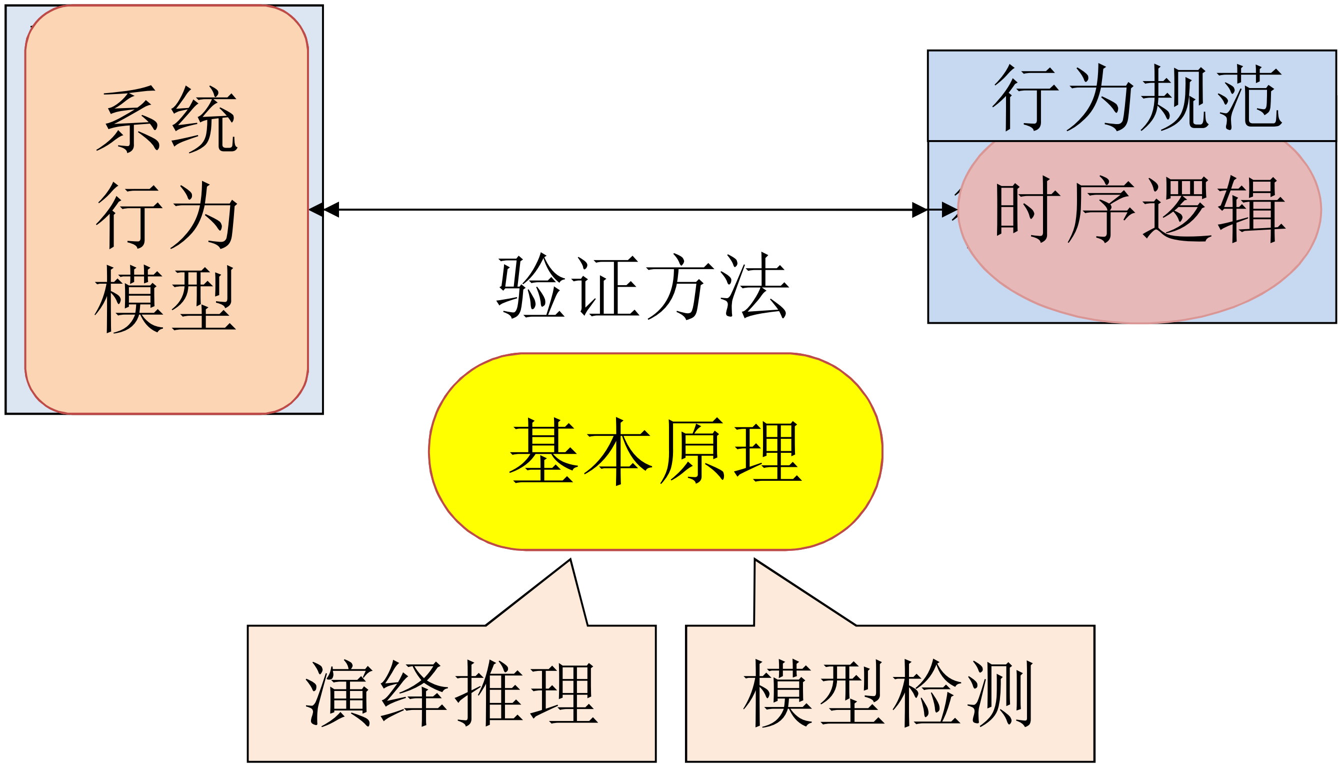
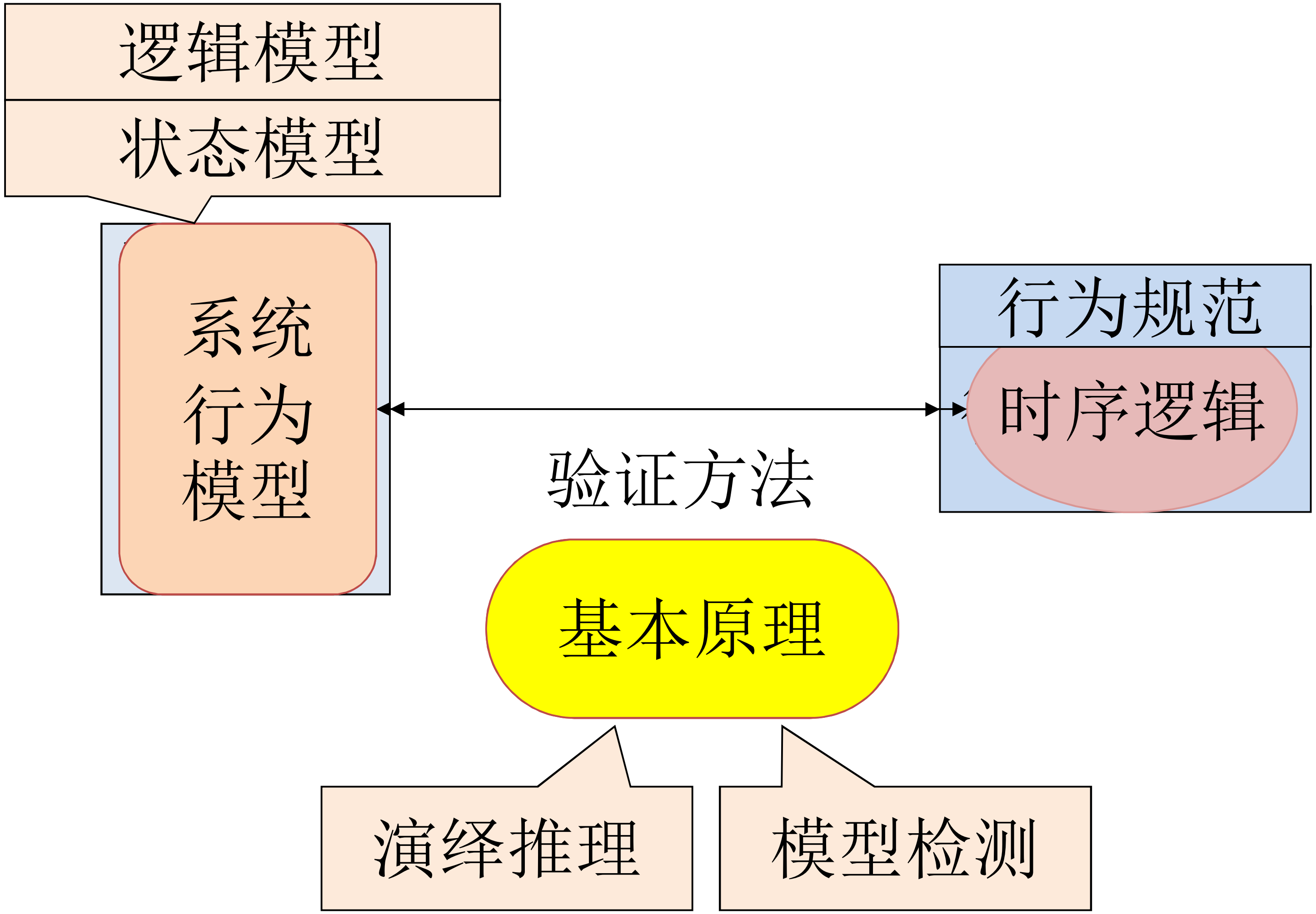
# Kripke Structures

中国科学院软件研究所
计算机科学国家重点实验室
张文辉
http://lcs.ios.ac.cn/~zwh/

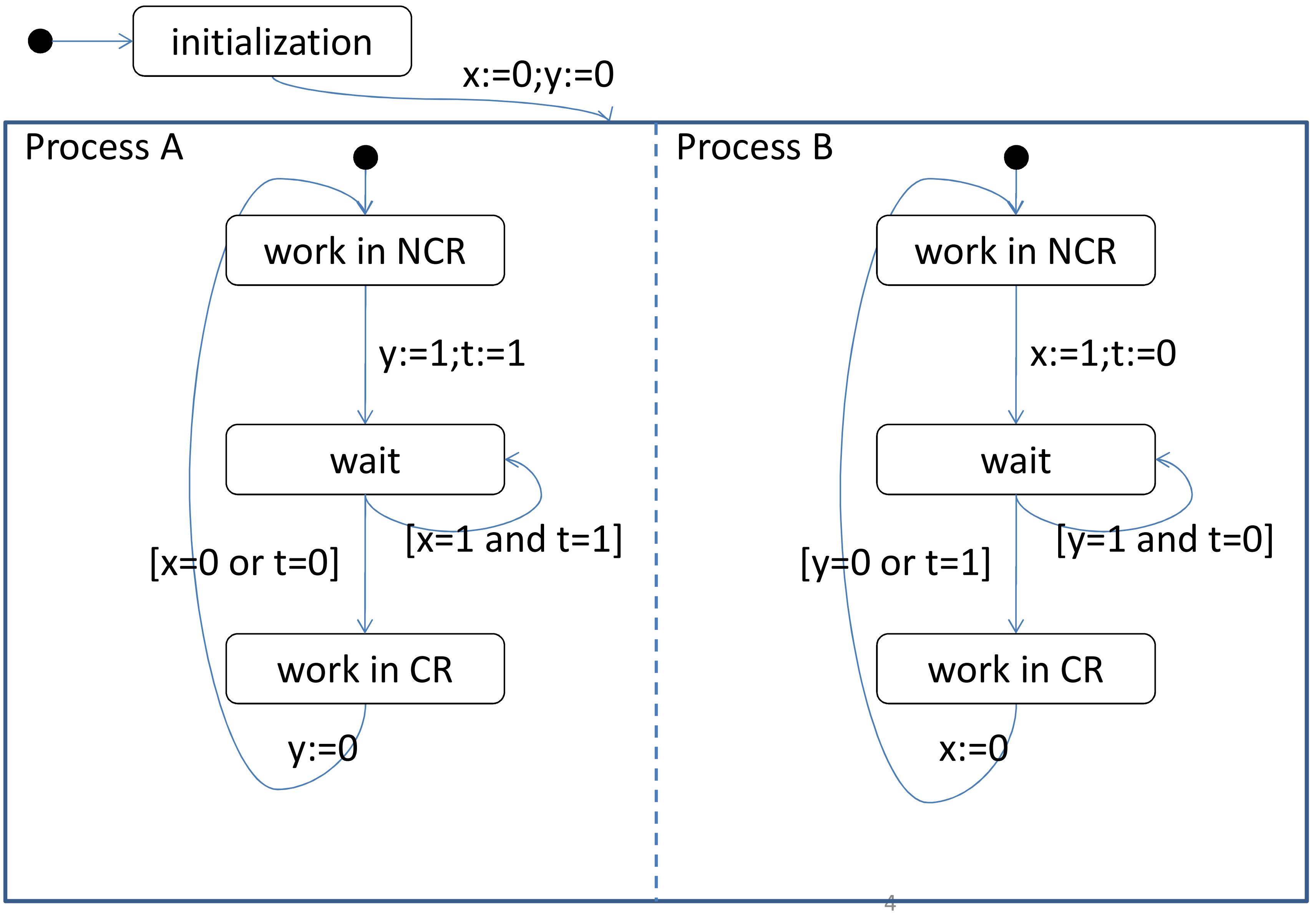
# 课程内容



# 课程内容



# 例子-互斥：状态图(State Diagram)



# 例子-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;

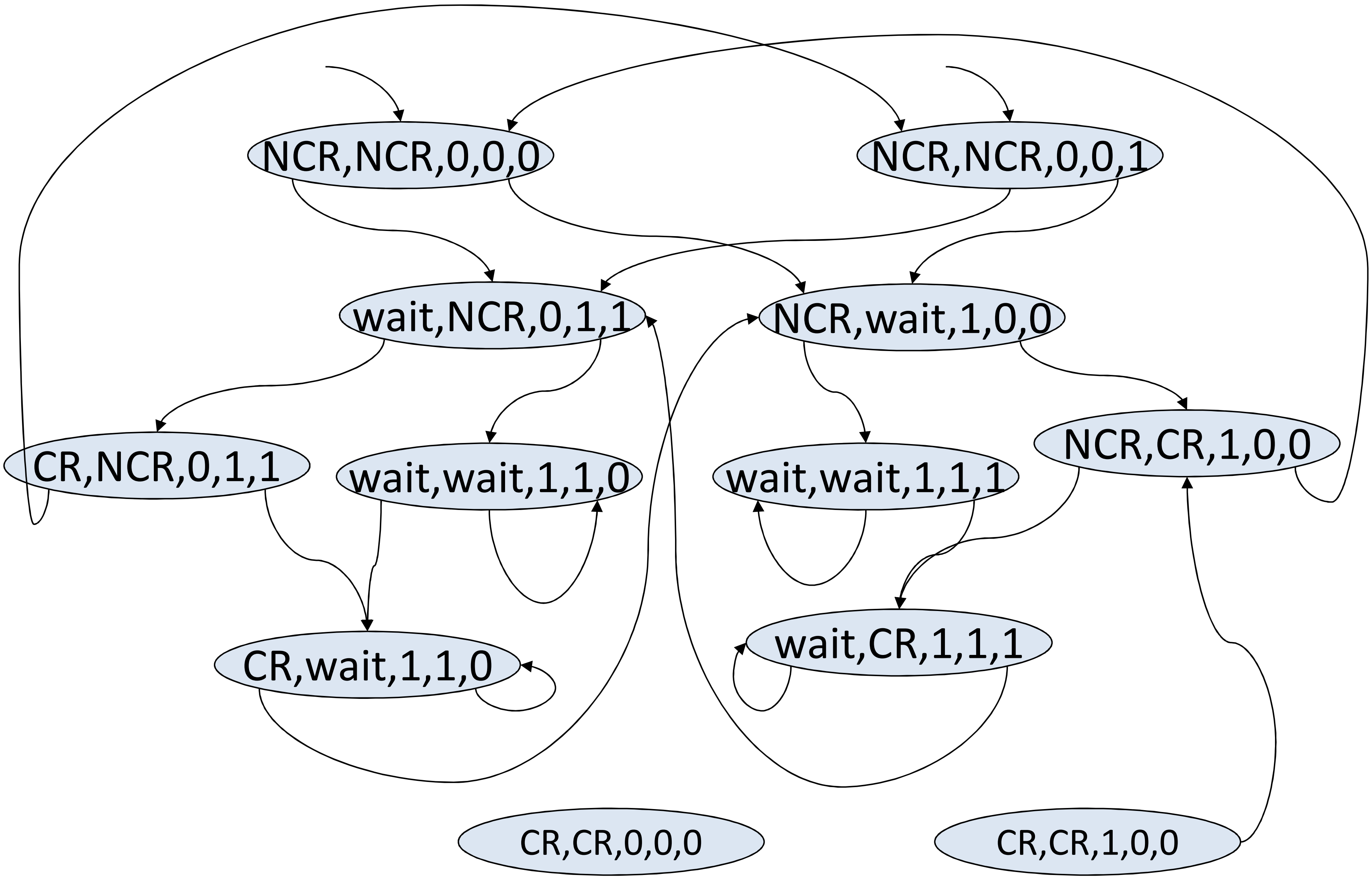
**Process A:**

| | |
|---|---|
| a=NCR | → (a,y,t):=(wait,1,1); |
| $a=wait \wedge (x=0 \vee t=0)$ | → (a):=(CR); |
| $a=wait \wedge \neg(x=0 \vee t=0)$ | → (a):=(wait); |
| a=CR | → (a,y):=(NCR,0); |

**Process B:**

| | |
|---|---|
| b=NCR | → (b,x,t):=(wait,1,0); |
| $b=wait \wedge (y=0 \vee t=1)$ | → (b):=(CR); |
| $b=wait \wedge \neg(y=0 \vee t=1)$ | → (b):=(wait); |
| b=CR | → (b,x):=(NCR,0); |

# 例子-互斥：可达状态 + 部分不可达状态



# 例子-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

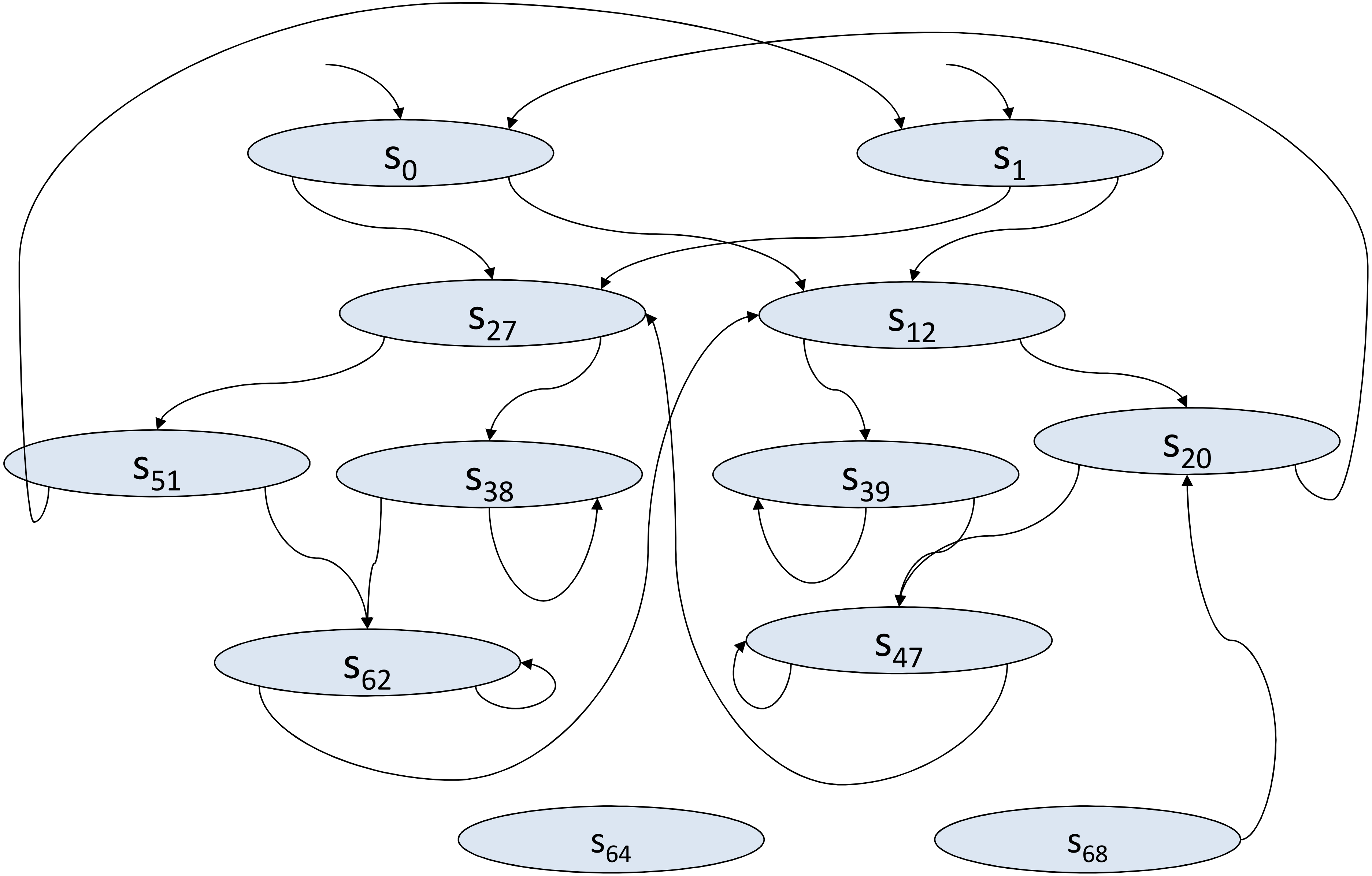
其中$s_i$代表 (a,b,x,y,t)
当且仅当　$i=a*24+b*8+x*4+y*2+t$

其中NCR代表0, wait代表1, CR代表2

例如:

$s_0$ = (0,0,0,0,0) = (NCR,NCR,0,0,0)
$s_{48}$ = (2,0,0,0,0) = (CR,NCR,0,0,0)

# 例子-互斥：状态迁移图(Kripke结构)



# 内容

- Kripke结构
- 安全性质相关概念及验证方法
- 必达性质相关概念及验证方法

# (I) Kripke Structures

Definition

A Kripke structure is a triple $K=\langle S,R,I\rangle$

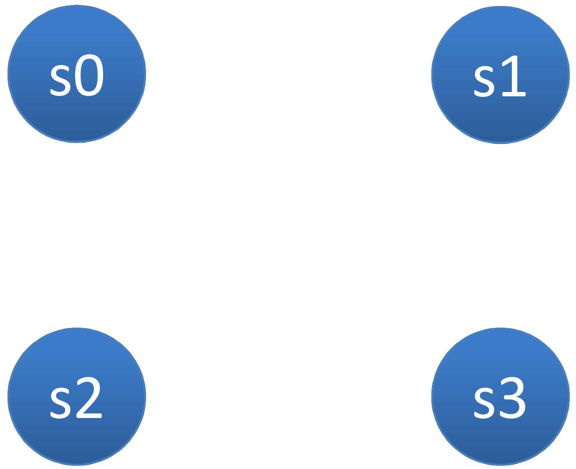
- $S$ : A set of states
- $R \subseteq S \times S$ : A total transition relation
- $I \subseteq S$ : A set of initial states

R is total, if $\forall s.\exists s'.(s,s') \in R$

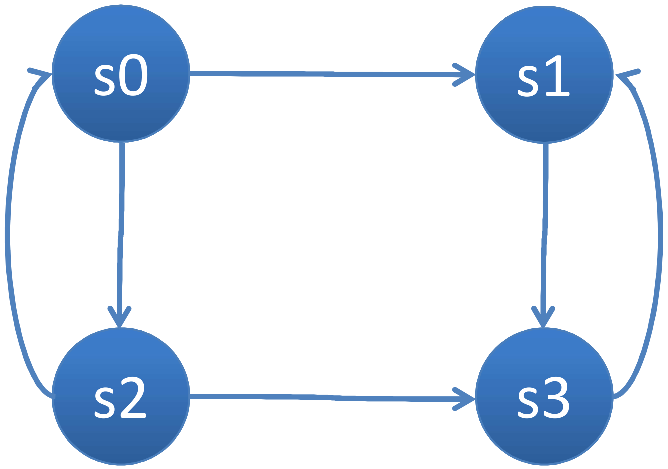
# Kripke Structures

- Basic Concepts
  - States, Transition Relation, Initial States
  - Successors, Predecessors
  - Reachable States, Reachability Relation
  - Paths (Finite and Infinite), Computation, Behavior
  - Properties
- Basic System Properties
  - Reachability, Safety
  - Avoidability, Inevitability

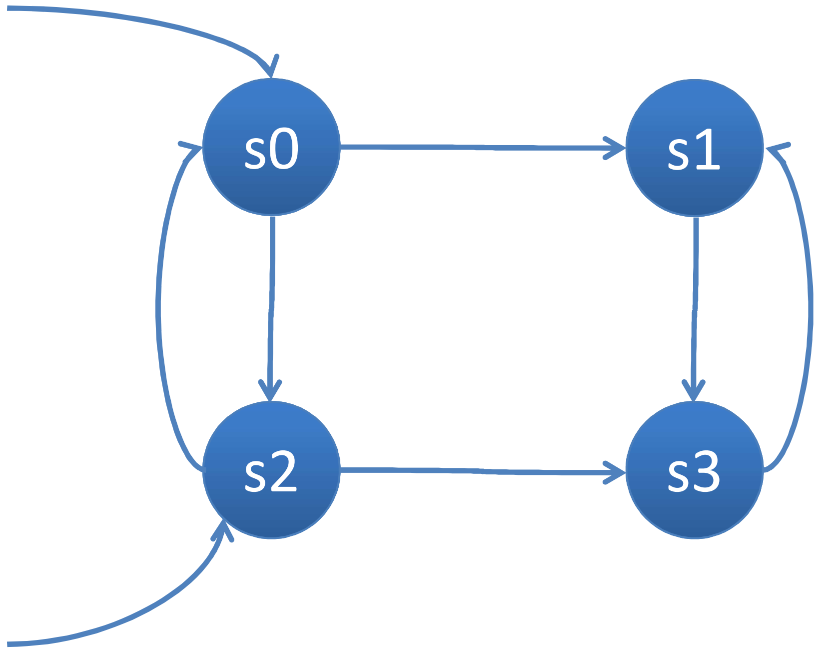
# Example: S



# Example: R



# Example: I



# Basic Concepts

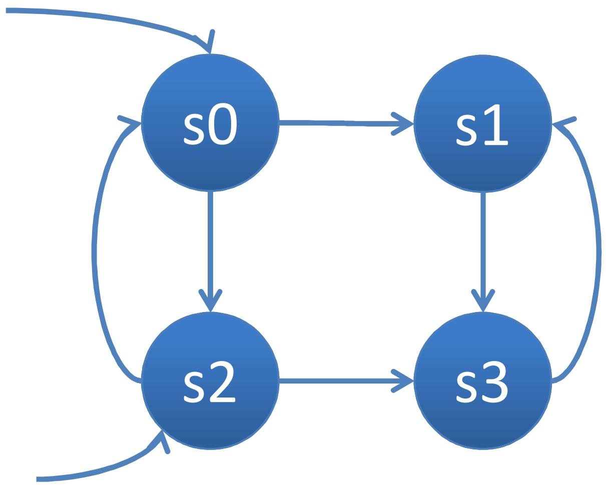
# 1. Successors

s' is a successor of s, if s $\rightarrow$ s', i.e., R(s,s')

The set of successors of s: R(s)

The set of successors of X: $R(X) = \cup_{s \in X} R(s)$

R is total if $R(s) \neq \emptyset$ for all $s \in S$




Example:

R(s0) = { s1,s2}

R({s0,s1}) = { s1,s2, s3 }

# 1a. Predecessors

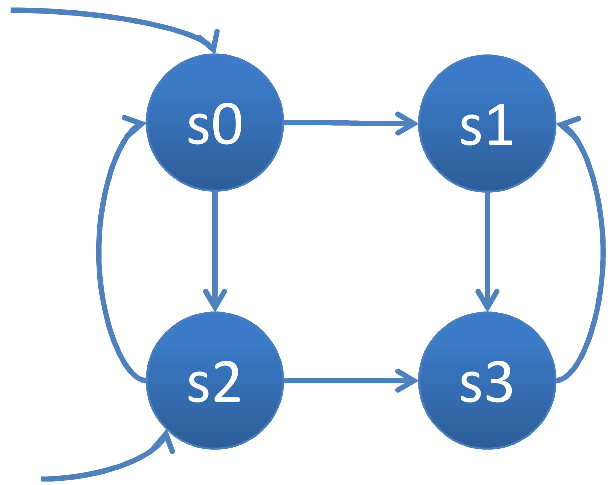
s is a predecessor of s', if s → s', i.e., $R(s,s')$, $R^{-1}(s',s)$

The set of predecessors of s:

$R^{-1}(s)$

The set of predecessors of X :

$R^{-1}(X) = \cup_{s \in X} R^{-1}(s)$




Example:

$R^{-1}(s0) = \{ s2\}$

$R^{-1}(\{s0,s1\}) = \{ s0,s2, s3 \}$

# 2. Reachable States (from s)

$\rightarrow^*$ :

the reflexive and transitive closure of $\rightarrow$

s’ is reachable from s, if

$s \rightarrow^* s'$

The set of states reachable from s is $\{ s' \mid s \rightarrow^* s' \}$:

$R^*(s)$.

# 2. Reachable States (from A,K)

s' is reachable from A, if $s \rightarrow^* s'$ for some $s \in A$

The set of states reachable from A is $\{ s' \mid s \rightarrow^* s', s \in A \}$: $R^*(A)$.

The set of states reachable in K is $R^*(I) = \{ s' \mid s \rightarrow^* s', s \in I \}$: $Rh(K)$.

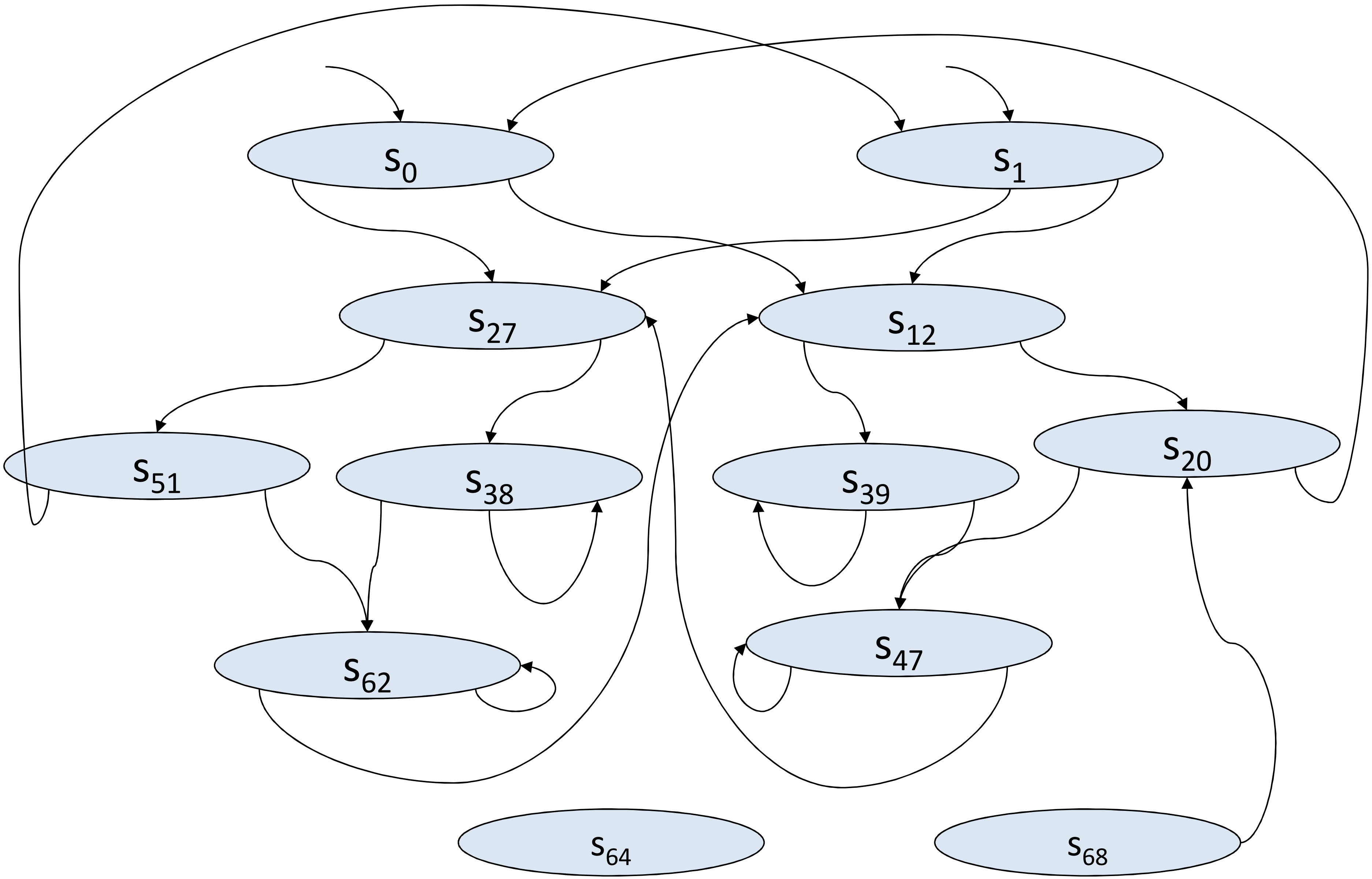
# Reachable States

Reachable states of s: $R^*(s) = \{ s' \mid s \rightarrow^* s' \}$

Reachable states of X: $R^*(X) = \cup_{s \in X} R^*(s)$

Reachable states of K: $Rh(K) = R^*(I)$

# 例子-互斥：状态迁移图(Kripke结构)



# 2a. Reachability Relation

A set Y is reachable from s,

if there is a state s’ of Y,

such that $s \rightarrow^* s'$

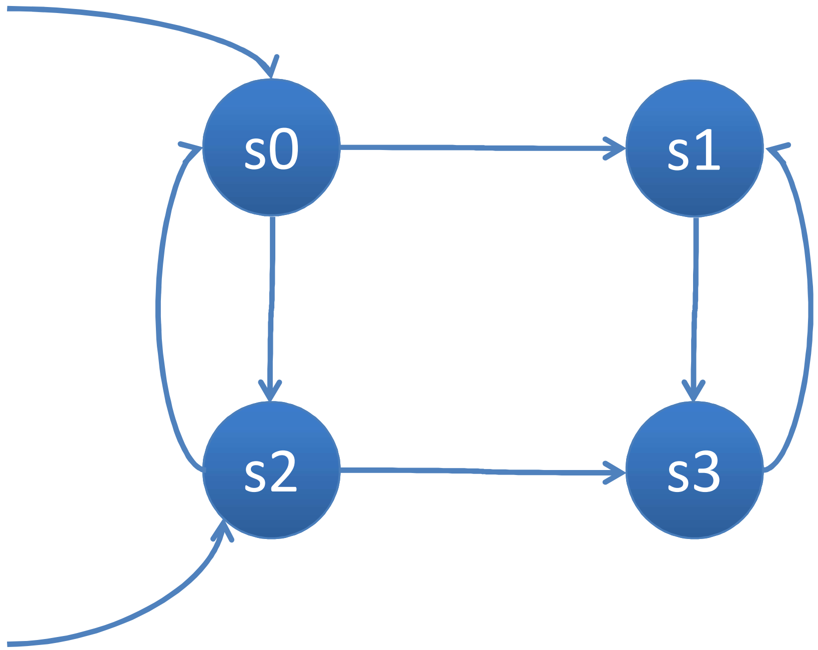
A set Y is reachable from X,

if there is a state s’ of Y and a state s of X,

such that $s \rightarrow^* s'$

$X \rightarrow^* Y$

# Example:




The set of reachable states ($Y=R^*(X)$):

X = { s1 }, Y = { s1, s3 }; X = { s0,s1 }, Y = { s0,s1,s2,s3 }

Reachability relation ($X \rightarrow^* Y$):

X = { s1 }, Y = { s2, s3 }; X = { s0,s1,s2 }, Y = { s3 }

# 3. Path

Definition

An infinite path is an infinite sequence of S:

$s_0\ s_1\ s_2\ ....$

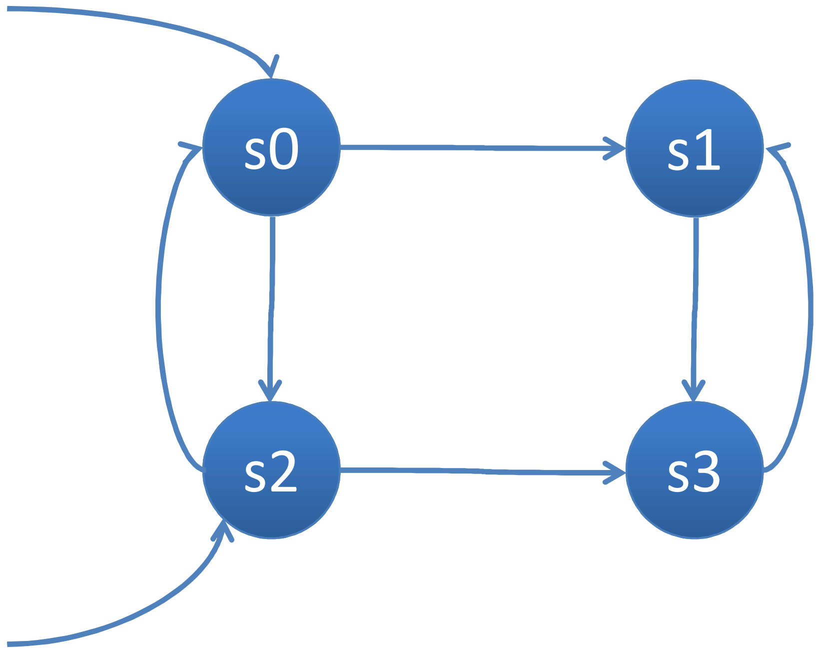
such that $s_i \rightarrow s_{i+1}$ for all $i \geq 0$

Definition

A finite path is a finite prefix of an infinite path:

$s_0 \ldots s_n$

# Example: Paths




Infinite paths:

s0s1s3s1s3s1…

s0(s1s3) $^{\omega}$

s1s3s1s3s1s3s1…

Finite paths:

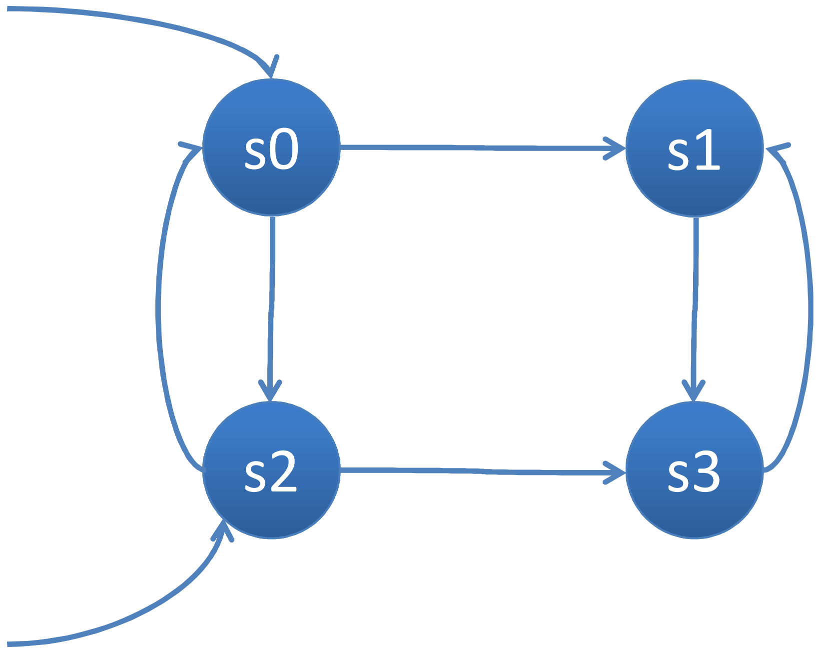
s0s1s3s1

s1s3s1s3s1

# 3a. Computation

A **computation** of K is an infinite path $s_0\, s_1\, s_2 \ldots.$ such that $s_0 \in I$

# Example: Computations




s0(s1s3) $^{\omega}$

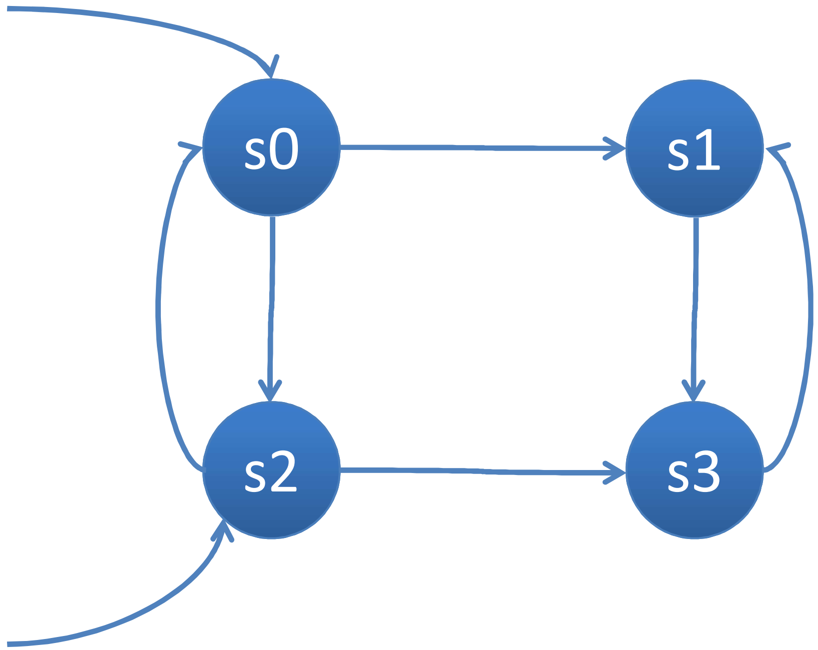
s2(s3s1) $^{\omega}$

s2(s0s2)*(s3s1) $^{\omega}$

# 3b. Behavior

The behavior of K is the set of computations of K, denoted [[K]].

# Example: Behavior




$(s0s2)^{\omega}$ $\quad$ $(s2s0)^{\omega}$

$s0(s2s0)^*(s1s3)^{\omega}$ $\quad$ $s2(s0s2)^*(s3s1)^{\omega}$

$s0s2(s0s2)^*(s3s1)^{\omega}$ $\quad$ $s2s0(s2s0)^*(s1s3)^{\omega}$

# 4. Properties

A property is represented by a set of states

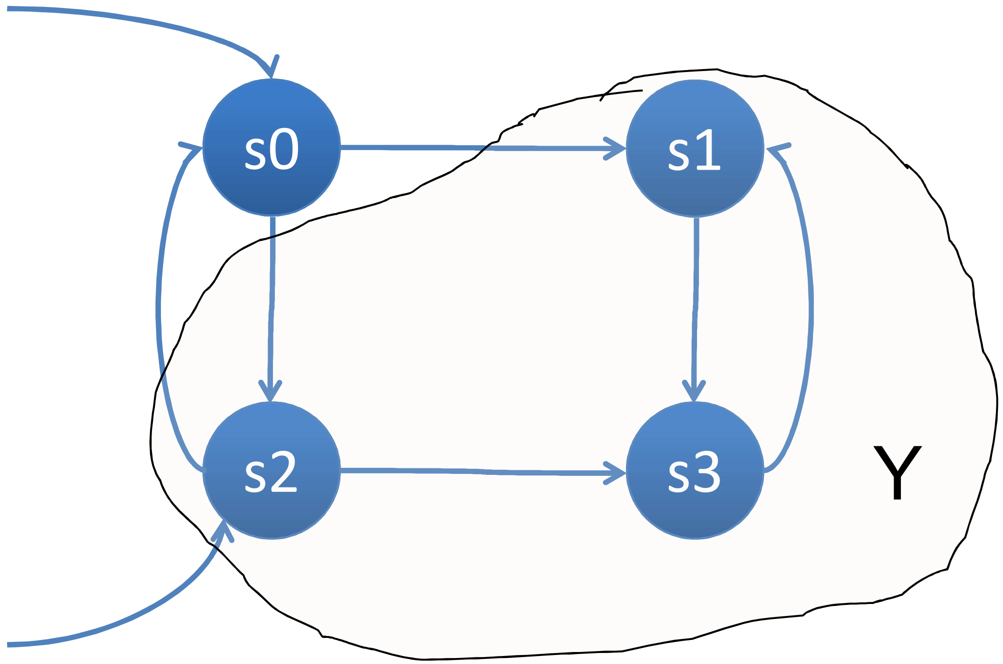
- X
- Y

DEF: s satisfies Y, if $s \in Y$.

DEF: X satisfies Y, if for all $s \in X$, s satisfies Y, i.e., $X \subseteq Y$

s is called a Y-state, if s satisfies Y.

# Example: Y-States



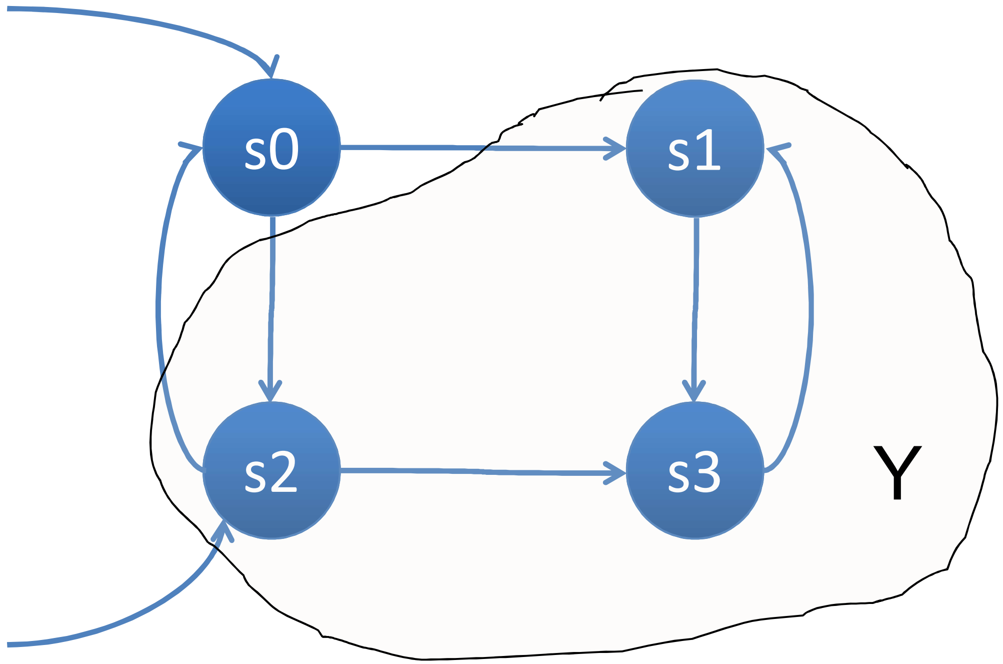

s1, s2 and s3 are Y-states

# 4a. Path Properties

A path (computation) $\pi$
is called a Y-path (Y-computation),
if all states appear on $\pi$ satisfy Y.

A path (computation) $\pi$ reaches (passes) Y-states,
if some state appears on $\pi$ satisfies Y.

# Example: Y-Paths, Y-Computations




(s1s3) is a finite Y-path
$(s1s3)^{\omega}$ is a Y-path
$s2(s1s3)^{\omega}$ is a Y-computation

# Complementary Sets - Negation

Y-States, Y-Paths, Y-Computations

Notation: $\neg Y \equiv (S \setminus Y)$

$\neg$Y-States, $\neg$Y-Paths, $\neg$Y-Computations

# Reachability

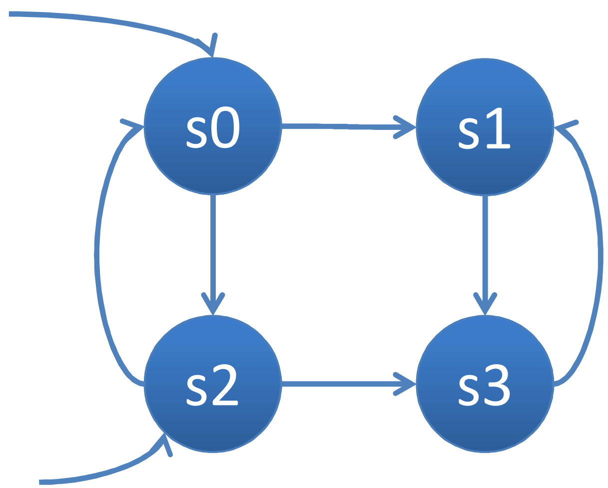
Y is a reachable, if

there is a computation of K that reaches a Y-state.

Reachability Problem:

Given a set Y.

Is Y reachable?



# Avoidability

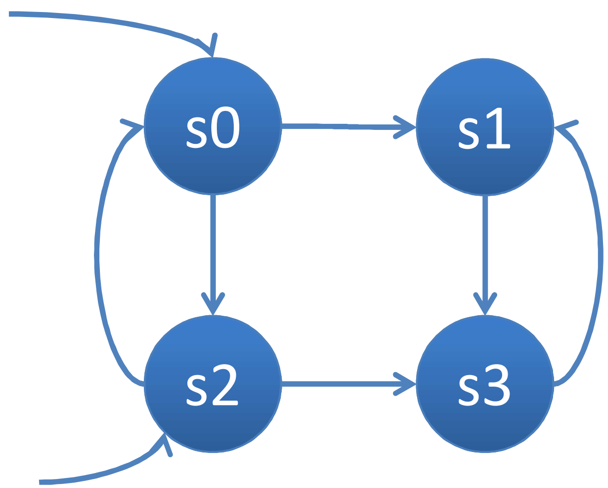
Y is avoidable, if

there a computation that does not reach any Y-state.

Avoidability Problem:

Given a set Y.

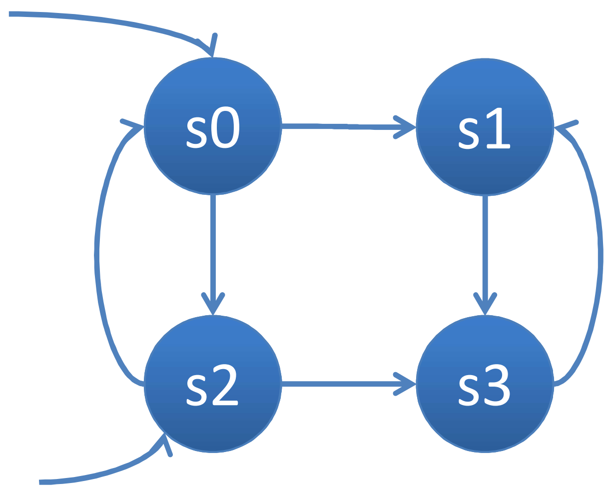
Is Y avoidable?



# Basic System Properties

(1) Reachability, Safety

(2) Avoidability, Inevitability



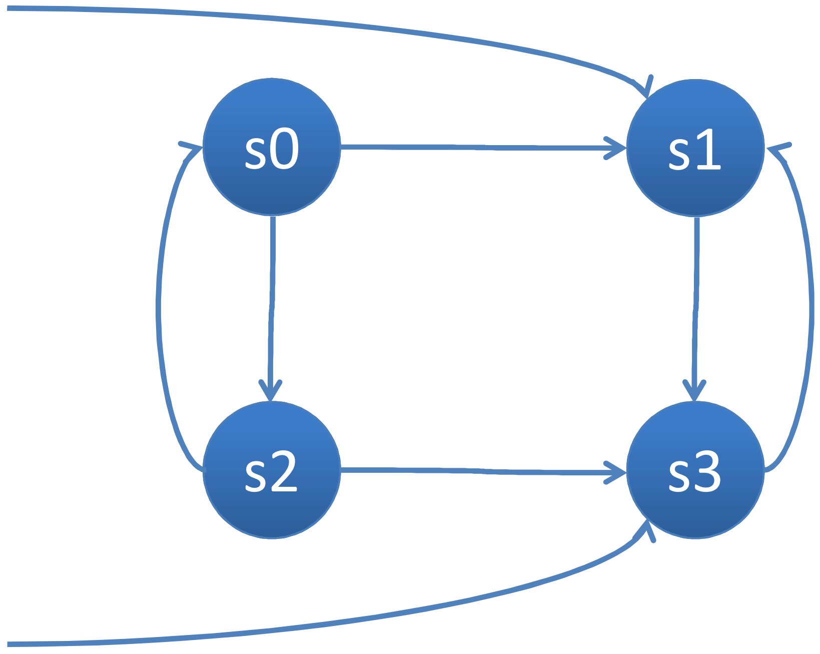
# (II) Reachability Property - Possibility

Let Y be a set of states.

Y is a reachability property, if

there is a computation of K that reaches a Y-state.

# Example: {s0,s2}, {s0,s1,s2}

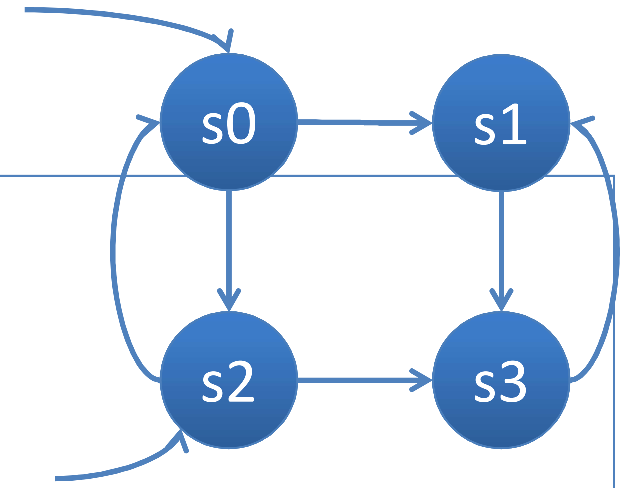


# Reachability Problem

Given a set $A \subseteq S$.

Is A a reachability property?

# Reachability Analysis




Let $K=\langle S,R,I\rangle$ and $A \subseteq S$.

```
BOOL ReachabilityAnalysis(K,A)
  {     w:=I;
        repeat until w={};
                s:=w.getElement(); if (s in A) return true;
                visited[s]:=true;
                for each (s' in R(s)),
                        if (visited[s']=false) w.putElement(s');
                w.removeElement(s);
        return false;
  }
```

# Reachability Analysis

Let K=<S,R,I> be a Kripke structure, and A$\subseteq$S.

**Proposition:**

ReachabilityAnalysis(K,A) =true

Iff

A is a reachability property

# Reachability Analysis (Alternatively)

Let $K = \langle S,R,I \rangle$ be a Kripke structure, and $A \subseteq S$.

**Proposition:**

A is a reachability property, iff $Rh(K) \cap A \neq \emptyset$

➔ Reachability analysis based on fixpoint computation

# Fixpoints

- Let f: A $\rightarrow$ A be a function.
- $a \in A$ is a fixpoint of f, if f(a)=a
- Questions:

  Existence of a fixpoint?

  Computation of a fixpoint?

# Fixpoints

- Let S be a finite set and $(2^S, \subseteq)$ be a lattice.
- Let f: $2^S \rightarrow 2^S$ be a monotonic function.

- The least fixpoint of f denoted $\mu f$ (or $\mu Z.f(Z)$) is:

$\mu f = \cup \{ f^k(\emptyset) \mid k=0,1,2,\ldots \}$

- The greatest fixpoint of f denoted $\nu f$ (or $\nu Z.f(Z)$) is:

$\nu f = \cap \{ f^k(S) \mid k=0,1,2,\ldots \}$

# Reachability Analysis (FP)

Let $K=<S,R,I>$ be a Kripke structure, and $A \subseteq S$.

Let $f: 2^S \rightarrow 2^S$ be defined by $f(w) = I \cup R(w)$

THEN

$Rh(K) = \mu f$

A is a reachability property, iff $\mu f \cap A \neq \emptyset$

# Reachability Analysis (FP)

Let K=<S,R,I> be a Kripke structure, and A $\subseteq$S.

```
SET leastfixpoint(f,K)
{
   w={};
   repeat
         w':=w;
         w :=f(w,K);
   until w'=w;
   return w;
}
```

# Reachability Analysis (FP)

Let K=<S,R,I> be a Kripke structure, and $A \subseteq S$.

| SET leastfixpoint(f,K) | SET f(w,K) |
|---|---|

```
SET leastfixpoint(f,K)
{
    w={};
    repeat
        w':=w;
        w :=f(w,K);
    until w'=w;
    return w;
}
```

```
SET f(w,K)
{
    return I ∪ R(w);
}

BOOL ReachabilityAnalysisFP(K,A)
{
    w=leastfixpoint(f,K);
    return ( w ∩ A ≠ ∅ ) ;
}
```

# Reachability Analysis (FP)

Let K=<S,R,I> be a Kripke structure, and A $\subseteq$S.

**Proposition:**

ReachabilityAnalysisFP(K,A) =true

Iff

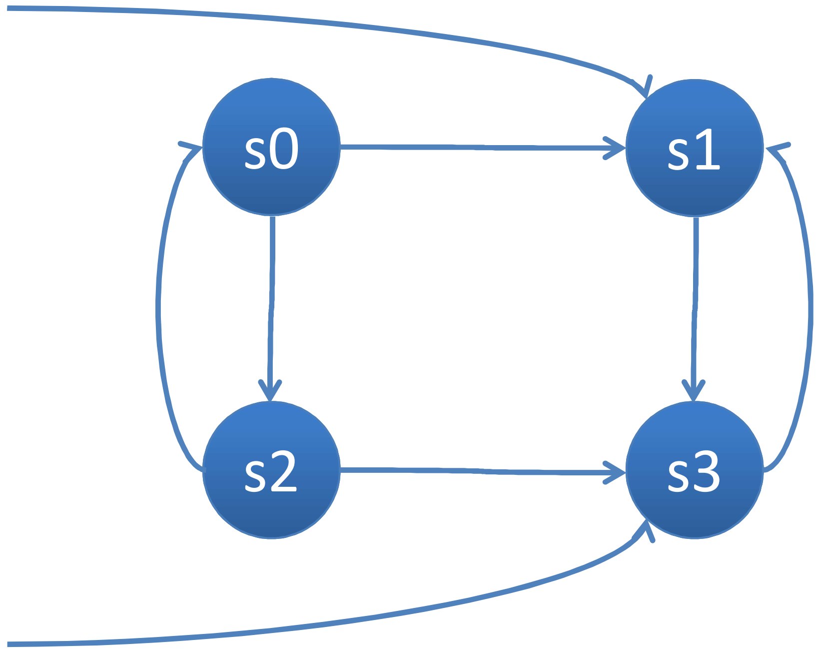
A is a reachability property

# Safety Property - Universality

Let Y be a set of states.

Y is a safety property, if
every computation is a Y-computation.

# Example: {s2,s3}, {s1,s2,s3}



# Reachability & Safety

Safety is a dual property of reachability.

**Proposition**
A system K=<S,R,I> is safe with respect to Y, iff
$Rh(K) \subseteq Y$.

**Corollary [Duality]**
A system K=<S,R,I> is safe with respect to Y, iff
$\neg Y$ is not reachable in K.

# Safety Analysis

Let K=<S,R,I> be a Kripke structure, and A $\subseteq$S.

```
BOOL SafetyAnalysis(K,A)
 {   w:=I;
     repeat until w={};
          s:=w.getElement(); if (s not in A) return false;
               visited[s]:=true;
               for each (s' in R(s)),
                    if (visited[s']=false) w.putElement(s');
               w.removeElement(s);
      return true;
   }
```

# Safety Analysis

Let K=<S,R,I> be a Kripke structure, and A $\subseteq$S.

**Proposition:**

SafetyAnalysis(K,A) =true

Iff

A is a safety property

# Safety Analysis (FP)

Let K=<S,R,I> be a Kripke structure, and A ⊆S.

```
SET leastfixpoint(f,K)
{
    w={};
    repeat
        w':=w;
        w :=f(w,K);
    until w'=w;
    return w;
}
```

```
SET f(w,K)
{
    return I ∪ R(w);
}

BOOL SafetyAnalysisFP(K,A)
{
    w=leastfixpoint(f,K);
    return ( w ∩ (¬A) = ∅ ) ;
}
```

# Safety Analysis (FP)

Let K=<S,R,I> be a Kripke structure, and A $\subseteq$S.

**Proposition:**

SafetyAnalysisFP(K,A) =true

Iff

A is a safety property

# Reachability & Safety

SafetyAnalysis(K,A) =true $\Leftrightarrow$
ReachabilityAnalysis(K,$\neg$A) =false

# Deductive Safety Analysis

- Transition invariant
- System invariant
- Inductive invariant

# Transition Invariant

$K=(S,R,I)$

Definition

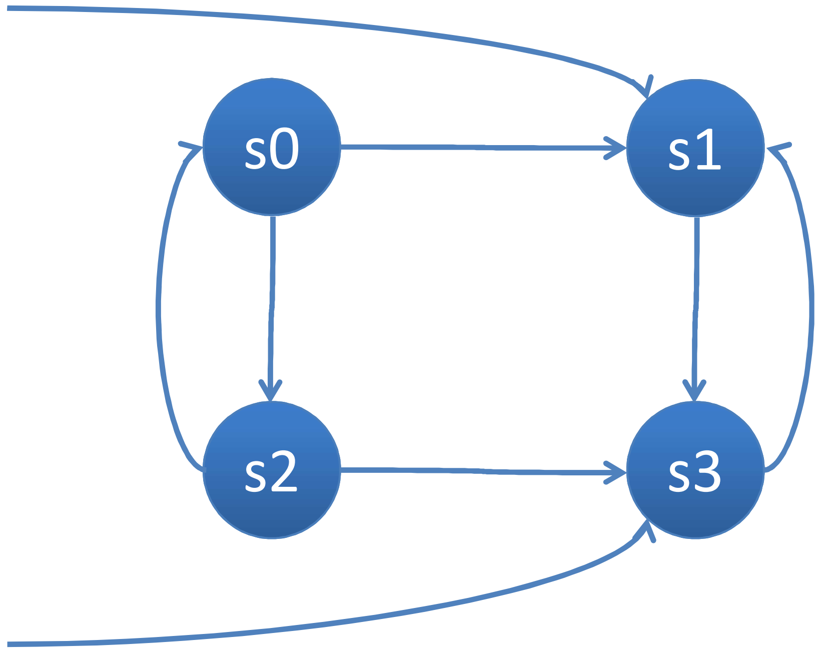
X is a transition invariant, if

for every $s \in X$, if $s \to s'$, then $s' \in X$.

Lemma

If $R(X) \subseteq X$, then X is a transition invariant.

# Example: {s1,s2,s3}×, {s0,s2,s3}×, {s1,s3}√



# System Invariant

Definition

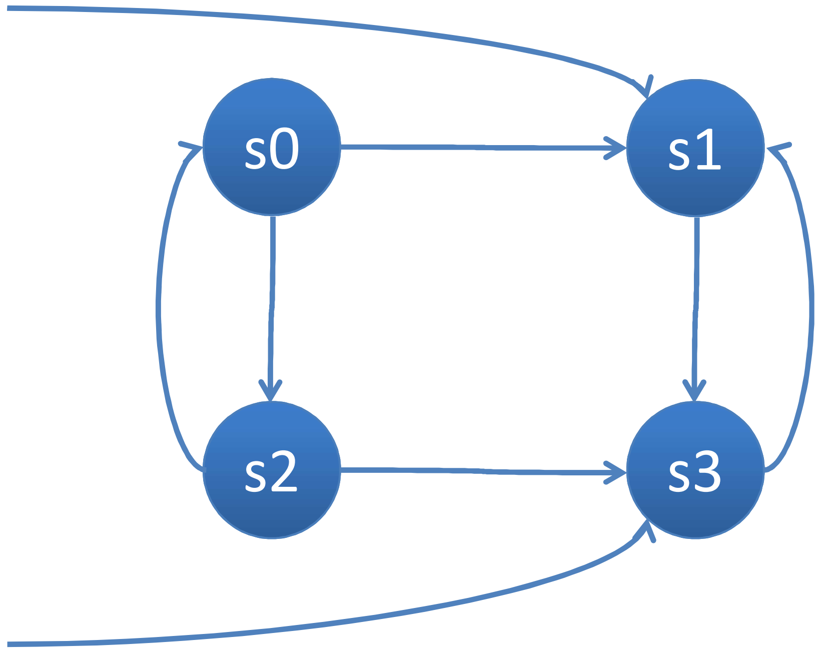
X is a system invariant, if

$I \subseteq X$ and $X \cap Rh(K)$ is a transition invariant.

Lemma

X is a system invariant iff X is safety property.

Example: {s1,s2,s3}√, {s0,s2,s3}×, {s1,s3}√



# Inductive Invariant

Definition

X is an inductive invariant, if

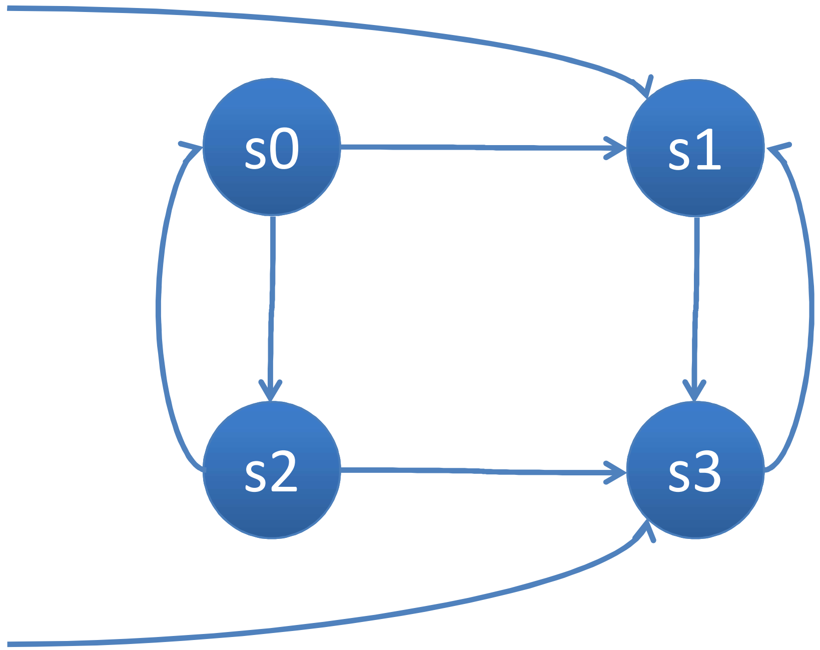
$I \subseteq X$ and X is a transition invariant.

Lemma

X is an inductive invariant iff

X is a system invariant and a transition invariant.

# Example: {s1,s2,s3}×, {s0,s2,s3}×, {s1,s3}√



# Comparison

Rh(K) is an inductive Invariant；

an inductive invariant is a system invariant；

X is a system invariant iff X is a safety property.

# Proof of Safety

Given X.

Question: is X a safety property of K?

# Proof of Safety (1)

Given X.

Question: is X a safety property of K?

Lemma

If X is an inductive invariant,

then K is safe w.r.t. X.

# Proof of Safety (1)

Given X.

Question: is X a safety property of K?

$I \subseteq X$

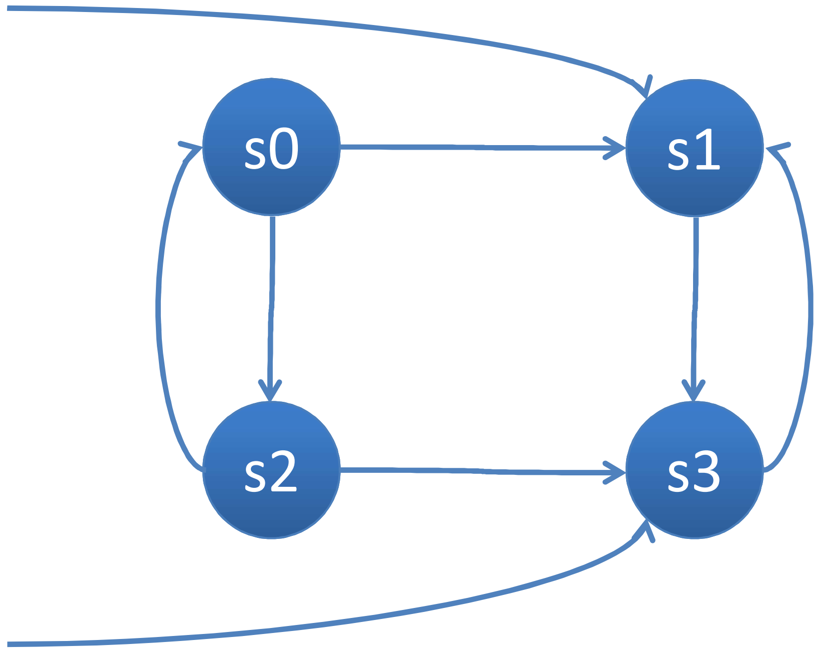
$R(X) \subseteq X$

--------------------------------

X is a safety property

(not always applicable)

# Example: {s1,s3}, {s1,s2,s3}



# Proof of Safety (2)

Given X.

Question: is X a safety property of K?

Lemma

If there is an inductive invariant Y such that $Y \subseteq X$,

then K is safe w.r.t. X.

Completeness

If the conclusion holds, then such a Y exists.

# Proof of Safety (2)

Given X.

Question: is X a safety property of K?

$I \subseteq Y$

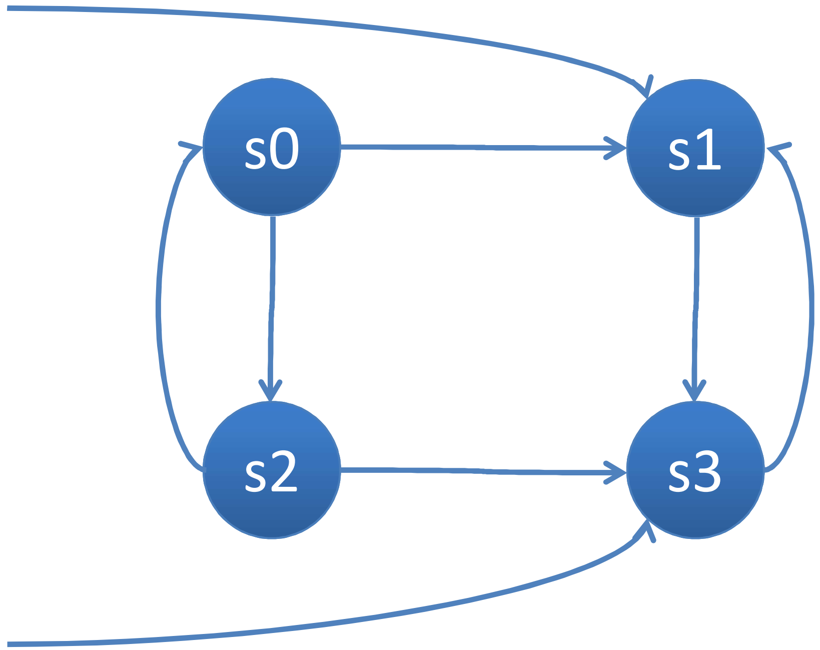
$R(Y) \subseteq Y$

$Y \subseteq X$

---------------------------------

X is a safety property

# Example: {s1,s3}, {s1,s2,s3}



# Safety Analysis – An Example

Contents

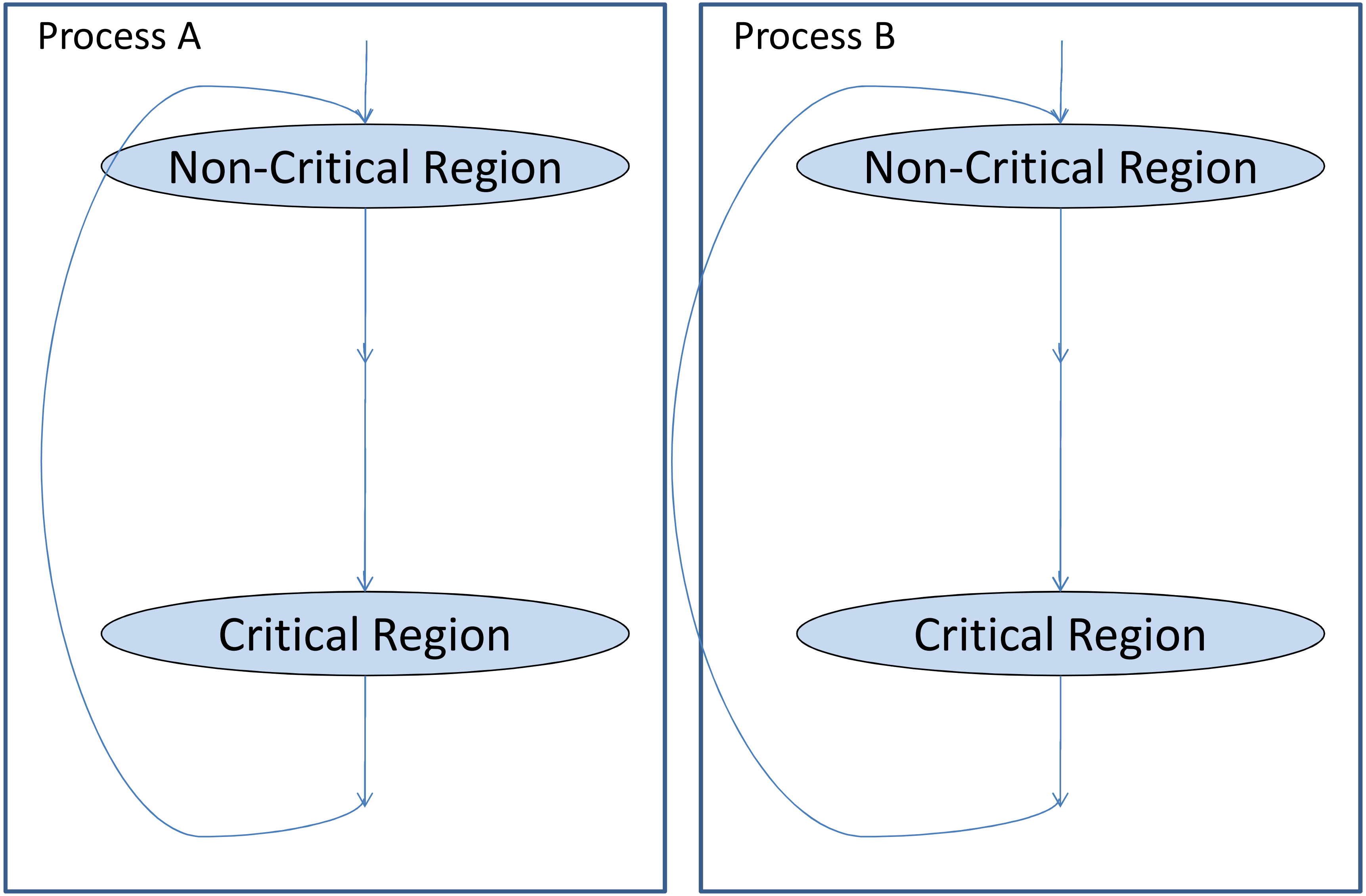
- Mutual Exclusion
- Kripke Structure
- Safety Analysis
- Summary

# Safety Analysis – An Example

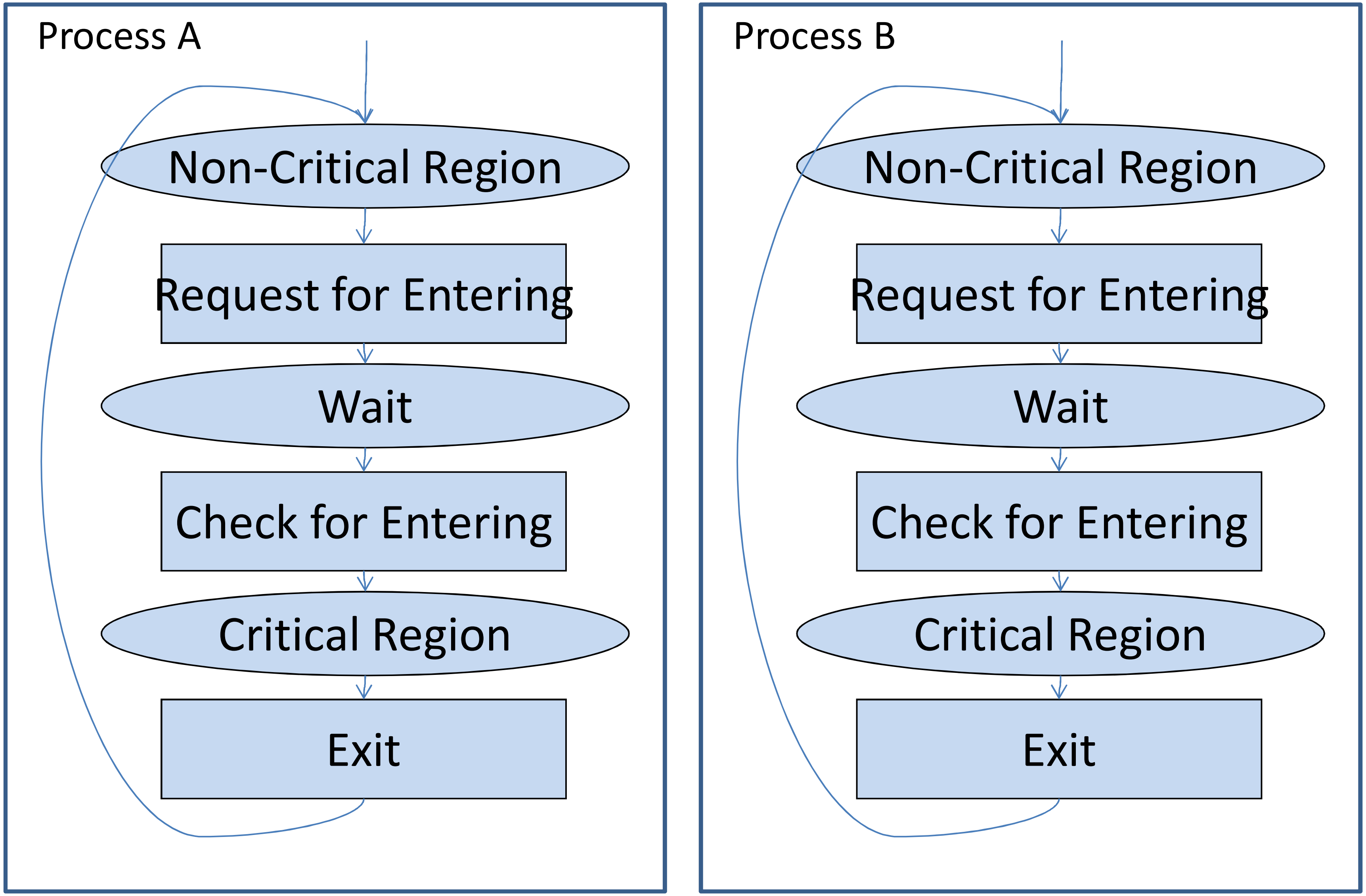
## Contents

- Mutual Exclusion
- Kripke Structure
- Safety Analysis
- Summary

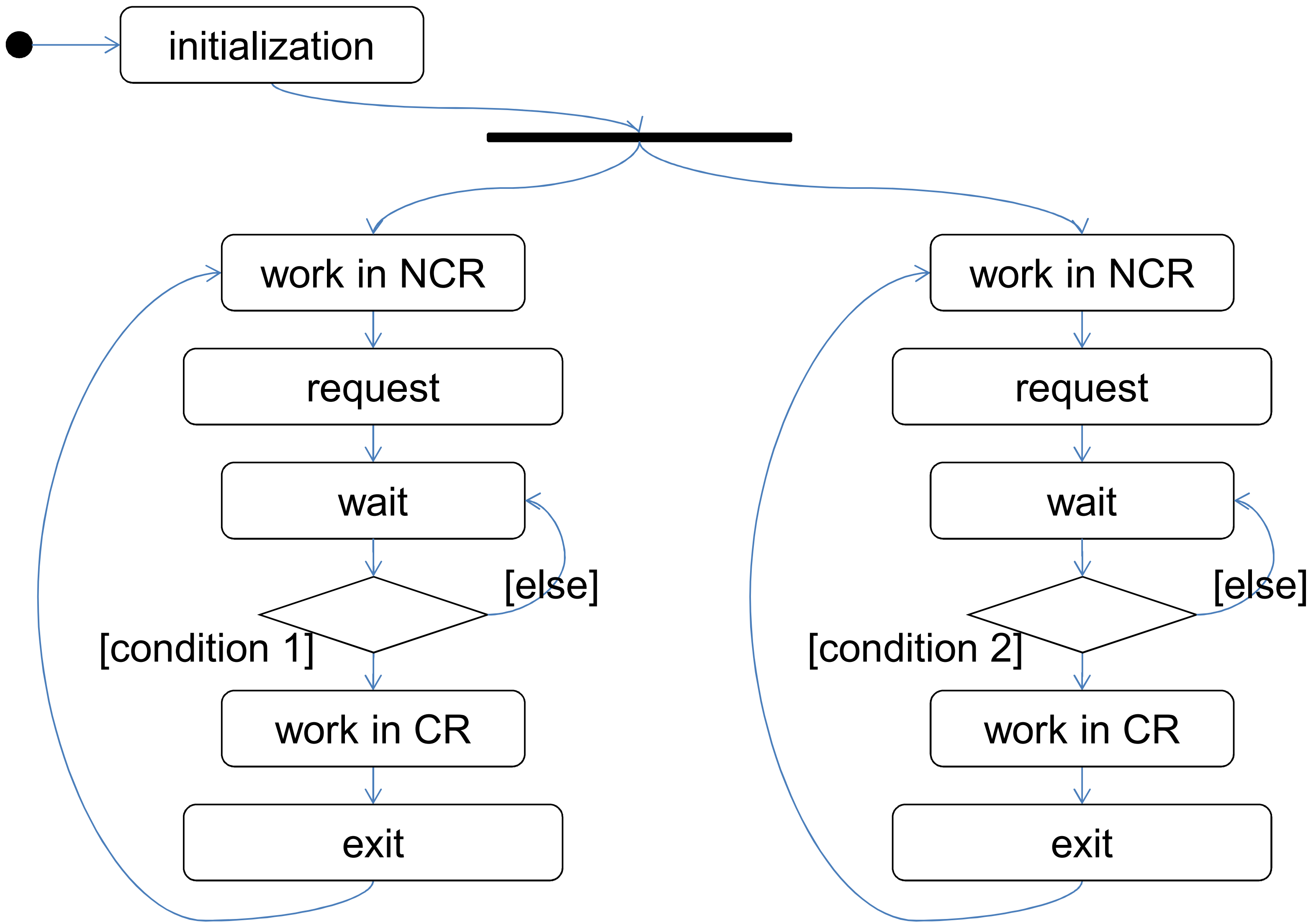
# Example: Mutual Exclusion



# Example: Mutual Exclusion



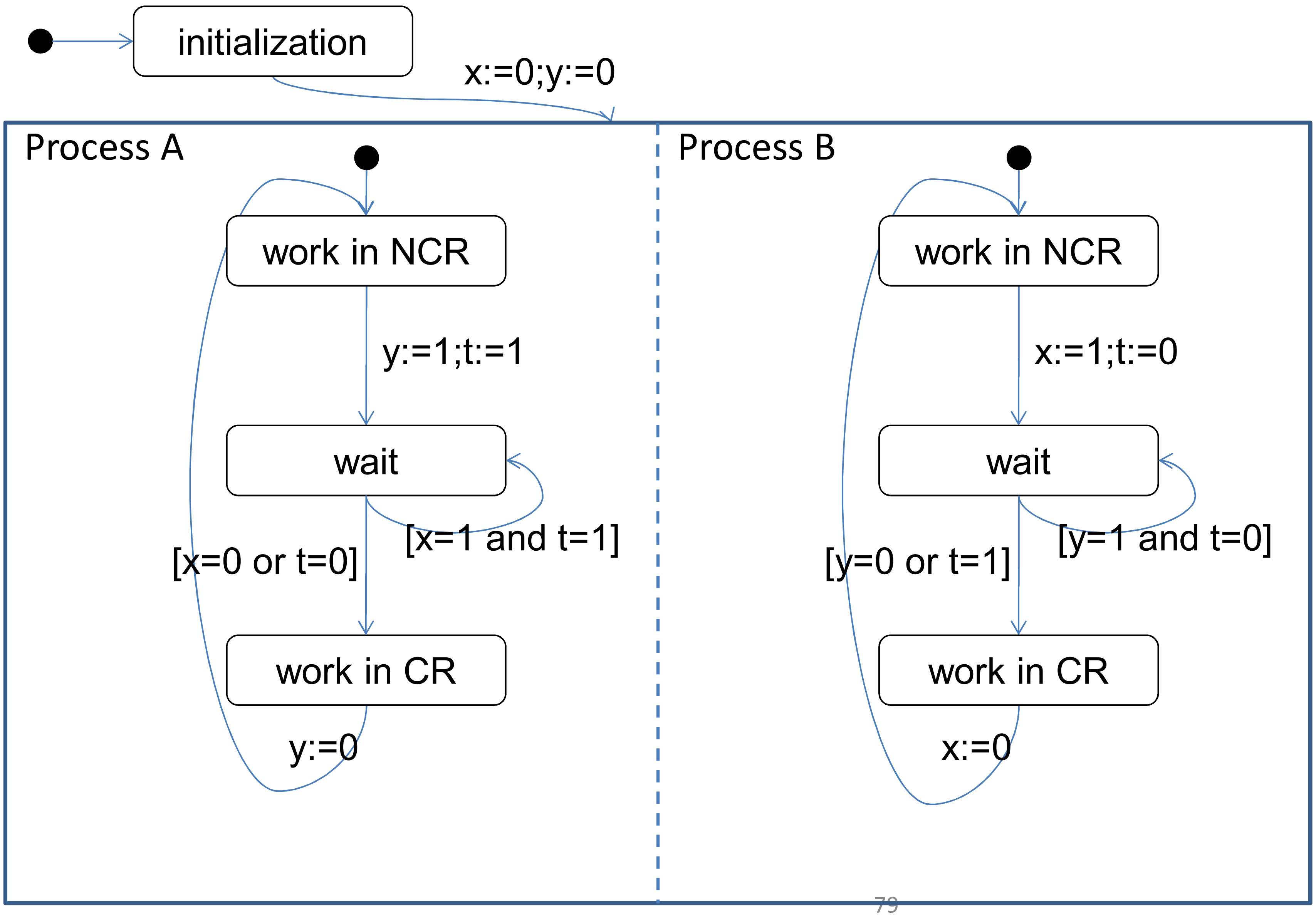
# Design of Mutual Exclusion (Activity)



# Design of Mutual Exclusion

- Purpose:
  - ensure that not both processes are working in the critical region (CR)
- Mechanism:
  - use shared variables
  - y=1: the first process is applying for entering CR or it is in CR
  - x=1: the second process is applying for entering CR or it is in CR
  - t=(i-1): the i-th process has priority for entering CR

# Design of Mutual Exclusion (State)



# Correctness of the Design

- How do we know that the design is correct?

# Combined States of the Two Processes

| Process A | Process B | x,y,t | Remark |
|---|---|---|---|
| NCR | NCR | | |
| NCR | wait | | |
| NCR | CR | | |
| wait | NCR | | |
| wait | wait | | |
| wait | CR | | |
| CR | NCR | | |
| CR | wait | | |
| CR | CR | | Bad state |

# Correctness of the Design

- How do we know that the design is correct?
  - We have to be sure that the bad state is not reachable in all possible executions of the algorithm
  - We may use state exploration (model checking) techniques or deductive proof methods

# Safety Analysis – An Example

## Contents

- Mutual Exclusion
- Kripke Structure
- Safety Analysis
- Summary

# Kripke Structures

A Kripke structure is a triple K=<S,R,I>

- S : A finite set of states
- R $\subseteq$ S x S : A total transition relation
- I $\subseteq$ S : A set of initial states

# Domains of Process and Variable States

| Process A | Process B | x | y | t |
|---|---|---|---|---|
| NCR | NCR | 1 | 1 | 1 |
| wait | wait | 0 | 0 | 0 |
| CR | CR | | | |

(a,b,x,y,t)

# The Set of States: S

$\{(a,b,x,y,t) \mid a,b \in \{NCR, wait, CR\} \text{ and } x,y,t \in \{0,1\}\}$

# Transition Relation: R

| | |
|---|---|
| (NCR,b,x,y,t) | → (wait,b,x,1,1) |
| (wait,b,0,y,t) | → (CR,b,0,y,t) |
| (wait,b,x,y,0) | → (CR,b,x,y,0) |
| (wait,b,1,y,1) | → (wait,b,1,y,1) |
| (CR,b,x,y,t) | → (NCR,b,x,0,t) |
| (a,NCR,x,y,t) | → (a,wait,1,y,0) |
| (a,wait,x,0,t) | → (a,CR,x,0,t) |
| (a,wait,x,y,1) | → (a,CR,x,y,1) |
| (a,wait,x,1,0) | → (a,wait,x,1,0) |
| (a,CR,x,y,t) | → (a,NCR,0,y,t) |

# The Set of Initial States: I

{ (NCR,NCR,0,0,0), (NCR,NCR,0,0,1) }

# Safe States

(NCR,b,x,y,t)

(wait,b,x,y,t)

(CR,NCR,x,y,t)

(CR,wait,x,y,t)

The set of unsafe states: (CR,CR,x,y,t)

# Safety Analysis – An Example

Contents

- Mutual Exclusion
- Kripke Structure
- Safety Analysis
- Summary

# Safety Analysis

Let K=<S,R,I> be a Kripke structure, and A $\subseteq$S.

```
BOOL SafetyAnalysis(K,A)
 {   w:=I;
     repeat until w={};
          s:=w.getElement(); if (s not in A) return false;
               visited[s]:=true;
               for each (s' in R(s)),
                      if (visited[s']=false) w.putElement(s');
               w.removeElement(s);
      return true;
   }
```

# Safety Analysis

Let K=<S,R,I> be a Kripke structure, and Y $\subseteq$S.

**Proposition:**

SafetyAnalysis(K,Y) =true

Iff

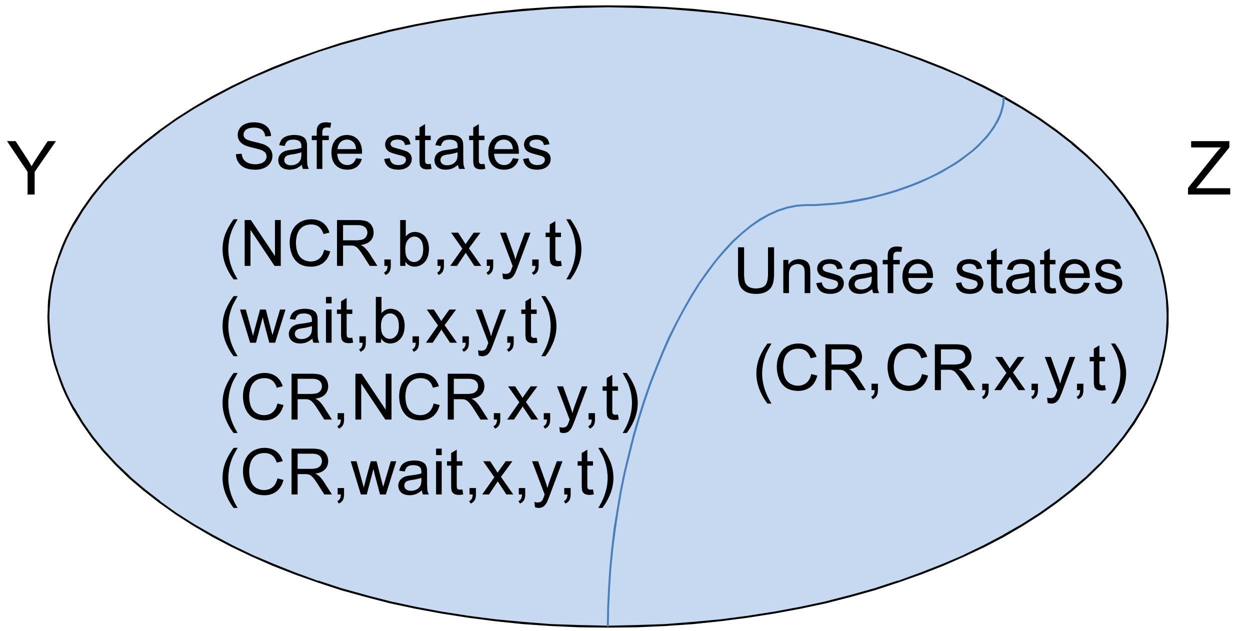
Y is a safety property

Iff

Every reachable state of K is in Y

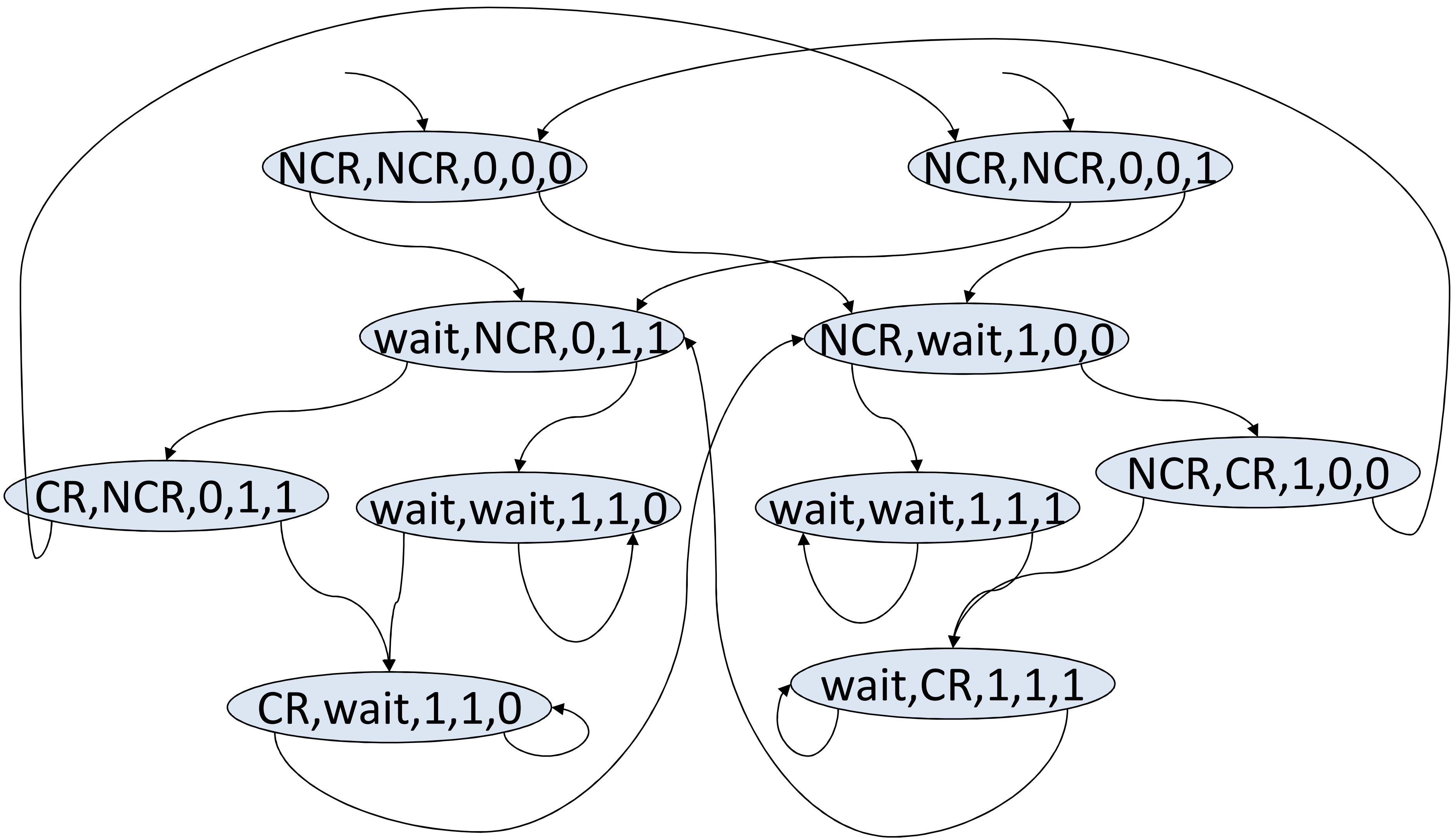
(K is safe with respect to Y)

# Safety Analysis

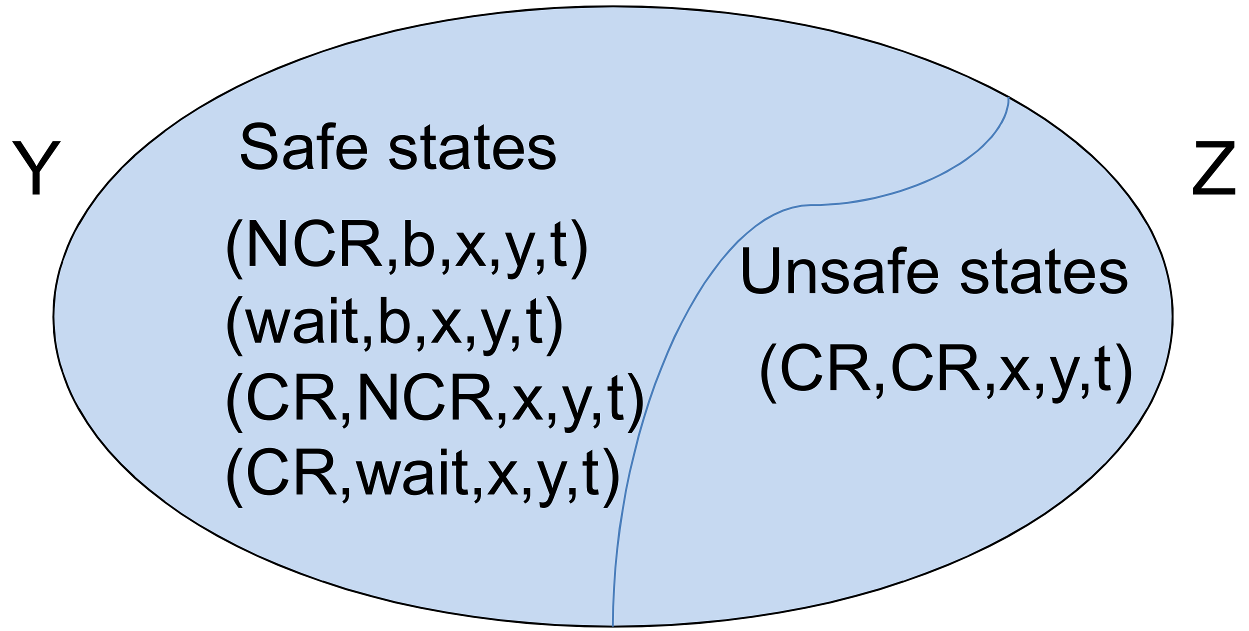



SafetyAnalysis(K,Y)=?

# Safety Analysis



# Safety Analysis




SafetyAnalysis(K,Y)=true $\Leftrightarrow$
Every reachable state is a Y-state (safe state) $\Leftrightarrow$
K is safe w.r.t. Y

# Deductive Proof of the Safety Property

(NCR,b,x,y,t)

(wait,b,x,y,t)

(CR,NCR,x,y,t)

(CR,wait,x,y,t)

Is this set (Y) a tranition invariant?

No, e.g., (CR,wait,x,y,1)→ (CR,CR,x,y,1)

# Inductive Set (X)

| | |
|---|---|
| (NCR,NCR,0,0,t) | $I \subseteq X$ |
| (NCR,wait,1,0,t) | $s \in X \Rightarrow ((s \rightarrow s') \Rightarrow s' \in X)$ |
| (wait,NCR,0,1,t) | $X \subseteq Y$ |
| (NCR,CR,1,0,t) | ------------------------------- |
| (CR,NCR,0,1,t) | Y is a safety property |
| (wait,wait,1,1,t) | |
| (CR,wait,1,1,0) | |
| (wait,CR,1,1,1) | |

# Safety Analysis – An Example

Contents

- Mutual Exclusion
- Kripke Structure
- Safety Analysis
- Summary

# Correctness of the Design

- How do we know that the design is correct?
  - We have to be sure that the bad state is not reachable in all possible executions of the algorithm
  - We may use state exploration (model checking) techniques, or deductive proof methods
  - We have shown that the bad states are not reachable

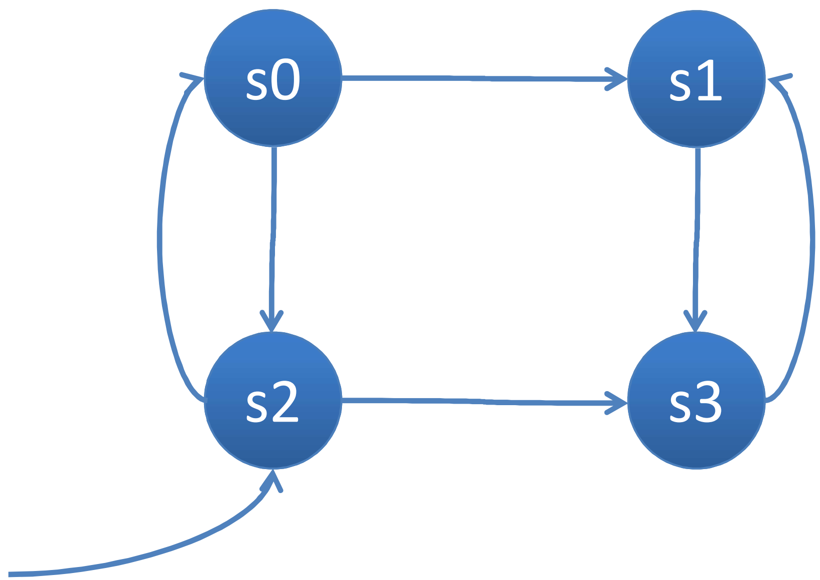
# (III) Avoidability Property

Let Y be a set of states.

Y is avoidable, if

there a computation that does not reach any Y-state.

# Example: {s1,s3}√, {s0}√, {s0,s1} x



# Avoidability Problem

Given a set A$\subseteq$S.

Is A an avoidability property?

# Avoidability Problem

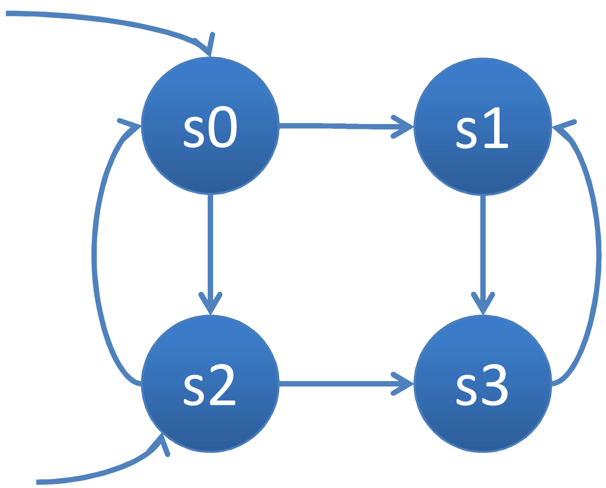
Lemma

For a finite state system <S,R,I>:

A is avoidable

iff

there is a $(\neg A)$-computation



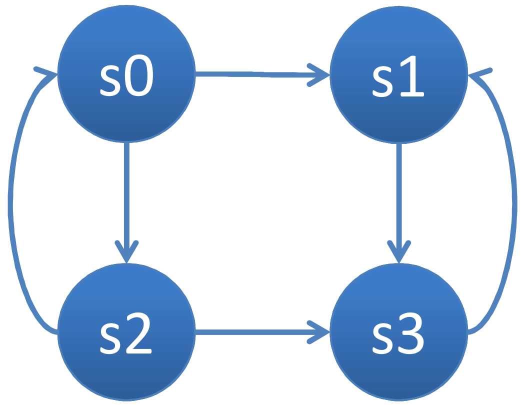

Example:

A = { s1,s3}

$(\neg A)$-computation: $(s0s2)^{\omega}$

# Directed Graphs, G=(V,E)

- Strongly connected graphs
- Subgraphs
- Strongly connected subgraphs
- Derived subgraphs
- Strongly connected components
- Non-trivial Strongly connected components



# Avoidability Problem

Let $R|Y = R \cap (Y \times Y)$

A is avoidable

iff

there is a $(\neg A)$-computation

iff

there is a $(\neg A)$-path starting from I that reaches a non-trivial strongly connected component of $< \neg A, R|\neg A >$.

# Avoidability Problem

For a finite state system $K=\langle S,R,I\rangle$:

Let $K|X = \langle X, R|X, I \cap X\rangle$

Let $K' = \langle S', R', I'\rangle = K|\neg A$

A is avoidable, iff
there is a reachable non-trivial SCC of $(S',R')$ in $K'$ .

SCC

# Tarjan Algorithm (scctarjan)

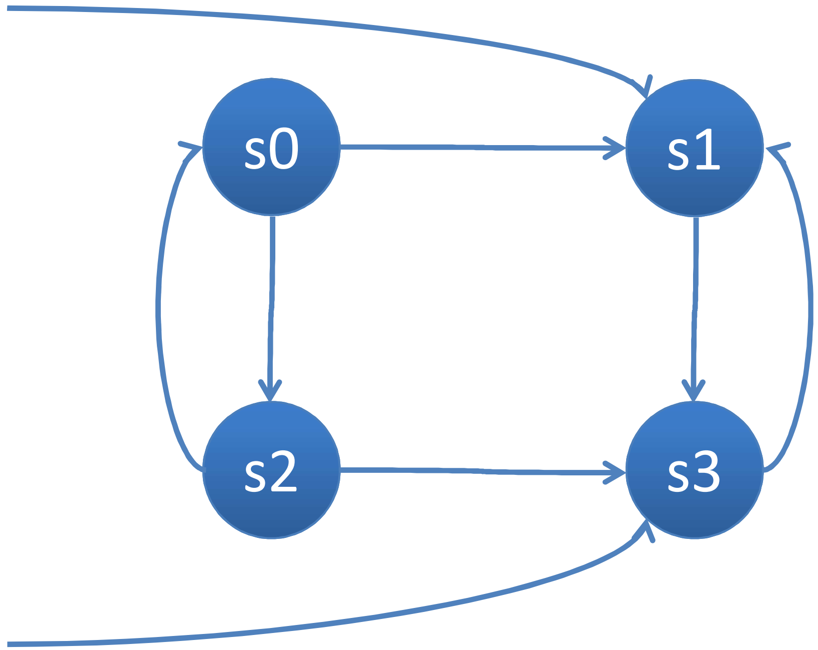
**algorithm** scctarjan； **input:** *G* = (*V*, *E*)； **output:** A set of SCCs;

    *index* := 0; *S* := empty;
    **for each** *v* **in** *V* **do if** (*v*.index is undefined) strongconnect(*v*) **endif endfor**

**function** strongconnect； **input:** *v*； **output:** A set of SCCs;

    *v*.index := *v*.lowlink := *index; index := index +1*; *S*.push(*v*);
    **for each** (*v*, *w*) **in** *E* **do**
        **if** (*w*.index is undefined) **then**
            strongconnect(*w*);  *v*.lowlink := min(*v*.lowlink, *w*.lowlink)
        **else if** (*w* is in *S*) **then**
            *v*.lowlink := min(*v*.lowlink, *w*.index)
        **endif**
    **endfor**
    **if** (*v*.lowlink = *v*.index) **then** // SCCok(v);
        currentSCC := empty;
        **repeat** *w* := *S*.pop(); add *w* to currentSCC; **until** (*w* = *v*);
        output currentSCC
    **endif**

# Example:




{s0,s2}, {s1,s3}

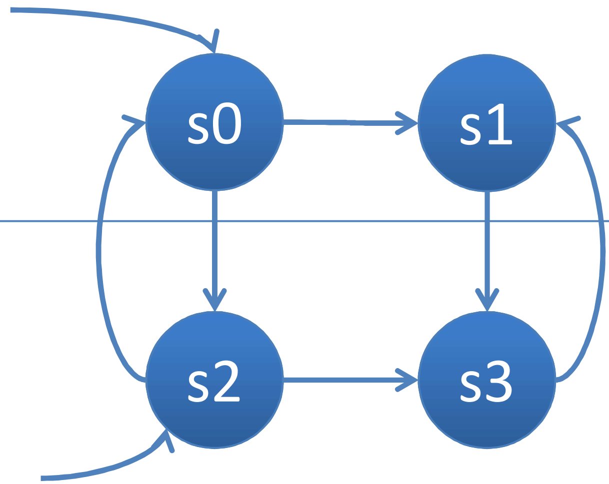
# Correctness and Complexity

Given G=(V,E).

The output of scctarjan(G) is a set of SCCs
that is a partition of G.

The complexity of scctarjan(G) is O(|V|+|E|).

# Avoidability Analysis




Let K=<S,R,I> and A $\subseteq$S.

```
BOOL AvoidabilityAnalysis(K,A)
{     K':=(S',R',I'):= K|¬A;
      G:=(S',R');
      scclist:=scctarjan(G);
      w:={};
      for each (e in scclist) if (nontrivial(e)) w:=w∪e;
      return ReachbilityAnalysis(K',w);
}
```

# Avoidability Analysis

Let $K=<S,R,I>$ be a Kripke structure, and $A \subseteq S$.

**Proposition:**

AvoidabilityAnalysis(K,A) =true

iff

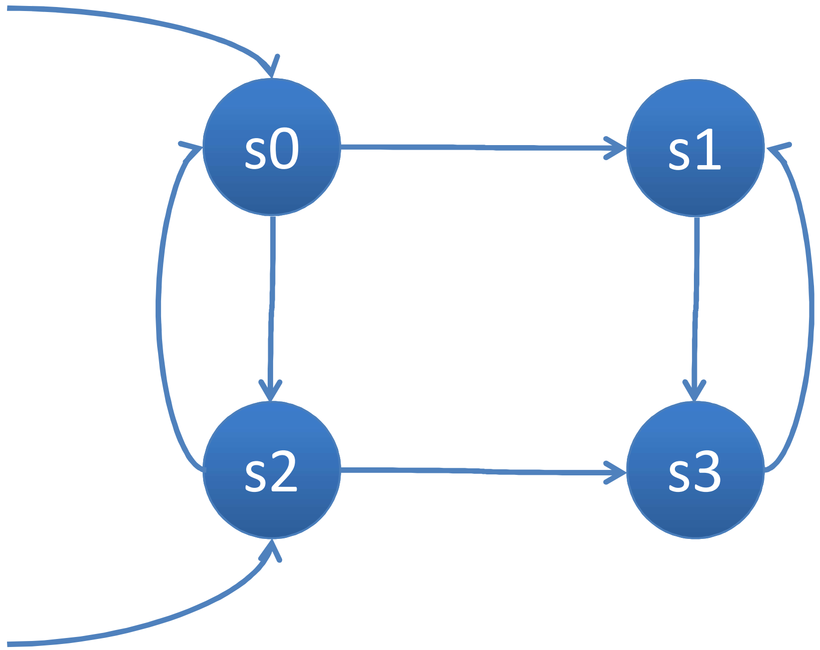
A is an avoidability property

# Inevitability

Given a set Y.

Y is an inevitability property, if
every computation reaches a Y-state

# Example: {s1,s3}, {s2,s3}



# Avoidability & Inevitability

Inevitability is a negation of avoidability.

**Proposition**
Y is an inevitability property of K, iff
Y is not an avoidability property.

# Inevitability Analysis

Let K=<S,R,I> be a Kripke structure, and A $\subseteq$S.

```
BOOL InevitabilityAnalysis(K,A)
{    K':=(S',R',I'):=K|¬A;
     G:=(S',R');
     scclist:=scctarjan(G);
     w:={};
     for each (e in scclist) if (nontrivial(e)) w:=w ∪ e;
     return (not ReachbilityAnalysis(K',w));
 }
```

# Inevitability Analysis

Let K=<S,R,I> be a Kripke structure, and A $\subseteq$S.

**Proposition:**

InevitabilityAnalysis(K,A) =true

Iff

A is an inevitability property

# Inevitability & Avoidability

Let $K=<S,R,I>$ be a Kripke structure, and $A \subseteq S$.

InevitabilityAnalysis(K,A) =true $\Leftrightarrow$

AvoidabilityAnalysis(K,A) =false

# Deductive Inevitability Analysis

# Proof of Inevitability

If there is a sequence of sets of states:

$X_0$, $X_1$,…,$X_n$ such that

$I \subseteq X_0$,

$R(X_i \setminus Y) \subseteq X_{i+1}$, for i=0,1,…,n-1

$X_n \subseteq Y$,

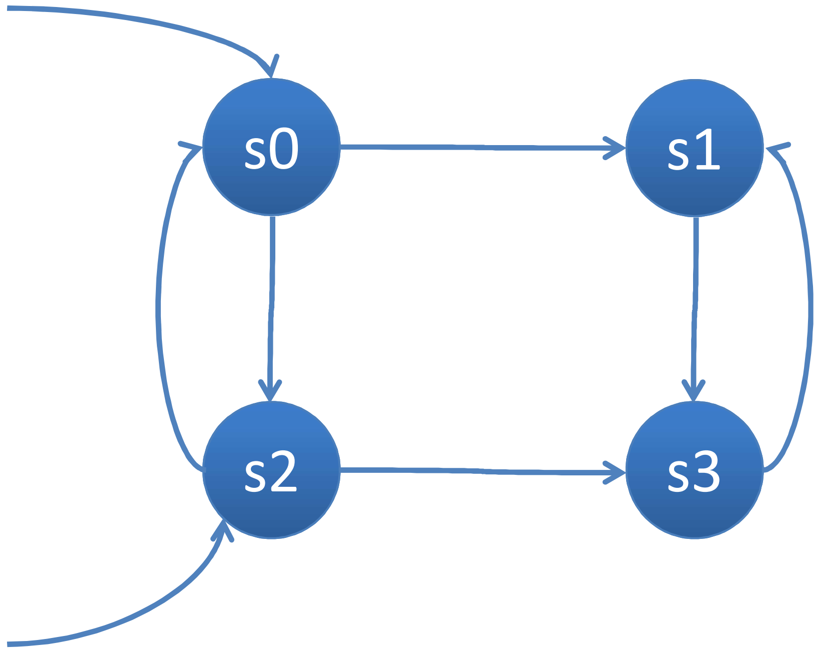
Then Y an inevitability property.

Completeness

For finite state systems,

if the conclusion holds, then such a sequence exists.

# Example: Y={s2,s3}




$X_0$={s0,s2}, $X_1$={s1,s2}, $X_2$={s3}

# Inevitability Analysis – An Example

## Contents

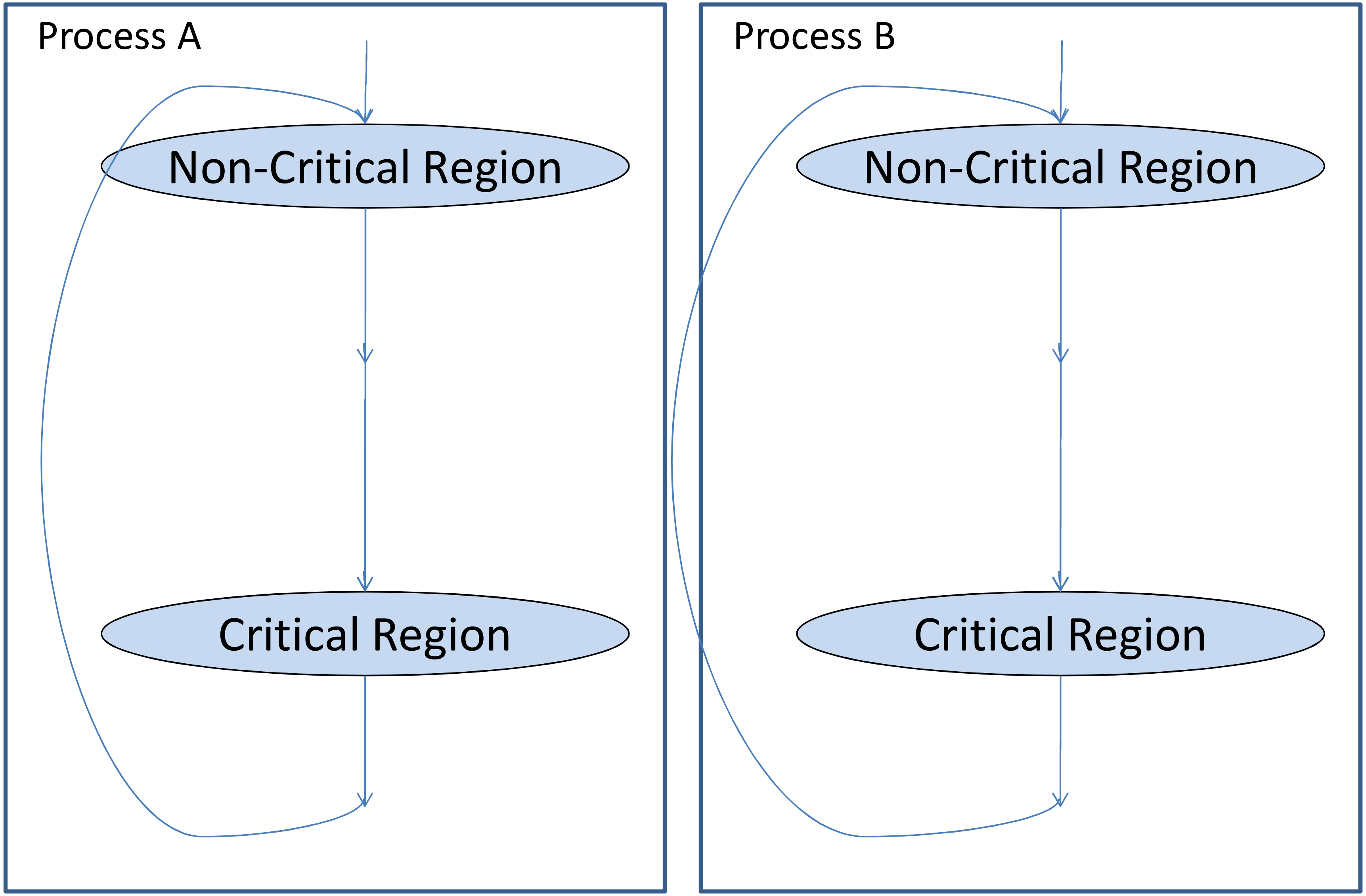
- Mutual Exclusion
- Kripke Structure
- Inevitability Analysis
- Summary

# Inevitability Analysis – An Example

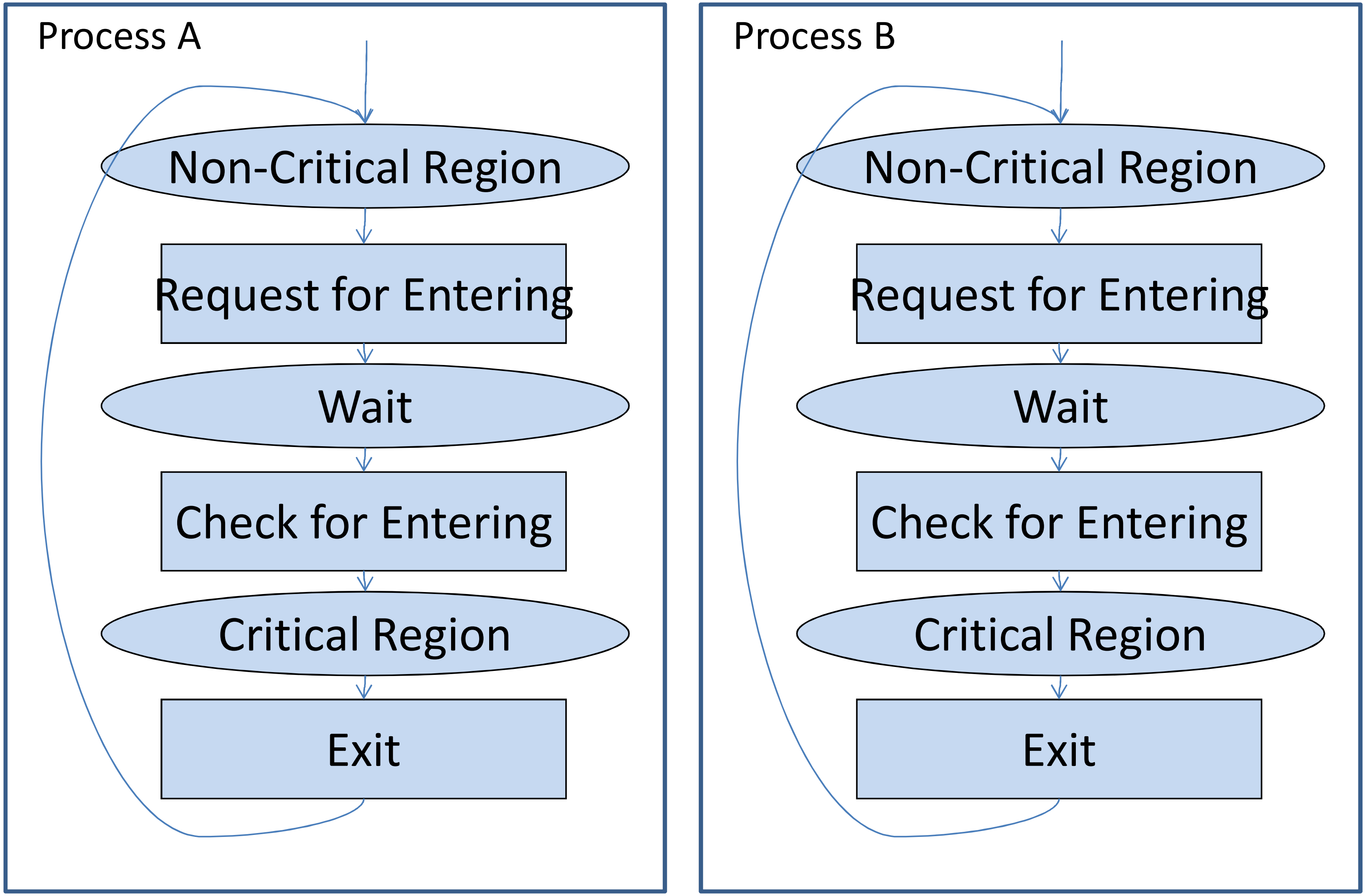
Contents

- Mutual Exclusion
- Kripke Structure
- Inevitability Analysis
- Summary

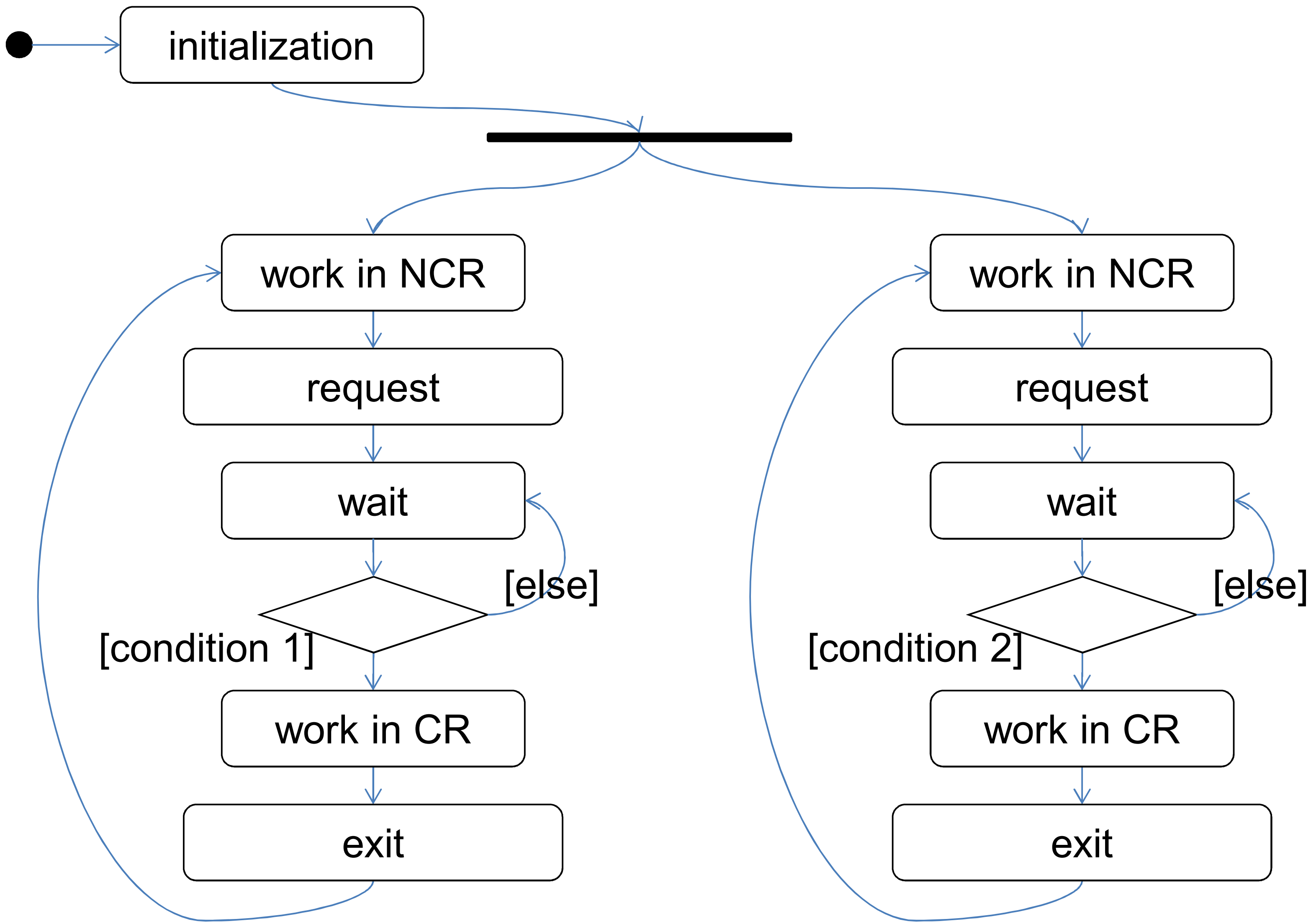
# Example: Mutual Exclusion



# Example: Mutual Exclusion



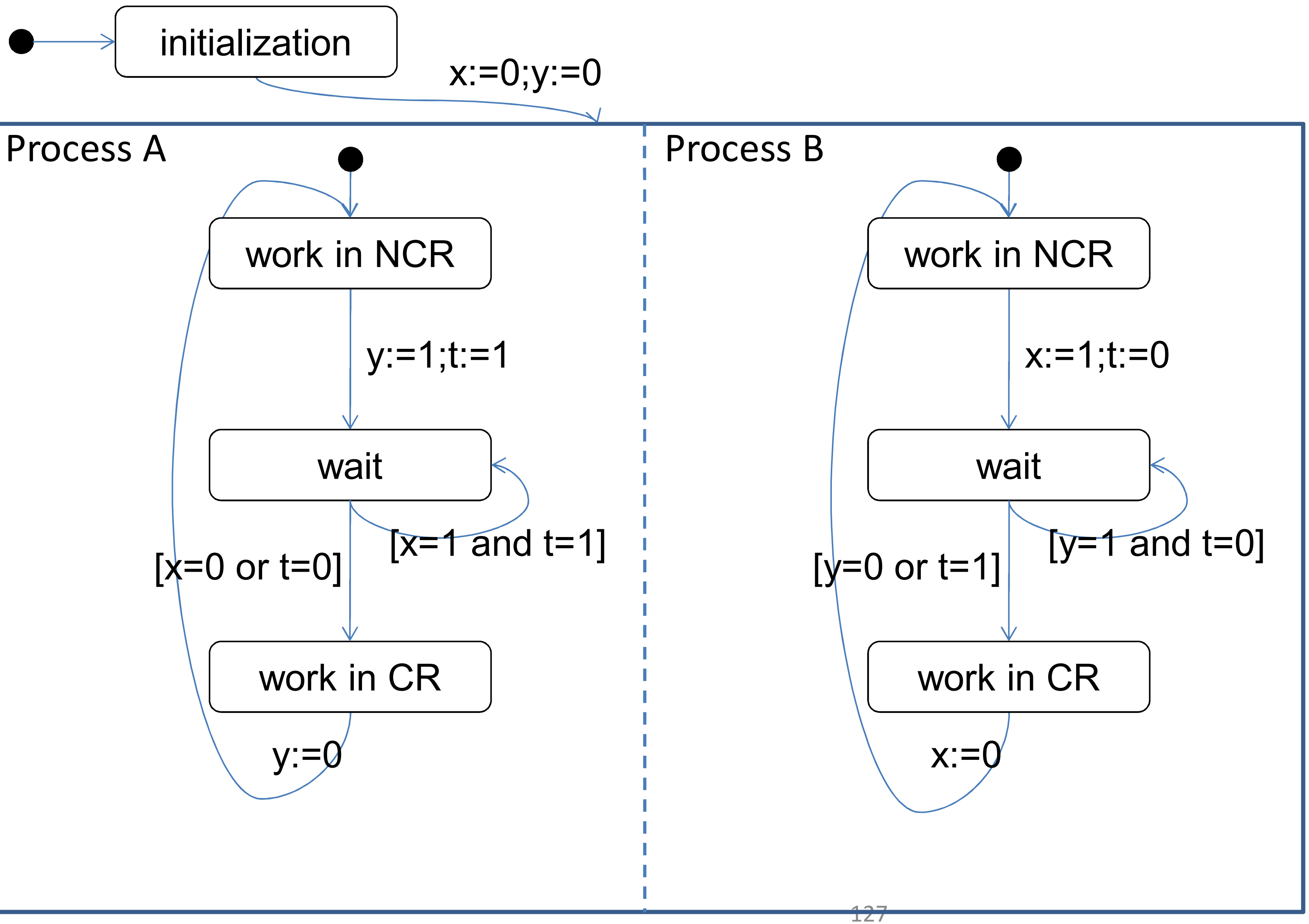
# Design of Mutual Exclusion (Activity)



# Design of Mutual Exclusion

- Purpose:
  - ensure that not both processes are working in the critical region (CR)
- Mechanism:
  - use shared variables
  - $y=1$: the first process is applying for entering CR or it is in CR
  - $x=1$: the second process is applying for entering CR or it is in CR
  - $t=(i-1)$: the i-th process has priority for entering CR

# Design of Mutual Exclusion (State)



# Correctness of the Design

- Ensure that not both processes are working in the critical region (CR)
- Ensure that at least one process reaches CR

# Combined States of the Two Processes

| Process A | Process B | x,y,t | Remark |
|---|---|---|---|
| NCR | NCR | | |
| NCR | wait | | |
| NCR | CR | | Good states |
| wait | NCR | | |
| wait | wait | | |
| wait | CR | | Good states |
| CR | NCR | | Good states |
| CR | wait | | Good states |
| CR | CR | | Good states |

# Correctness of the Design

- How do we know that the design is correct?
  - We have to be sure that the good states are inevitable, i.e., reachable in all possible executions of the algorithm
  - We may use state exploration (model checking) techniques or deductive proof methods

# Inevitability Analysis – An Example

Contents

- Mutual Exclusion
- Kripke Structure
- Inevitability Analysis
- Summary

# Kripke Structures

A Kripke structure is a triple K=<S,R,I>

- S : A finite set of states
- R $\subseteq$ S x S : A total transition relation
- I $\subseteq$ S : A set of initial states

# Domains of Process and Variable States

| Process A | Process B | x | y | t |
|---|---|---|---|---|
| NCR | NCR | 1 | 1 | 1 |
| wait | wait | 0 | 0 | 0 |
| CR | CR | | | |

(a,b,x,y,t)

# The Set of States: S

$\{(a,b,x,y,t) \mid a,b \in \{NCR, wait, CR\} \text{ and } x,y,t \in \{0,1\}\}$

# Transition Relation: R

| | |
|---|---|
| (NCR,b,x,y,t) | → (wait,b,x,1,1) |
| (wait,b,0,y,t) | → (CR,b,0,y,t) |
| (wait,b,x,y,0) | → (CR,b,x,y,0) |
| (wait,b,1,y,1) | → (wait,b,1,y,1) |
| (CR,b,x,y,t) | → (NCR,b,x,0,t) |
| (a,NCR,x,y,t) | → (a,wait,1,y,0) |
| (a,wait,x,0,t) | → (a,CR,x,0,t) |
| (a,wait,x,y,1) | → (a,CR,x,y,1) |
| (a,wait,x,1,0) | → (a,wait,x,1,0) |
| (a,CR,x,y,t) | → (a,NCR,0,y,t) |

# The Set of Initial States: I

{ (NCR,NCR,0,0,0), (NCR,NCR,0,0,1) }

# Good States

(CR,b,x,y,t)

(a,CR,x,y,t)

The set of other states:

(NCR,NCR,x,y,t)

(wait,wait,x,y,t)

(wait,NCR,x,y,t)

(NCR,wait,x,y,t)

# Inevitability Analysis – An Example

Contents

- Mutual Exclusion
- Kripke Structure
- Inevitability Analysis
- Summary

# Inevitability Analysis

Let K=<S,R,I> be a Kripke structure, and $A \subseteq S$.

```
BOOL InevitabilityAnalysis(K,A)
{    K':=(S',R',I'):=K|¬A;
     G:=(S',R');
     scclist:=scctarjan(G);
     w:={};
     for each (e in scclist) if (nontrivial(e)) w:=w ∪ e;
     return (not ReachbilityAnalysis(K',w));
 }
```

# Inevitability Analysis

Let K=<S,R,I> be a Kripke structure, and A $\subseteq$S.

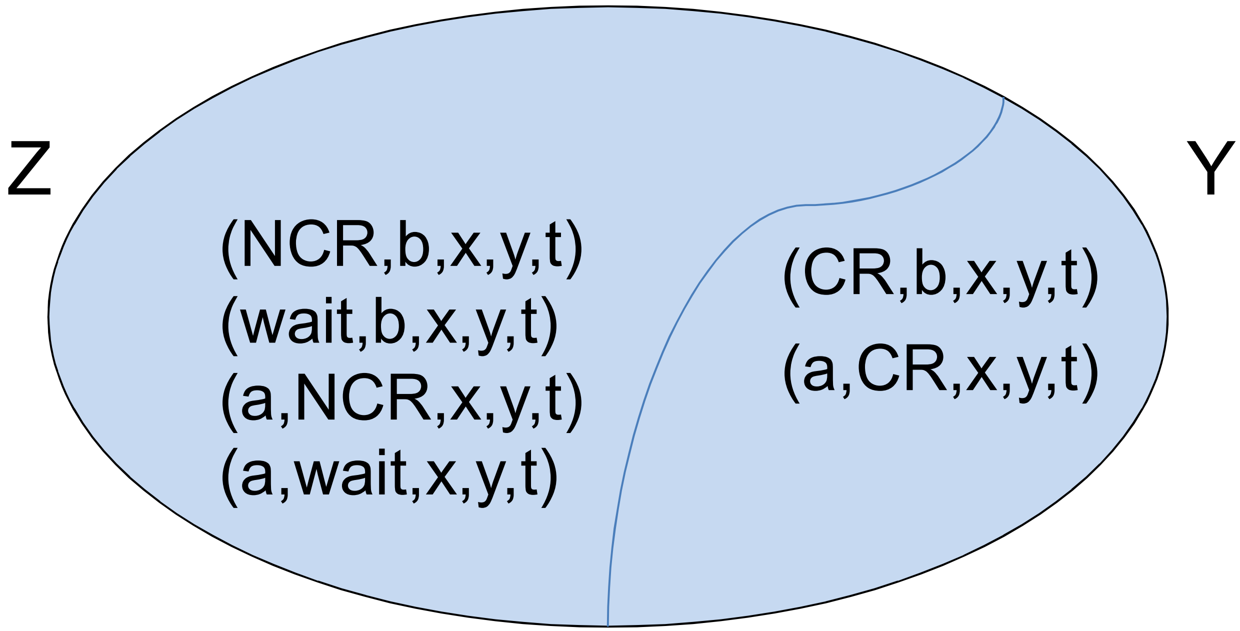
**Proposition:**

InevitabilityAnalysis(K,A) =true

Iff

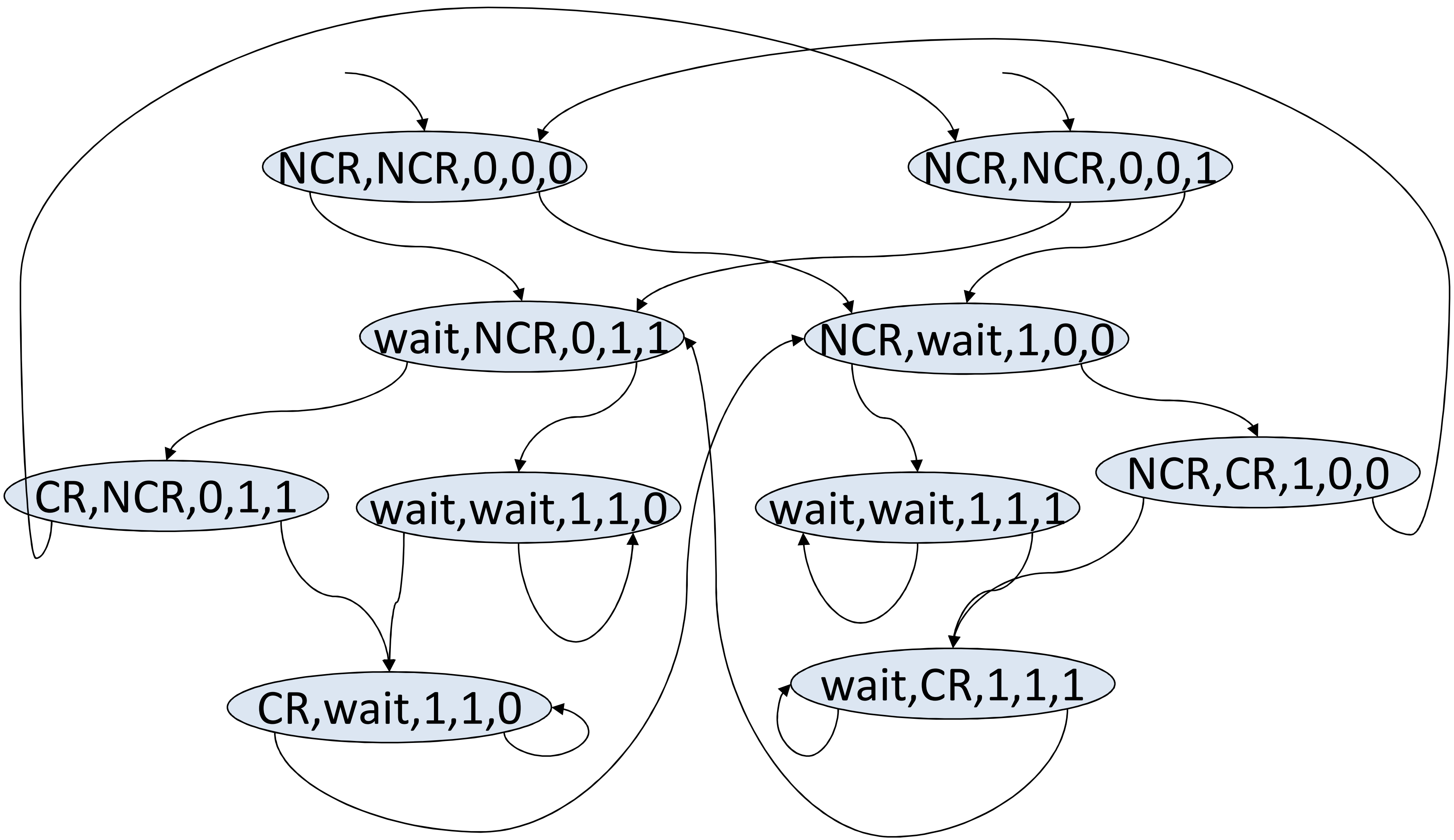
A is an inevitability property

# Inevitability Analysis

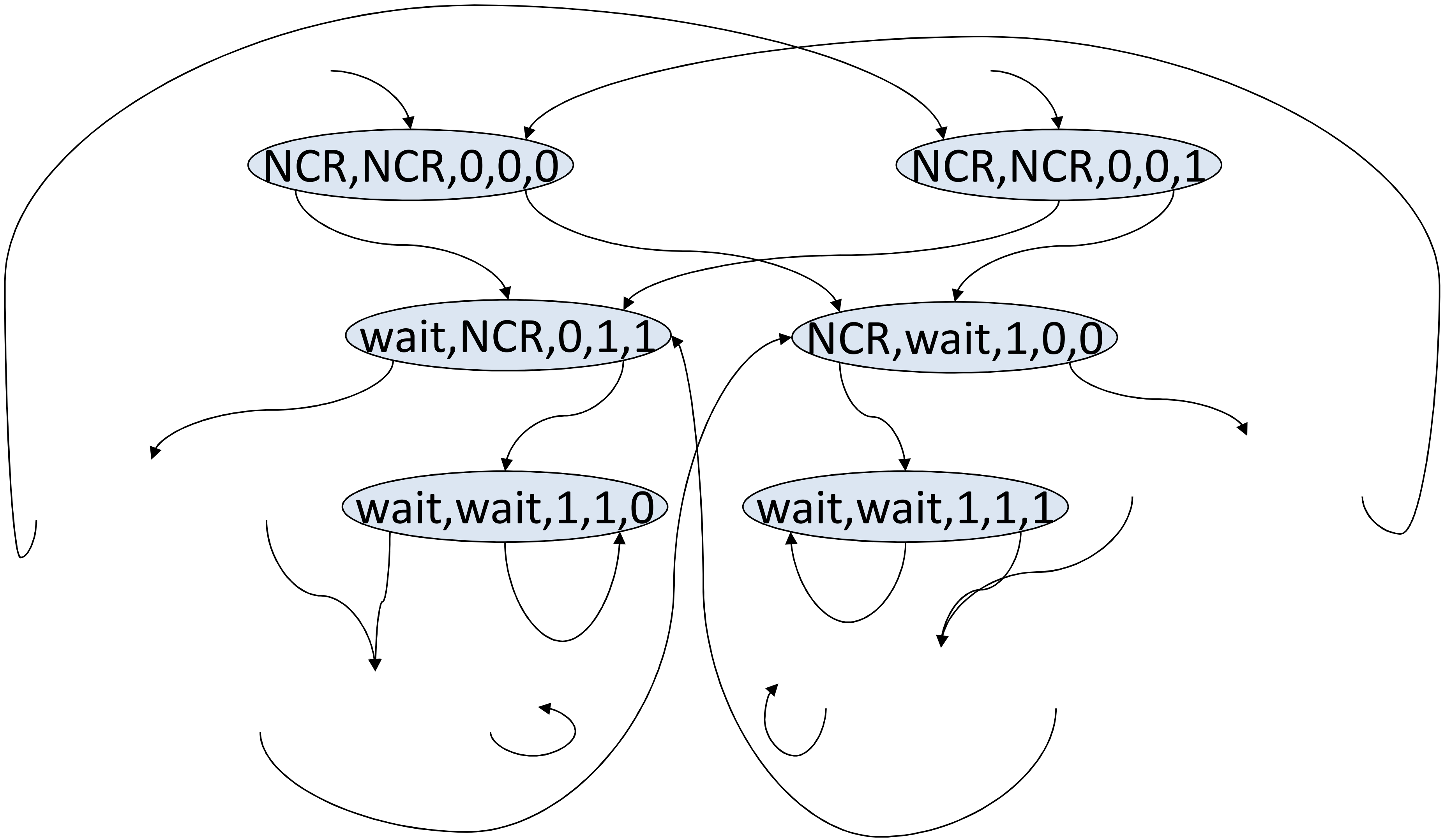



InivitabilityAnalysis(K,Y)=?

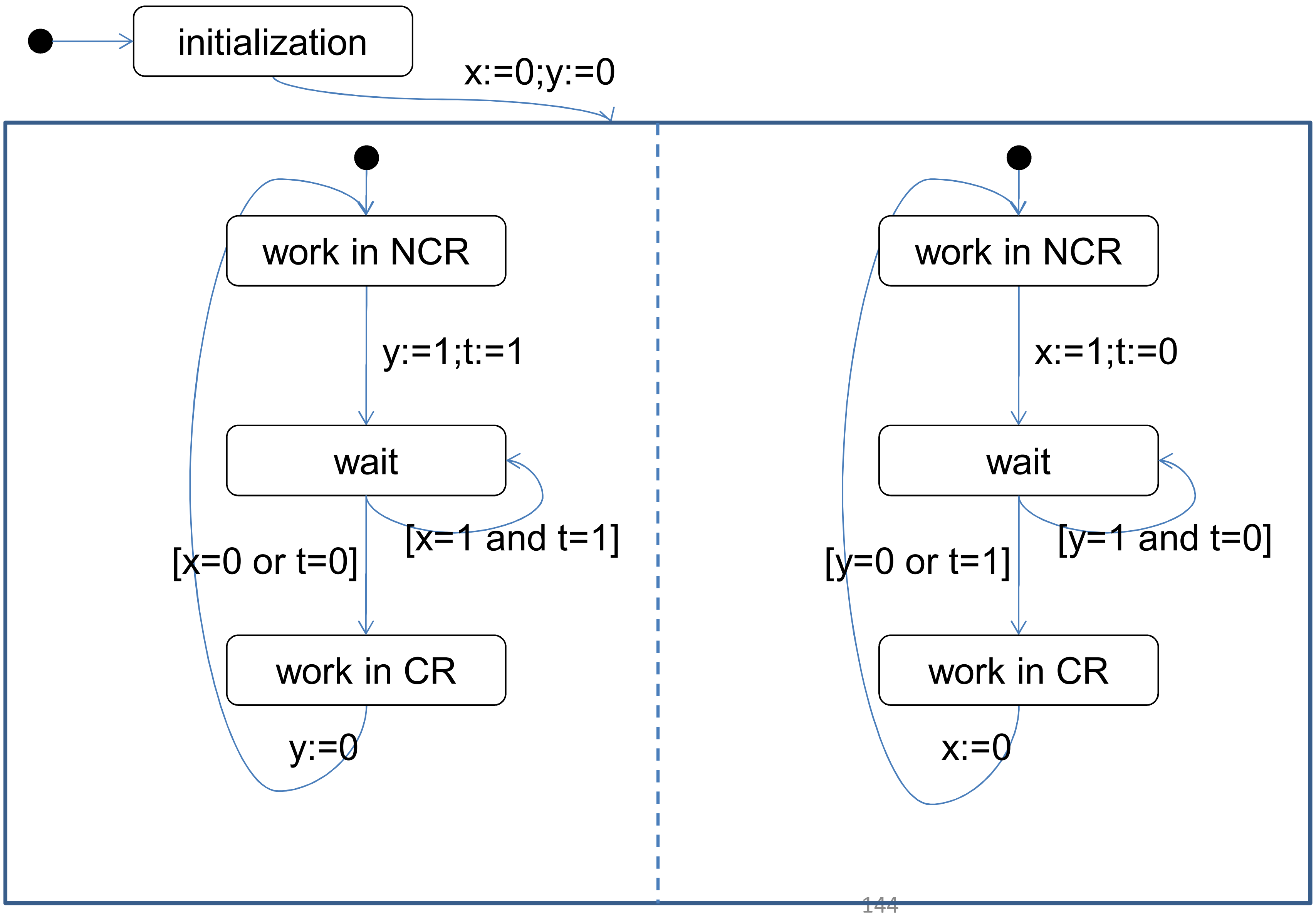
# Inevitability Analysis



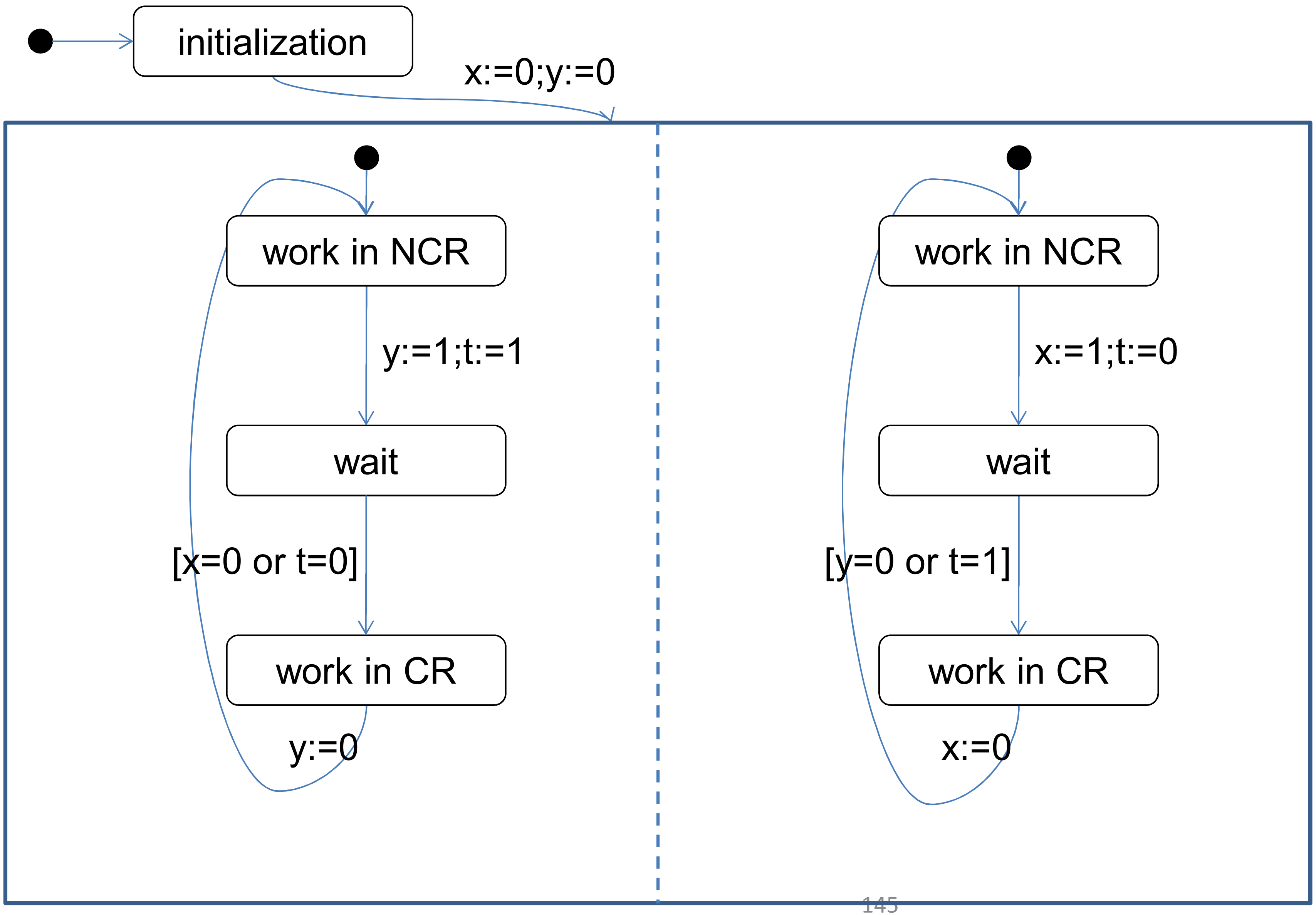
# Inevitability Analysis (Returns False ??)



# Design of Mutual Exclusion (State)



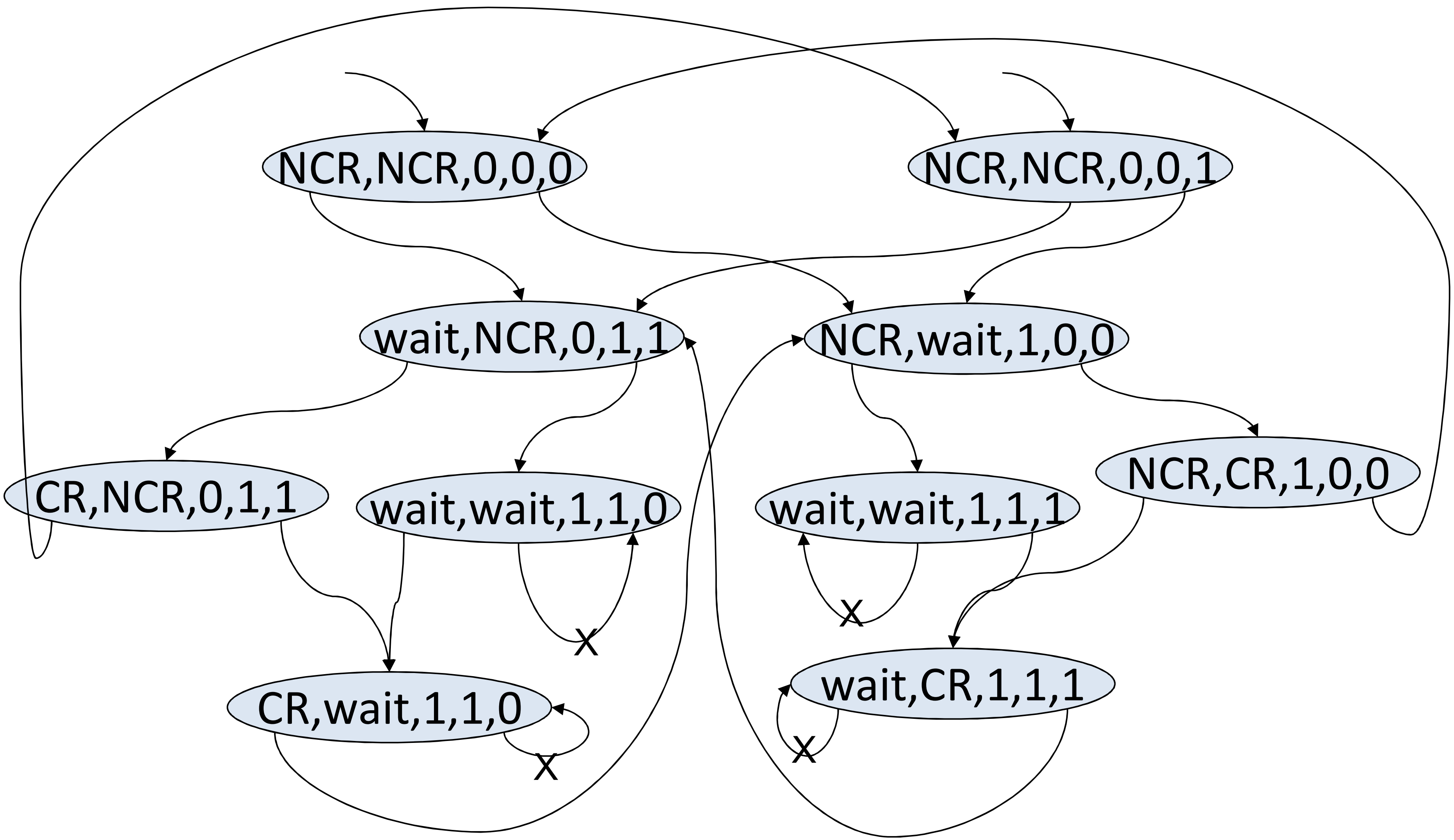
# Revised Model of Mutual Exclusion



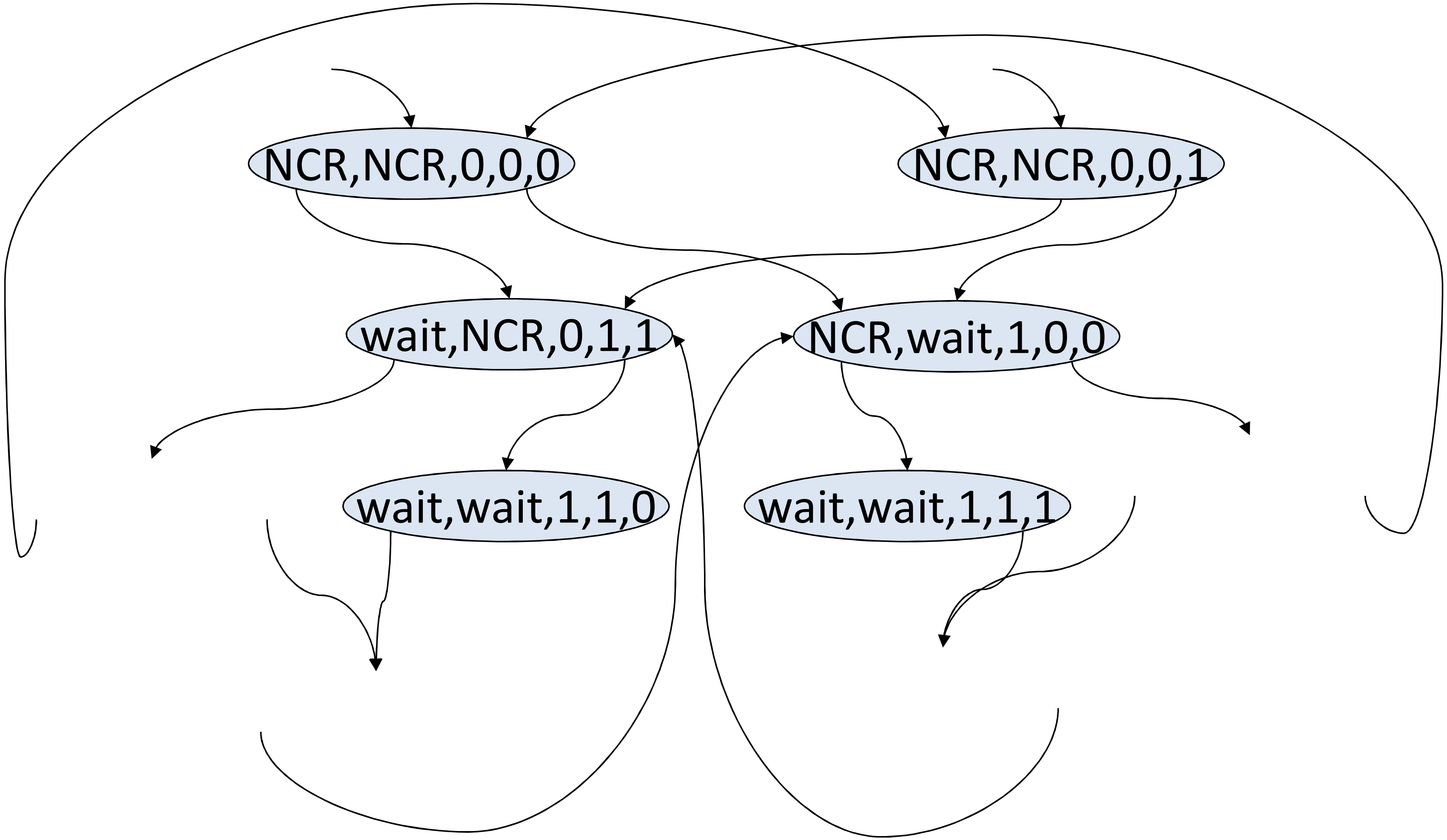
# Transition Relation: R

| | |
|---|---|
| (NCR,b,x,y,t) | → (wait,b,x,1,1) |
| (wait,b,0,y,t) | → (CR,b,0,y,t) |
| (wait,b,x,y,0) | → (CR,b,x,y,0) |
| (wait,b,1,y,1) | → (wait,b,1,y,1) |
| (CR,b,x,y,t) | → (NCR,b,x,0,t) |
| | |
| (a,NCR,x,y,t) | → (a,wait,1,y,0) |
| (a,wait,x,1,t) | → (a,CR,x,1,t) |
| (a,wait,x,y,1) | → (a,CR,x,y,1) |
| (a,wait,x,1,0) | → (a,wait,x,1,0) |
| (a,CR,x,y,t) | → (a,NCR,0,y,t) |

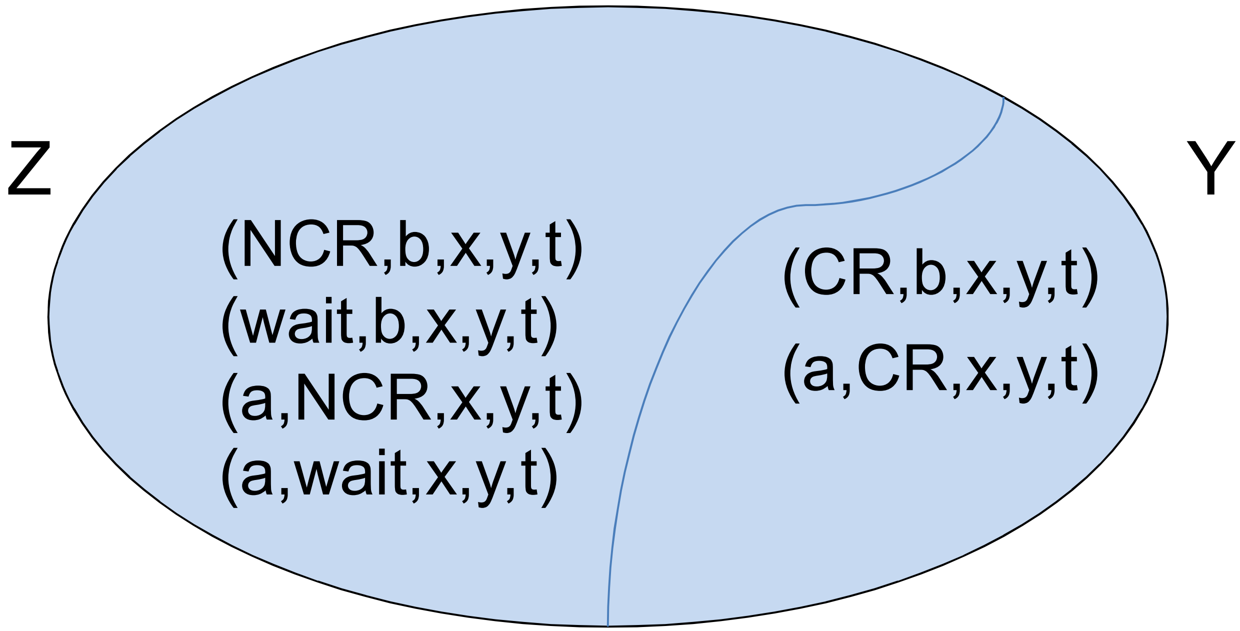
# Inevitability Analysis



# Inevitability Analysis

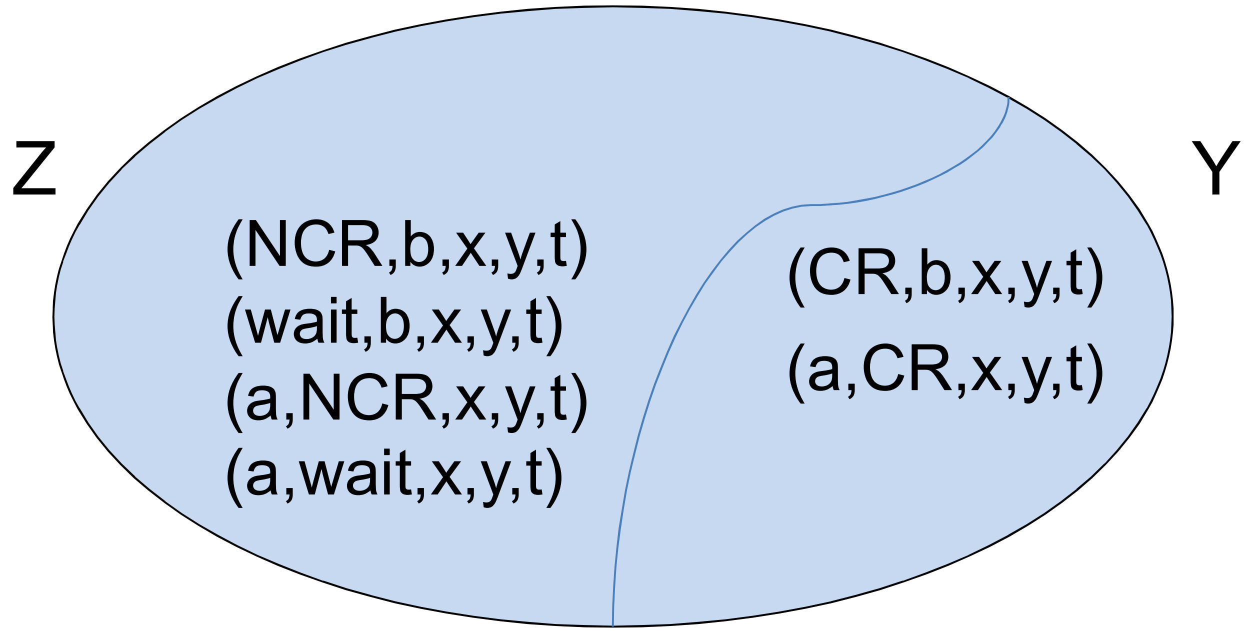


# Inevitability Analysis




InivitabilityAnalysis(K',Y)=?

# Inevitability Analysis




InevitabilityAnalysis(K',Y)=true $\Leftrightarrow$
Every computation reaches a Y-states $\Leftrightarrow$
Y is inevitable in K'

# Deductive Proof of Inevitability

$X_0$ = { (NCR,NCR,0,0,0), (NCR,NCR,0,0,1) }

$X_1$ = { (wait,NCR,0,1,1), (NCR,wait,1,0,0) }

$X_2$ = { (wait,wait,1,0,0), (wait,wait,1,1,1) } $\cup$ Y

$X_3$ = { (CR,wait,1,1,0), (wait,CR,1,1,1) }

We have

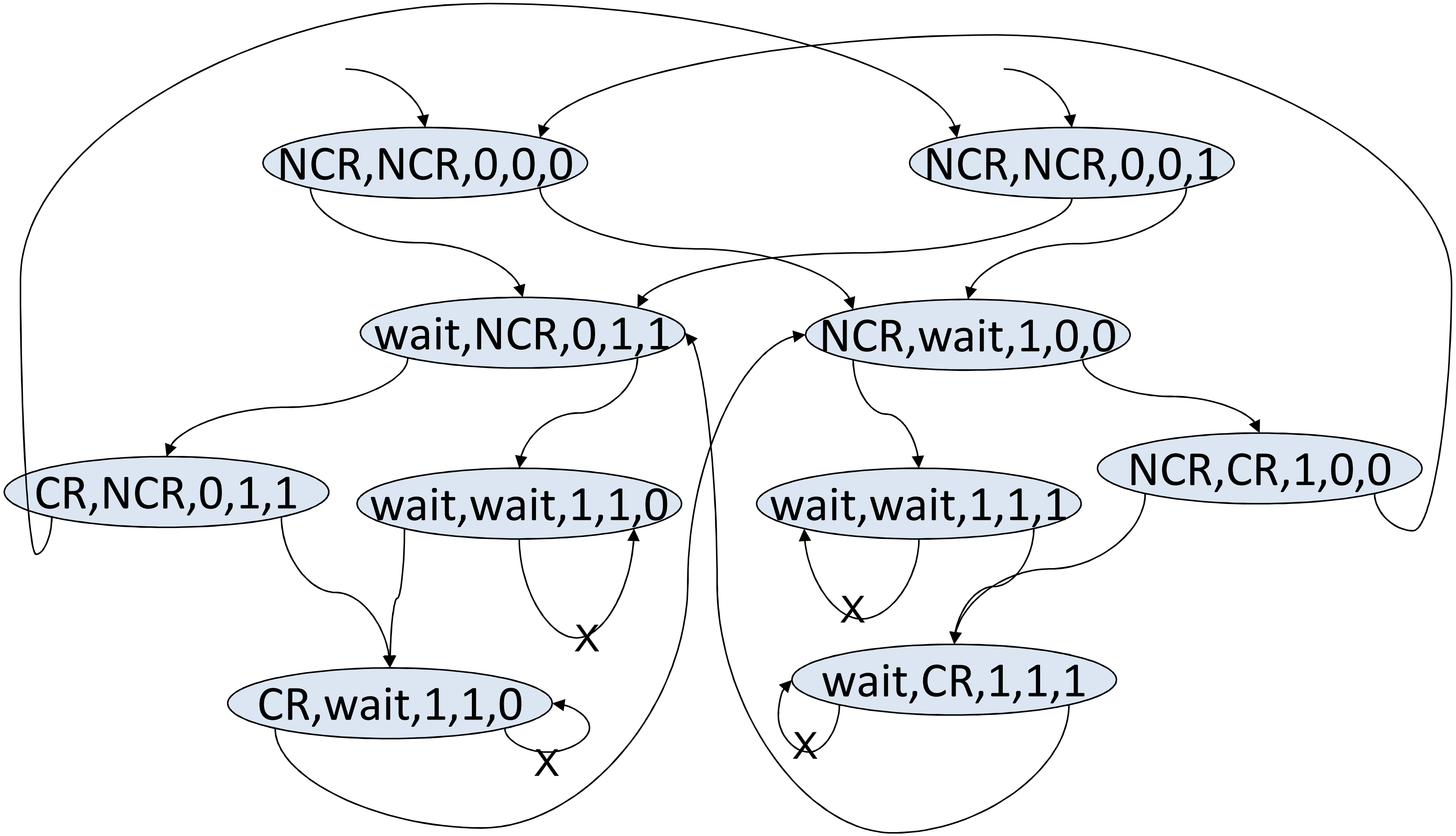
$I \subseteq X_0$,

$R(X_i \setminus Y) \subseteq X_{i+1}$, for i=0,1,2

$X_3 \subseteq Y$,

Therefore Y an inevitability property of K'.

# Deductive Proof of Inevitability



# Inevitability Analysis – An Example

Contents

- Mutual Exclusion
- Kripke Structure
- InevitabilityAnalysis
- Summary

# Correctness of the Design

- How do we know that the design is correct?
  - We have to be sure that the good states are inevitable, i.e., reachable in all possible executions of the algorithm
  - We may use state exploration (model checking) techniques or deductive proof methods
  - We have shown that the good states are inevitable.

# (IV) Summary

- Kripke结构 --- 基本概念
- 安全性质 --- 模型检测算法、推理方法
- 必达性质 --- 模型检测算法、推理方法

| 表达能力问题 | 表达方式问题 | 算法和推理问题 |
| --- | --- | --- |



EE

# 思考：

| 给定一个Kripke结构K=<S,R,I>和集合B,A$\subseteq$S. |
| --- |
| (1)<br>判断以下说法的正确性：<br>A是可避免性质，当且仅当<br>K有一条 由I可达 非A非平凡强连通分量的 非A路径。<br>(2)<br>定义(B,A)路径为满足以下条件的路径：<br>至少一个B状态出现在该路径且同时或之后有A状态出现。<br>设计基于不动点计算的算法以检查K中是否存在<br>初始状态为起点的(B,A)路径。 |