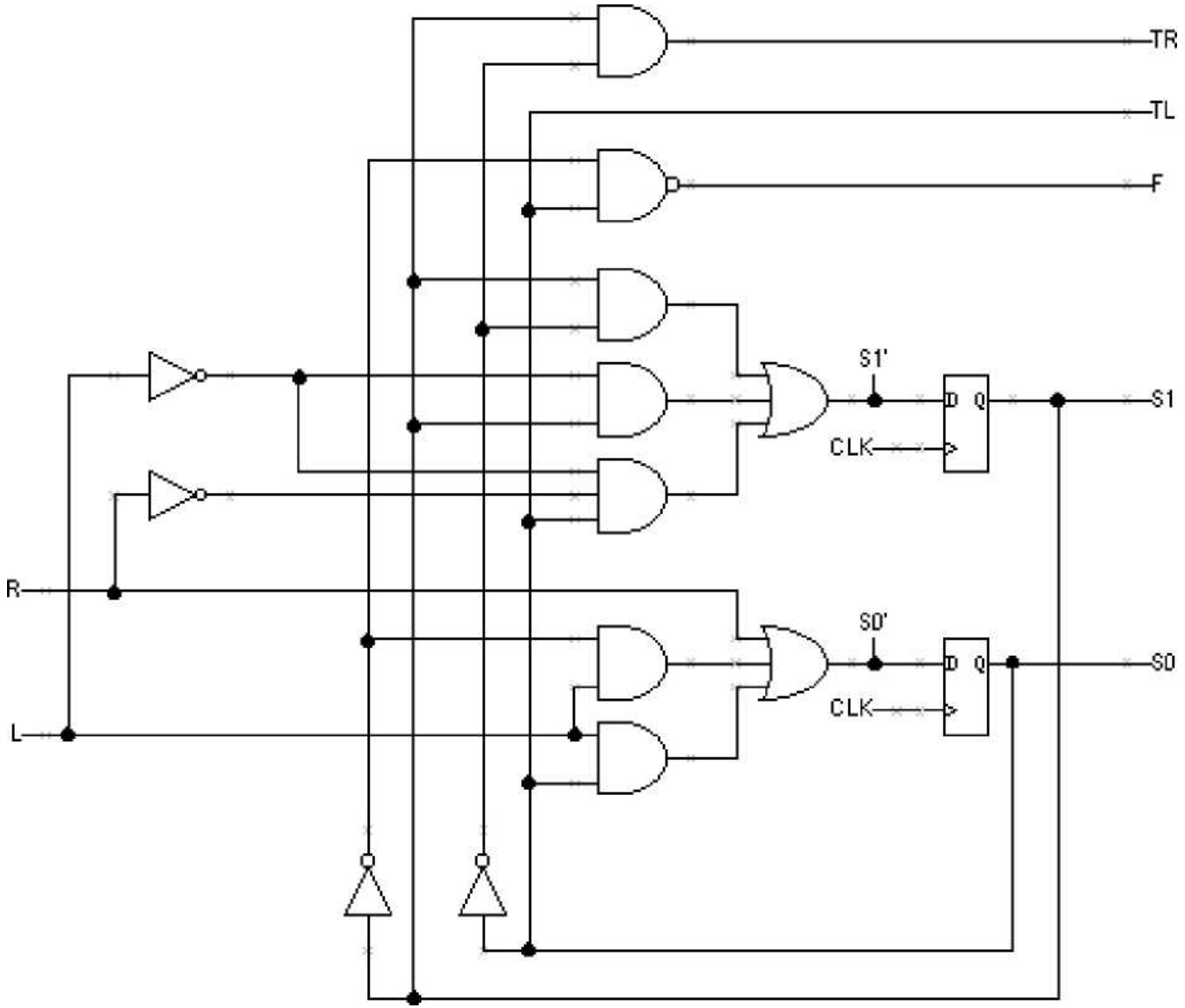
$$S_1' = S_1\overline{S_0} + \overline{L}S_1 + \overline{L}\,\overline{R}S_0$$

S0′

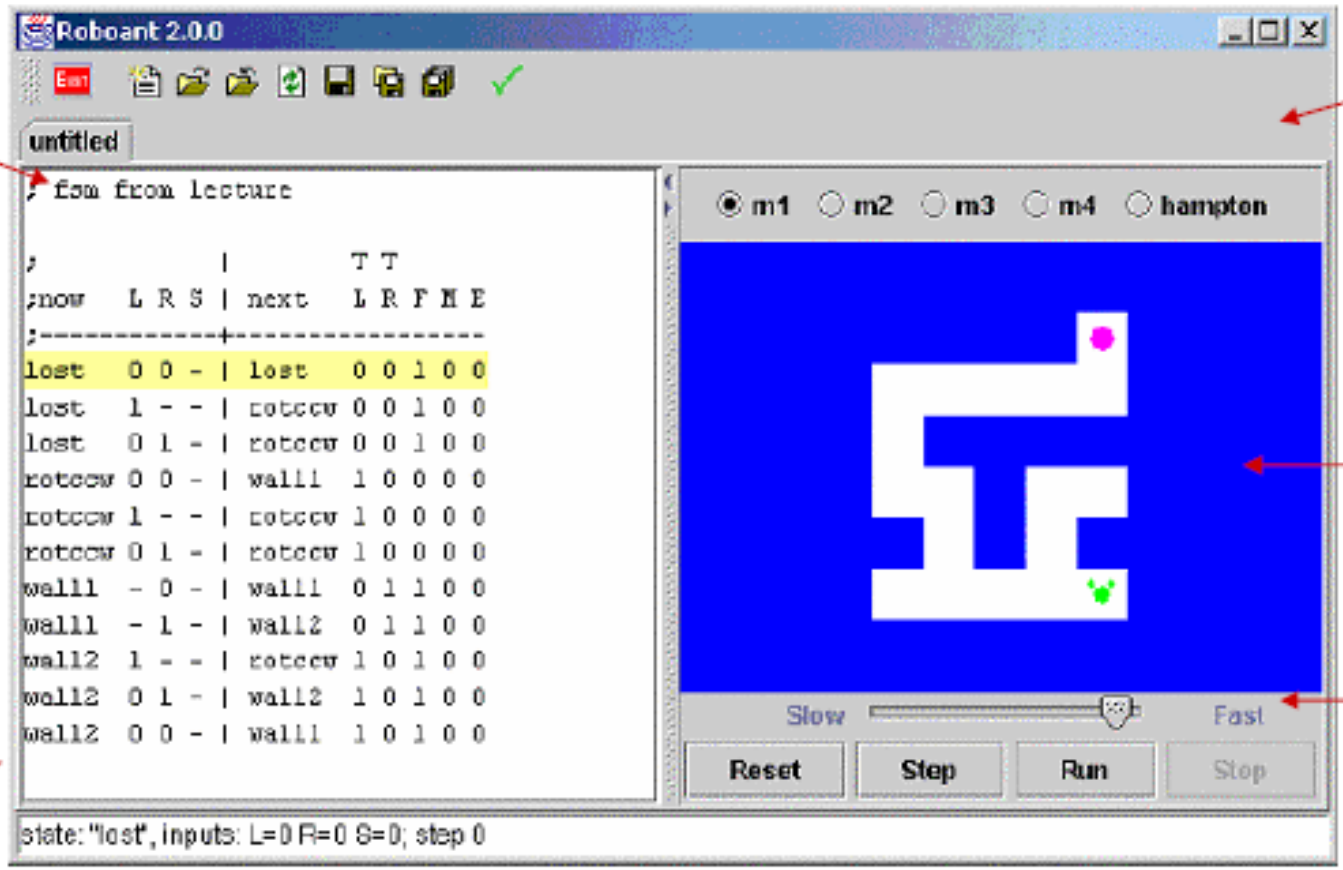
LR \ S_1S_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	0

$$S_0' = R + L\overline{S_1} + LS_0$$

蚂蚁电路示意图



Roboant®



有限状态机状态表

迷宫选择

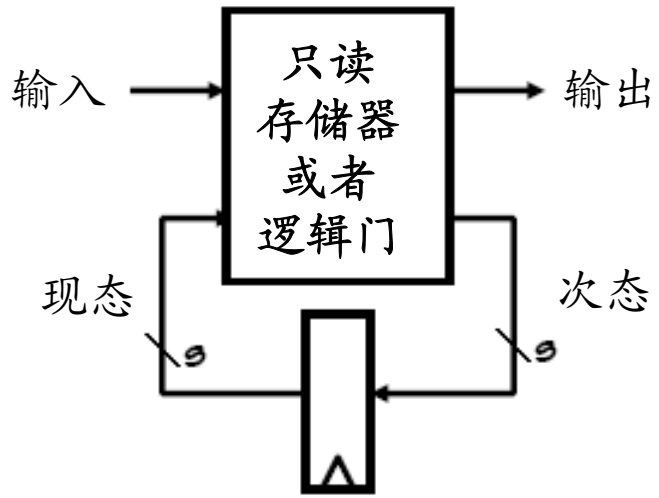
迷宫平面图

模拟控制

状态显示

新的Mark-II蚂蚁的特点是：可以添加（M）、擦除（E）和感知（S）所经过路径上的标记。

家务管理的一些问题...



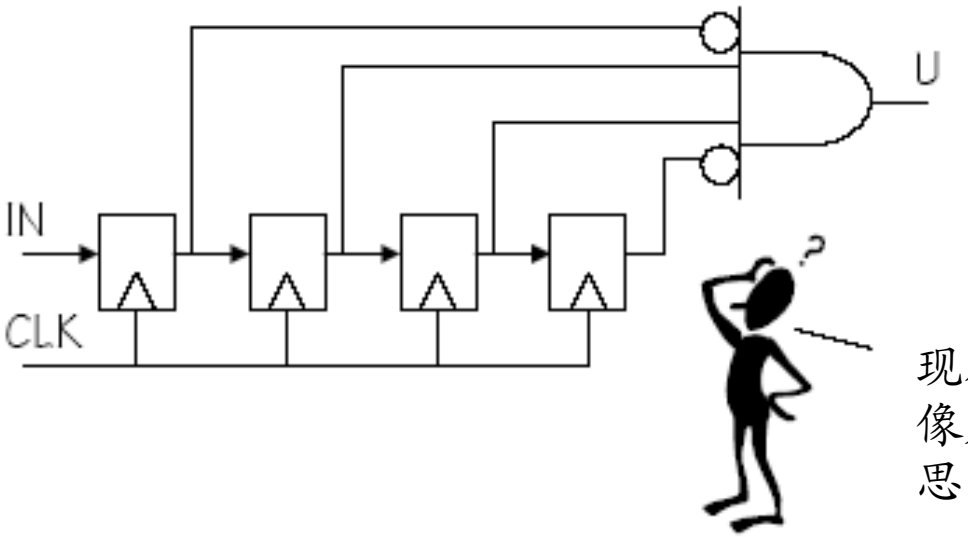
1. 是否进行初始化？是否清空存储器？

2. 是否存在无用的状态编码？

——浪费只读存储器
（使用可编程逻辑阵列或者逻辑门）
——有何意义？
——有限状态机能否恢复？

3. 是否为状态选择编码方案？

4. 状态更新是否与输入变化同步？



现在，它看上去像是一个很有意思的状态机。

进一步复杂化...

- 除了蚂蚁以外:
  蜂群、羊群和学校教育等都可以看作是非常简单的有限状态机的集合。

- 或许是大多数自然物:
  细胞自动机、简单有限状态机阵列、为流体建模要比偏微分方程的数值解决方案更加精确。

- 条件假设:
  假如我们用随机存储器来替换只读存储器，并且输出可以改变随机存储器的内容，情况会怎样呢?

...今后，你会看到许多有限状态机（我们目前创建的所有计算机恰好都是有限状态机）!