形式化方法

-- 课程简介

中国科学院软件研究所
张文辉
http://lcs.ios.ac.cn/~zwh/

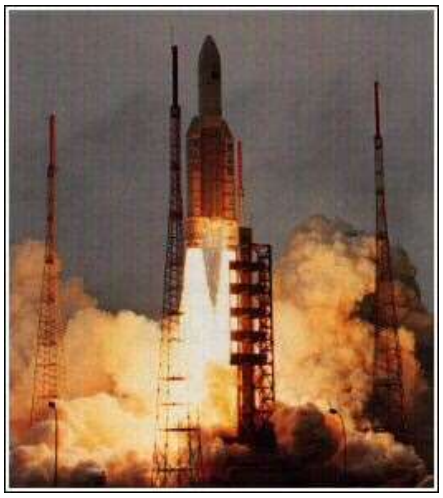
形式化方法

- 目标：开发程序与软件
  - 满足给定需求
  - 具有正确性保证

软件正确性：
在程序与软件基础上建立的软件系统行为符合给定的行为规范

软件正确性的重要性

- 安全攸关系统：运载火箭、高铁、飞机
- 软件问题举例：



软件正确性

- 软件正确性：软件工程及应用中的重要问题
- 形式化方法：保证软件系统正确性的重要手段

形式化方法

- 目标：开发具有正确性保证的程序与软件
- 途径：数学模型与形式化的分析验证方法

由软件系统数学模型定义的软件系统行为符合给定的行为规范

内容

- 形式化方法
- 软件正确性与软件系统行为
- 课程关注点

一、形式化方法

- 软件开发流程
- 理想化的软件开发流程
- 例子

软件开发流程

- 需求分析
- 概要设计
- 详细设计
- 软件编码
- 软件测试
- 软件交付……

软件系统行为符合需求?

软件开发流程

- 需求分析
- 概要设计
- 详细设计
- 软件编码
- 软件测试
- 软件交付……

软件系统行为符合需求?

系统行为 ➔软件代码 ➔详细设计 ➔概要设计 ➔给定需求

软件正确性？

传统的正确性保障手段

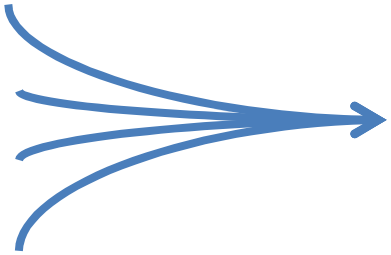
- 过程管理、软件评审
- 设计方法、编程方法、代码复查
- 软件测试

Program testing can be used to show the presence of bugs, but never to show the absence.

-- Edsger W. Dijkstra

理想化的软件开发流程

- 需求分析
- 概要设计
- 详细设计
- 软件编码
- 软件测试
- 软件交付……



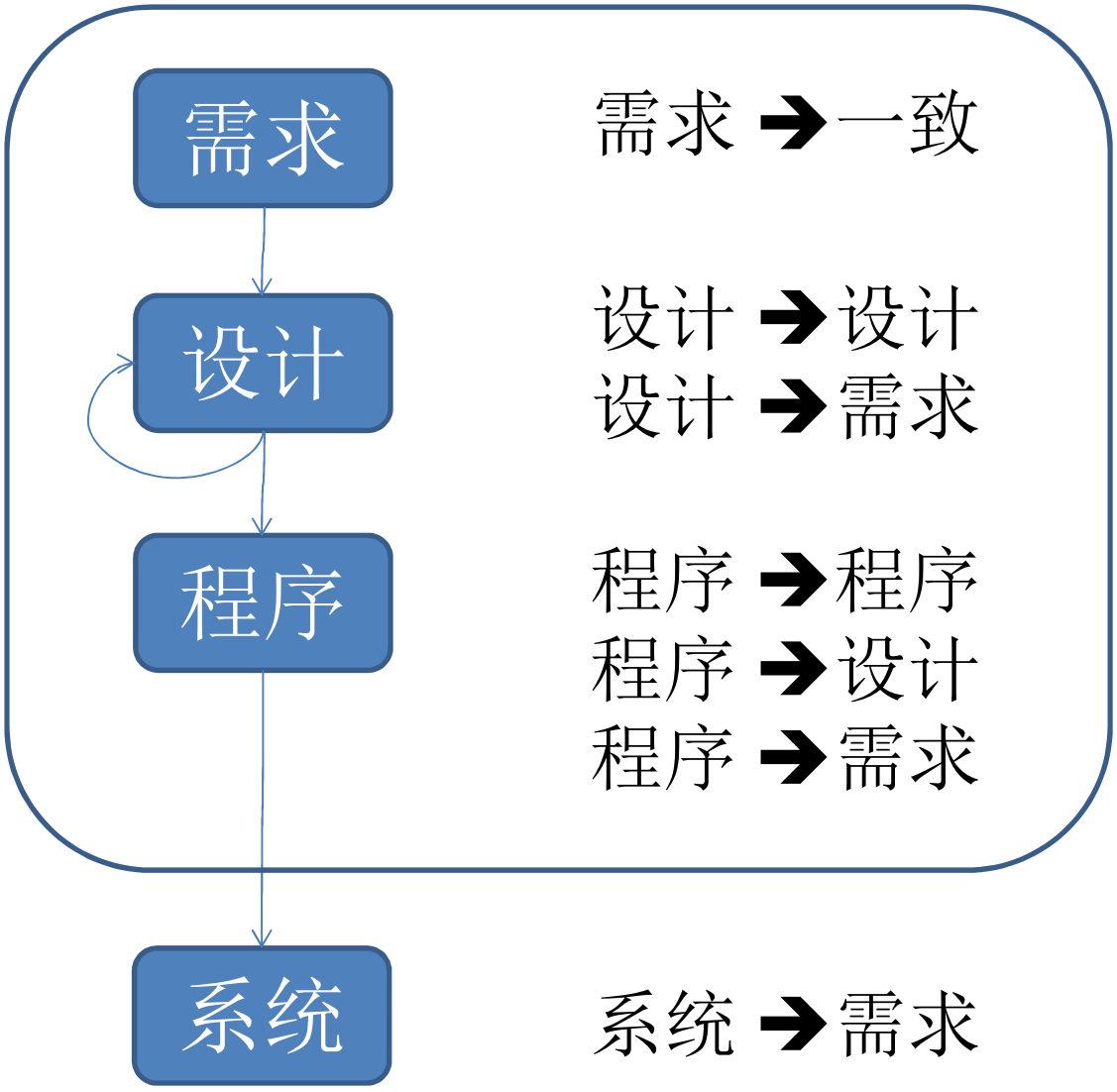
形式语言的描述

软件系统行为符合需求?

系统行为 ➔软件代码 ➔详细设计 ➔概要设计 ➔给定需求

可证明的软件正确性?

可证明的软件正确性



形式化方法：例子

由于软件系统的复杂性和形式化方法的局限性，
对于不同类型的系统，关注的侧重点有所不同。

- 顺序程序
- 反应式系统
- 并发程序

例子：顺序程序

- 顺序程序：不受干扰、一步一步做计算。
- 具体例子：假定我们需要在自然数四则运算的基础上计算阶乘。

阶乘

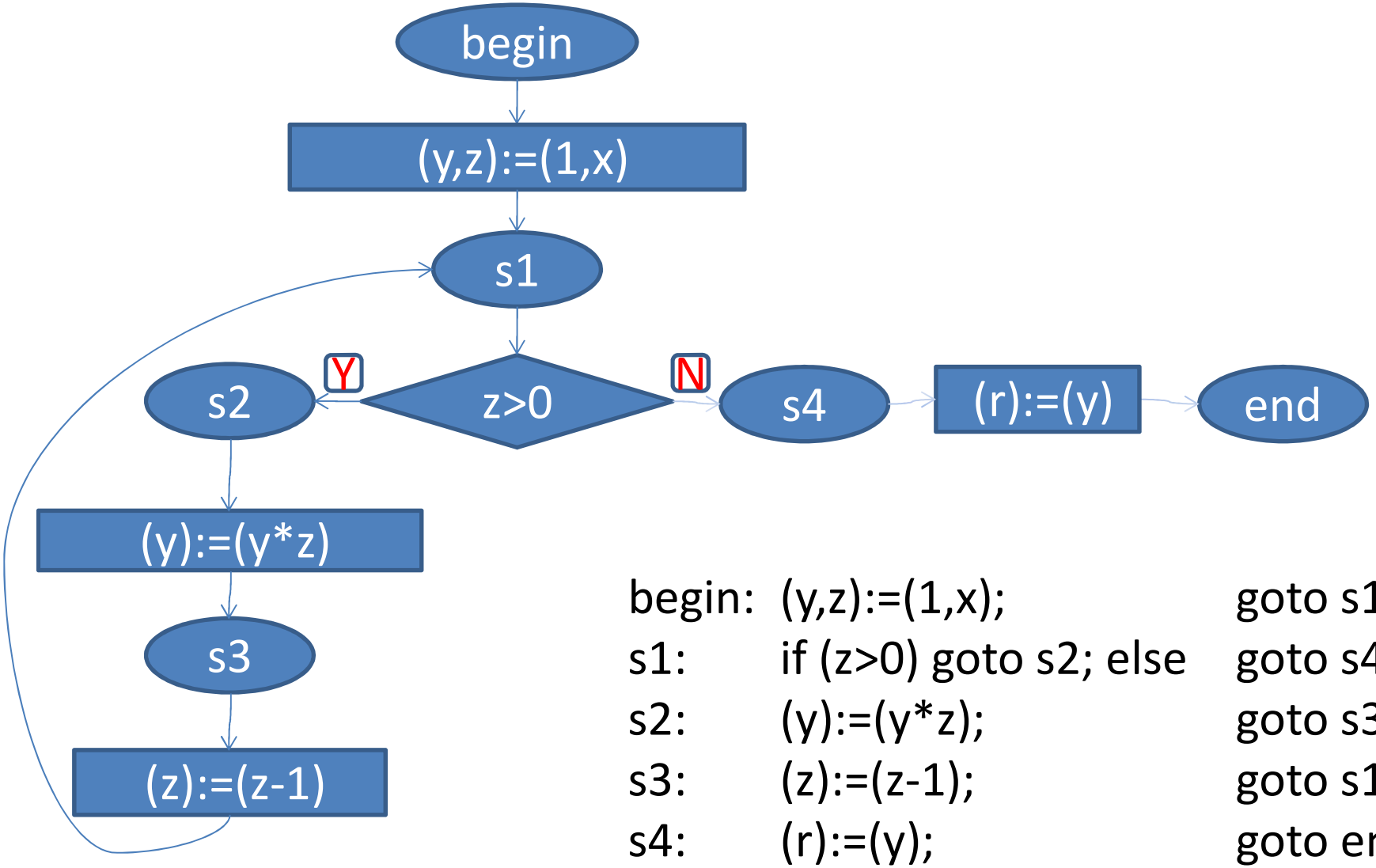
- 阶乘的定义如下：

> y 是 x 的阶乘，记做 y=x! 。
> y=x! 当且仅当 x=0且y=1或者(x=z+1)且(y=x*z!) 。

- 我们的需求如下：

> 输入x。
> 输出x!。

设计

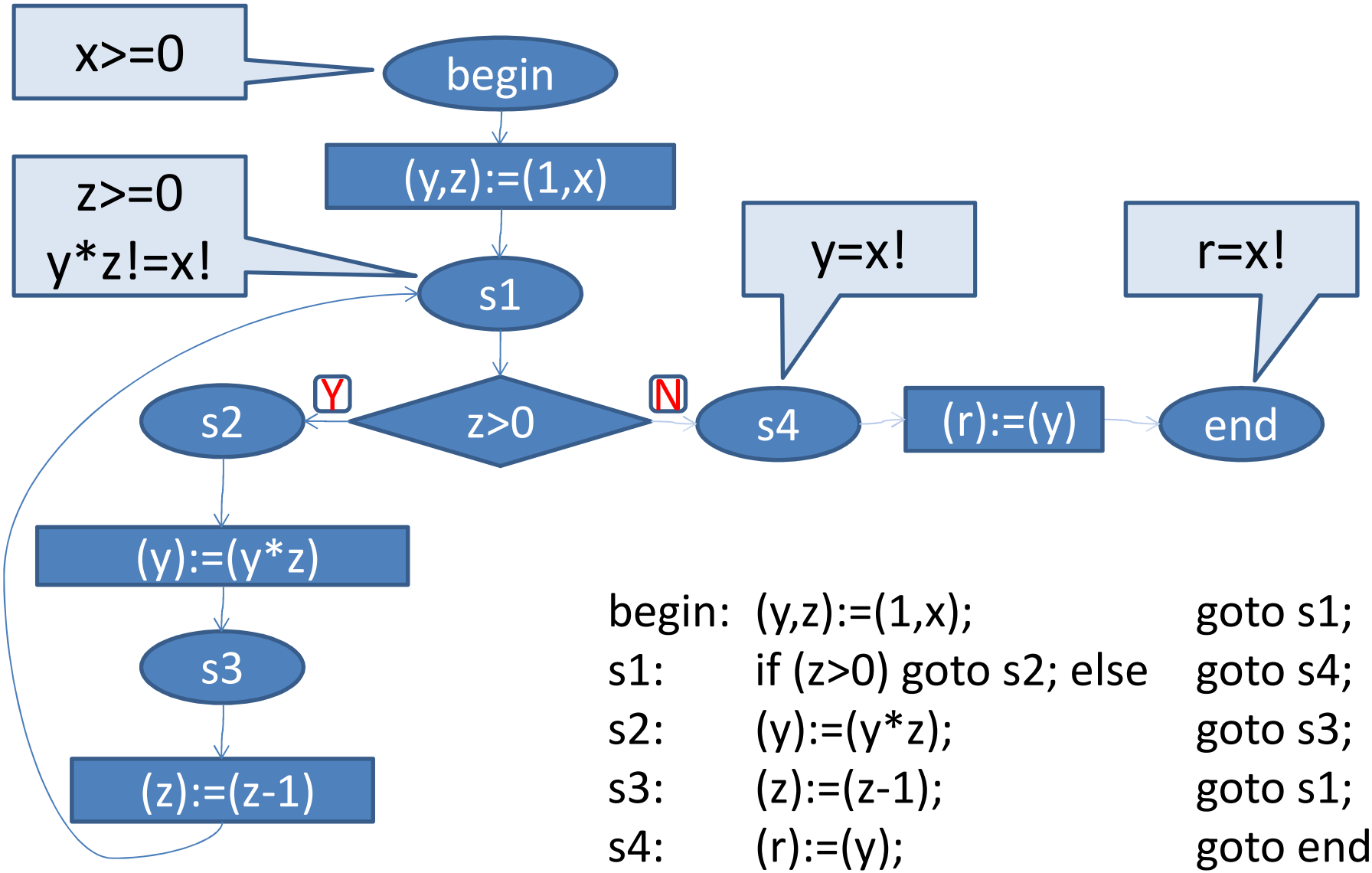



```
begin:  (y,z):=(1,x);                goto s1;
s1:     if (z>0) goto s2; else       goto s4;
s2:     (y):=(y*z);                  goto s3;
s3:     (z):=(z-1);                  goto s1;
s4:     (r):=(y);                    goto end;
```

正确性

- 正确性证明
  - 形式的流程设计语言
  - 形式的性质描述语言
  - 建立在形式语义上的推导规则

正确性




```
begin:  (y,z):=(1,x);              goto s1;
s1:     if (z>0) goto s2; else     goto s4;
s2:     (y):=(y*z);                goto s3;
s3:     (z):=(z-1);                goto s1;
s4:     (r):=(y);                  goto end;
```

正确性

- 该设计的正确性证明是可行的
  - 证明的是数学上的正确性
  - 软件实现的正确性受限制于整数规模的大小

程序代码(1)

```
int jc(int x)
{
   int y,z,r;
   begin:     y=1; z=x; goto s1;
   s1:        if (z>0) goto s2; else goto s4;
   s2:        y=y*z; goto s3;
   s3:        z=z-1; goto s1;
   s4:        r=y; goto end;
   end:       return r;
}
```

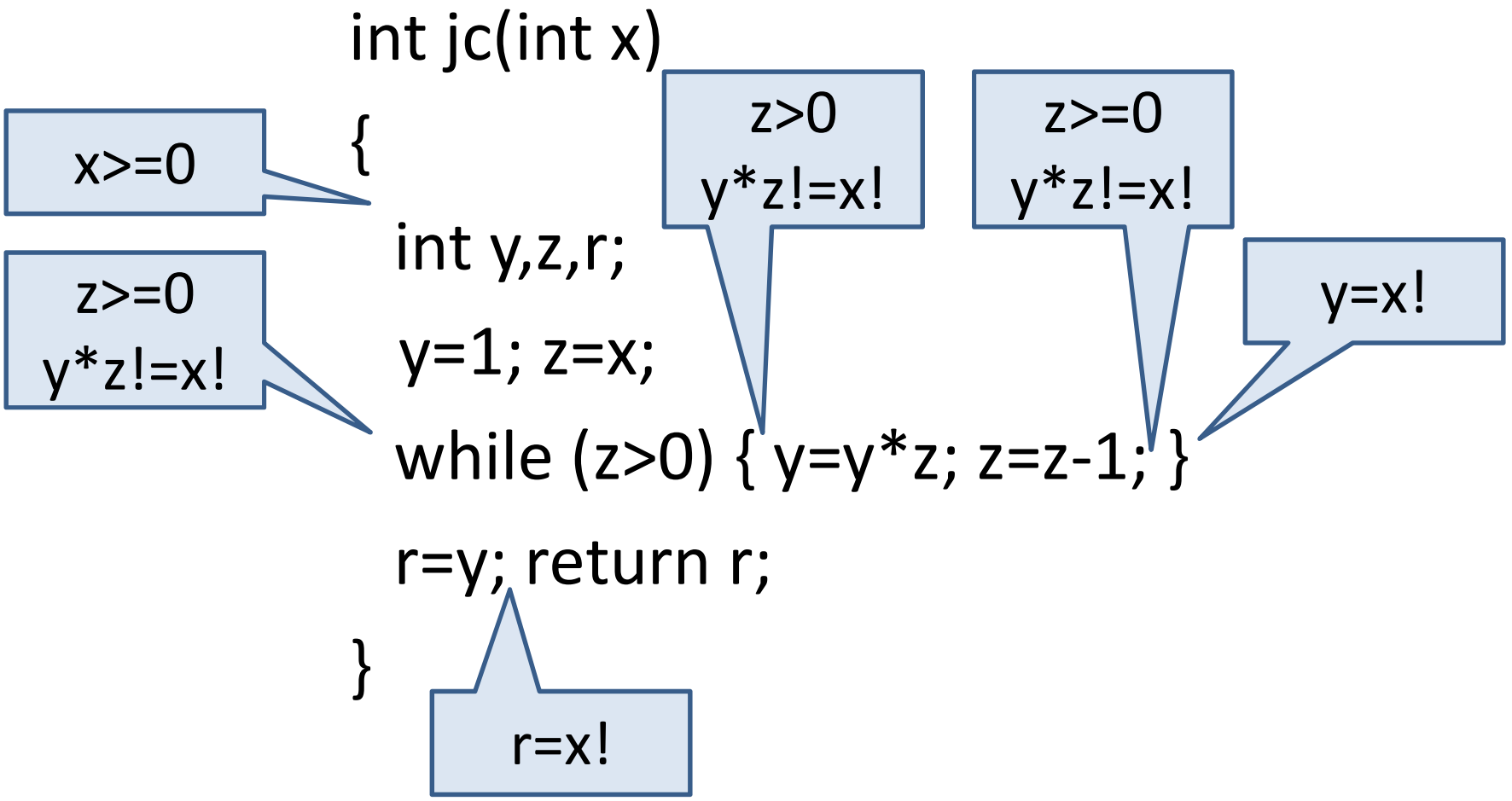
程序代码(2)

```
int jc(int x)
{
   int y,z,r;
   y=1; z=x;
   while (z>0) { y=y*z; z=z-1; }
   r=y; return r;
}
```

正确性

- 正确性证明
  - 形式的程序设计语言
  - 形式的性质描述语言
  - 建立在形式语义上的推导规则

正确性


int jc(int x)
{
int y,z,r;
y=1; z=x;
while (z>0) { y=y*z; z=z-1; }
r=y; return r;
}
x>=0
z>=0
y*z!=x!
z>0
y*z!=x!
z>=0
y*z!=x!
y=x!
r=x!

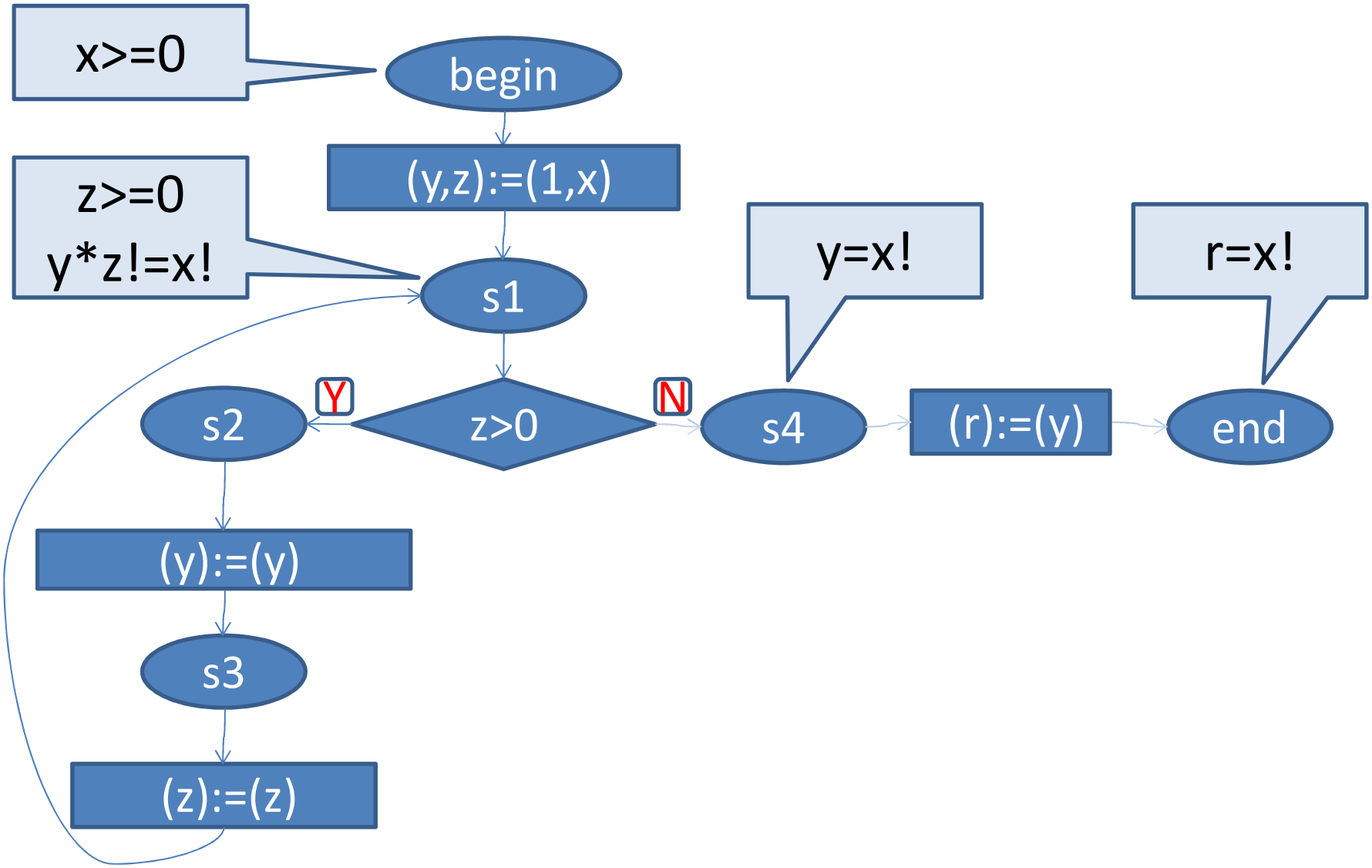
正确性

- 证明该程序的正确性是可行的
  - 证明的是数学上的正确性
  - 软件实现的正确性受限制于整数规模的大小

正确性

- 什么样的正确性?

正确性



部分正确性

- 在给定输入范围内程序运行终止时，程序运行所得到的结果符合要求。

- 上述例子：
- 如果jc(x)返回一个值，那么这个值是x!。

- 可以证明这个阶乘程序在以自然数为输入值的条件下是部分正确的。

终止性质

- 程序在给定输入范围内能够终止。

- 上述例子：
- 要求jc(x)必须能够返回一个值。

- 可以证明这个阶乘程序在以自然数为输入值的条件下是能够终止的。

完全正确性

- 在给定输入范围内程序能够终止并且得到的结果符合要求。

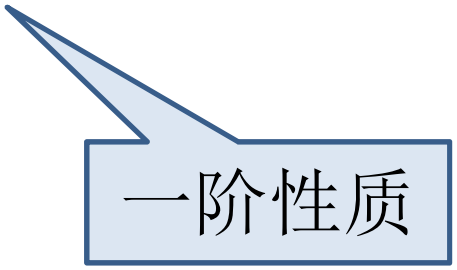
- 上述例子：
- 要求jc(x)必须返回一个值，且这个值是x!。

- 可以证明这个阶乘程序在以自然数为输入值的条件下是完全正确的，即部分正确且终止的。

正确性

- 什么样的正确性？

- 部分正确性
- 终止性质
- 完成正确性 = 部分正确+终止



- 安全性质

安全性质

- 安全性质是一种要求程序在运行的任何时候都满足的性质。
- 抽象地说，在程序运行过程中能够确定性质被违反的这一类性质称为安全性质。
- 关于命题p的安全性质，记为AG(p)。

安全性质

- 从以上例子看，我们希望程序变量的值在任何时候都在0到2147483647之间，从而保证程序在普通32位计算机的运行环境下不会产生整数溢出的问题。

- 定义q1(x)为(0≤x≤2147483647)。

- 前述安全性质为AG(q1(x)∧ q1(y)∧q1(z) ∧q1(r))。

- 可以证明当输入x在0和12之间时，这个安全性质是满足的。

正确性相关问题

- 设计 ➔ 需求
- 编码 ➔ 设计
- 编码 ➔ 需求
- 编码1 ≈ 编码2

形式化方法：例子

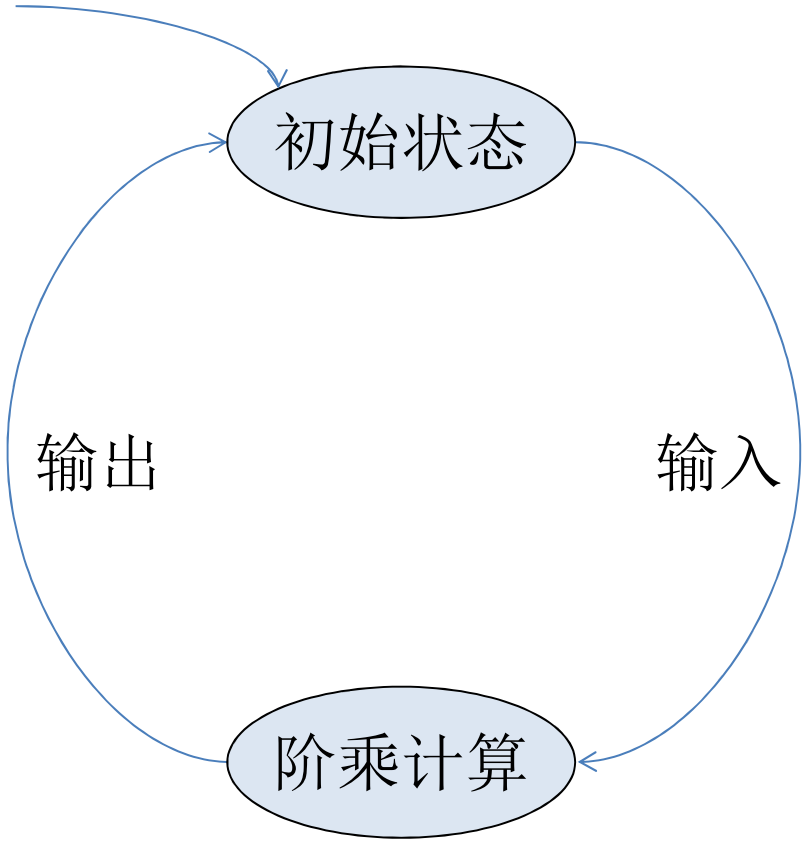
- 顺序程序
- 反应式系统
- 并发程序

例子：反应式系统

- 反应式系统：与外界交互、处理外界的输入。

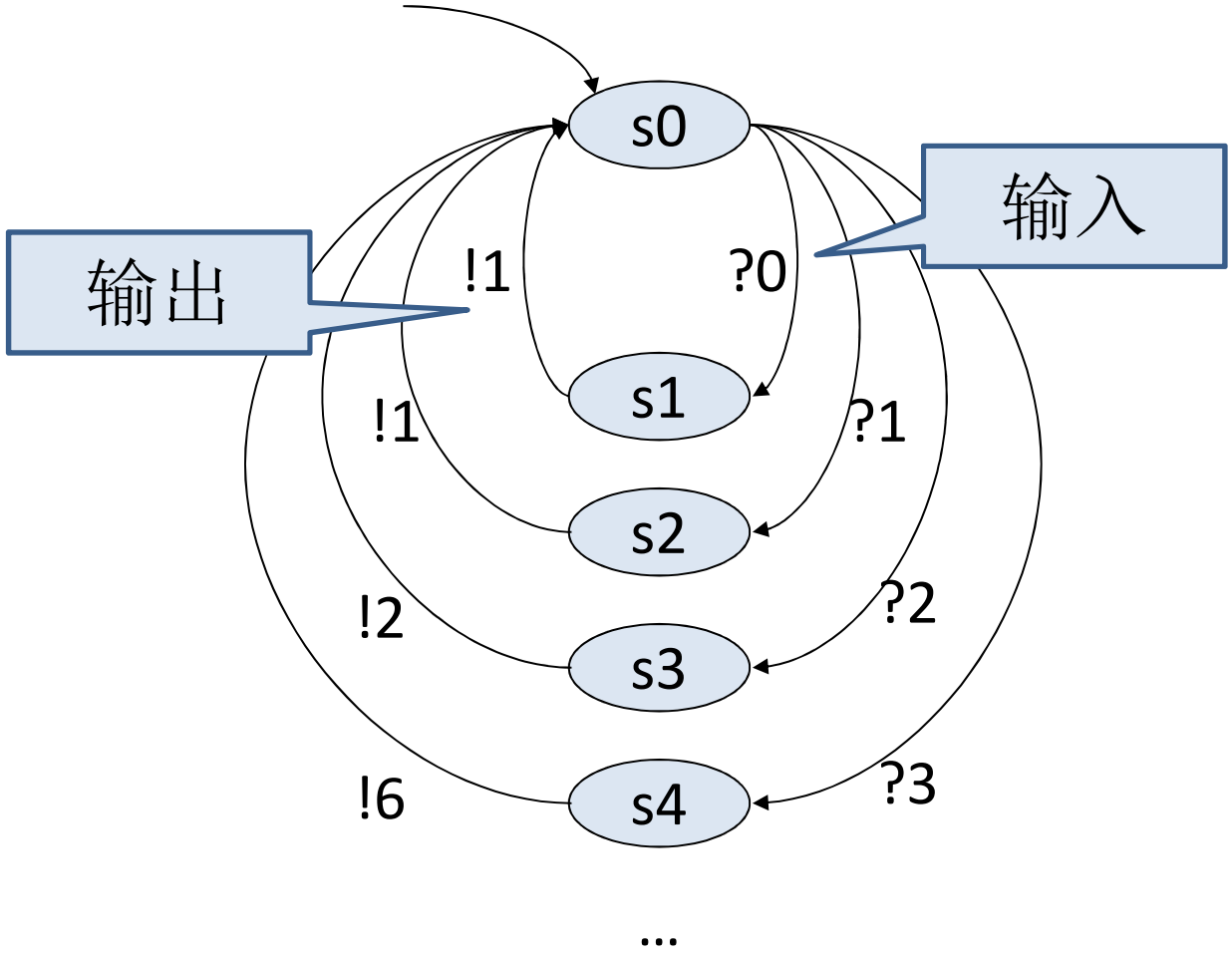
- 从设计的角度，我们关心的是输入动作和对输入的处理的抽象流程，具体内容的处理方式可使用顺序程序来进行。

阶乘服务器



对用户的输入是否有相应的输出?

阶乘服务器



自动售茶机

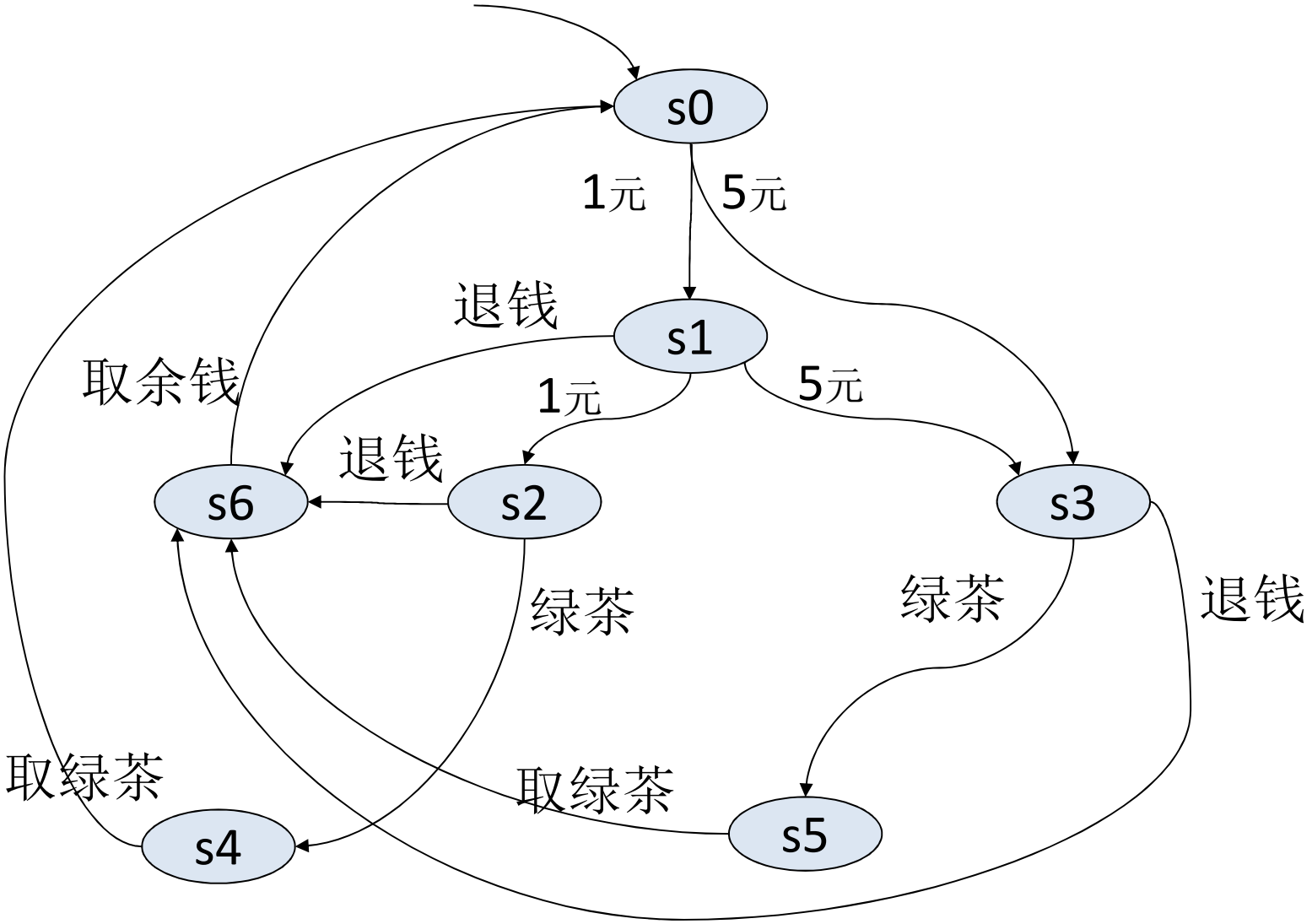
- 售绿茶(价格2元)。
- 输入动作包括：

> 1元、5元、绿茶、退钱、取绿茶、取余钱。

- 部分需求如下：

> 投入足够多钱后不能再投钱。
> 投入足够多钱后可以选择绿茶。
> 投入钱后可以马上选择退钱。

设计



正确性

- 正确性证明
  - 形式的设计语言
  - 形式的性质描述语言
  - 建立在形式语义和图算法上的推理方法

- 可以证明我们的设计满足需求

自动售茶机(设计的细化)

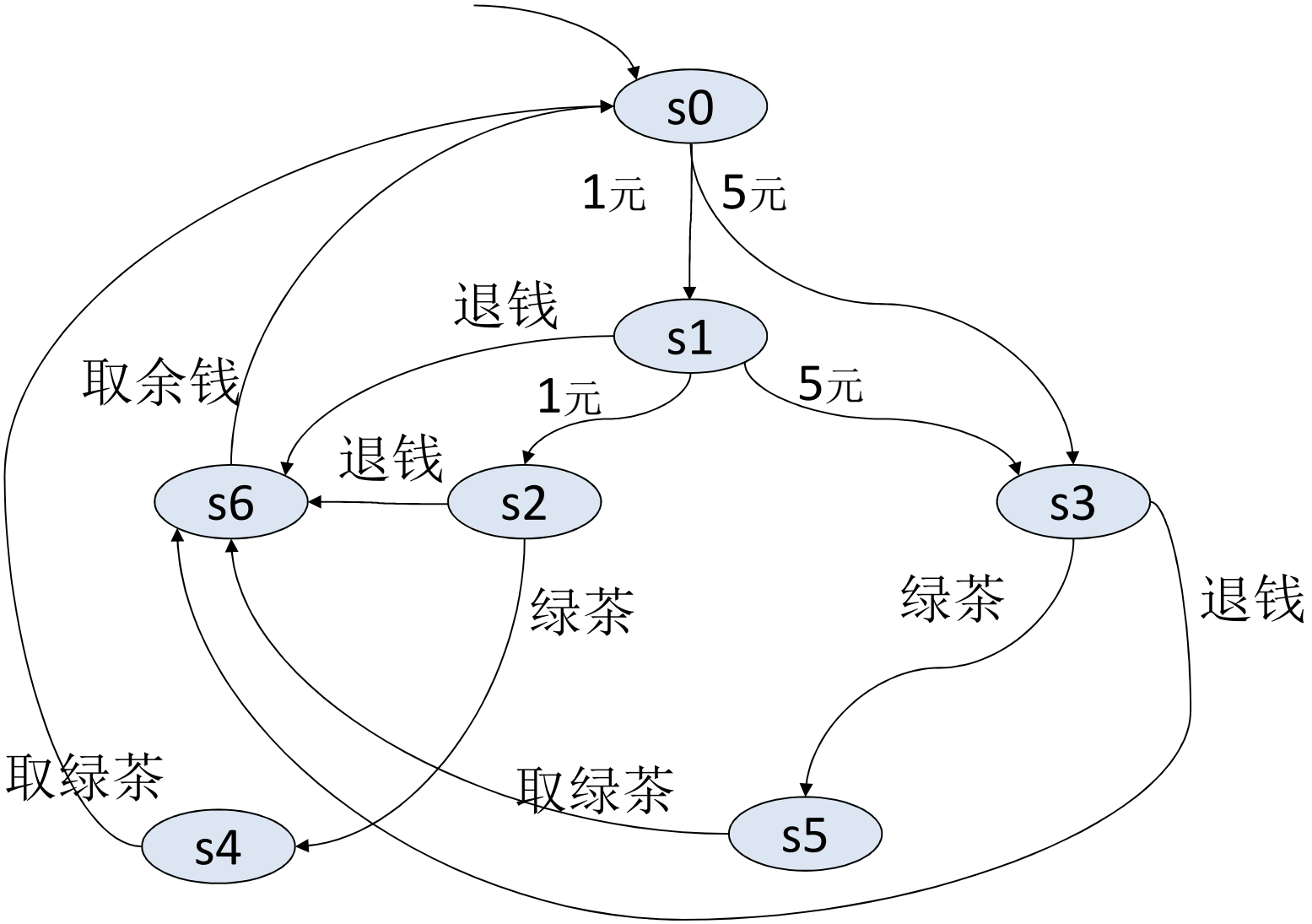
- 如果我们需要知道退多少钱，我们可先对动作进行细化，然后细化我们的设计。
- 输入动作包括：

> 1元、5元、绿茶、退钱、取绿茶、
> 取1元、取2元、取3元、取4元、取5元、取6元。

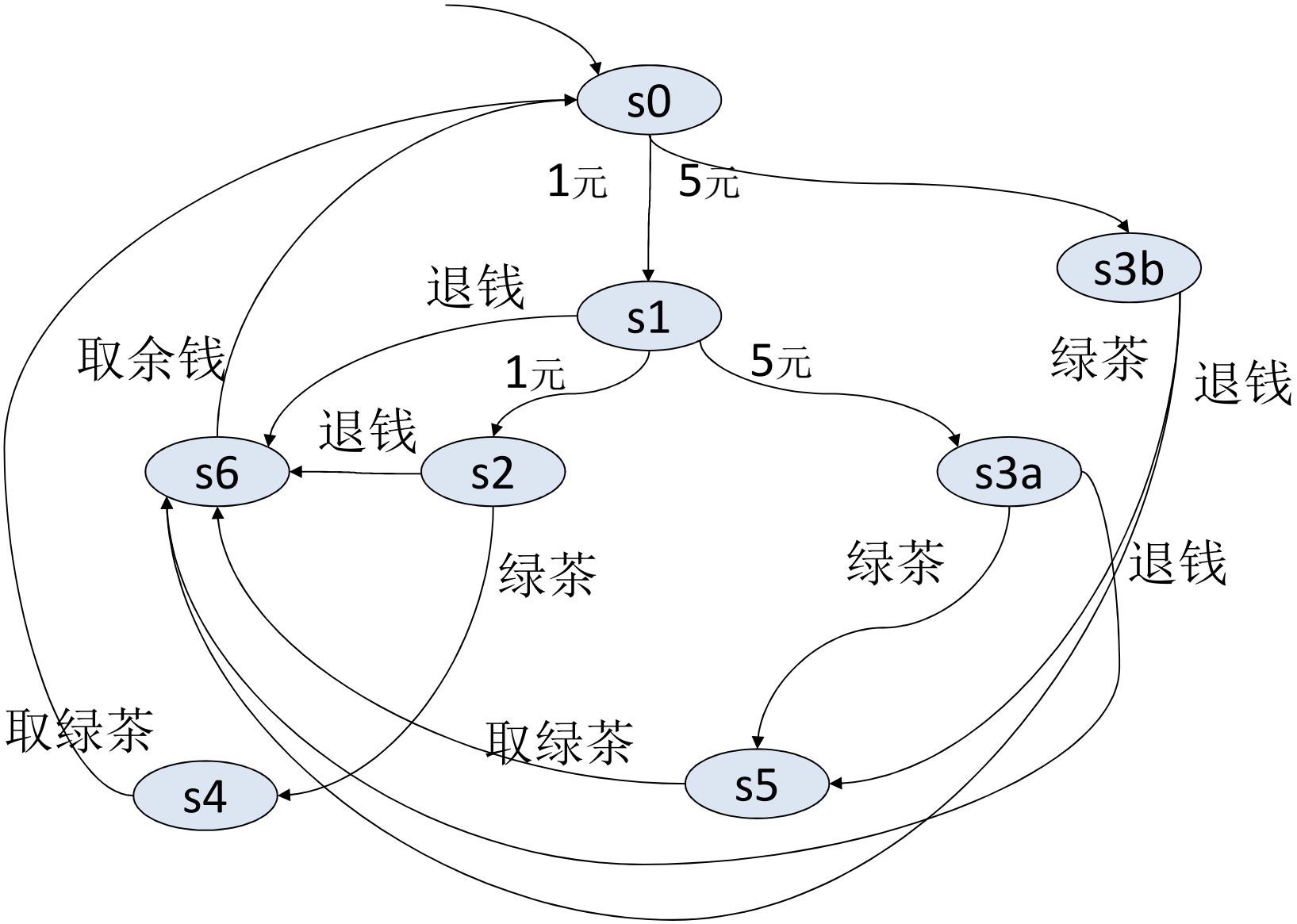
- 部分增加的需求如下：

> 投入1元和5元后选择绿茶，可以取回4元。
> 投入1元和5元后选择退钱，可以取回6元。

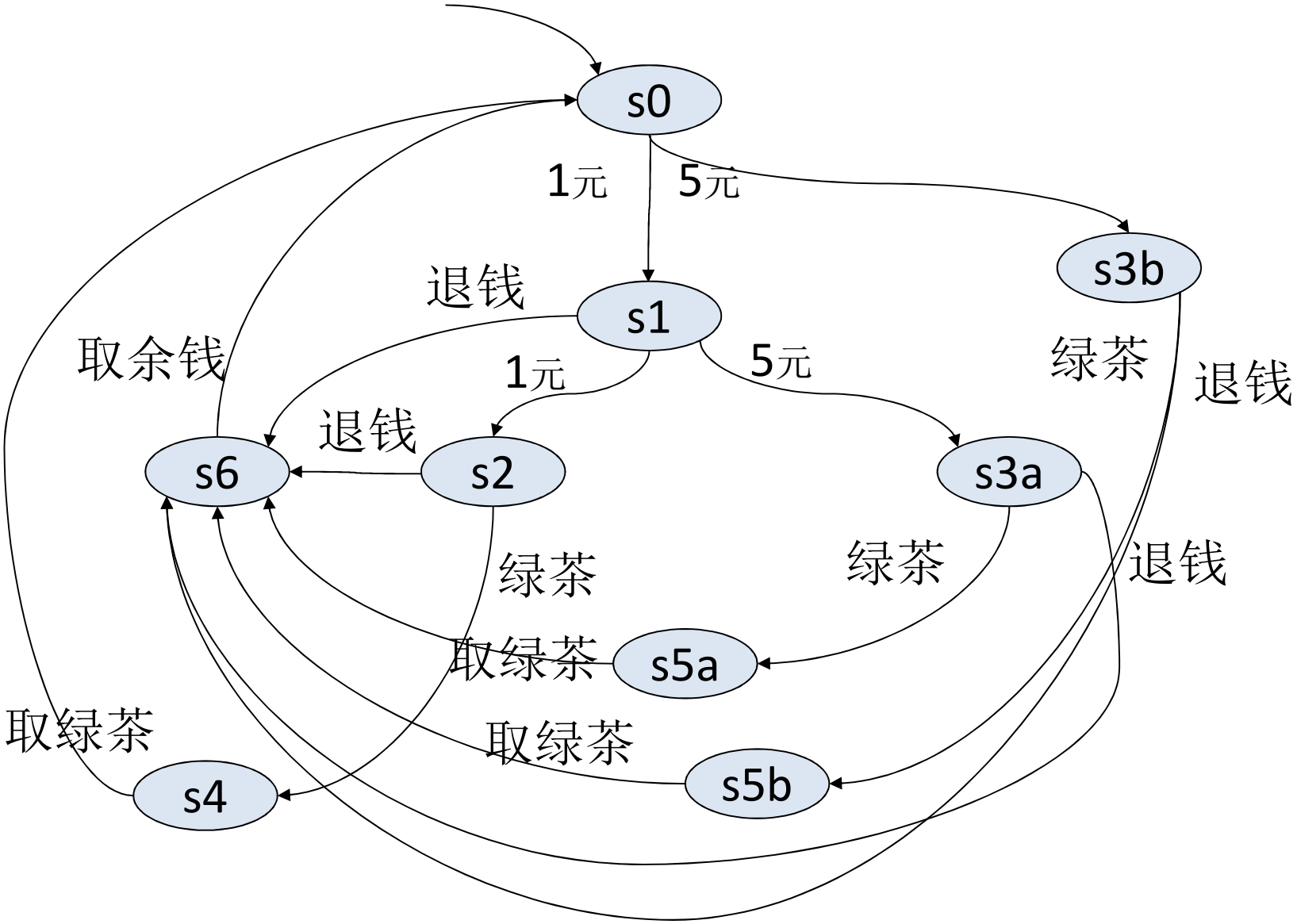
设计(0)



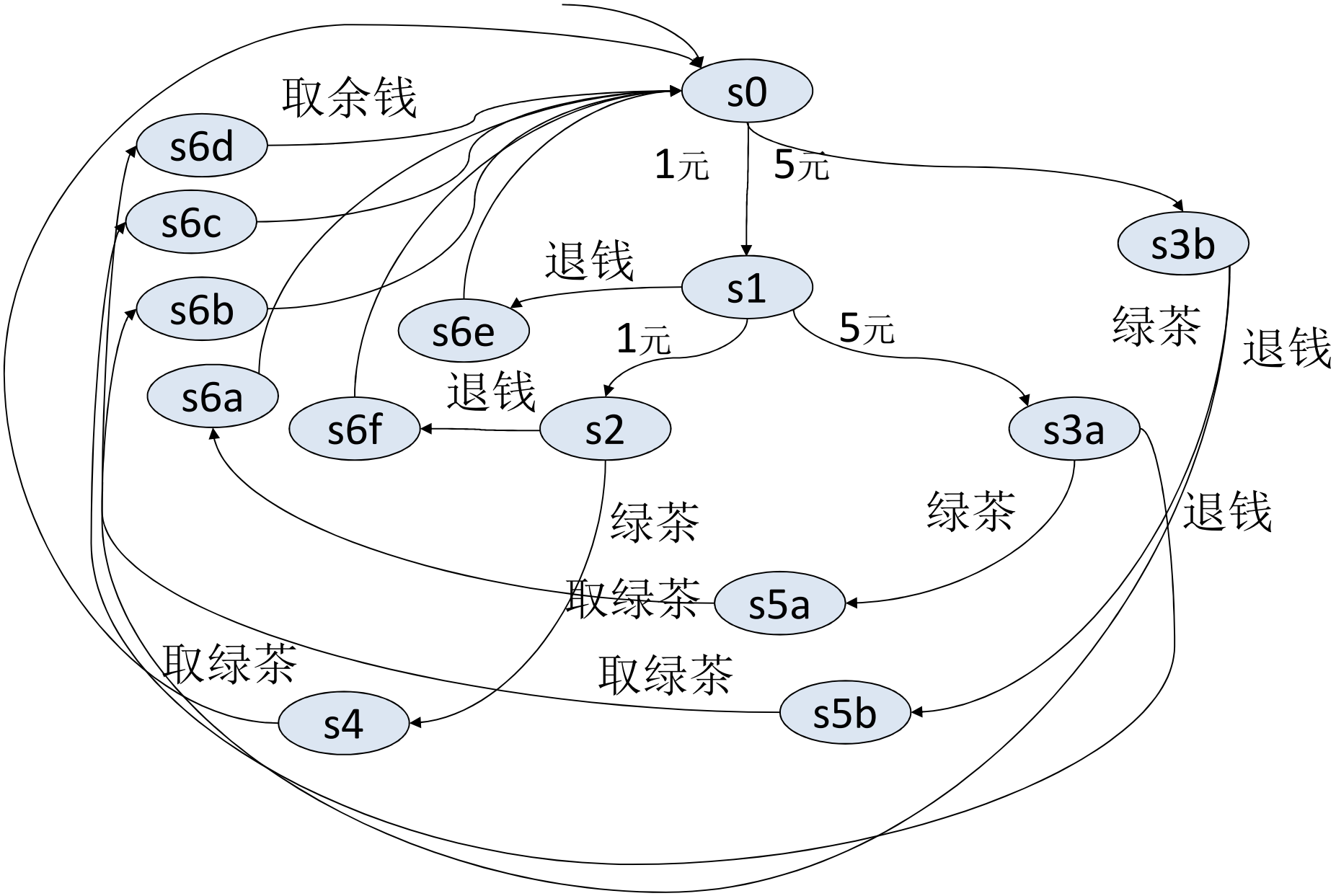
设计(1)



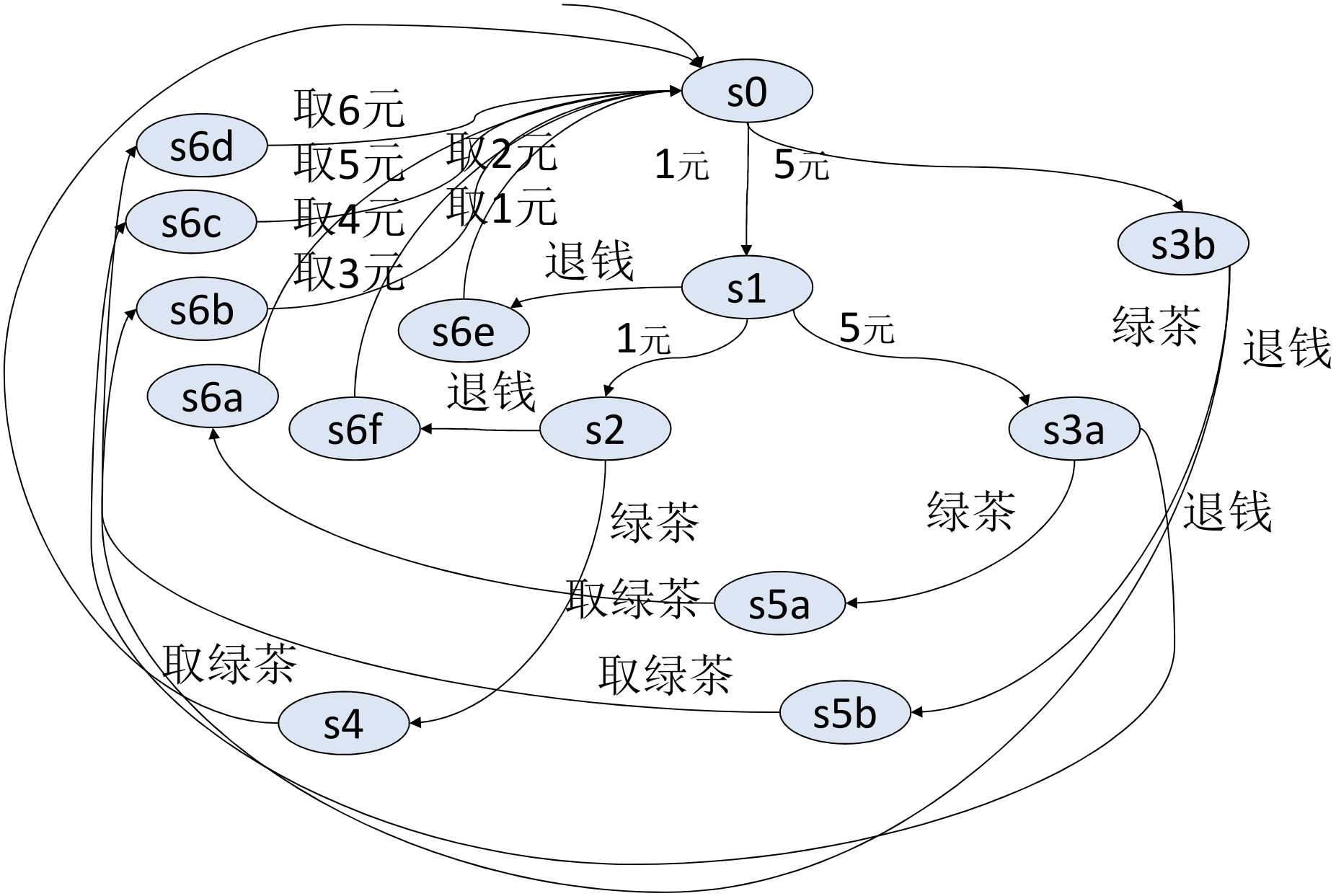
设计(2)


s0
1元
5元
s3b
退钱
s1
取余钱
1元
5元
绿茶
退钱
退钱
s6
s2
s3a
绿茶
绿茶
退钱
取绿茶
s5a
取绿茶
取绿茶
s4
s5b

设计(3)


s0
s6d
s6c
s6b
s6a
s6e
s6f
s1
s2
s3b
s3a
s4
s5a
s5b
取余钱
1元
5元
退钱
1元
5元
退钱
绿茶
退钱
退钱
绿茶
绿茶
取绿茶
取绿茶
取绿茶

设计(4)


s0
s6d
s6c
s6b
s6a
s6e
s6f
s1
s2
s3a
s3b
s4
s5a
s5b
取6元
取5元
取4元
取3元
取2元
取1元
1元
5元
退钱
1元
5元
退钱
绿茶
退钱
绿茶
绿茶
退钱
取绿茶
取绿茶
取绿茶

正确性

- 正确性证明
  - 形式的设计语言
  - 形式的性质描述语言
  - 建立在形式语义和图算法上的推理方法

- 可以证明我们的设计满足需求

- 设计的细化可以用模拟关系来说明
- 可以证明这些设计之间存在模拟关系

形式化方法：例子

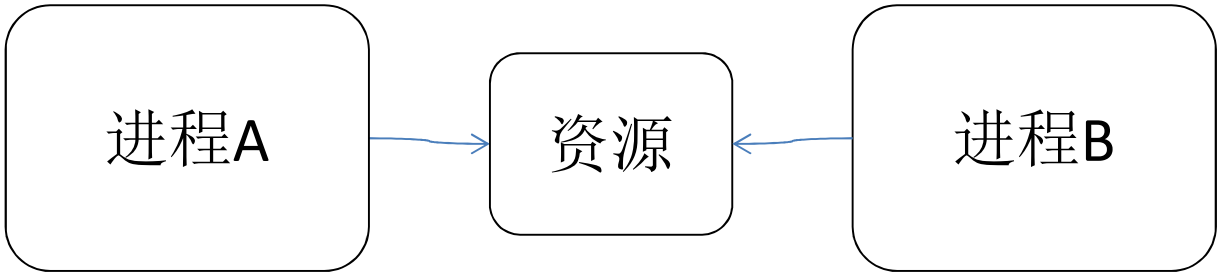
- 顺序程序
- 反应式系统
- 并发程序

例子：并发程序

- 并发程序：由多个顺序程序组成的一起运行、通过共享变量或通信通道互相影响的程序系统。
- 这里所说的程序通常称为进程，这样的系统通常称为并发系统。

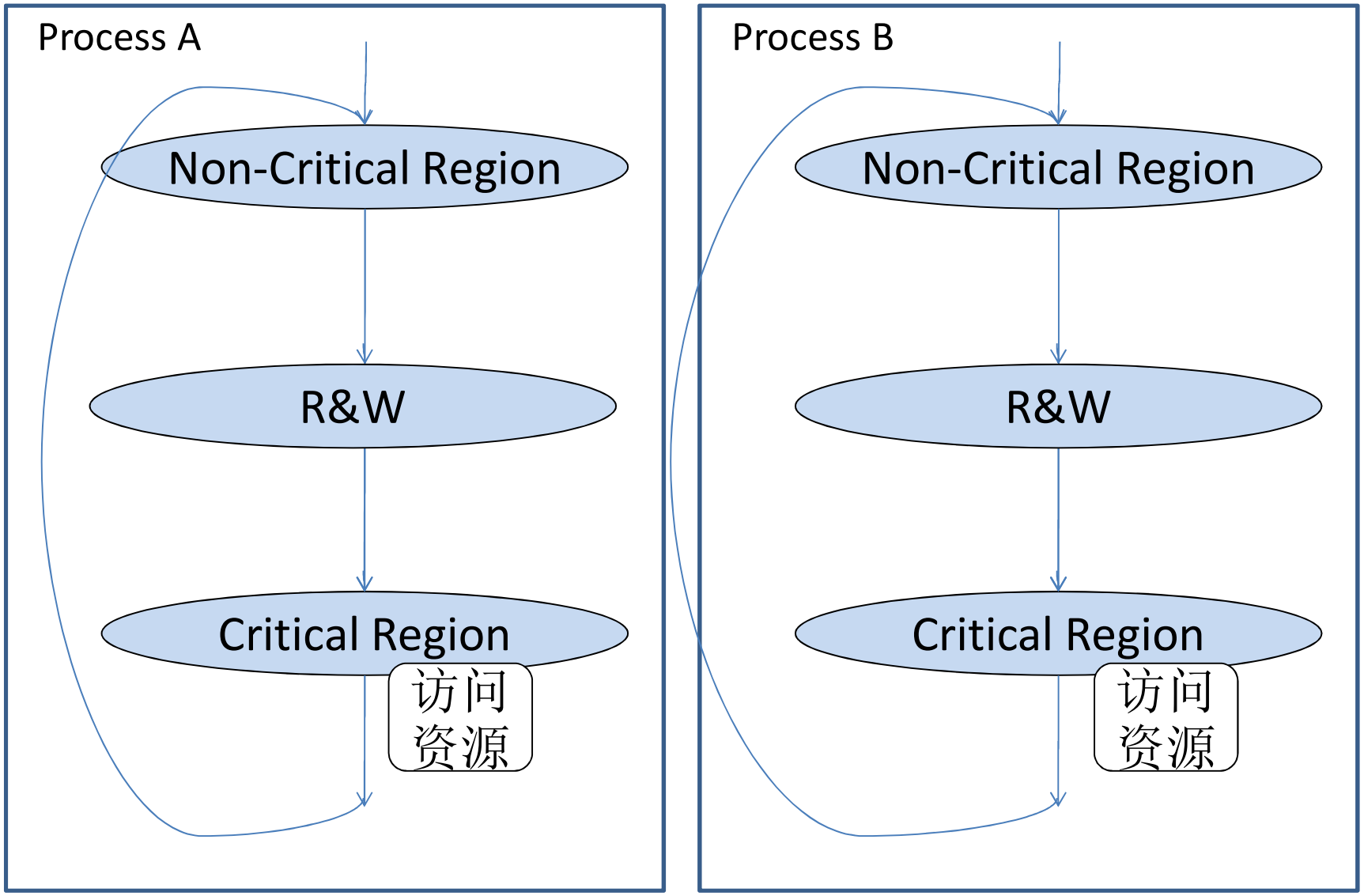
例子：资源共享与互斥使用

- 假定我们需要设计一个由两个可使用共享变量的和公共资源的进程组成的并发系统。

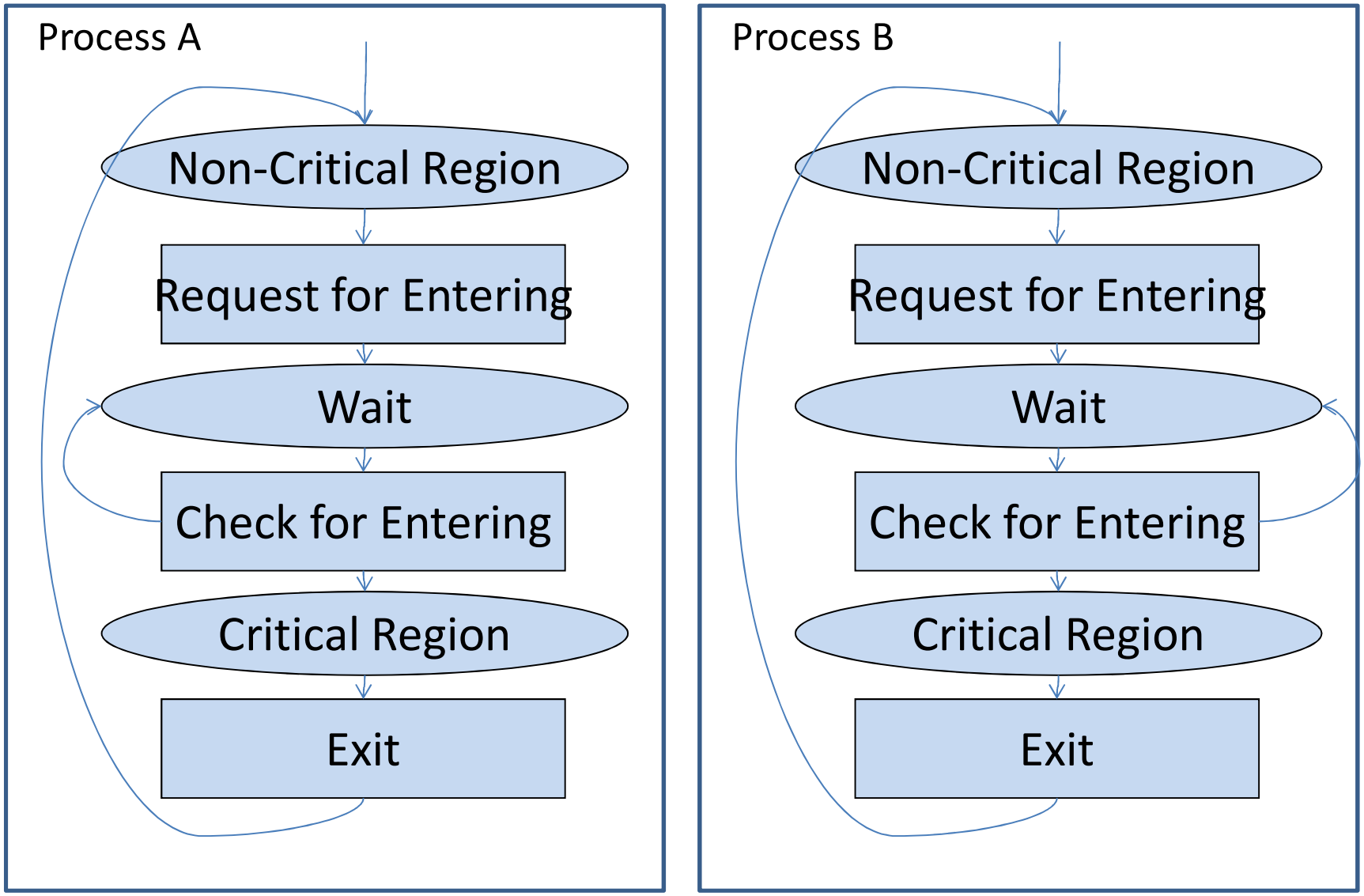


- 我们需要一个称为互斥算法的机制以控制公共资源的使用，使得不能两个进程同时访问该资源。

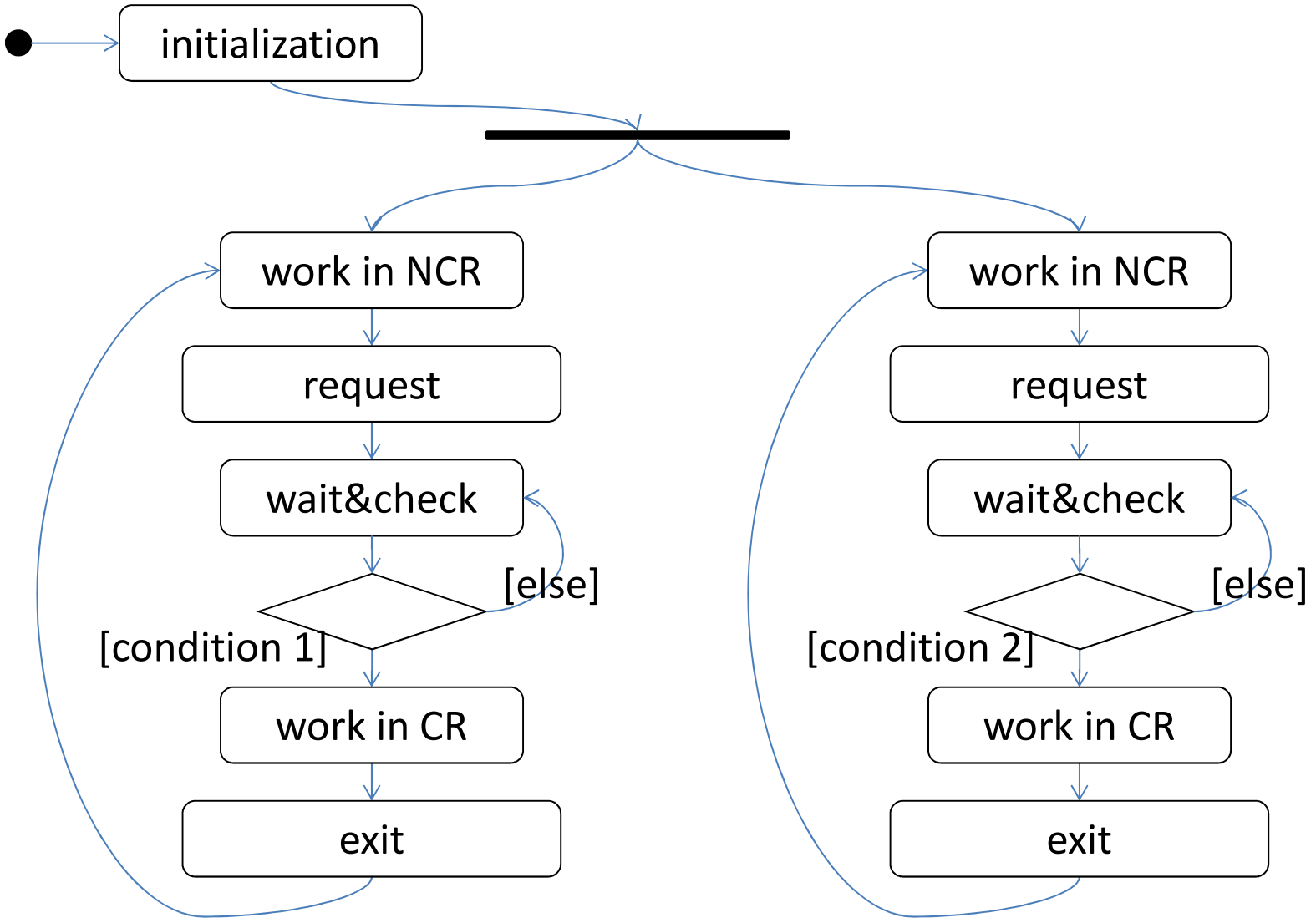
设计


Process A
Non-Critical Region
R&W
Critical Region
访问
资源
Process B
Non-Critical Region
R&W
Critical Region
访问
资源

设计


Process A
Non-Critical Region
Request for Entering
Wait
Check for Entering
Critical Region
Exit
Process B
Non-Critical Region
Request for Entering
Wait
Check for Entering
Critical Region
Exit

设计(活动图)


initialization
work in NCR
request
wait&check
[else]
[condition 1]
work in CR
exit
work in NCR
request
wait&check
[else]
[condition 2]
work in CR
exit

设计(共享变量)

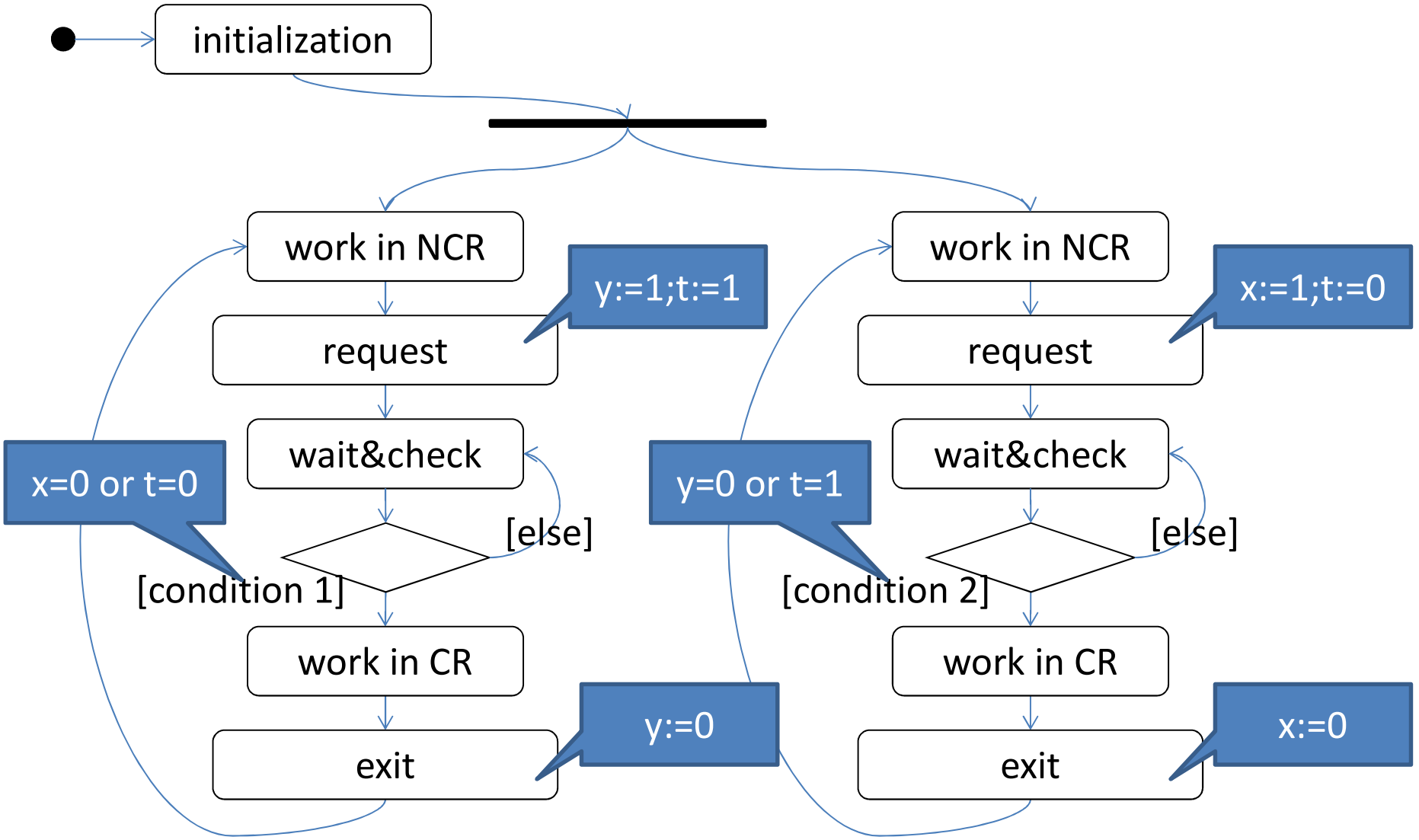
目标：至多一个进程可以在CR状态

机制：使用共享变量

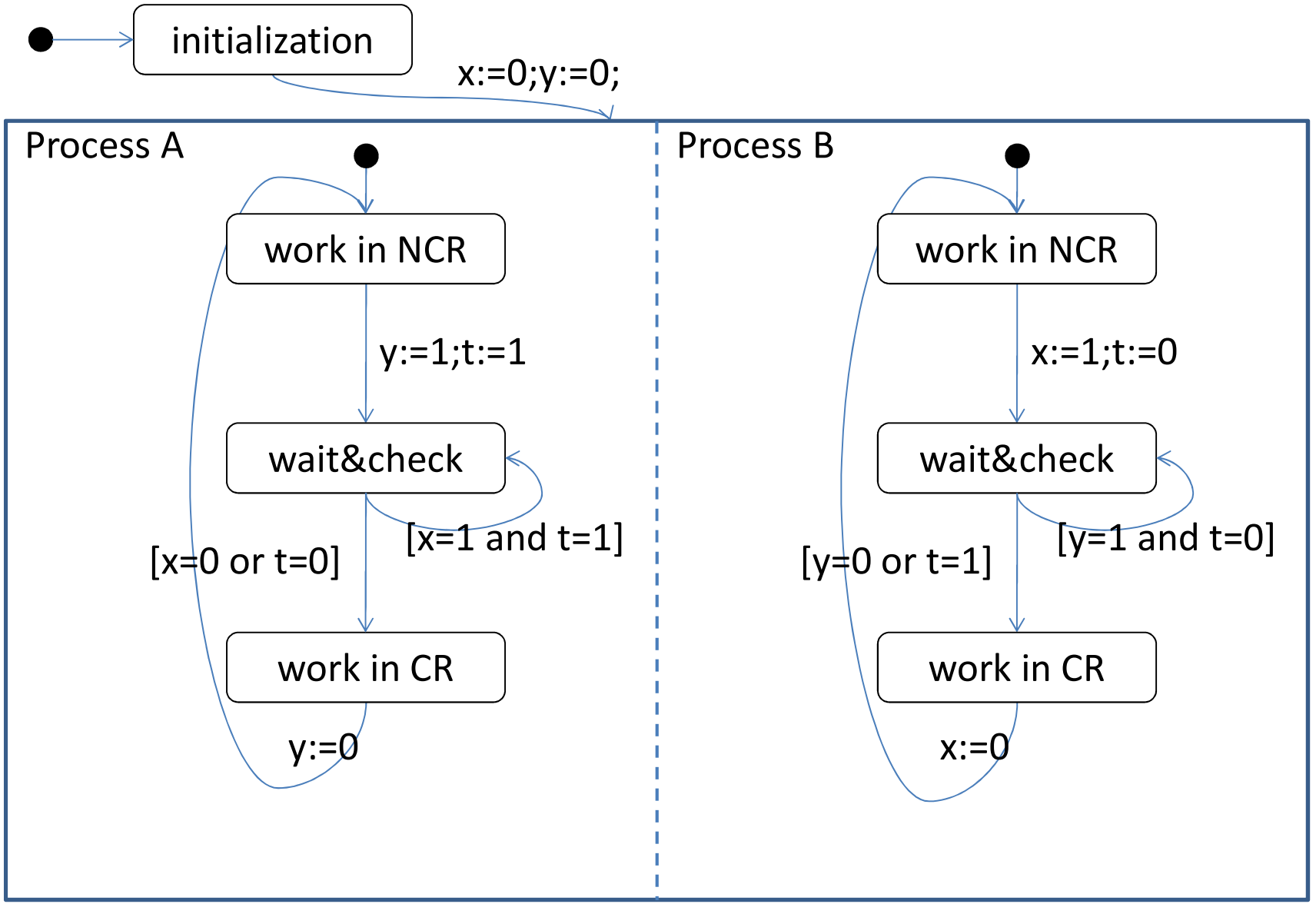
y=1:　　进程A 在申请进入CR 或在CR状态
x=1:　　进程B 在申请进入CR 或在CR状态

t=0:　　A 有进入CR 的优先权
t=1:　　B 有进入CR 的优先权

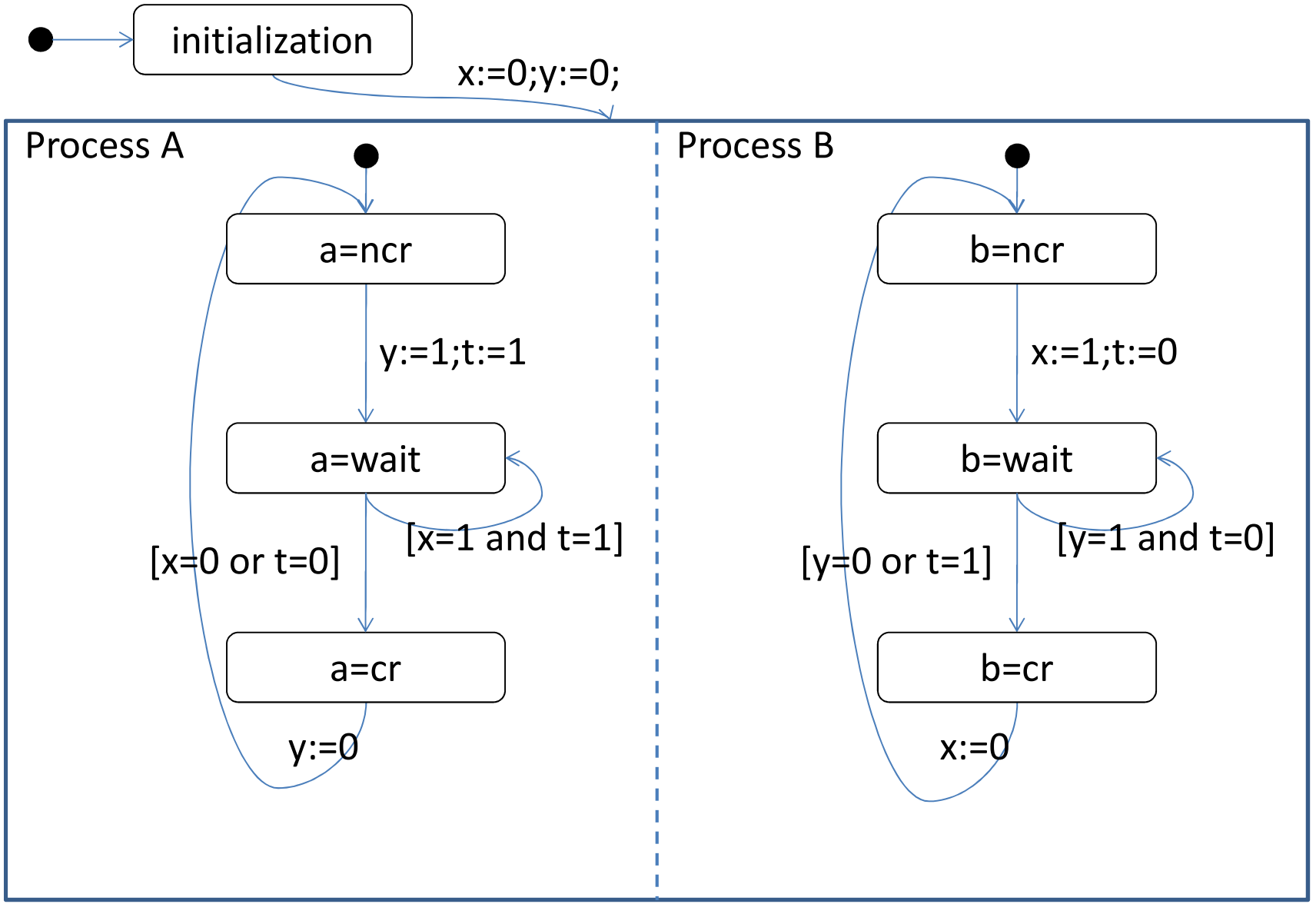
设计(活动图)


initialization
work in NCR
request
wait&check
[else]
[condition 1]
work in CR
exit
y:=1;t:=1
x=0 or t=0
y:=0
work in NCR
request
wait&check
[else]
[condition 2]
work in CR
exit
x:=1;t:=0
y=0 or t=1
x:=0

设计(状态图)


initialization
x:=0;y:=0;
Process A
work in NCR
y:=1;t:=1
wait&check
[x=1 and t=1]
[x=0 or t=0]
work in CR
y:=0
Process B
work in NCR
x:=1;t:=0
wait&check
[y=1 and t=0]
[y=0 or t=1]
work in CR
x:=0

设计(状态图)


initialization
x:=0;y:=0;
Process A
a=ncr
y:=1;t:=1
a=wait
[x=1 and t=1]
[x=0 or t=0]
a=cr
y:=0
Process B
b=ncr
x:=1;t:=0
b=wait
[y=1 and t=0]
[y=0 or t=1]
b=cr
x:=0

设计(算法)

VAR: a: {ncr,wait,cr}; b: {ncr,wait,cr}; x: 0..1; y: 0..1; t: 0..1;
INIT: a=ncr; b=ncr; x=0; y=0;

Process A:

a=ncr	→ (a,y,t):=(wait,1,1);
a=wait∧(x=0∨t=0)	→ (a):=(cr);
a=wait∧¬(x=0∨t=0)	→ (a):=(wait);
a=cr	→ (a,y):=(ncr,0);

Process B:

b=ncr	→ (b,x,t):=(wait,1,0);
b=wait∧(y=0∨t=1)	→ (b):=(cr);
b=wait∧¬(y=0∨t=1)	→ (b):=(wait);
b=cr	→ (b,x):=(ncr,0);

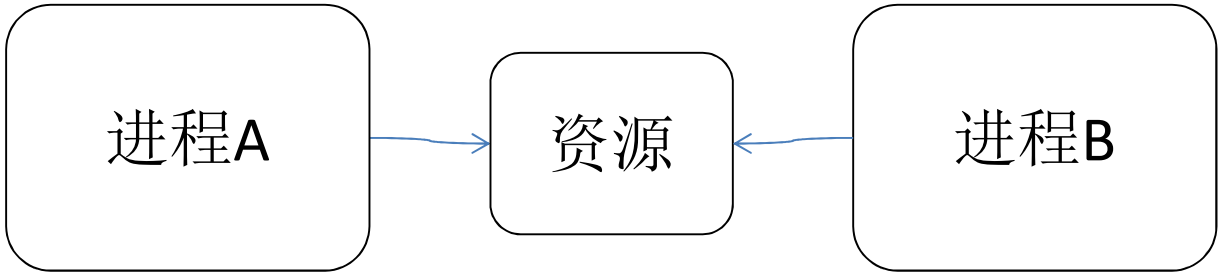
互斥性质：AG(¬((a=cr) ∧ (b=cr))

正确性

- 正确性证明
  - 形式的设计语言
  - 形式的性质描述语言
  - 建立在形式语义和图算法上的推理方法

- 可以证明以上并发系统满足 $AG(\neg((a=cr) \wedge (b=cr))$

正确性相关问题

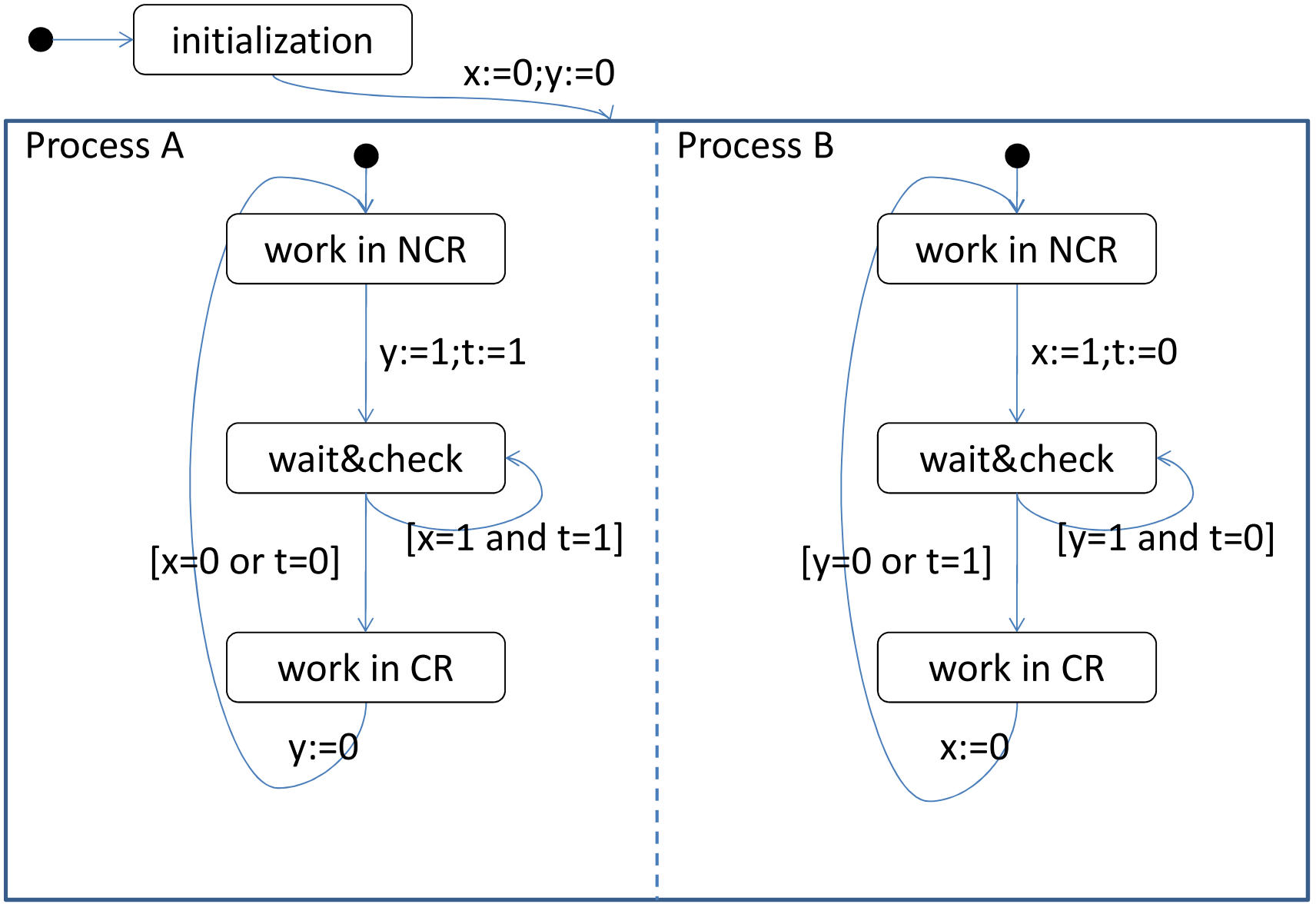


算法➔需求？

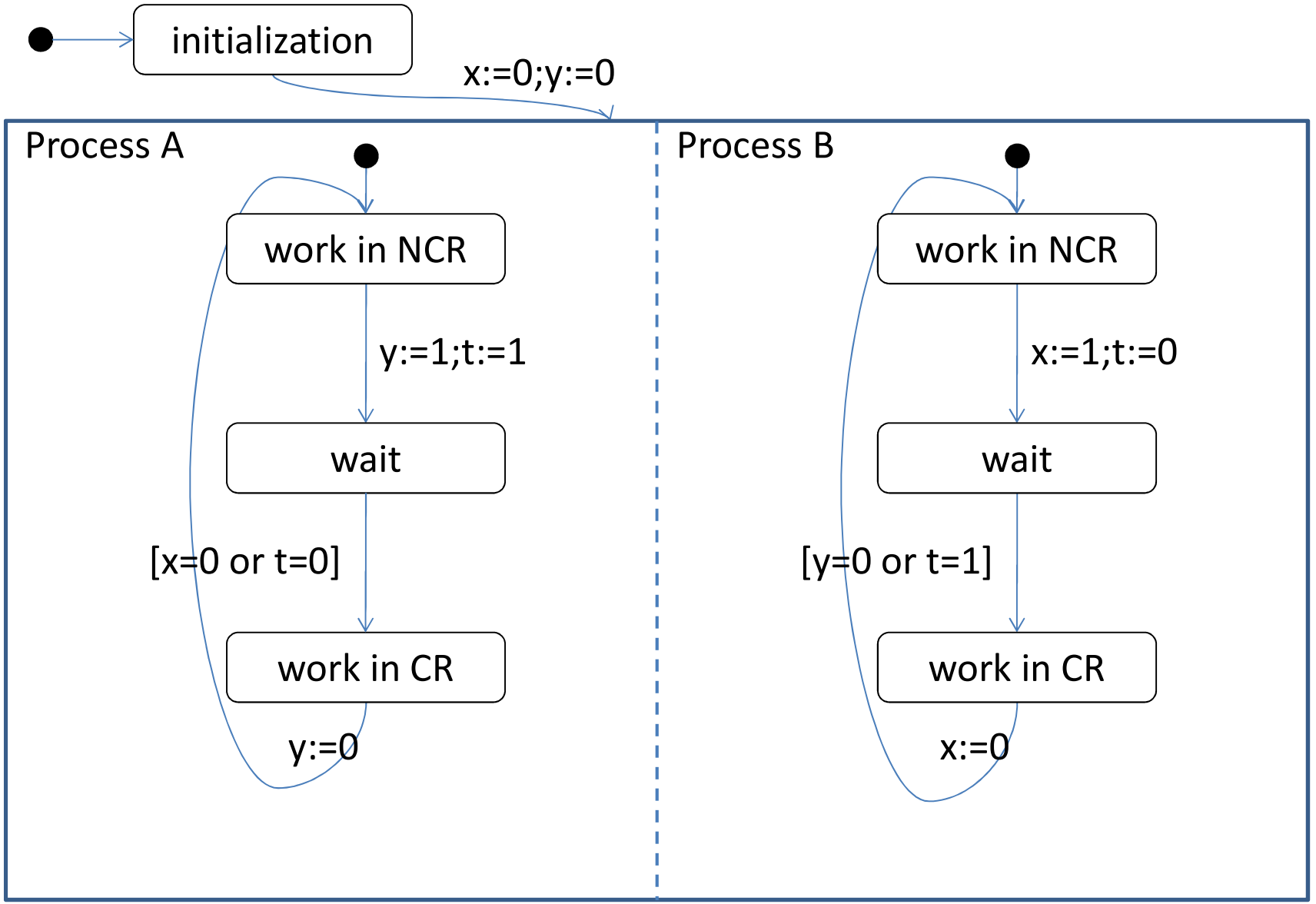
需求的完整性？

- 互斥？
- 若有进程申请资源，则该进程一定有机会使用资源？
- 若有进程申请资源，则资源一定有机会被使用？

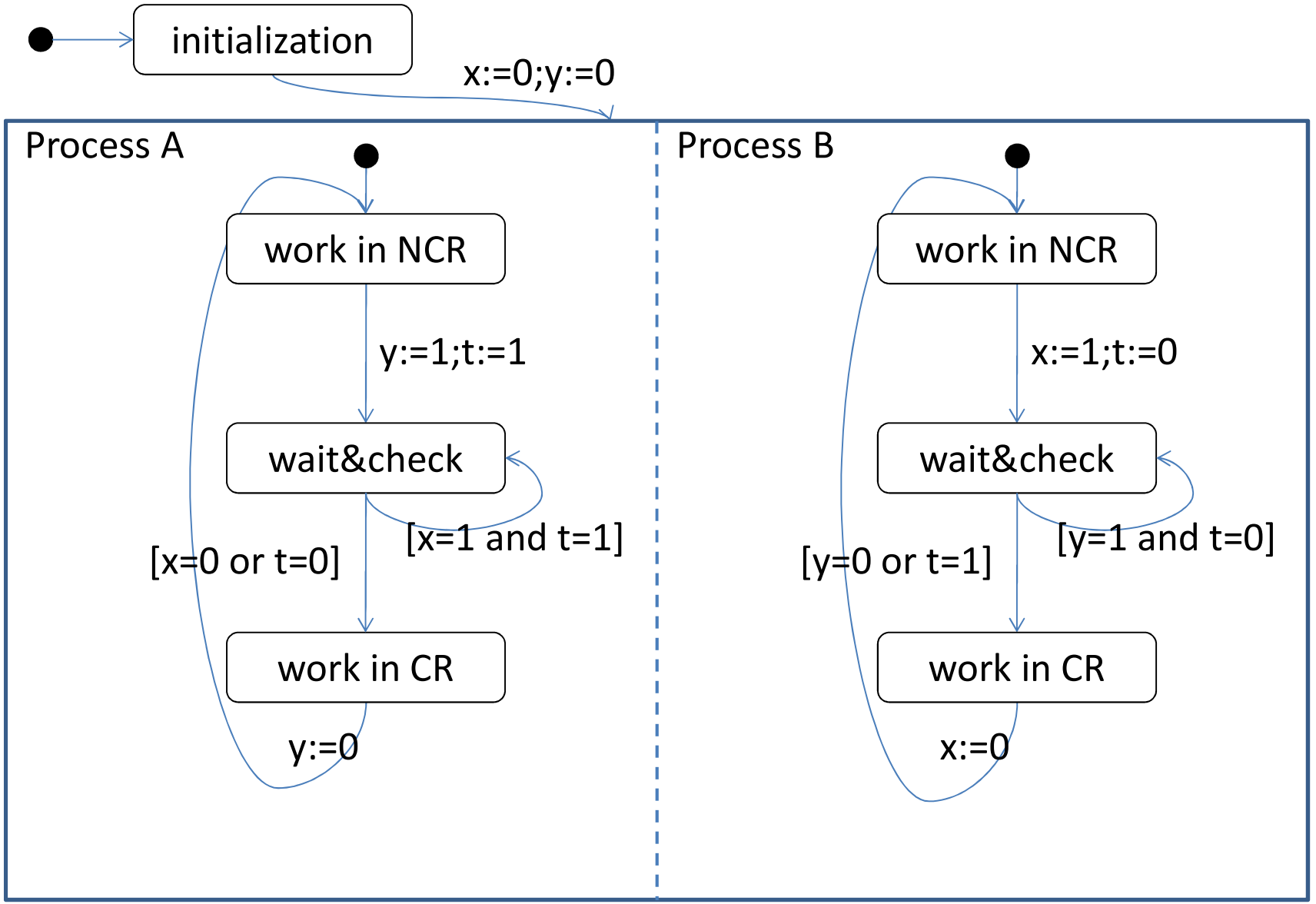
无死锁性质(wait&check,ok)



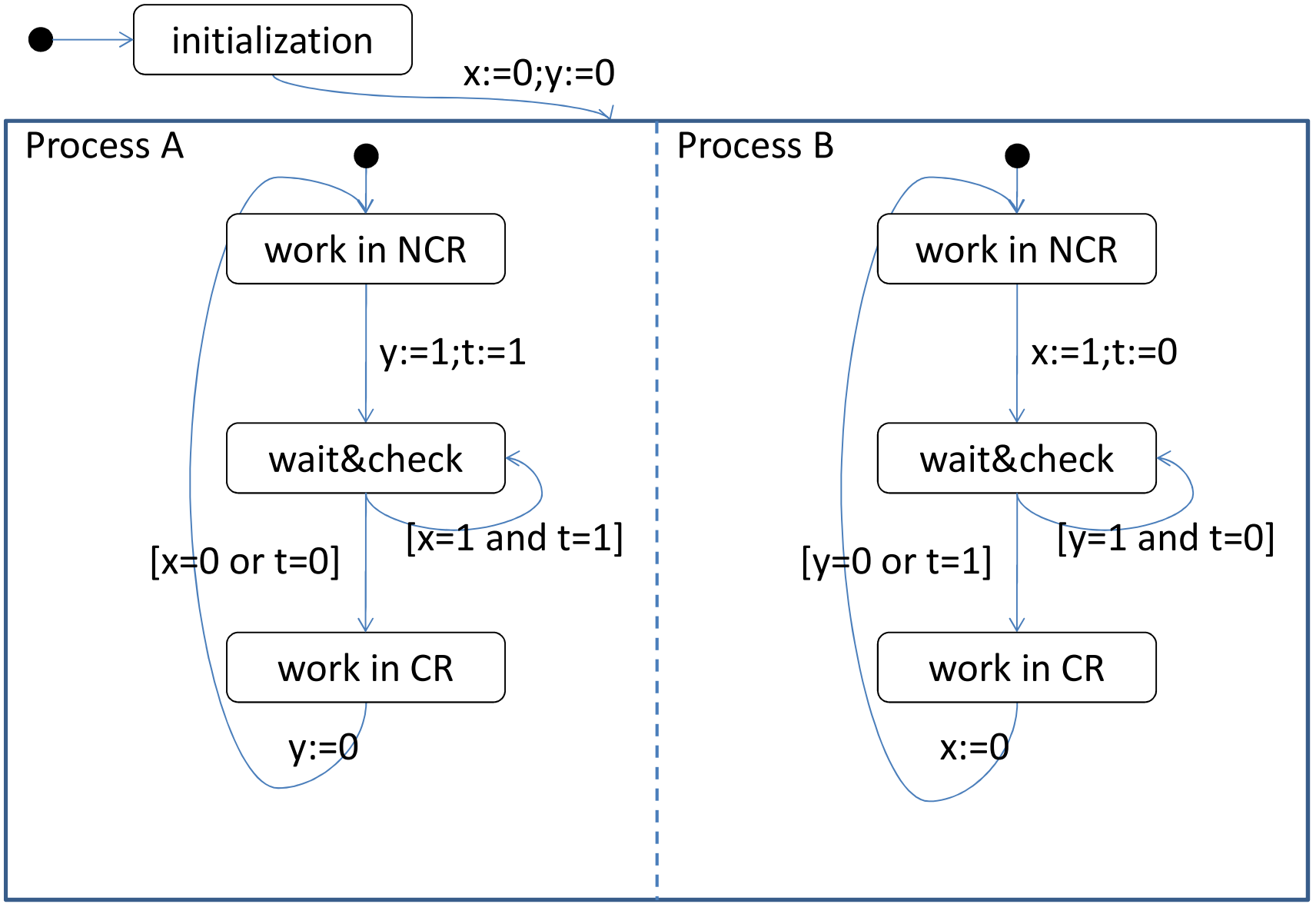
无死锁性质(wait,ok)



无活锁性质(wait&check,live-lock)

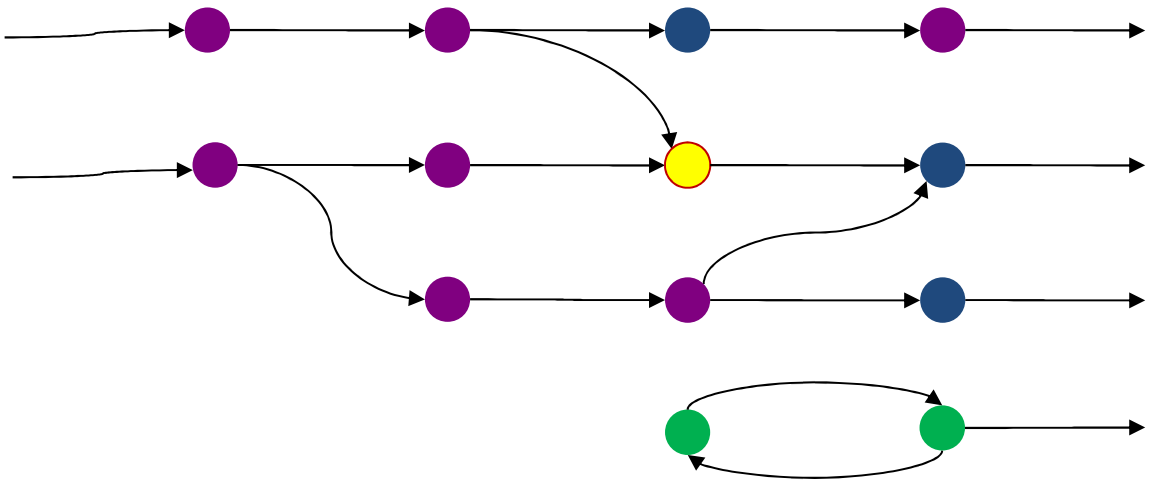


关于活性性质(AF(a=cr))



系统正确性 – 相关概念

- 安全性质　—　普适性 (universality)
- 活性性质　—　必然性 (inevitability)
- 可达性质　—　或然性 (possibility)



- 逻辑 --- 时序逻辑

形式化方法与软件正确性：小结

- 理想化的软件开发方法：
- 形式规格说明：用形式的语言描述软件规格说明
- 形式验证方法：用形式的方法证明软件满足规格说明

- 现状：
- 小规模应用有一定可行性
- 整体而言处于探索研究性质

- 对形式化方法基本理论和概念的学习有助于：
- 研究更好的软件开发方法
- 提升理论水平、使用更好的开发方法、开发高质量软件

形式化方法的部分研究课题

- 形式语言及基于形式语言的需求分析与软件设计
- 不同软件开发阶段的软件代码和软件设计之间的关系
- 自动生成满足需求或设计的代码
- 软件代码或软件设计相对于给定性质的正确性 (√)

二、软件正确性与软件系统行为

- 软件正确性
- 软件系统行为与行为规范

1. 软件正确性：

- 计算结果正确性
- 系统行为(计算过程)正确性

软件正确性

计算：

状态---(动作)---状态---(动作)---状态---(动作)---状态.....

结果？
状态序列？
动作序列？

软件正确性

计算结果正确性

系统行为(计算过程)正确性

a) 软件正确性：计算结果

例1a - 整数平方根

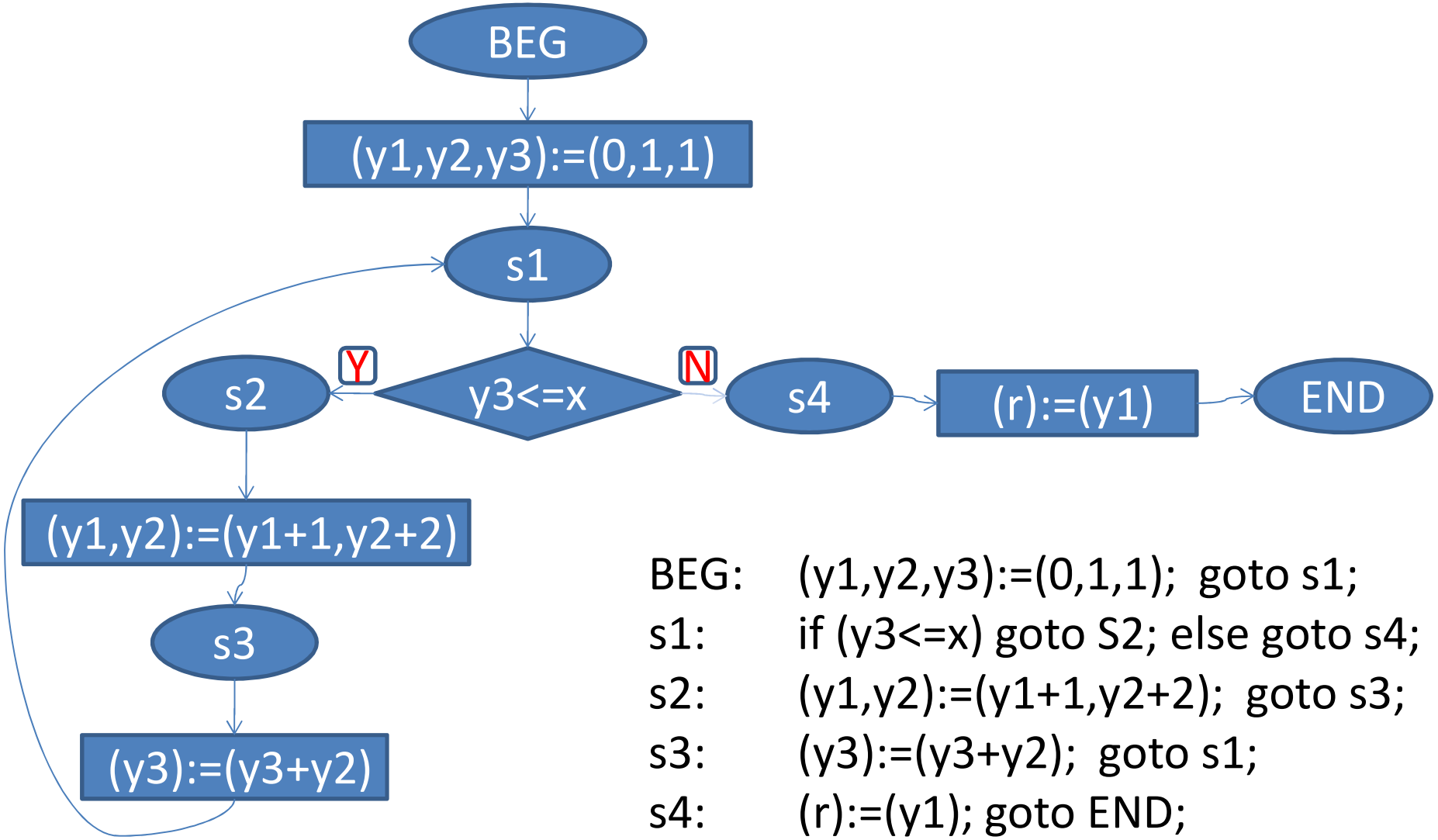
- 整数平方根的定义如下：

> r是 x 的整数平方根当且仅当 $r*r \le x$ 且 $x<(r+1)*(r+1)$。

- 我们的需求如下：

> 输入x。
> 输出x 的整树平方根。

例1a - 整数平方根

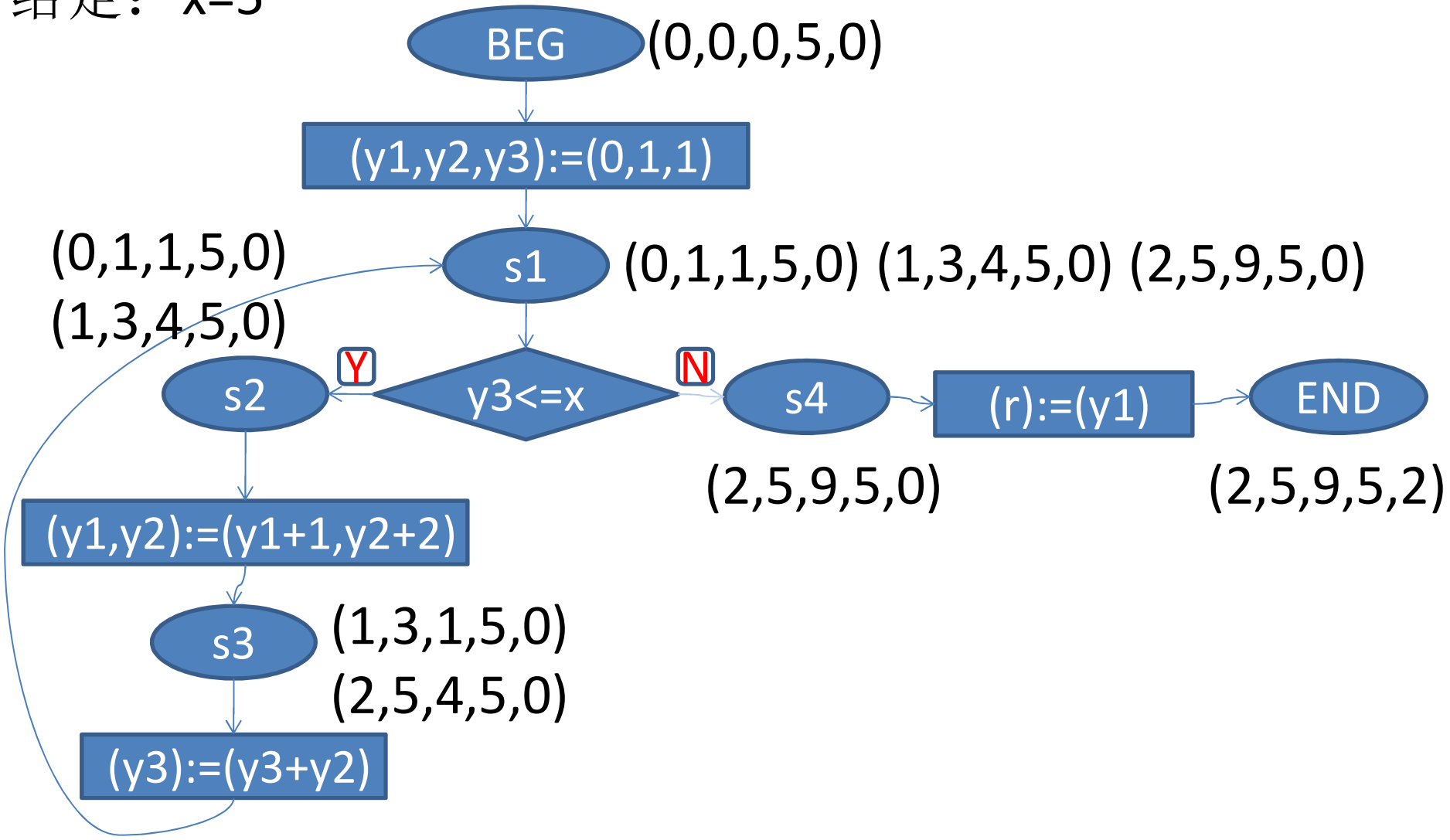



```
BEG:    (y1,y2,y3):=(0,1,1);  goto s1;
s1:     if (y3<=x) goto S2; else goto s4;
s2:     (y1,y2):=(y1+1,y2+2);  goto s3;
s3:     (y3):=(y3+y2);  goto s1;
s4:     (r):=(y1); goto END;
```

例1a - 整数平方根(计算过程实例)

状态：LAB+(y1,y2,y3,x,r)

给定：x=5



例1a - 整数平方根：计算结果

(LAB,y1,y2,y3,x,r)

(BEG,0,0,0,5,0) … … … (END,2,5,9,5,2)

(BEG,0,0,0,6,0) … … … (END,2,5,9,6,2)
(BEG,0,0,0,7,0) … … … (END,2,5,9,7,2)
(BEG,0,0,0,8,0) … … … (END,2,5,9,8,2)
(BEG,0,0,0,9,0) … … … (END,3,7,16,9,3)

例1a - 整数平方根：计算结果正确性

(LAB,y1,y2,y3,x,r)

(BEG,0,0,0,5,0) … … … (END,2,5,9,5,2)

(BEG,0,0,0,6,0) … … … (END,2,5,9,6,2)
(BEG,0,0,0,7,0) … … … (END,2,5,9,7,2)
(BEG,0,0,0,8,0) … … … (END,2,5,9,8,2)
(BEG,0,0,0,9,0) … … … (END,3,7,16,9,3)

(BEG,-,-,-,9,-) … … … (END,-,-,-,9,3)　　x>=r*r且x<(r+1)*(r+1);

例1a - 整数平方根：抽象计算结果

x=5,r=0
pc=BEG

x=5,r=0
pc=s_1

x=5,r=0
pc=s_2

x=5,r=2
pc=END

$\phi(x,r)$: x>=r*r $\wedge$ x<(r+1)*(r+1);

x=8,r=0
pc=BEG

x=8,r=0
pc=s_1

x=8,r=0
pc=s_2

x=8,r=2
pc=END

软件正确性：计算结果正确性

计算终止➔输出r ➔ r满足给定性质

计算终止➔终止状态r ➔ r满足给定性质

计算是否能够得到一个计算结果：程序终止性问题

正确性保障方法

计算结果的观测 ➔测试 ➔ 计算的部分实例

计算的抽象模型 ➔形式验证 ➔ 所有的可能的计算

其他：符号执行 、静态分析等

软件正确性

- 计算结果正确性
- 系统行为正确性

b) 软件正确性：系统行为正确性

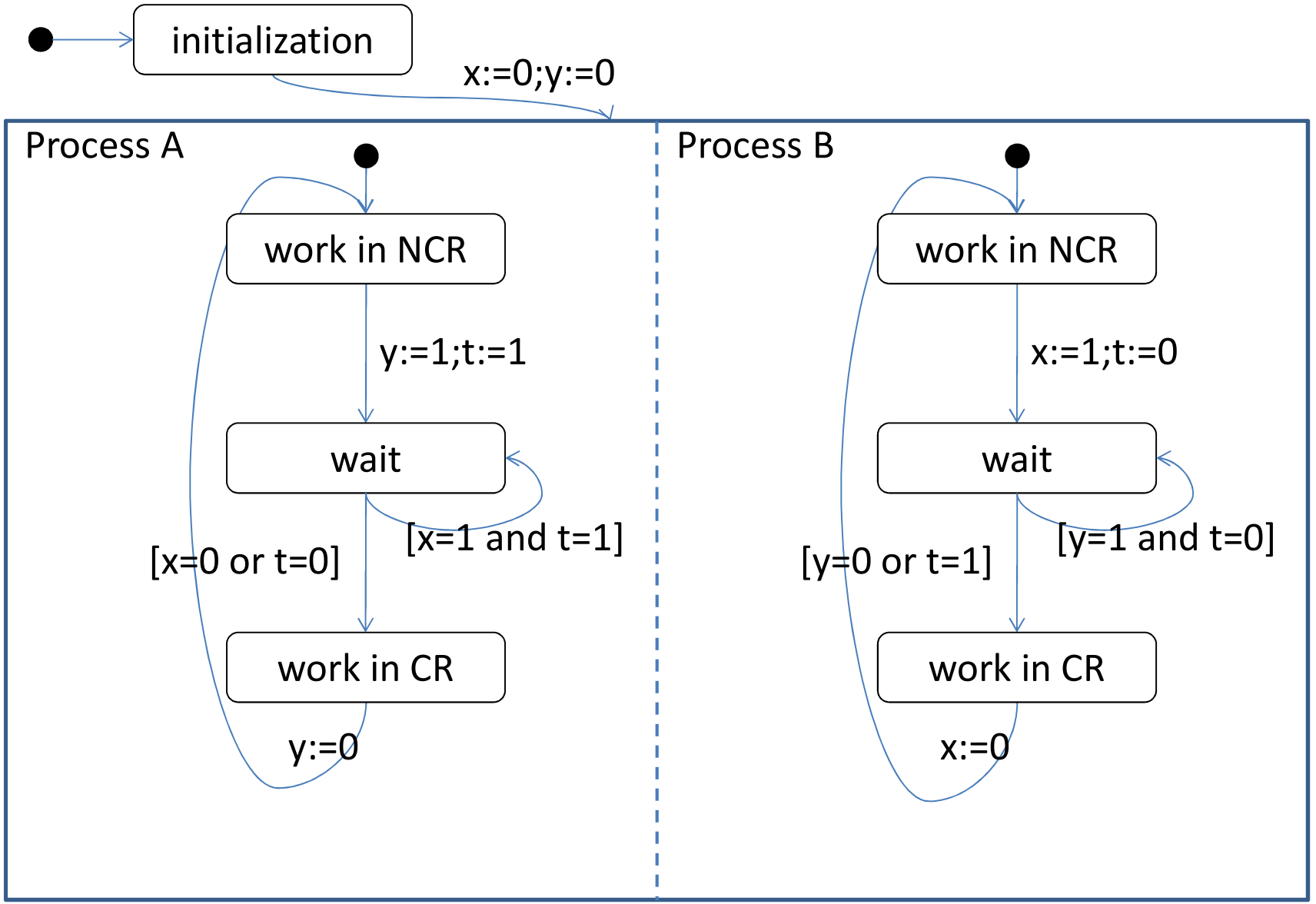
可观测状态序列

可观测动作序列

系统行为正确性

可观测状态序列

例2-互斥：状态图(State Diagram)



例2-互斥：算法

VAR: x: 0..1; y: 0..1; t: 0..1;
a: {NCR,wait,CR}; b: {NCR,wait,CR};
INIT: x=0; y=0;
a=NCR; b=NCR;

Process A:

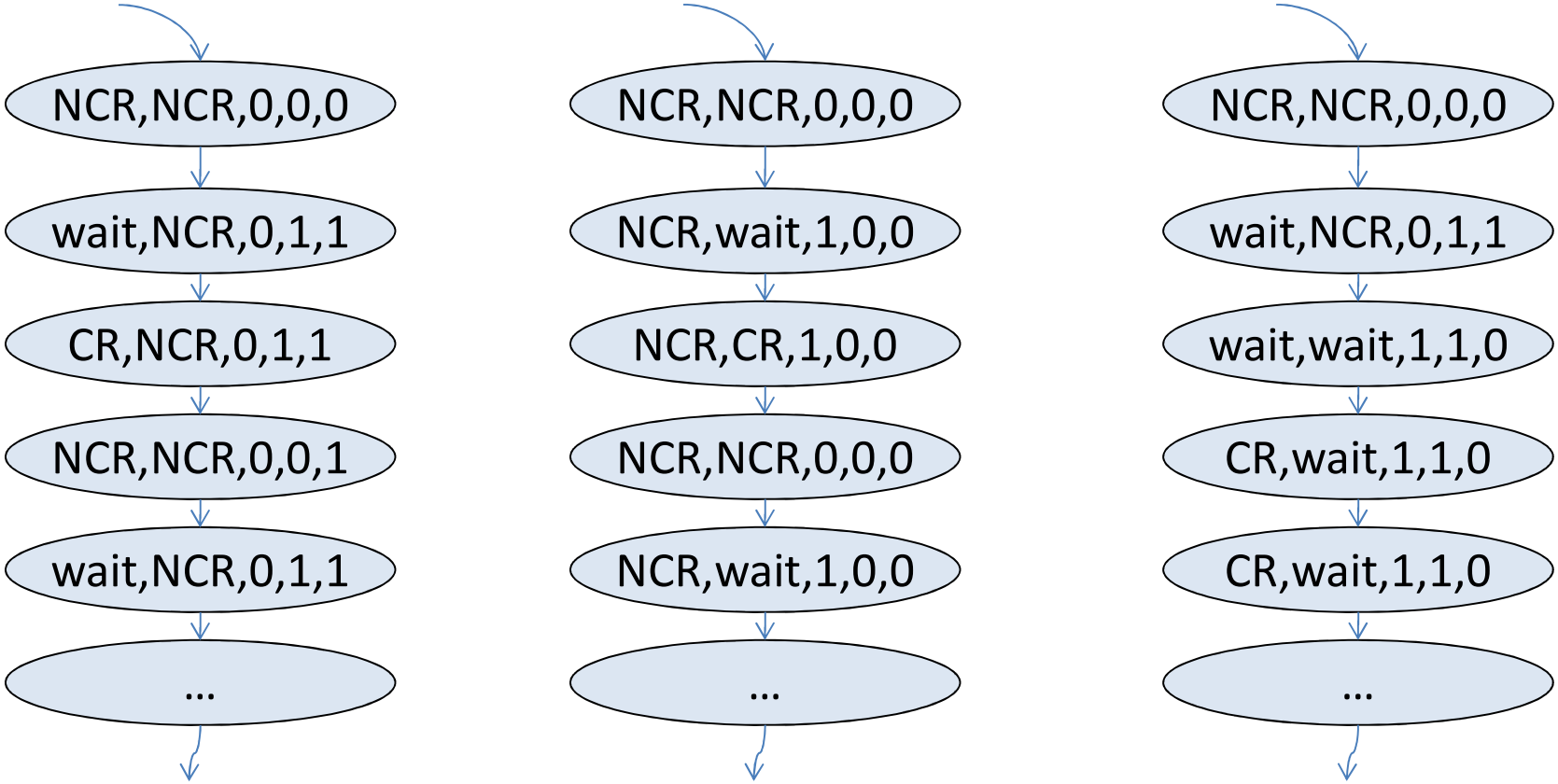
a=NCR	→ (a,y,t):=(wait,1,1);
a=wait∧(x=0∨t=0)	→ (a):=(CR);
a=wait∧¬(x=0∨t=0)	→ (a):=(wait);
a=CR	→ (a,y):=(NCR,0);

Process B:

b=NCR	→ (b,x,t):=(wait,1,0);
b=wait∧(y=0∨t=1)	→ (b):=(CR);
b=wait∧¬(y=0∨t=1)	→ (b):=(wait);
b=CR	→ (b,x):=(NCR,0);

例2-互斥：计算过程(状态)

(a,b,x,y,t)



给定类型状态不在计算过程中出现：CR,CR,-,-,-

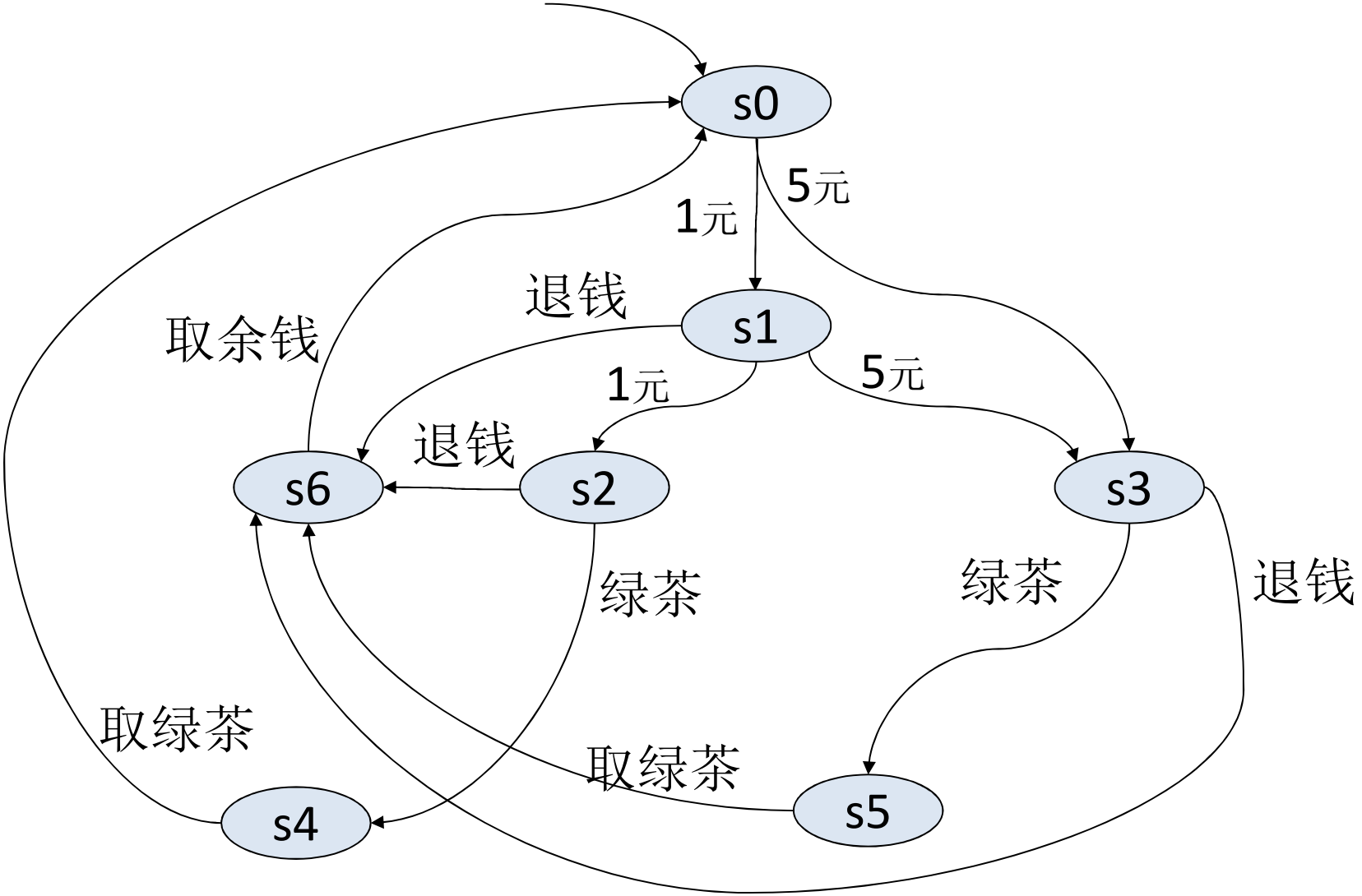
软件正确性：系统行为正确性

计算过程➔可观测状态序列π ➔ π满足给定性质

系统行为正确性

可观测动作序列

例3 -自动售茶机：设计



例3 -自动售茶机：抽象计算过程(动作)

1元,1元,绿茶,取绿茶, ,… …

1元,5元,绿茶,取绿茶,取余钱,… …

例3 -自动售茶机：动作相关的问题

投入1元钱后是否可以取消购买(退钱)？

投入1元钱后又投入5元钱后是否会找钱(取余钱)？

软件正确性：系统行为正确性

计算过程➔可观测动作序列π ➔ π满足给定性质

动作与状态

有一定对应关系，侧重点不一样

例3 -自动售茶机：动作与状态的对应

1元,1元,绿茶,取绿茶,1元,5元,绿茶,取绿茶,取余钱,… …

s0,s1,s2,s4,s0,s1,s3,s5,s6,s0,… …

用x表示用户在系统中钱数：

s0: x=0
s1: x=1
s2: x=2
s3: x=6 或 x=5
s4: x=0
s5: x=4 或 x=3
s6: x=4 或 x=3 或 x=2 或 …

过程与结果

计算过程：更为通用

计算结果：可以看成是计算过程的一部分

正确性保障方法

计算过程的观测 ➔测试 ➔ 计算过程的部分实例

虚拟环境中计算过程的观测 ➔仿真 ➔ 计算过程的部分实例

计算过程的抽象模型 ➔形式验证 ➔ 所有的抽象计算过程

其他：符号执行 、静态分析等

2. 软件系统行为与行为规范

系统行为：系统模型

行为规范：系统性质的描述(被允许的系统行为)

a) 系统行为

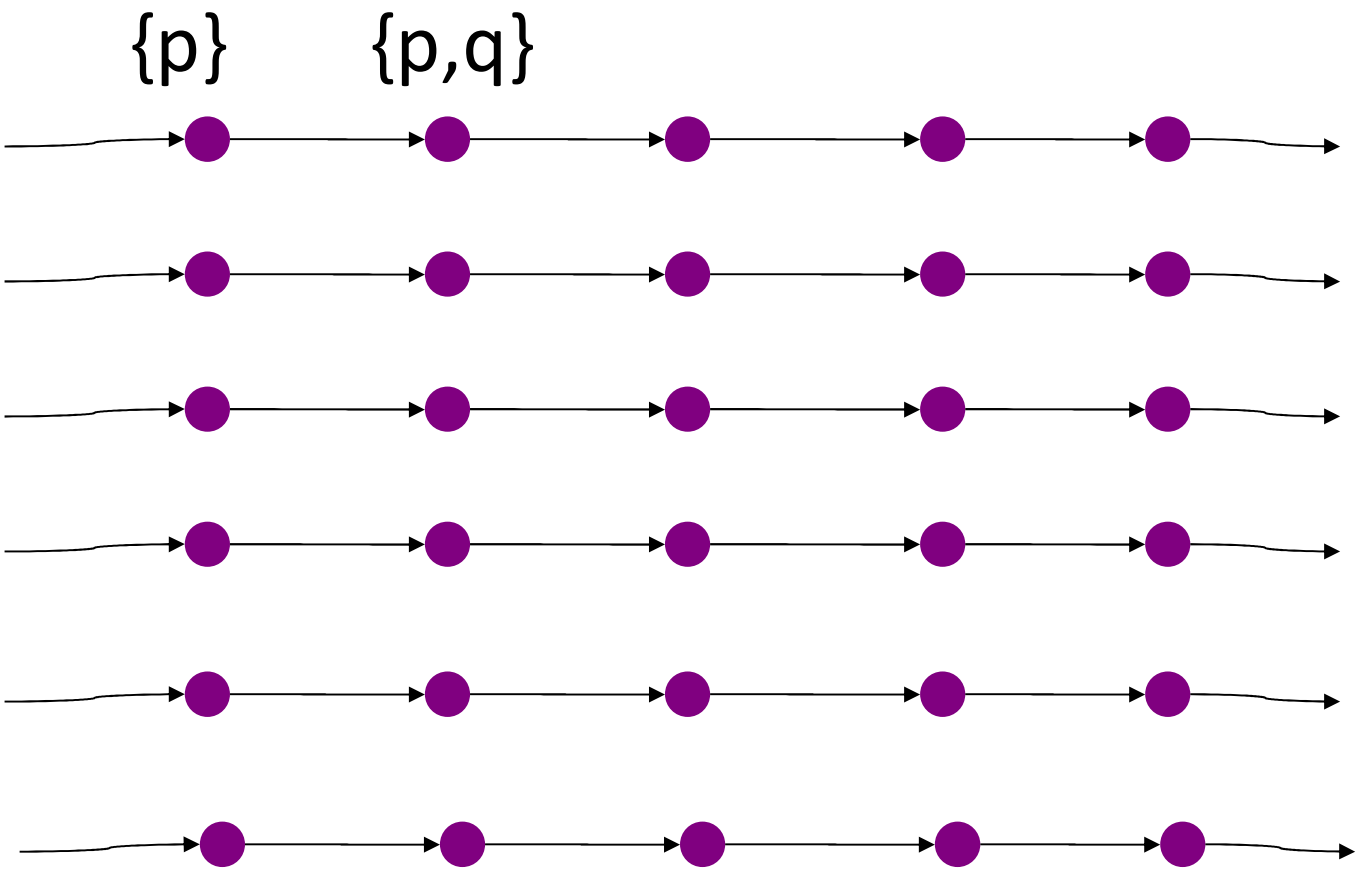
计算：

状态---(动作)---状态---(动作)---状态---(动作)---状态.....

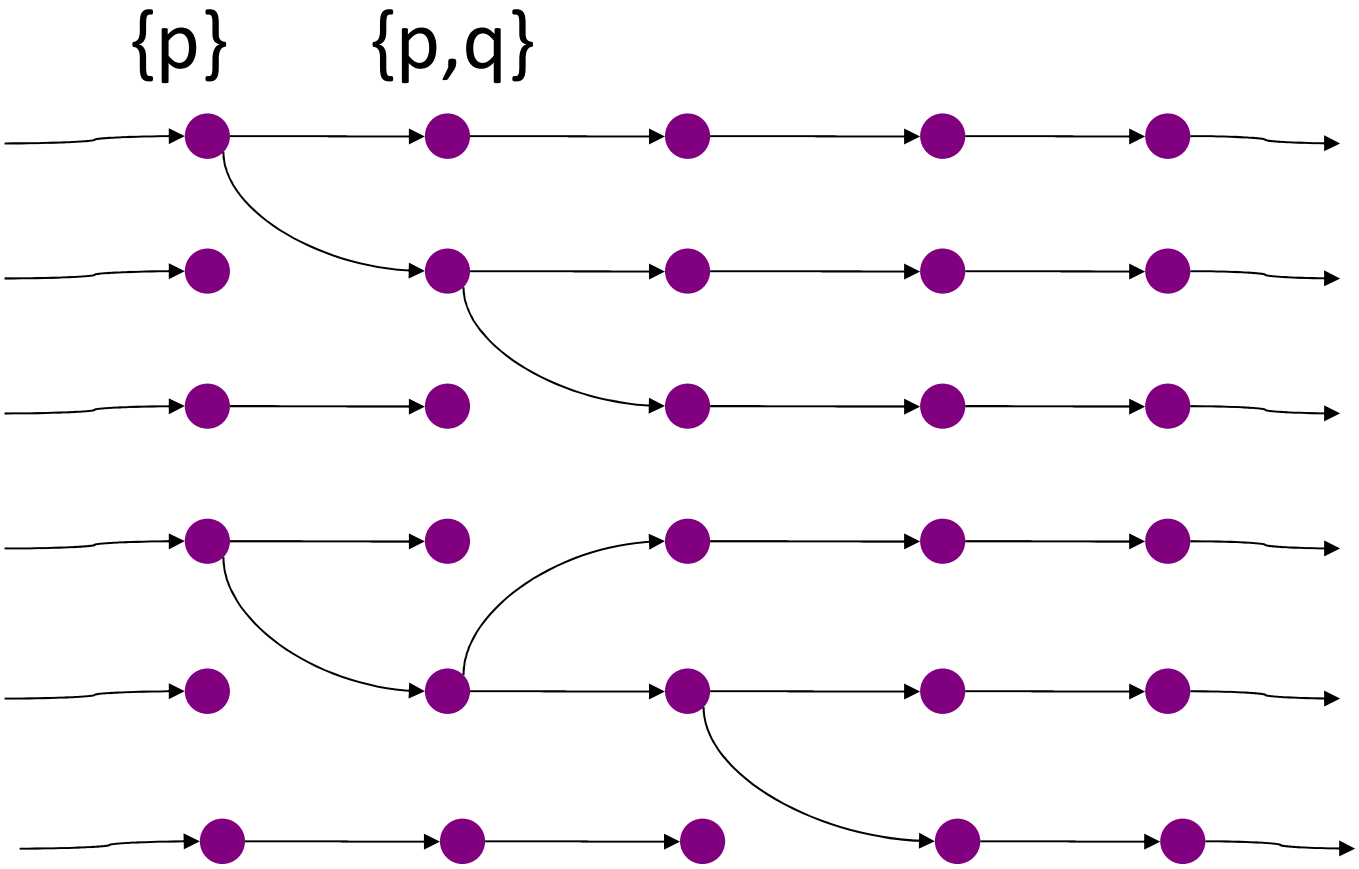
状态---(动作)---状态---(动作)---状态---(动作)---状态.....

状态---(动作)---状态---(动作)---状态---(动作)---状态.....

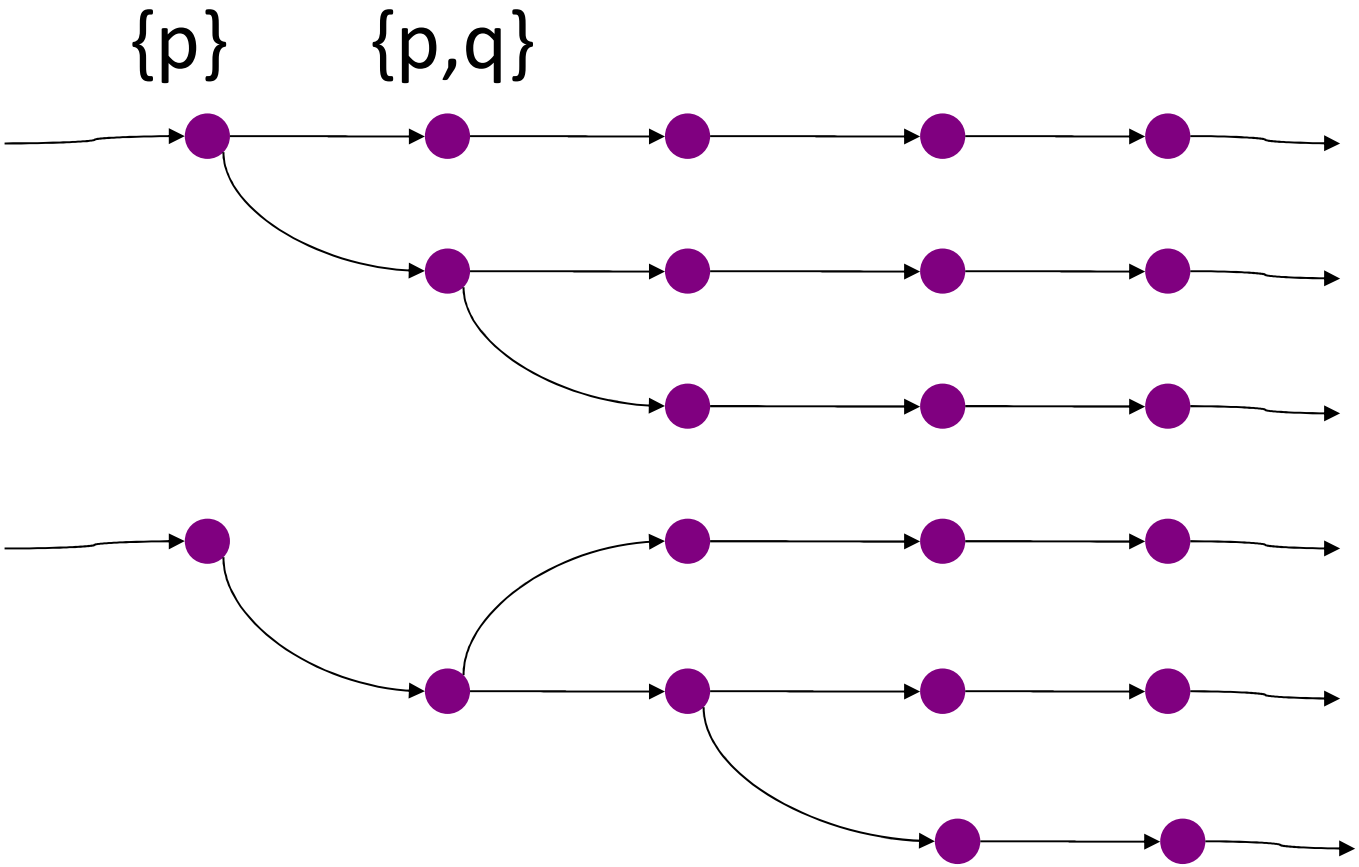
软件系统行为(线性运行集合)



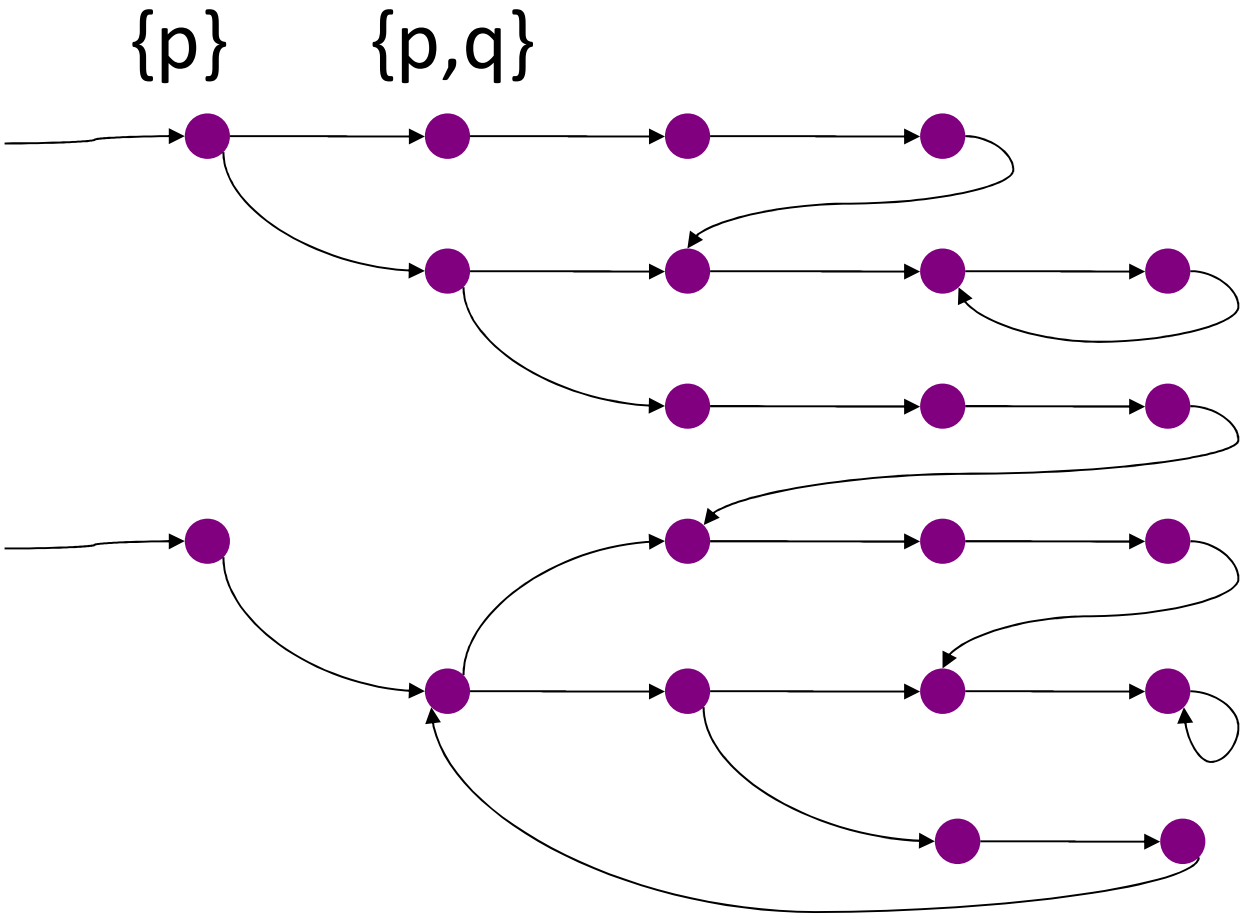
软件系统行为(树状运行集合)



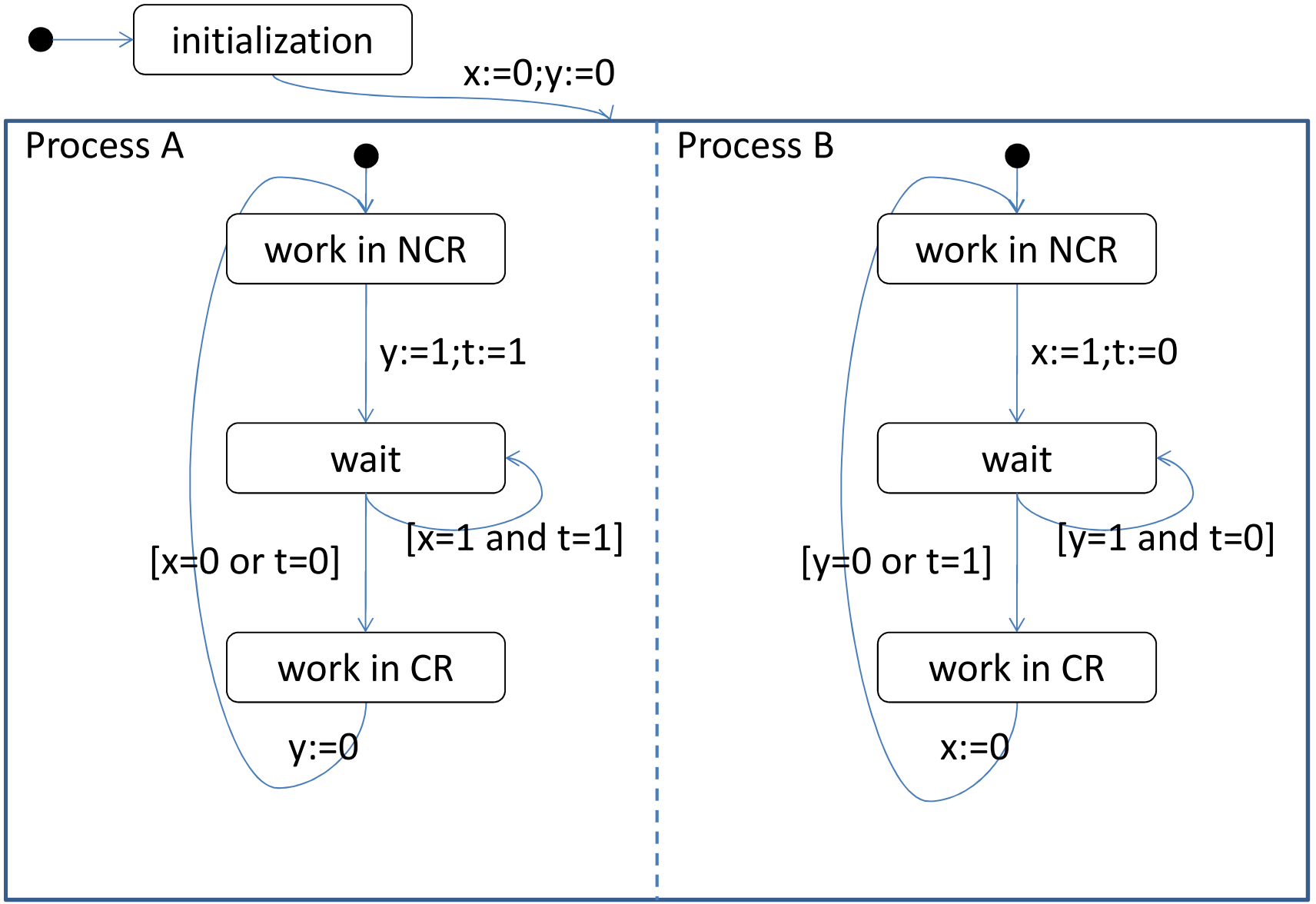
软件系统行为(树状运行集合)



软件系统行为(图状模型)



例2-互斥：状态图(State Diagram)



例2-互斥：算法

VAR: x: 0..1; y: 0..1; t: 0..1;
a: {NCR,wait,CR}; b: {NCR,wait,CR};
INIT: x=0; y=0;
a=NCR; b=NCR;

Process A:

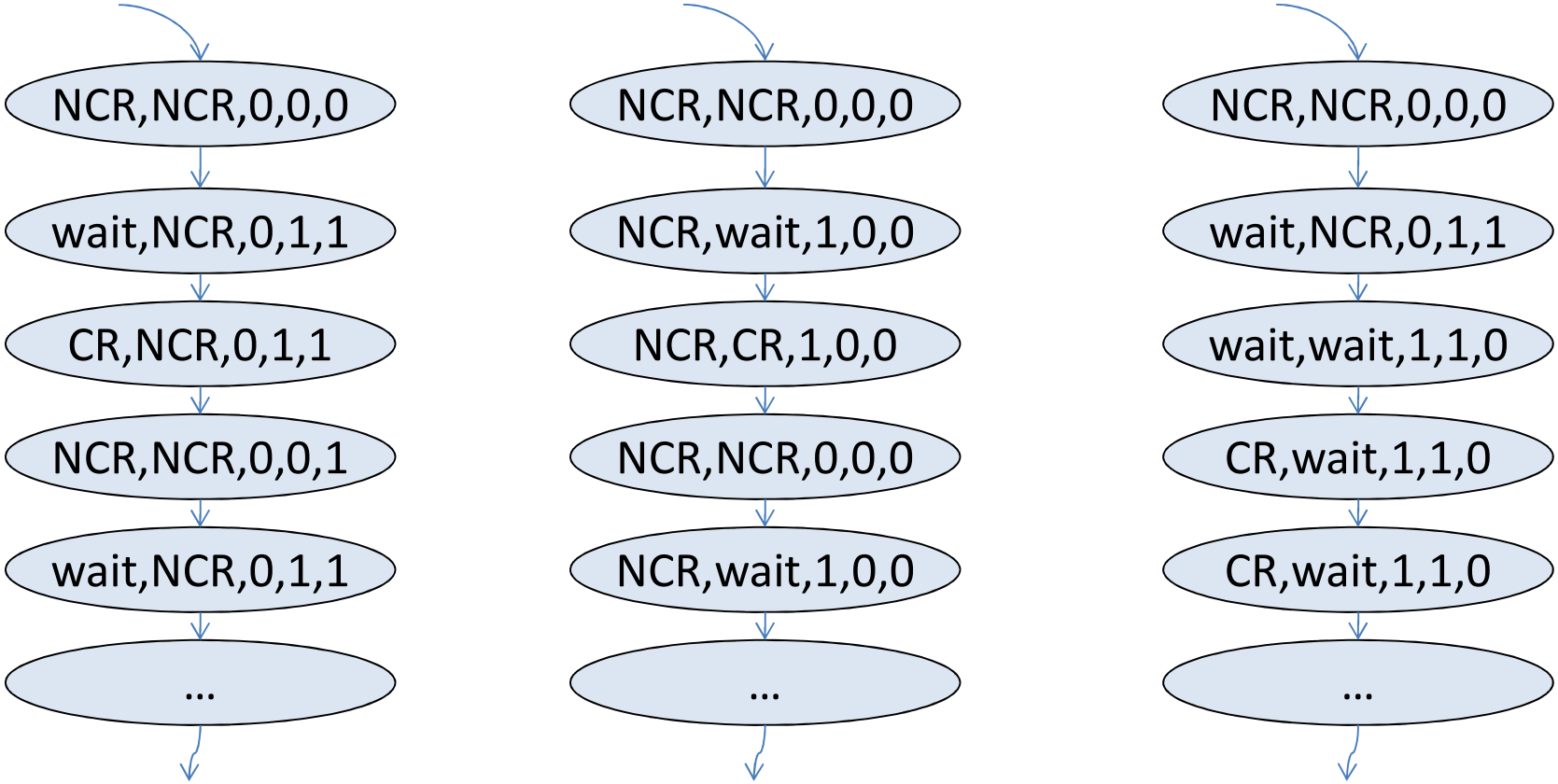
a=NCR	→ (a,y,t):=(wait,1,1);
a=wait∧(x=0∨t=0)	→ (a):=(CR);
a=wait∧¬(x=0∨t=0)	→ (a):=(wait);
a=CR	→ (a,y):=(NCR,0);

Process B:

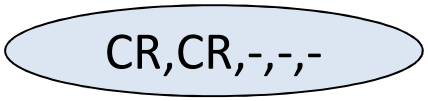
b=NCR	→ (b,x,t):=(wait,1,0);
b=wait∧(y=0∨t=1)	→ (b):=(CR);
b=wait∧¬(y=0∨t=1)	→ (b):=(wait);
b=CR	→ (b,x):=(NCR,0);

例2-互斥：计算过程(状态)

(a,b,x,y,t)

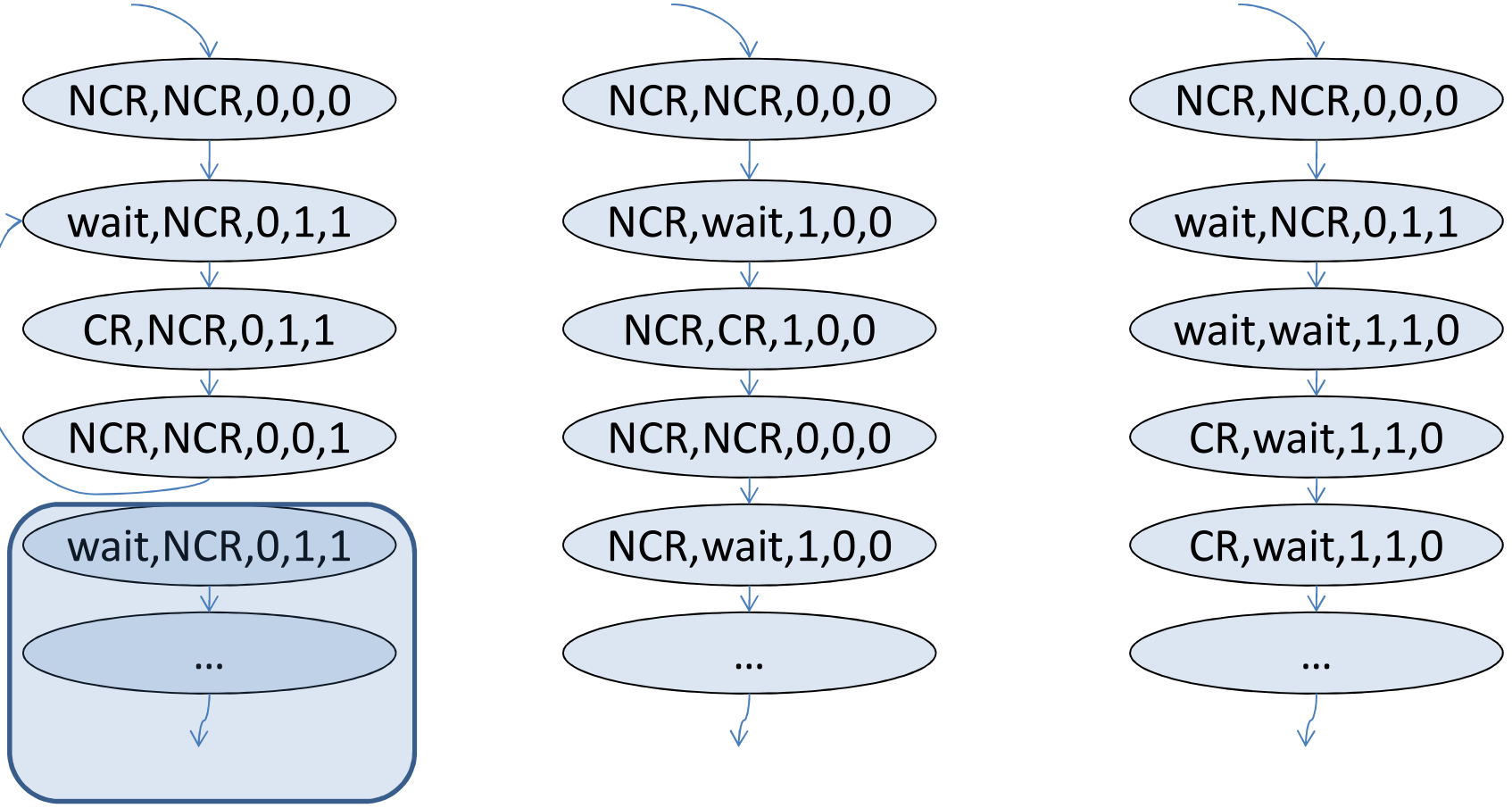


给定类型状态不可达：

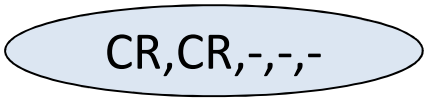


例2-互斥：计算过程(状态)

(a,b,x,y,t)

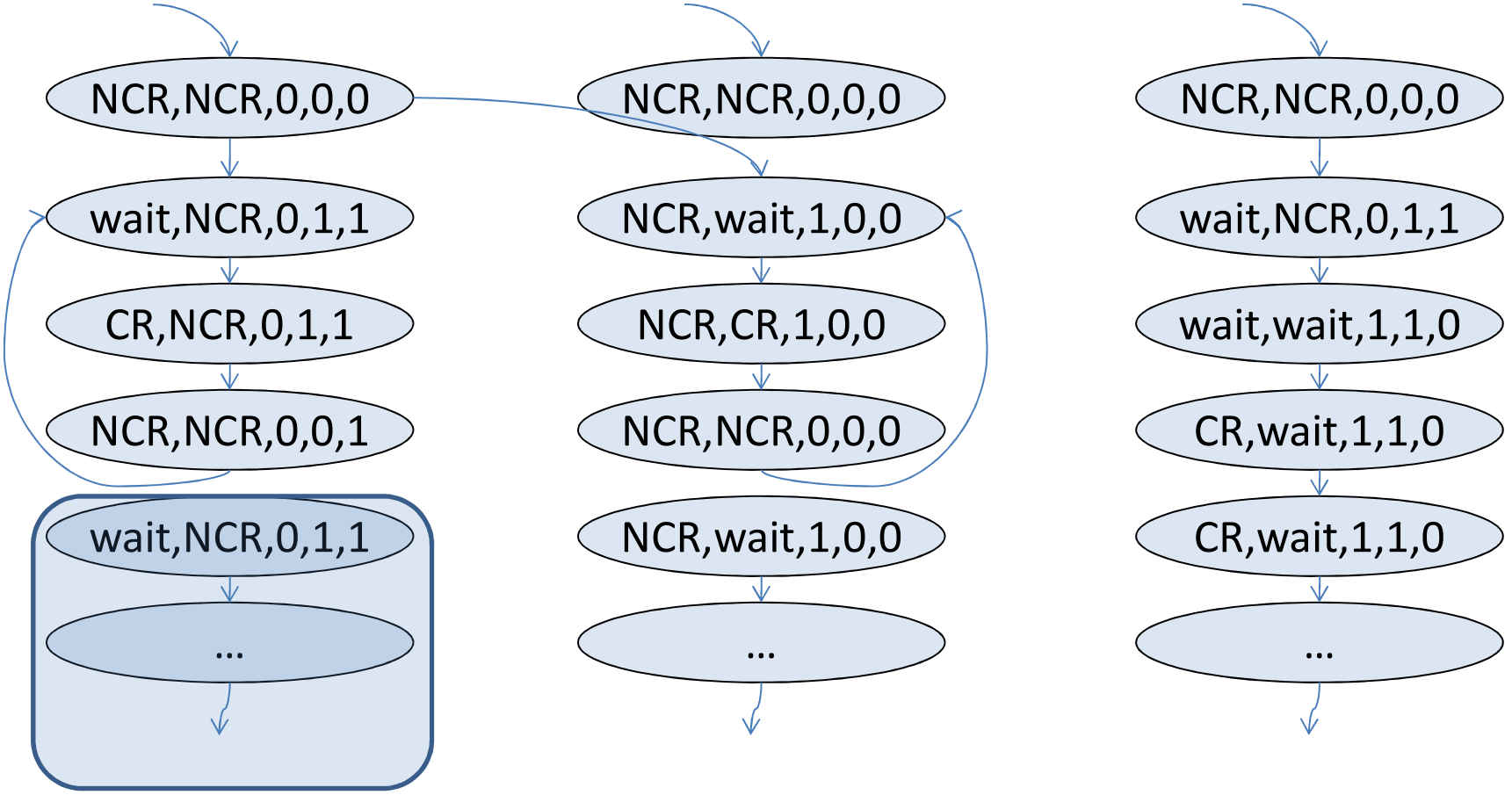


给定类型状态不可达：

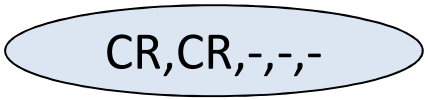


例2-互斥：计算过程(状态)

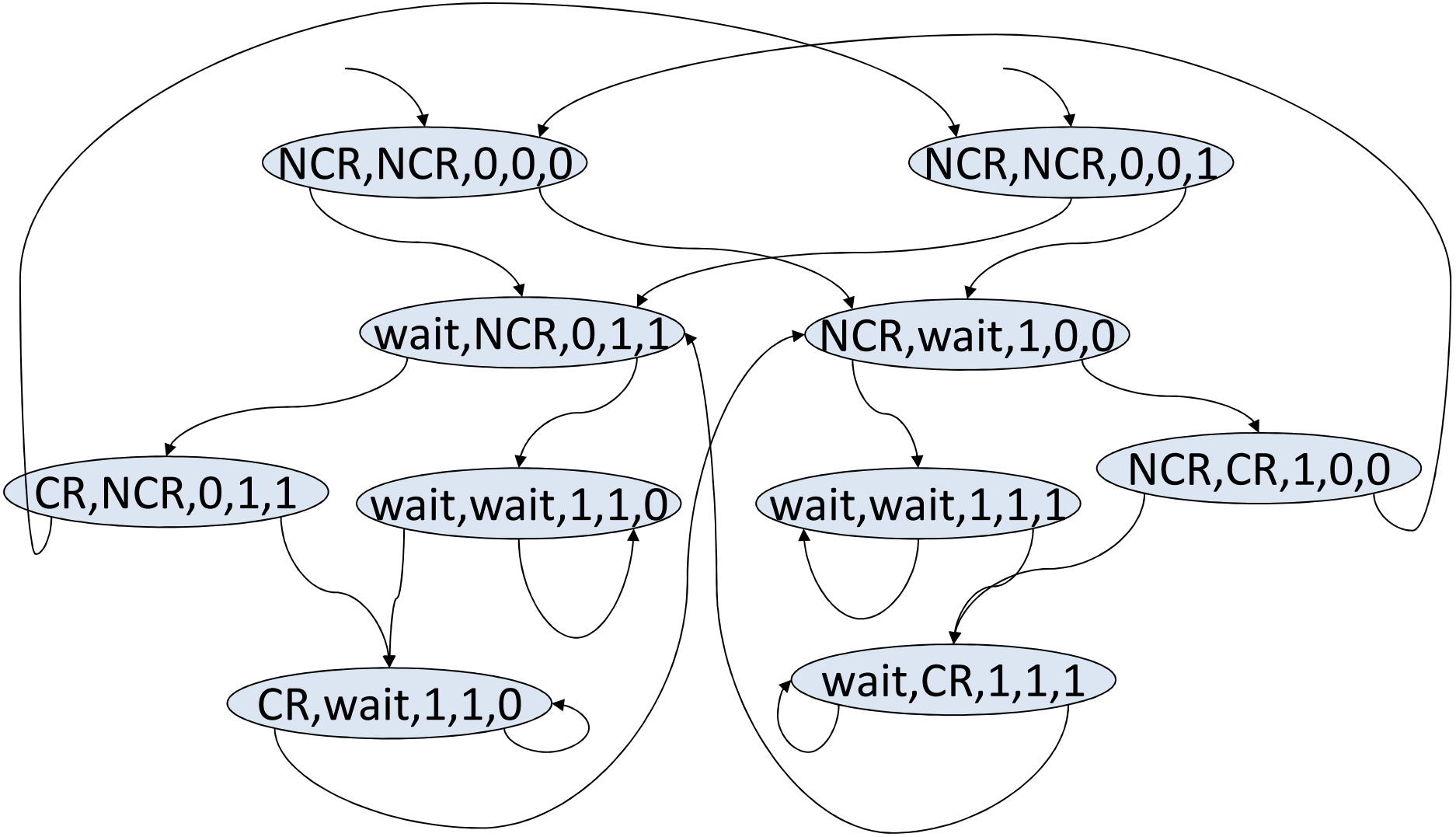
(a,b,x,y,t)



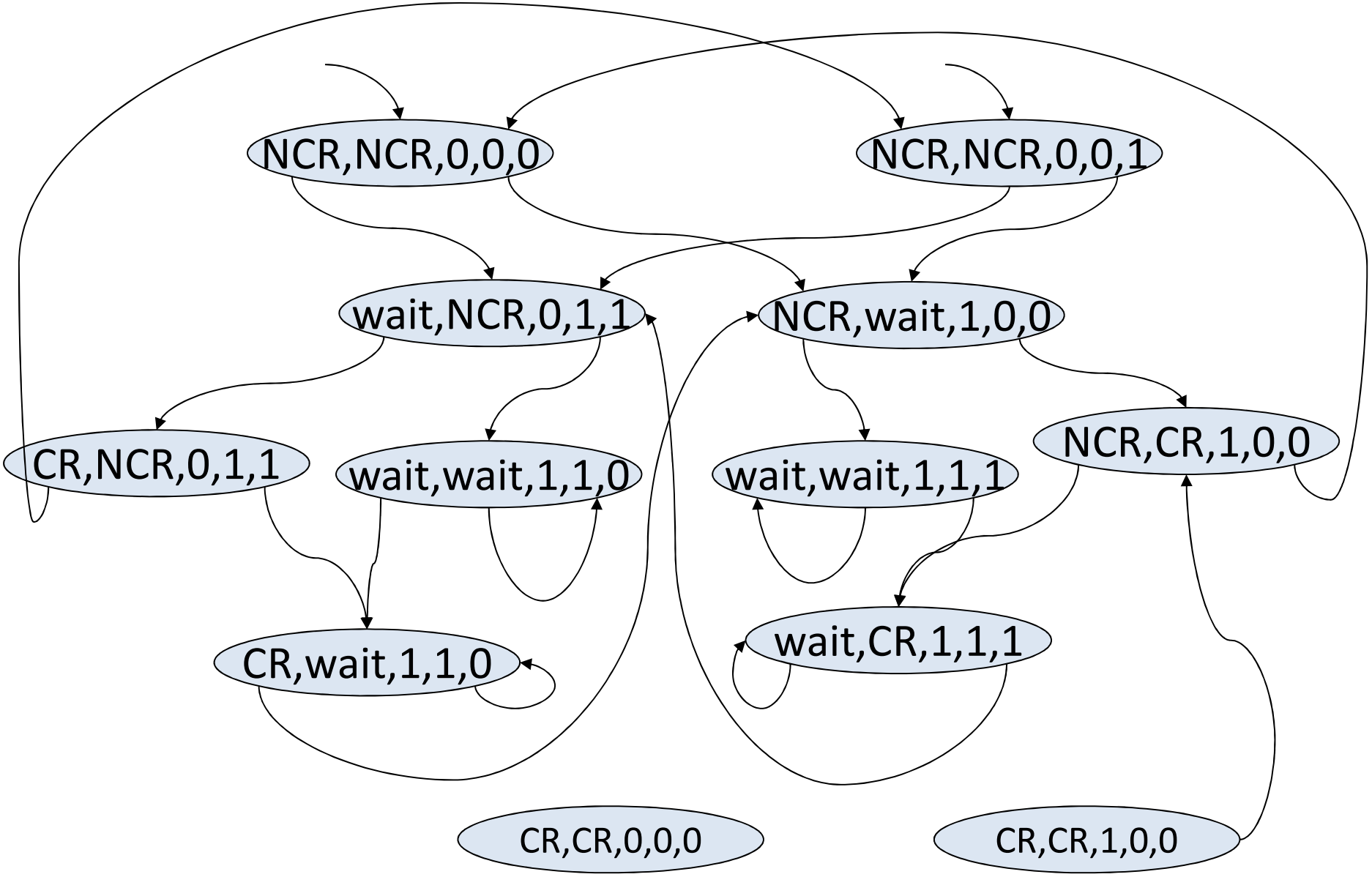
给定类型状态不可达：



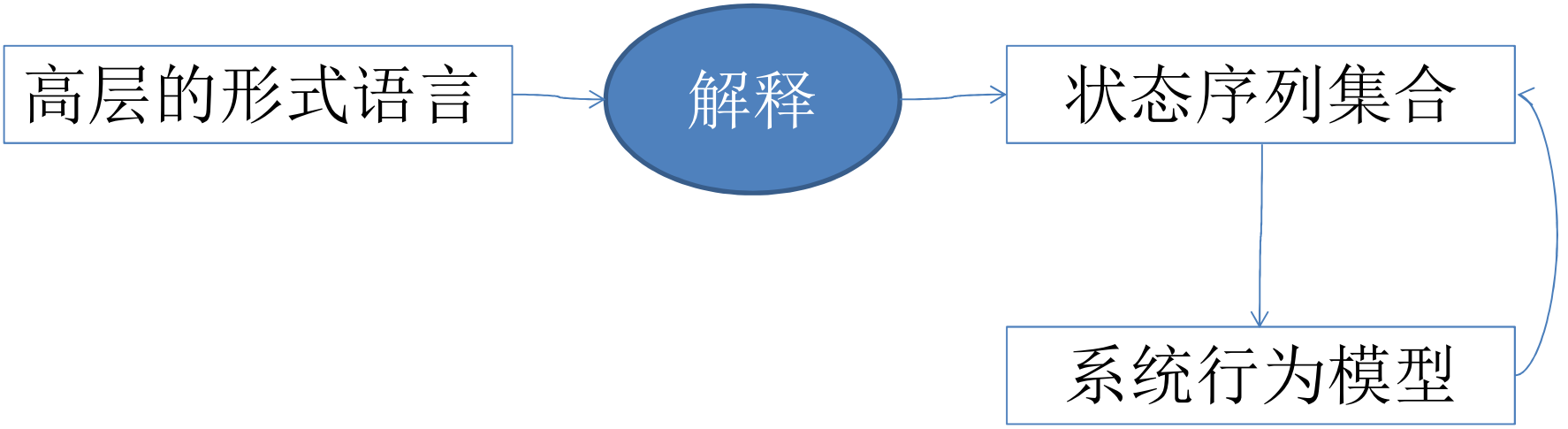
例2-互斥：可达状态


NCR,NCR,0,0,0
NCR,NCR,0,0,1
wait,NCR,0,1,1
NCR,wait,1,0,0
CR,NCR,0,1,1
wait,wait,1,1,0
wait,wait,1,1,1
NCR,CR,1,0,0
CR,wait,1,1,0
wait,CR,1,1,1

例2-互斥：可达状态 + 部分不可达状态



软件系统行为模型



系统行为(离散行为)

状态集合: S

初始状态集合: I

状态迁移关系: $R \subseteq S \times S$

系统: $M=\langle S,R,I \rangle$

系统行为: $[[M]] \subseteq S^{\omega}$

例2-互斥：状态集合

$S = \{ s_0, s_1, \ldots, s_{71} \}$

s_i 对应于五元组(a,b,x,y,t)的一个元素

其中s_i代表　(a,b,x,y,t)
当且仅当　　i=a*24+b*8+x*4+y*2+t

其中NCR代表0, wait代表1, CR代表2

例如:

(CR,NCR,0,0,0)　→　(2,0,0,0,0)　→　s_{48}
(NCR,CR,0,0,0)　→　(0,2,0,0,0)　→　s_{16}

例2-互斥：初始状态集合

$S = \{ s_0, s_1, \ldots, s_{71} \}$

$I = \{ s_0, s_1 \}$

其中：

$s_0 = (0,0,0,0,0) = (NCR,NCR,0,0,0)$

$s_1 = (0,0,0,0,1) = (NCR,NCR,0,0,1)$

例2-互斥：状态迁移关系

$S = \{ s_0, s_1, \ldots, s_{71} \}$

$R = \{(s_k, s_j) \mid \ldots\ldots \}$

((0,-,-,-,-),(1,-,-,1,1)),
((1,-,0,-,-),(2,-,-,-,-)), ((1,-,-,-,0),(2,-,-,-,-)),
((1,-,1,-,1),(1,-,-,-,-))
((2,-,-,-,-),(1,-,-,-,0,-))

a=NCR → (a,y,t):=(wait,1,1);
a=wait∧(x=0∨t=0) → (a):=(CR);
a=wait∧¬(x=0∨t=0) → (a):=(wait);
a=CR → (a,y):=(NCR,0);

例2-互斥：状态迁移关系

$S = \{ s_0, s_1, \ldots, s_{71} \}$

$R = \{(s_k, s_j) \mid \ldots\ldots \}$

((0,-,-,-,-),(1,-,-,1,1)),
((1,-,0,-,-),(2,-,-,-,-)), ((1,-,-,-,0),(2,-,-,-,-)),
((1,-,1,-,1),(1,-,-,-,-))
((2,-,-,-,-),(1,-,-,-,0,-))

➔ $\{(s_0,s_{27}), (s_1,s_{27}), \ldots\ldots \}$

a=NCR → (a,y,t):=(wait,1,1);
a=wait∧(x=0∨t=0) → (a):=(CR);
a=wait∧¬(x=0∨t=0) → (a):=(wait);
a=CR → (a,y):=(NCR,0);

例2-互斥：状态迁移关系

$S = \{ s_0, s_1, \ldots, s_{71} \}$

$R = \{(s_k, s_j) \mid \ldots\ldots \}$

((-,0,-,-,-),(-,1,1,-,0)),
((-,1,-,0,-),(-,2,-,-,-)), ((-,1,-,-,1),(-,2,-,-,-)),
((-,1,1,-,1),(-,1,-,-,-))
((-,2,-,-,-),(-,1,-,0,-,-))

b=NCR	→ (b,x,t):=(wait,1,0);
b=wait∧(y=0∨t=1)	→ (b):=(CR);
b=wait∧¬(y=0∨t=1)	→ (b):=(wait);
b=CR	→ (b,x):=(NCR,0);

例2-互斥：状态迁移关系

$S = \{ s_0, s_1, \ldots, s_{71} \}$

$R = \{(s_k, s_j) \mid \ldots\ldots \}$

((0,-,-,-,-),(1,-,-,1,1)),
((1,-,0,-,-),(2,-,-,-,-)), ((1,-,-,-,0),(2,-,-,-,-)),
((1,-,1,-,1),(1,-,-,-,-))
((2,-,-,-,-),(1,-,-,-,0,-))

((-,0,-,-,-),(-,1,1,-,0)),
((-,1,-,0,-),(-,2,-,-,-)), ((-,1,-,-,1),(-,2,-,-,-)),
((-,1,1,-,1),(-,1,-,-,-))
((-,2,-,-,-),(-,1,-,0,-,-))

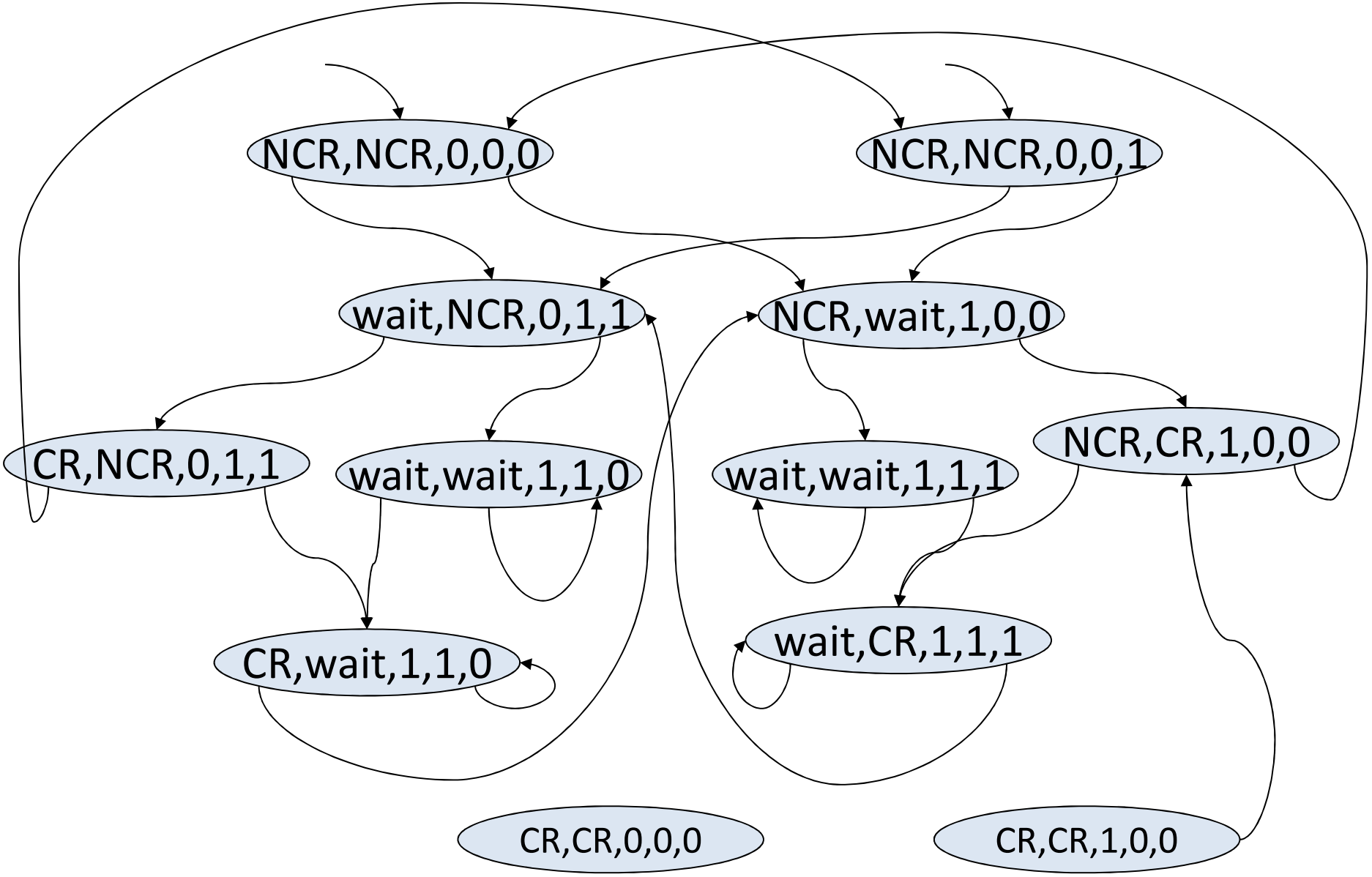
例2-互斥：系统行为模型

状态集合: $S = \{ s_0, s_1, \ldots, s_{71} \}$

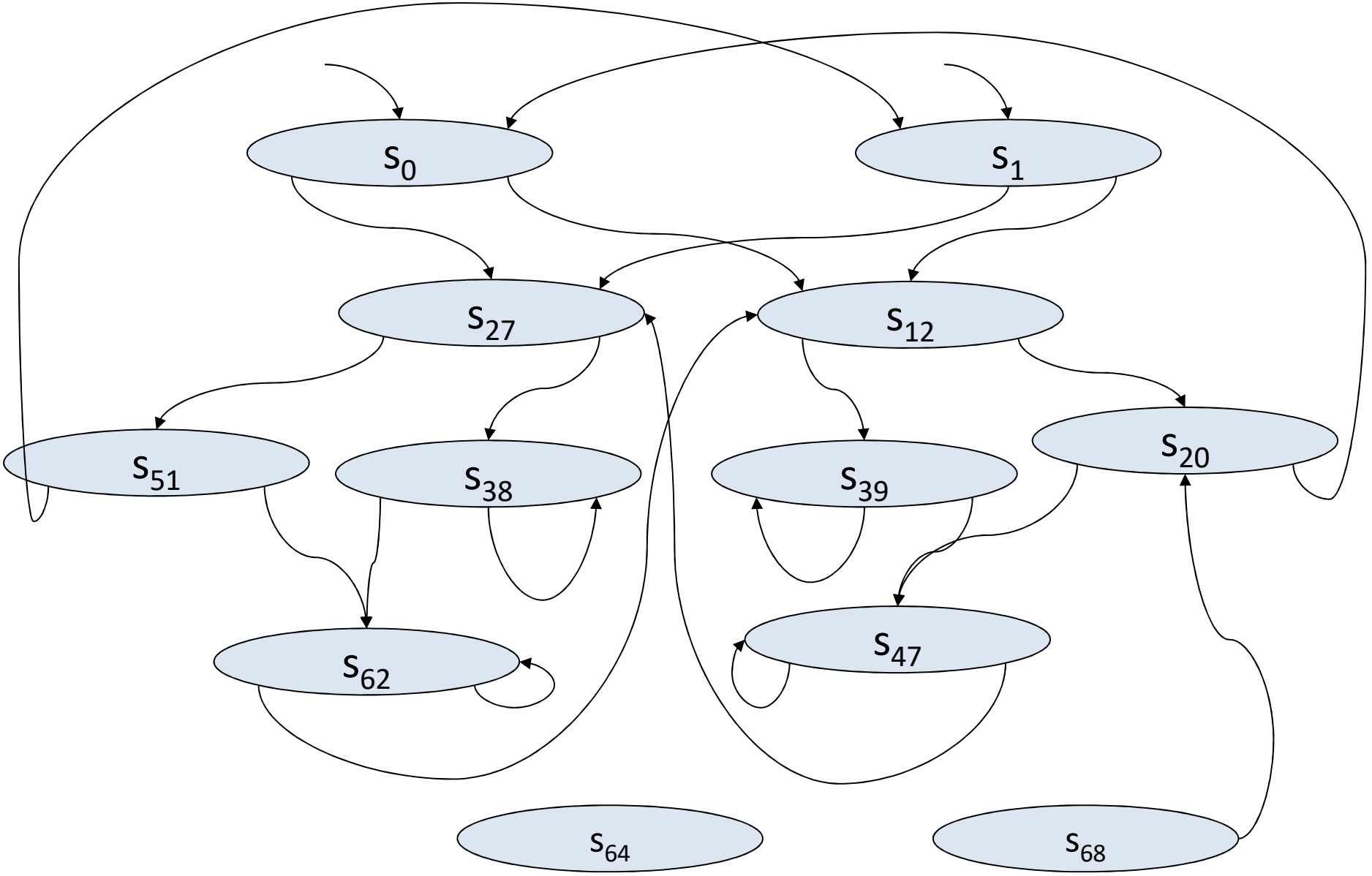
初始状态集合: $I = \{ s_0, s_1 \}$

状态迁移关系: $R = \{ (s_0, s_{27}), (s_0, s_{12}), \ldots\ldots \}$

例2-互斥：可达状态 + 部分不可达状态



例2-互斥：状态迁移图

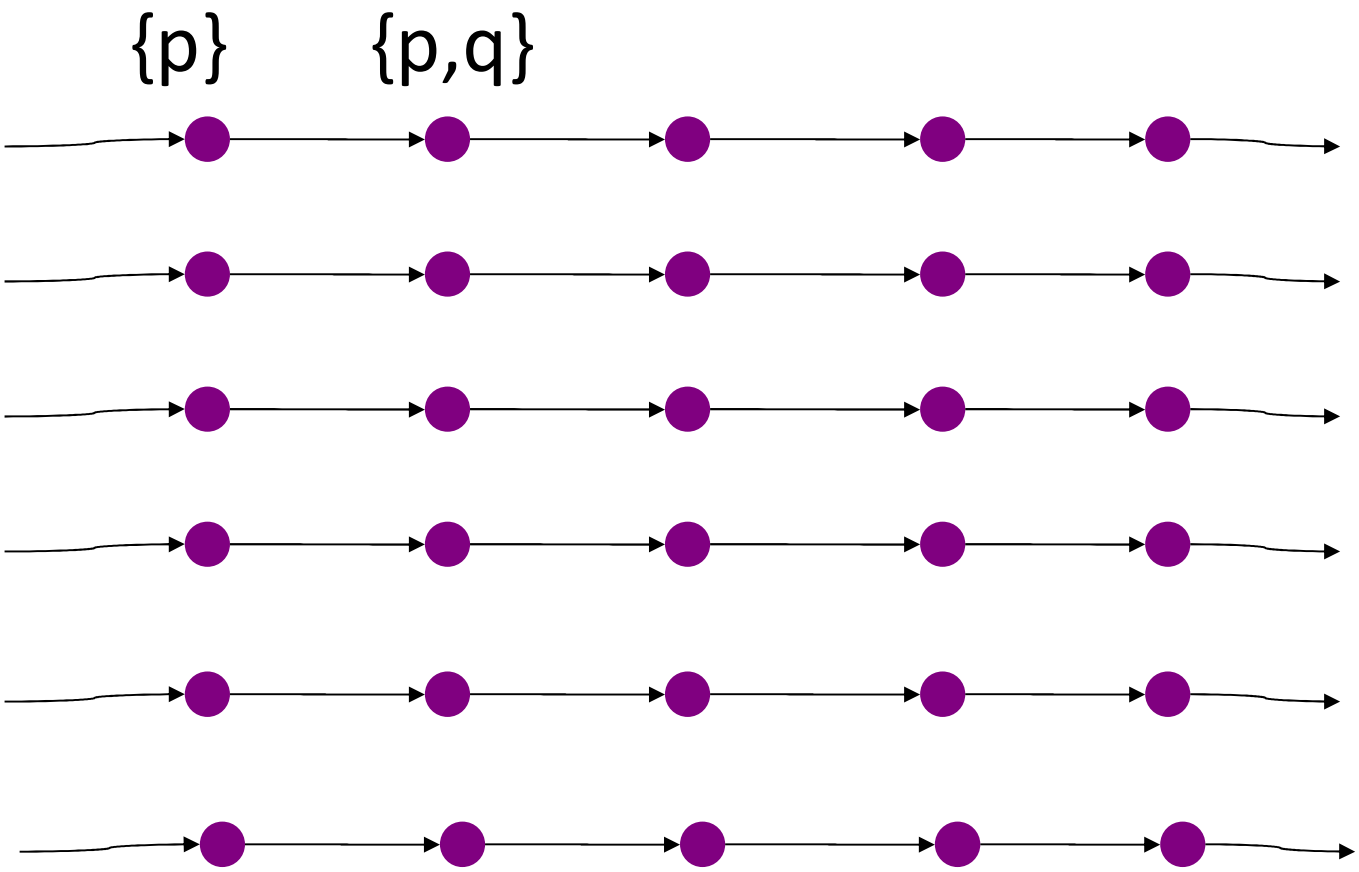


b) 行为规范

系统行为：系统模型

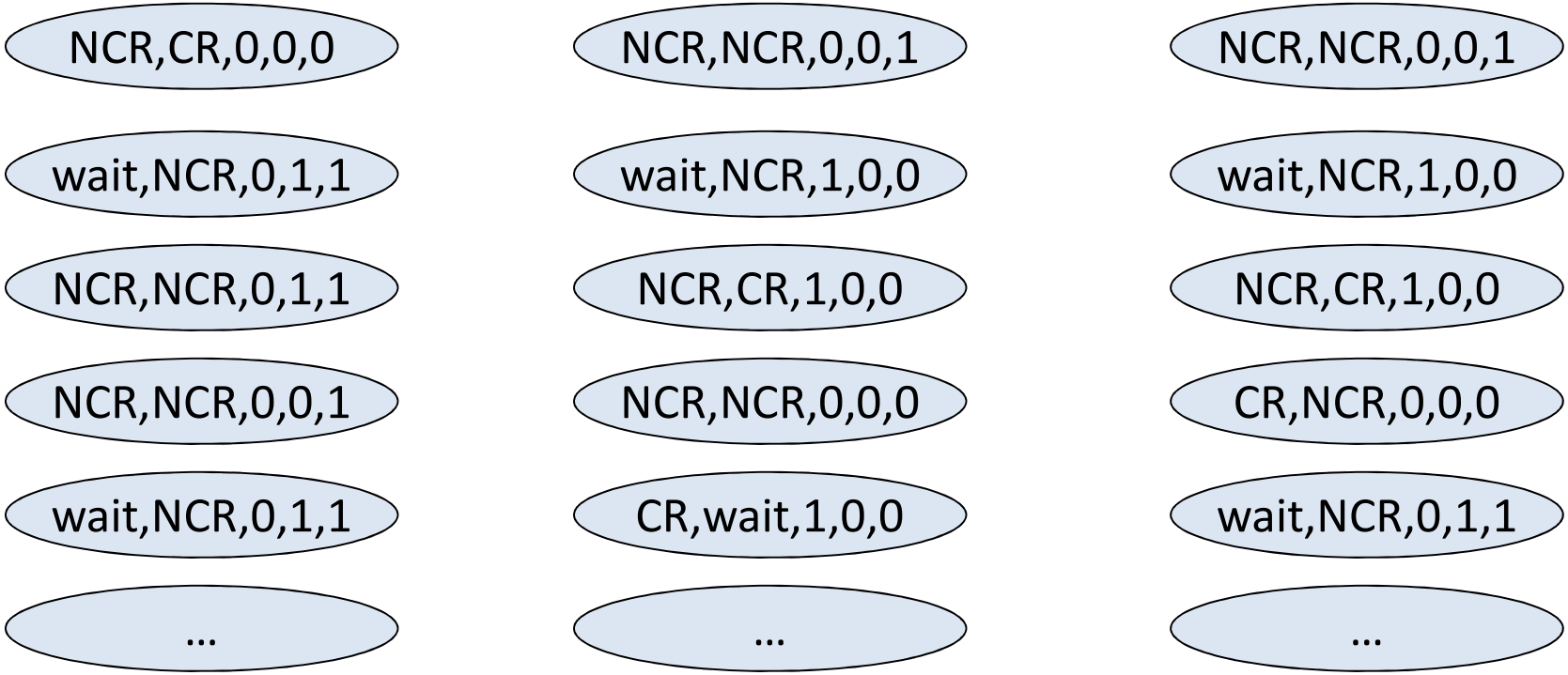
行为规范：系统性质的描述(被允许的系统行为)

系统性质的描述：可被允许的行为

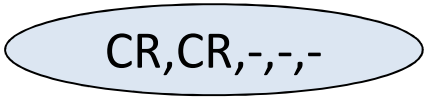


例2-互斥：满足互斥性质的序列

(a,b,x,y,t)

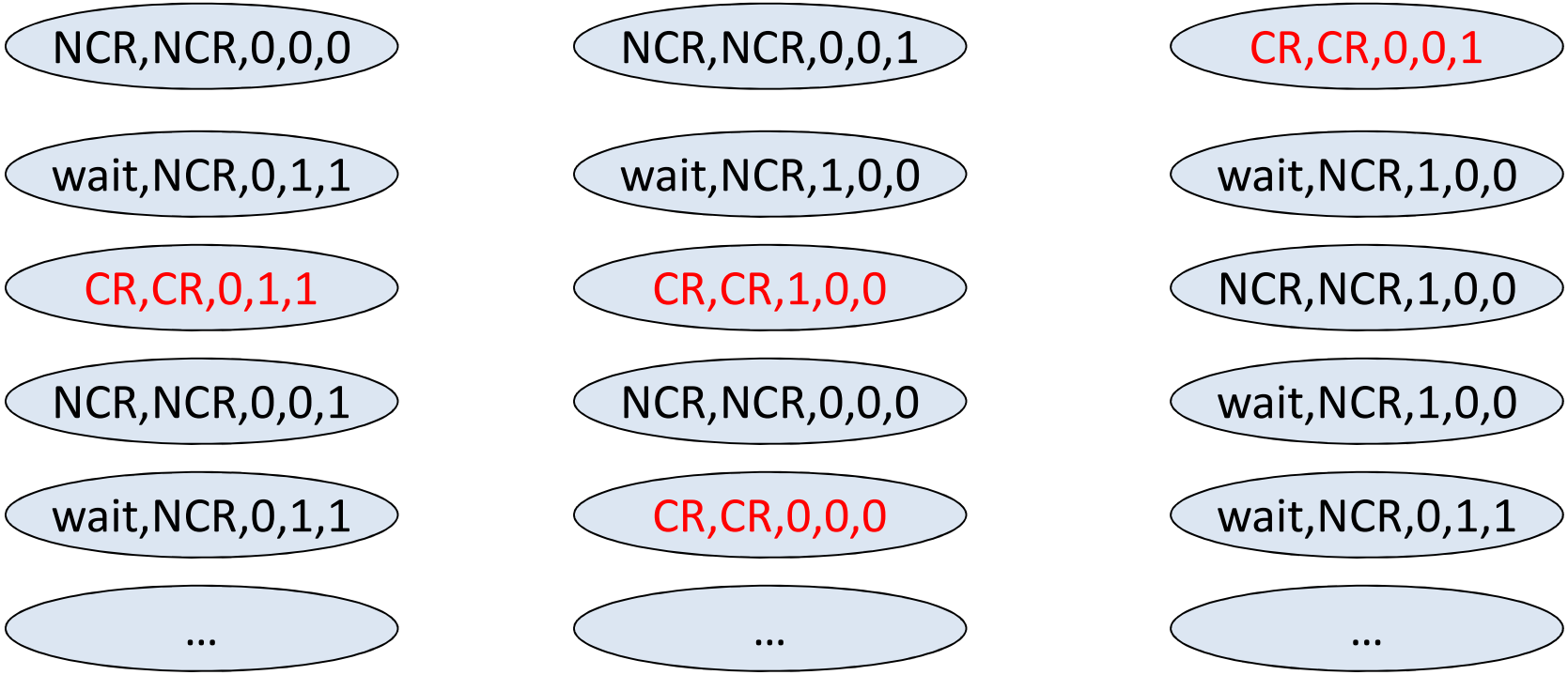


给定类型状态不可达：

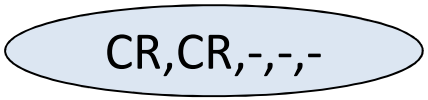


例2-互斥：不满足互斥性质的序列

(a,b,x,y,t)



给定类型状态不可达：



正则性质

正则语言(字母表Σ)：

- 空集∅是正则语言；
- 如果a ∈ Σ，则{ a }是正则语言；
- 如果A和B是正则语言，
  则A ∪ B，A • B，A* 是正则语言；

正则性质

ω正则语言：

- 如果A是非空正则语言，则A^{ω}是ω正则语言；
- 如果A是正则且B是ω正则，则A • B是ω正则语言；
- 如果A和B是ω正则语言，则A ∪ B是ω正则语言；

c). 行为与规范

行为: [[M]]

规范: B

行为满足规范： $[[M]] \subseteq B$?

例2-互斥：互斥性质

S = { (NCR,NCR,0,0,0), (NCR,NCR,0,0,1), ……}

A = { (a,b,x,y,t) | a,b∈{NCR,wait,CR}, x,y,t ∈ {0,1},
a ≠ CR or b ≠ CR
}

互斥性质：A^{ω}

系统满足互斥性质：$[[M]] \subseteq A^{\omega}$

例2-互斥：互斥性质(抽象表示)

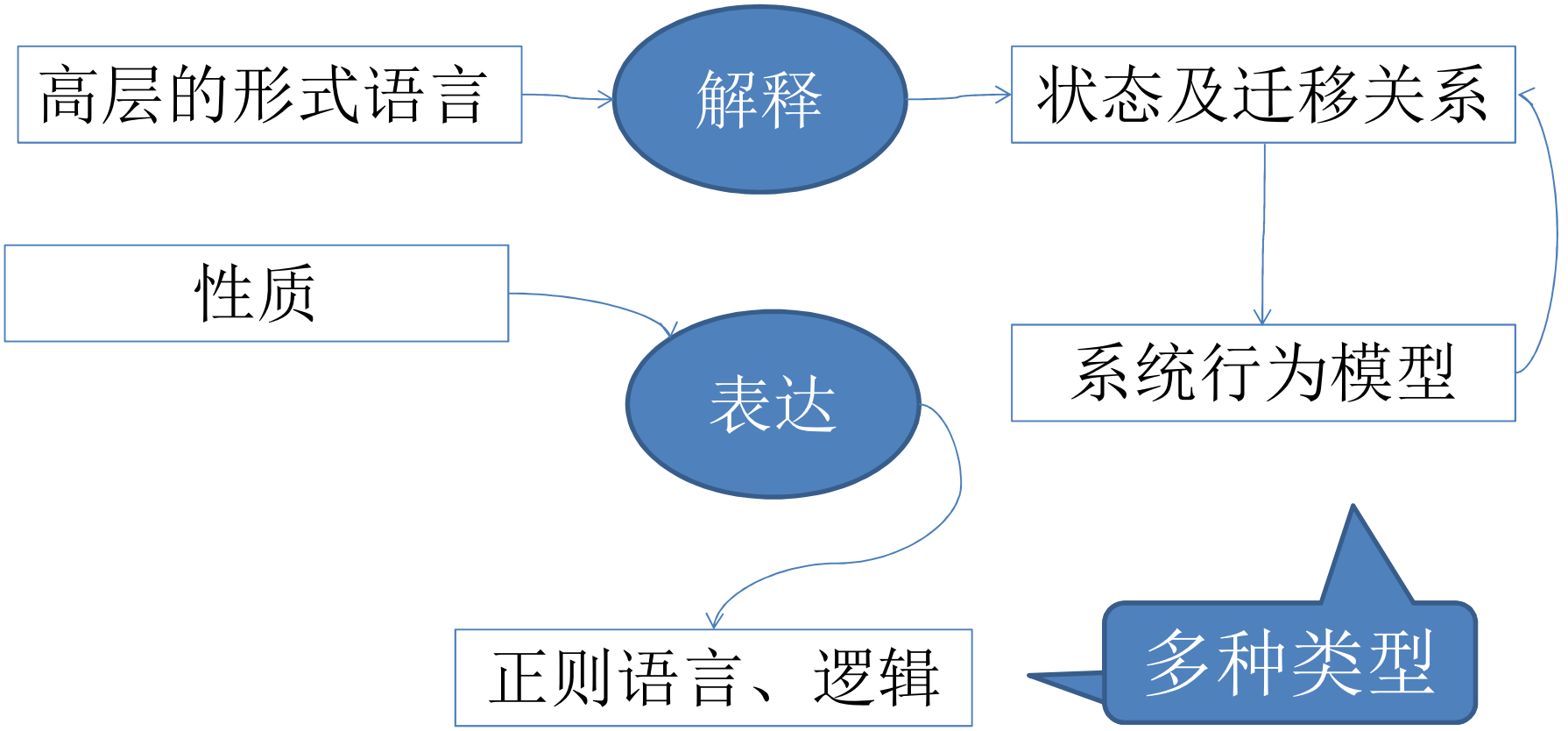
$S = \{ s_0, s_1, \ldots, s_{71} \}$

$A = \{ s_k \mid k < 48 \text{ or } (k\%24) < 16 \}$

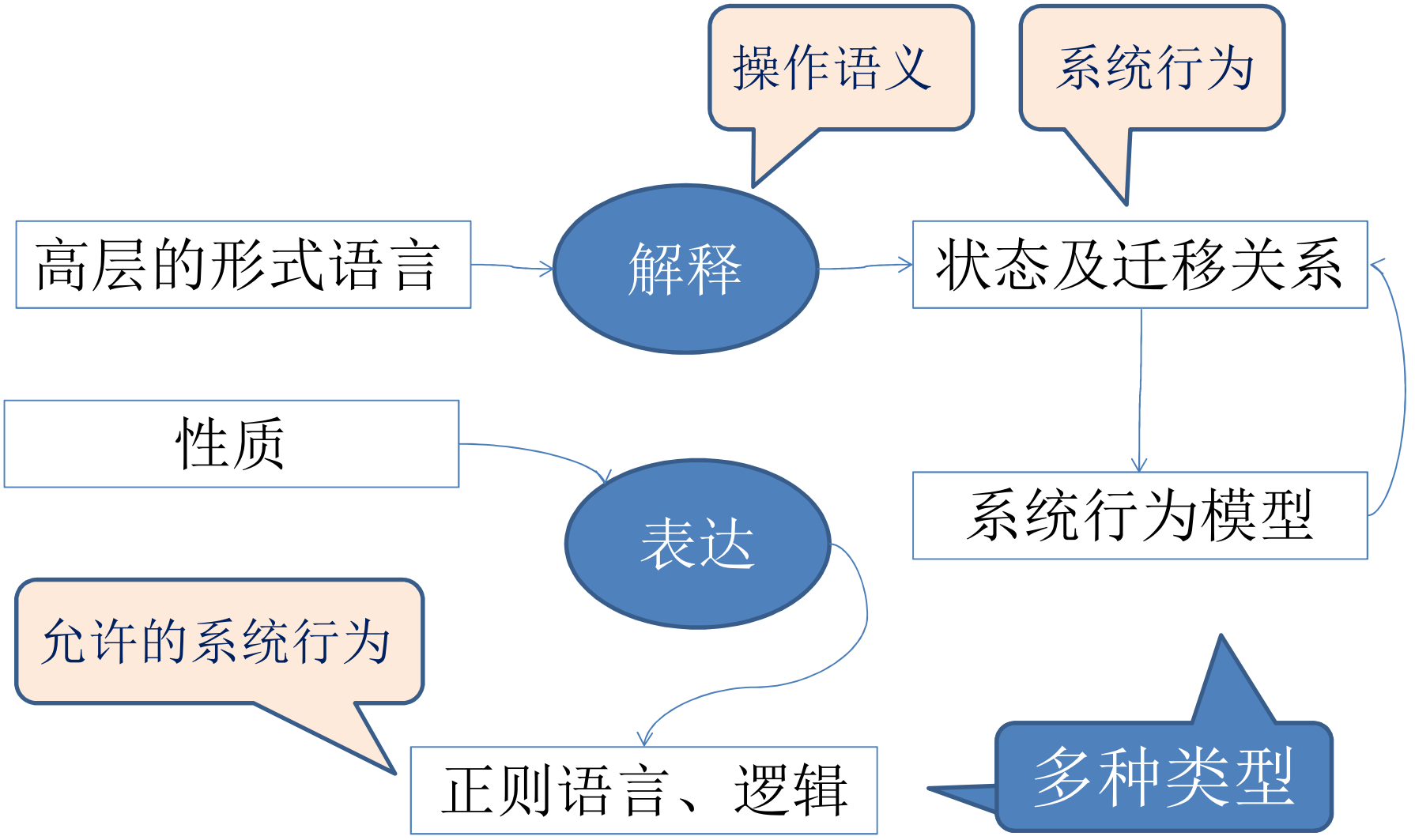
互斥性质：A^{ω}

系统满足互斥性质：$[[M]] \subseteq A^{\omega}$

软件系统行为模型与系统性质的表示



软件系统行为模型与系统性质的表示



软件系统行为模型与系统性质的表示

线性模型
树状模型
图状模型

正则语言、逻辑

三、形式化方法 – 课程关注点

- 软件正确性：软件工程及应用中的重要问题
- 形式化方法：保证软件系统正确性的重要手段
- 课程关注点：软件正确性与形式验证的理论

形式验证 - 程序性质的证明

- 演绎推理：

  把程序或系统模型当成是系统状态变化关系的逻辑描述，
  用演绎推理的方法来证明系统性质。

- 模型检测：

  把系统模型当成是系统状态迁移图的描述，
  用检查系统状态及系统运行路径的算法来证明系统性质。
  这类方法适用于状态数量有一定限制的有穷状态系统。

形式验证方法的基础理论

计算过程的抽象模型：系统模型

计算过程性质描述语言：时序逻辑

验证方法：演绎推理与模型检测

背景知识：离散数学、自动机、图算法

预修课程：数理逻辑与程序理论、形式语言与自动机理论

主要问题

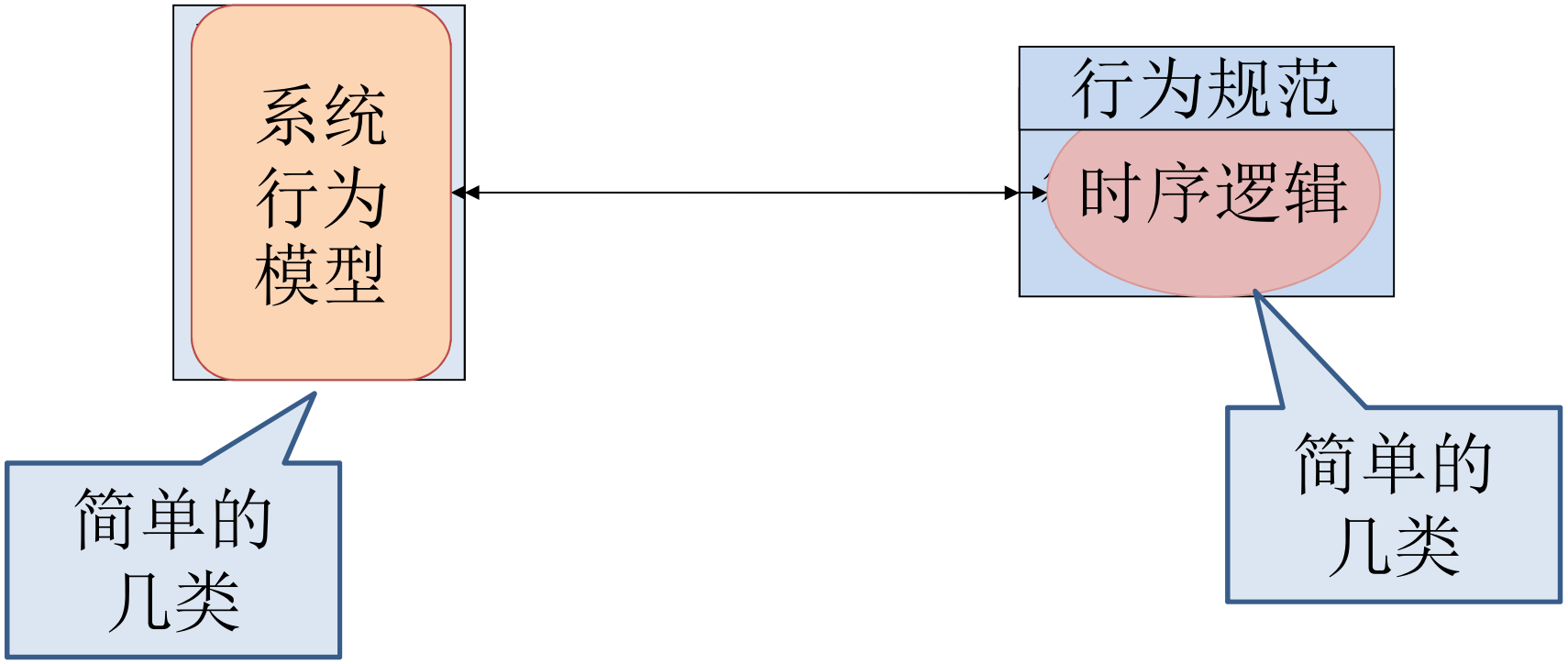
软件正确性

软件正确性的形式化分析与验证

软件正确性的形式化分析与验证的基本概念与方法

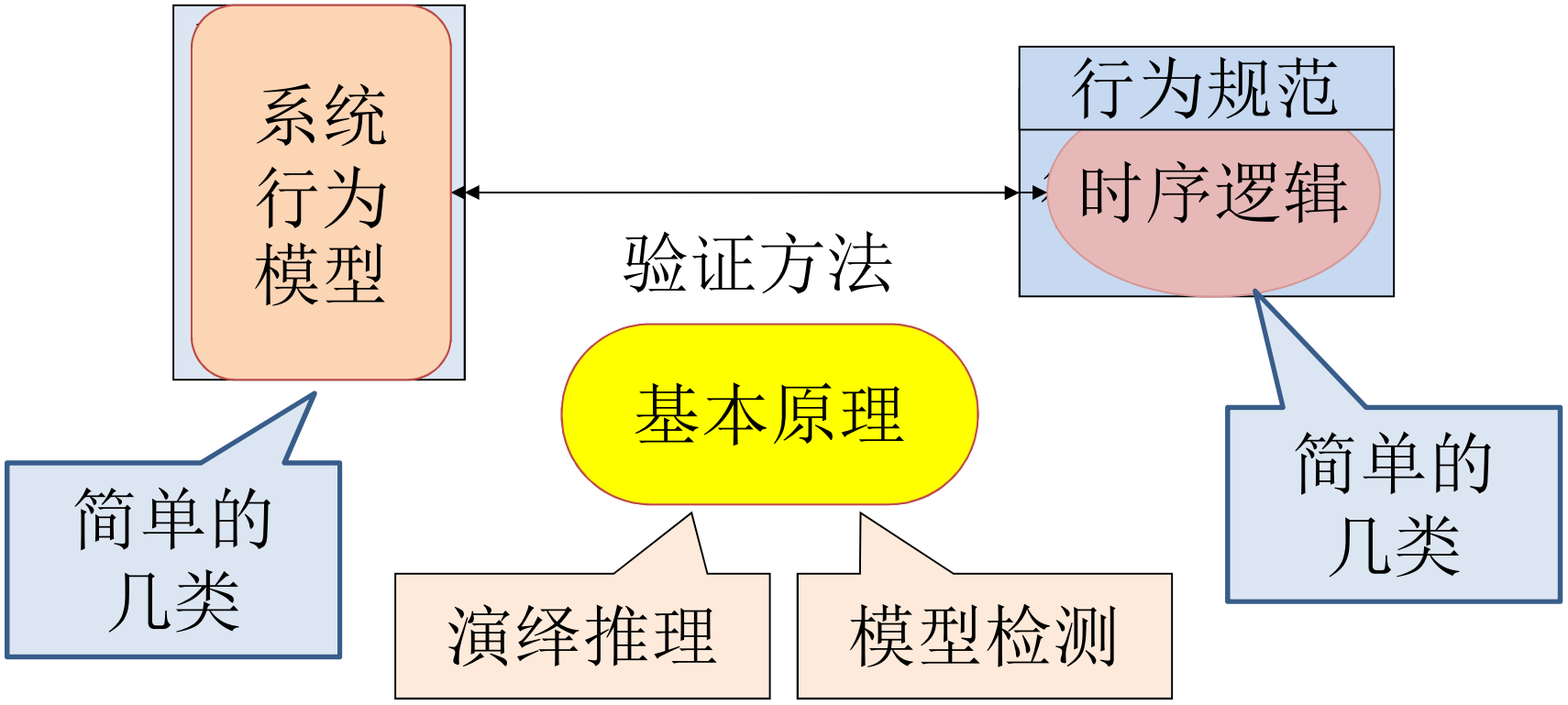
预期目标

掌握并能够综合应用以下知识：



预期目标

掌握并能够综合应用以下知识：



参考资料

J. Loeckx and K. Sieber.
The Foundation of Program Verification. John Wiley & Sons Ltd., 1984.

E. M. Clarke, O. Grumberg, D. Peled.
Model Checking. The MIT Press. 1999.

D. A. Peled.
Software Reliability Methods. Springer-Verlag. 2001.

C. Baier and J.-P. Katoen.
Principles of Model Checking. MIT Press. 2008.

K. R. Apt, F. S. de Boer, E.-R. Olderog.
Verification of Sequential and Concurrent Programs. Springer-Verlag. 2009.

形式化方法(讲义)
http://lcs.ios.ac.cn/~zwh/fm
http://lcs.ios.ac.cn/~zwh/fm/formalmethods.pdf

资料网页

http://lcs.ios.ac.cn/~zwh/fm

- 课时: 40
- 学分: 2.0
- 考试: 课堂开卷

- 思考题/练习题

思考：

1. 形式化方法与软件正确性

2. 软件系统行为与行为规范问题