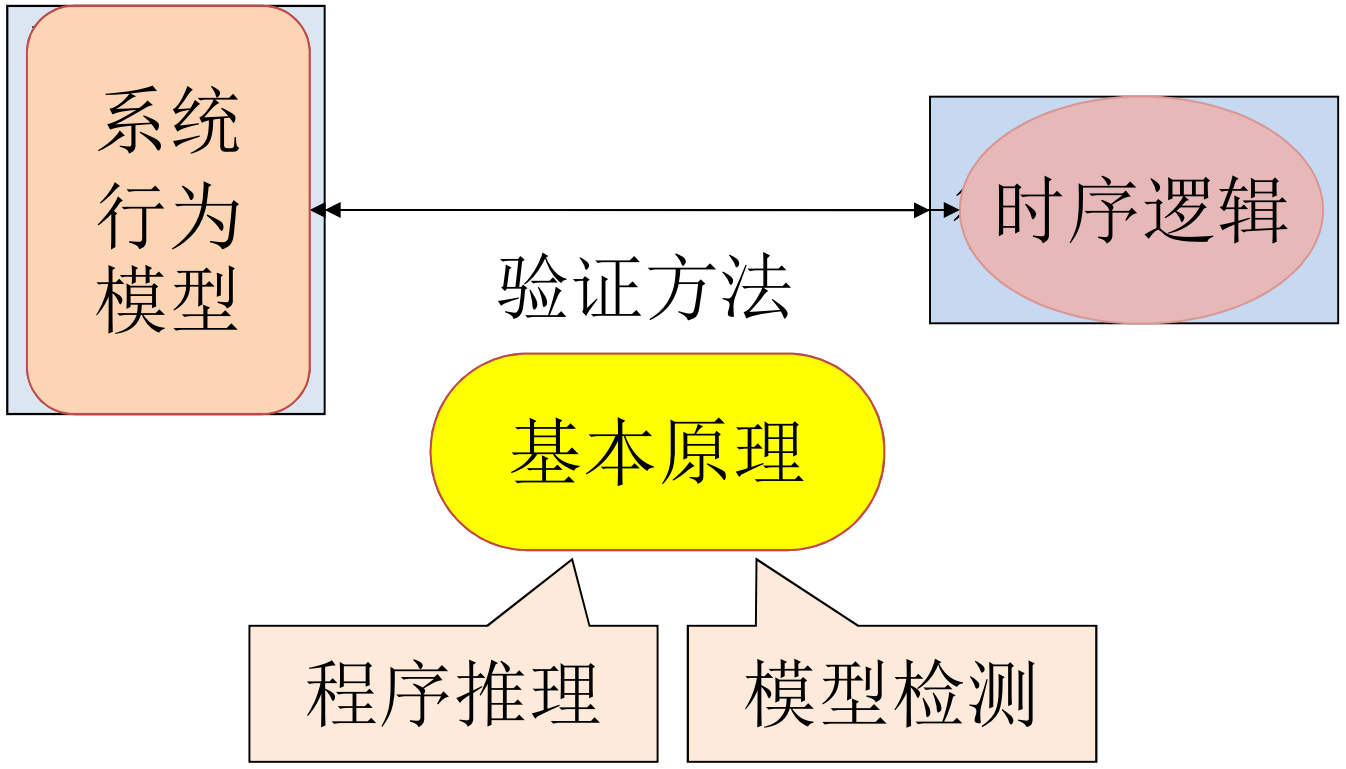
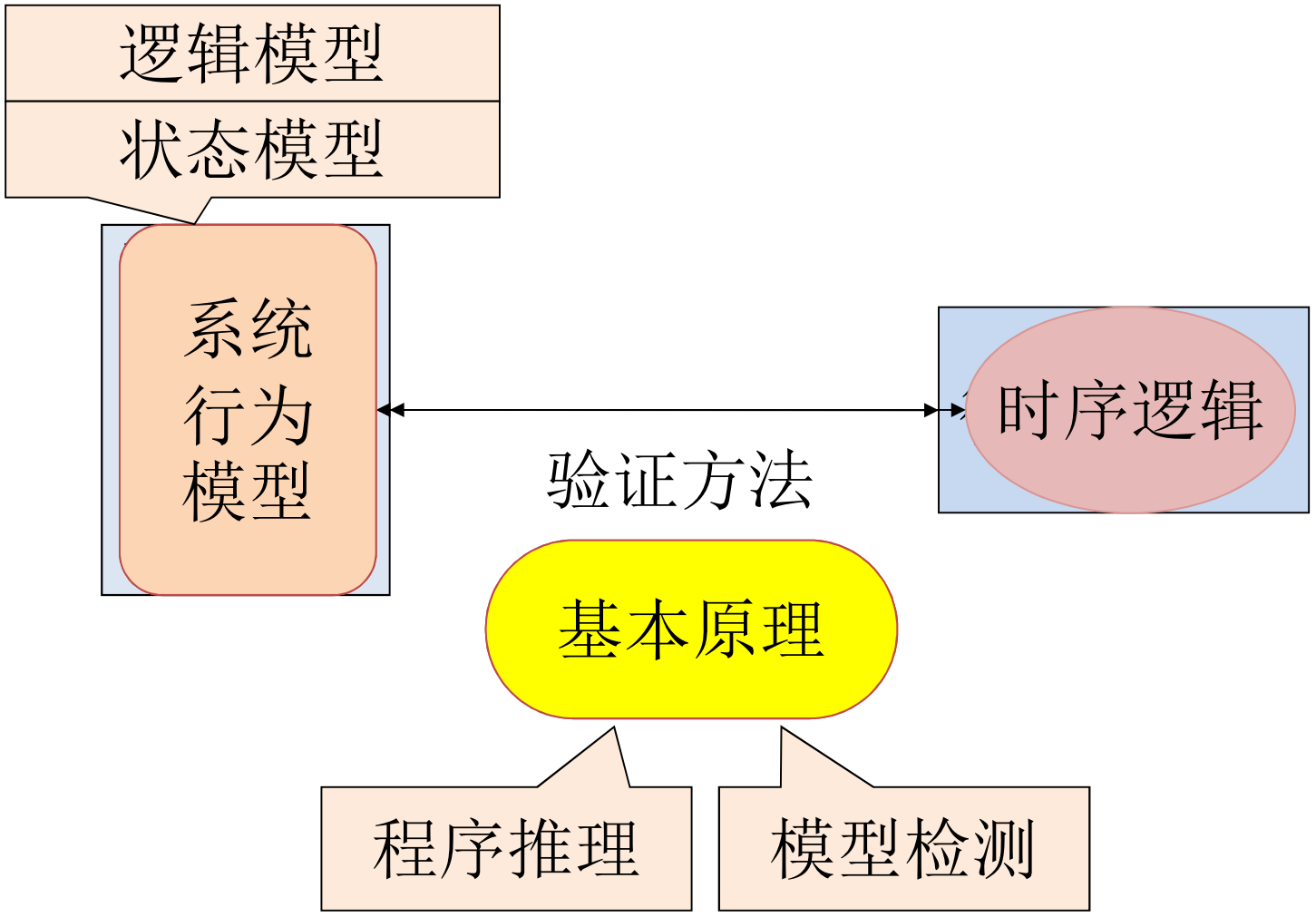
Fair and Labeled Kripke Structures

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



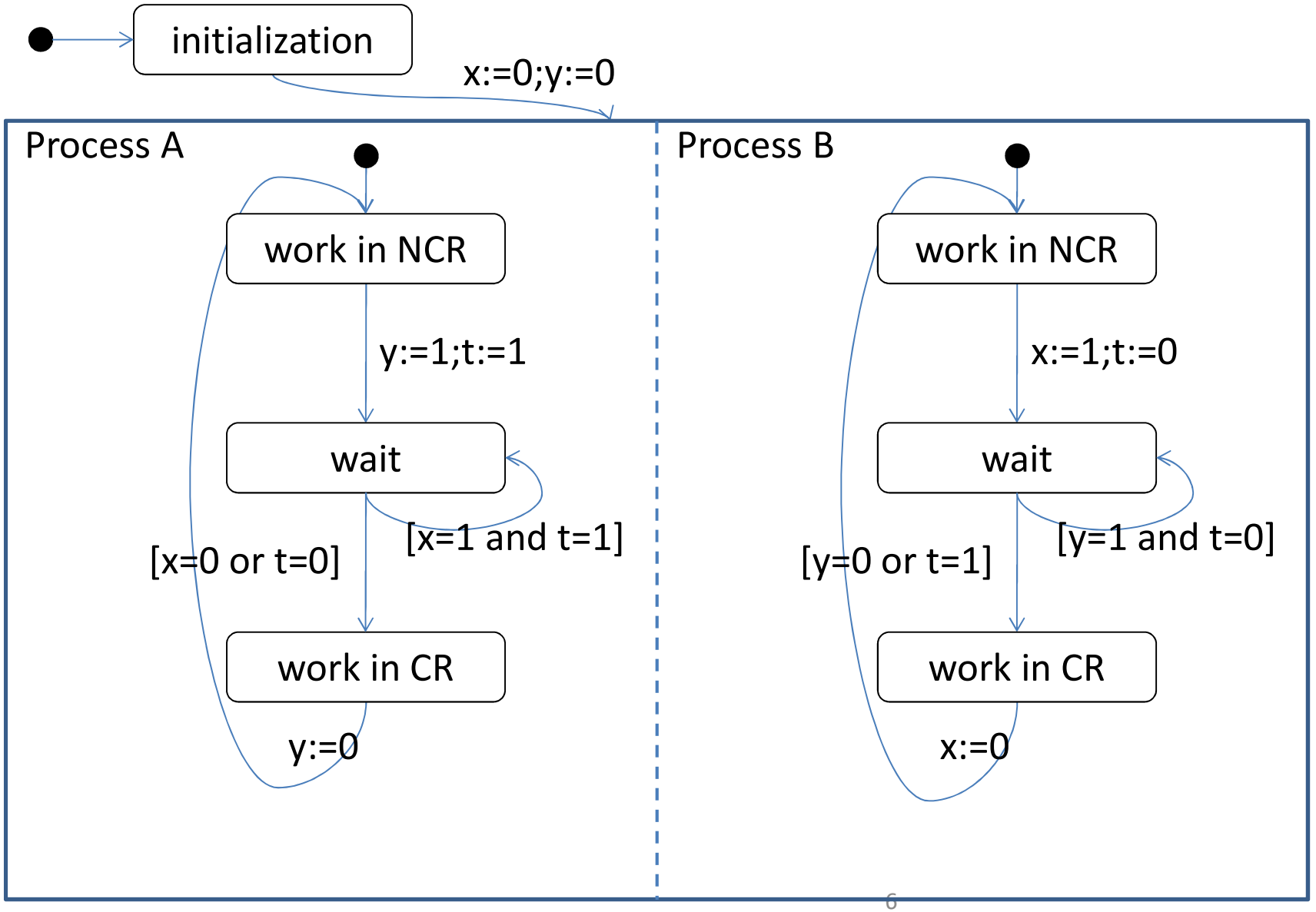
内容

- 公平Kripke结构
- 标号Kripke结构
- 公平标号Kripke结构

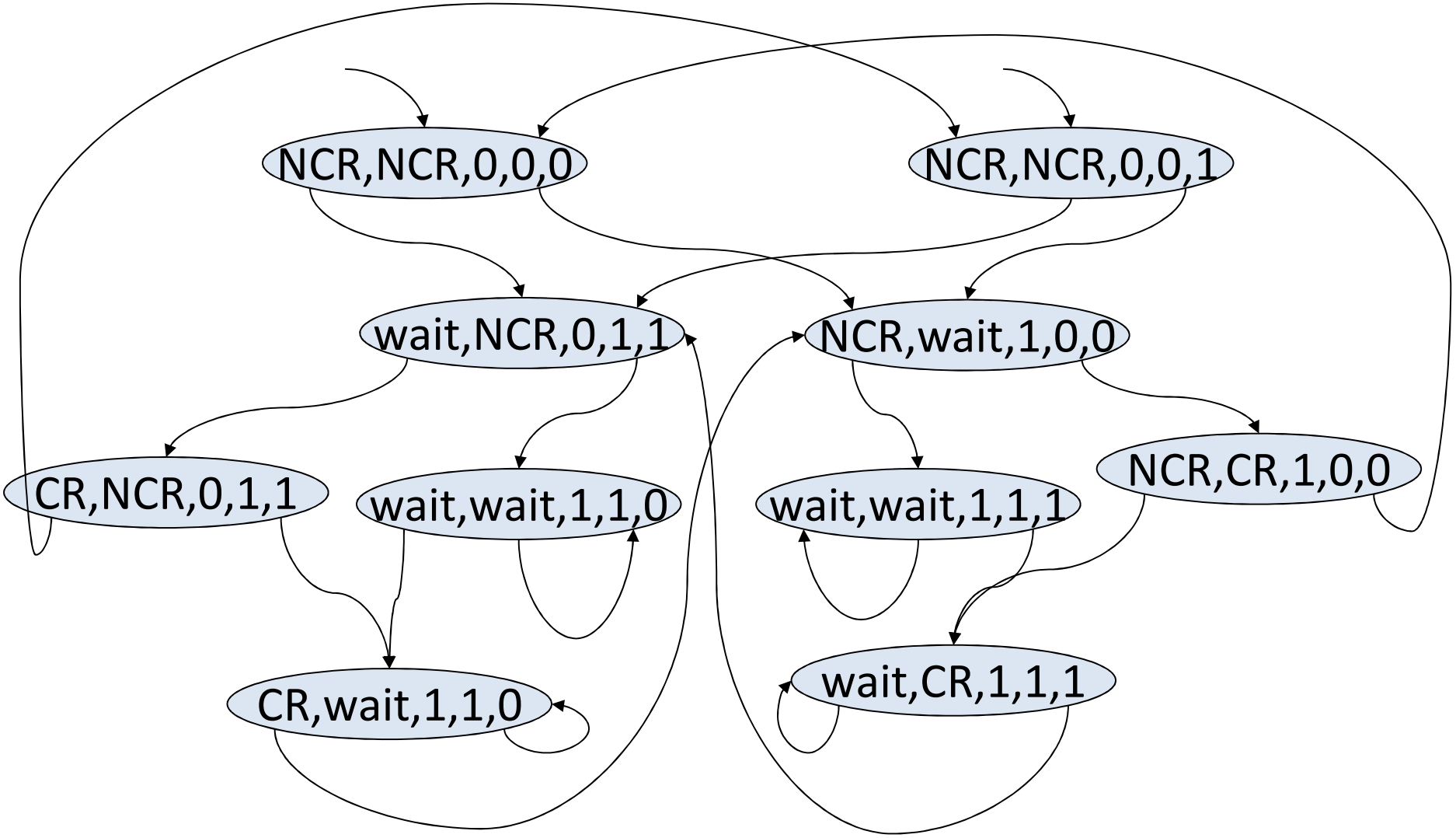
(I) Fair Kripke Structures

- Motivation

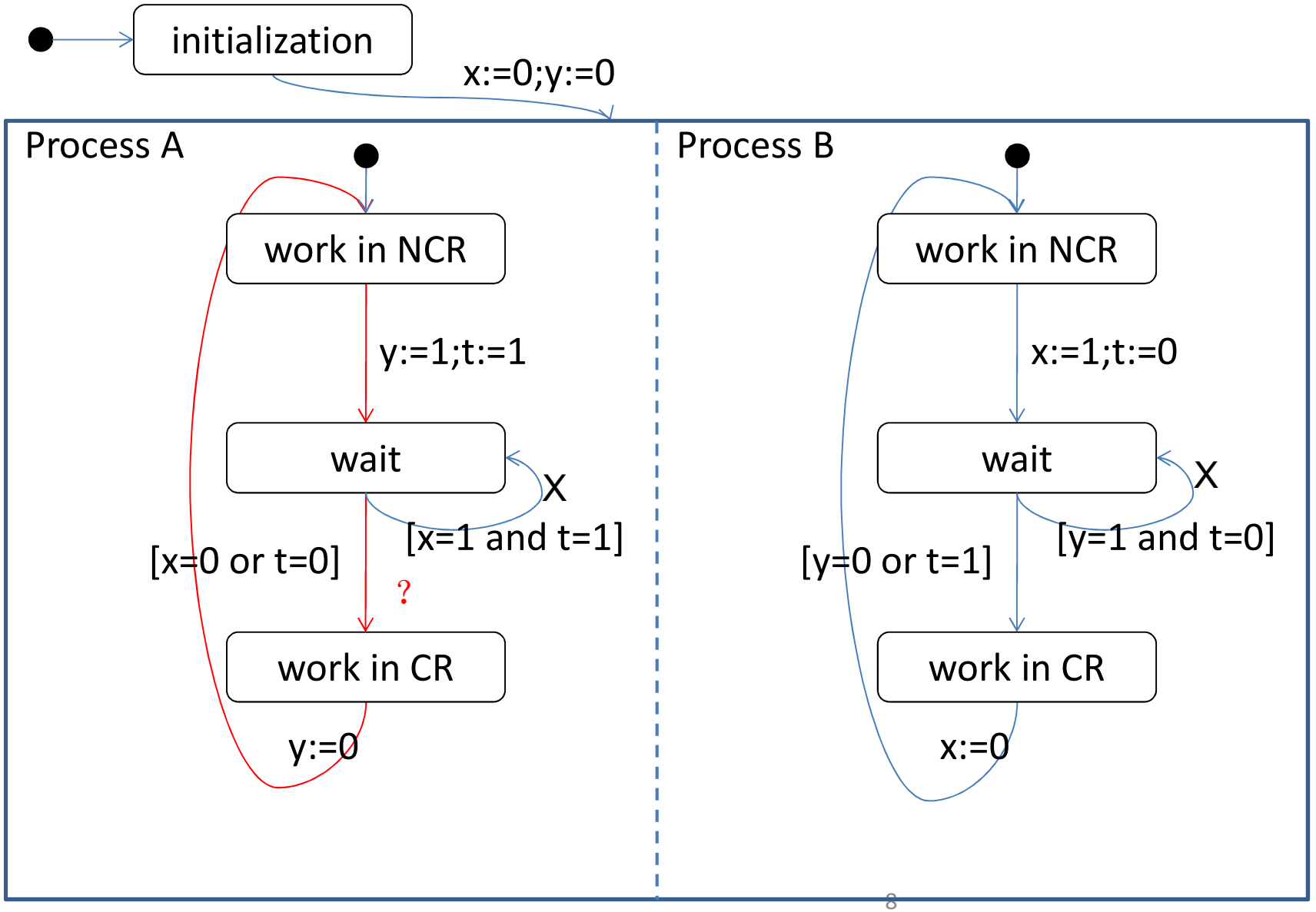
例子-互斥：状态图(State Diagram)



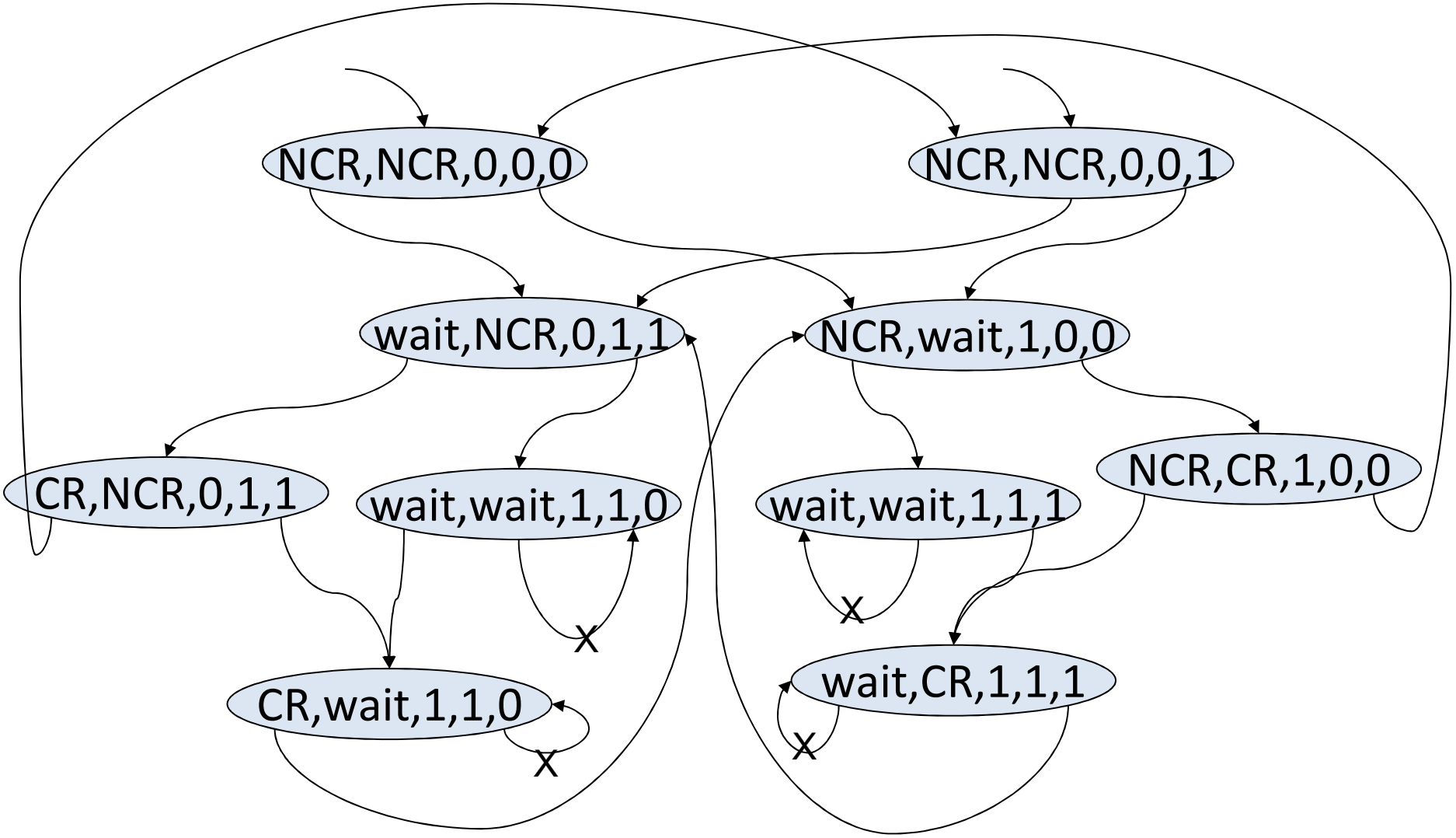
Kripke Structures



例子-互斥：状态图(State Diagram)



Kripke Structure (Modified)



Motivation

Restrict relevant computations to fair computations.

Fair Kripke Structures

- Formulation
- Basic Concepts
- Basic System Properties

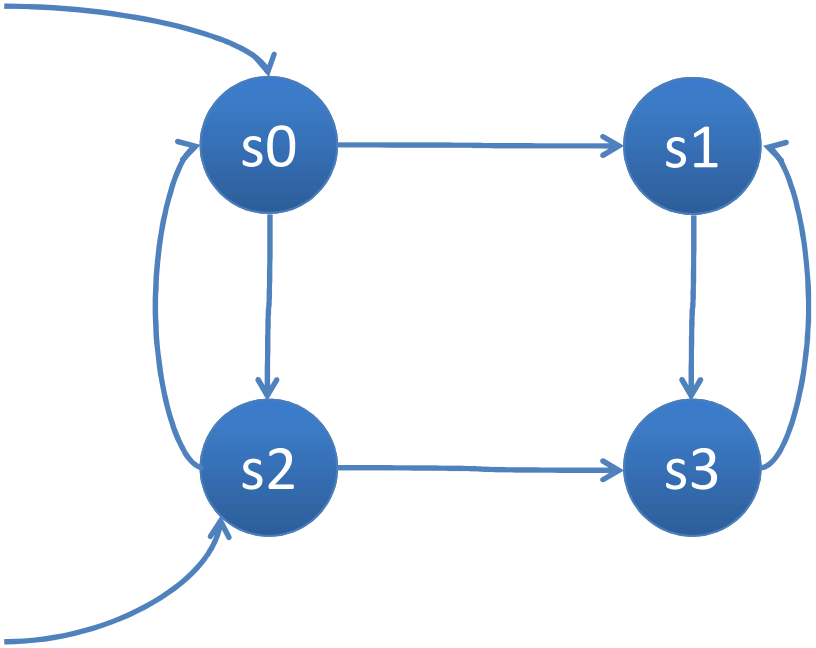
Fair Kripke Structures

Definition

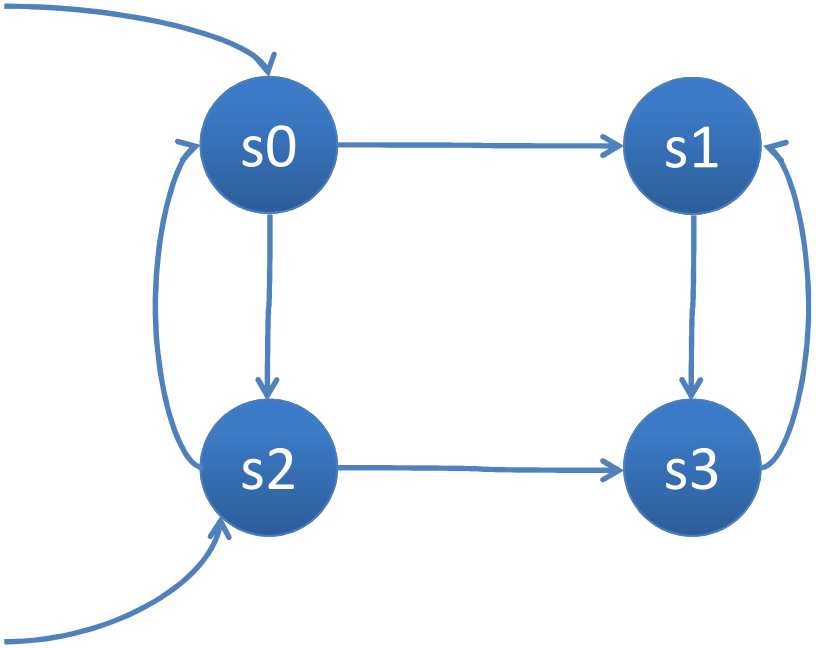
A fair Kripke structure is a quadruple <S,R,I,F>

- S : A finite set of states
- R $\subseteq$ S x S : A total transition relation
- I $\subseteq$ S : A set of initial states
- F $\subseteq$ 2^S : A set of sets of states

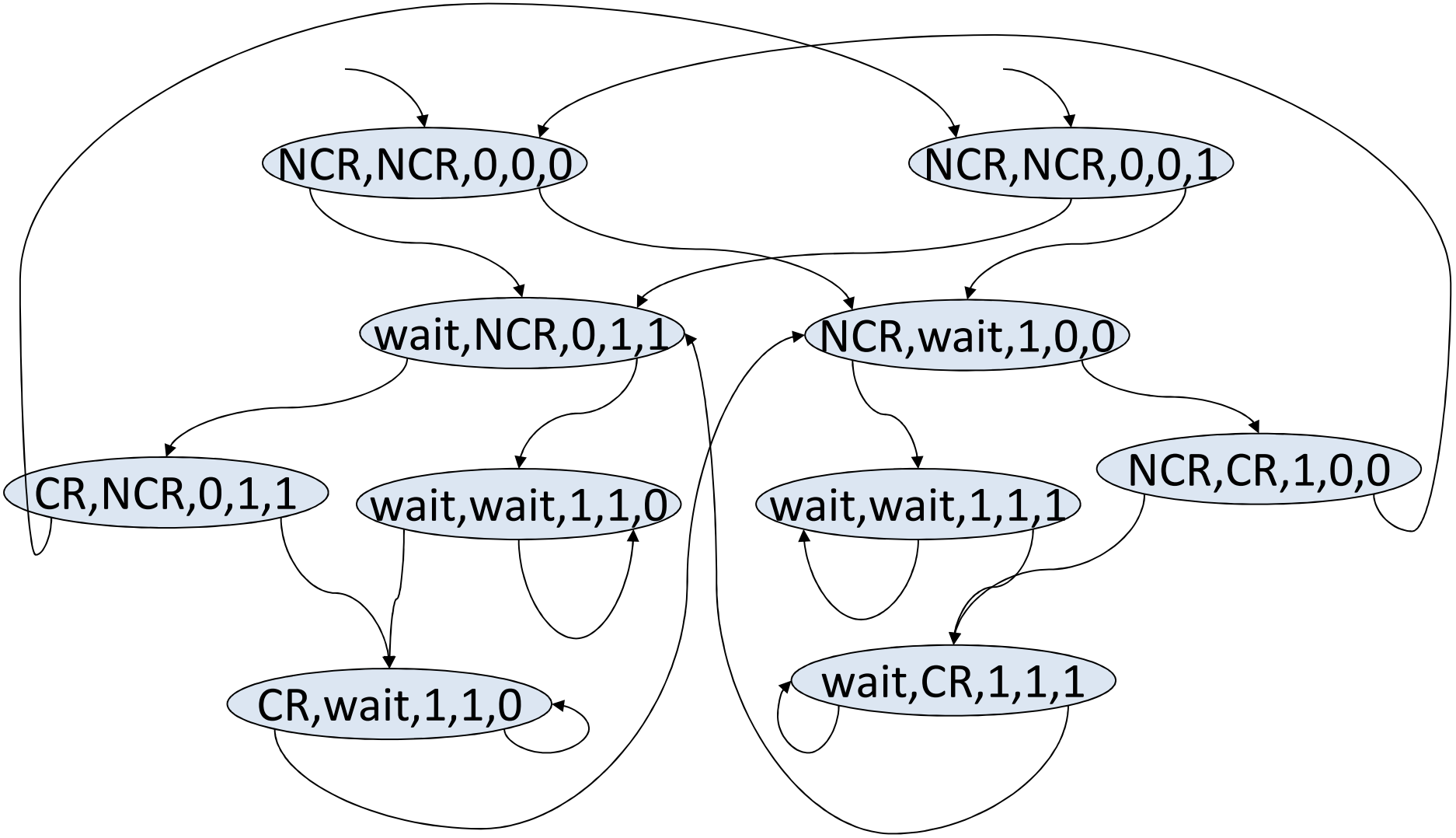
Example: S,R,I



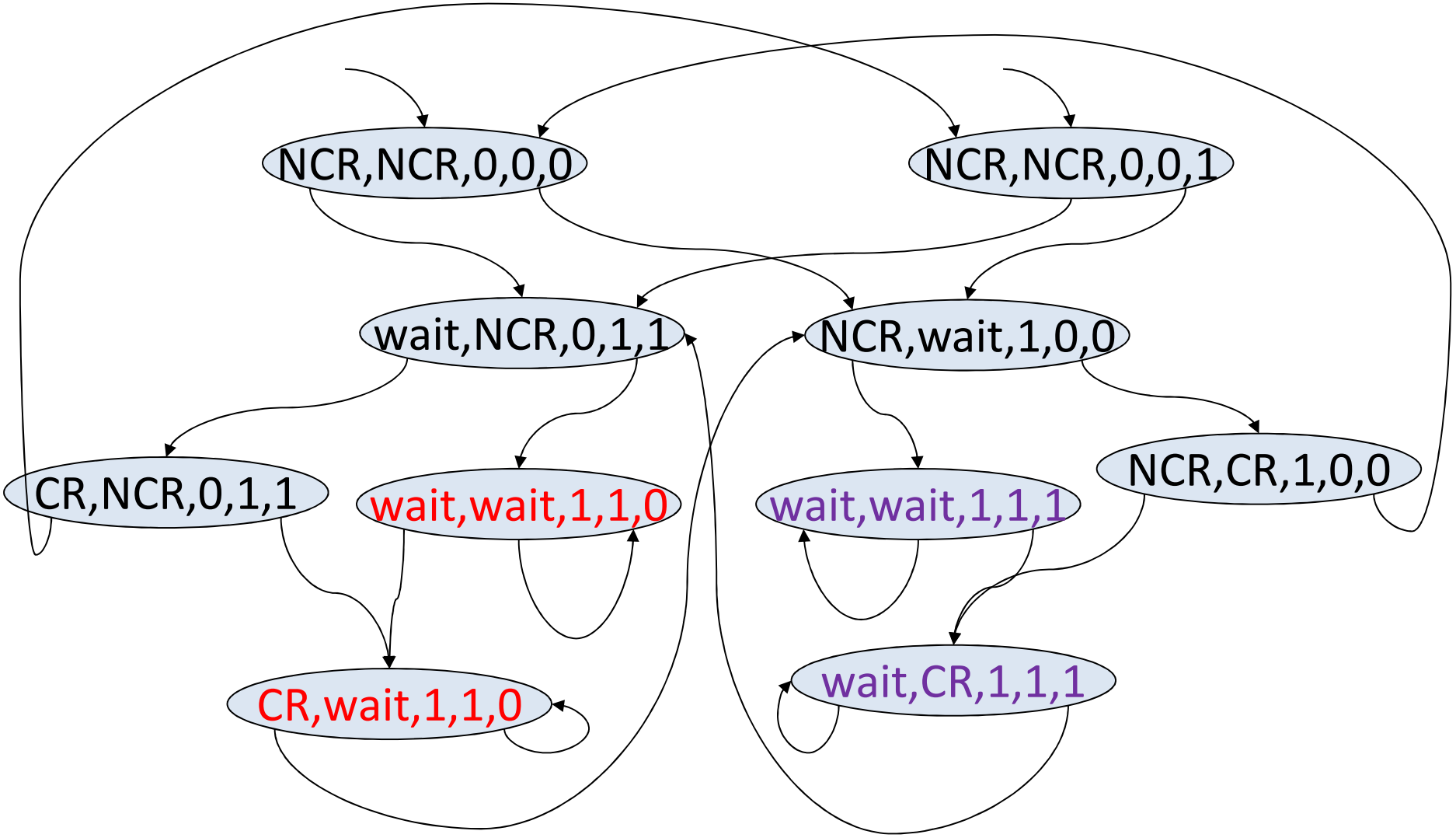
Example: F={{s2,s3},{s0,s2}}



Example:


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

Example:


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

Basic Concepts

Basic Concepts

- Basic Concepts (Kripke Structures)
  - States, Transition Relation, Initial States
  - Successors, Predecessors,
  - Reachable States, Reachability Relation,
  - Paths (Finite and Infinite), Computation, Behavior
  - Properties
- Basic Concepts (Fair Kripke Structures)
  - Paths – Fair Paths
  - Computations – Fair Computations
  - States – Fair States

Paths

Definition

An infinite path is an infinite sequence of S:

$s_0\, s_1\, s_2 \ldots$

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:

$s_0 \ldots s_n$

Fair Paths

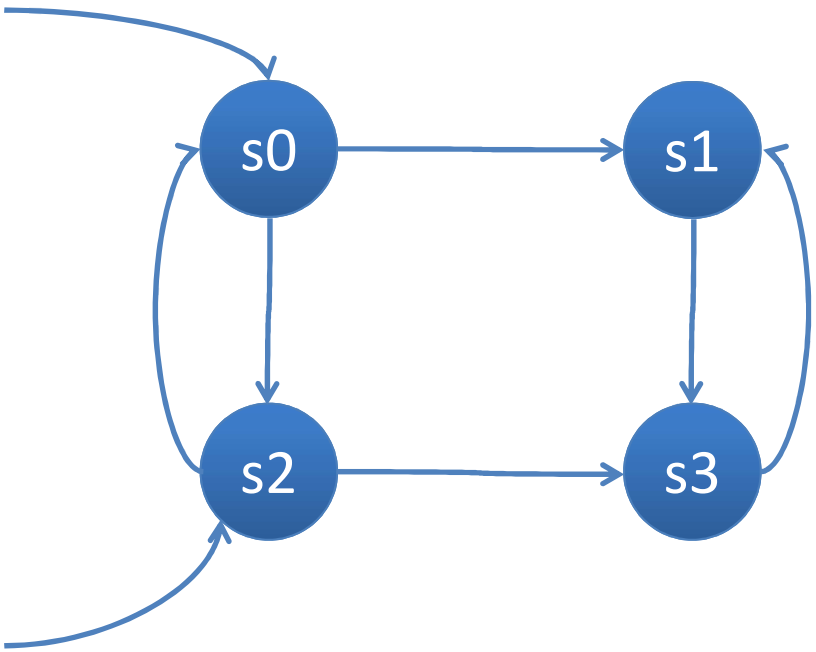
Definition

A fair (infinite) path is a path such that

for every f $\in$ F, there is an infinite number of states

on the path satisfying f.

Example: Paths



F={{s2,s3},{s0,s2}}

Infinite paths:

s0s1s3s1s3s1…

$s0(s1s3)^{\omega}$

s1s3s1s3s1s3s…

Finite paths:

s0s1s3s1

s1s3s1s3s1

Computations

Given a Kripke structure K=<S,R,I>.

Definition

A **computation** of K is an infinite sequence of S:

$s_0\, s_1\, s_2 \ldots$

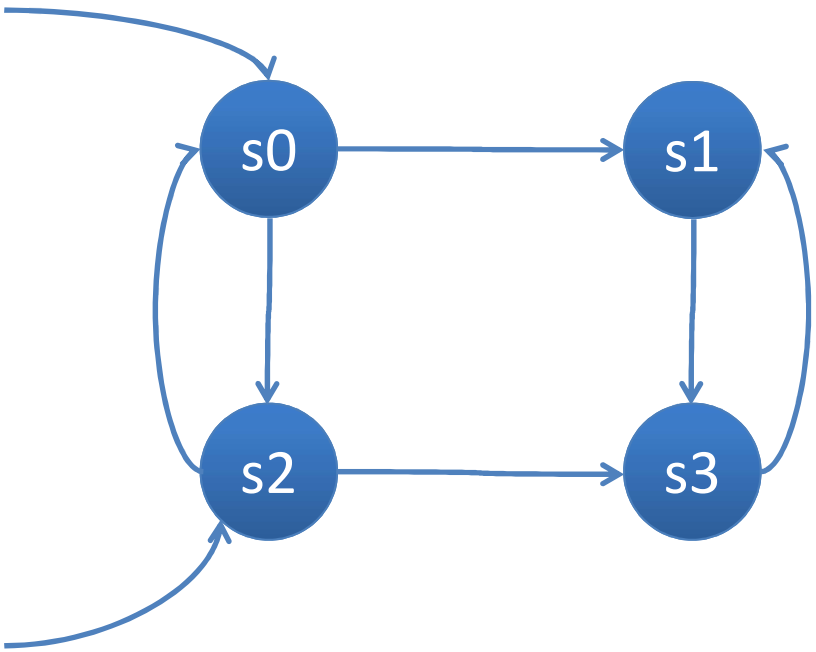
such that $s_0 \in I$, and $s_i \rightarrow s_{i+1}$ for all $i \geq 0$

Fair Computations

Definition

A **fair computation** is a computation such that for every $f \in F$, there is an infinite number of states on the computation satisfying f.

Example: Computations



F={{s2,s3},{s0,s2}}

$s0(s1s3)^{\omega}$

$s2(s3s1)^{\omega}$

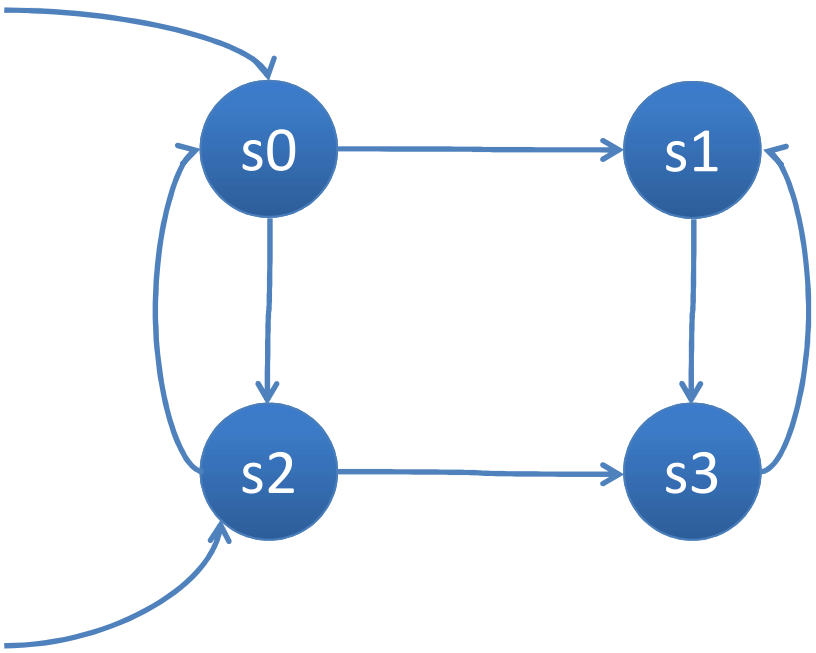
$s2(s0s2)^{*}(s1s3)^{\omega}$

System Behavior

System behavior = the set of fair computations

[[K]]

Example: Behavior



F={{s2,s3},{s0,s2}}

$(s0s2)^{\omega}$

$s0(s2s0)^{*}(s1s3)^{\omega}$

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

$(s2s0)^{\omega}$

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

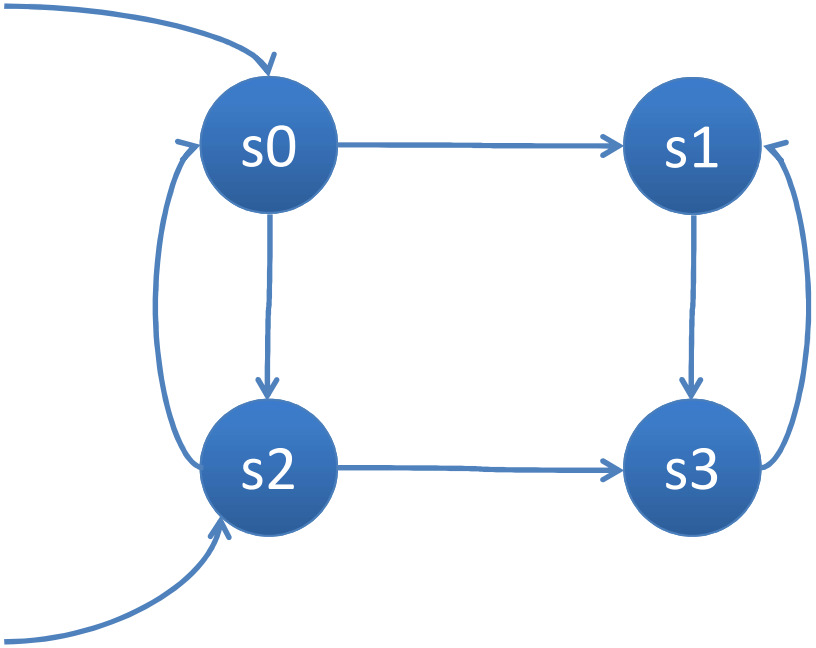
$s2s0(s2s0)^{*}(s1s3)^{\omega}$

Fair States

Definition

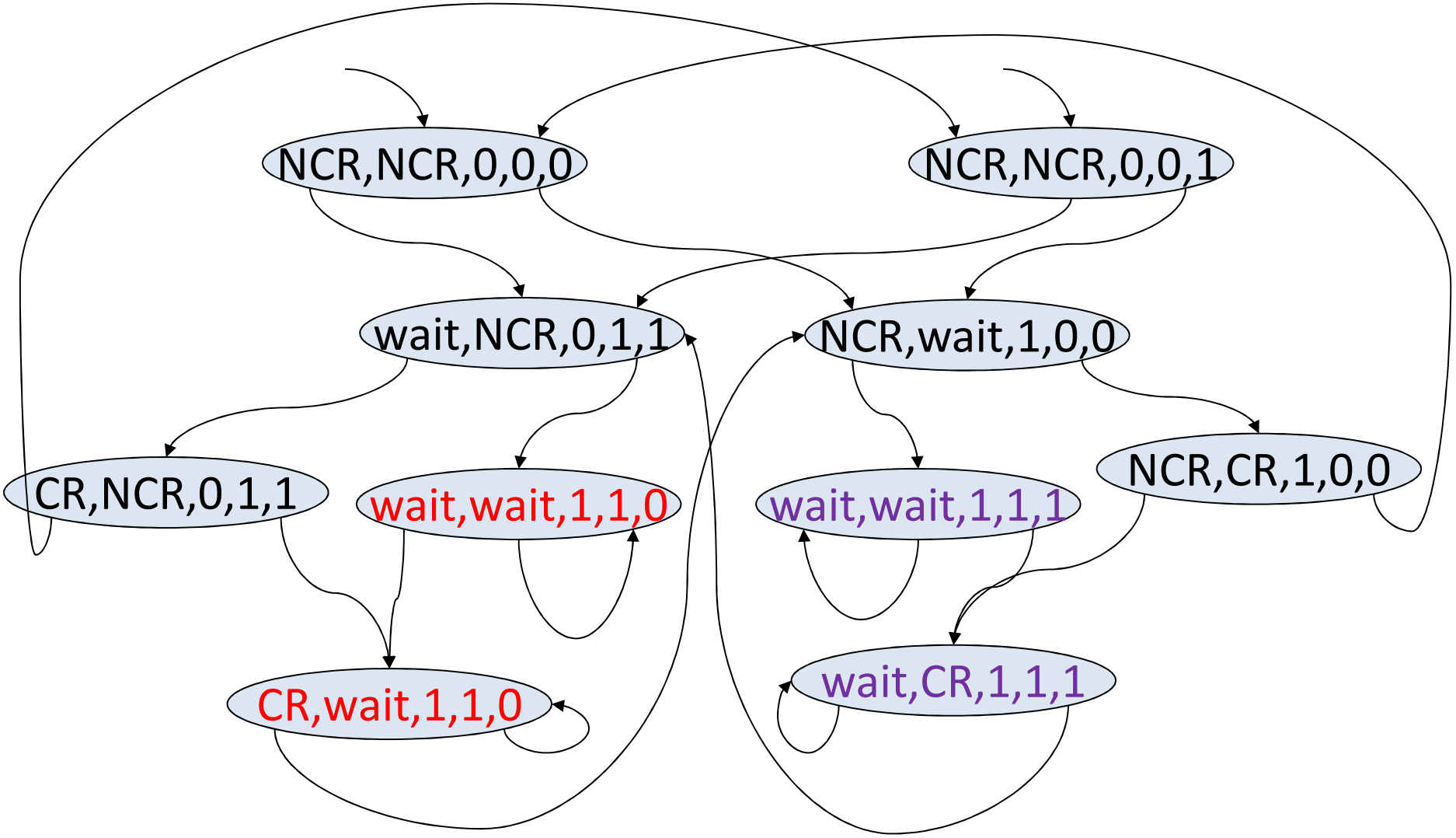
A fair state is a starting state of some fair path.

Example: Fair States



F={{s2,s3},{s0,s2}}

Example： F={ {..}, {..} }


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

Checking Fair States (Fair SCC)

Definition

A fair SCC is a non-trivial SCC

such that each fairness requirement is satisfied by some state of the SCC.

Lemma

A state is fair, iff it may reach a fair SCC.

Fair SCC

Let K=<S,R,I,F> be a Kripke structure, and e an SCC.

```
bool Fairscc(K,e)
{   if (nontrivial(K,e)=false) return false;
    for (each f in F) if (e∩f= ∅) { return false; }
    return true;
}
```

Fair States

Let K=<S,R,I,F> be a Kripke structure, and A a set of states.

```
bool ExistFairState(K,A)
{   G:=(S,R); scclist:=scctarjan(G);
    w:={};
    for (each e in scclist) if (Fairscc(K,e)) w=w∪e;
    K'=(S,R,A);
    return ReachbilityAnalysis(K',w);
}
```

Fair States

Let K=<S,R,I,F> be a Kripke structure, A a set and s a state.

Proposition

There exists some fair state in A iff

ExistFairState(K,A)=true.

Proposition

s is a fair state iff

ExistFairState(K,{s})=true.

Basic Properties

- Fair Reachability, Fair Safety
- Fair Avoidability, Fair Inevitability
- Emptiness

Basic Properties (1)

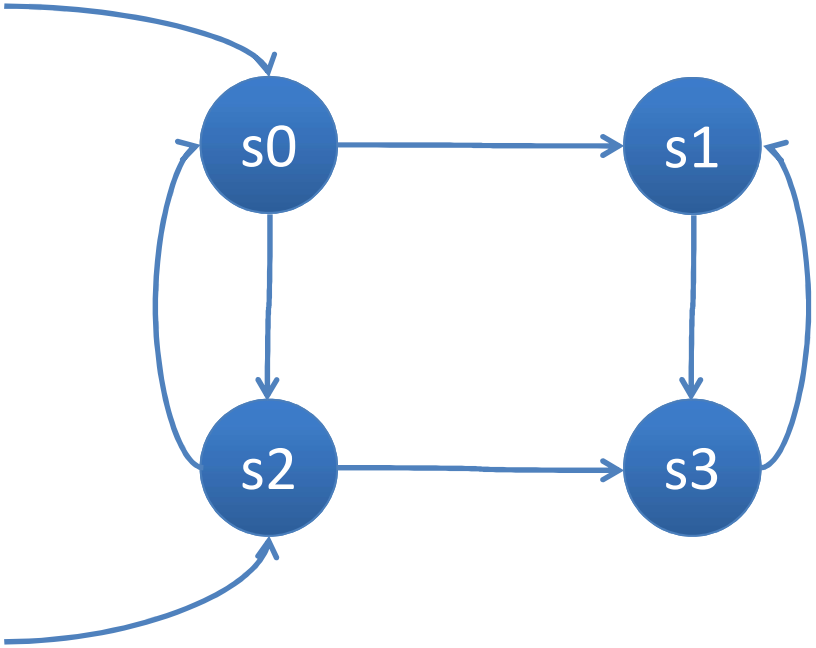
- Fair Reachability, Fair Safety

Fair Reachability Property - Possibility

Let X be a set of states.

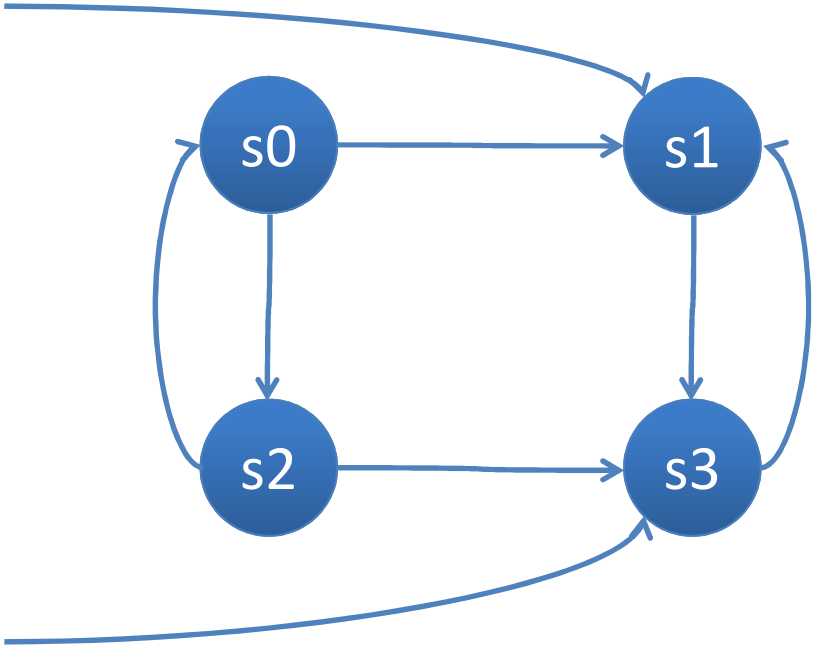
X is a fair reachability property, if
There is a fair computation of K that reaches
an X-state.

Example: {s0,s2}√, {s0,s1,s2}√



F={{s2,s3},{s0,s2}}

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



F={{s2,s3},{s0,s2}}

Fair Reachability Problem

Given a set $X \subseteq S$.

Is X a fair reachability property?

Reachability Analysis (Existing One)

Let K=<S,R,I> be a Kripke structure, 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;
}
```

Fair Reachability Analysis

- Is $A \subseteq S$ a fair reachability property?

- Is there a state s such that the following holds?
  - s satisfies A
  - s is fair
  - s is reachable (from I)

Fair Reachability Analysis

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

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

Fair Reachability Analysis

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

Proposition:

FairReachabilityAnalysis(K,A) =true

Iff

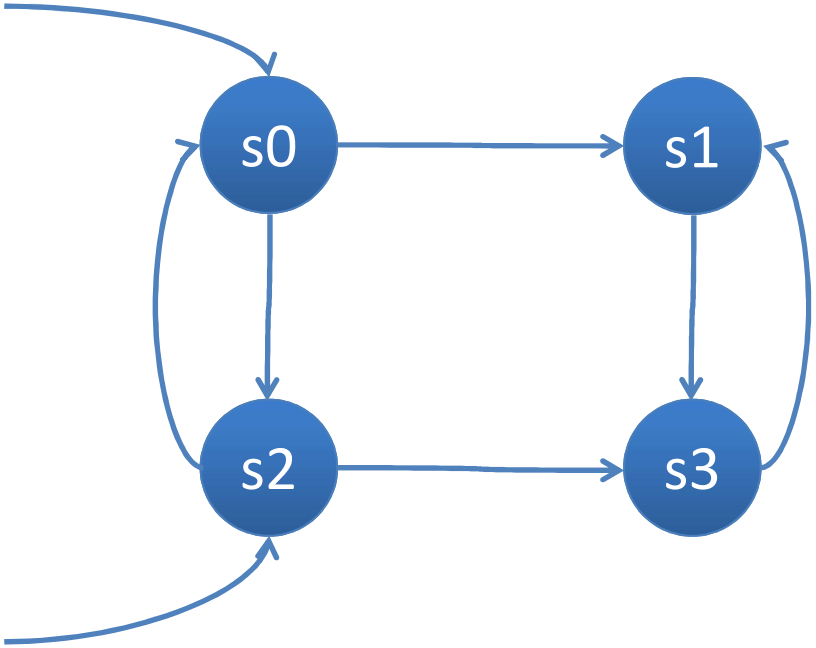
A is a fair reachability property

Fair Safety Property - Universality

Let Y be a set of states.

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

Example: {s0,s2}√, {s1,s2,s3}x



F={{s2,s3},{s0,s2}}

Fair Safety Analysis

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

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

Basic Fair Safety Analysis

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

Proposition:

FairSafetyAnalysis(K,A) =true

Iff

A is a fair Safety property

Reachability & Safety

Fair safety is a dual property of fair reachability.

Proposition

A is a fair safety property of a system K=<S,R,I,F>,
iff S\A is not a fair reachability property of K.

FairSafetyAnalysis(K,A) =true $\Leftrightarrow$
FairReachabilityAnalysis(K,S\A) =false

Basic Properties (2)

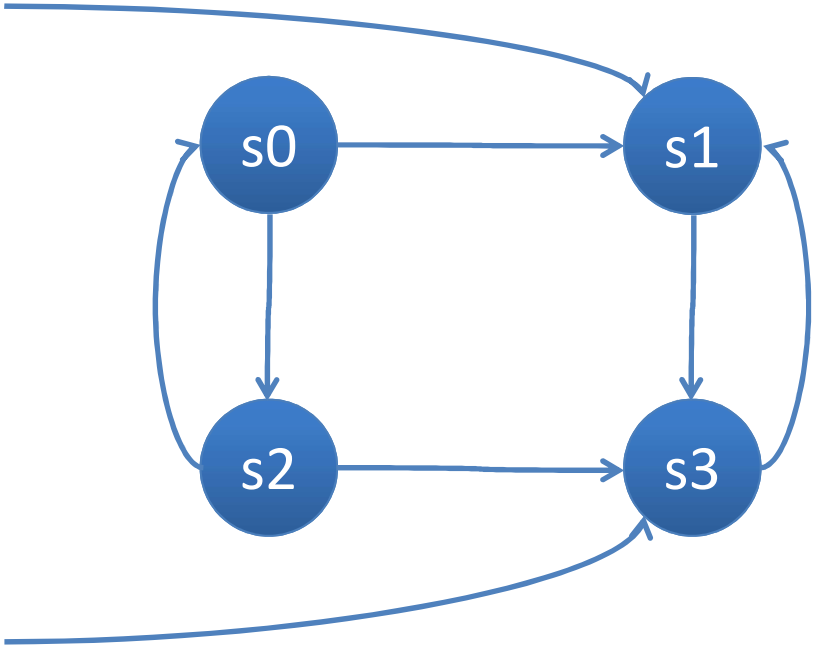
- Fair Avoidability, Fair Inevitability

Avoidability Property

Let X be a set of states.

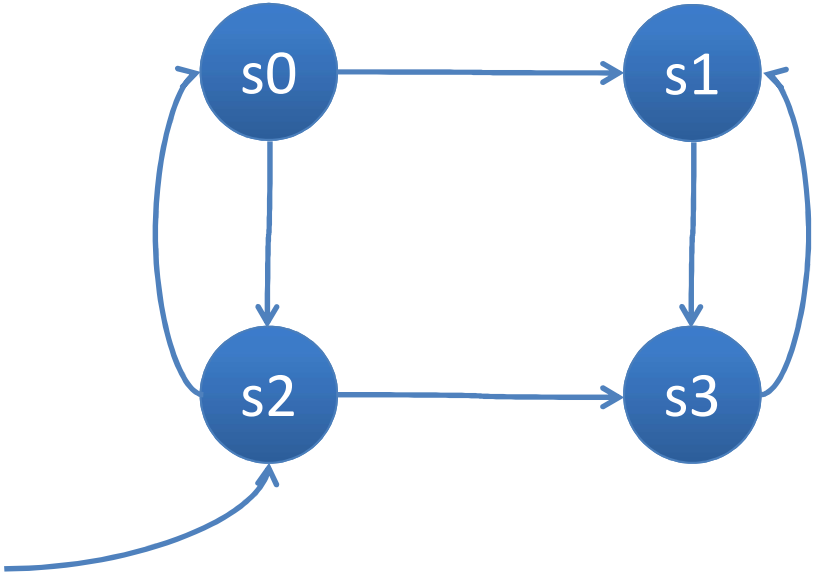
X is a fair avoidability property, if
there exists some fair computation of K that
does not pass any X-state.

Example: {s0,s2}x



F={{s2,s3},{s0,s2}}

Example: {s1,s3}√



F={{s2,s3},{s0,s2}}

Fair Avoidability Problem

Given a set $X \subseteq S$.

Is X a fair avoidability property?

Fair Avoidability Problem

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

Define $R|Y=R\cap(Y\times Y)$; $\{f1,\ldots,fn\}|Y=\{f1\cap Y,\ldots,fn\cap Y\}$

Define $K|Y = \langle Y,R|Y,I\cap Y,F|Y\rangle$

Let $A\subseteq S$. Let $K'= \langle S',R',I',F'\rangle = K|(S\setminus A)$

A is a fair avoidability property, iff

there is a reachable fair SCC of K' .

Fair Avoidability Analysis

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

```
bool FairAvoidabilityAnalysis(K,A)
{  K':=(S',R',I',F'):=K|(S\A); G:=(S',R');  K'':=(S',R',I');
    scclist:=scctarjan(G);
   w:={}; for each (e in scclist) if (Fairscc(K',e)) w:=w∪e;
    return ReachbilityAnalysis(K'',w);
}
```

Fair Avoidability: A Simpler Formulation

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

Let $A \subseteq S$. Let K'= <S',R',I',F'> = K|(S\A)

A is a fair avoidability property, iff

there is a reachable fair SCC of K', iff

there is a reachable fair state of K', iff

there is a fair state in I', iff

ExistFairState(K',I')=true.

Fair Avoidability: A Simpler Formulation

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

```
bool FairAvoidabilityAnalysis(K,A)
{
      K’:= (S’,R’,I’,F’) := K|(S\A);
      return (ExistFairState(K’,I’));
}
```

Fair Avoidability Analysis

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

Proposition:

FairAvoidabilityAnalysis(K,A) =true

iff

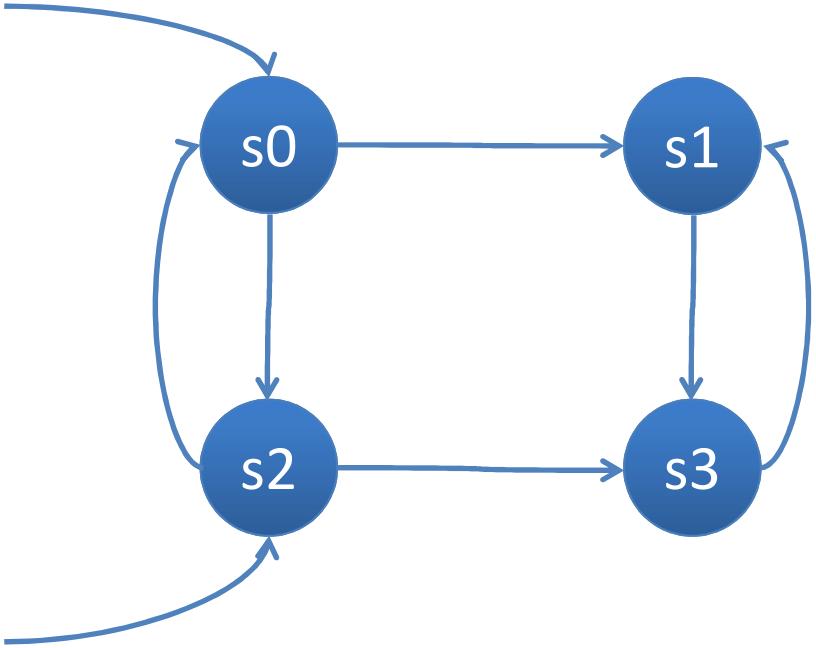
A is a fair avoidability property

Fair Inevitability

Given a set Y.

Y is a fair inevitability property, if
every fair computation passes a Y-state

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



F={{s2,s3},{s0,s2}}

Inevitability & Avoidability

Fair inevitability is the negation of fair avoidability.

Proposition

A is a fair inevitability property of a system K,

iff A is not a fair avoidability property of K.

Fair Inevitability Analysis

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

```
bool FairInevitabilityAnalysis(K,A)
   {  K':=(S',R',I',F'):=K|(S\A);  G:=(S',R');
      scclist:=scctarjan(G);
      w:={}; for each (e in scclist) if (Fairscc(K',e)) w:=w∪e;
      return (not ReachbilityAnalysis(K',w));
   }
```

Fair Inevitability Analysis

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

Proposition:

FairInevitabilityAnalysis(K,A) =true

Iff

A is a fair inevitability property

Inevitability & Avoidability

Fair inevitability is the negation of fair avoidability.

FairInevitabilityAnalysis(K,A) =true $\Leftrightarrow$
FairAvoidabilityAnalysis(K,A) =false

Basic Properties (3)

- Emptiness

Emptiness

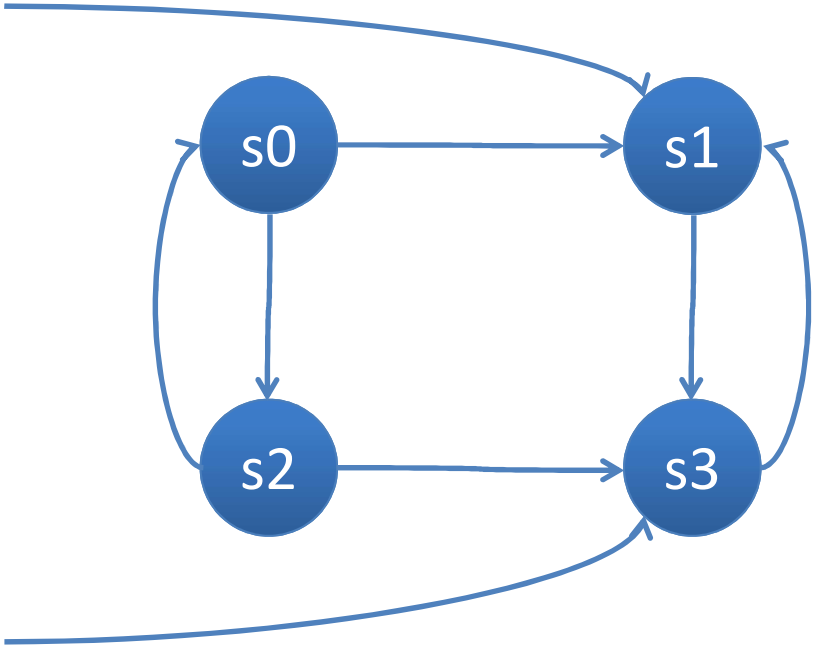
Let K=<S,R,I,F> .

K is empty,

the set of fair computations of K is empty,

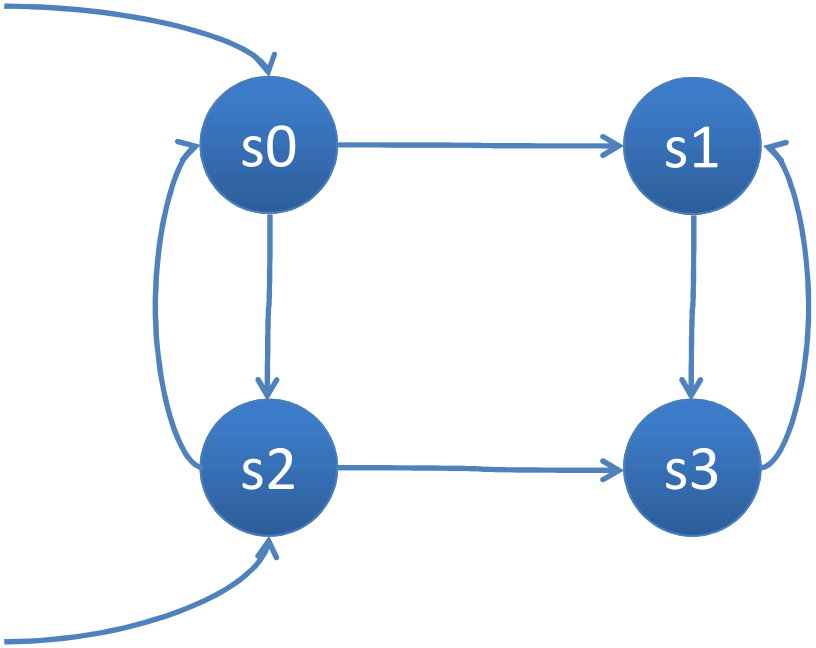
[[K]] = ∅

Example: Empty



F={{s2,s3},{s0,s2}}

Example: Non-empty



F={{s2,s3},{s0,s2}}

Emptiness Problem

Given K=<S,R,I,F> .

Is K empty?

Emptiness Problem

Given K=<S,R,I,F> .

Is K empty?

K is nonempty, iff
there is an I-state that reaches a fair SCC, iff
there is a fair initial state.

Emptiness Checking

Let K=<S,R,I,F> be a fair Kripke structure.

```
bool EmpChecking(K)
{
    return ExistFairState(K,I)=false;
}
```

Emptiness Checking

Let K=<S,R,I,F> be a fair Kripke structure.

Proposition:

EmpChecking(K) =true

iff

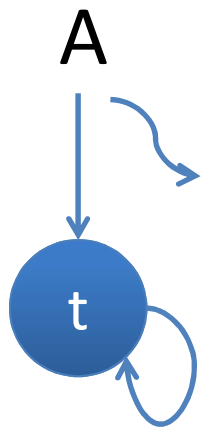
K is empty

Application to Reachability

Given K=<S,R,I>, and A⊆S.

Define S',R',I',F as follows:

- S'=S∪{t}
- R'= R ∪ { (s,t) | s∈A } ∪ { (t,t) }
- I'=I
- F={{t}}



A is a reachability property, iff <S',R',I',F>

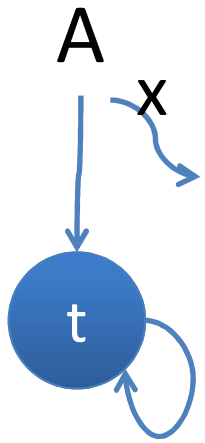
is nonempty

Application to Avoidability

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

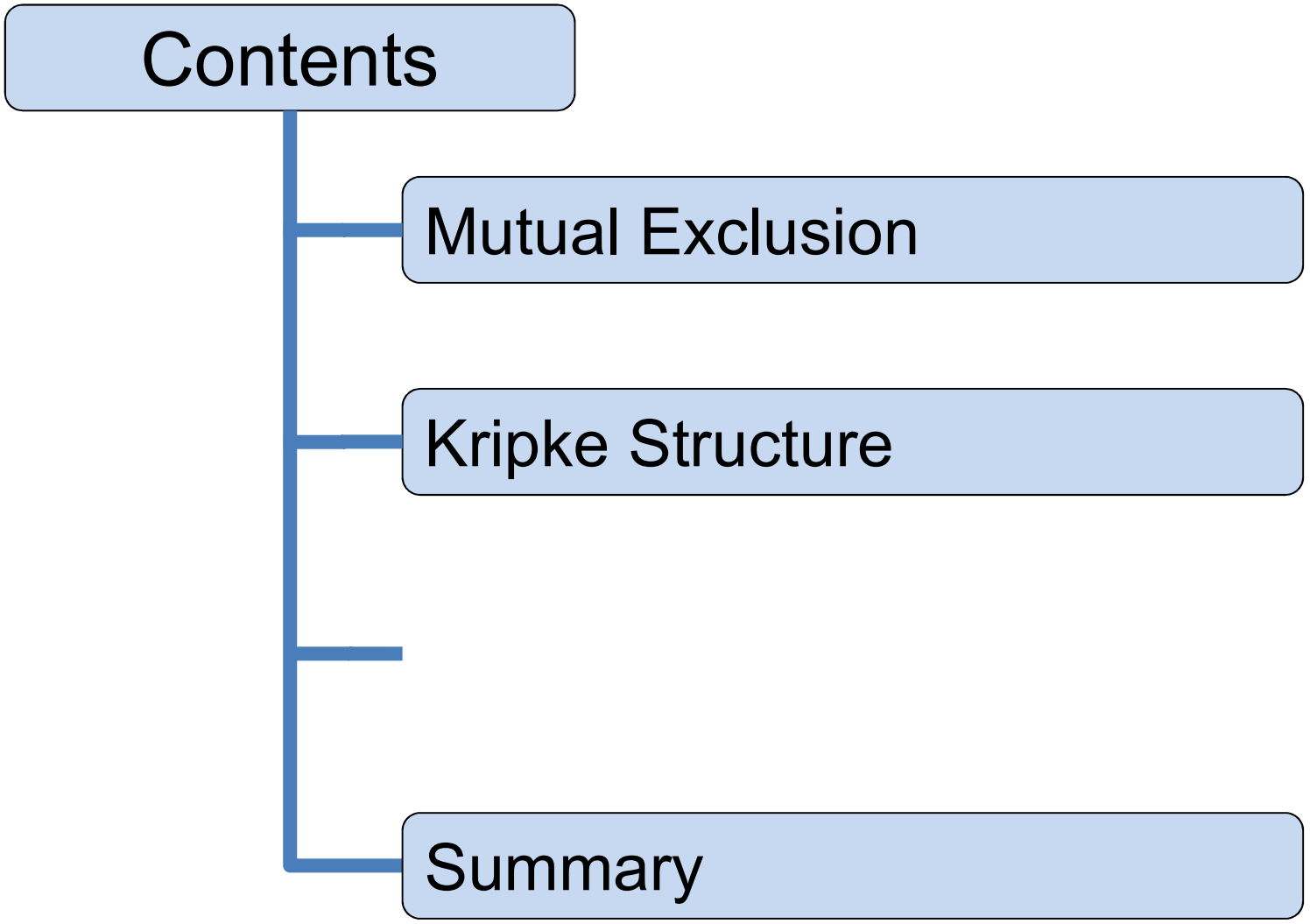
Define S',R',I',F as follows:

- $S'=S \cup \{t\}$
- $R'= \{ (s,s') \mid (s,s') \in R, s \notin A \} \cup \{ (s,t) \mid s \in A\} \cup \{ (t,t) \}$
- $I'=I$
- $F=\{S\}$

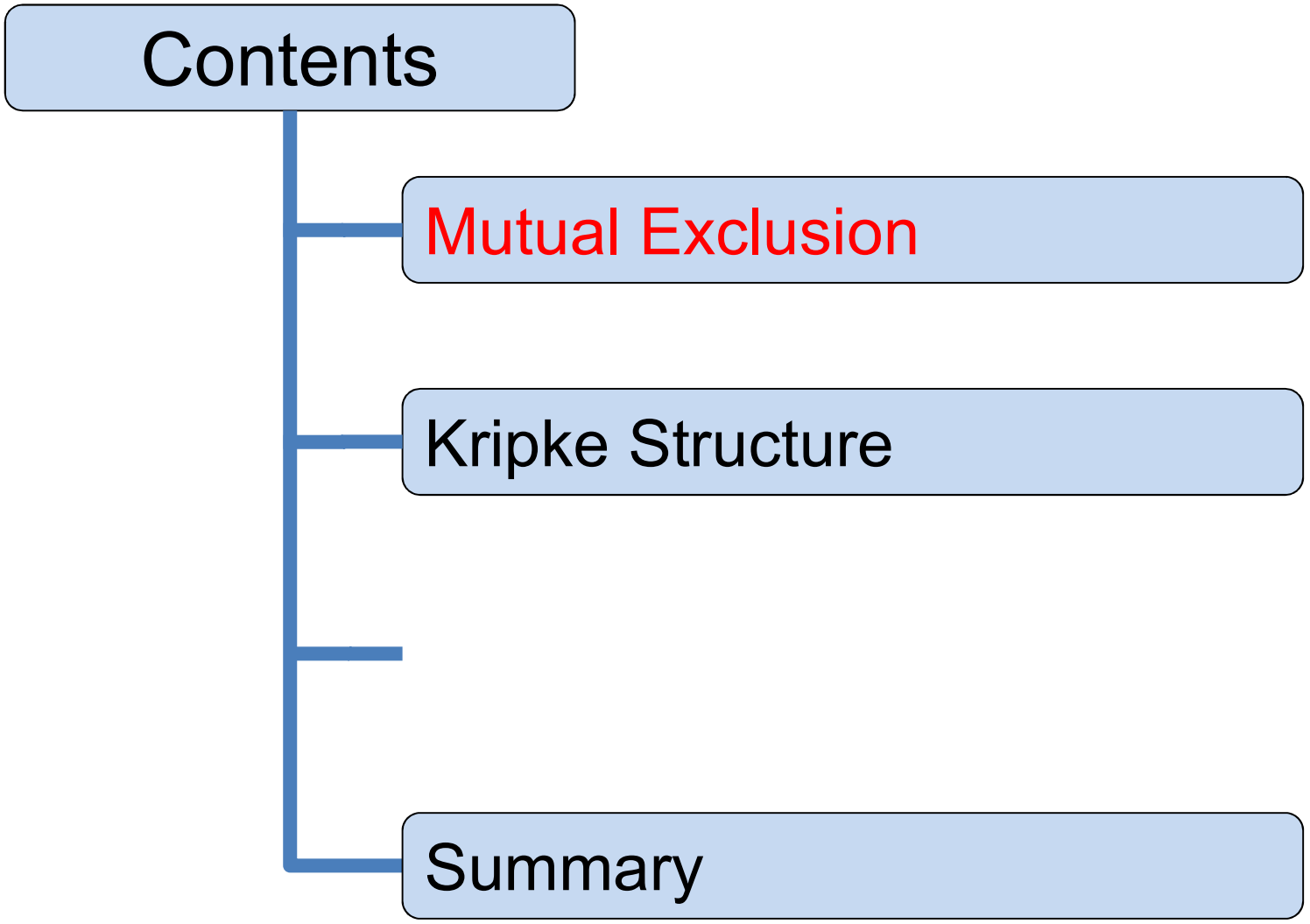


A is an avoidability property, iff $\langle S',R',I',F\rangle$ is nonempty

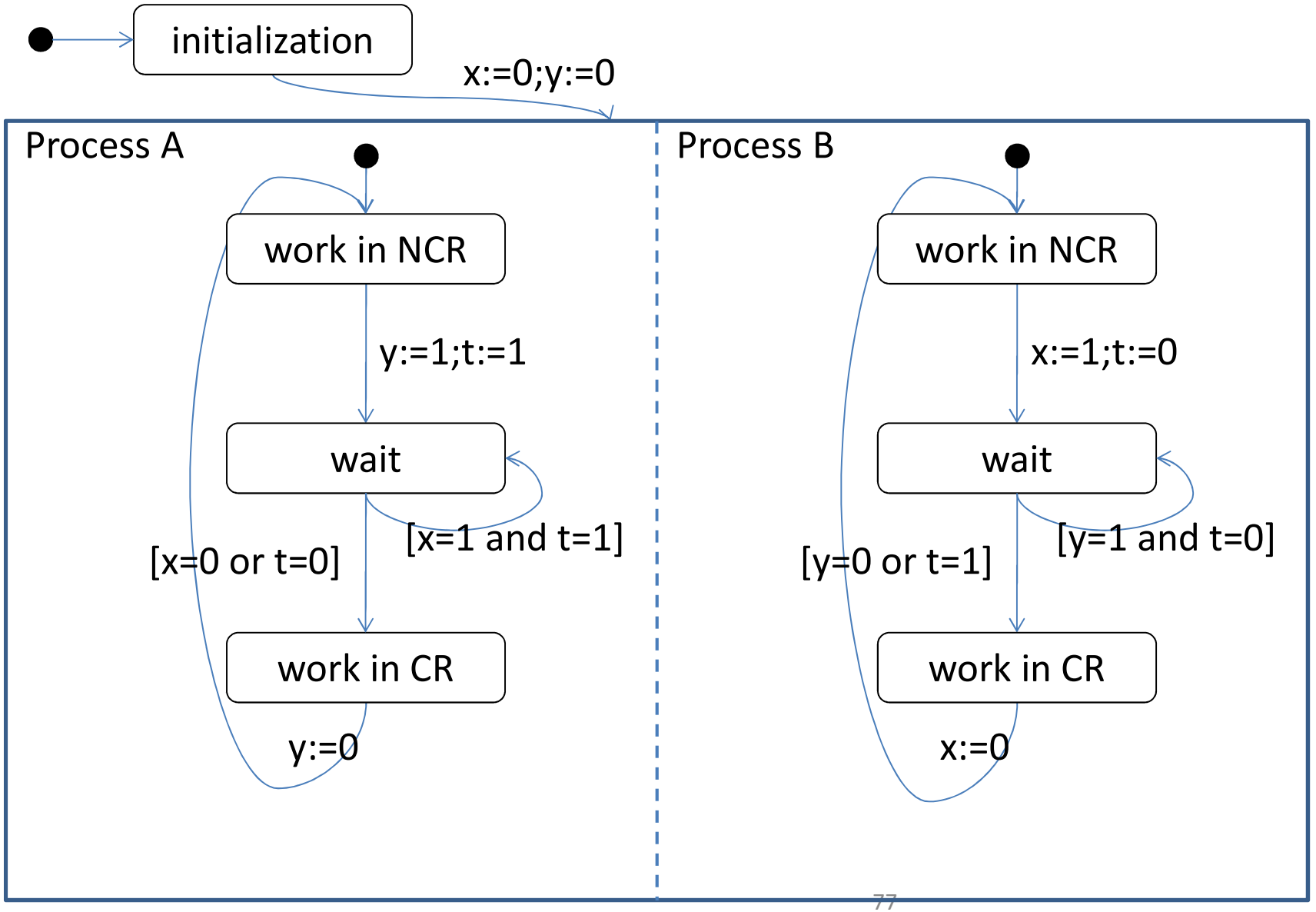
An Example



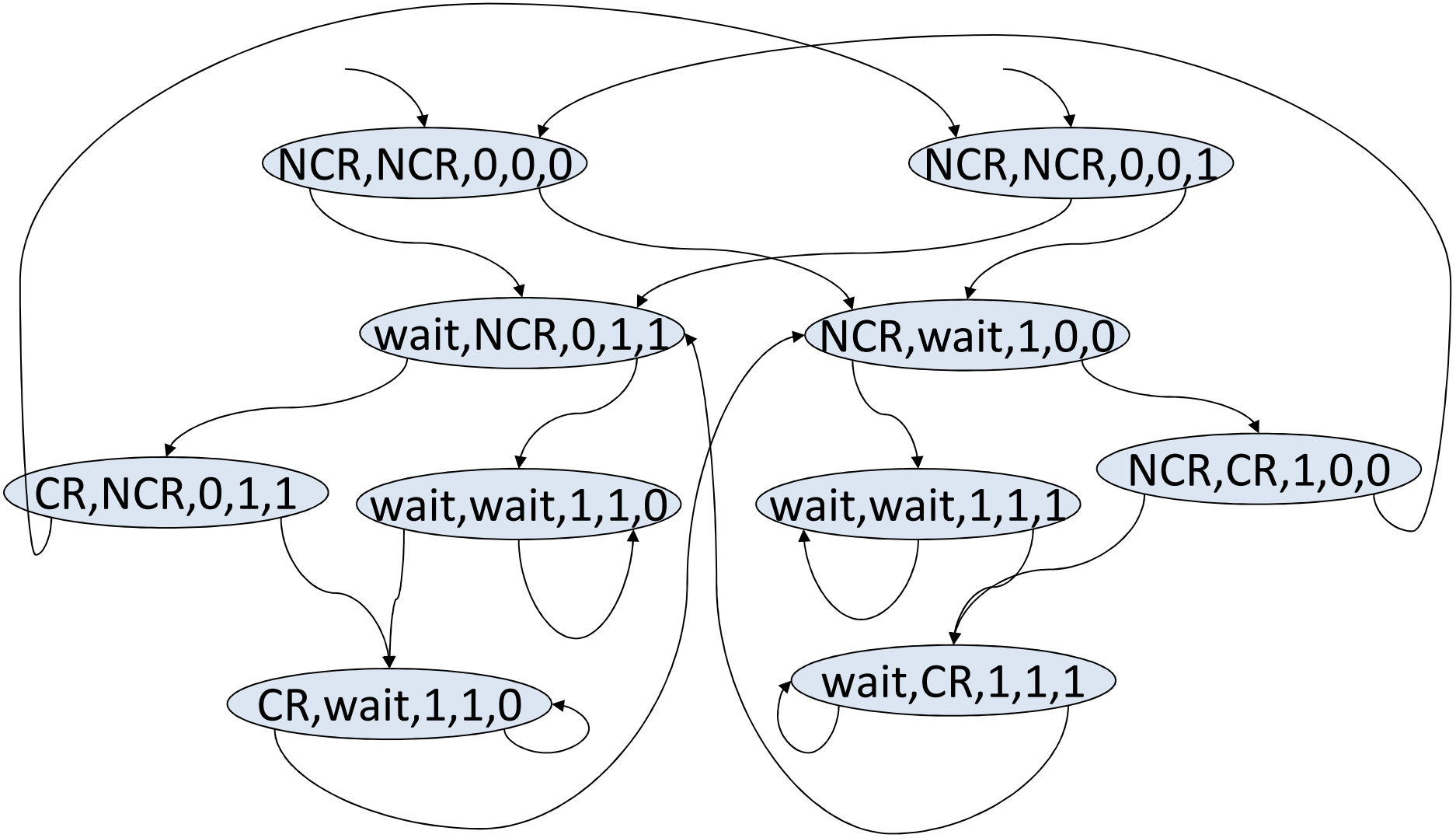
An Example



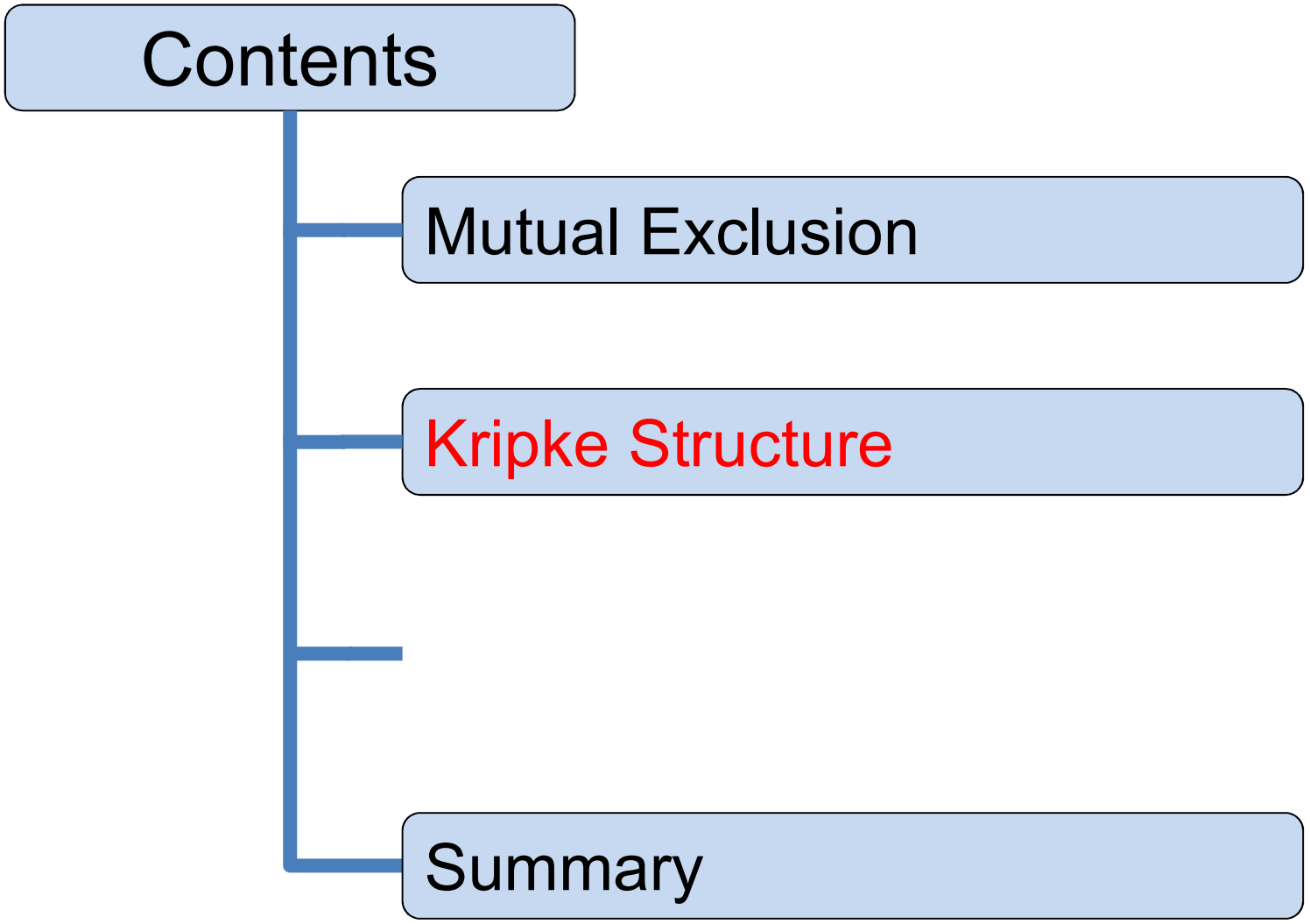
例子-互斥：状态图(State Diagram)



Example


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

An Example



Fair Kripke Structures

Definition

A fair Kripke structure is a quintuple <S,R,I,F>

- S : A finite set of states
- R $\subseteq$ S x S : A total transition relation
- I $\subseteq$ S : A set of initial states
- F $\subseteq$ 2^S : A set of sets of states

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,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)

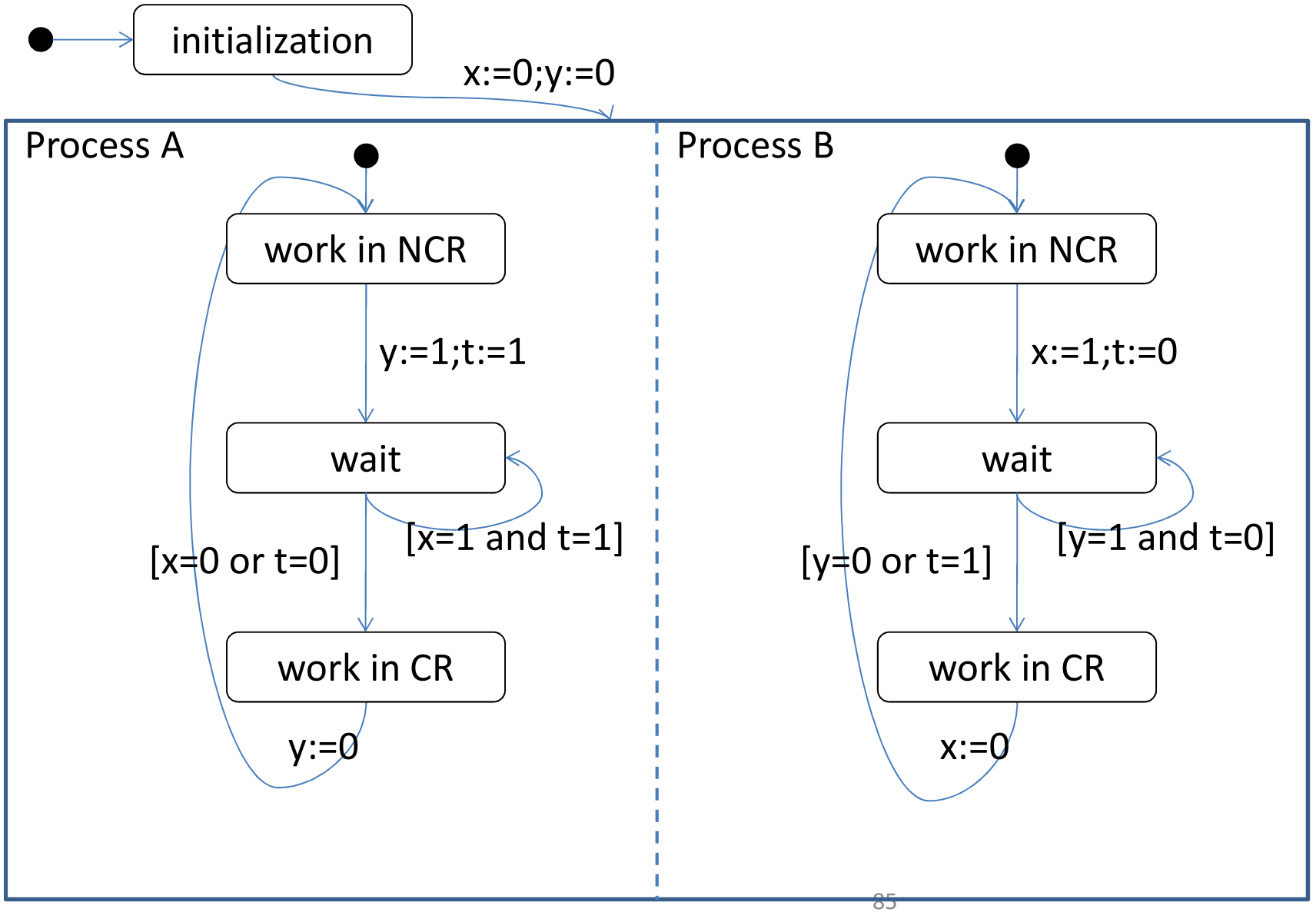
The Set of Initial States: I

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

The Set of Fairness Constraints: F

Apparently, we need fairness constraints:

例子-互斥：状态图(State Diagram)



The Set of Fairness Constraints: F

In fact, we may construct 6 fairness constraints for:

(a=NCR)
(a=wait and x=0|t=0)
(a=CR)

(b=NCR)
(b=wait and y=0|t=1)
(b=CR)

The Set of Fairness Constraints: F

S1 = { (a,b,x,y,t) | a=NCR }

S2 = { (a,b,x,y,t) | a=wait and x=0|t=0 }

wait,wait,1,1,0

S3 = { (a,b,x,y,t) | a=CR }

S4 = { (a,b,x,y,t) | b=NCR}

S5 = { (a,b,x,y,t) | b=wait and y=0|t=1 }

wait,wait,1,1,1

S6 = { (a,b,x,y,t) | b=CR }

The Set of Fairness Constraints: F

S1 = { (a,b,x,y,t) | a=NCR }

S2 = { (a,b,x,y,t) | a=wait and x=0|t=0 }

S3 = { (a,b,x,y,t) | a=CR }

wait,wait,1,1,0

S4 = { (a,b,x,y,t) | b=NCR}

S5 = { (a,b,x,y,t) | b=wait and y=0|t=1 }

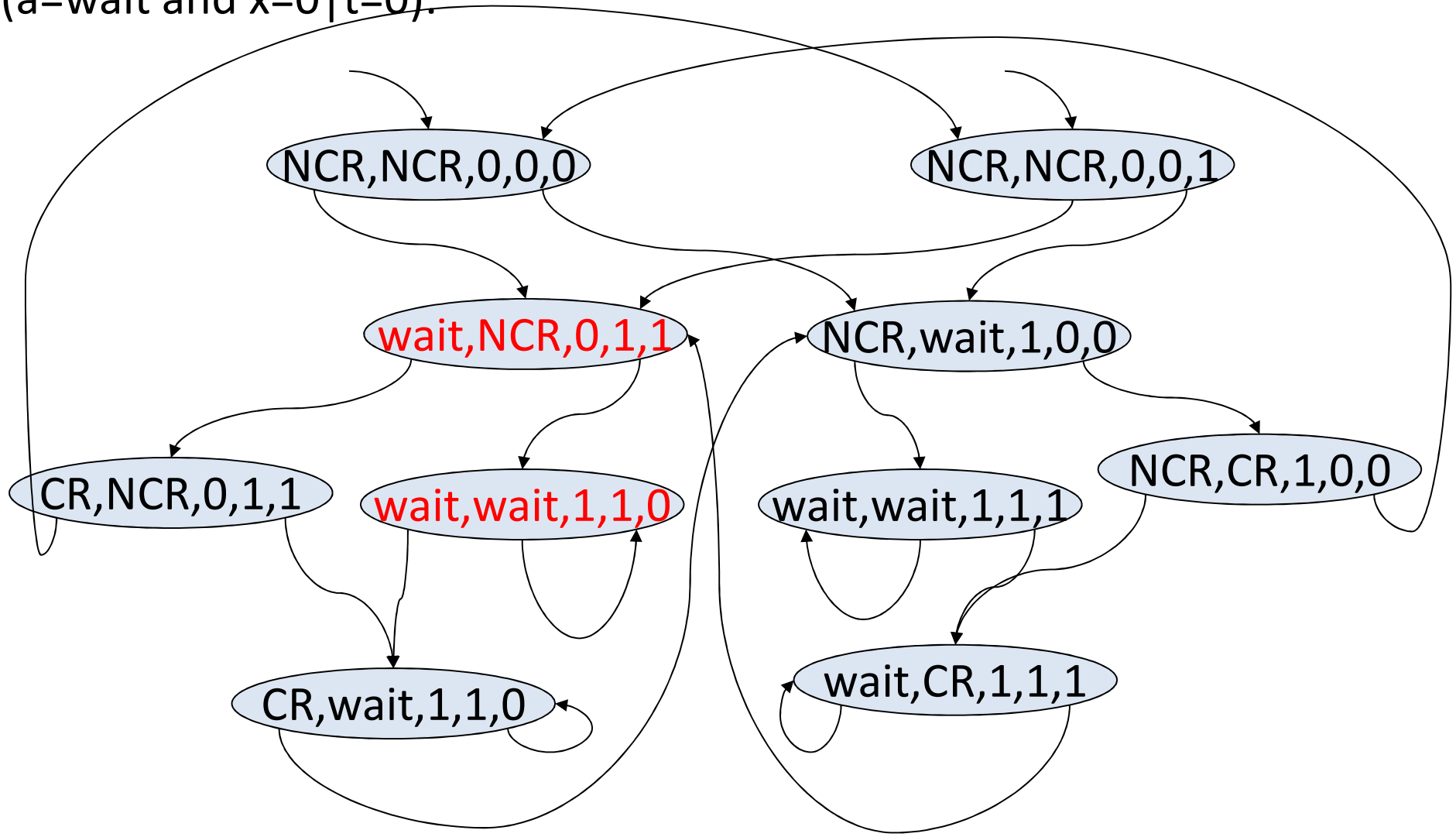
S6 = { (a,b,x,y,t) | b=CR }

wait,wait,1,1,1

F={ S\S1, S\S2, S\S3, S\S4, S\S5, S\S6 }

Example

(a=wait and x=0|t=0):



Example: Properties

- Emptiness
- Reachability & Safety
- Inevitability & Avoidability

Example: Emptiness

K = { S, R, I, F }

THEN

EmpChecking(K) = false

Example: Reachability & Safety

K = { S, R, I, F }

Y= { (a,b,x,y,t) | a!=CR or b!=CR }

THEN

FairSafetyAnalysis(K,Y) =true

Example: Inevitability & Avoidability

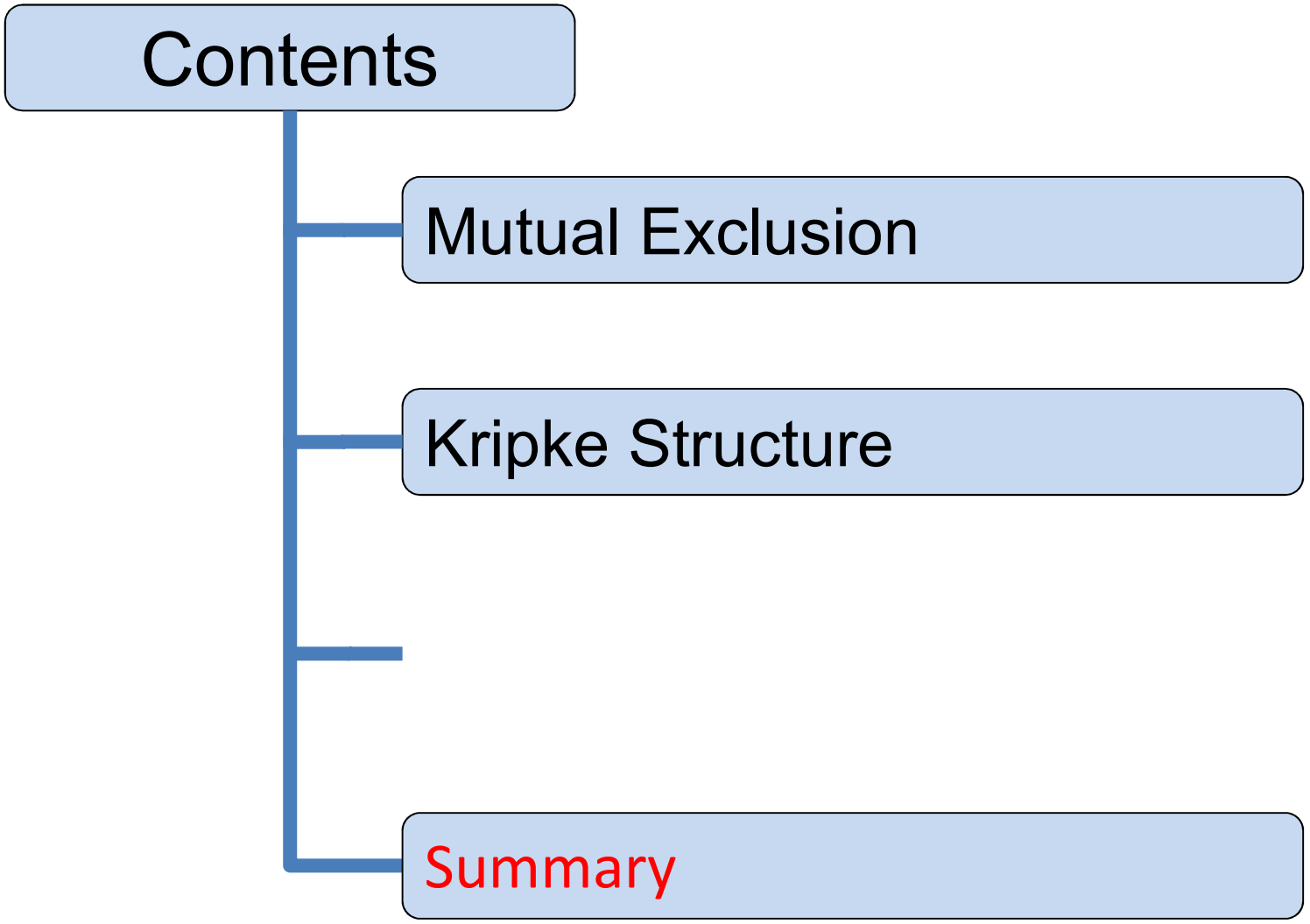
$K = \{ S, R, I, F \}$

$Y = \{ (a,b,x,y,t) \mid a=CR \text{ or } b=CR \}$

THEN

FairInevitabilityAnalysis(K,A) = true

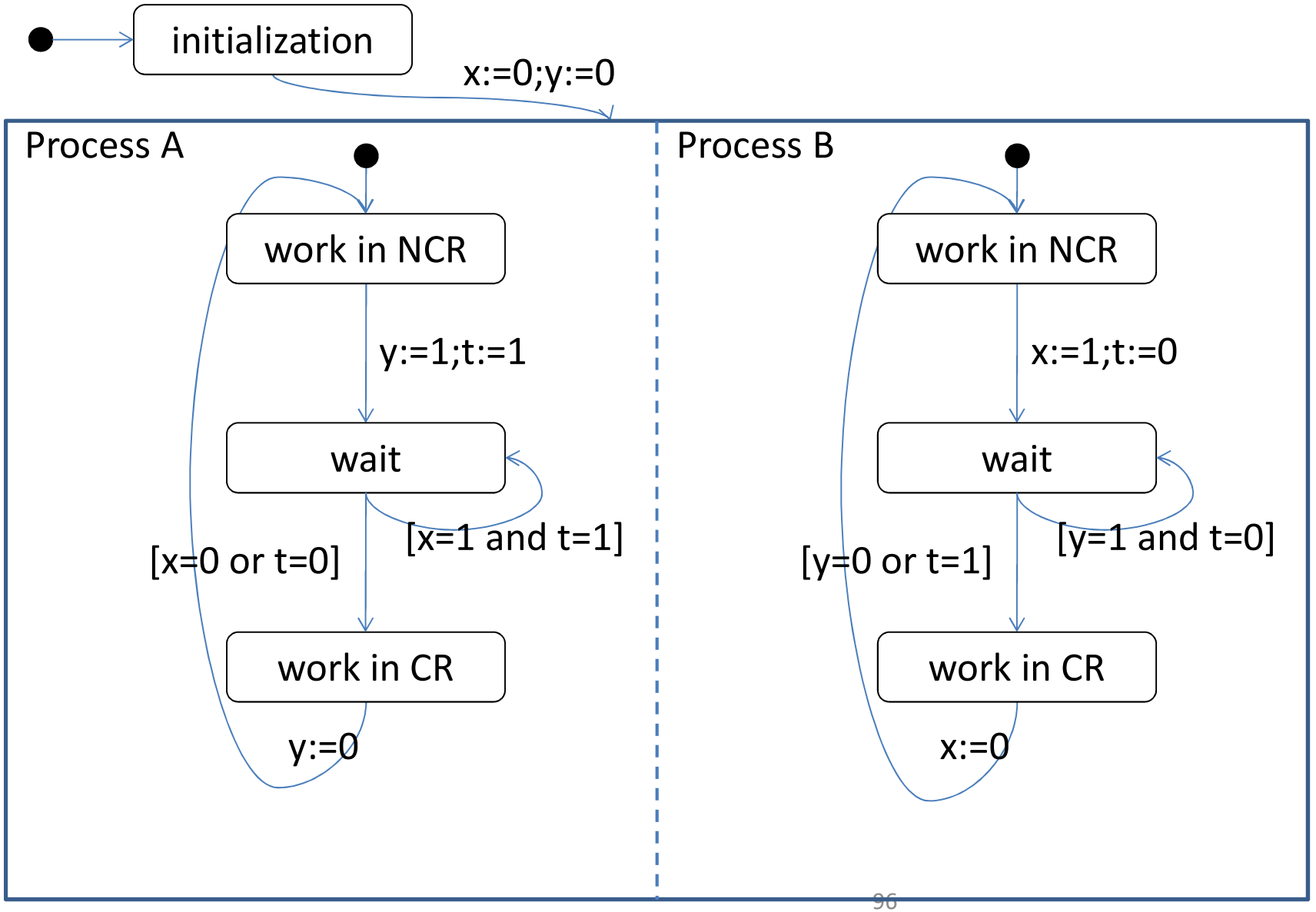
An Example



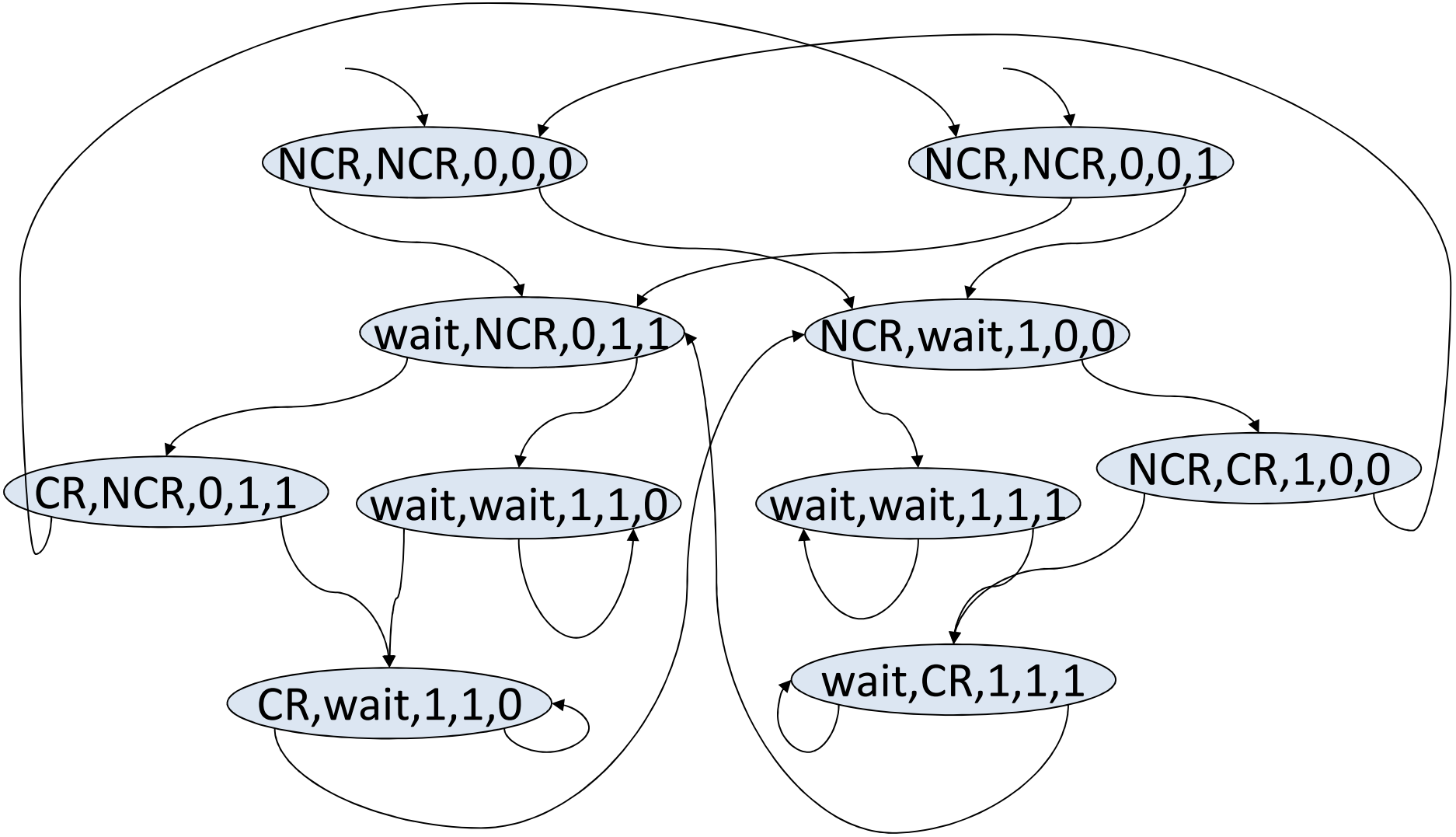
(II) Labeled Kripke Structures

- Motivation

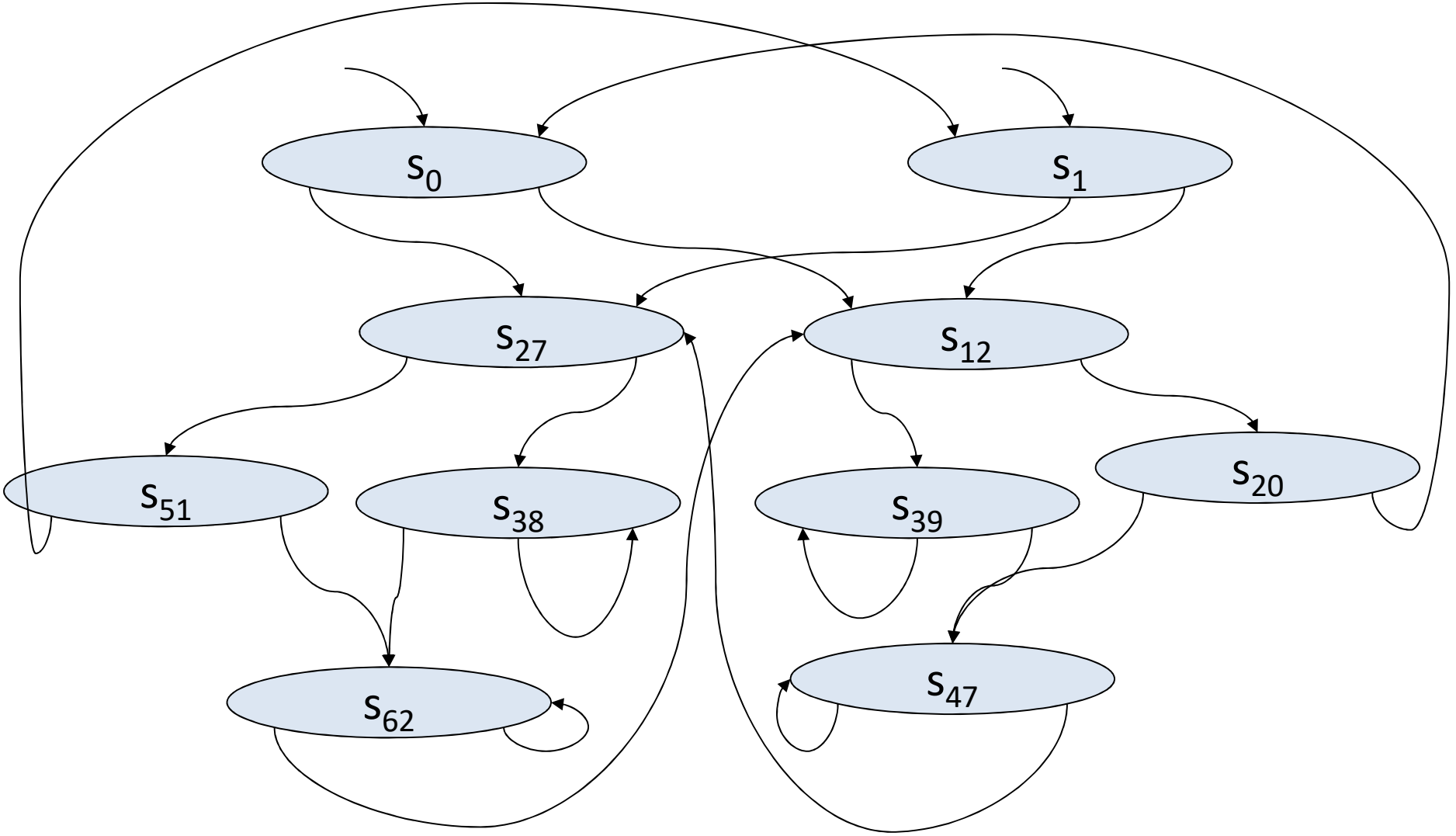
例子-互斥：状态图(State Diagram)



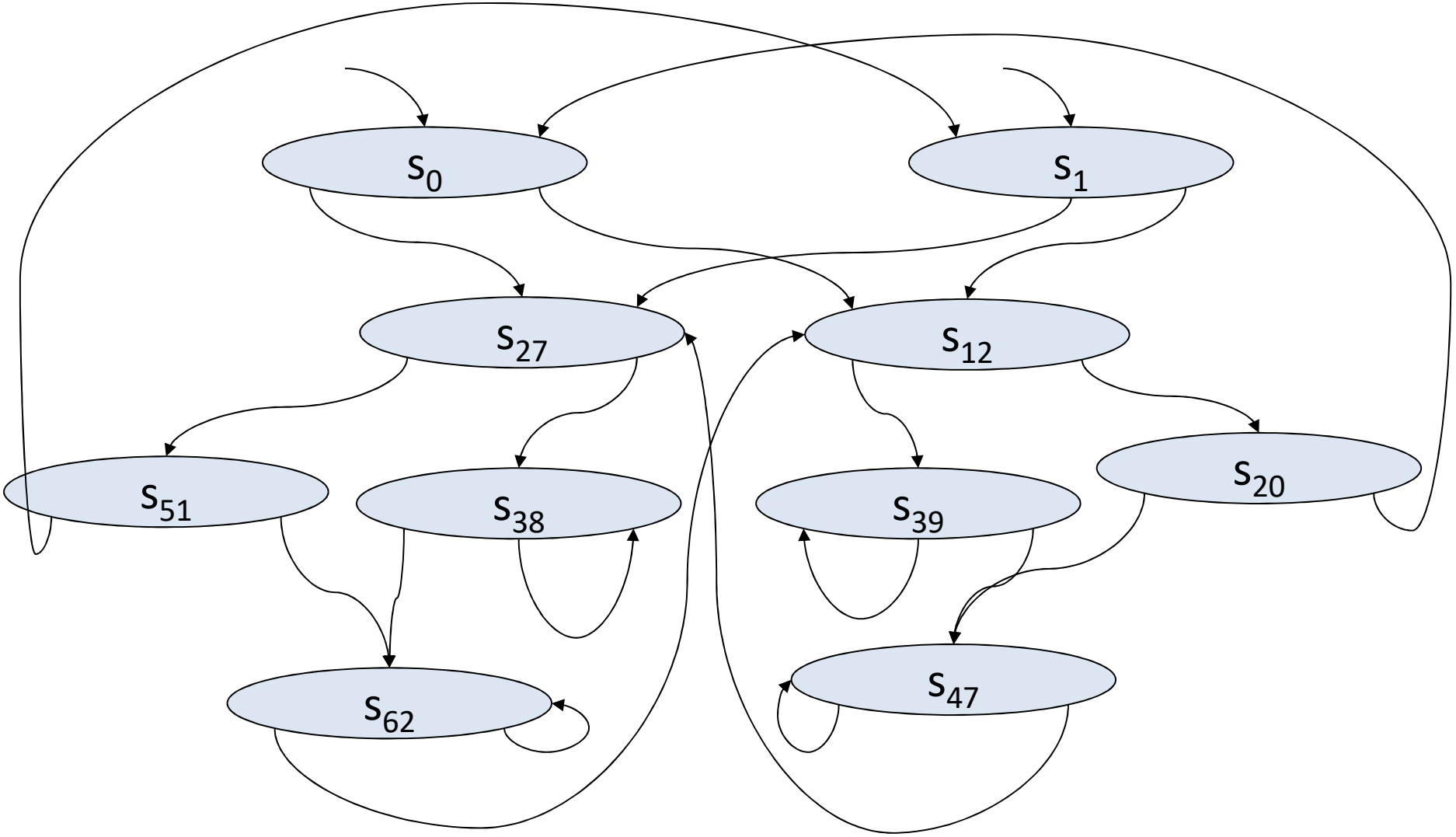
例子-互斥：状态迁移图(Kripke结构)



例子-互斥：状态迁移图(Kripke结构)



例子-互斥：状态迁移图(Kripke结构)



Good,Bad,Neutral?

Motivation

Distinguish states based on a set of basic properties (labels).

Labeled Kripke Structures

- Formulation
- Basic Concepts
- Basic System Properties

Labeled Kripke Structures

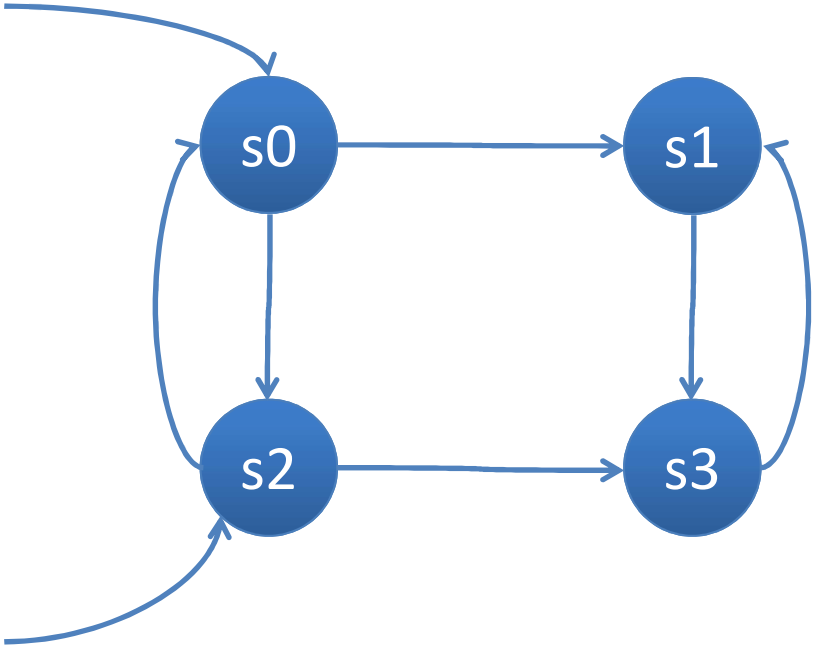
AP: a finite set of propositions.

Definition

A (Labeled) Kripke structure is a quadruple K=<S,R,I,L>

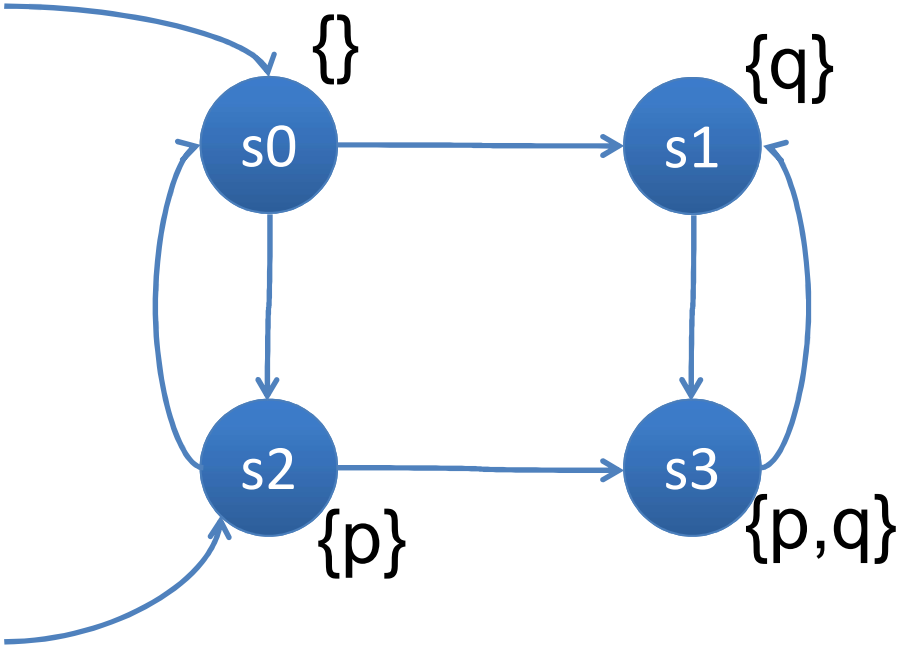
- S : A finite set of states
- $R \subseteq S \times S$: A total transition relation
- $I \subseteq S$: A set of initial states
- L: $S \rightarrow 2^{AP}$ is a labeling function

Example: S,R,I



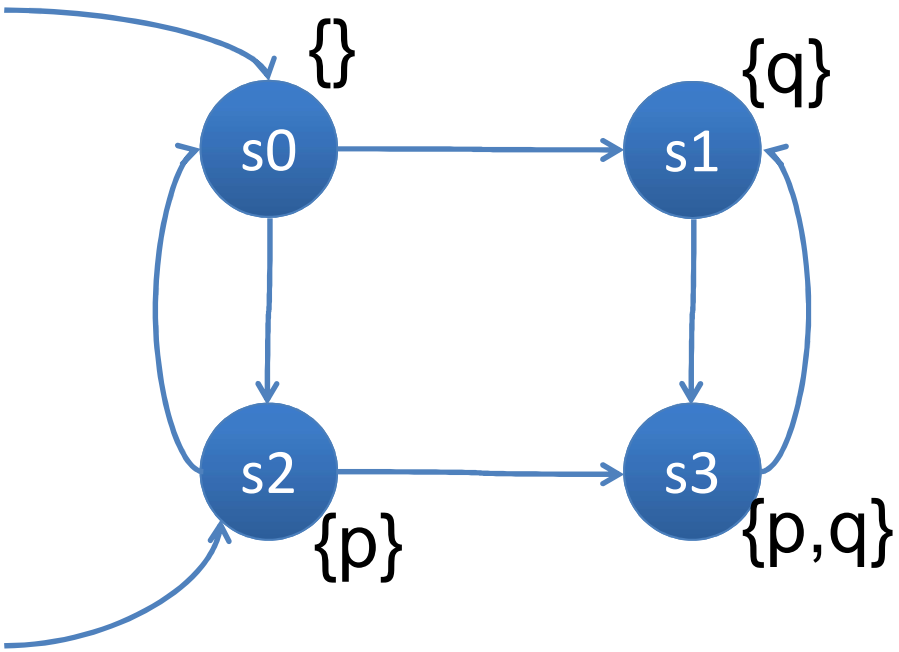
Example: L

AP: {p,q}



Example: Paths, Computations, Behavior

AP: {p,q}



$(s0s2)^{\omega}$

$s0(s2s0)^{*}(s1s3)^{\omega}$

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

$(s2s0)^{\omega}$

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

$s2s0(s2s0)^{*}(s1s3)^{\omega}$

Basic Concepts

- Basic Concepts (Kripke Structures)
  - States, Transition Relation, Initial States
  - Successors, Predecessors,
  - Reachable States, Reachability Relation,
  - Paths (Finite and Infinite), Computation, Behavior
  - Properties
- Basic Concepts (Labeled Kripke Structures)
  - Labels
  - Properties – Propositional Symbols – Formulas
  - Sets – Propositional Formulas

Labels

A proposition → a set of states: $S_p = \{ s \mid p \in L(s) \}$

A set of states → a proposition:

Given $X=\{x_1,x_2,\ldots,x_n\}$, we create a proposition px as follows.

Add px to AP,

Add px to $L(x_1)$, $L(x_2)$, …, $L(x_n)$

$X = S_{px} = \{ s \mid px \in L(s) \}$

Satisfiability Relation (State-based)

Let X,Y be sets of states. Let s be a state.

Definition

s satisfies Y (or s is a Y-state), if $s \in Y$,

Definition

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

Satisfiability with Formulas

Definition

Let φ be a formula.

$s \models \varphi$ is defined as follows.

- $s \models p$ if $p \in L(s)$
- $s \models \varphi \wedge \psi$ if $s \models \varphi$ and $s \models \psi$
- $s \models \varphi \vee \psi$ if $s \models \varphi$ or $s \models \psi$
- $s \models \neg\varphi$: if $s \not\models \varphi$

$X \models \varphi$, if for all $s \in X$, $s \models \varphi$

Correspondence: Formulas to Sets

Definition:

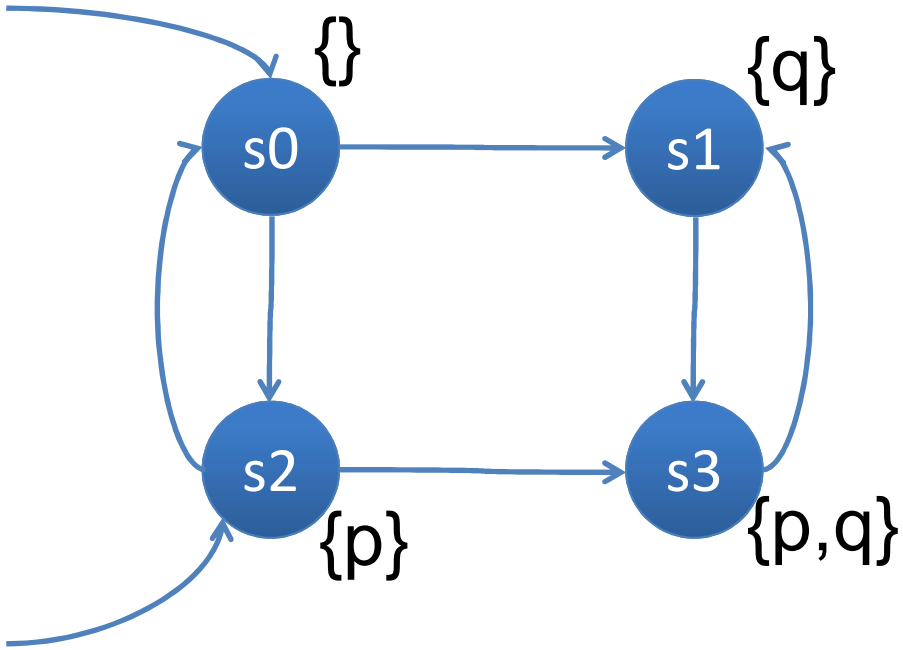
$[[\varphi]]$ is defined as follows.

- $[[p]] = \{ s \mid p \in L(s) \}$
- $[[\varphi \wedge \psi]] = [[\varphi]] \cap [[\psi]]$
- $[[\varphi \vee \psi]] = [[\varphi]] \cup [[\psi]]$
- $[[\neg\varphi]] = S \setminus [[\varphi]]$

Lemma: $s \models \varphi$ iff $s \in [[\varphi]]$, i.e., $[[\varphi]] = \{ s \mid s \models \varphi \}$.

Definition: s is a φ-state, if s is a $[[\varphi]]$-state.

Example: Correspondence



$[[p \wedge q]]=\{s3\}$, $[[p \vee q]]=\{s1,s2,s3\}$

Correspondence: Sets to Formulas

Construction

$[[\{s\}]] = (\wedge_{r \in L(s)} r) \wedge (\wedge_{r \notin L(s)} \neg r)$

$[[X \cup Y]] = [[X]] \vee [[Y]]$

Lemma:

if $s \in X$, then $s \models [[X]]$, i.e., $X \subseteq [[([[X]])]]$.

Abstraction

$X \models \varphi$ iff $[[X]] \rightarrow \varphi$

$X \subseteq Y \rightarrow [[X]] \rightarrow [[Y]]$

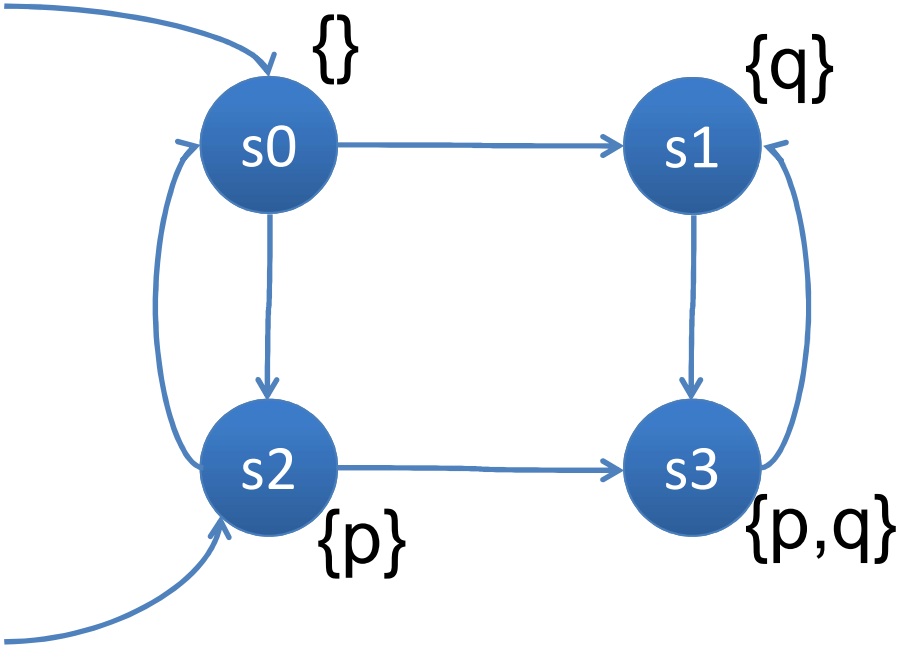
Correspondence: Sets to Formulas

Suppose that the function L: S → 2^{AP} is one-to-one.

Then for each $X \subseteq S$,
there is a formula characterizing the set X.

Lemma:
$s \in X$ iff s |= [[X]], i.e.,
X = [[([[X]])]].

Example: Correspondence



$[[\{s3\}]]=p \wedge q$, $[[\{s1,s2,s3\}]]=p \vee q$

Correspondence: non-one-to-one

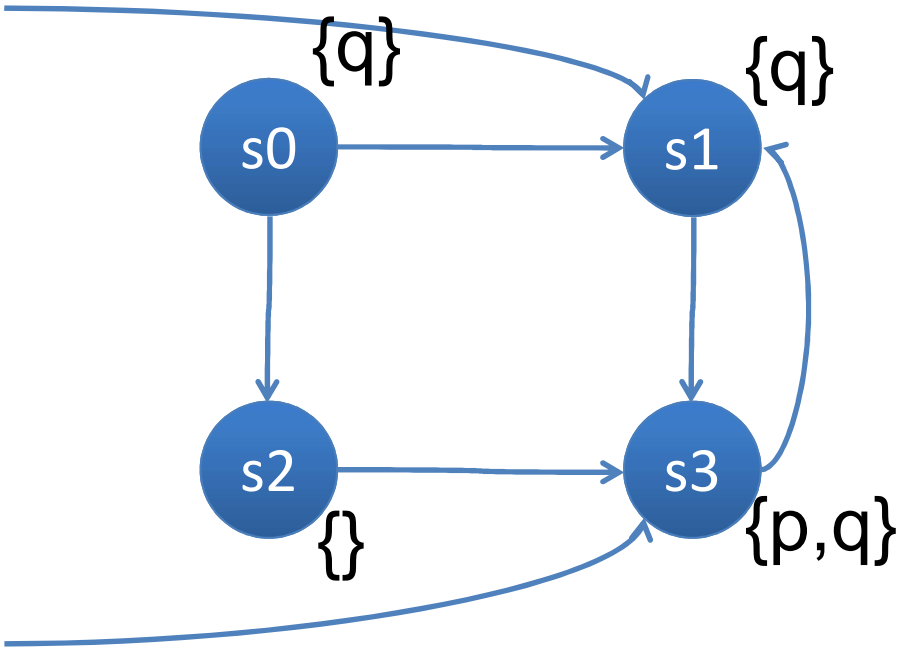
Suppose that the function L: S → 2^{AP} is not one-to-one (i.e., a kind of abstraction).

Then:

$s \in X \rightarrow s \models [[X]]$, i.e.,

$X \subseteq [[([[X]])]]$.

Example: L is not one-to-one



$[[\{s0\}]]=q\wedge\neg p$, $[[\{s1\}]]=q\wedge\neg p$, $[[q\wedge\neg p]]=\{s0, s1\}$

$[[\{s1, s3\}]]=q$, $[[\{s0, s1, s3\}]]=q$, $[[q]]=\{s0, s1, s3\}$

Basic Properties

- Reachability, Safety
- Avoidability, Inevitability

Basic Properties (1)

- Reachability, Safety

Reachability Property

Let K=<S,R,I,L> and φ be a formula.

φ is a reachability property, if
there is a computation of K that passes a φ-state.

Proposition

φ is a reachability property, iff
$[[\varphi]]$ is a reachability property of K'=<S,R,I>

Reachability Property

Let K=<S,R,I,L> and φ be a formula.

Proposition

φ is a reachability property,

iff

ReachabilityAnalysis(K',[[φ]]) = true

Safety Property

Let K=<S,R,I,L> and φ be a formula.

φ is a safety property, if

every state on every computation is a φ-state.

Proposition

φ is a safety property, iff

[[φ]] is a safety property of K'=<S,R,I>

Safety Property

Let K=<S,R,I,L> and φ be a formula.

Proposition

φ is a safety property,

iff

SafetyAnalysis(K',[[φ]]) = true

Q

Let K=<S,R,I,L> and φ be a formula.

How to develop a safety analysis algorithm with (K,φ) as input without calculating $[[\varphi]]$?

This problem is left to the readers.

Safety & Reachability

Safety is a dual property of reachability.

Proposition

φ is a safety property of K=<S,R,I,L>,

iff

$\neg\varphi$ is not reachable in K.

Deductive Safety Analysis

Successors

Define:

$R(\psi) = [[\{ s' \mid s \rightarrow s', s \models \psi \}]]$

Invariant

Definition

φ is a transition invariant, if $R(\varphi) \rightarrow \varphi$.

Inductive Invariant

Definition

φ is an inductive invariant, if

$[[I]] \rightarrow \varphi$, and φ is a transition invariant.

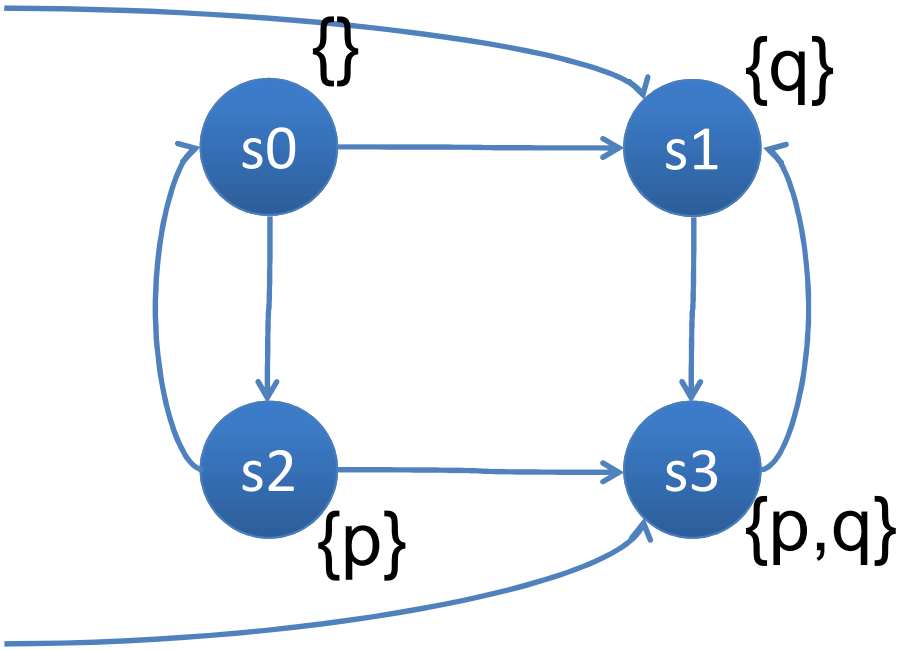
Proof of Safety

Given φ.

If there is a φ' such that
φ' is an inductive invariant and $\varphi' \rightarrow \varphi$,
then K is safe w.r.t. φ.

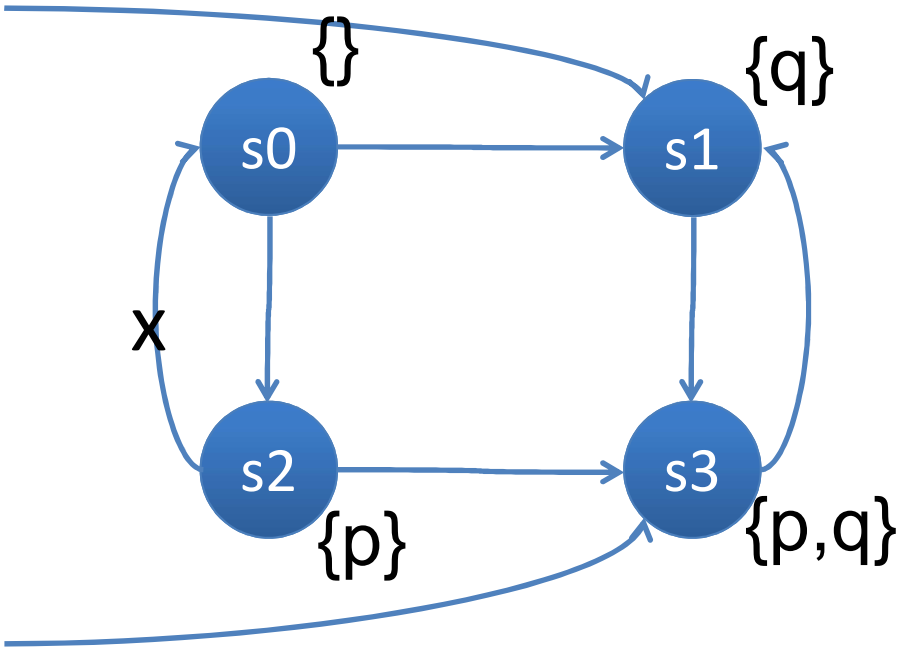
Relative Completeness
Suppose that L is a one-to-one mapping.
If the conclusion holds, then such a φ' exists.

Example: $\varphi = p \vee q = [[\{s1, s2, s3\}]]$



(1) We need φ' satisfying $([[I]] \rightarrow \varphi')$ and $(\varphi' \rightarrow \varphi)$ for the rule.
(2) We may choose $\varphi' = q$, which is transition invariant.

Example: $\varphi = p \vee q = [[\{s1, s2, s3\}]]$



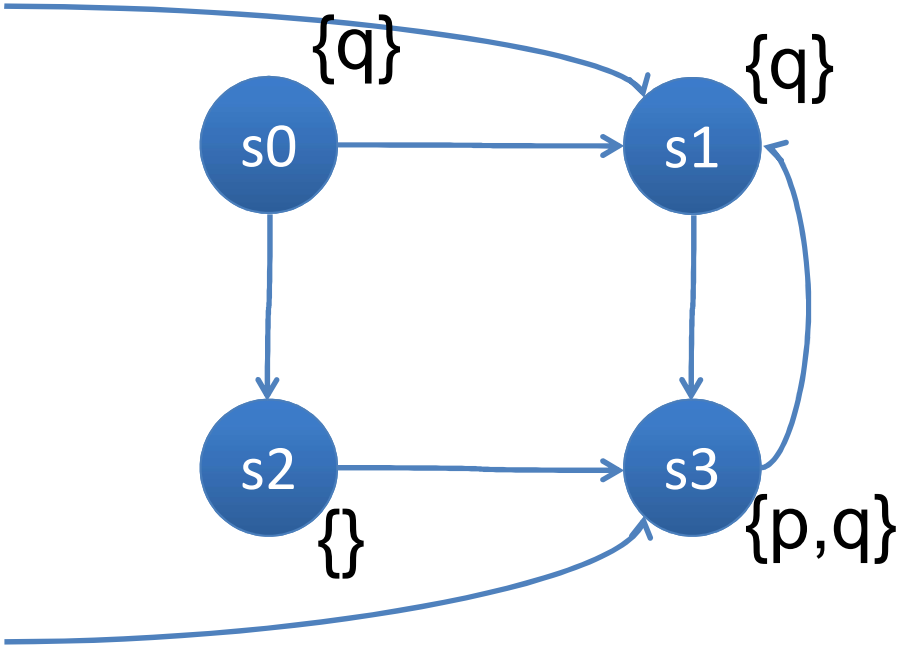
(1) We need φ' satisfying $([[I]] \rightarrow \varphi')$ and $(\varphi' \rightarrow \varphi)$ for the rule.
(2) We may choose $\varphi' = q$ or $\varphi' = p \vee q$, which are transition invariant.

Proof of Safety

Not complete, if L is not a one-to-one map.

Example: φ=q=[[{s0,s1,s3}]] is a safety property

(L is not one-to-one)



(1) We need φ' satisfying ([[I]] →φ') and (φ'→φ) for the rule.
(2) Since [[I]] = q = φ, φ' must be q.
(3) However q is not transition invariant.

Basic Properties (2)

- Avoidability, Inevitability

Avoidability Property

Let K=<S,R,I,L> and φ be a formula.

φ is an avoidability property, if
there exists some computation of K that
does not pass any φ-state.

Proposition

φ is an avoidability property, iff
$[[\varphi]]$ is an avoidability property of K'=<S,R,I>

Avoidability Property

Let K=<S,R,I,L> and φ be a formula.

Proposition:

φ is an avoidability property

iff

AvoidabilityAnalysis(K', [[φ]]) =true

Inevitability

Let K=<S,R,I,L> and φ be a formula.

Definition

φ is an inevitability property, if

every computation passes a φ-state

Proposition

φ is an inevitability property of K, iff

$[[\varphi]]$ is an inevitability property of K'=<S,R,I>.

Inevitability

Let K=<S,R,I,L> and φ be a formula.

Proposition:

φ is an inevitability property

iff

InevitabilityAnalysis(K', [[φ]]) =true

Inevitability & Avoidability

Inevitability is the negation of avoidability.

Proposition

φ is an inevitability property of K,

iff

φ is not avoidable in K.

Q

Let K=<S,R,I,L> and φ be a formula.

Is it easy (or possible) to develop
an inevitability analysis algorithm
with (K,φ) as input without calculating [[φ]] ?

This problem is left to the readers.

Deductive Inevitability Analysis

Proof of Inevitability

If there is a sequence of sets of states:

$\varphi_0, \varphi_1, \ldots, \varphi_n$ such that

$[[I]] \rightarrow \varphi_0$,

$R(\varphi_i \wedge \neg\varphi) \rightarrow \varphi_{i+1}$, for i=0,1,…,n-1

$\varphi_n \rightarrow \varphi$,

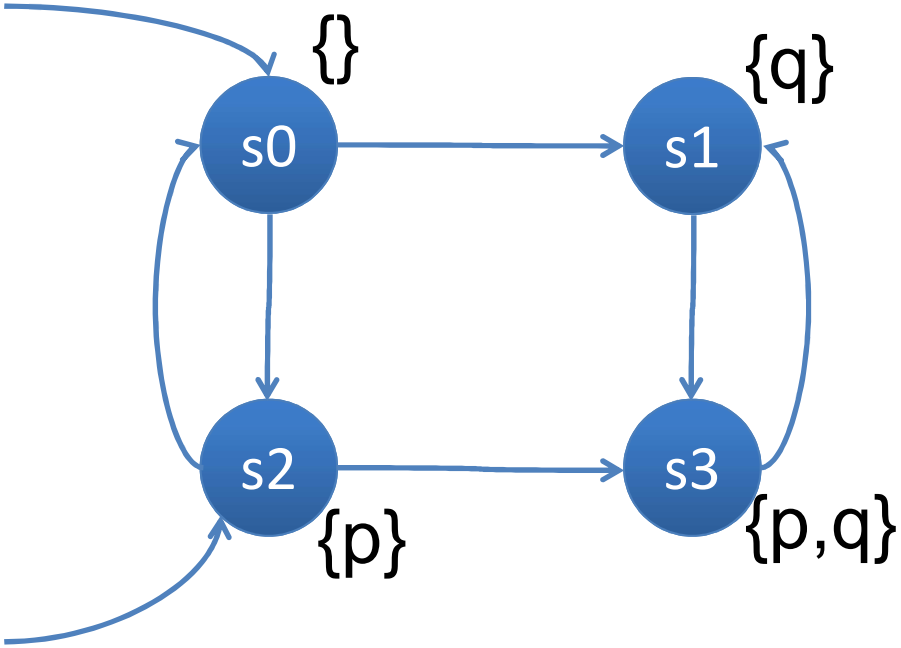
Then φ is an inevitability property.

Relative Completeness

Suppose that L is a one-to-one mapping.

For finite state systems, if the conclusion holds, then such a sequence exists.

Example: φ=p={s2,s3}



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

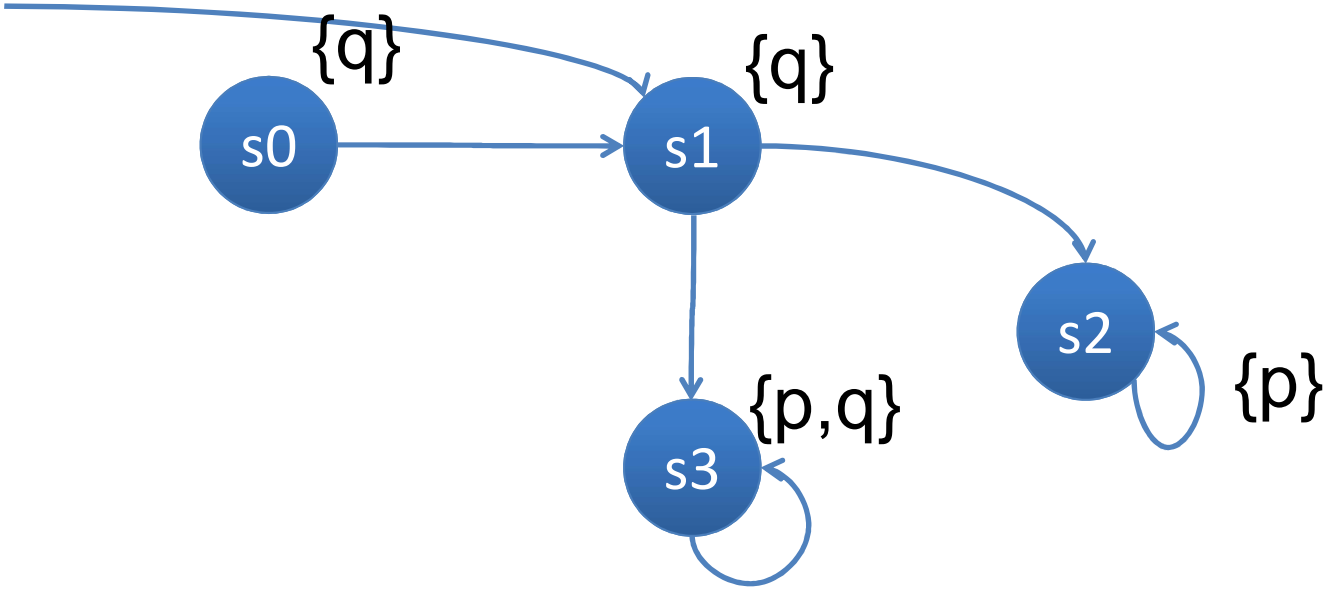
$\varphi_0 = \neg q$, $\varphi_1 = (p \wedge \neg q) \vee (q \wedge \neg p)$, $\varphi_2 = p \wedge q$

Proof of Inevitability

Not complete, if L is not a one-to-one map.

Example: φ=p={s2,s3} is an inevitability property

(L is not one-to-one)



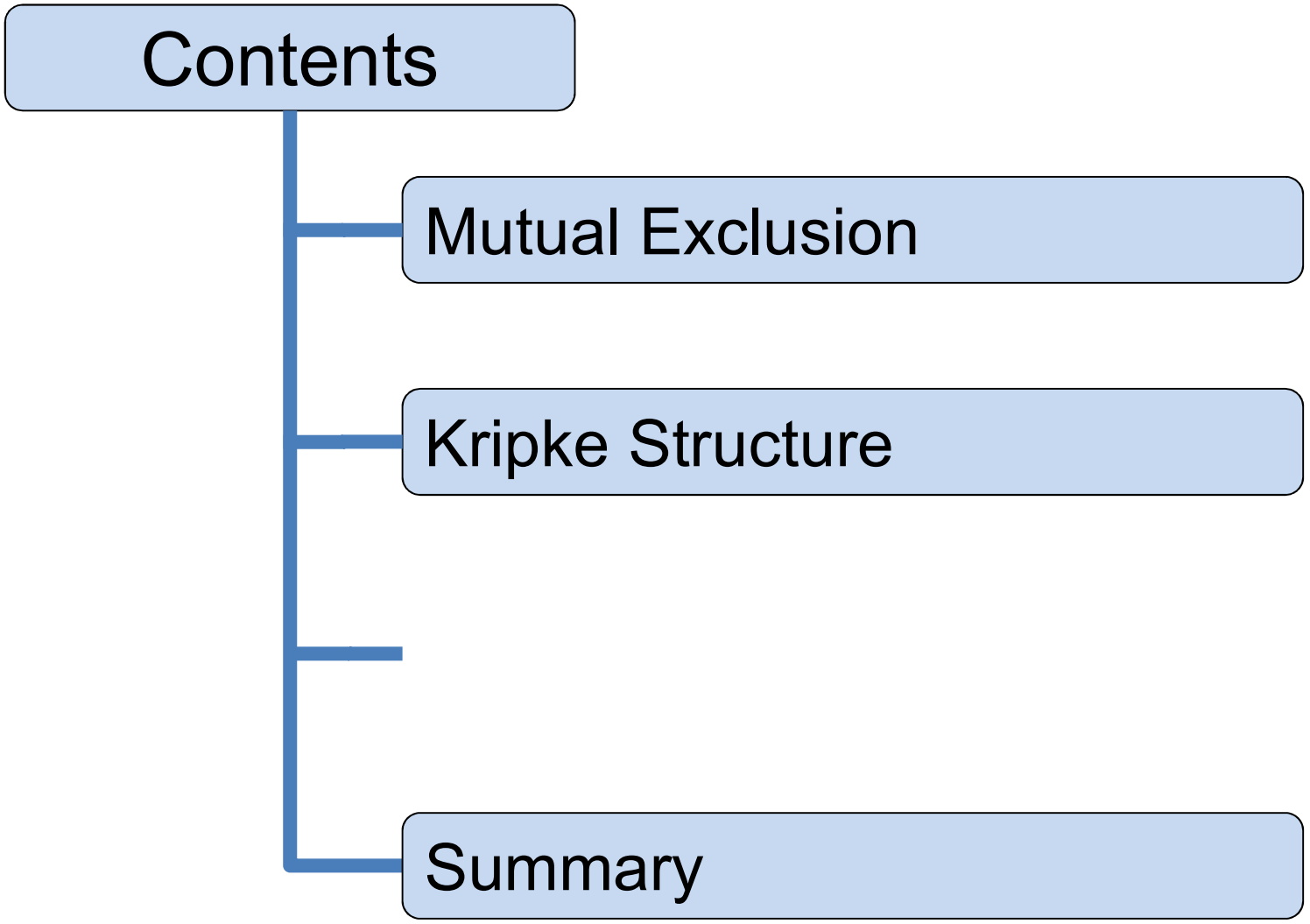
We have [[I]] = (q∧¬p).

(q∧¬p) → φ_0,

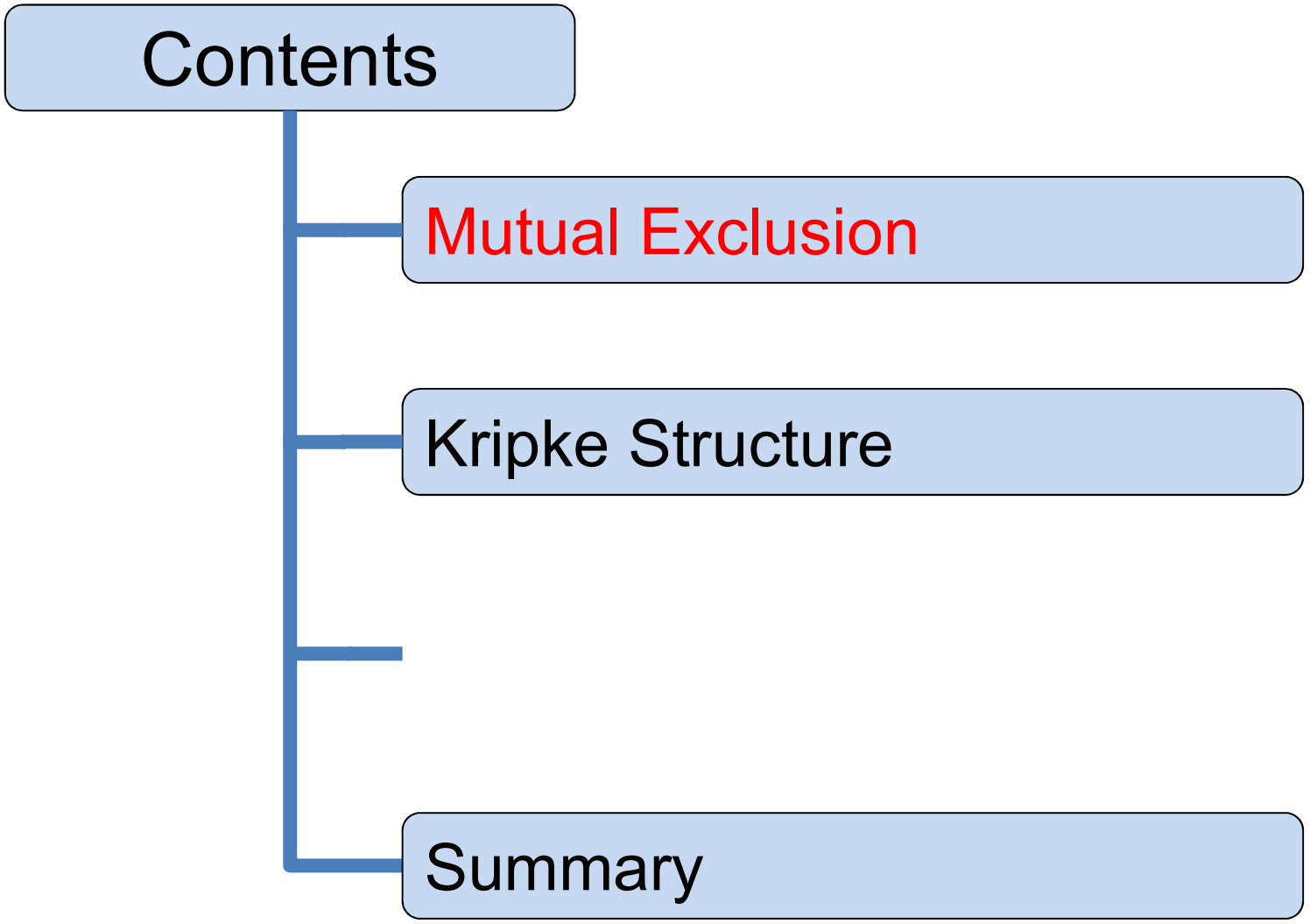
(q∧¬p)∨(p) → φ_1,

(q∧¬p)∨(p) → φ_2, ...

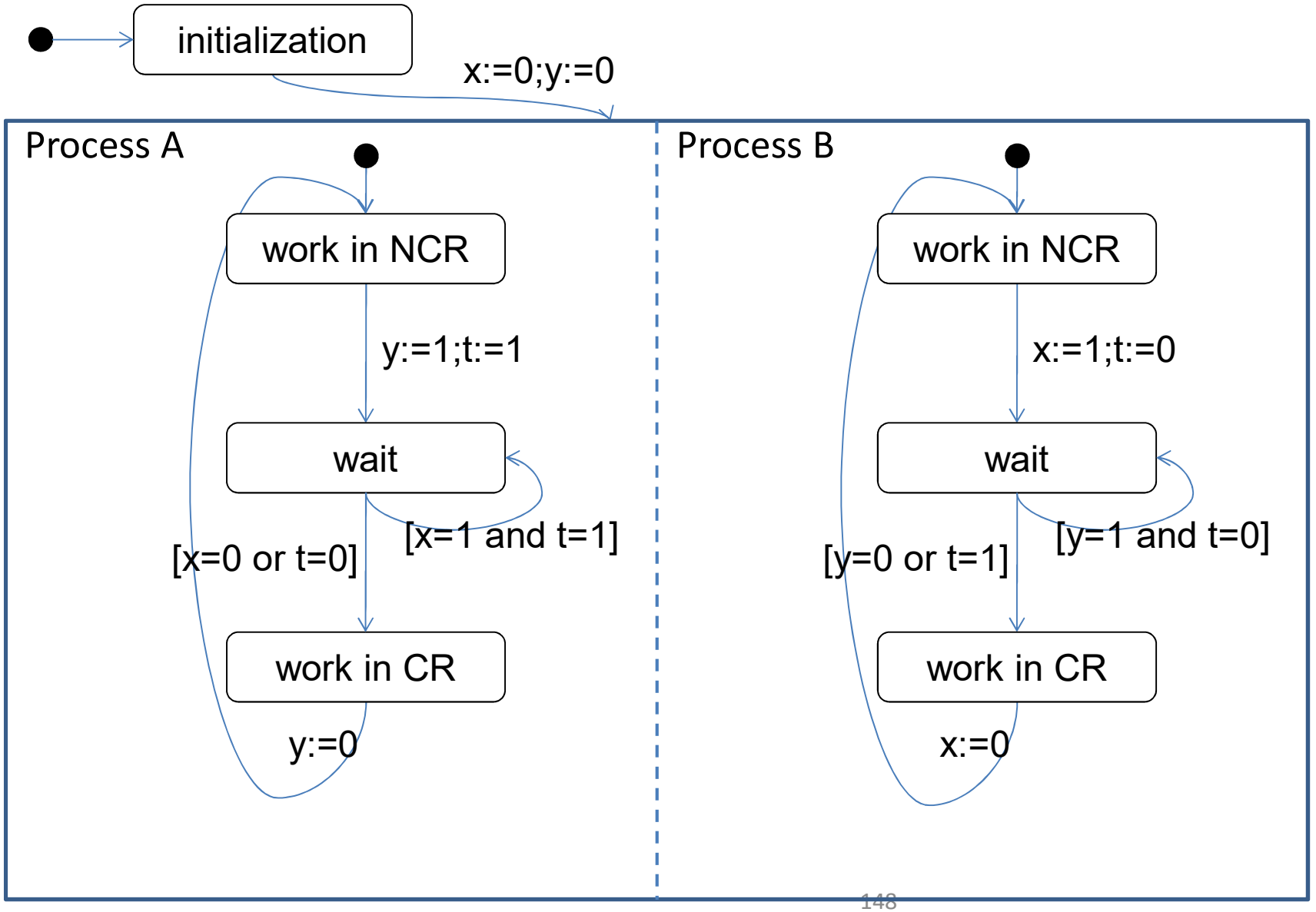
An Example



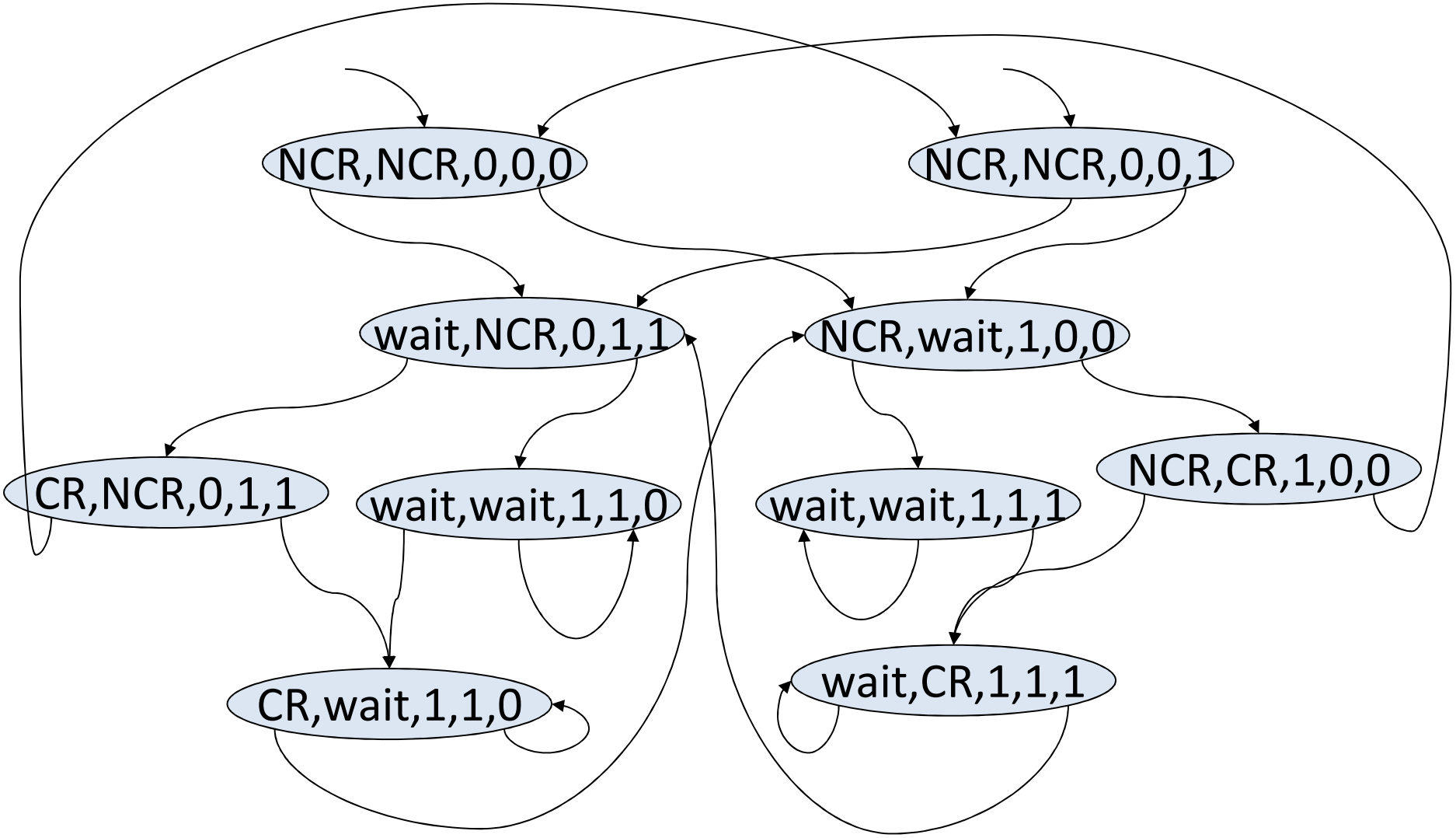
An Example



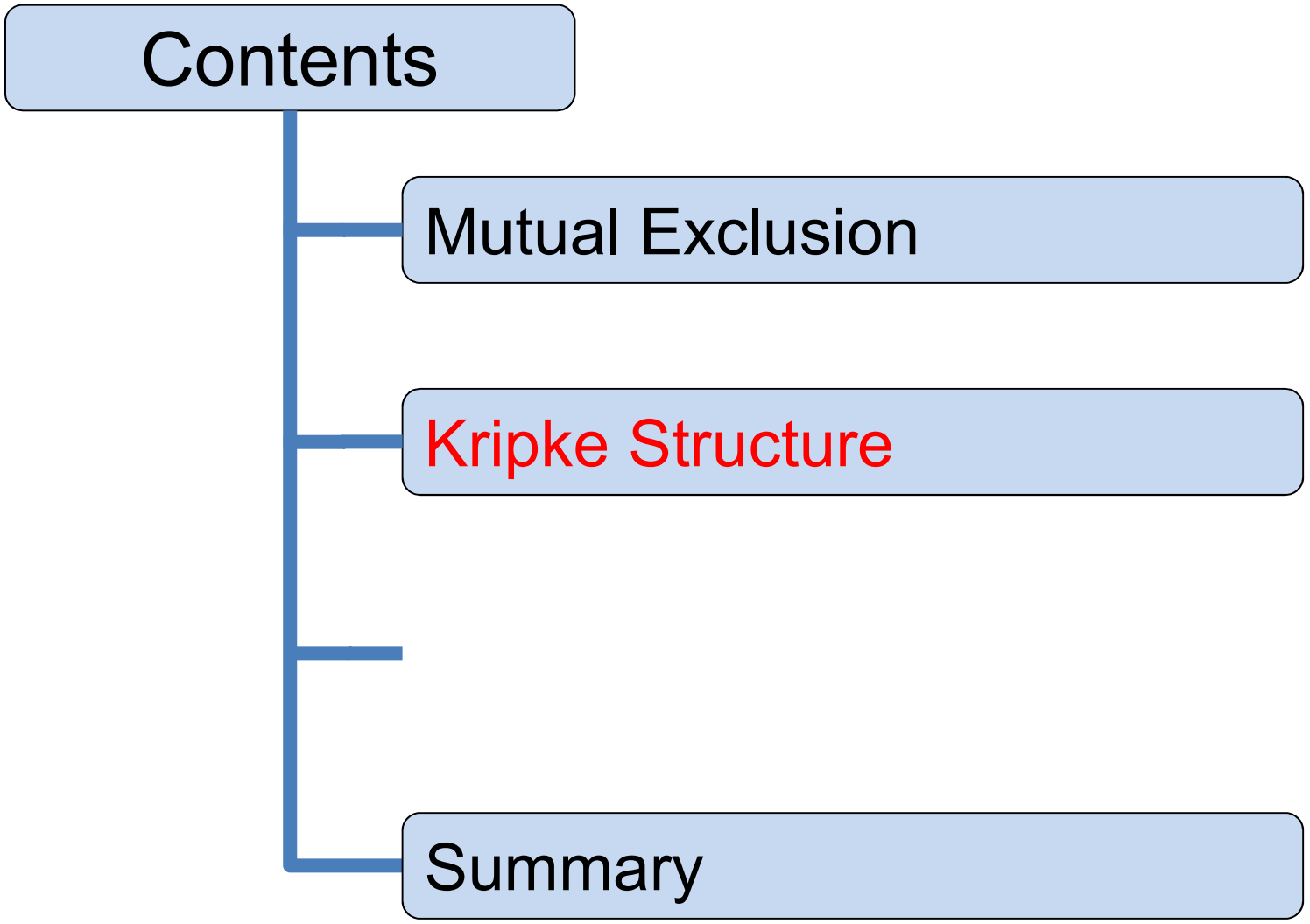
Design of Mutual Exclusion (State)



Example


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

An Example



Labeled Kripke Structures

AP: a finite set of propositions.

Definition

A (Labeled) Kripke structure is a quadruple K=<S,R,I,L>

- S : A finite set of states
- $R \subseteq S \times S$: A total transition relation
- $I \subseteq S$: A set of initial states
- L: $S \rightarrow 2^{AP}$ is a labeling function

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,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)

The Set of Initial States: I

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

Labeling Function

AP={

a=NCR,a=wait,a=CR,b=NCR,b=wait,b=CR,

x=1,x=0,y=1,y=0,t=1,t=0

}

L(NCR,NCR,0,0,0) = {a=NCR,b=NCR,x=0,y=0,t=0}

L(NCR,NCR,0,0,1) = {a=NCR,b=NCR,x=0,y=0,t=1}

…

Labeling Function

AP={ p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12 }

L(NCR,NCR,0,0,0) = {p1,p4,p7,p9,p11}
L(NCR,NCR,0,0,1) = {p1,p4,p7,p9,p12}
…

L(NCR,NCR,0,0,0) = {a=NCR,b=NCR,x=0,y=0,t=0}
L(NCR,NCR,0,0,1) = {a=NCR,b=NCR,x=0,y=0,t=1}
…

Labeling Function (with Abstraction)

OR

AP={ a=NCR,a=wait,a=CR,b=NCR,b=wait,b=CR }

L(NCR,NCR,x,y,t) = {a=NCR,b=NCR}

L(NCR,wait,x,y,t) = {a=NCR,b=wait}

L(NCR,CR,x,y,t) = {a=NCR,b=CR}

…

Labeling Function (with Abstraction)

OR

AP={ p1,p2,p3,p4,p5,p6 }

L(NCR,NCR,x,y,t) = {p1,p4}

L(NCR,wait,x,y,t) = {p1,p5}

L(NCR,CR,x,y,t) = {p1,p6}

…

Reachability & Safety

AP = {p1,p2,p3,p4,p5,p6 }

K = { S, R, I, L }

$\phi = \neg(p3 \wedge p6)$

THEN

SafetyAnalysis(K,[[ϕ]]) =true

Inevitability & Avoidability

AP = {p1,p2,p3,p4,p5,p6 }

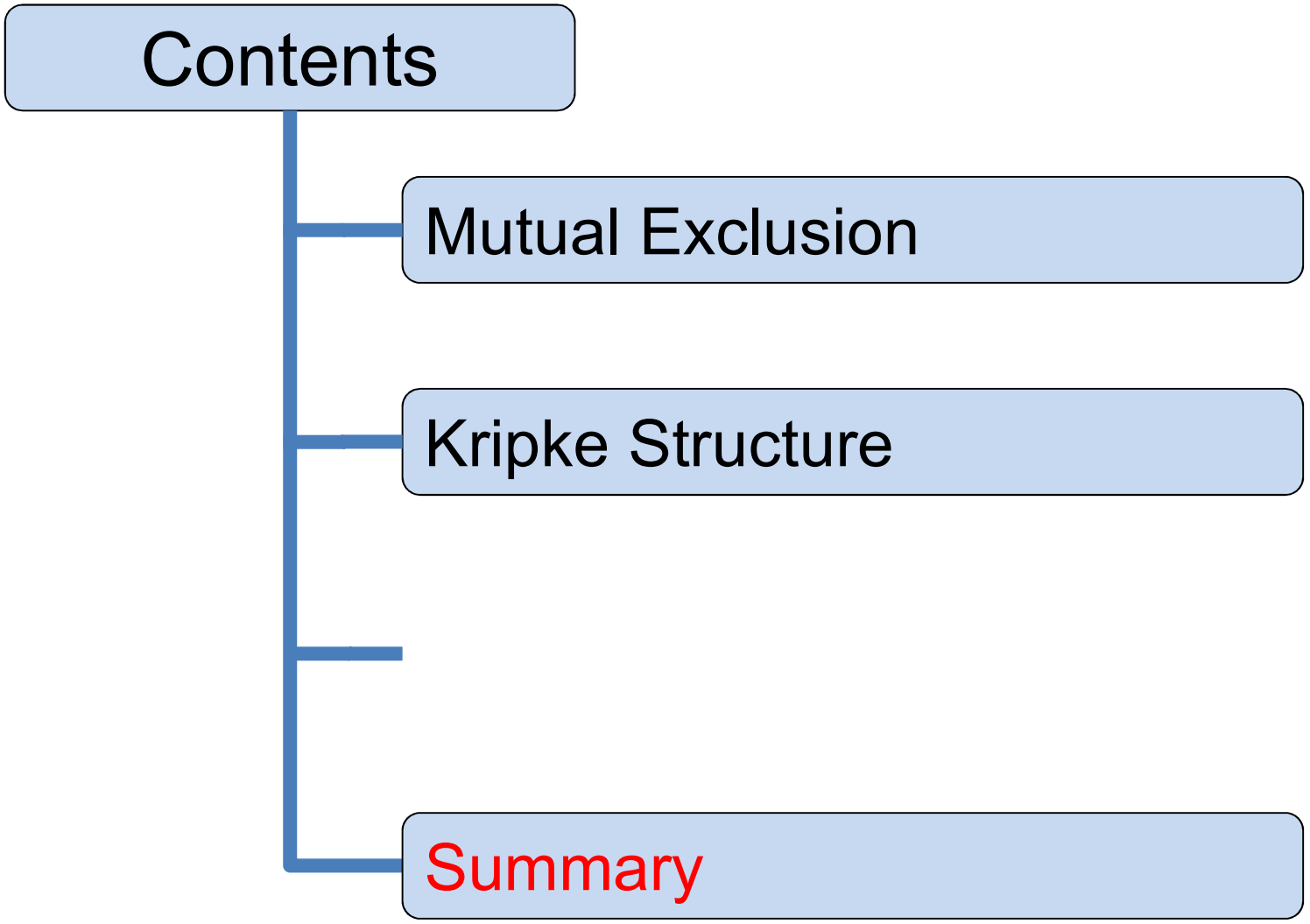
K = { S, R, I, L }

ϕ = (p3 $\vee$ p6 }

THEN

InevitabilityAnalysis(K,[[ϕ]]) = false

An Example



(III) Fair Labeled Kripke Structures

A fair labeled KS

=

a labeled KS + a fair KS

Fair Labeled Kripke Structures

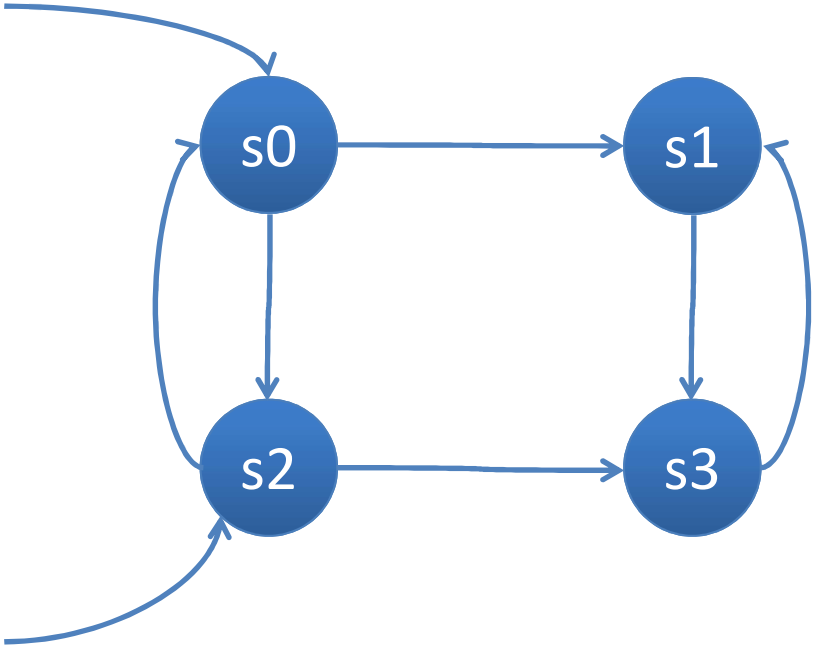
AP: A set of propositions.

Definition

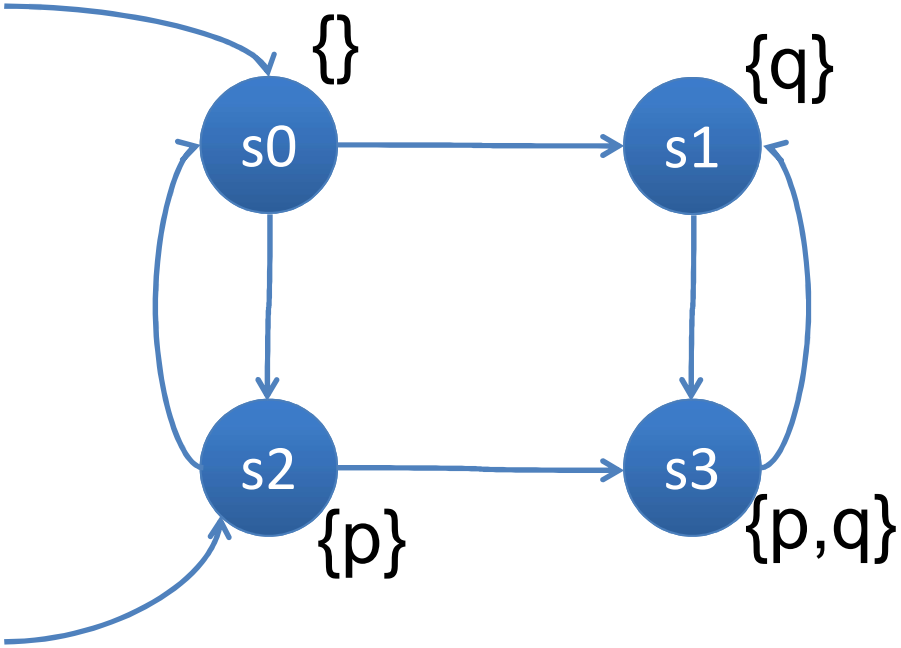
A fair Kripke structure is a quintuple <S,R,I,L,F>

- S : A finite set of states
- $R \subseteq S \times S$: A total transition relation
- $I \subseteq S$: A set of initial states
- L: $S \rightarrow 2^{AP}$ is a labeling function
- F: A set of formulas over AP

Example: S,R,I

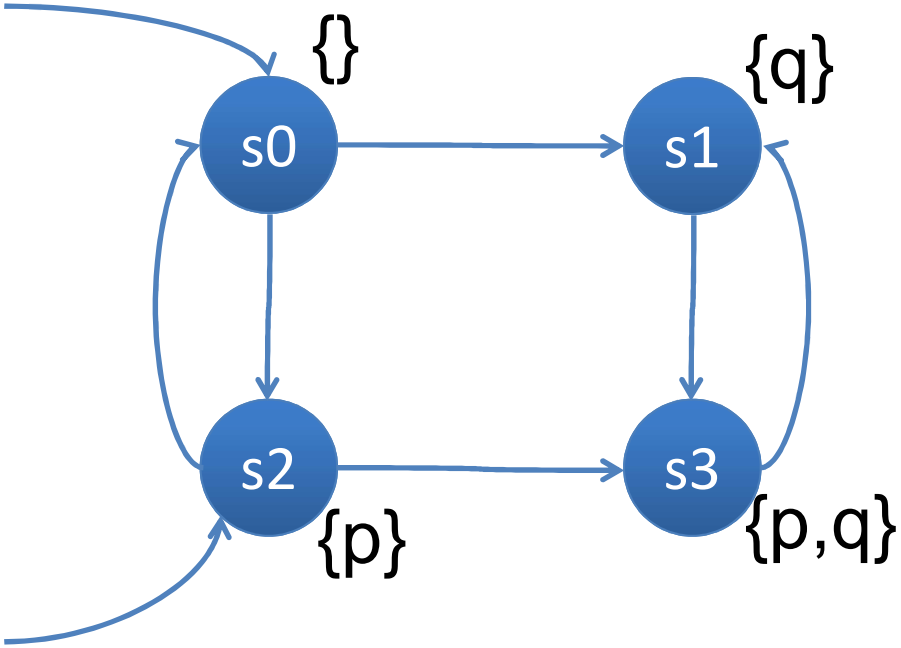


Example: L



AP: {p,q}

Example: F={p,¬q}



AP: {p,q}

Basic Concepts

- Basic Concepts (Kripke Structures)
  - States, Transition Relation, Initial States
  - Successors, Predecessors,
  - Reachable States, Reachability Relation,
  - Paths (Finite and Infinite), Computation, Behavior
  - Properties
- Basic Concepts (Fair Kripke Structures)
  - Paths – Fair Paths
  - Computations – Fair Computations
  - Behavior - Language
  - States – Fair States

Fair Labeled Kripke Structures

F = { f1,…,fn }, a set of propositional formulas

<S,R,I,L,F>

=

a labeled KS + a fair KS

<S,R,I,L> + <S,R,I,{[[f1]],…,[[fn]]}>

Paths

Definition

An infinite path is an infinite sequence of S:

$s_0\, s_1\, s_2 \ldots.$

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:

$s_0 \ldots s_n$

Fair Paths

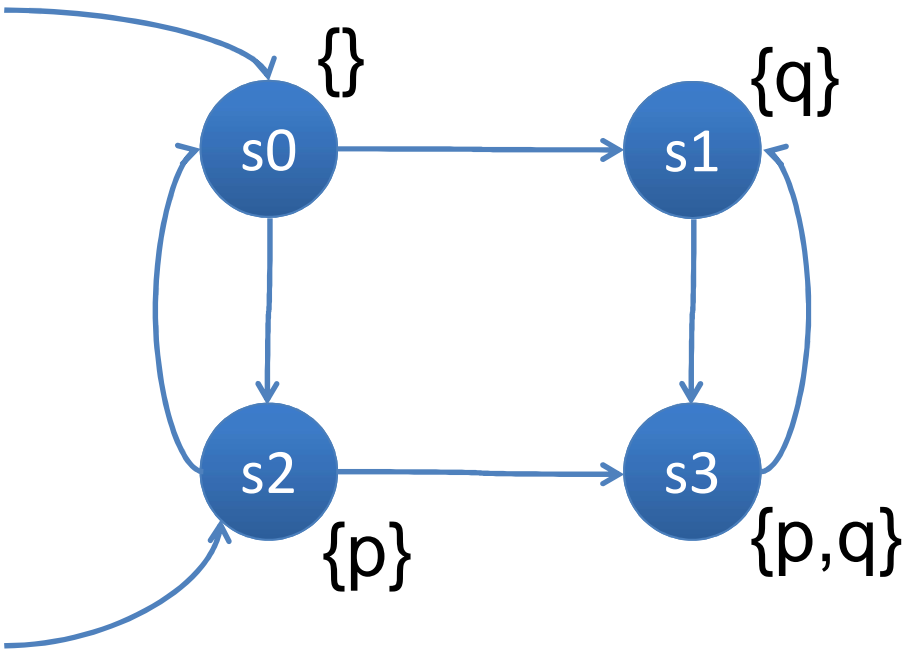
Definition

A fair (infinite) path is a path such that

for every f $\in$ F, there is an infinite number of states

on the path satisfying f.

Example: Paths



$F=\{p,\neg q\}$

Infinite paths:

s0s1s3s1s3s1…

$s0(s1s3)^{\omega}$

s1s3s1s3s1s3s…

Finite paths:

s0s1s3s1

s1s3s1s3s1

Computations

Given a Kripke structure K=<S,R,I>.

Definition

A **computation** of K is an infinite sequence of S:

$s_0\, s_1\, s_2 \ldots$

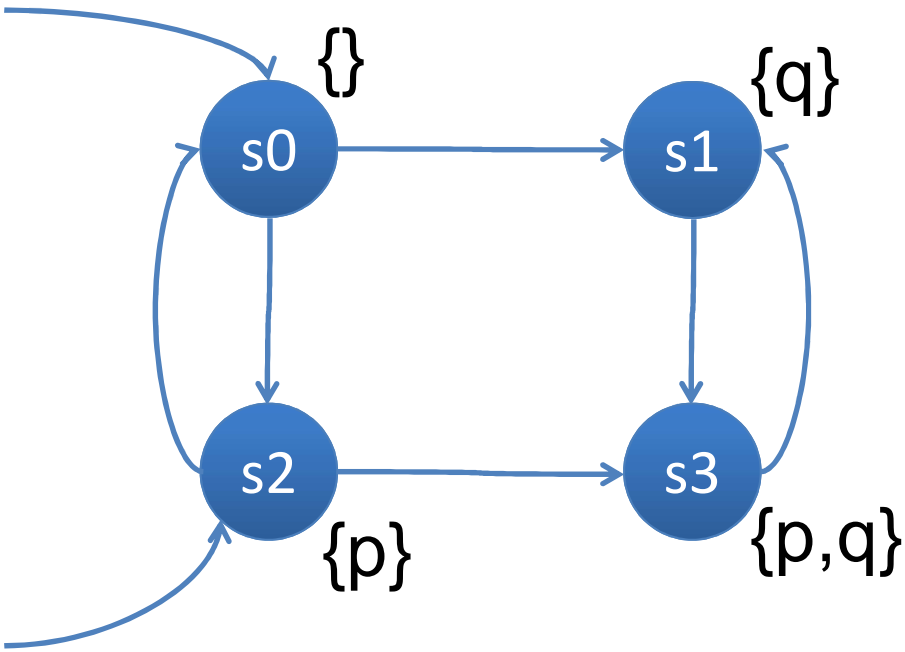
such that $s_0 \in I$, and $s_i \rightarrow s_{i+1}$ for all $i \geq 0$

Fair Computations

Definition

A **fair computation** is a computation such that for every $f \in F$, there is an infinite number of states on the computation satisfying f.

Example: Computations



$F=\{p, \neg q\}$

$s0(s1s3)^{\omega}$

$s2(s3s1)^{\omega}$

$s2(s0s2)^{*}(s1s3)^{\omega}$

$[\![p]\!] = \{s2,s3\}$

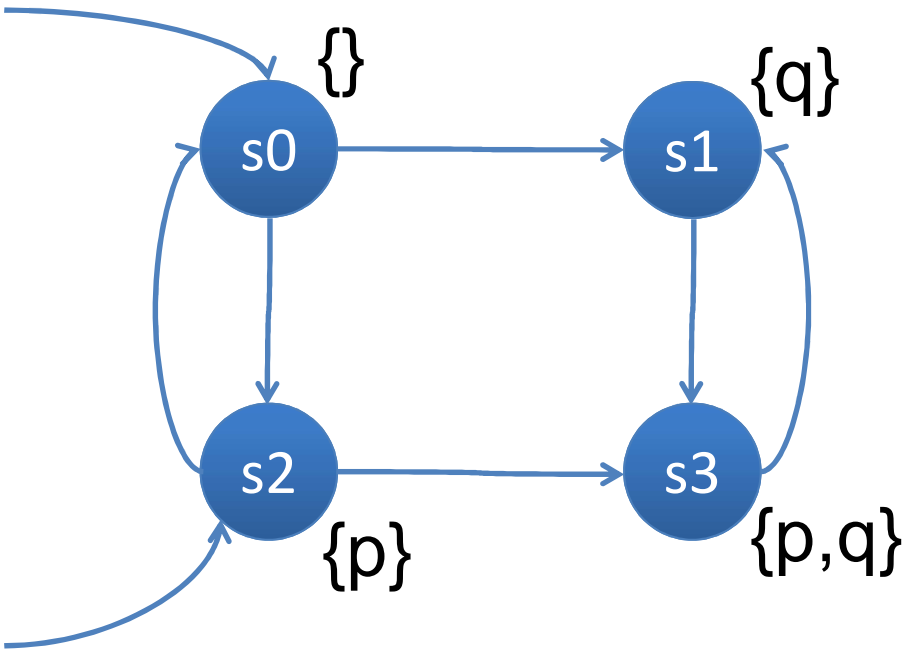
$[\![\neg q]\!] = \{s0,s2\}$

System Behavior

System behavior = the set of fair computations

[[K]]

Example: Behavior



$F=\{p,\neg q\}$

$(s0s2)^{\omega}$

$s0(s2s0)^{*}(s1s3)^{\omega}$

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

$(s2s0)^{\omega}$

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

$s2s0(s2s0)^{*}(s1s3)^{\omega}$

Language

Given AP.

Let K=<S,R,I,L,F> .

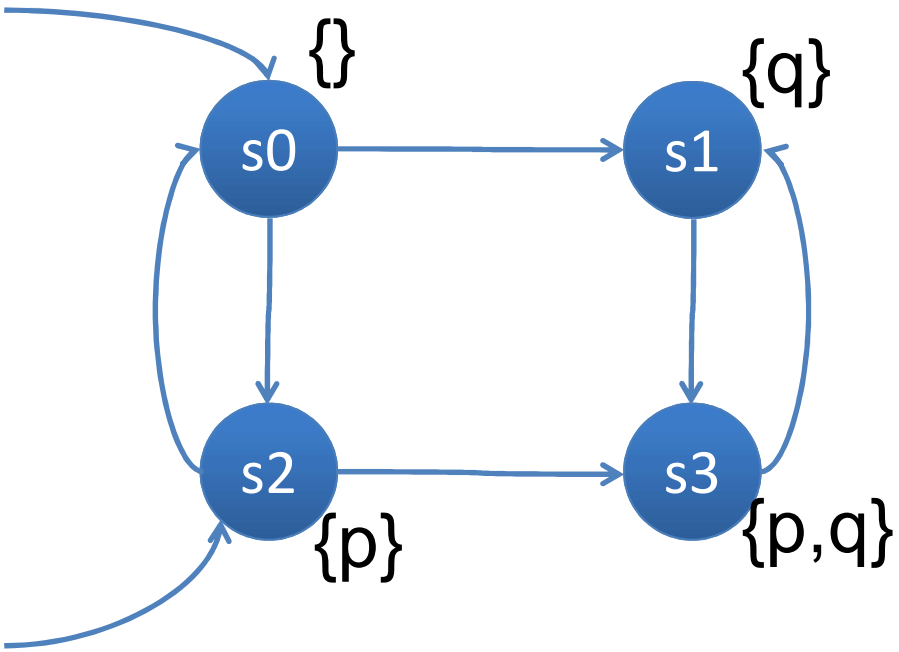
$L(K) \subseteq (2^{AP})^{\omega}$

L(s0s1s2……) = L(s0)L(s1)L(s2) ……

$L(K) = \{ L(\pi) \mid \pi \in [[K]] \}$

Example: Language

AP: {p,q}



F={p, ¬q}

$(s0s2)^{\omega}$

$(s2s0)^{\omega}$

$(\{\}\ \{p\})^{\omega}$

$(\{p\}\ \{\})^{\omega}$

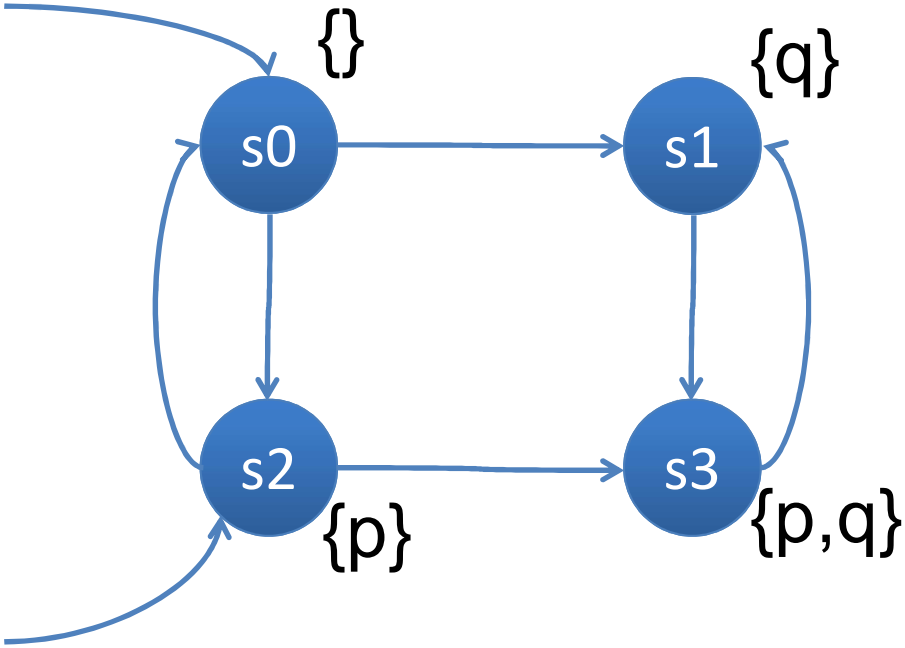
Fair States

Definition

A fair state is a starting state of some fair path.

Example: Fair States

$F=\{p,\neg q\}$



Basic Properties

- Fair Reachability, Fair Safety
- Fair Avoidability, Fair Inevitability
- Emptiness

Basic Properties (1)

- Fair Reachability, Fair Safety

Fair Reachability Property

Let K=<S,R,I,L,F> and φ be a formula.

φ is a fair reachability property, if
there is a fair computation of K that passes an φ-state.

Define [[F]] = { [[f]] | f $\in$ F }

Proposition

φ is a fair reachability property, iff
[[φ]] is a fair reachability property of K'=<S,R,I,[[F]]>

Fair Reachability Property

Let K=<S,R,I,L,F> and φ be a formula.

Proposition:

Let K'=<S,R,I,[[F]]>.

φ is a fair reachability property of K

iff

FairReachabilityAnalysis(K',[[φ]]) =true

Fair Safety Property

Let K=<S,R,I,L,F> and φ be a formula.

φ is a fair safety property, if
every state on every fair computation is a φ-states.

Proposition

φ is a fair safety property of K, iff
$[[\varphi]]$ is a fair safety property of K'=<S,R,I,[[F]]>

Fair Safety Property

Let K=<S,R,I,L,F> and φ be a formula.

Proposition:

Let K'=<S,R,I,[[F]]>.

φ is a fair safety property of K

iff

FairSafetyAnalysis(K',[[φ]]) =true

Reachability & Safety

Fair safety is a dual property of fair reachability.

Proposition

φ is a fair safety property of K=<S,R,I,L,F>, iff

$\neg\varphi$ is not a fair reachability property.

Basic Properties (2)

- Fair Avoidability, Fair Inevitability

Fair Avoidability Property

Let K=<S,R,I,L,F> and φ be a formula.

φ is a fair avoidability property, if
there exists some fair computation of K that
does not pass any φ-states.

Proposition

φ is a fair avoidability property, iff
[[φ]] is a fair avoidability property of K'=<S,R,I,[[F]]>

Fair Avoidability Property

Let K=<S,R,I,L,F> and φ be a formula.

Proposition

Let K'=<S,R,I,[[F]]>.

φ is a fair avoidability property of K,

iff

FairAvoidabilityAnalysis(K',[[φ]]) =true

Fair Inevitability

Let K=<S,R,I,L,F> and φ be a formula.

φ is a fair inevitability property, if
every fair computation passes a φ-state

Proposition

φ is a fair inevitability property of K, iff
$[[\varphi]]$ is a fair inevitability property of K'=<S,R,I,[[F]]>.

Fair Inevitability

Let K=<S,R,I,L,F> and φ be a formula.

Proposition

Let K'=<S,R,I,[[F]]>.

φ is a fair inevitability property of K,

iff

FairInevitabilityAnalysis(K',[[φ]]) =true

Avoidability & Inevitability

fair inevitability is a negation of fair avoidability.

Proposition

φ is a fair inevitability property of K, iff

φ is not a fair avoidability property.

Basic Properties (3)

- Emptiness

Emptiness

Let K=<S,R,I,L,F> .

L(K) is empty, iff

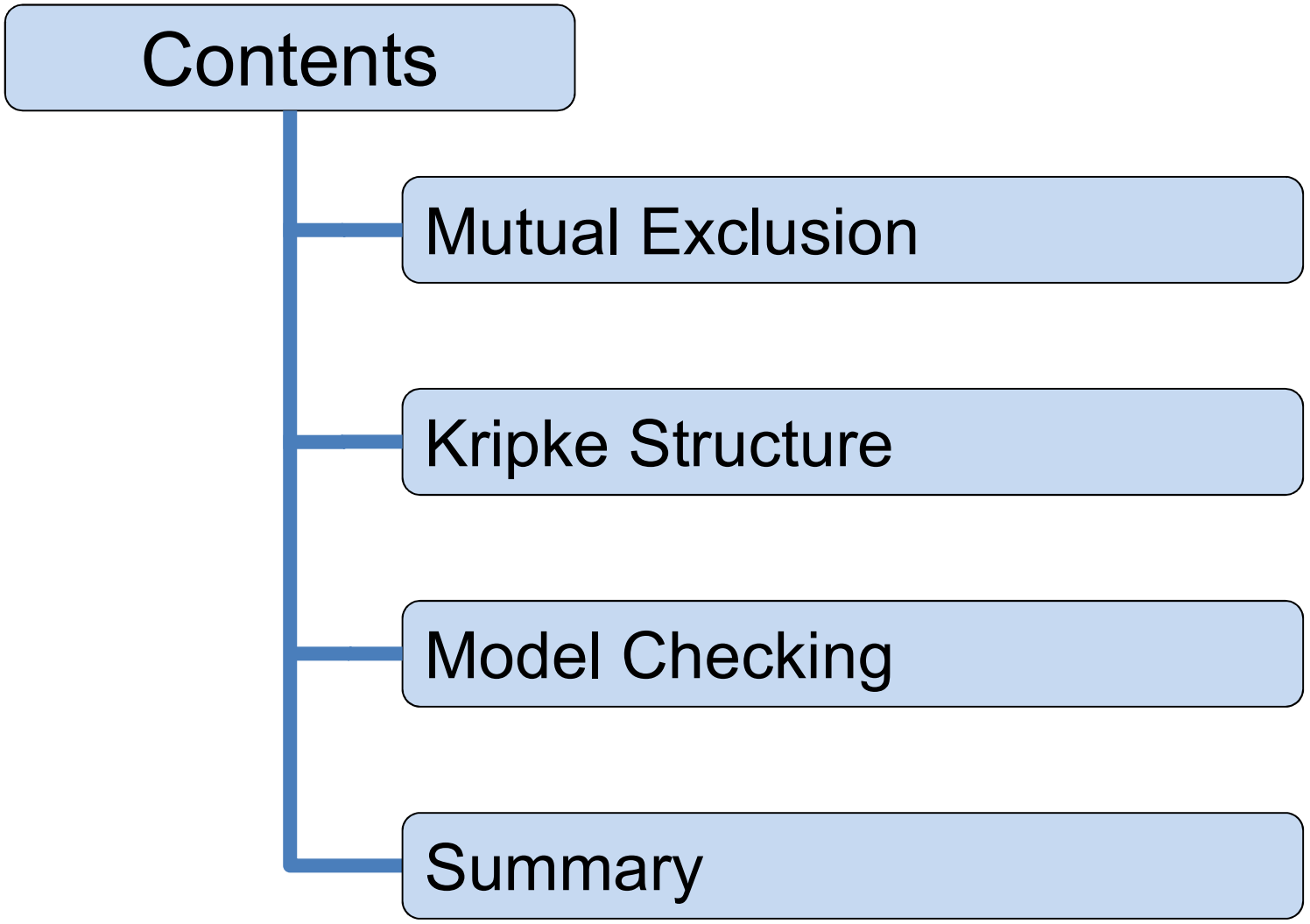
the set of fair computations of K is empty, iff

K'=<S,R,I,[[F]]> is empty, iff

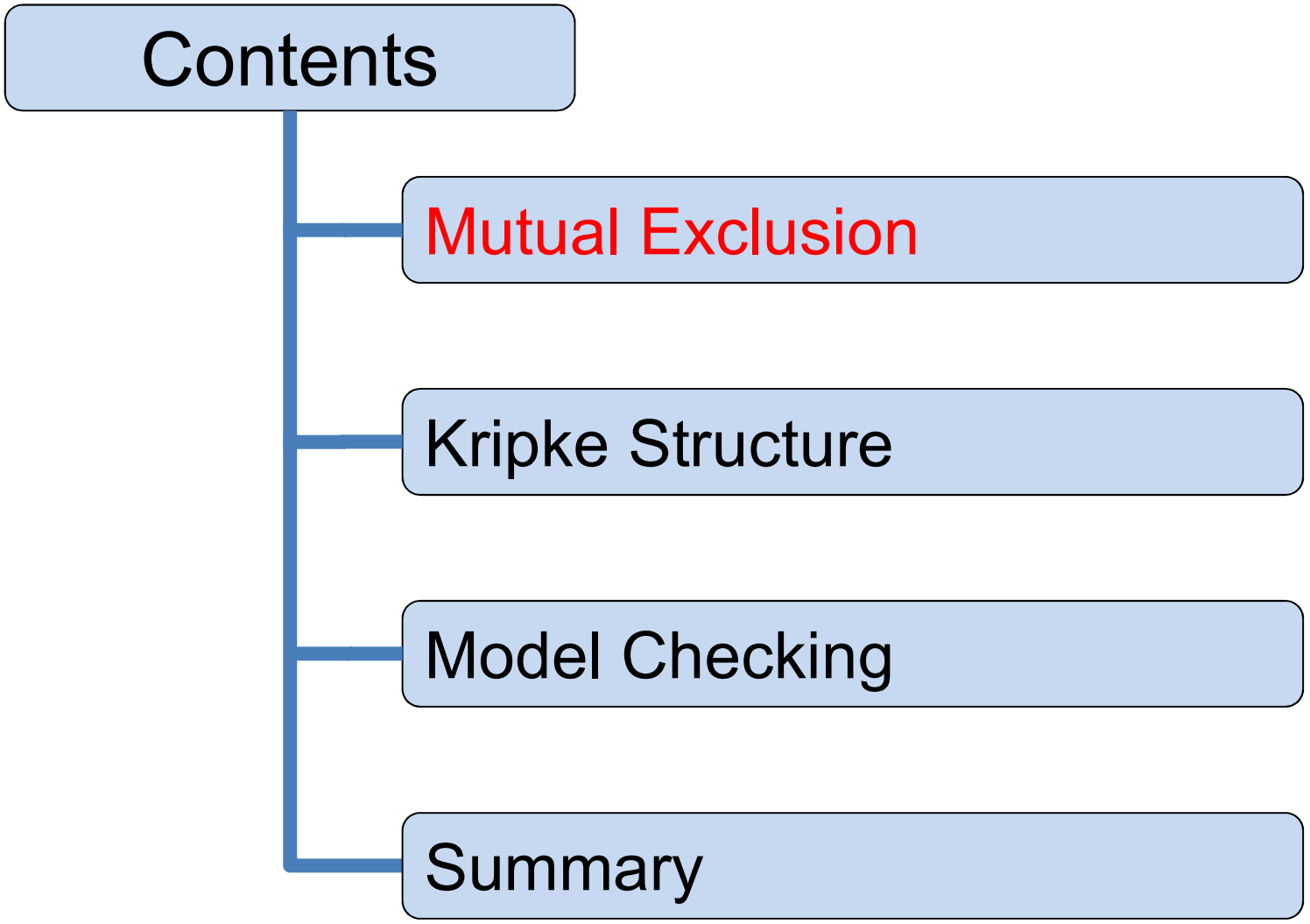
EmpChecking(K')

(IV) An Example

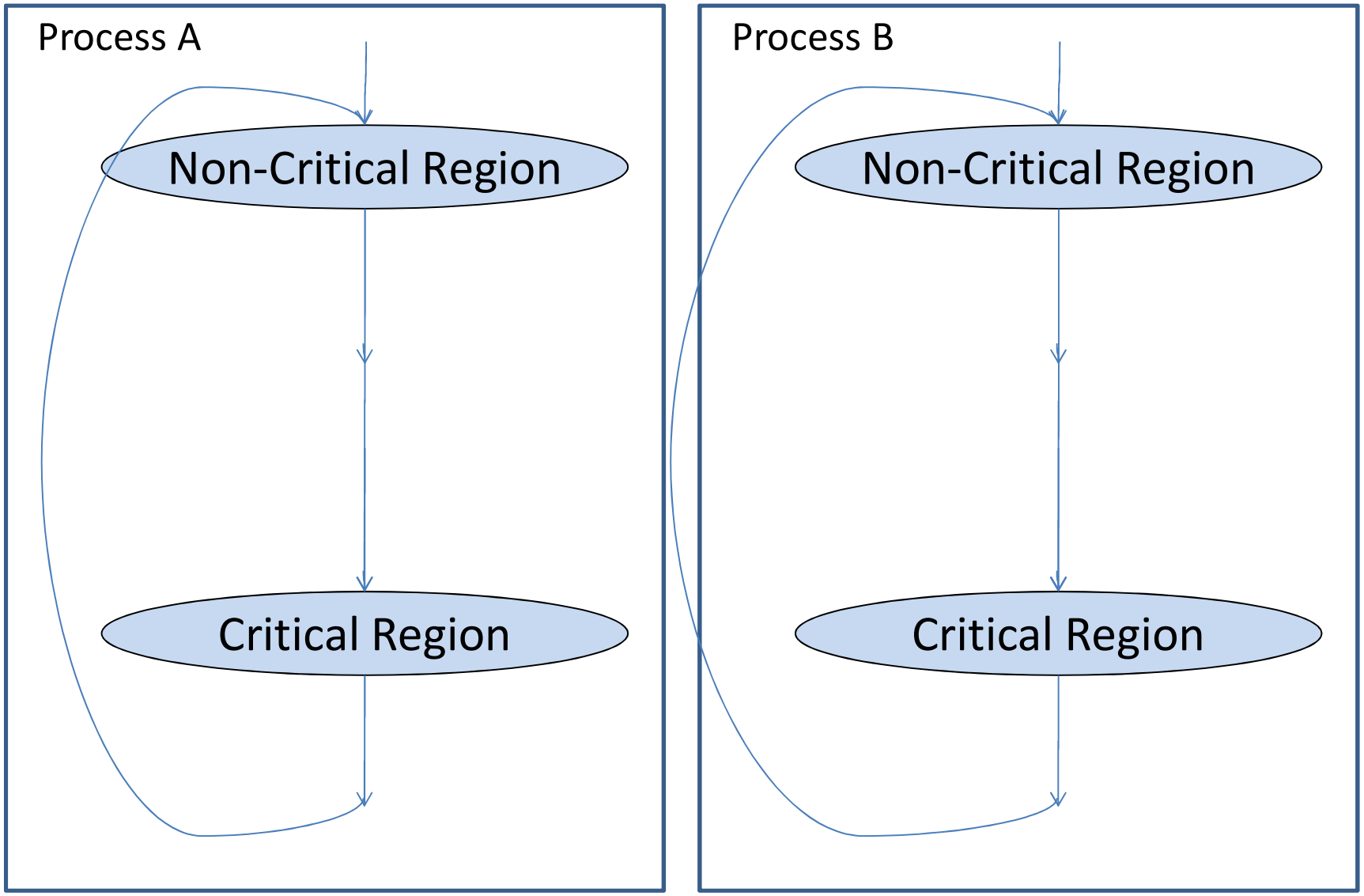
An Example


Contents
Mutual Exclusion
Kripke Structure
Model Checking
Summary

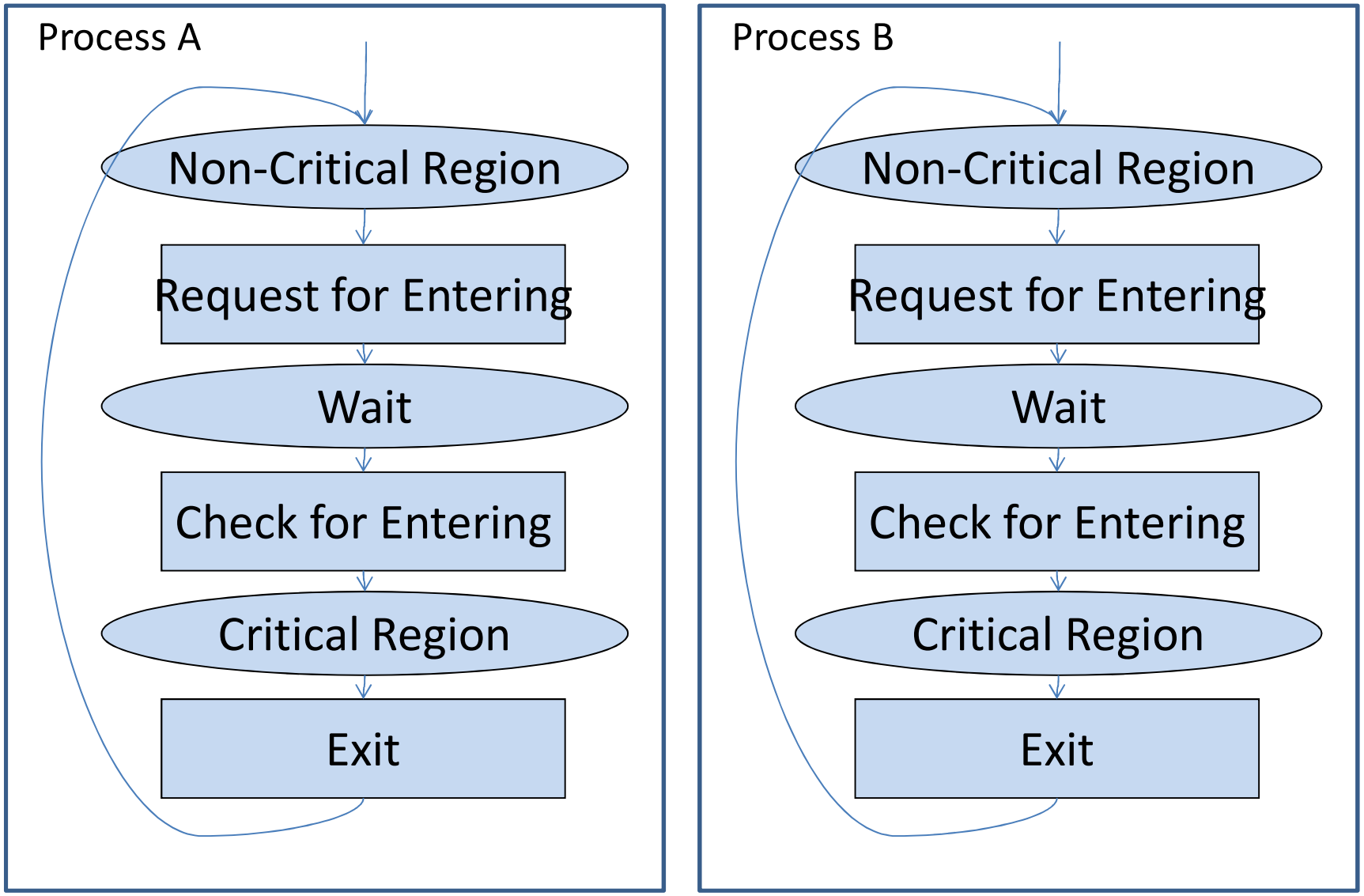
An Example


Contents
Mutual Exclusion
Kripke Structure
Model Checking
Summary

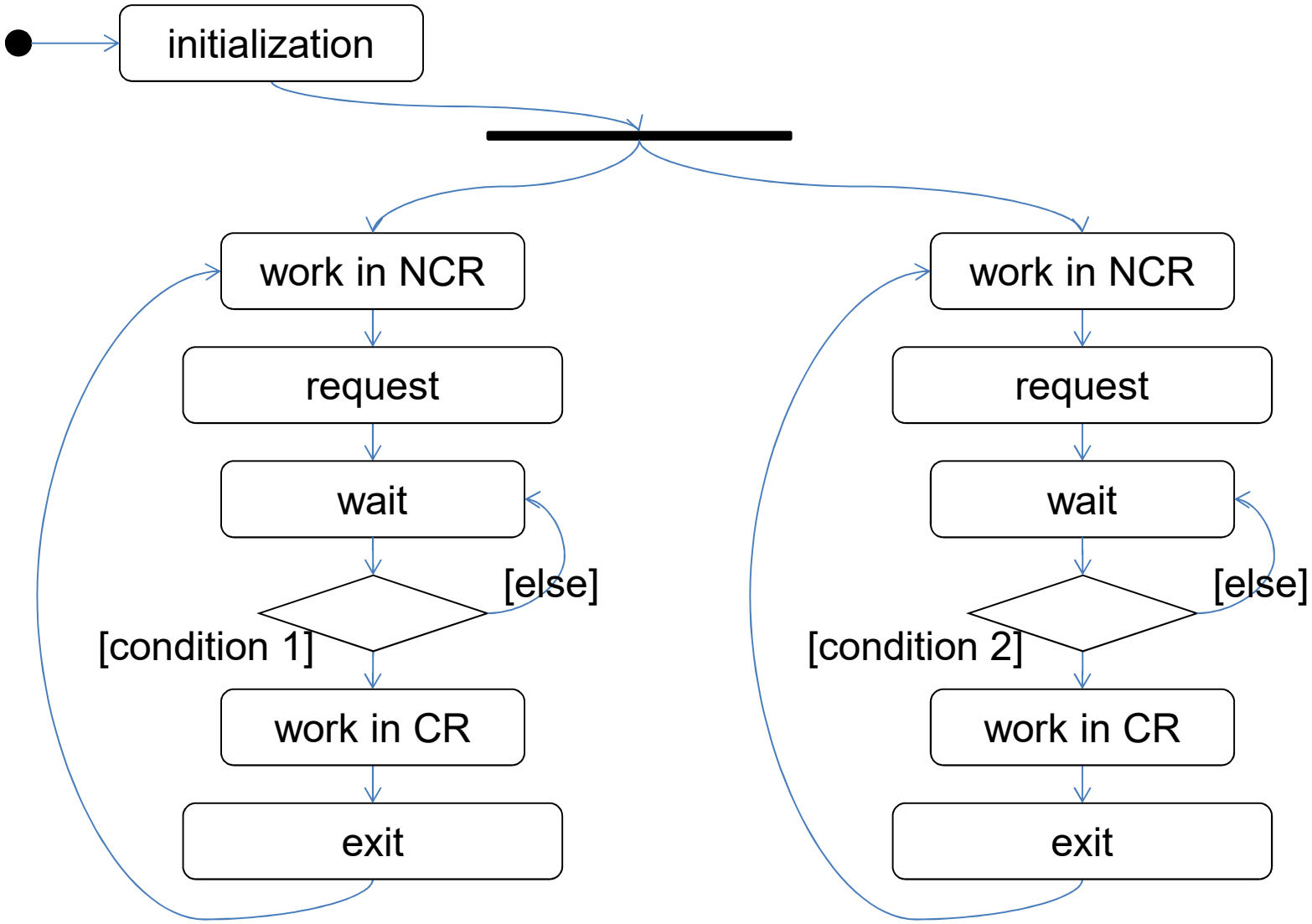
Example: Mutual Exclusion



Example: Mutual Exclusion



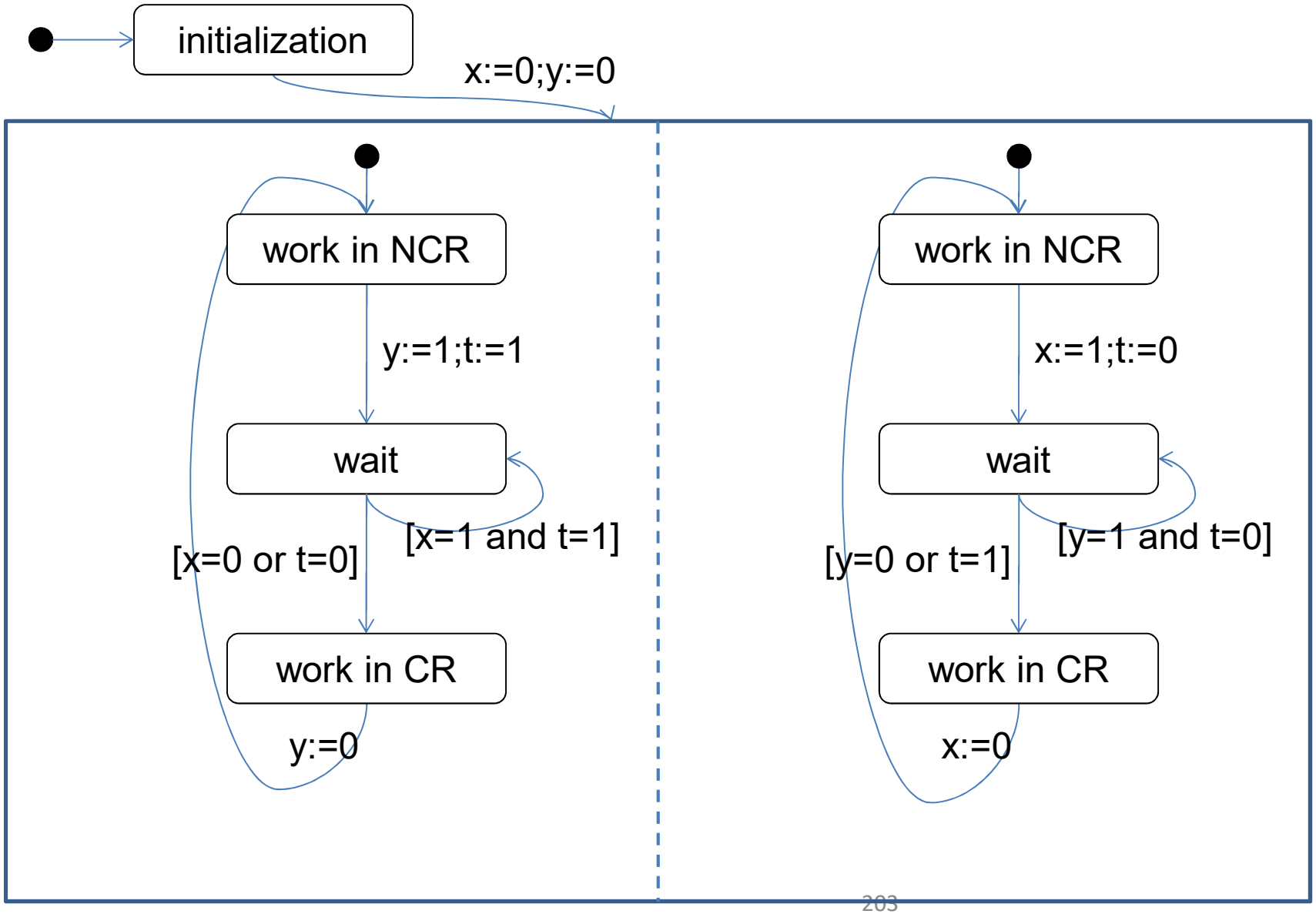
Design of Mutual Exclusion (Activity)



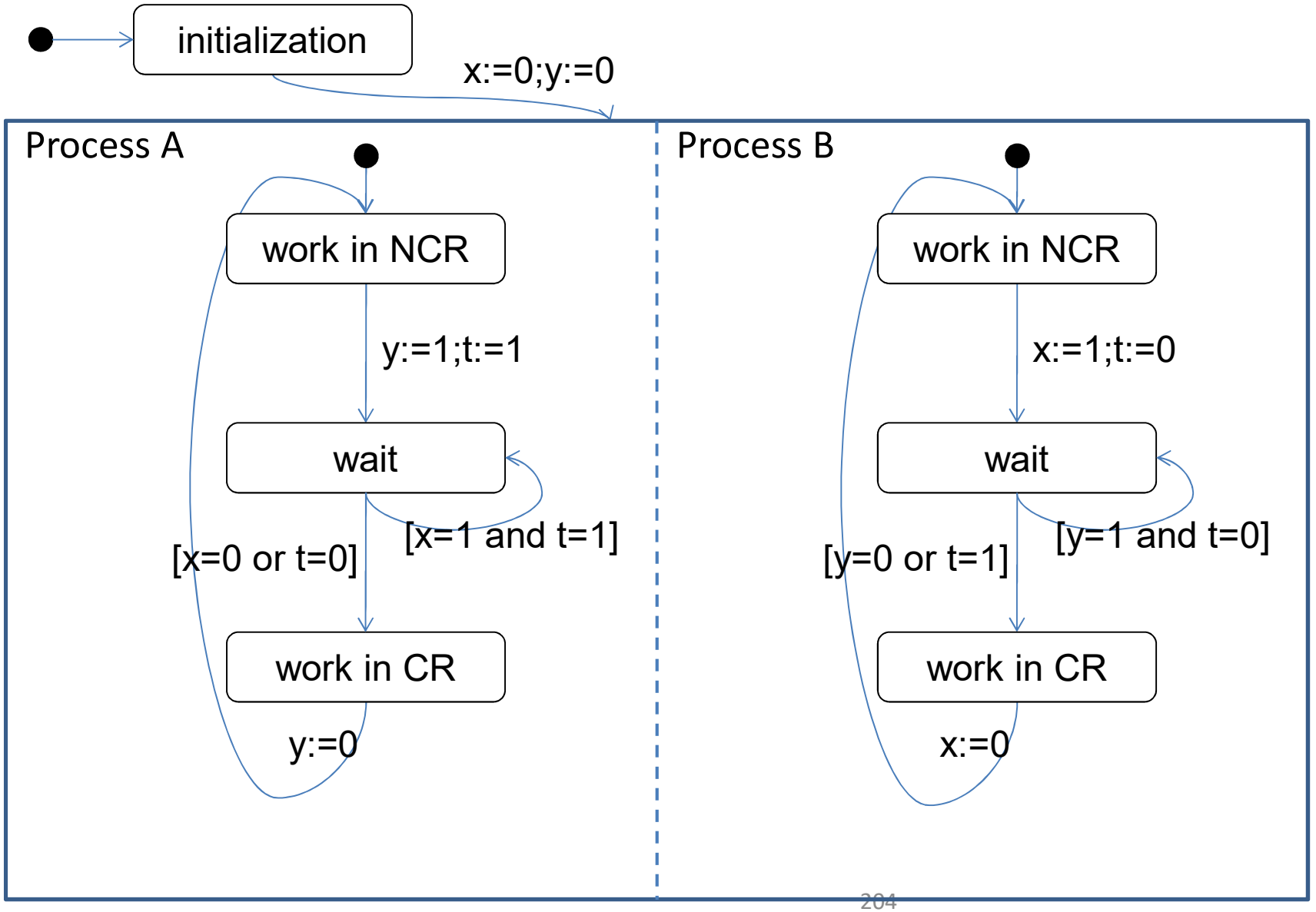
Design of Mutual Exclusion

- 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)



Design of Mutual Exclusion (State)



Correctness of the Design

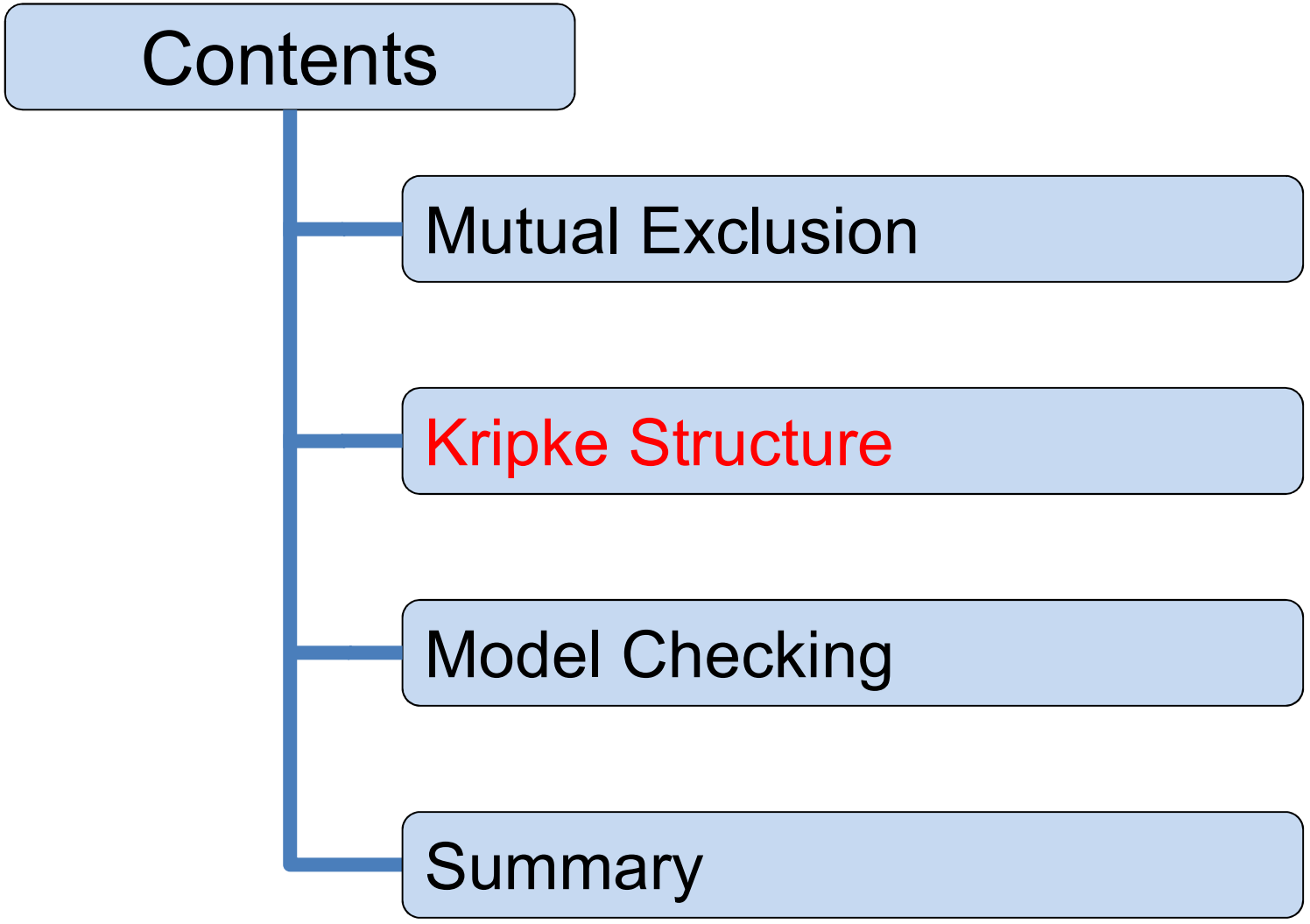
- How do we know that the design is correct?

Combined States of the Two Processes

Process A	Process B	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


Contents
Mutual Exclusion
Kripke Structure
Model Checking
Summary

Fair Kripke Structures

AP: A set of propositions.

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

- S : A finite set of states
- $R \subseteq S \times S$: A total transition relation
- $I \subseteq S$: A set of initial states
- L: S → 2^{AP} is a labeling function
- F: A set of formulas over AP

Process States 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,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)

The Set of Initial States: I

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

Labeling Function

AP={

a=NCR,a=wait,a=CR,b=NCR,b=wait,b=CR,

x=1,x=0,y=1,y=0,t=1,t=0

}

L(NCR,NCR,0,0,0)={a=NCR,b=NCR,x=0,y=0,t=0}

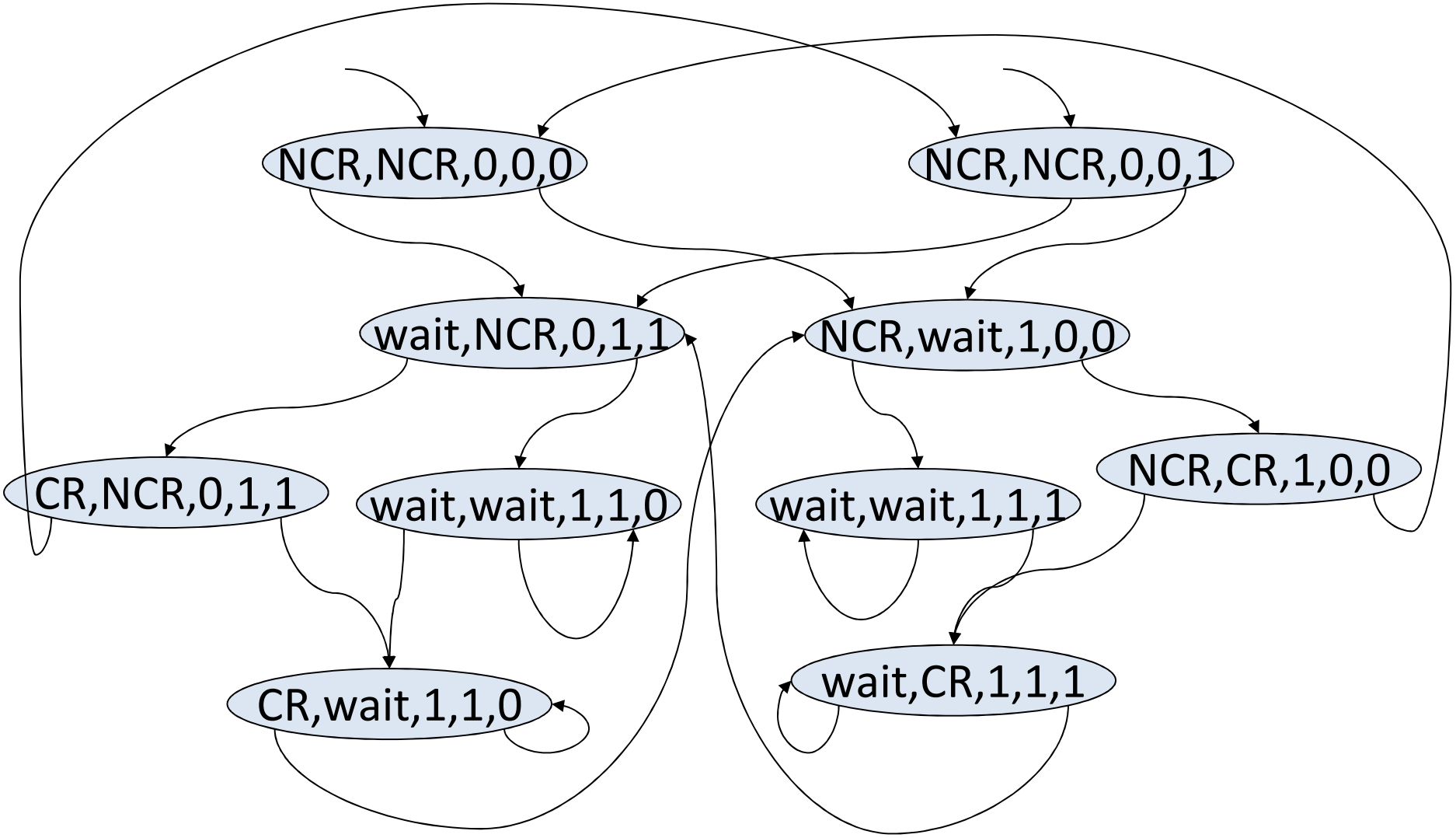
L(NCR,NCR,0,0,1)={a=NCR,b=NCR,x=0,y=0,t=1}

…

Fairness

$$
\begin{aligned}
F=\{ & \\
& \neg(b=NCR), \\
& \neg((x=0 \vee t=0) \wedge a=wait), \\
& \neg(a=CR), \\
& \neg(b=NCR), \\
& \neg((y=0 \vee t=1) \wedge b=wait), \\
& \neg(b=CR) \\
\} &
\end{aligned}
$$

Example


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

Safety Property

$\varphi = \neg(a=CR \wedge b=CR)$

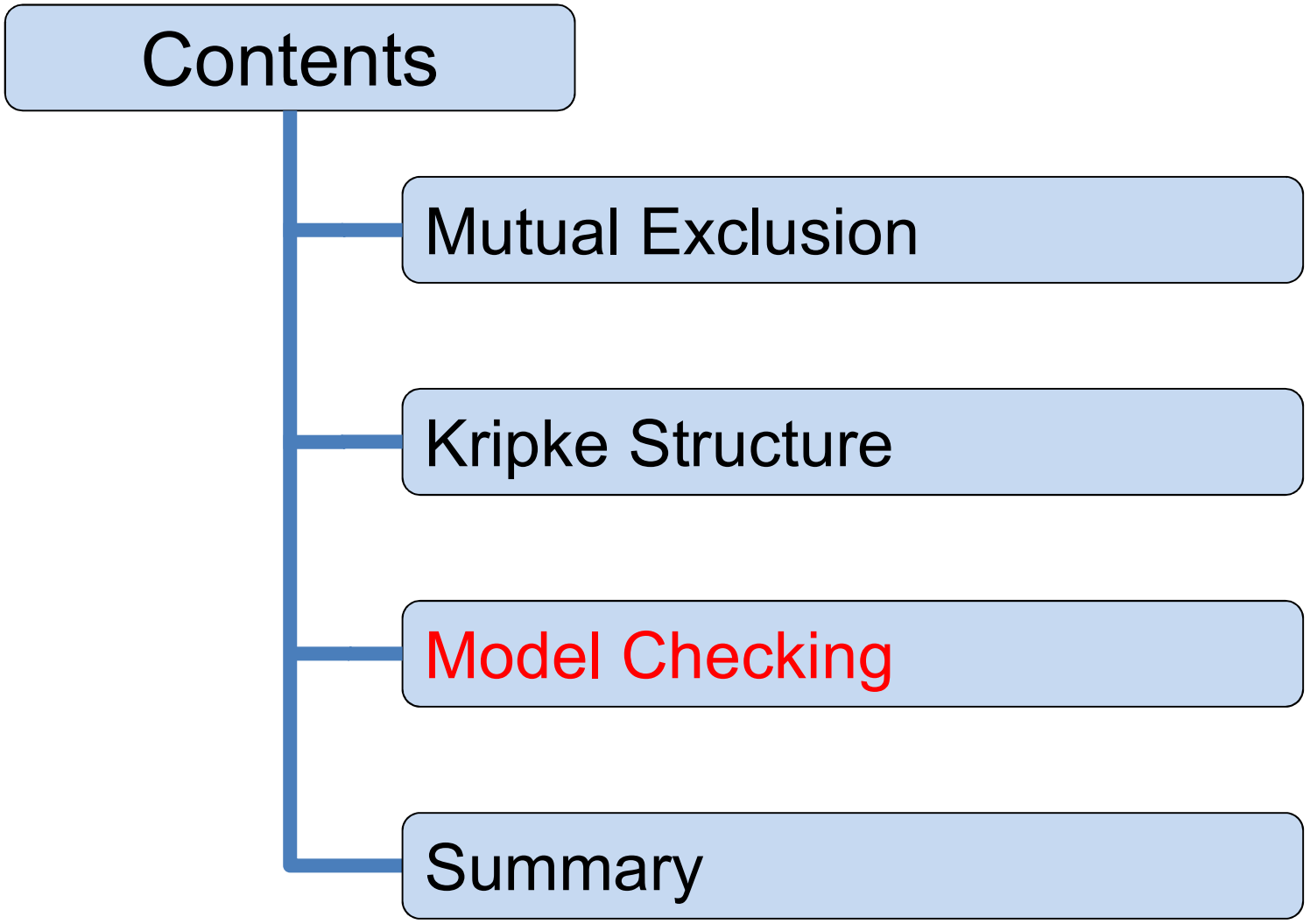
Is φ a safety property?

Inevitability Property

ψ = (a=CR$\vee$b=CR)

Is ψ an inevitability property?

An Example


Contents
Mutual Exclusion
Kripke Structure
Model Checking
Summary

Modeling and Model Checking

- Model Checking with VERDS
  - http://lcs.ios.ac.cn/~zwh/verds
- Input to VERDS
  - VVM (VERDS verification model)
- Modeling Language
  - VML (VERDS modeling langauge)

Without Fairness Specifications

Modeling in VML

```
VVM
VAR
        x: 0..1;
        y: 0..1;
        t: 0..1;
INIT
         x=0;
         y=0;
PROC
        p0: p0m();
        p1: p1m();
```

```
SPEC
        AG(!(p0.a=c0&p1.b=c0));
```

Safety:
Mutual exclusion

Modeling in VML

```
 MODULE  p0m()
VAR
      a: {n0,w0,c0};
INIT
      a=n0;
TRANS
      a=n0:                    (y,t,a):=(1,1,w0);
      a=w0&(x=0|t=0):  (a):=(c0);
      a=w0&!(x=0|t=0): (a):=(w0);
      a=c0:                    (y,a):=(0,n0);
```

```
MODULE  p1m()
VAR
      b: {n0,w0,c0};
INIT
      b=n0;
TRANS
      b=n0:                    (x,t,b):=(1,0,w0);
      b=w0&(y=0|t=1):  (b):=(c0);
      b=w0&!(y=0|t=1): (b):=(w0);
      b=c0:                    (x,b):=(0,n0);
```

The Complete Model in VML

```
VVM
VAR
    x: 0..1;
    y: 0..1;
    t: 0..1;
INIT
    x=0;
    y=0;
PROC
    p0: p0m();
    p1: p1m();

SPEC
    AG(!(p0.a=c0&p1.b=c0));
```

```
 MODULE  p0m()
VAR
    a: {n0,w0,c0};
INIT
    a=n0;
TRANS
    a=n0:                  (y,t,a):=(1,1,w0);
    a=w0&(x=0|t=0):  (a):=(c0);
    a=w0&!(x=0|t=0): (a):=(w0);
    a=c0:                  (y,a):=(0,n0);

MODULE  p1m()
VAR
    b: {n0,w0,c0};
INIT
    b=n0;
TRANS
    b=n0:                 (x,t,b):=(1,0,w0);
    b=w0&(y=0|t=1):  (b):=(c0);
    b=w0&!(y=0|t=1): (b):=(w0);
    b=c0:                 (x,b):=(0,n0);
```

Verification with VERDS

```
./verds -ck 1 mutex3.vvm
VERSION:    verds 1.46 - JAN 2015
FILE:       mutex3.vvm
INFO:       int=i0
PROPERTY:   A G ! ((p0.a = 2 )& (p1.b = 2 ))
check:   0
----------
check:   1
----------
…
...
check:   5
----------
CONCLUSION: TRUE
```

Consider the Inevitability Property

```
VVM
VAR
      x: 0..1;
      y: 0..1;
      t: 0..1;
INIT
       x=0;
       y=0;
PROC
      p0: p0m();
      p1: p1m();
```

```
SPEC
      AG(!(p0.a=c0&p1.b=c0));
      AF((p0.a=c0)|(p1.b=c0));
```

Inevitability:
Working in critical region

Verification with VERDS

```
./verds -ck 2  mutex3.vvm
VERSION:    verds 1.46 - JAN 2015
FILE:       mutex3.vvm
INFO:       int=i0
PROPERTY:   A F ((p0.a = 2 )| (p1.b = 2 ))
check:   0
----------
check:   1
----------
…
…
check:   4
----------
The property is false, preparing mutex3.cex ...
CONCLUSION: FALSE
```

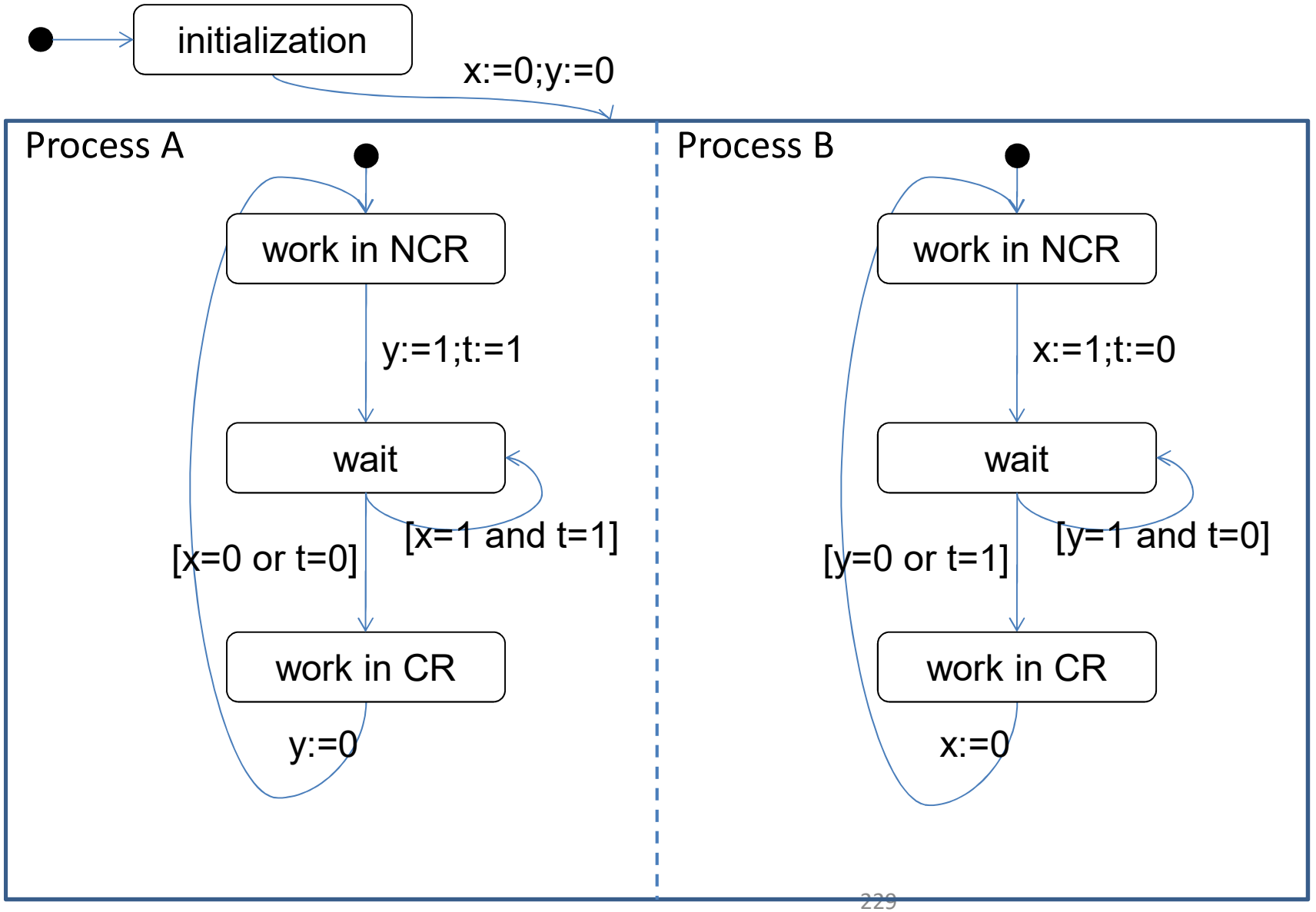
Verification with VERDS

LOOP starts from STATE 2:

--- STATE 0 ---
x =0
y =0
t =1
p0.a =0
p1.b =0
--- TRANS 5 ---
--- STATE 1 ---
x =1
y =0
t =0
p0.a =0
p1.b =1

--- TRANS 1 ---
--- STATE 2 ---
x =1
y =1
t =1
p0.a =1
p1.b =1
--- TRANS 3 ---
--- STATE 3 ---
x =1
y =1
t =1
p0.a =1
p1.b =1

Checking the Model



With Fairness Specifications

Modified Model (with Fairness)

```
MODULE p0m()
VAR
    a: {n0,w0,c0};
INIT
    a=n0;
TRANS
    a=n0:                  (y,t,a):=(1,1,w0);
    a=w0&(x=0|t=0):  (a):=(c0);
    a=w0&!(x=0|t=0): (a):=(w0);
    a=c0:                  (y,a):=(0,n0);
FAIRNESS
        !(a=n0);
    !((x=0|t=0)&(a=w0));
        !(a=c0);
```

```
MODULE p1m()
VAR
    b: {n0,w0,c0};
INIT
    b=n0;
TRANS
    b=n0:                  (x,t,b):=(1,0,w0);
    b=w0&(y=0|t=1):  (b):=(c0);
    b=w0&!(y=0|t=1): (b):=(w0);
    b=c0:                  (x,b):=(0,n0);
FAIRNESS
    !(b=n0);
    !((y=0|t=1)&(b=w0));
    !(b=c0);
```

Verification with VERDS

```
./verds -ck 1 mutex3f.vvm
VERSION:    verds 1.46 - JAN 2015
FILE:       mutex3f.vvm
INFO:       int=i0
PROPERTY:   A G ! ((p0.a = 2 )& (p1.b = 2 ))
check:   0
----------
check:   1
----------
…
...
check:   19
----------
CONCLUSION: TRUE
```

Verification with VERDS

```
./verds -ck 2 mutex3f.vvm
VERSION:    verds 1.46 - JAN 2015
FILE:       mutex3f.vvm
INFO:       int=i0
PROPERTY:   A F ((p0.a = 2 )| (p1.b = 2 ))
check:   0
----------
check:   1
----------
…
...
check:  99
----------
CONCLUSION: TRUE
```

Correctness of the Design

- How do we know that the design is correct?
  - Safety property: $\neg(a=CR \wedge b=CR)$
  - Inevitability property: $(a=CR \vee b=CR)$
  - We have shown that a safety property holds, i.e., the bad states are not reachable
  - We have also shown that an inevitability property holds under a fairness assumption, i.e., some good states must be reached in every computation

Remarks on the Correctness

- Only verified against the given properties:
  - The safety property
  - The inevitability property
- Rely on:
  - The model
  - The verification tool
  - The fairness assumption

as a part of the model, for the verification of the inevitability property

Modified Model (with Inappropr. Fairness)

```
 MODULE  p0m()
VAR
      a: {n0,w0,c0};
INIT
      a=n0;
TRANS
      a=n0:                    (y,t,a):=(1,1,w0);
      a=w0&(x=0|t=0):  (a):=(c0);
      a=w0&!(x=0|t=0): (a):=(w0);
      a=c0:                    (y,a):=(0,n0);
FAIRNESS
     !(a=c0|a=w0|a=n0);
```

```
MODULE  p1m()
VAR
      b: {n0,w0,c0};
INIT
      b=n0;
TRANS
      b=n0:                    (x,t,b):=(1,0,w0);
      b=w0&(y=0|t=1):  (b):=(c0);
      b=w0&!(y=0|t=1): (b):=(w0);
      b=c0:                    (x,b):=(0,n0);
FAIRNESS
     !(b=n0);
     !((y=0|t=1)&(b=w0));
     !(b=c0);
```

Verification with VERDS

../verds -ck 1 mutex3fw.vvm

VERSION: verds 1.46 - JAN 2015

FILE: mutex3wf.vvm

INFO: int=i0

PROPERTY: A G ! ((p0.a = 2)& (p1.b = 2))

check: 0

---

check: 1

…

...

check: 25

CONCLUSION: TRUE

Verification with VERDS

```
./verds -ck 2 mutex3fw.vvm
VERSION:    verds 1.46 - JAN 2015
FILE:       mutex3wf.vvm
INFO:       int=i0
PROPERTY:   A F ((p0.a = 2 )| (p1.b = 2 ))
check:   0
----------
check:   1
----------
…
...
check:   40
----------
CONCLUSION: TRUE
```

Modified Model (with Inappropr. Fairness)

```
 MODULE  p0m()
VAR
      a: {n0,w0,c0};
INIT
      a=n0;
TRANS
      a=n0:                    (y,t,a):=(1,1,w0);
      a=w0&(x=0|t=0):  (a):=(c0);
      a=w0&!(x=0|t=0): (a):=(w0);
      a=c0:                    (y,a):=(0,n0);
FAIRNESS
     !(a=c0|a=w0|a=n0);
```

```
MODULE  p1m()
VAR
      b: {n0,w0,c0};
INIT
      b=n0;
TRANS
      b=n0:                    (x,t,b):=(1,0,w0);
      b=w0&(y=0|t=1):  (b):=(c0);
      b=w0&!(y=0|t=1): (b):=(w0);
      b=c0:                    (x,b):=(0,n0);
FAIRNESS
     !(b=n0);
     !((y=0|t=1)&(b=w0));
     !(b=c0);
```

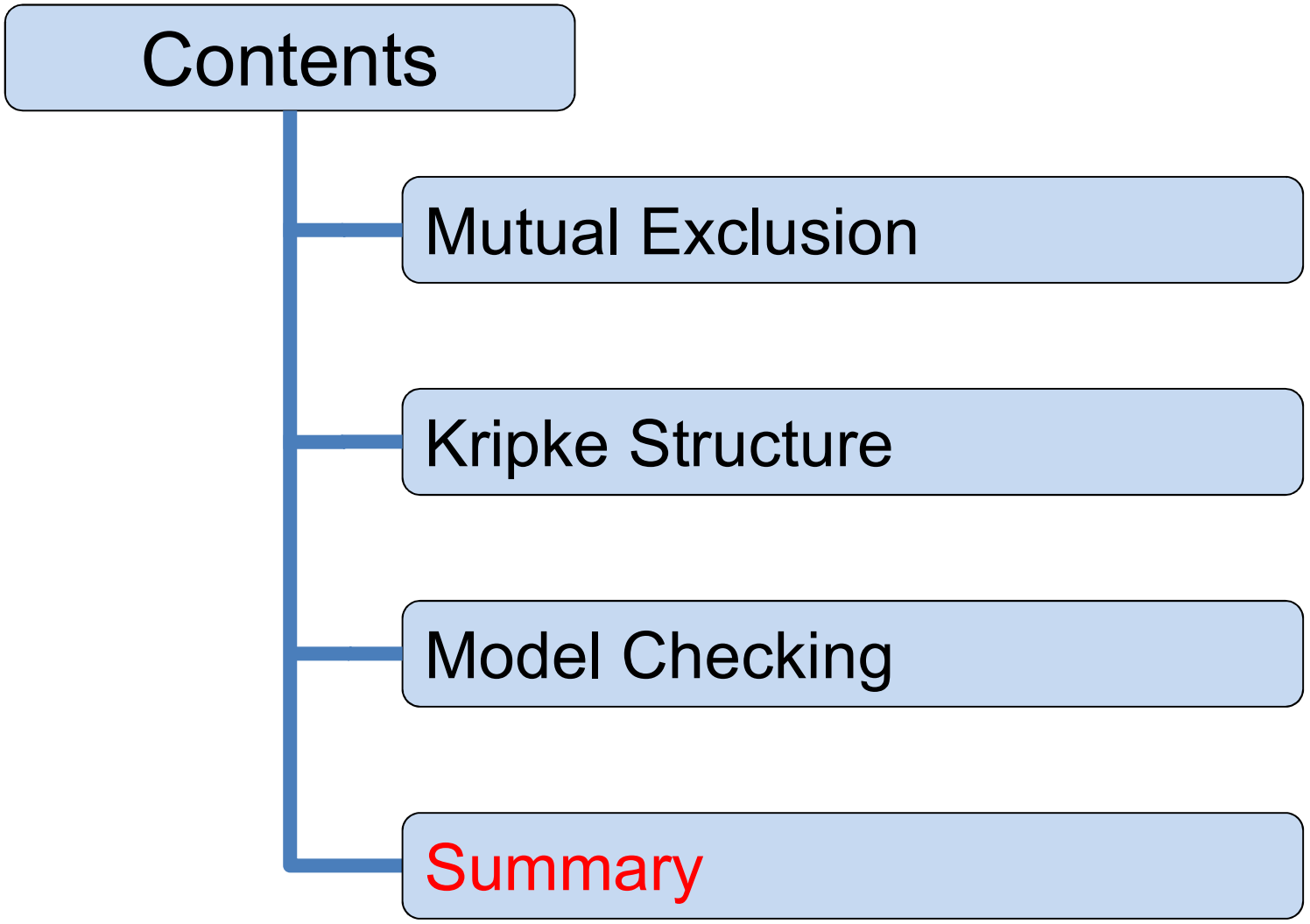
Modified Model (with Inappropr. Fairness)

```
MODULE p0m()
VAR
    a: {n0,w0,c0};
INIT
    a=n0;
TRANS
    a=n0:                  (y,t,a):=(1,1,w0);
    a=w0&(x=0|t=0):  (a):=(c0);
    a=w0&!(x=0|t=0): (a):=(w0);
    a=c0:                  (y,a):=(0,n0);
FAIRNESS
    !(a=c0|a=w0|a=n0);
```

```
MODULE p1m()
VAR
    b: {n0,w0,c0};
INIT
    b=n0;
TRANS
    b=n0:                  (x,t,b):=(1,0,w0);
    b=w0&(y=0|t=1):  (b):=(c0);
    b=w0&!(y=0|t=1): (b):=(w0);
    b=c0:                  (x,b):=(0,n0);
FAIRNESS
    !(b=c0);
```

This model is empty

An Example


Contents
Mutual Exclusion
Kripke Structure
Model Checking
Summary

(V) Summary

- Fair Kripke Structures
- Labeled Kripke Structures
- Fair Labeled Kripke Structures
- An Example

思考题：

给定一个公平Kripke模型K=<S,R,I,F>和一个集合A⊆S.

(1)
通过对模型进行改造，用公平Kripke模型非空问题算法验证A是否是K的公平可达性质

(2)
通过对模型进行改造，用公平Kripke模型非空问题算法验证A是否是K的公平可避免性质