Overview

Introduction

Modelling parallel systems

- Transition systems
- Modeling hard- and software systems
- Parallelism and communication ←

Linear Time Properties

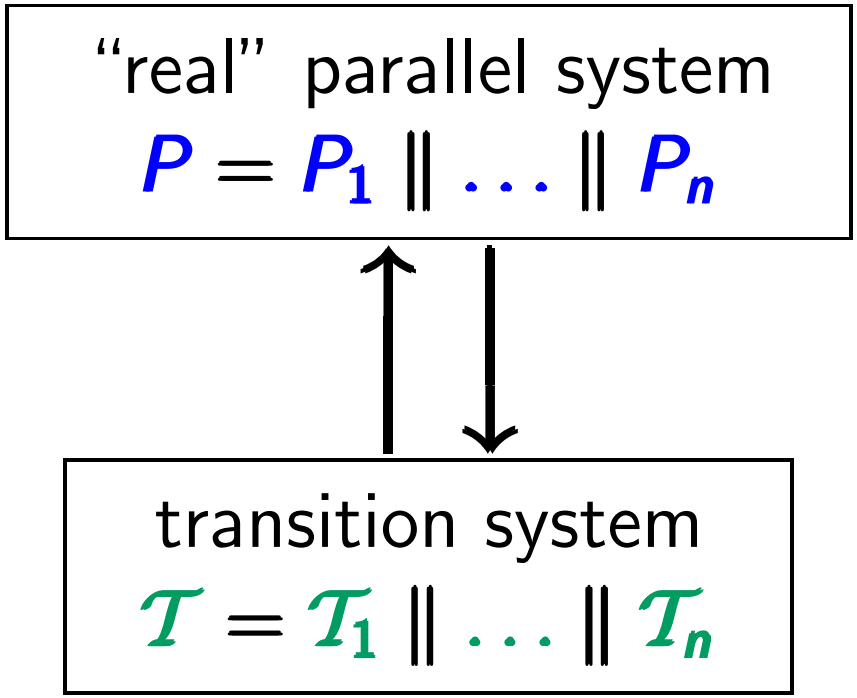
Regular Properties

Linear Temporal Logic

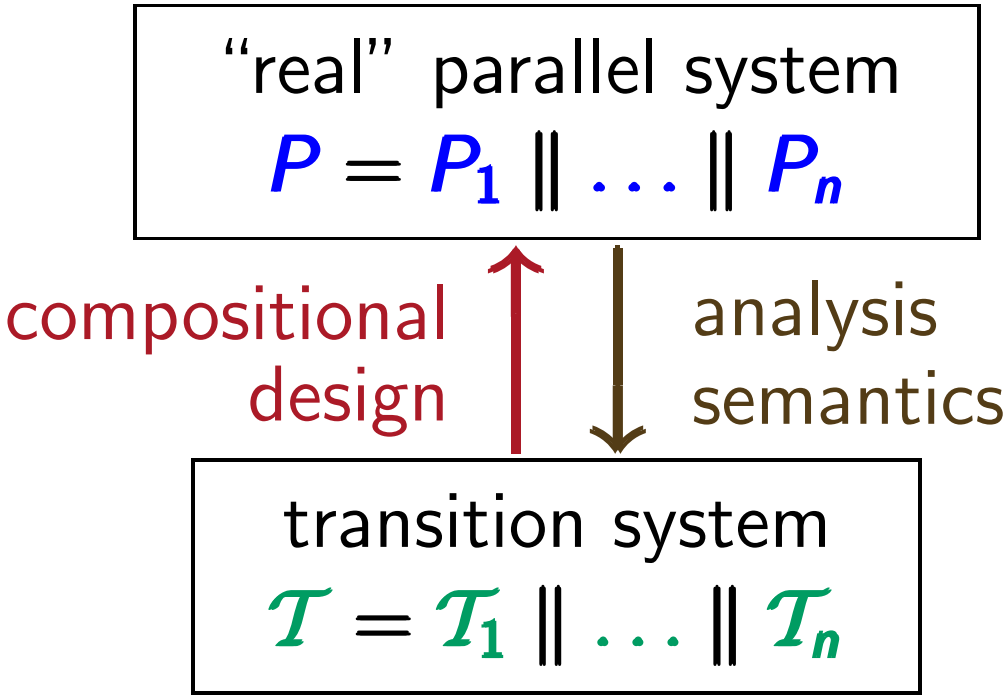
Computation-Tree Logic

Equivalences and Abstraction

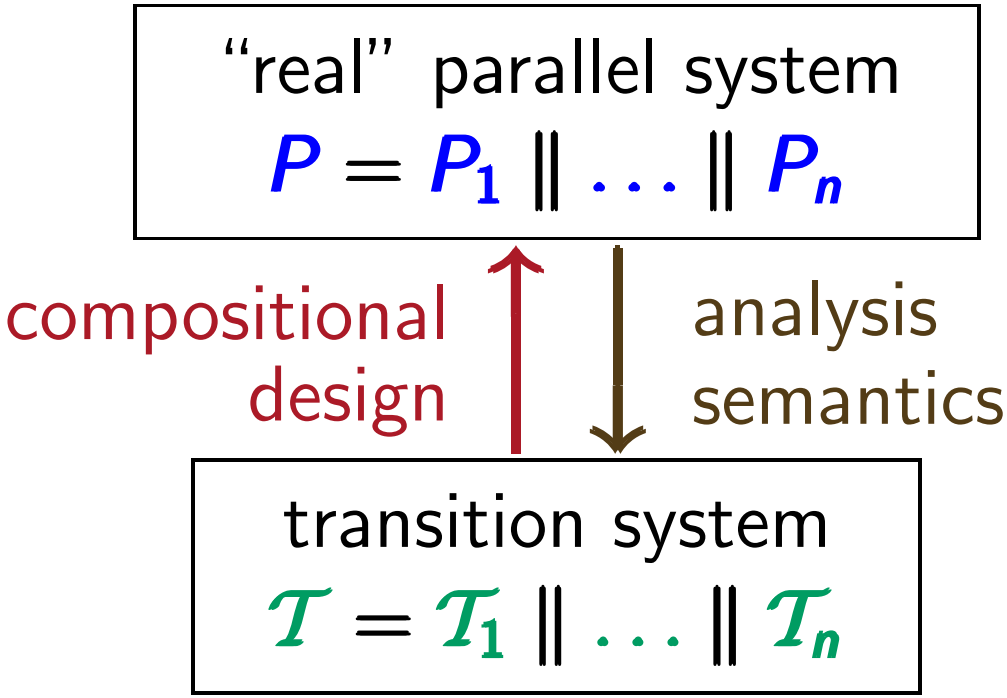
Parallelism and communication



Parallelism and communication



Parallelism and communication



goal: define semantic parallel operators on transition systems or program graphs that model "real" parallel operators

Interleaving operator ||| for TS

- interleaving of concurrent, independent actions of parallel processes (modelled by TS)
- representation by nondeterministic choice:

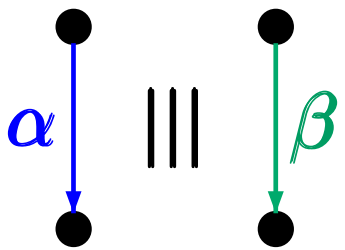
  "which subprocess performs the next step?"

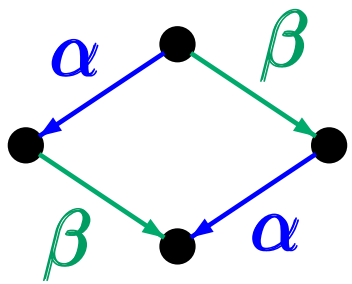
Interleaving operator ||| for TS

- interleaving of concurrent, independent actions of parallel processes (modelled by TS)
- representation by nondeterministic choice:

  "which subprocess performs the next step?"

$$\mathit{effect}(\alpha|||\beta) \;=\; \mathit{effect}(\alpha;\beta+\beta;\alpha)$$



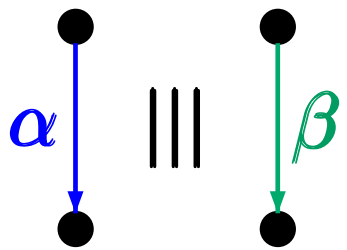


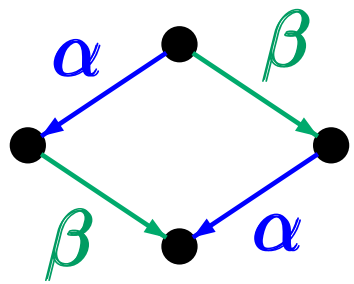
Interleaving operator ||| for TS

- interleaving of concurrent, independent actions of parallel processes (modelled by TS)
- representation by nondeterministic choice:

  "which subprocess performs the next step?"

$$\mathit{effect}(\alpha|||\beta) \;=\; \mathit{effect}(\alpha;\beta+\beta;\alpha)$$





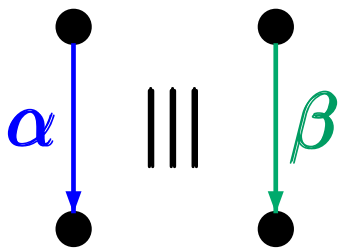
parallel execution of α and β on *two processors*

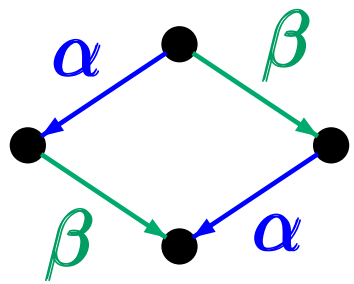
Interleaving operator ||| for TS

- interleaving of concurrent, independent actions of parallel processes (modelled by TS)
- representation by nondeterministic choice:

  "which subprocess performs the next step?"

$$\mathit{effect}(\alpha|||\beta) \;=\; \mathit{effect}(\alpha;\beta+\beta;\alpha)$$





parallel execution of α and β on *two processors* $\;\widehat{=}\;$ serial execution on a *single processor* in arbitrary order

Interleaving operator $|||$ for TS

$$\mathcal{T}_1 = (S_1, Act_1, \longrightarrow_1, S_{0,1}, AP_1, L_1)$$

$$\mathcal{T}_2 = (S_2, Act_2, \longrightarrow_2, S_{0,2}, AP_2, L_2)$$

Interleaving operator $|||$ for TS

$$\mathcal{T}_1 = (S_1, Act_1, \longrightarrow_1, S_{0,1}, AP_1, L_1)$$

$$\mathcal{T}_2 = (S_2, Act_2, \longrightarrow_2, S_{0,2}, AP_2, L_2)$$

The transition system $\mathcal{T}_1 ||| \mathcal{T}_2$ is defined by:

$$\mathcal{T}_1 ||| \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \longrightarrow, S_{0,1} \times S_{0,2}, AP, L)$$

where the transition relation $\longrightarrow$ is given by:

Interleaving operator $|||$ for TS

$$\mathcal{T}_1 = (S_1, Act_1, \longrightarrow_1, S_{0,1}, AP_1, L_1)$$

$$\mathcal{T}_2 = (S_2, Act_2, \longrightarrow_2, S_{0,2}, AP_2, L_2)$$

The transition system $\mathcal{T}_1 ||| \mathcal{T}_2$ is defined by:

$$\mathcal{T}_1 ||| \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \longrightarrow, S_{0,1} \times S_{0,2}, AP, L)$$

where the transition relation $\longrightarrow$ is given by:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

Interleaving operator $|||$ for TS

$$\mathcal{T}_1 = (S_1, Act_1, \longrightarrow_1, S_{0,1}, AP_1, L_1)$$

$$\mathcal{T}_2 = (S_2, Act_2, \longrightarrow_2, S_{0,2}, AP_2, L_2)$$

The transition system $\mathcal{T}_1 ||| \mathcal{T}_2$ is defined by:

$$\mathcal{T}_1 ||| \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \longrightarrow, S_{0,1} \times S_{0,2}, AP, L)$$

where the transition relation $\longrightarrow$ is given by:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

atomic propositions: $AP = AP_1 \uplus AP_2$

labeling function: $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$

SOS-notation (structured operational semantics)

just a simple notation for operational semantics

$$\frac{\textbf{premise}}{\textbf{conclusion}}$$

SOS-notation (structured operational semantics)

just a simple notation for operational semantics

$$\frac{\textbf{premise}}{\textbf{conclusion}}$$

E.g., *"the relation $\longrightarrow$ is given by ..."*

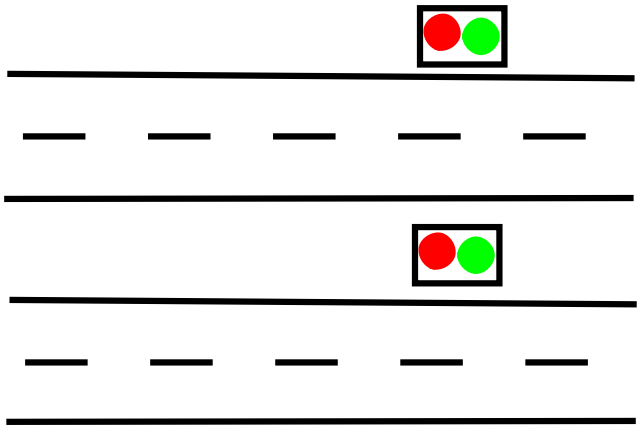
$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2\rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1, s_2'\rangle}$$

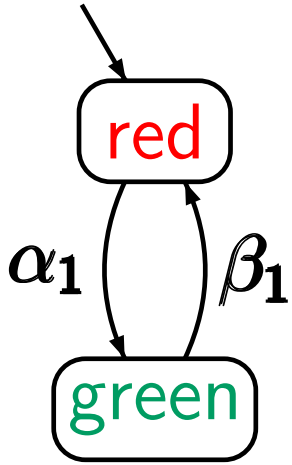
means that $\longrightarrow$ is the smallest relation such that:

(1) If $s_1 \xrightarrow{\alpha}_1 s_1'$, then $\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2\rangle$

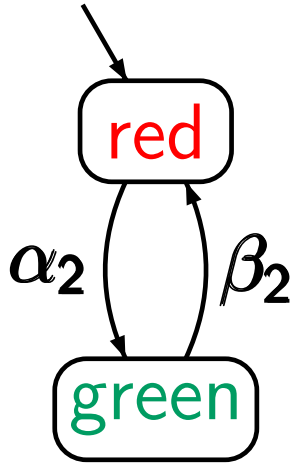
(2) If $s_2 \xrightarrow{\alpha}_2 s_2'$, then $\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1, s_2'\rangle$

Useless lights for non-crossing streets



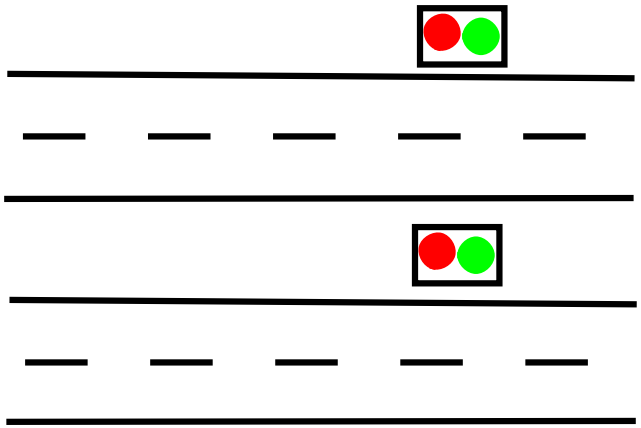


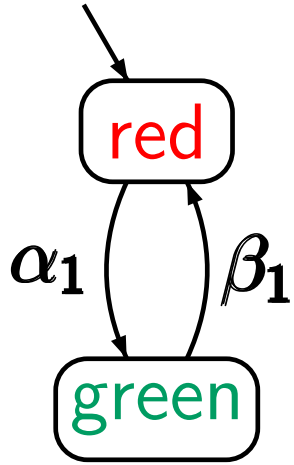
light 1



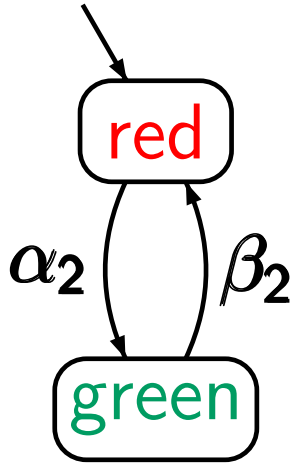
light 2

Useless lights for non-crossing streets



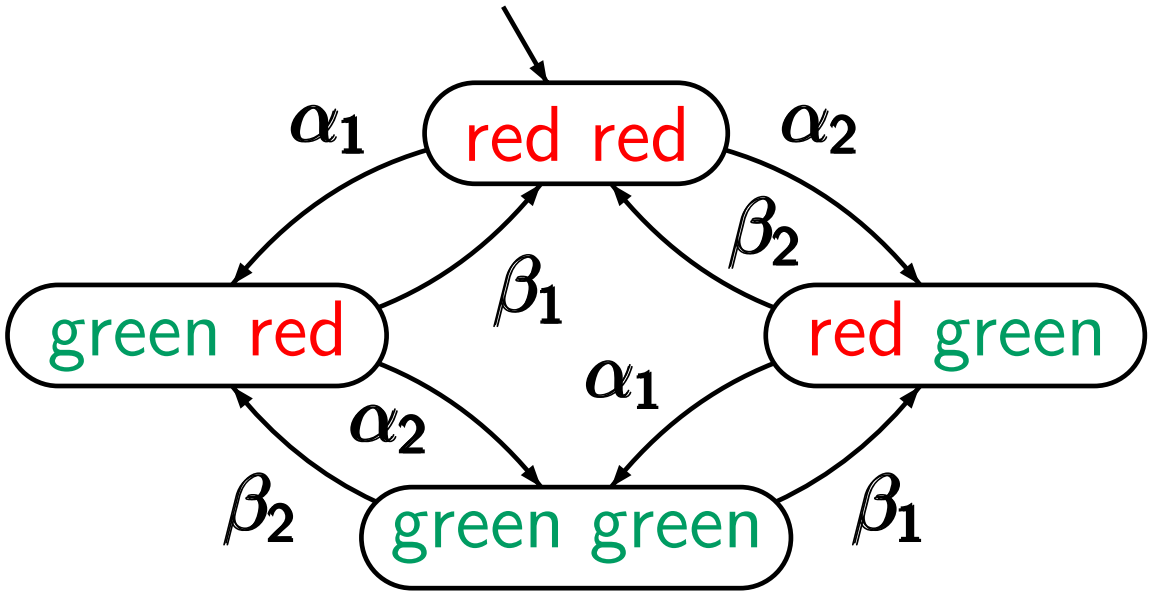


light 1



light 2

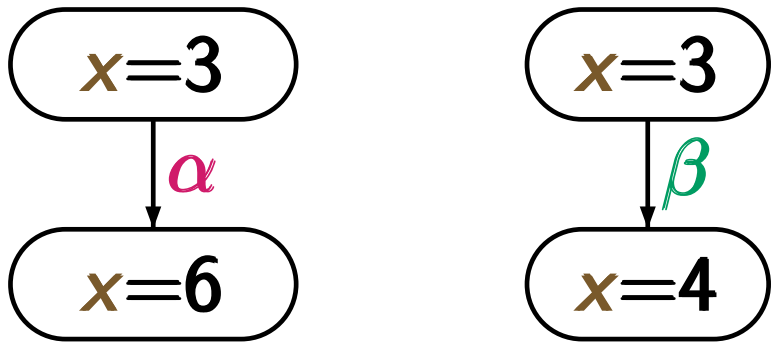
light 1 ||| light 2



Dependent actions

dependent actions $\alpha \mathrel{\widehat{=}} x{:=}2x$ and $\beta \mathrel{\widehat{=}} x{:=}x{+}1$

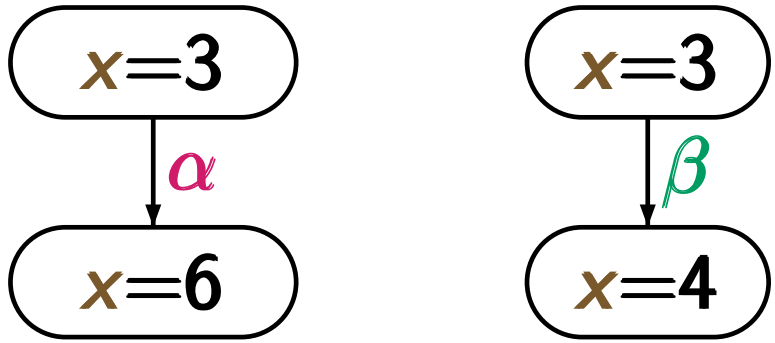
representations in transition systems



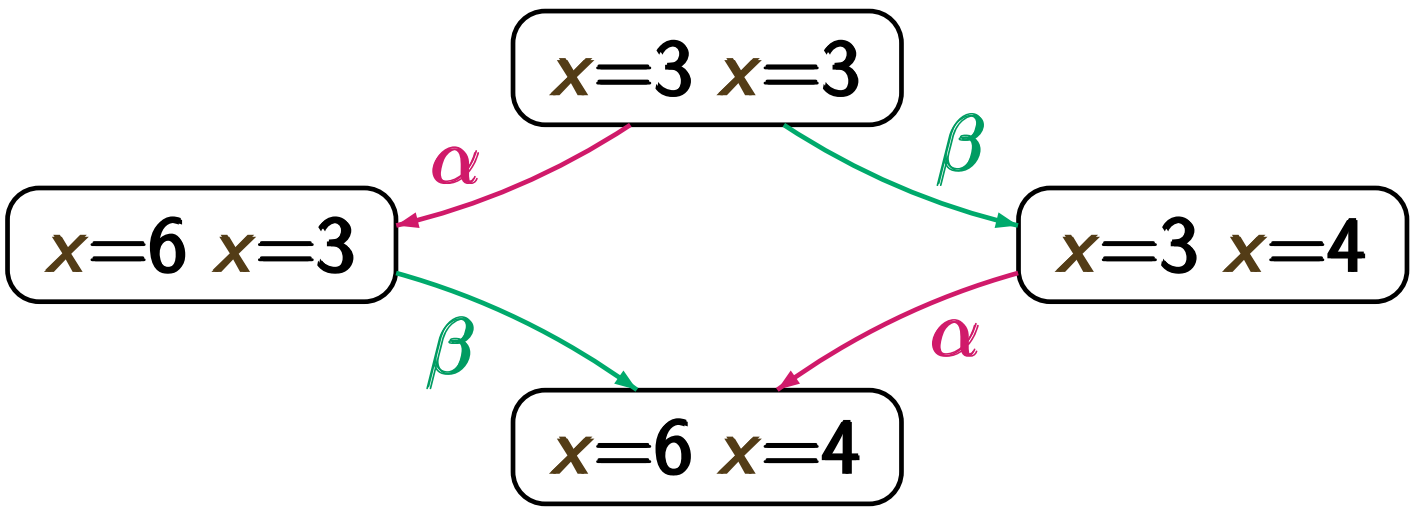
Dependent actions

dependent actions $\alpha \mathrel{\widehat{=}} x{:=}2x$ and $\beta \mathrel{\widehat{=}} x{:=}x{+}1$

representations in transition systems



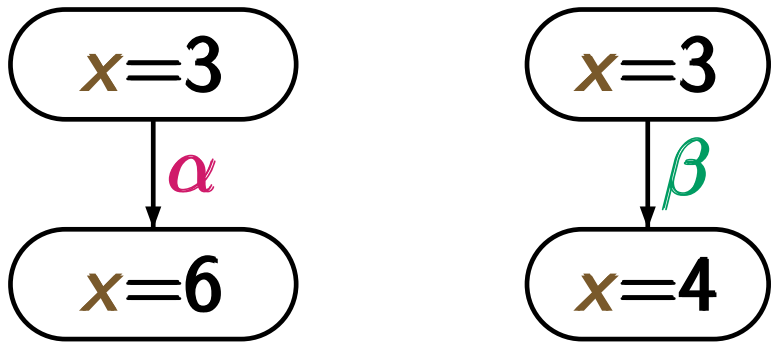
interleaving operator $|||$



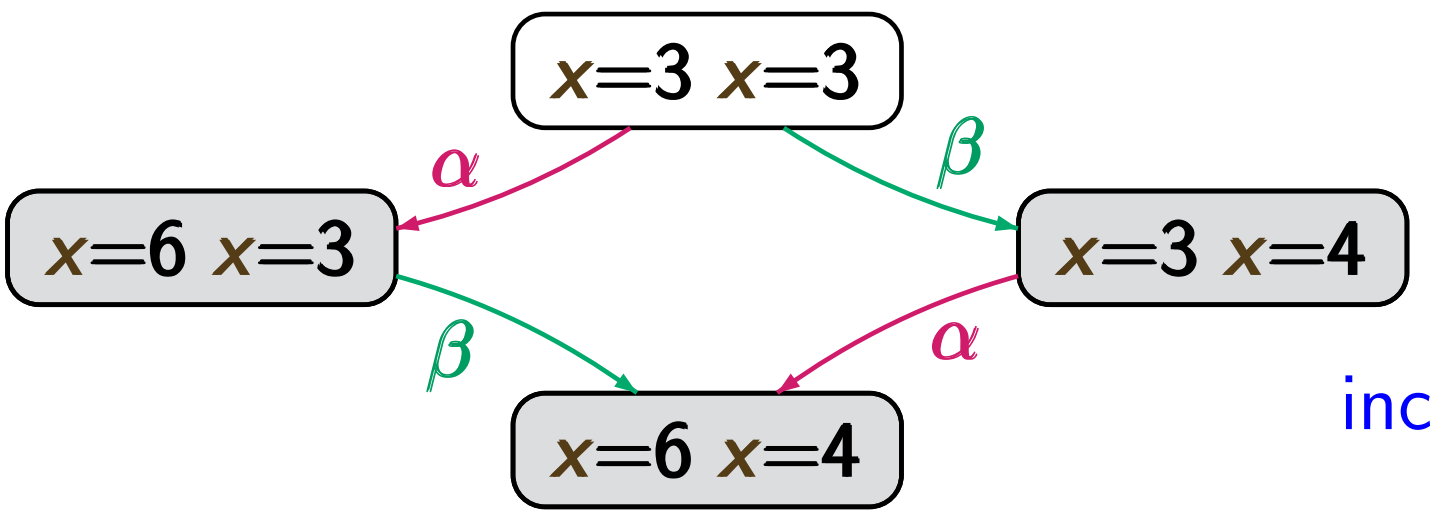
Interleaving fails for dependent actions

dependent actions $\alpha \mathrel{\widehat{=}} x{:=}2x$ and $\beta \mathrel{\widehat{=}} x{:=}x{+}1$

representations in transition systems



interleaving operator $|||$ for transition systems "fails"



inconsistent states

Interleaving for program graphs

Interleaving for program graphs

. . . for modeling parallel systems with
subprocesses communicating via shared variables

Interleaving for program graphs

program graph $\mathcal{P}_1$
$(Loc_1, \ldots, \hookrightarrow_1, \ldots)$

program graph $\mathcal{P}_2$
$(Loc_2, \ldots, \hookrightarrow_2, \ldots)$

interleaving operator

$$\mathcal{P}_1 \,|||\, \mathcal{P}_2 \;=\; (Loc_1 \times Loc_2, \ldots, \hookrightarrow, \ldots)$$

Interleaving for program graphs

program graph $\mathcal{P}_1$
$(Loc_1, \ldots, \hookrightarrow_1, \ldots)$

program graph $\mathcal{P}_2$
$(Loc_2, \ldots, \hookrightarrow_2, \ldots)$

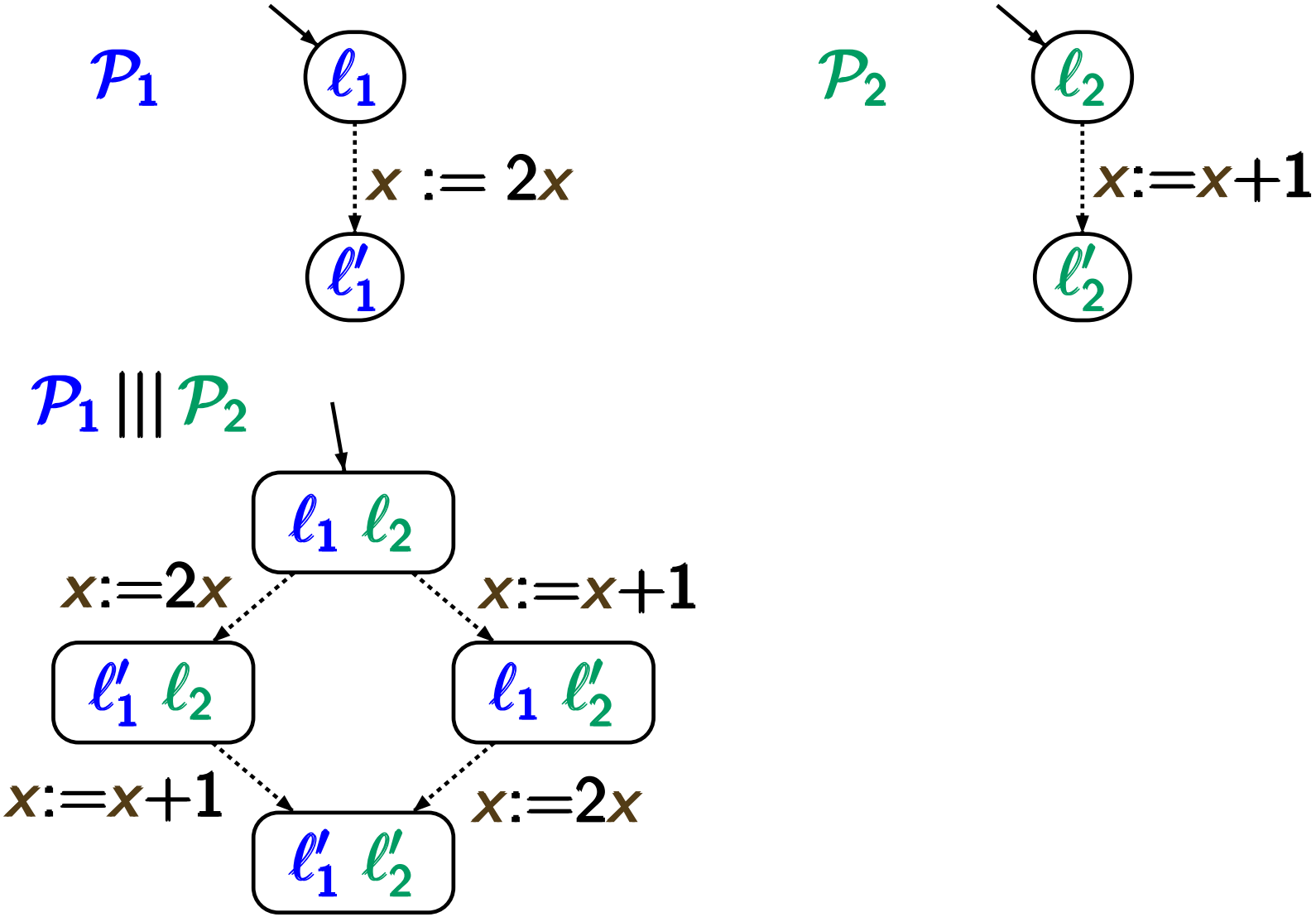
interleaving operator

$$\mathcal{P}_1 \,|||\, \mathcal{P}_2 \;=\; (Loc_1 \times Loc_2, \ldots, \hookrightarrow, \ldots)$$

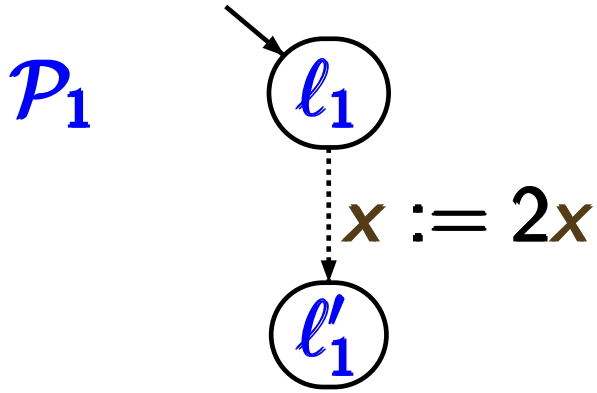
$$\frac{\ell_1 \overset{g\,:\,\alpha}{\hookrightarrow_1} \ell_1'}{\langle \ell_1, \ell_2 \rangle \overset{g\,:\,\alpha}{\hookrightarrow} \langle \ell_1', \ell_2 \rangle} \qquad \frac{\ell_2 \overset{g\,:\,\alpha}{\hookrightarrow_2} \ell_2'}{\langle \ell_1, \ell_2 \rangle \overset{g\,:\,\alpha}{\hookrightarrow} \langle \ell_1, \ell_2' \rangle}$$

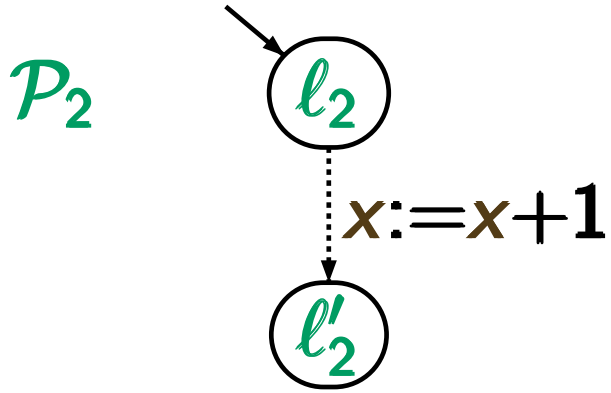
Example: interleaving for PG

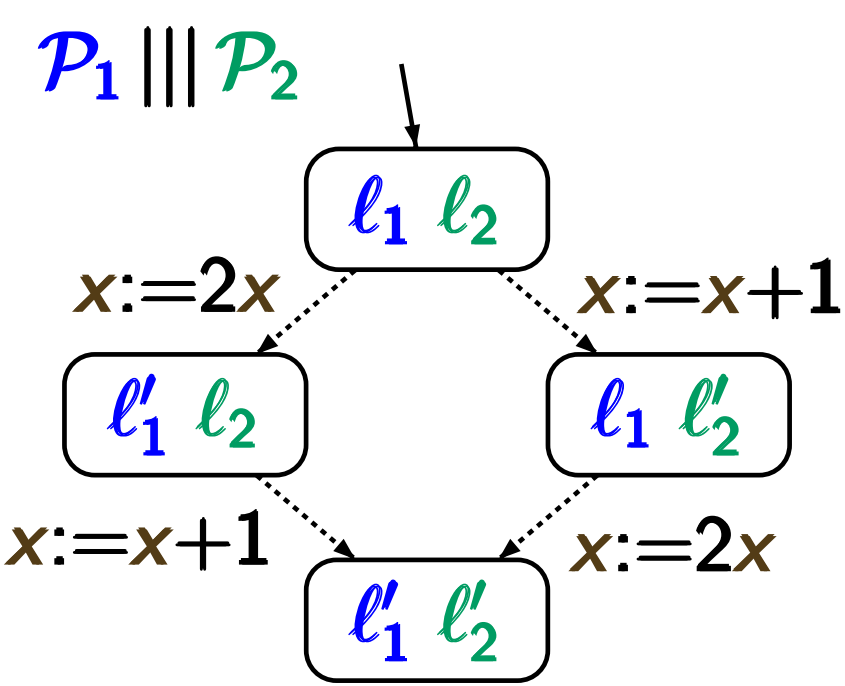
Example: interleaving for PG

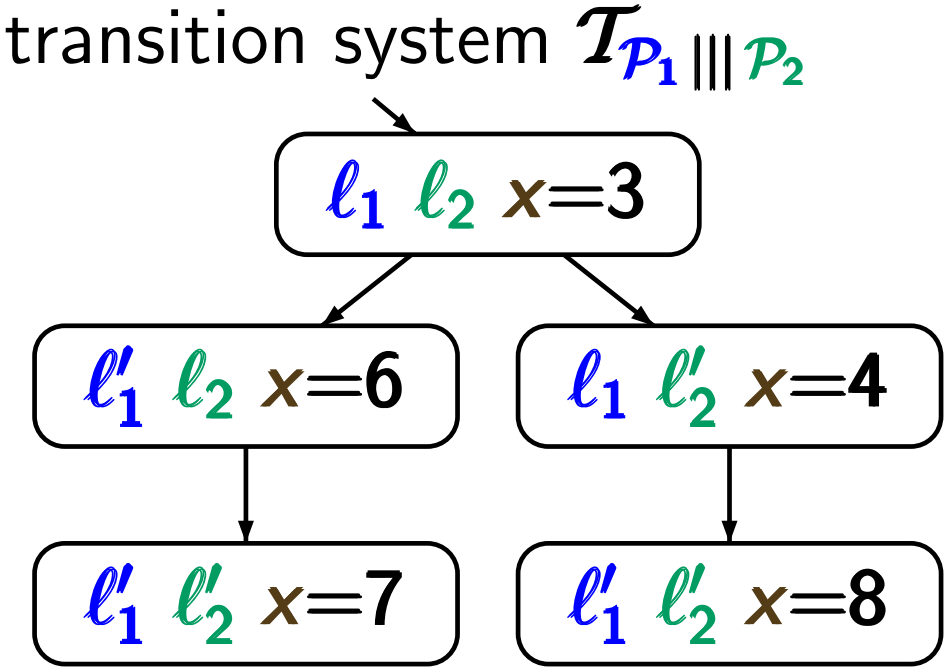


Example: interleaving for PG

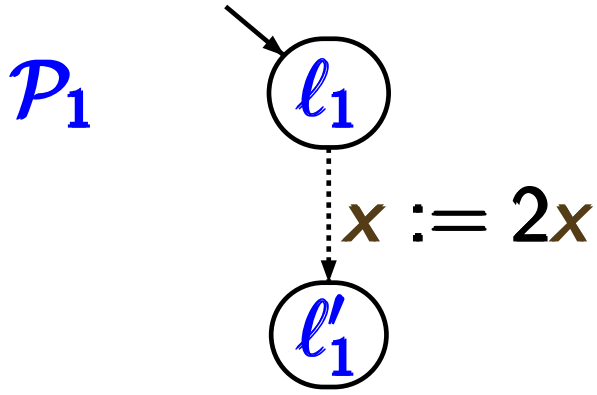


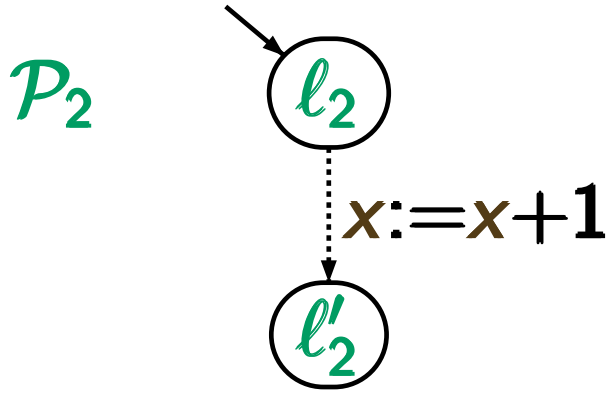


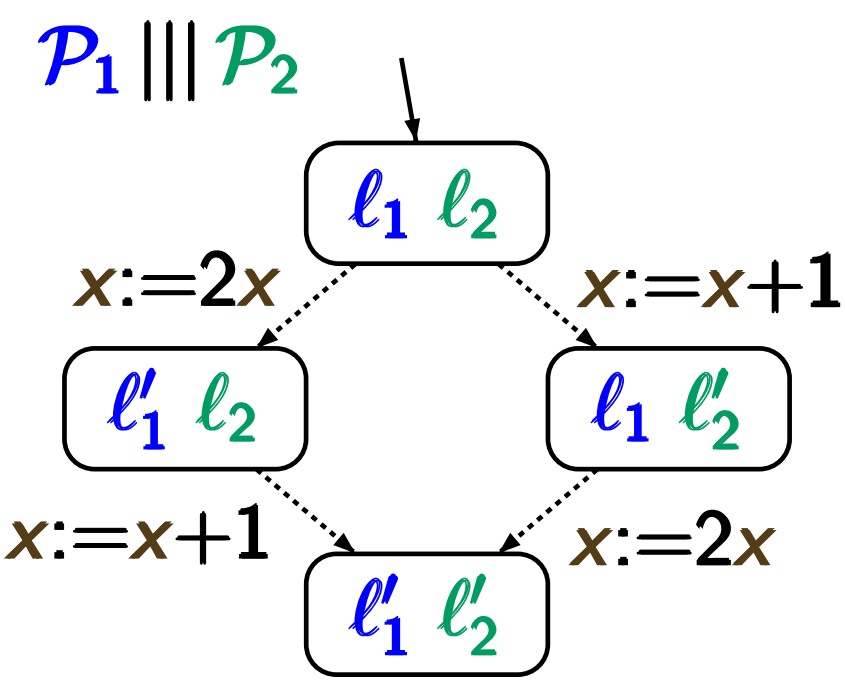


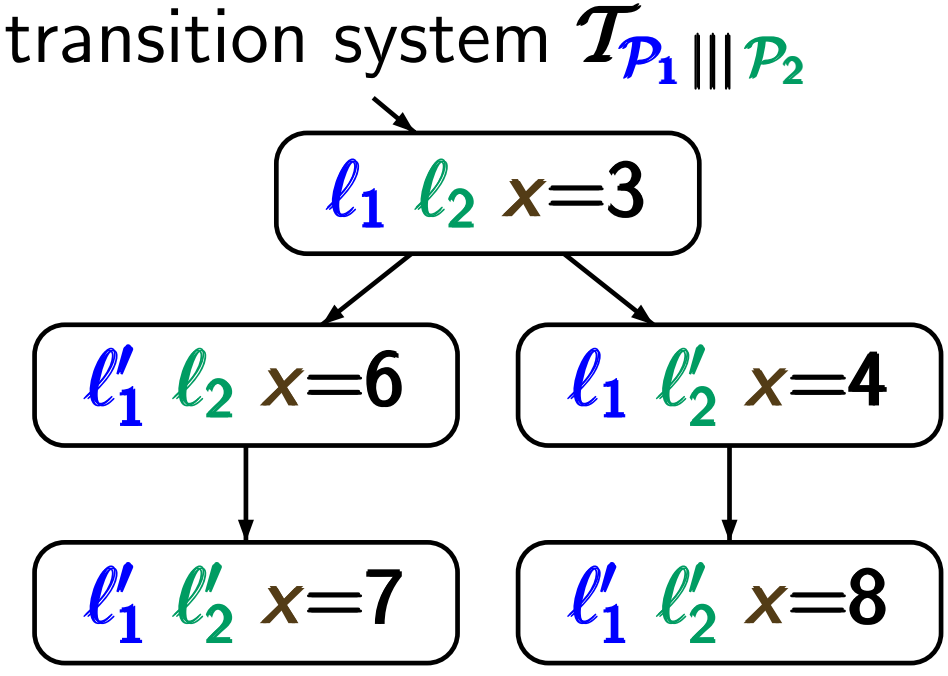


Example: interleaving for PG



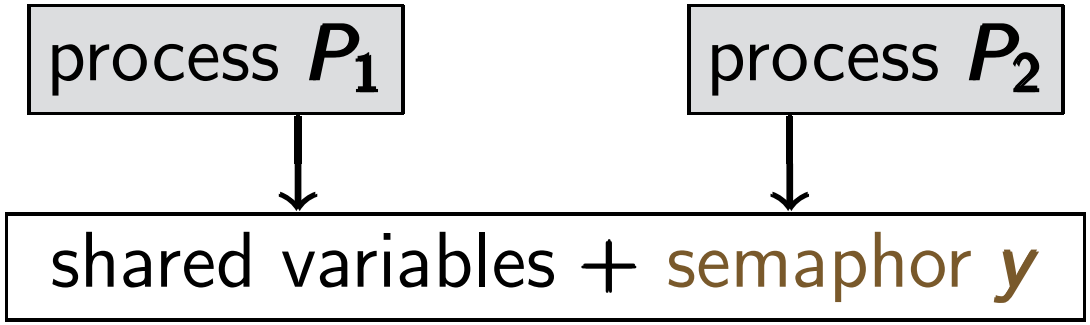




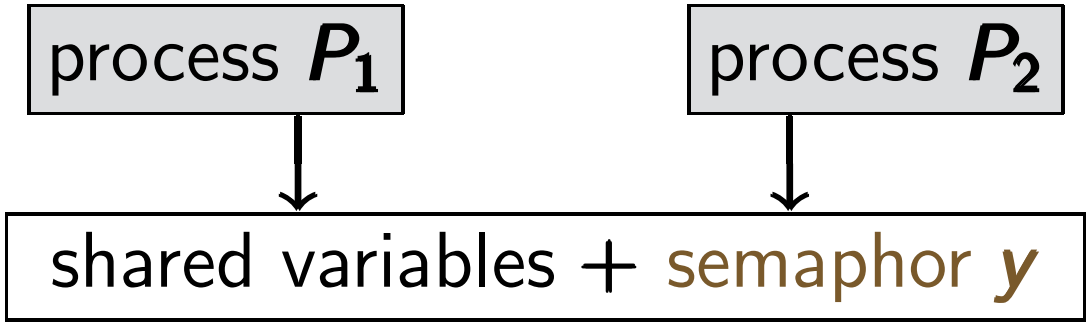


note: $\mathcal{T}_{\mathcal{P}_1 \,|||\, \mathcal{P}_2} \neq \mathcal{T}_{\mathcal{P}_1} \,|||\, \mathcal{T}_{\mathcal{P}_2}$

Mutual exclusion with semaphore



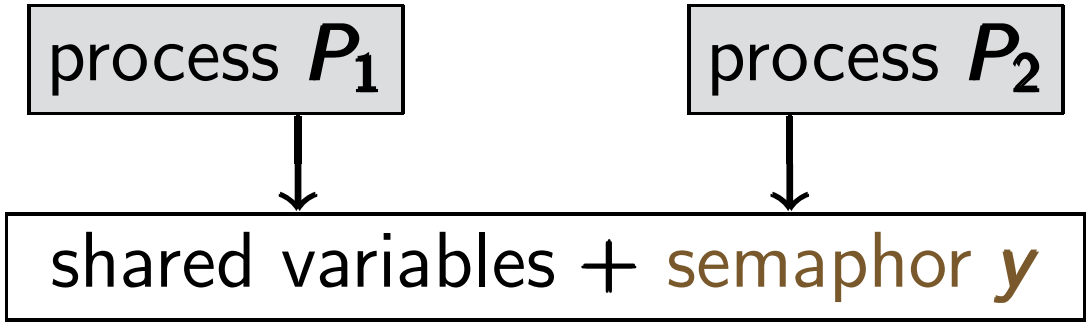
Mutual exclusion with semaphore



protocol for process P_i

```
LOOP FOREVER
  noncritical actions;
  AWAIT y > 0 DO
        y:=y−1
  OD
  critical actions;
  y:=y+1
END LOOP
```

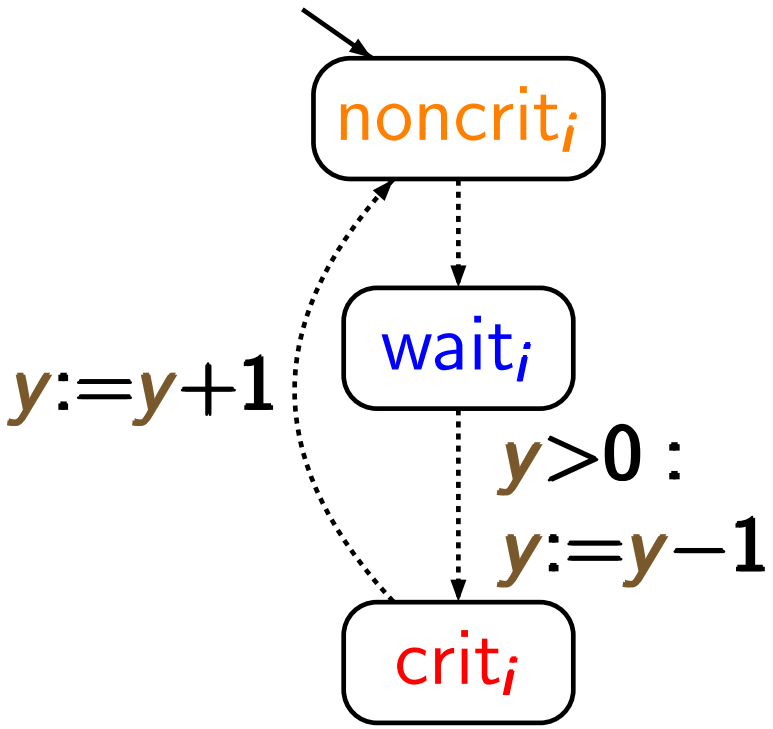
Mutual exclusion with semaphore



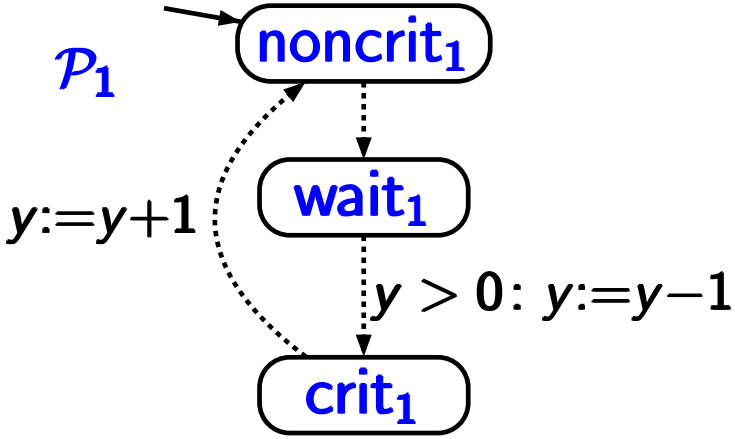
protocol for process P_i

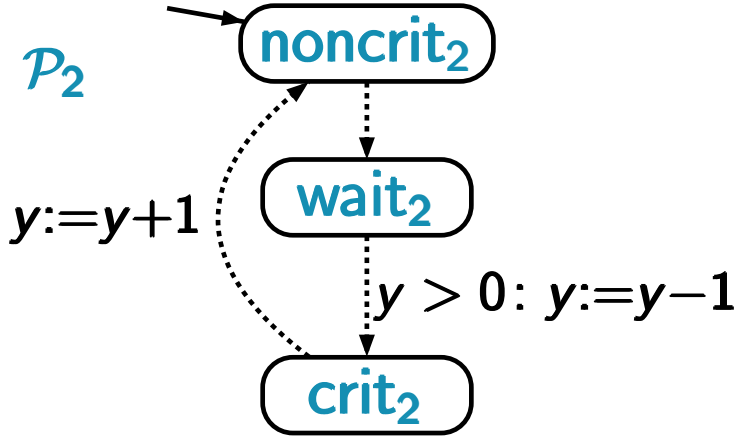
```
LOOP FOREVER
  noncritical actions;
  AWAIT y > 0 DO
        y:=y−1
  OD
  critical actions;
  y:=y+1
END LOOP
```

program graph $\mathcal{P}_i$

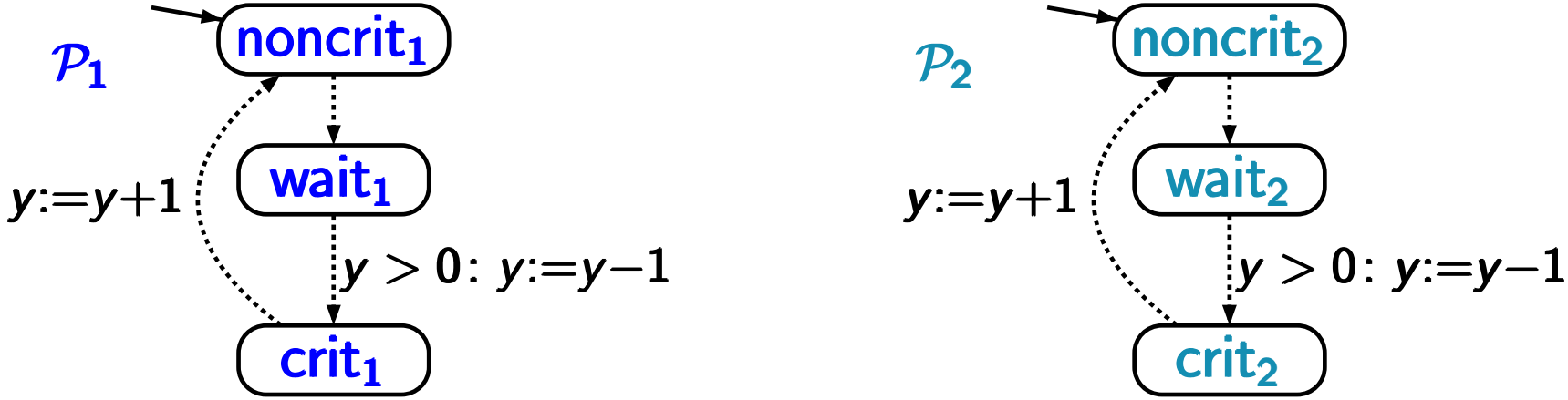


Mutual exclusion with semaphore



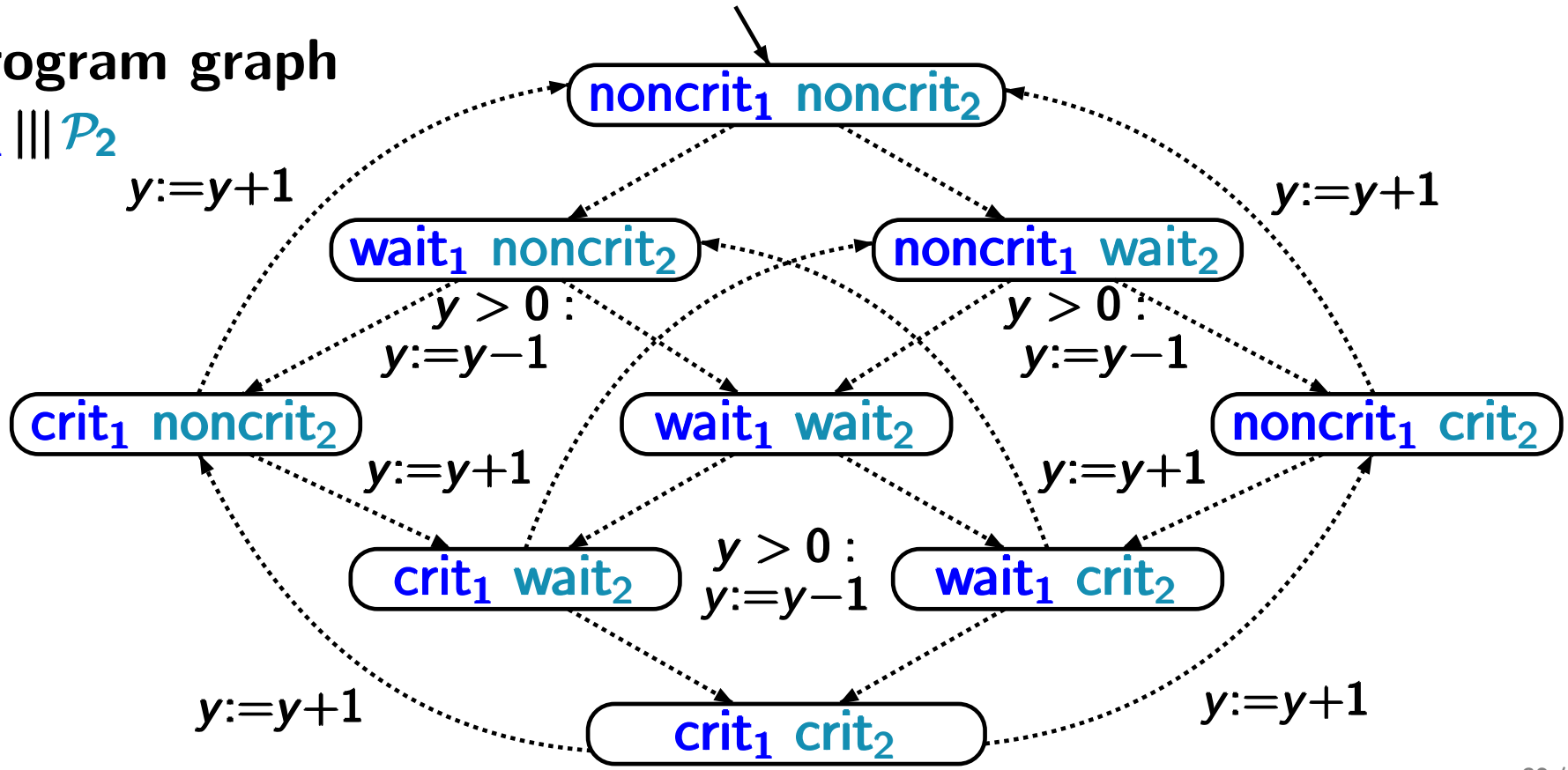


Mutual exclusion with semaphore

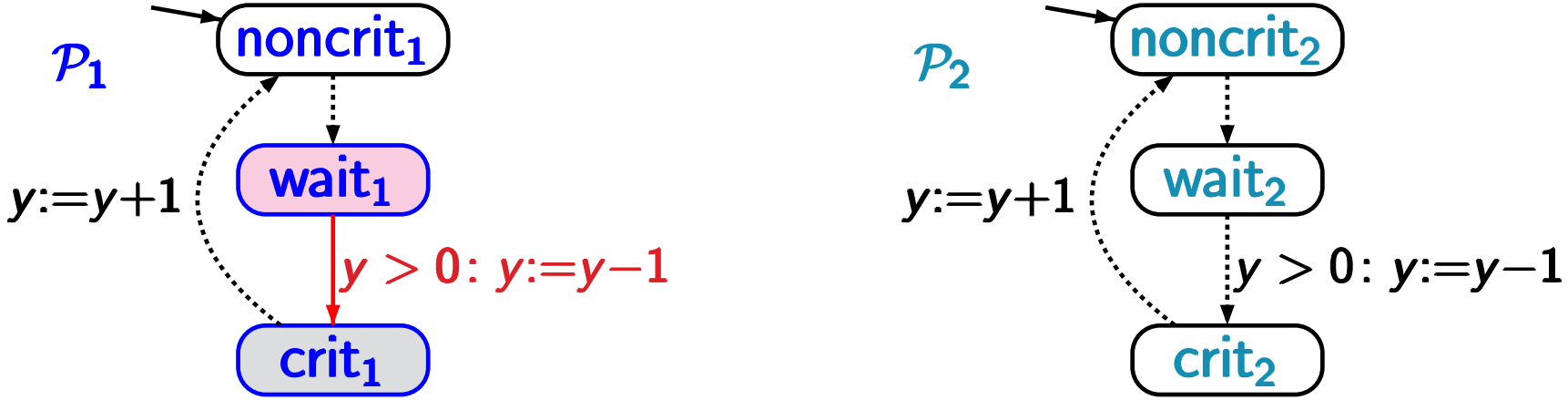


program graph

$\mathcal{P}_1 \,|||\, \mathcal{P}_2$

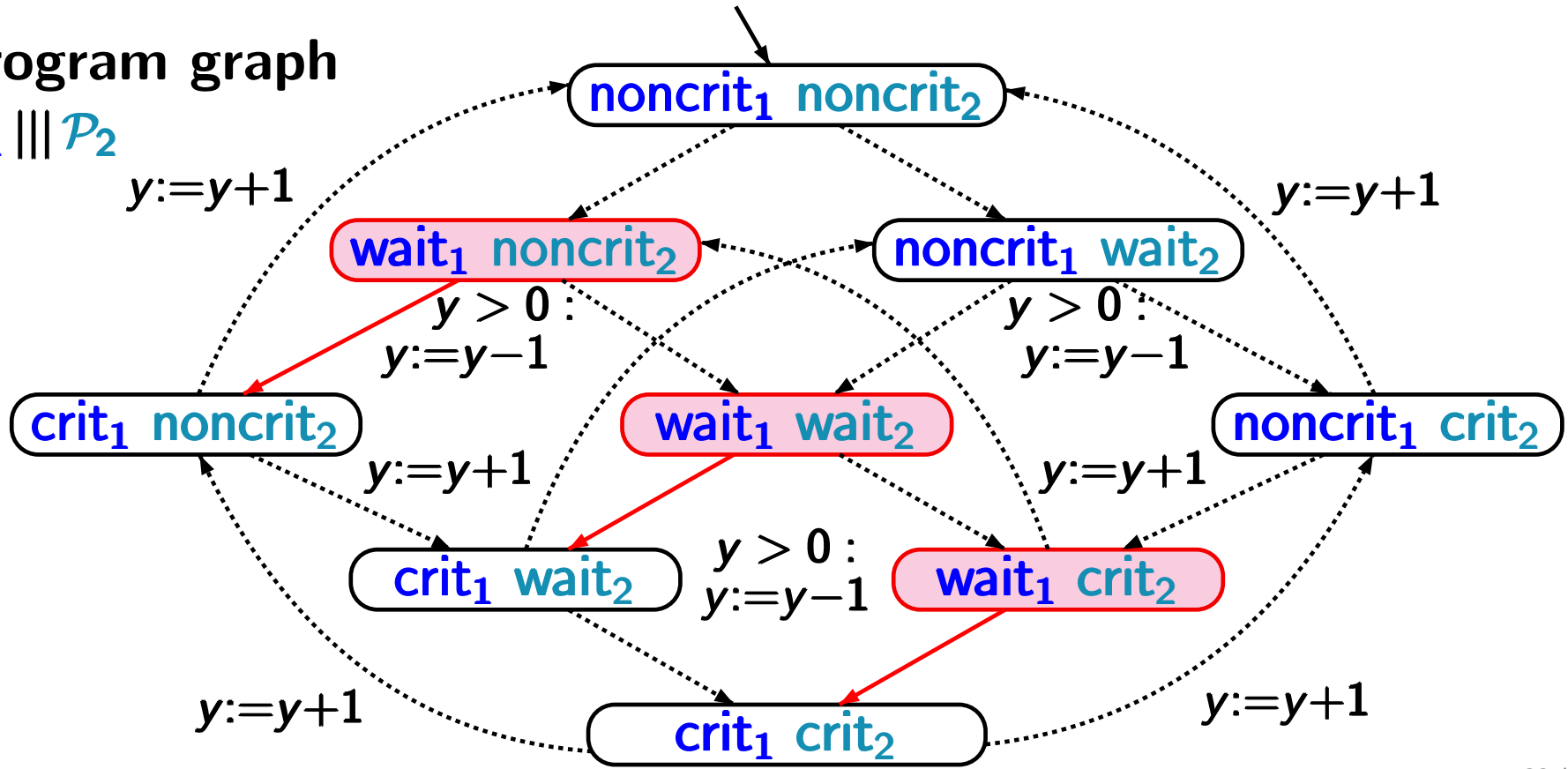


Mutual exclusion with semaphore

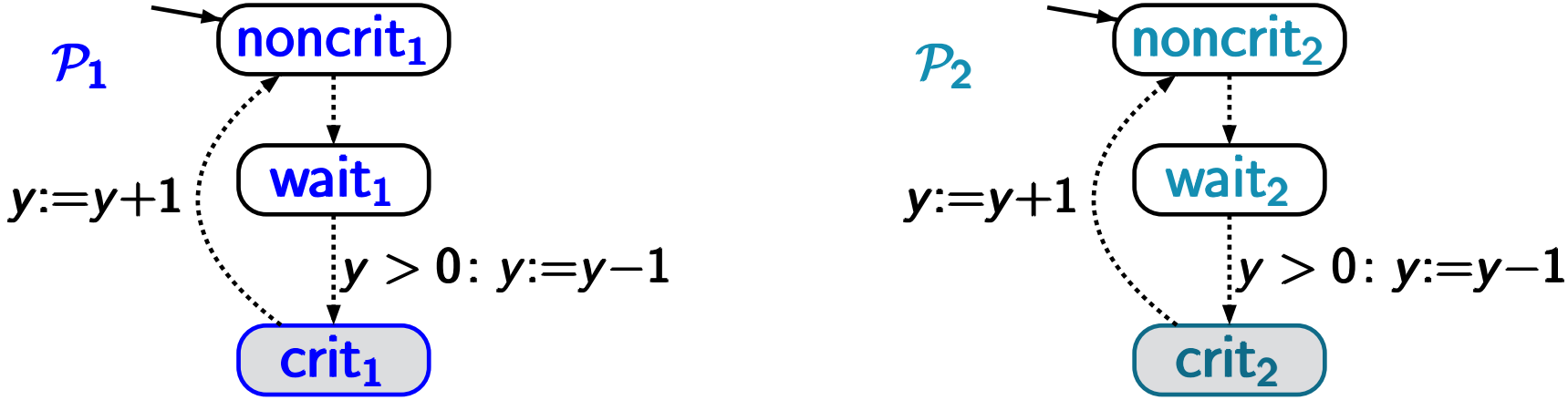


program graph

$\mathcal{P}_1 \,|||\, \mathcal{P}_2$

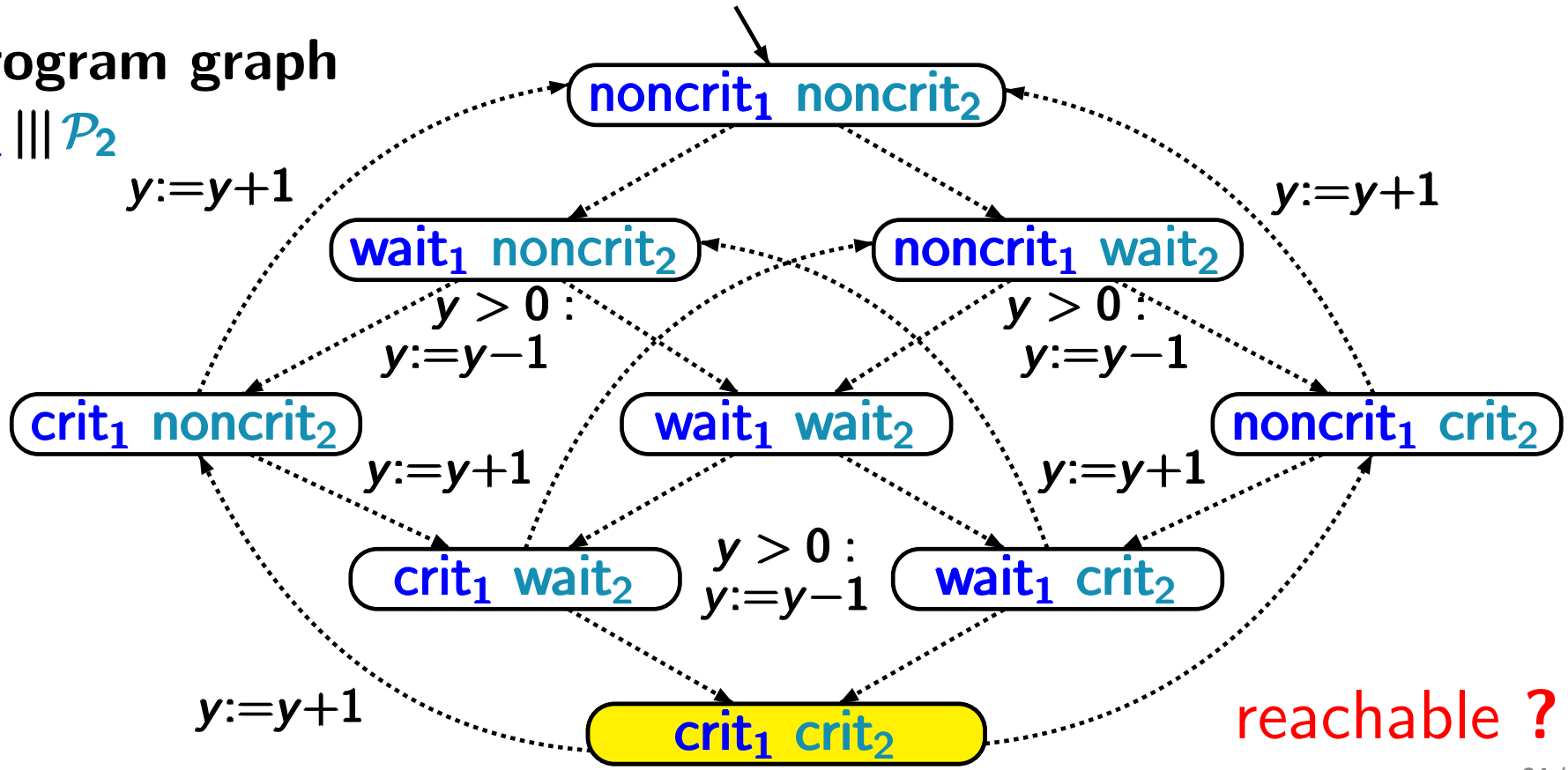


Mutual exclusion with semaphore



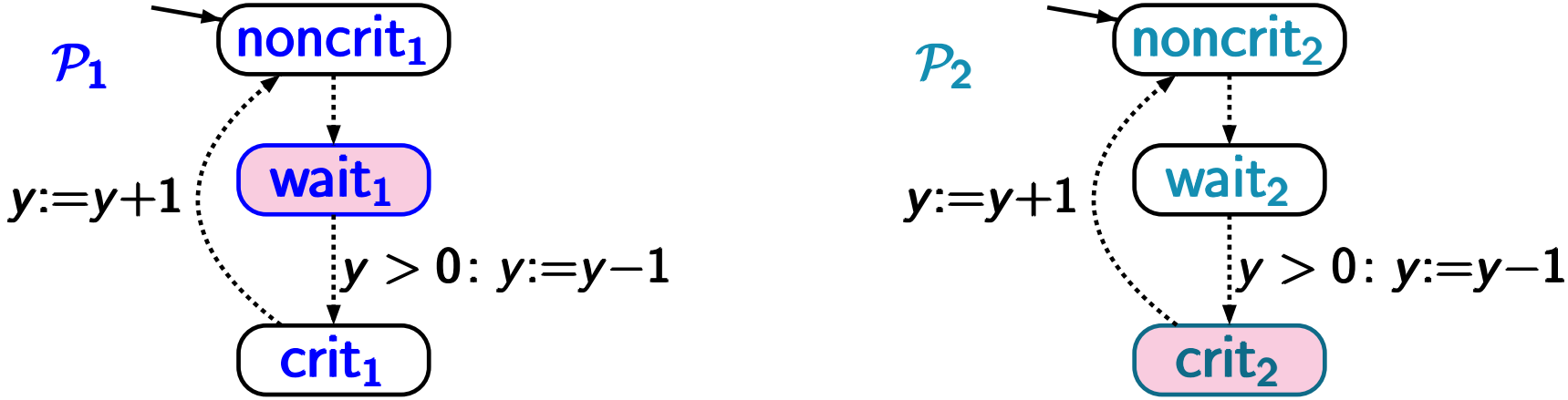
program graph

$\mathcal{P}_1 \,|||\, \mathcal{P}_2$



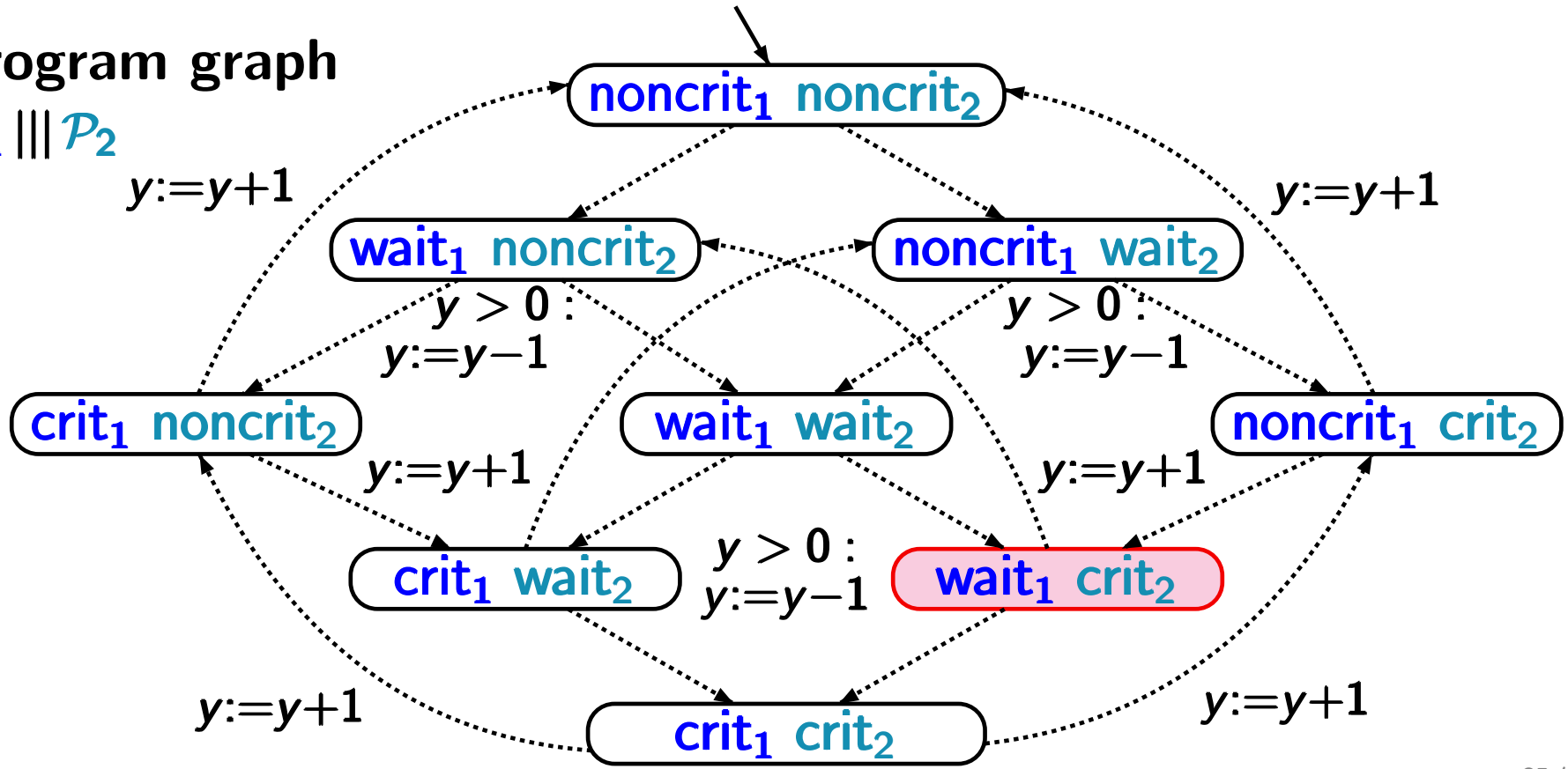
reachable ?

Mutual exclusion with semaphore

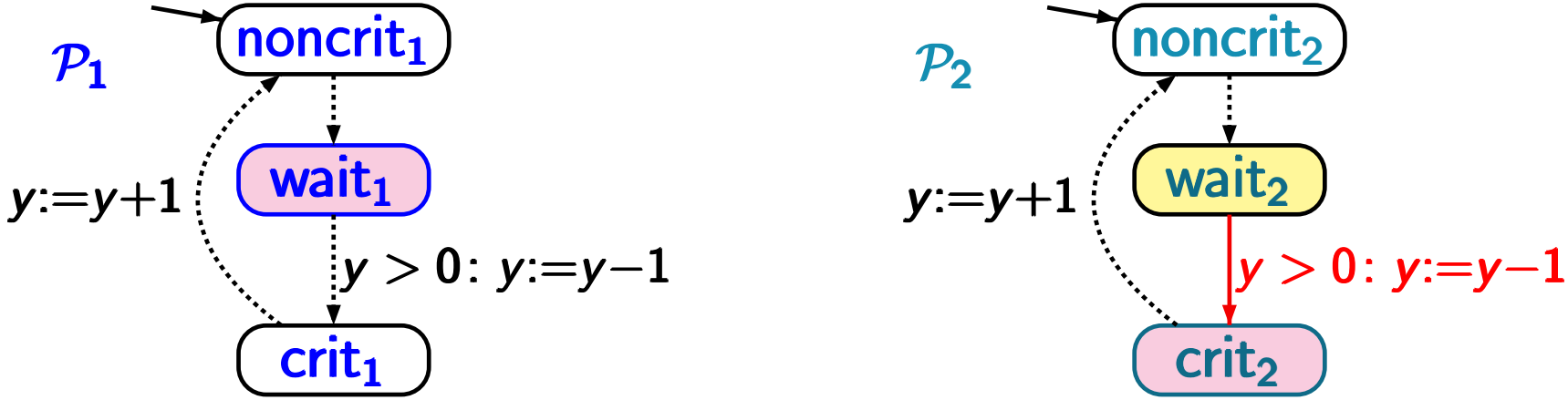


program graph

$\mathcal{P}_1 \,|||\, \mathcal{P}_2$

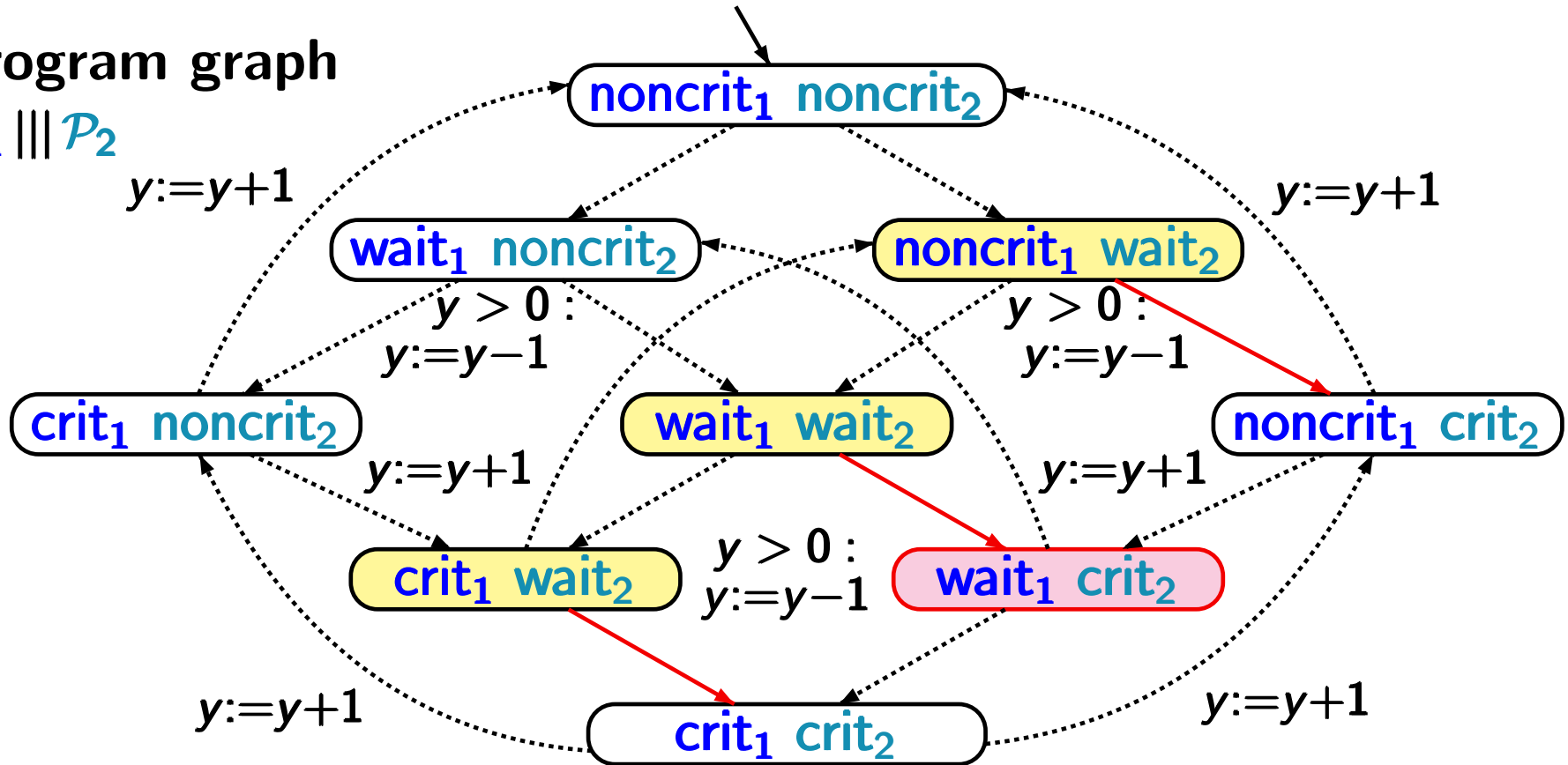


Mutual exclusion with semaphore

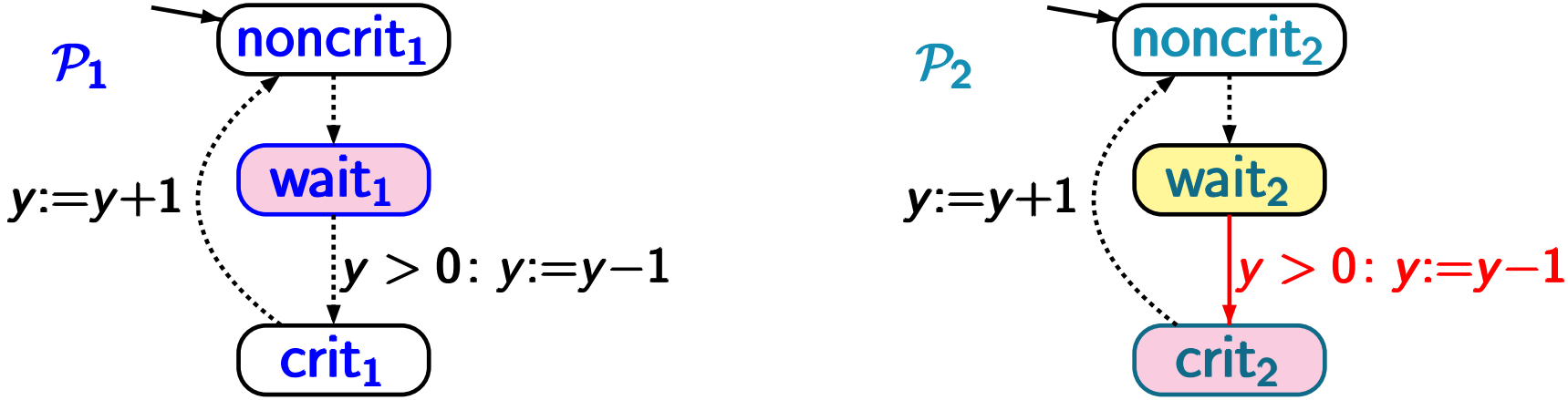


program graph

$\mathcal{P}_1 \,|||\, \mathcal{P}_2$

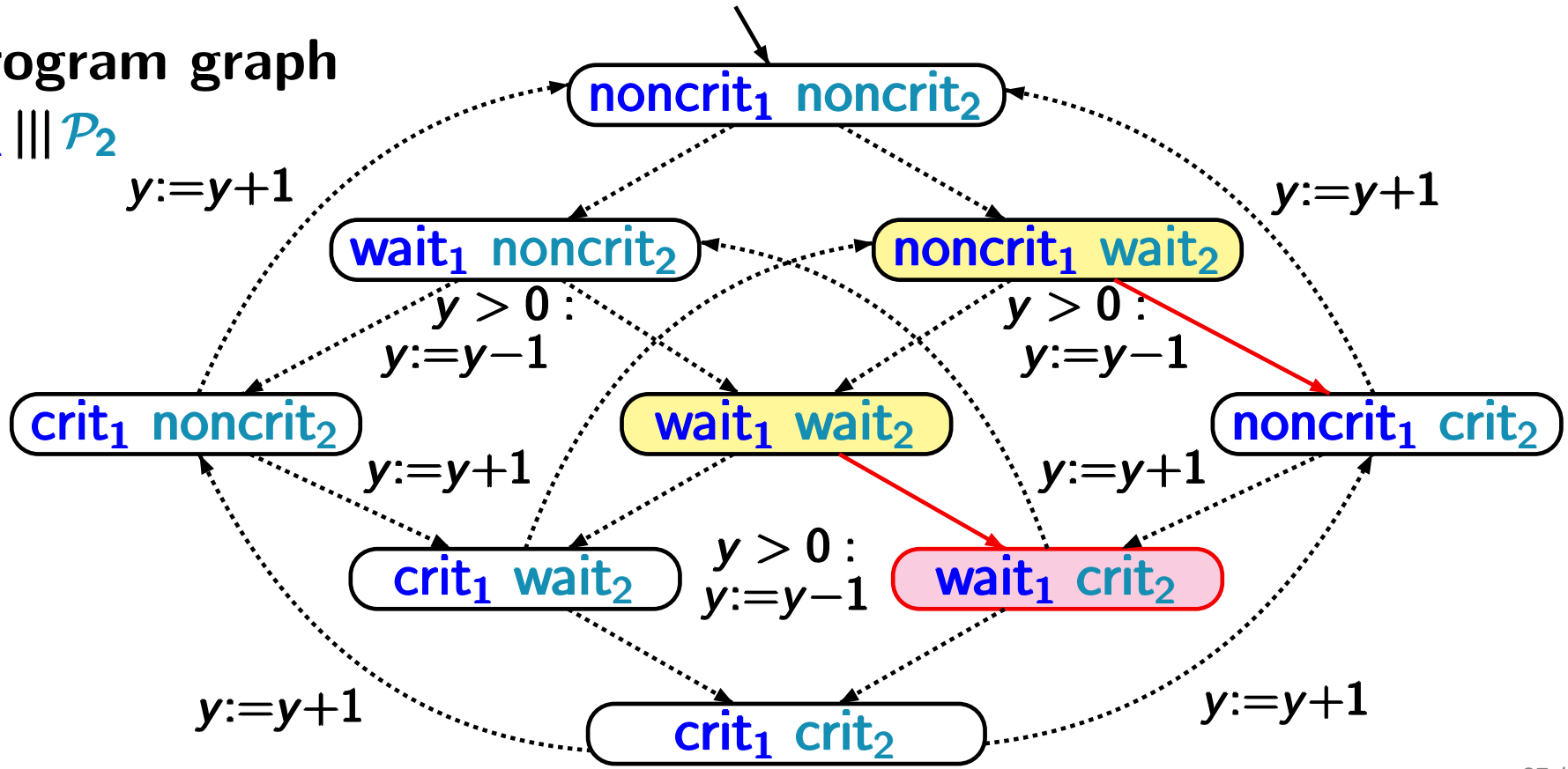


Mutual exclusion with semaphore

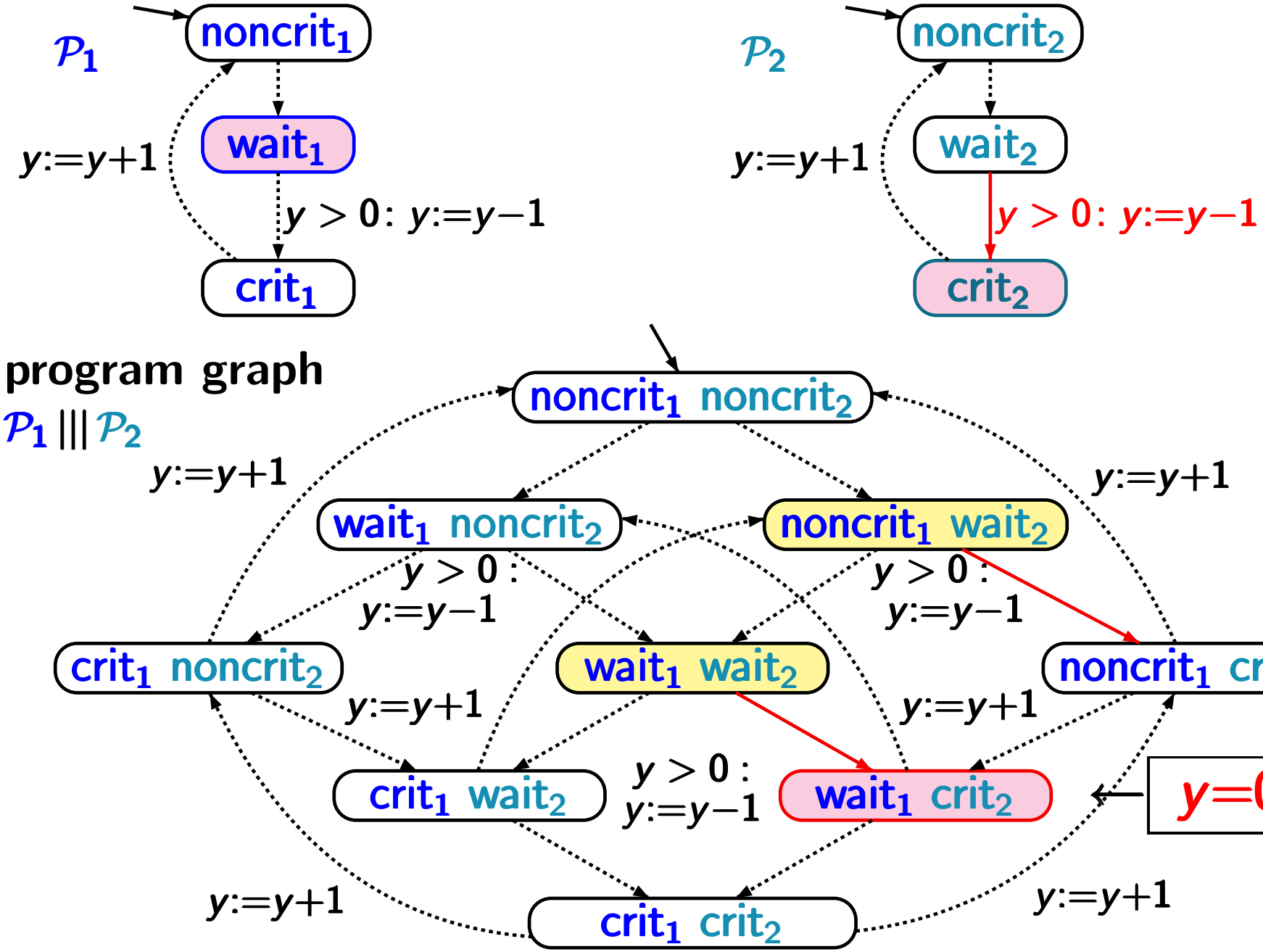


program graph

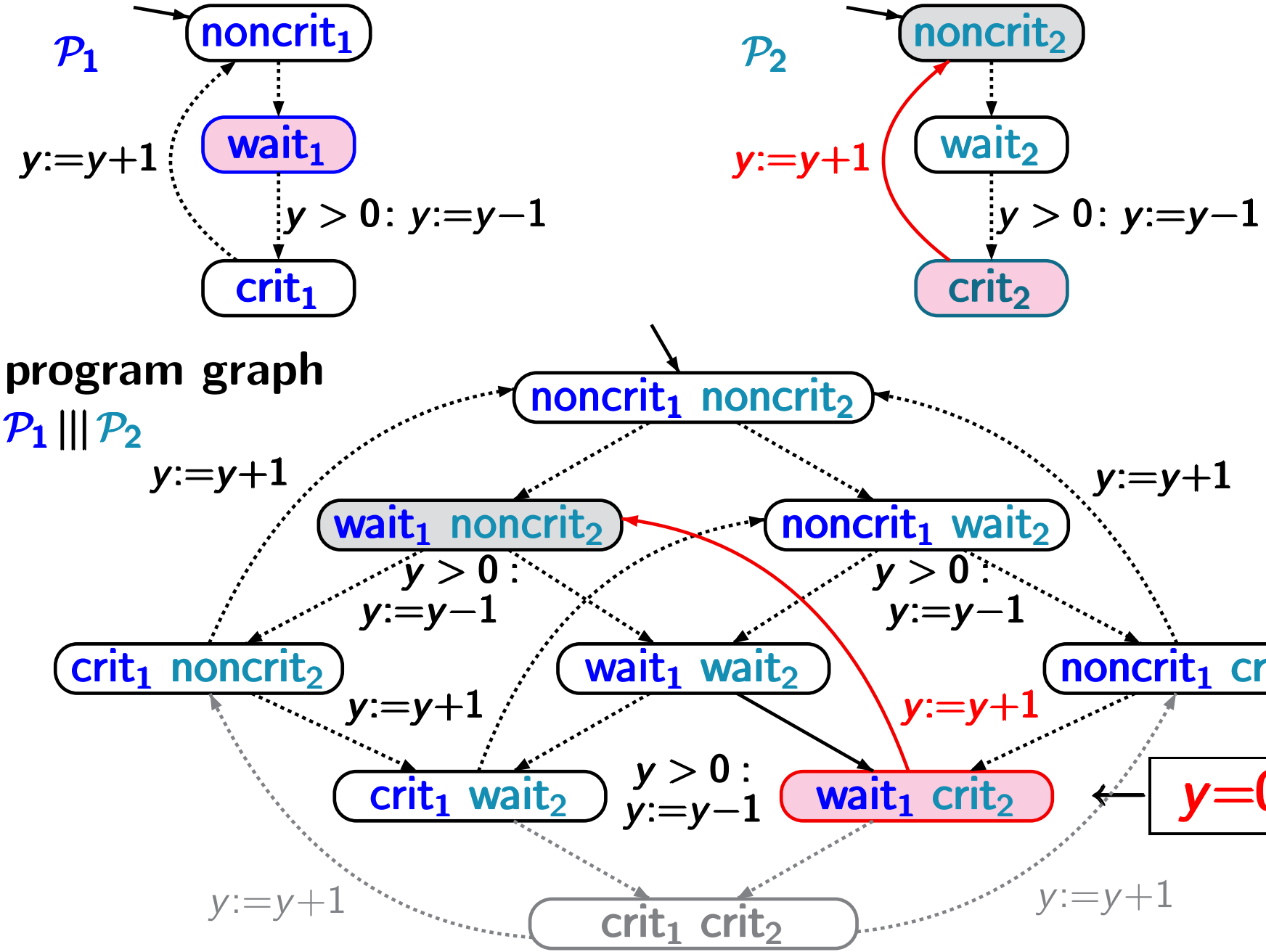
$\mathcal{P}_1 \,|||\, \mathcal{P}_2$



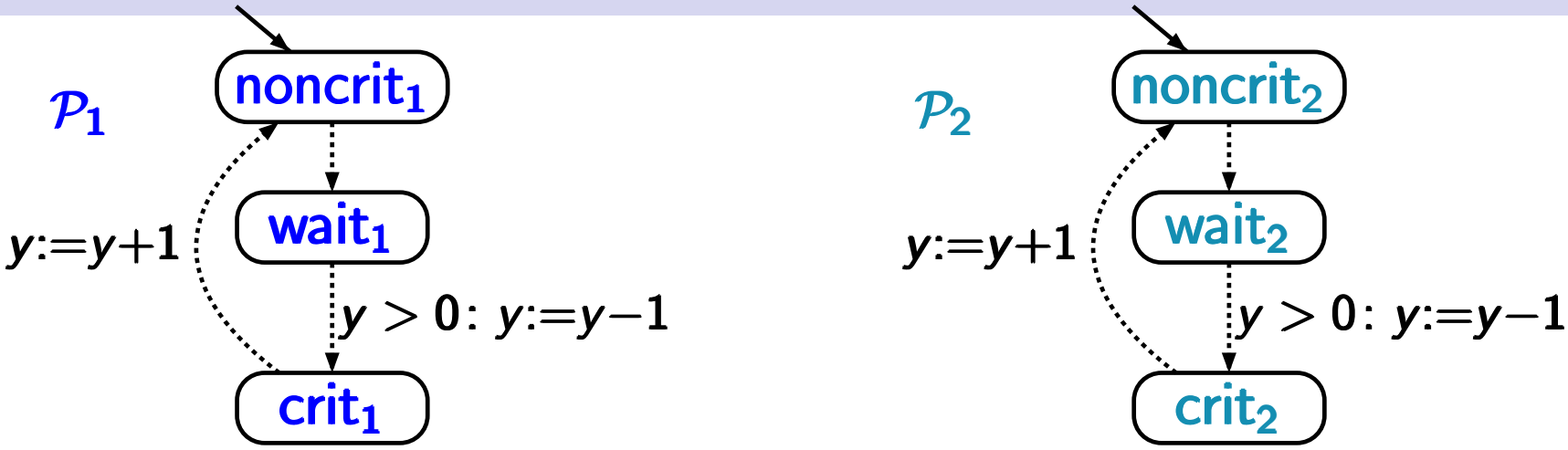
Mutual exclusion with semaphore



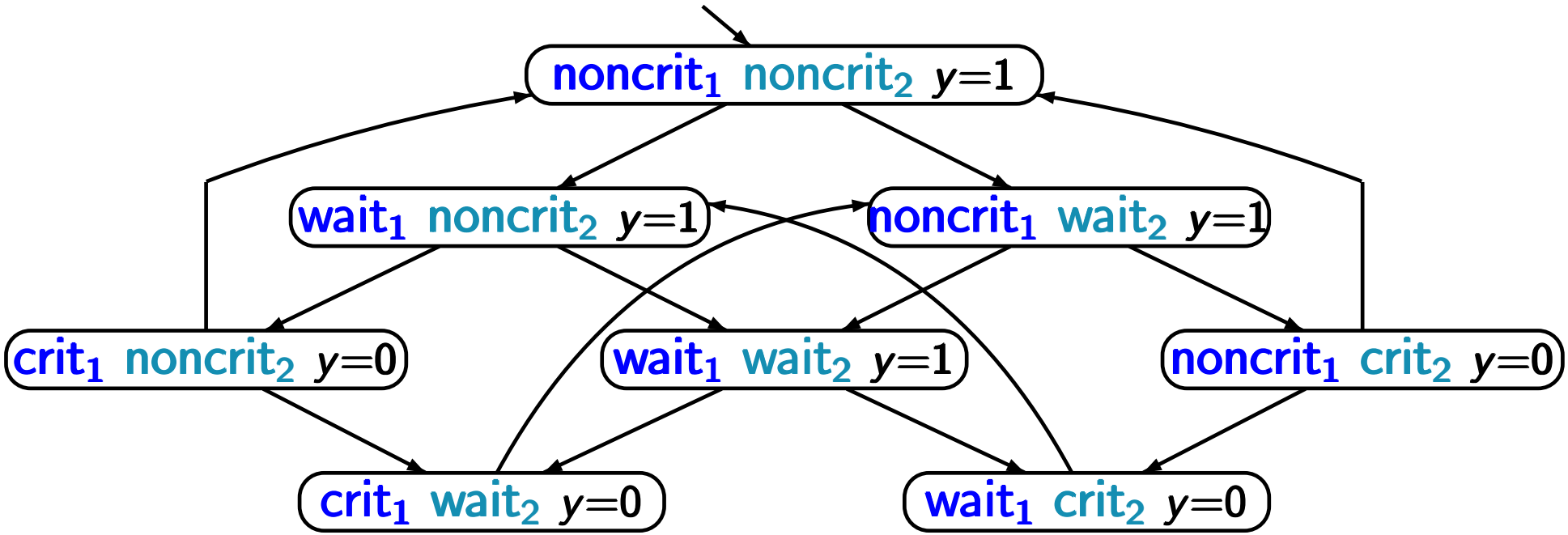
Mutual exclusion with semaphore



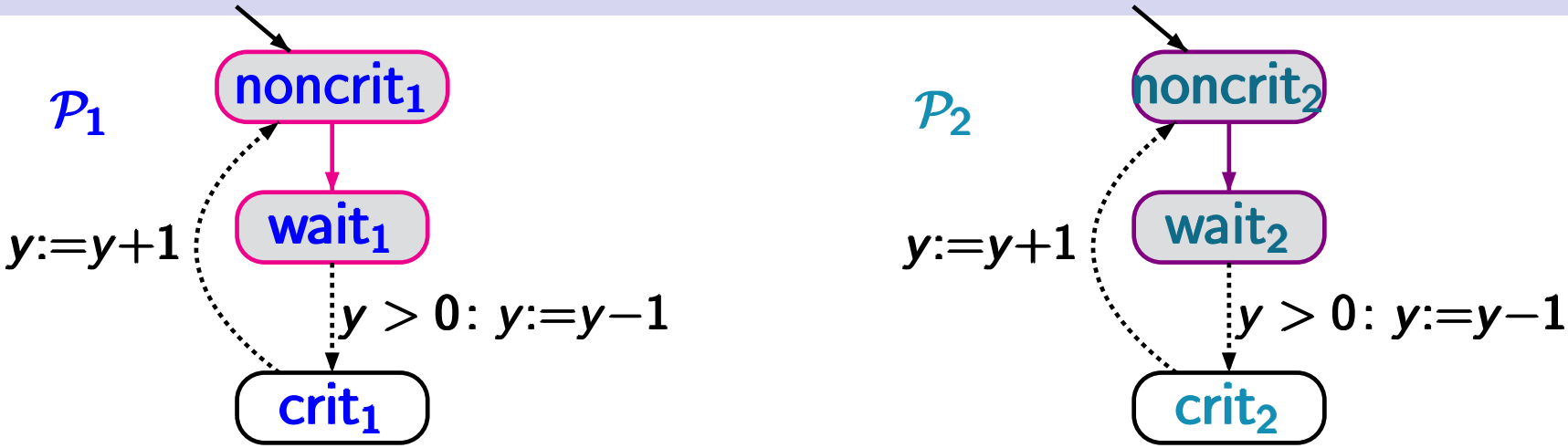
TS for mutual exclusion with semaphore



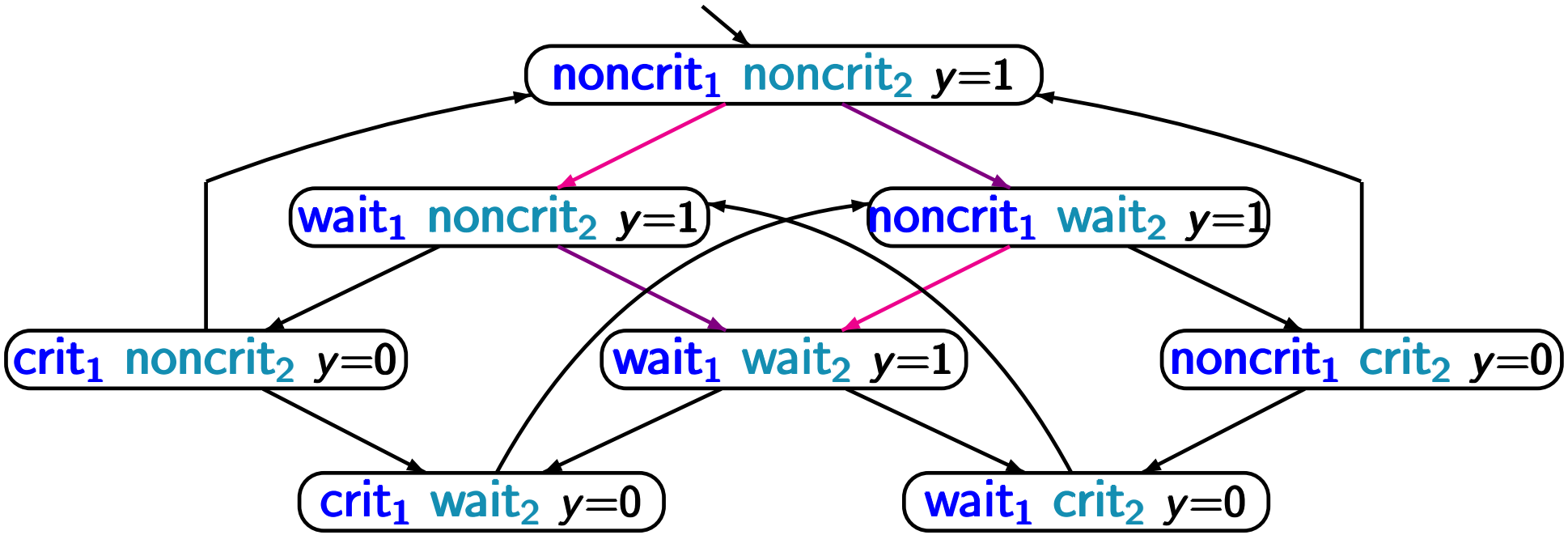
reachable fragment of the transition system $\mathcal{T}_{\mathcal{P}_1 \,|||\, \mathcal{P}_2}$



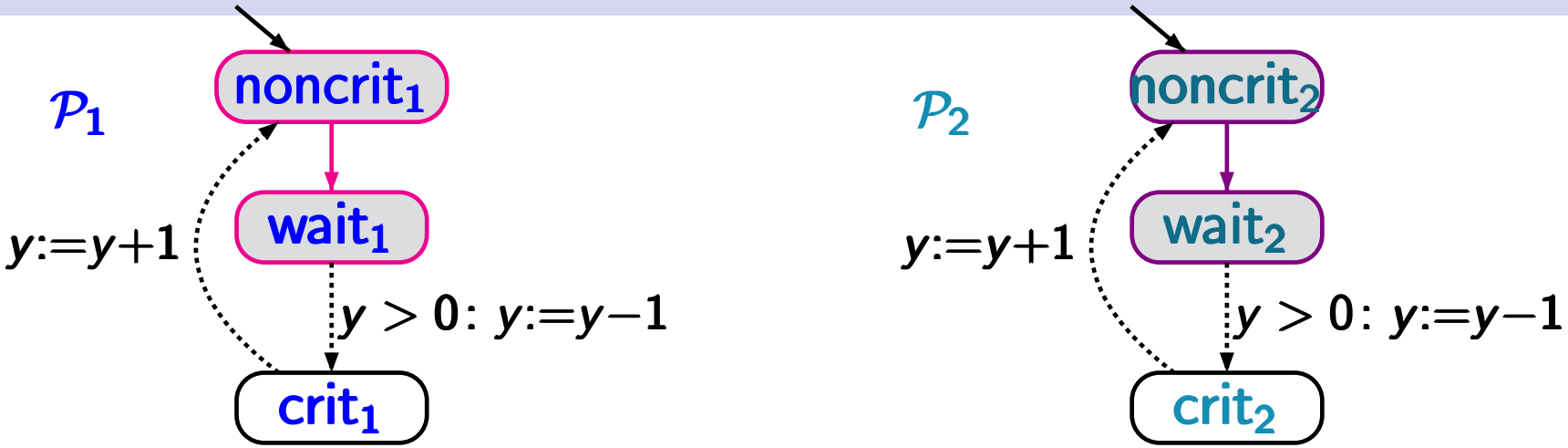
Concurrency of the request actions



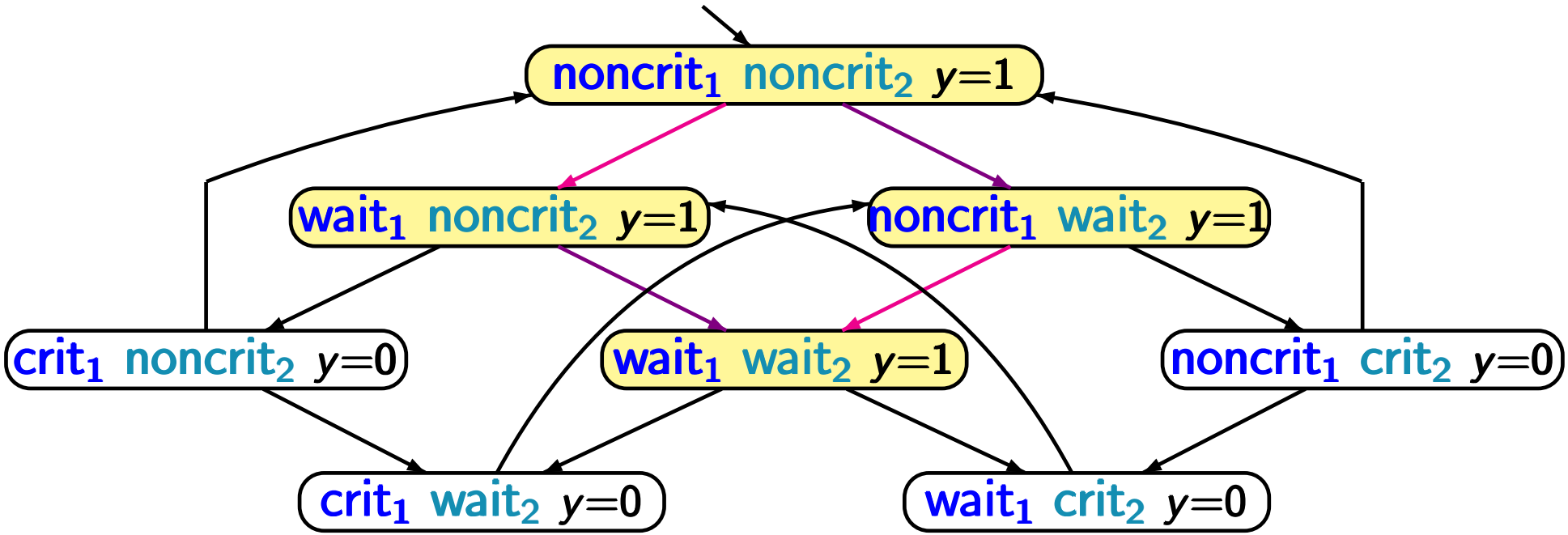
reachable fragment of the transition system $\mathcal{T}_{\mathcal{P}_1 \,|||\, \mathcal{P}_2}$



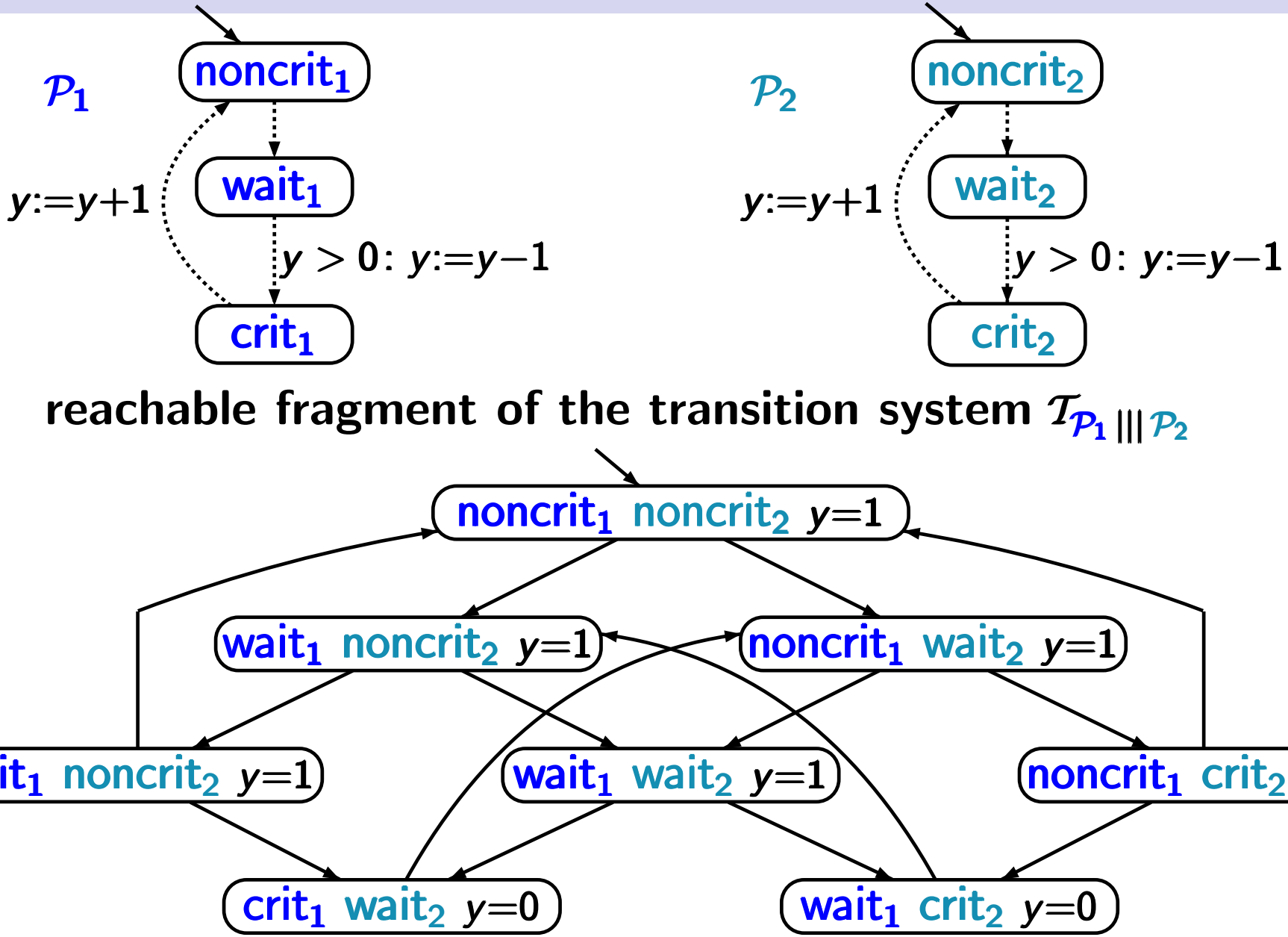
Concurrency of the request actions



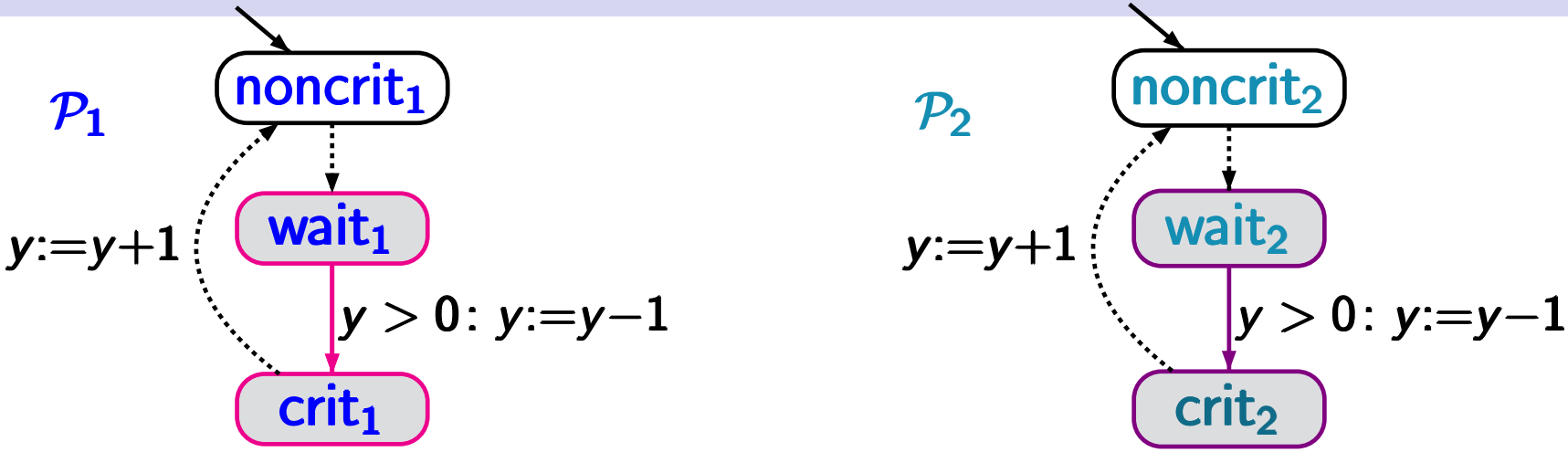
interleaving of the independent request actions



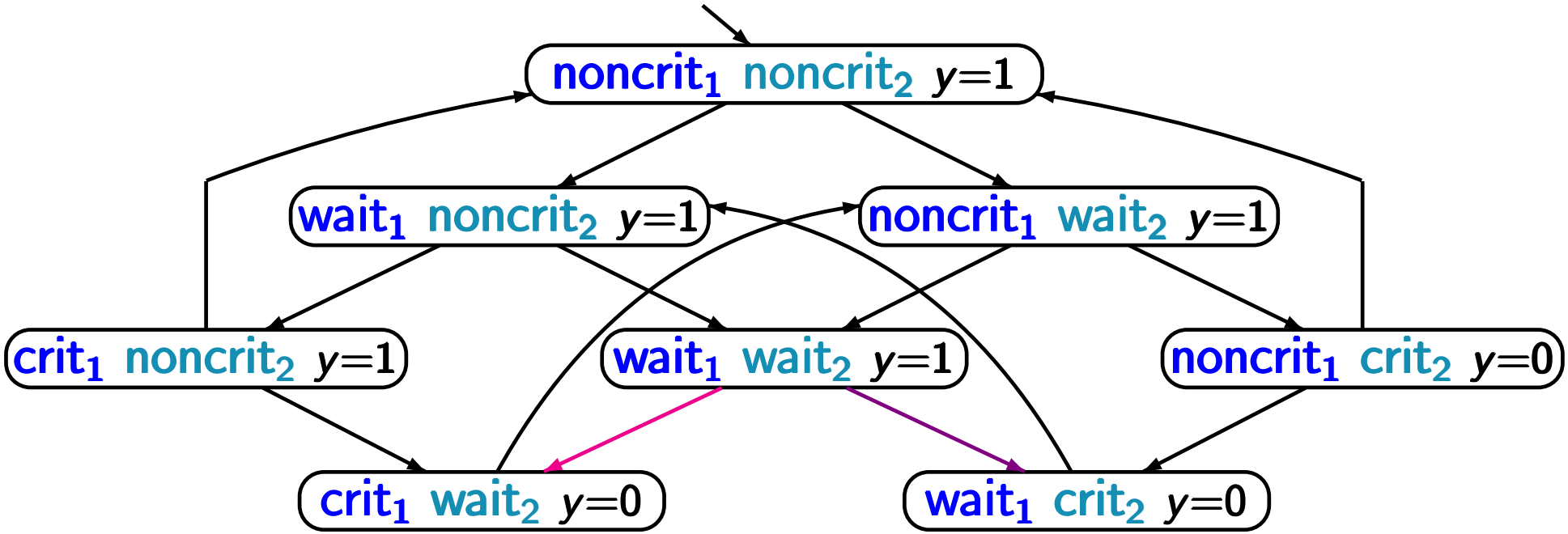
Competition



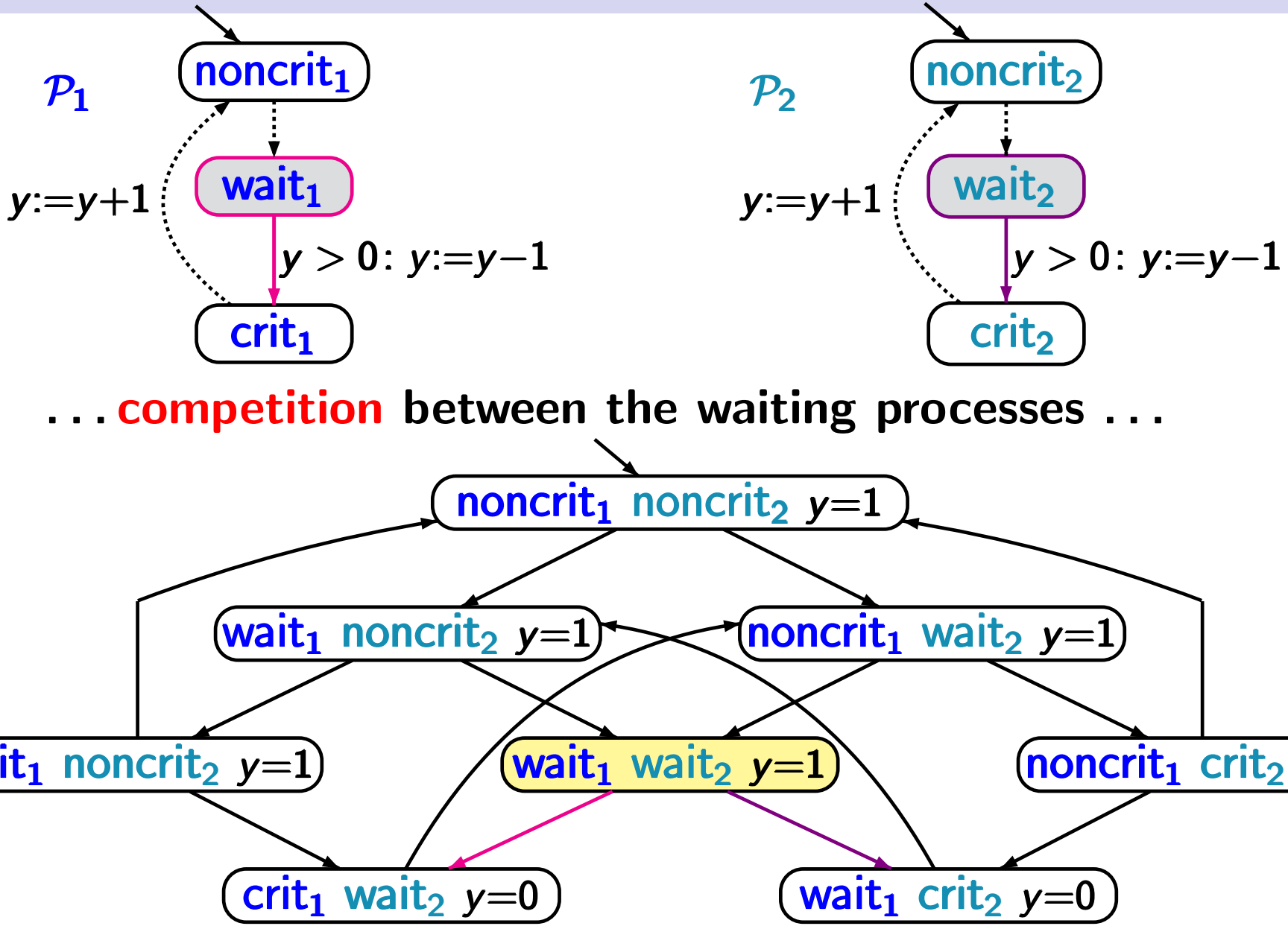
Competition



reachable fragment of the transition system $\mathcal{T}_{\mathcal{P}_1 \,|||\, \mathcal{P}_2}$

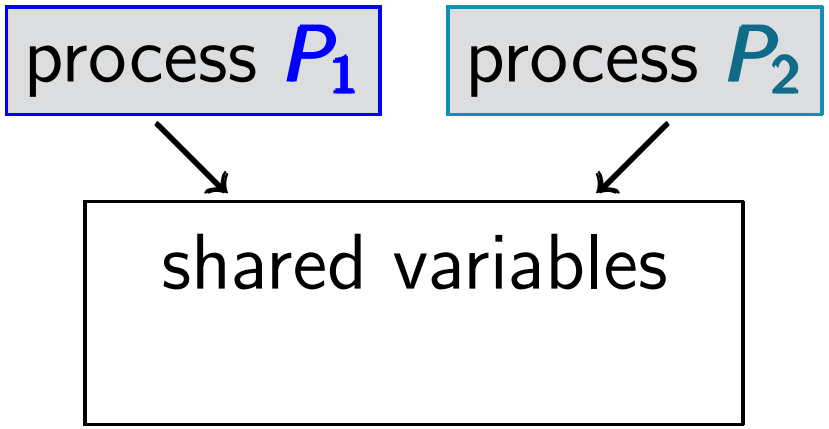


Competition

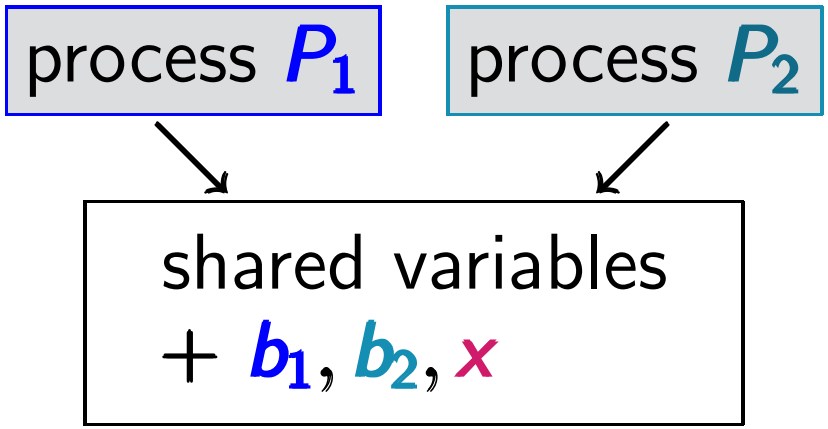


Peterson algorithm for mutual exclusion

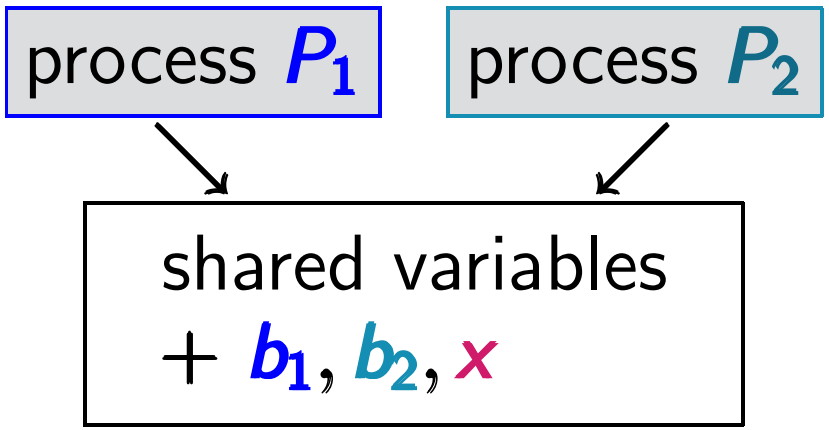
Peterson algorithm for mutual exclusion



Peterson algorithm for mutual exclusion

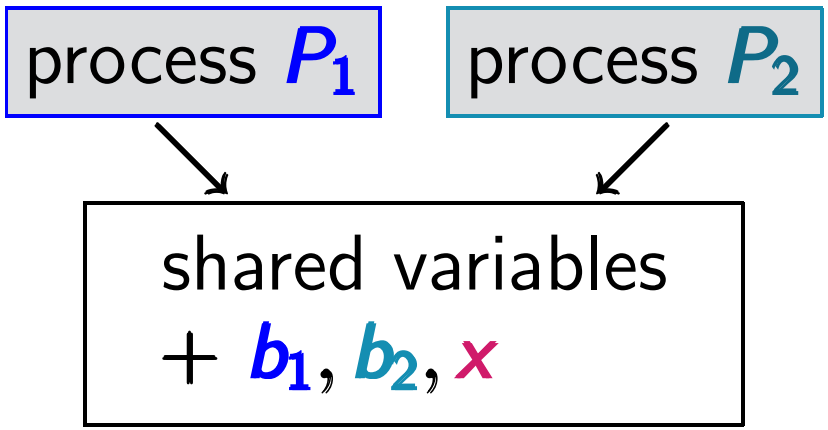


Peterson algorithm for mutual exclusion



b_1, b_2 Boolean variables, $x \in \{1, 2\}$

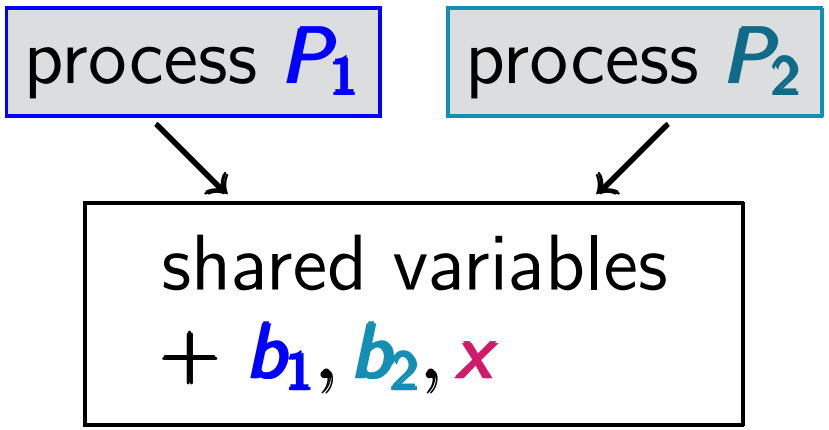
Peterson algorithm for mutual exclusion

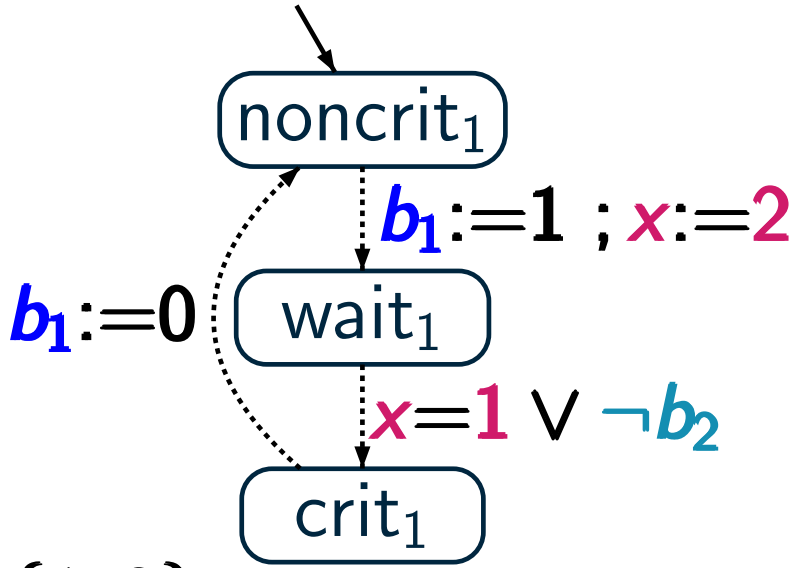


b_1, b_2 Boolean variables, $x \in \{1, 2\}$

```
LOOP FOREVER                                   (* protocol for P1 *)
    noncritical actions;
    b1:=1 ; x:=2;
    AWAIT x=1 ∨ ¬b2 DO critical section OD
    b1:=0
END LOOP
```

Peterson algorithm for mutual exclusion

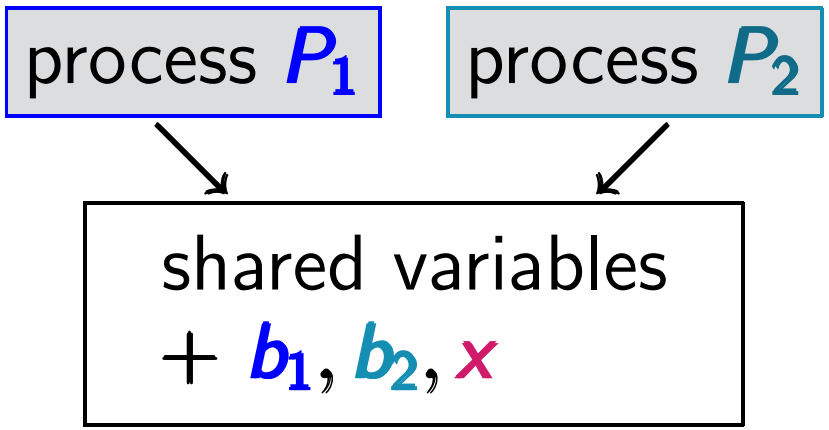


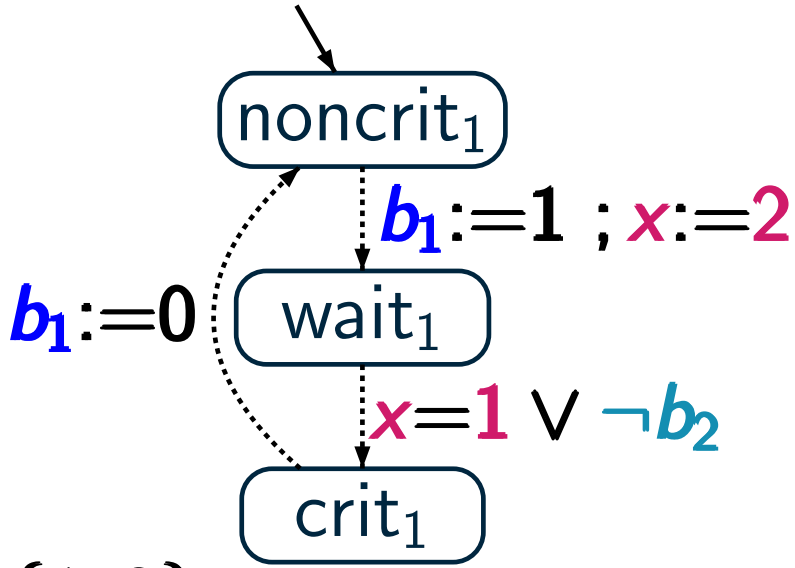


b_1, b_2 Boolean variables, $x \in \{1, 2\}$

```
LOOP FOREVER                                   (* protocol for P1 *)
    noncritical actions;
    b1:=1 ; x:=2;
    AWAIT x=1 ∨ ¬b2 DO critical section OD
    b1:=0
END LOOP
```

Peterson algorithm for mutual exclusion

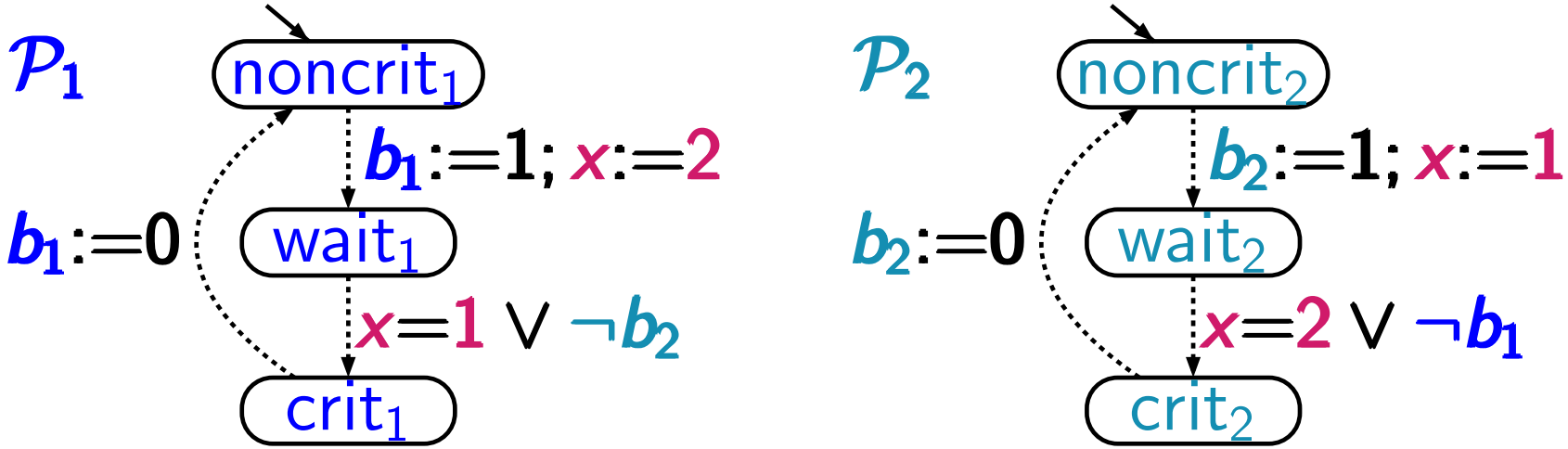




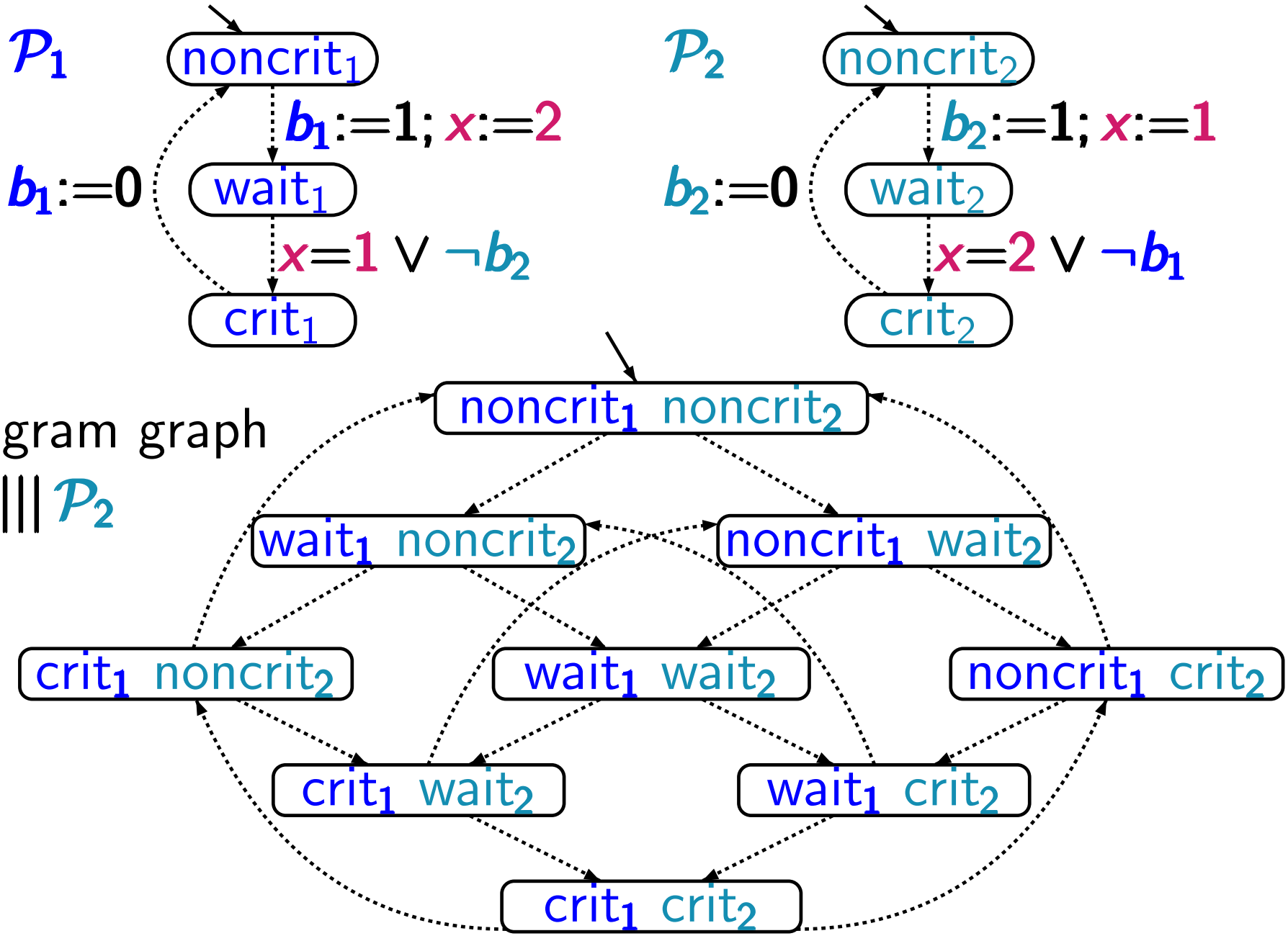
b_1, b_2 Boolean variables, $x \in \{1, 2\}$

```
LOOP FOREVER                          (* protocol for P1 *)
    noncritical actions;
    atomic{b1:=1 ; x:=2};
    AWAIT x=1 ∨ ¬b2 DO critical section OD
    b1:=0
END LOOP
```

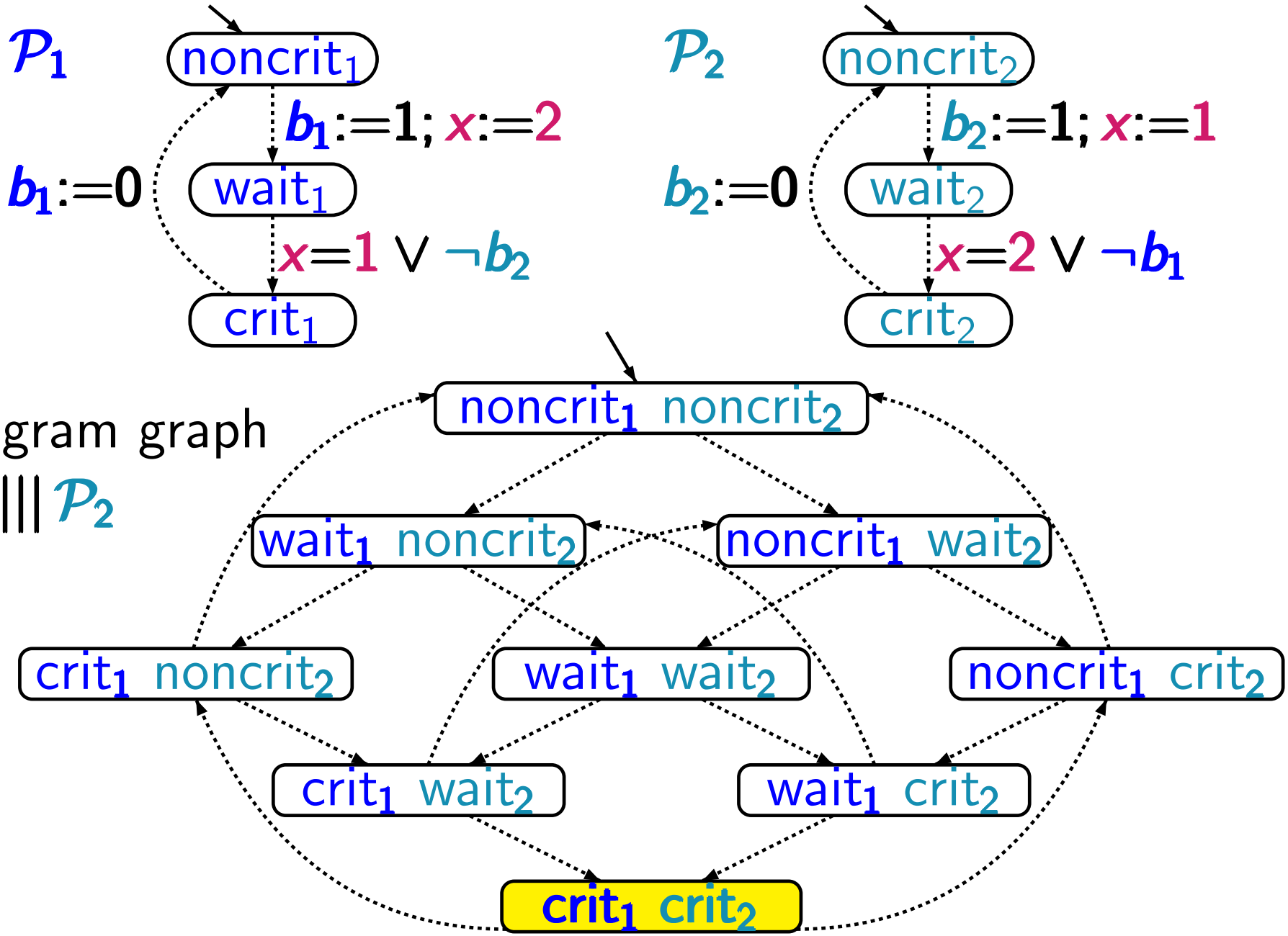
Program graphs for Peterson algorithm



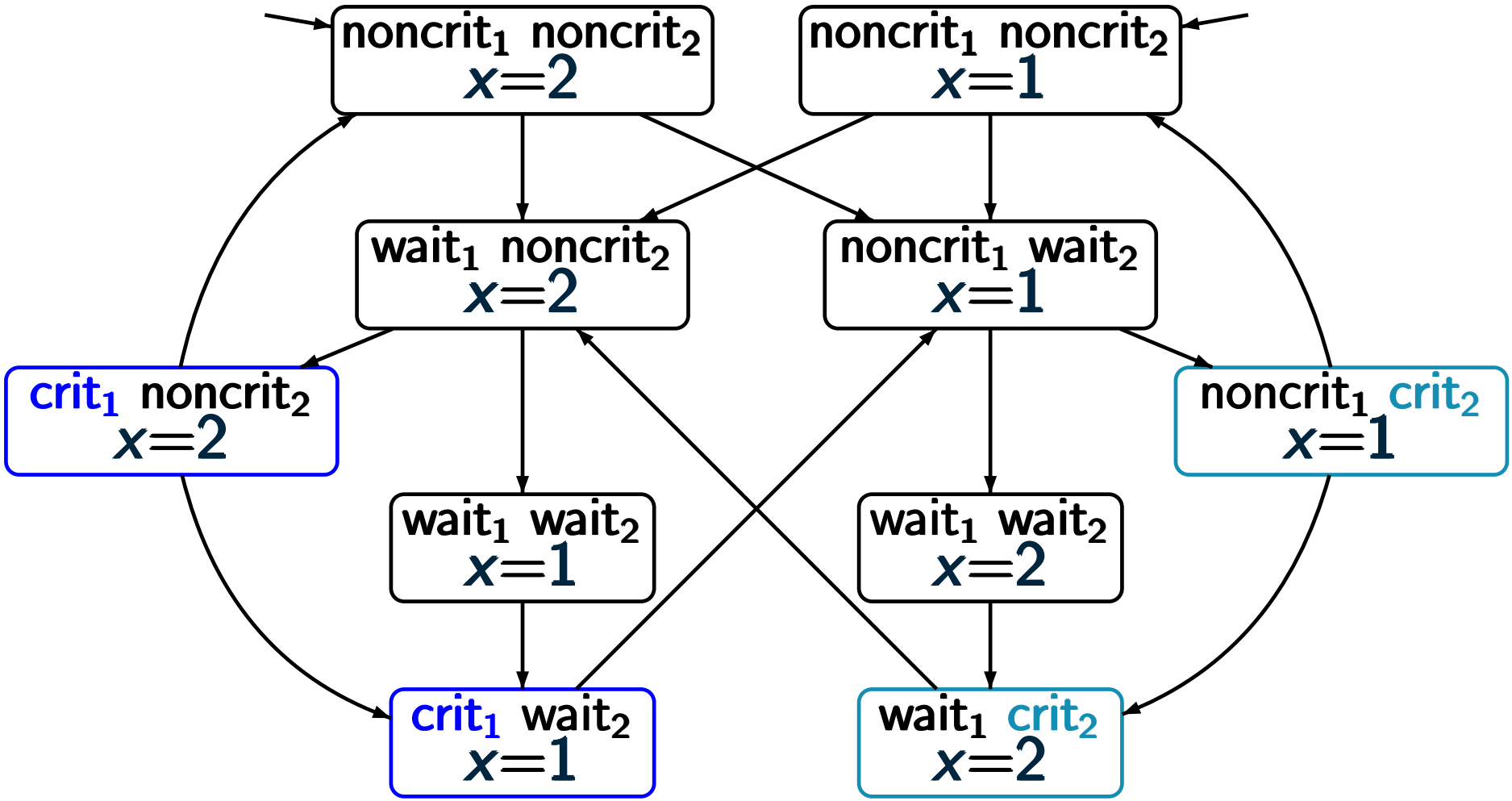
Program graphs for Peterson algorithm



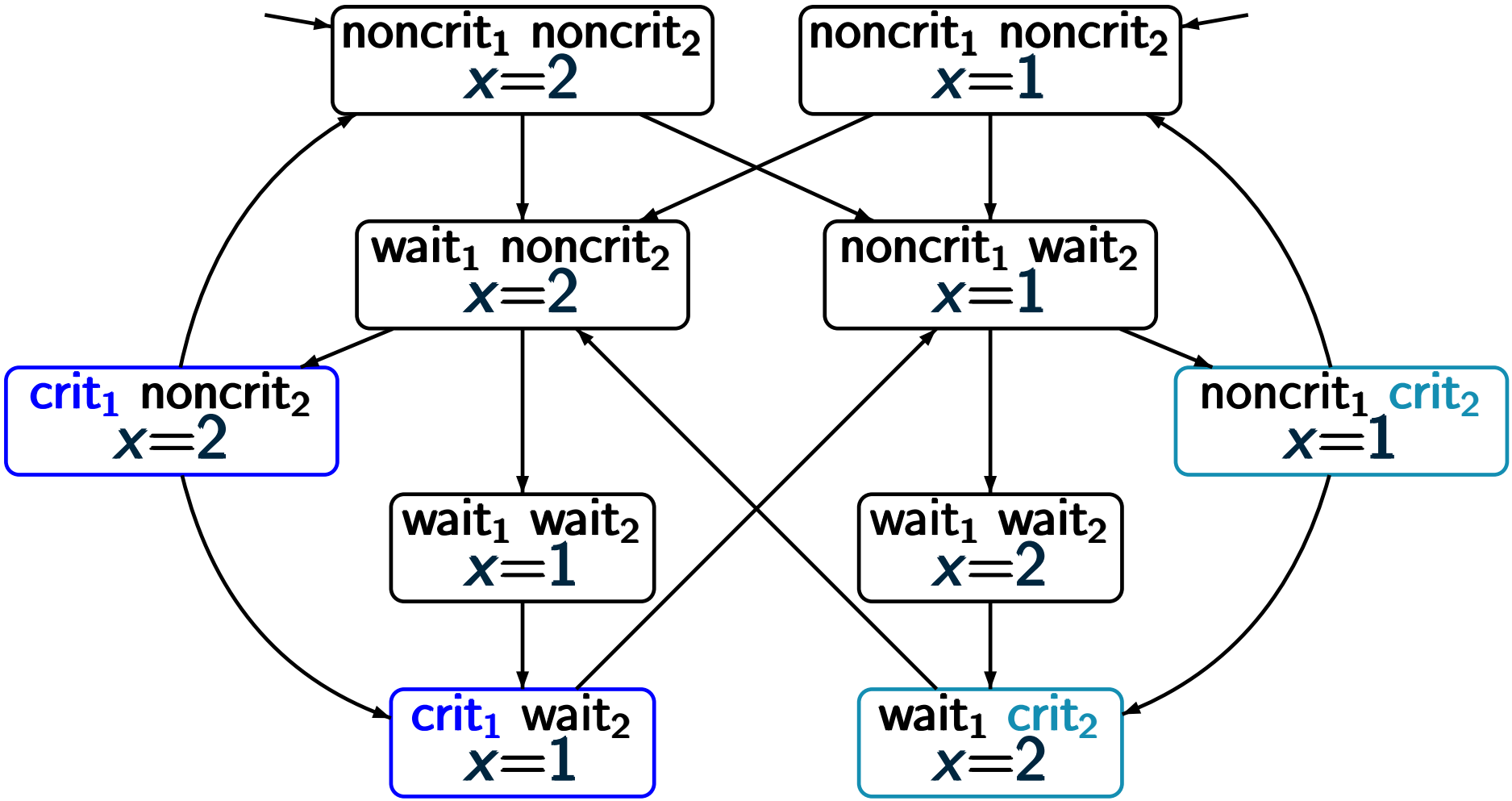
Program graphs for Peterson algorithm



TS for the Peterson algorithm



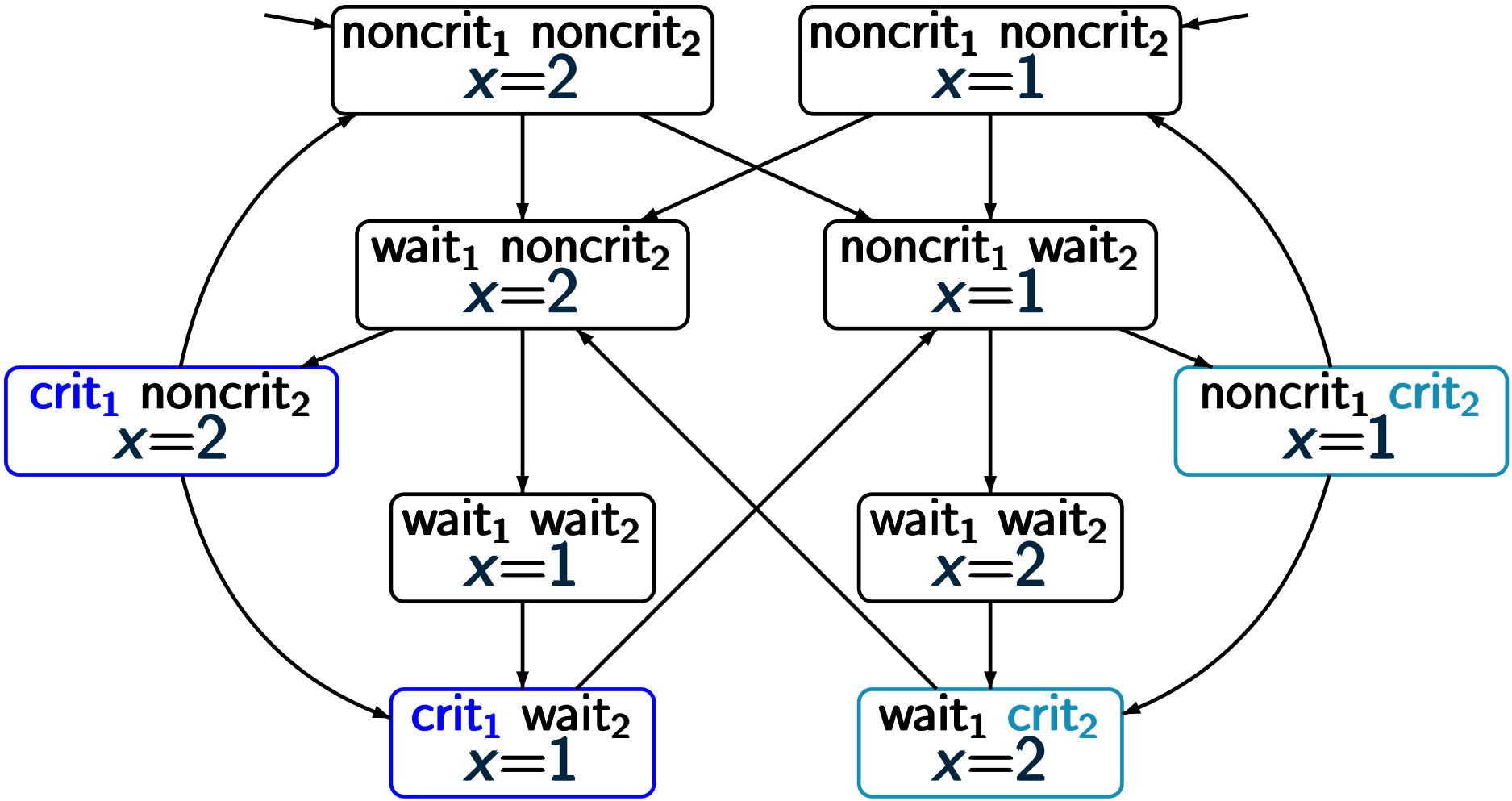
TS for the Peterson algorithm



value of b_1 is given by $\mathbf{wait_1} \vee \mathbf{crit_1}$

value of b_2 is given by $\mathbf{wait_2} \vee \mathbf{crit_2}$

TS for the Peterson algorithm

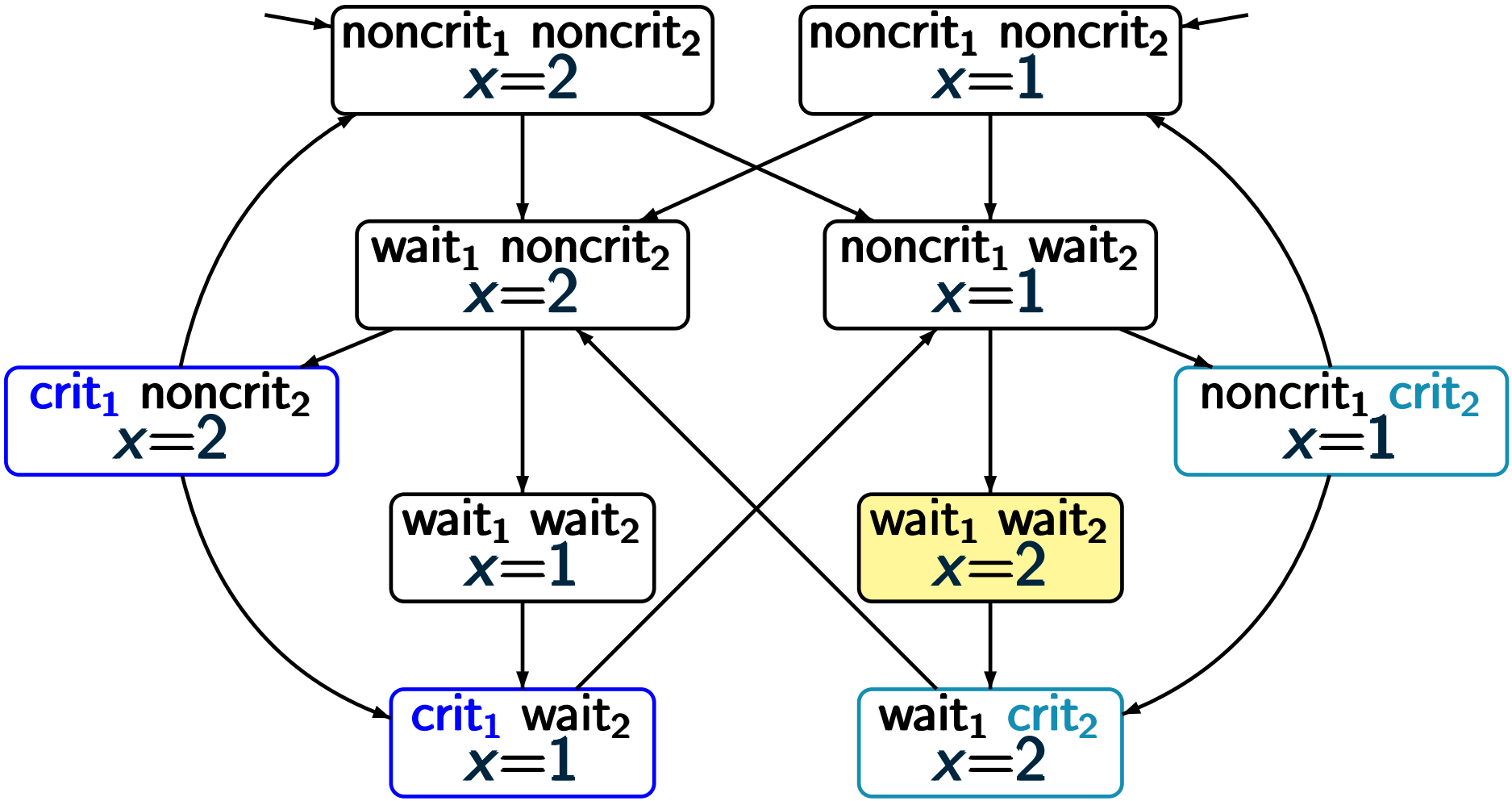


value of b_1 is given by $\mathbf{wait_1} \vee \mathbf{crit_1}$

value of b_2 is given by $\mathbf{wait_2} \vee \mathbf{crit_2}$

+ unreachable states

TS for the Peterson algorithm

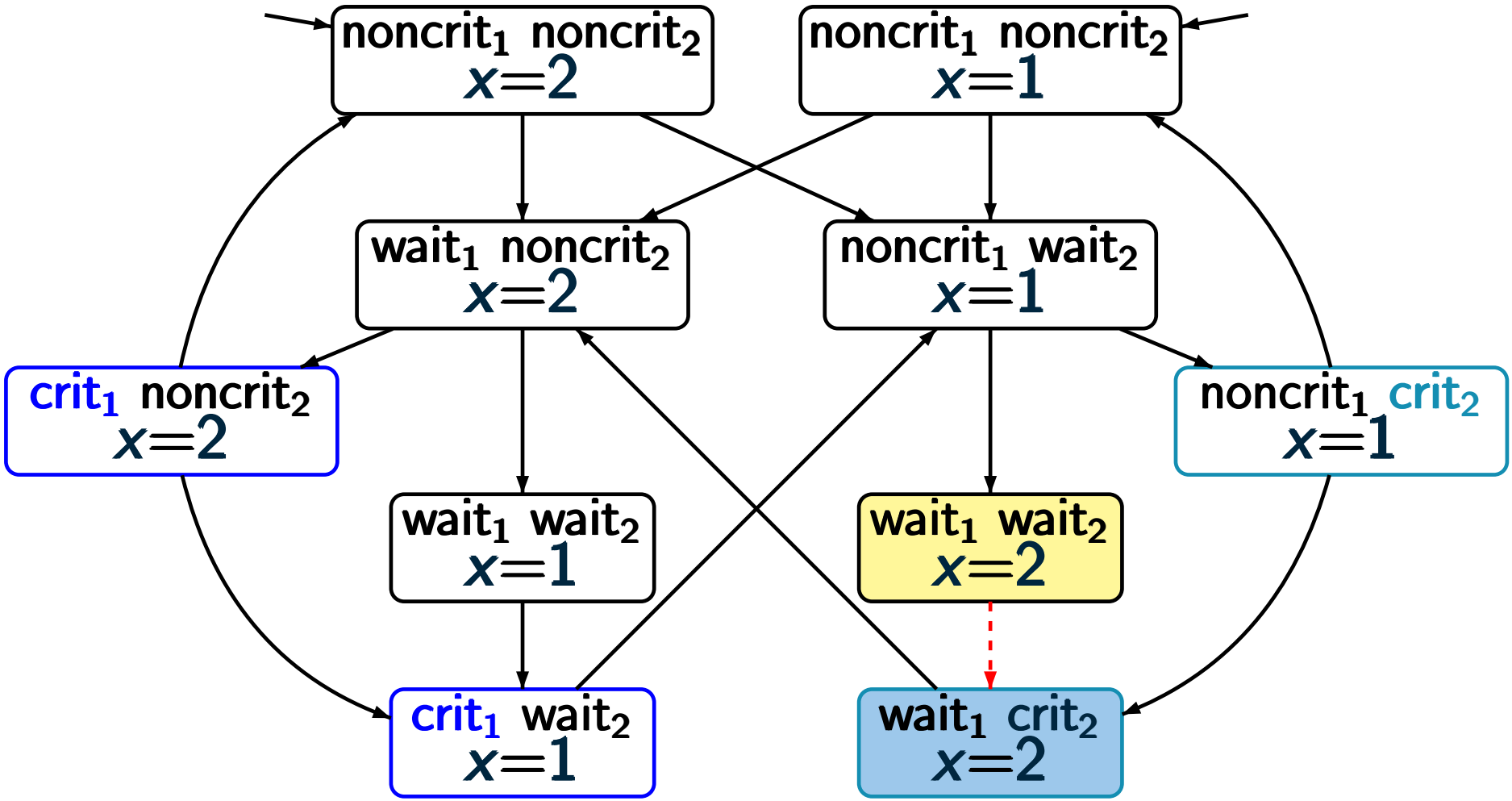


value of b_1 is given by $\mathbf{wait_1} \vee \mathbf{crit_1}$

value of b_2 is given by $\mathbf{wait_2} \vee \mathbf{crit_2}$

+ unreachable states

TS for the Peterson algorithm

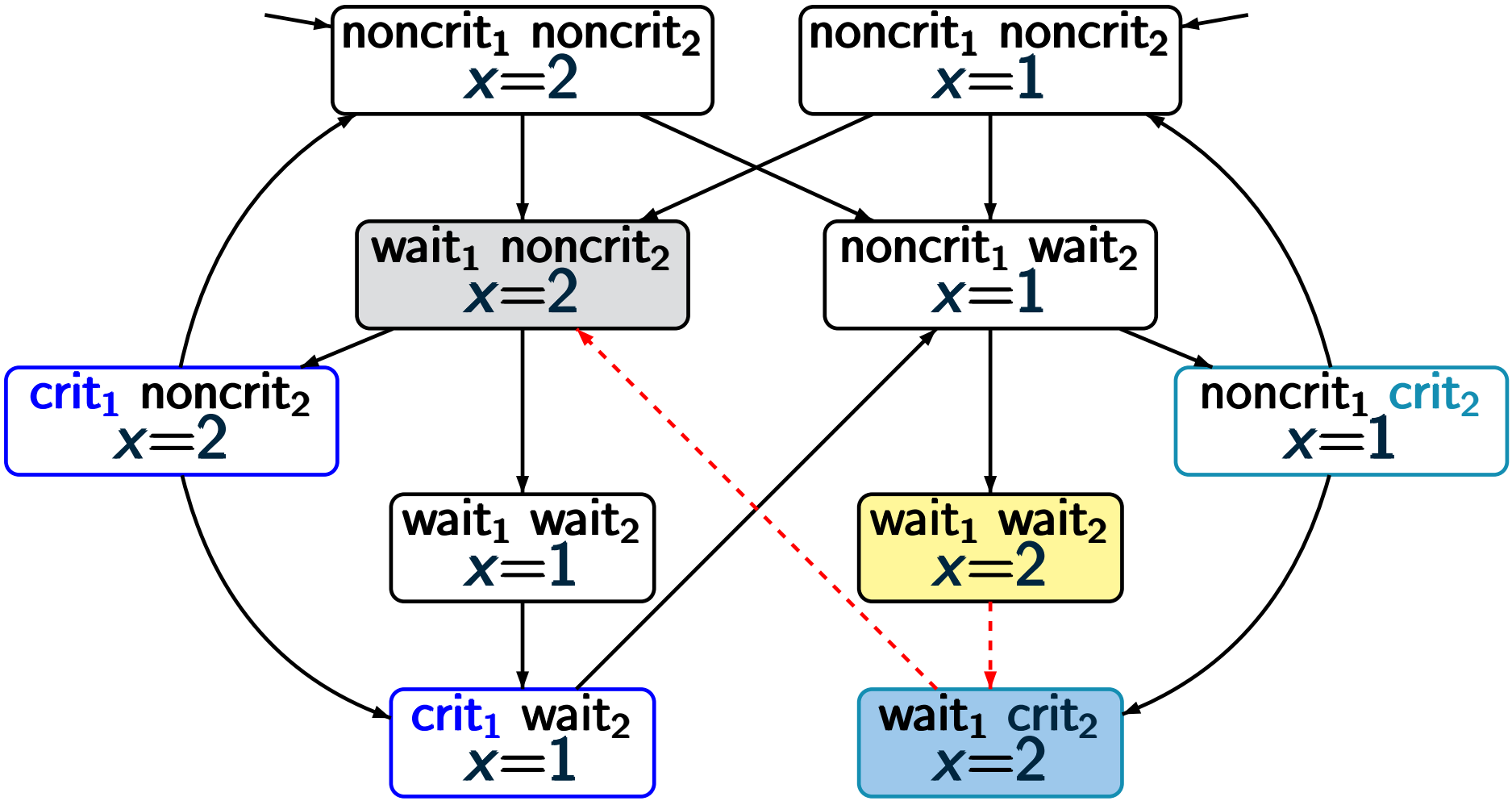


value of b_1 is given by $\mathbf{wait_1} \vee \mathbf{crit_1}$

value of b_2 is given by $\mathbf{wait_2} \vee \mathbf{crit_2}$

+ unreachable states

TS for the Peterson algorithm

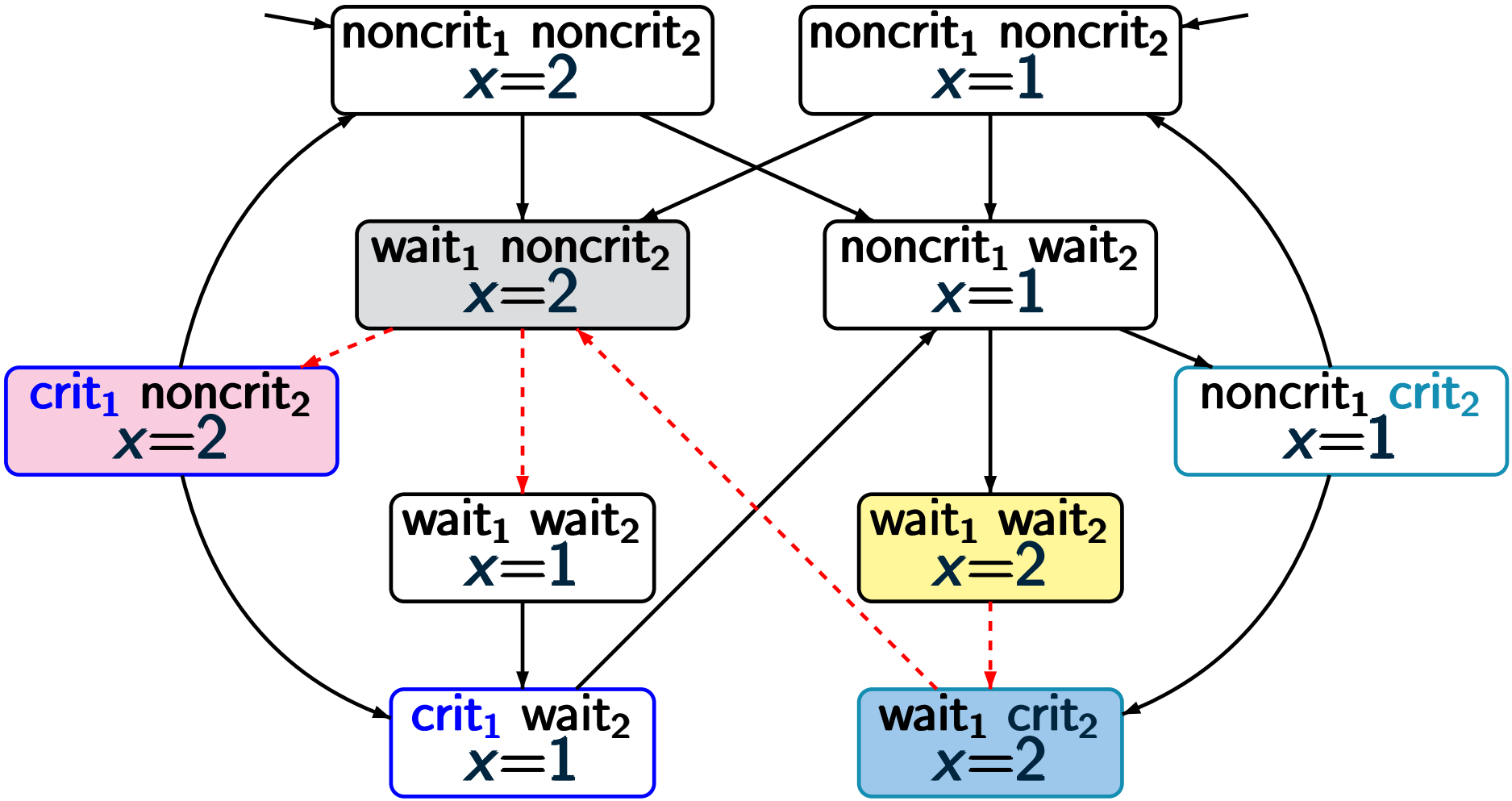


value of b_1 is given by $\mathbf{wait_1} \vee \mathbf{crit_1}$

value of b_2 is given by $\mathbf{wait_2} \vee \mathbf{crit_2}$

\+ unreachable states

TS for the Peterson algorithm

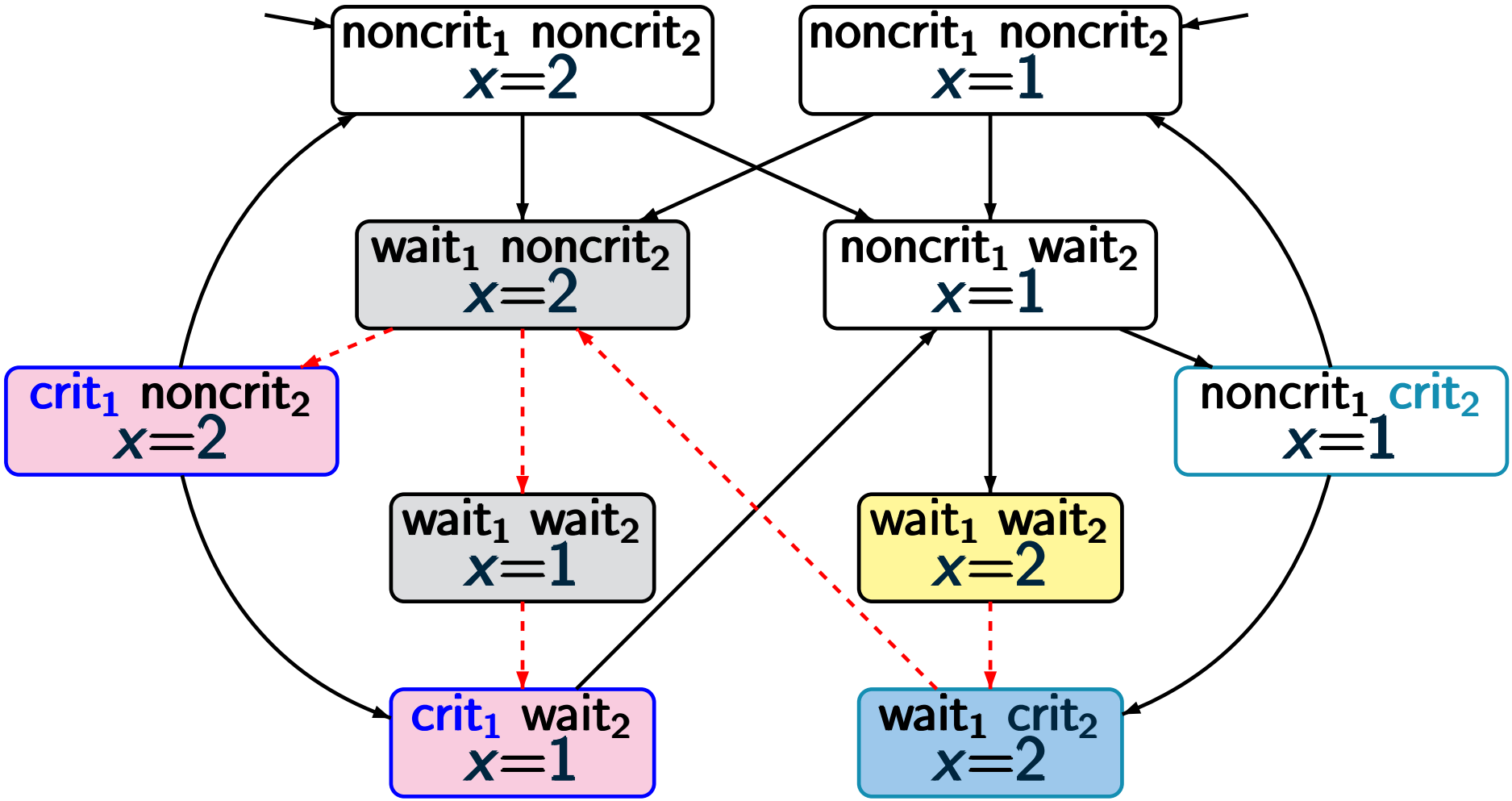


value of b_1 is given by $\mathbf{wait_1} \vee \mathbf{crit_1}$

value of b_2 is given by $\mathbf{wait_2} \vee \mathbf{crit_2}$

+ unreachable states

TS for the Peterson algorithm

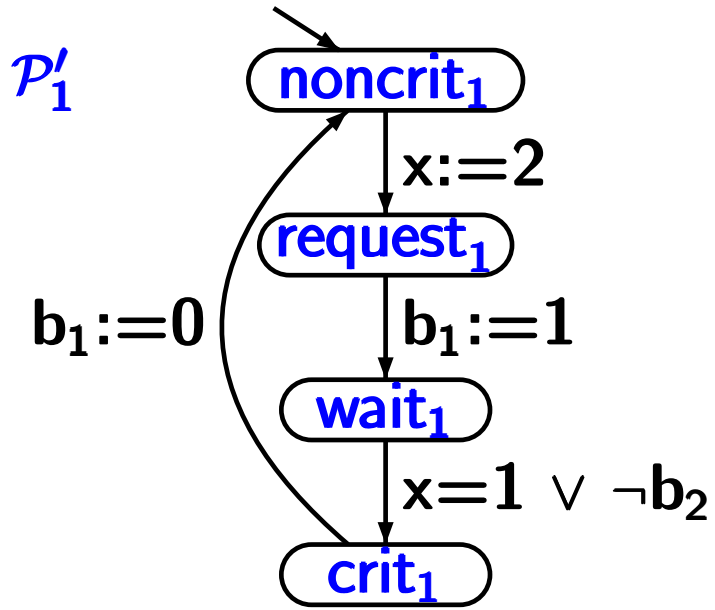


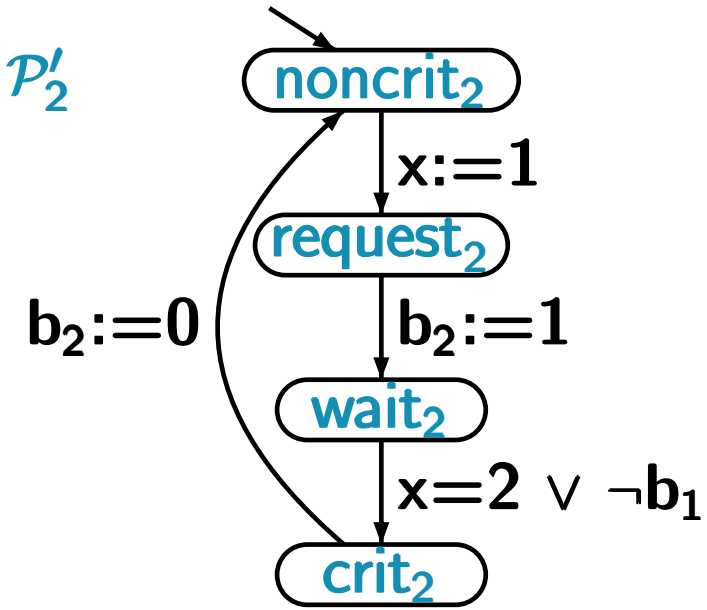
value of b_1 is given by $\mathbf{wait_1} \vee \mathbf{crit_1}$

value of b_2 is given by $\mathbf{wait_2} \vee \mathbf{crit_2}$

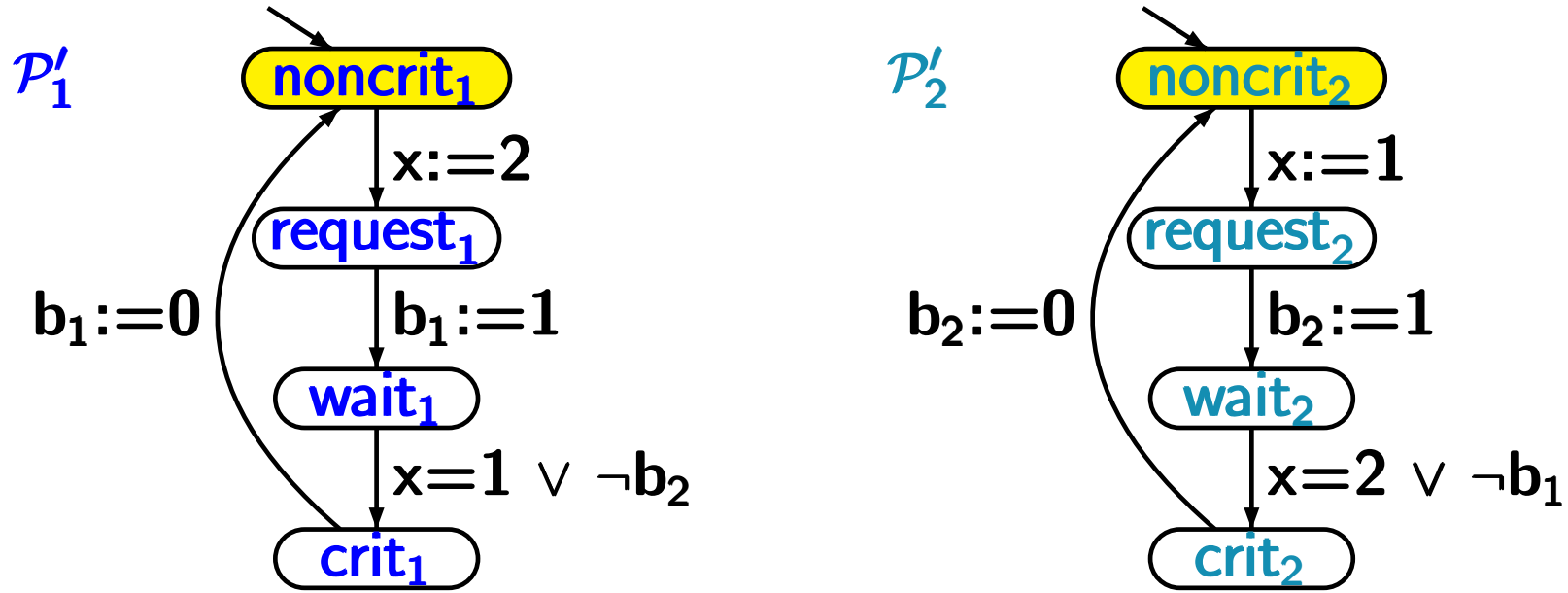
+ unreachable states

Variant of Peterson algorithm





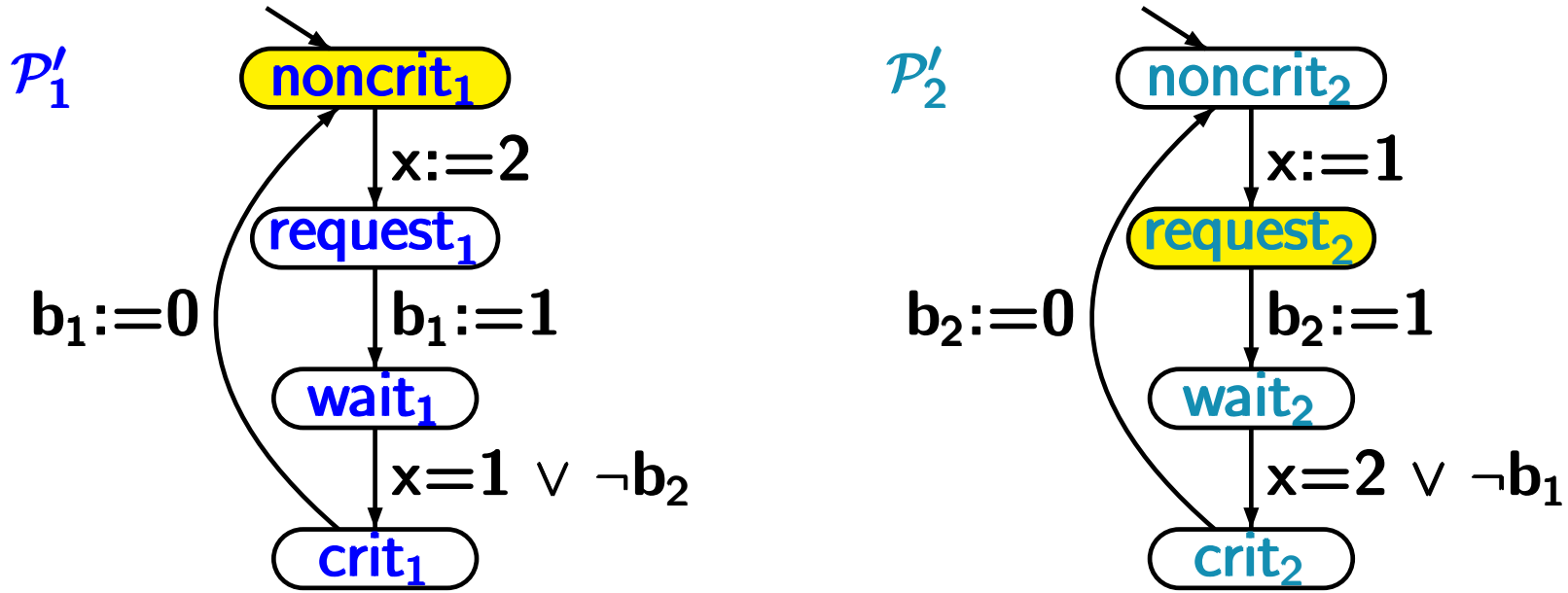
Variant of Peterson algorithm



possible executions

noncrit$_1$ noncrit$_2$ x=1 ¬b_1 ¬b_2

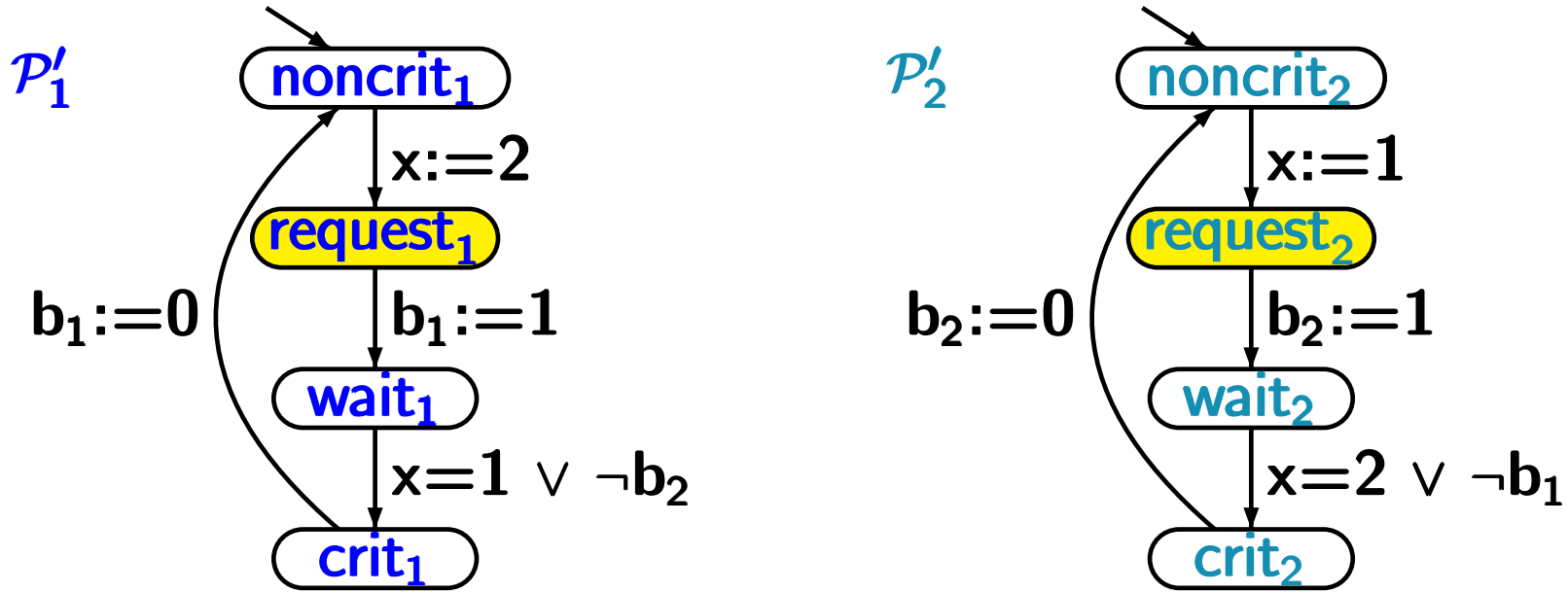
Variant of Peterson algorithm



possible executions

noncrit$_1$	noncrit$_2$	x=1	¬b$_1$	¬b$_2$
noncrit$_1$	request$_2$	x=1	¬ b$_1$	¬b$_2$

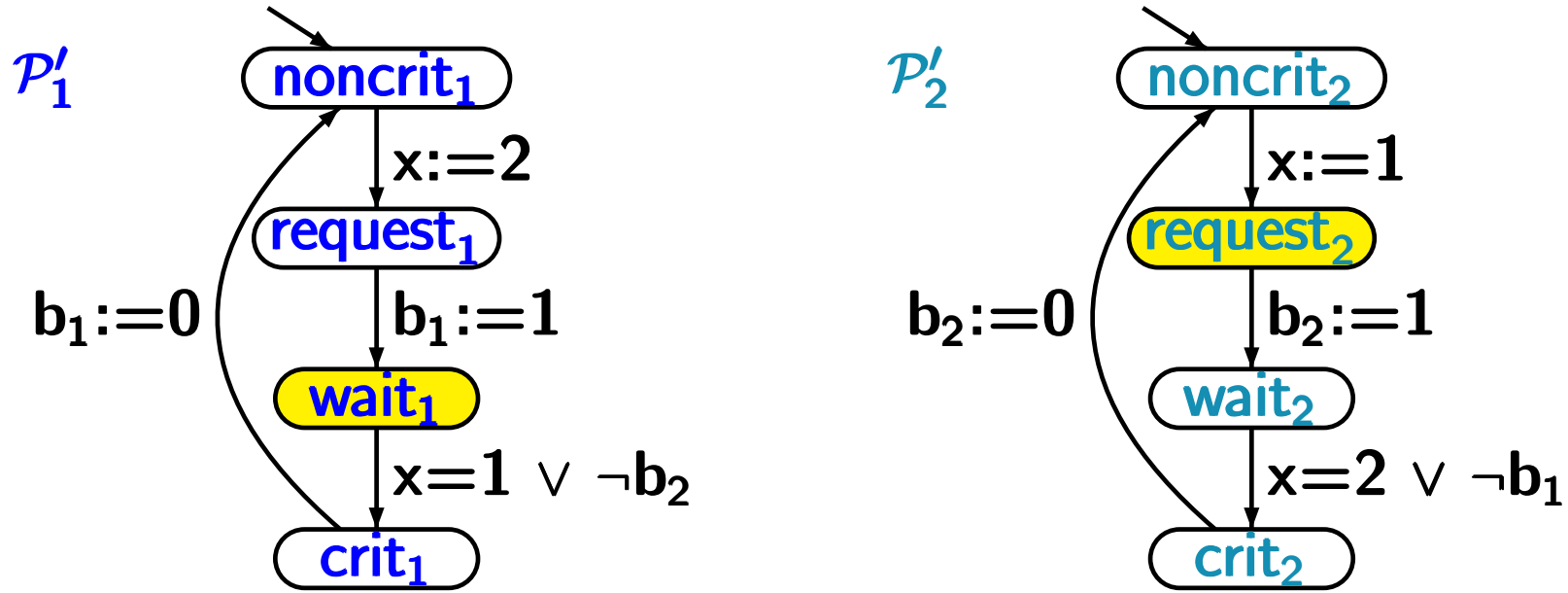
Variant of Peterson algorithm



possible executions

noncrit$_1$	noncrit$_2$	x=1	¬b$_1$	¬b$_2$
noncrit$_1$	request$_2$	x=1	¬ b$_1$	¬b$_2$
request$_1$	request$_2$	x=2	¬b$_1$	¬b$_2$

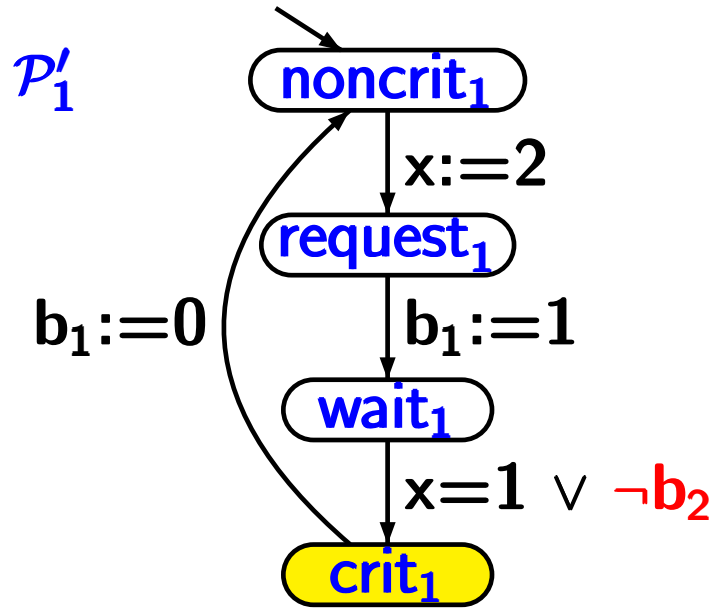
Variant of Peterson algorithm

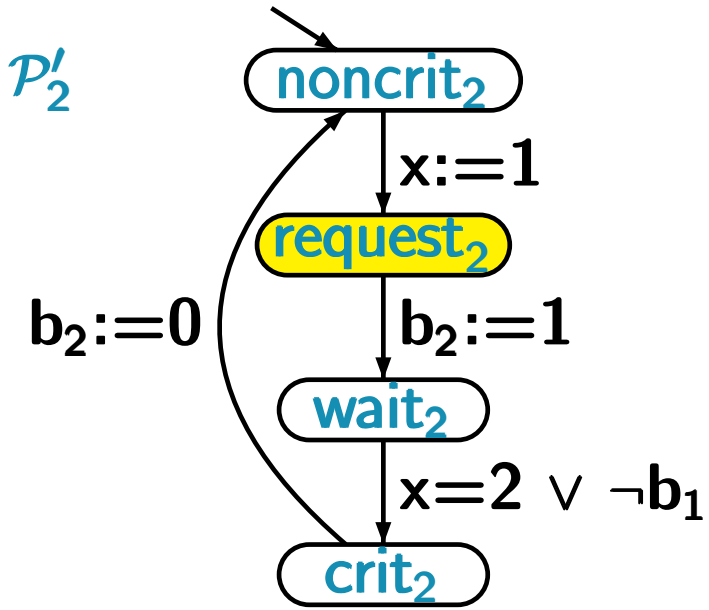


possible executions

noncrit$_1$	noncrit$_2$	x=1	¬b$_1$	¬b$_2$
noncrit$_1$	request$_2$	x=1	¬ b$_1$	¬b$_2$
request$_1$	request$_2$	x=2	¬b$_1$	¬b$_2$
wait$_1$	request$_2$	x=2	b$_1$	¬b$_2$

Variant of Peterson algorithm

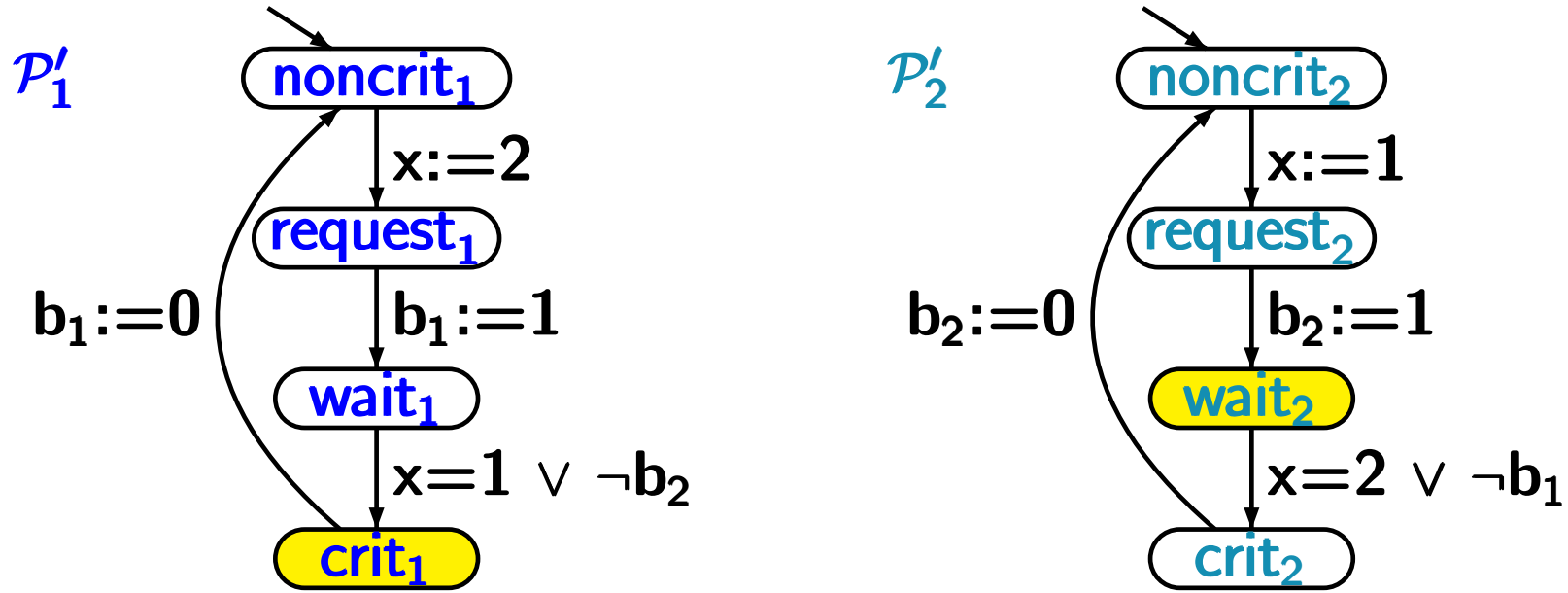




possible executions

noncrit$_1$	noncrit$_2$	x=1	¬b$_1$	¬b$_2$
noncrit$_1$	request$_2$	x=1	¬ b$_1$	¬b$_2$
request$_1$	request$_2$	x=2	¬b$_1$	¬b$_2$
wait$_1$	request$_2$	x=2	b$_1$	¬b$_2$
crit$_1$	request$_2$	x=2	b$_1$	¬b$_2$

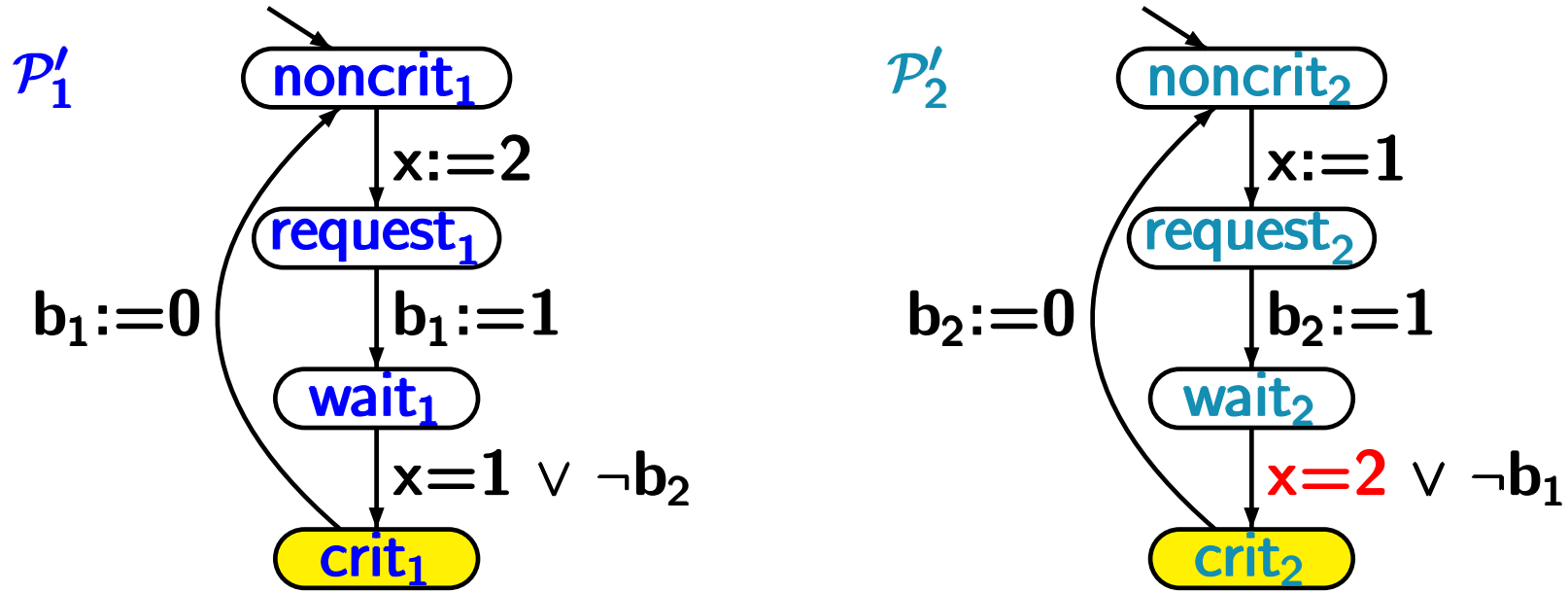
Variant of Peterson algorithm



possible executions

noncrit$_1$	noncrit$_2$	x=1	¬b$_1$	¬b$_2$
noncrit$_1$	request$_2$	x=1	¬ b$_1$	¬b$_2$
request$_1$	request$_2$	x=2	¬b$_1$	¬b$_2$
wait$_1$	request$_2$	x=2	b$_1$	¬b$_2$
crit$_1$	request$_2$	x=2	b$_1$	¬b$_2$
crit$_1$	wait$_2$	x=2	b$_1$	b$_2$

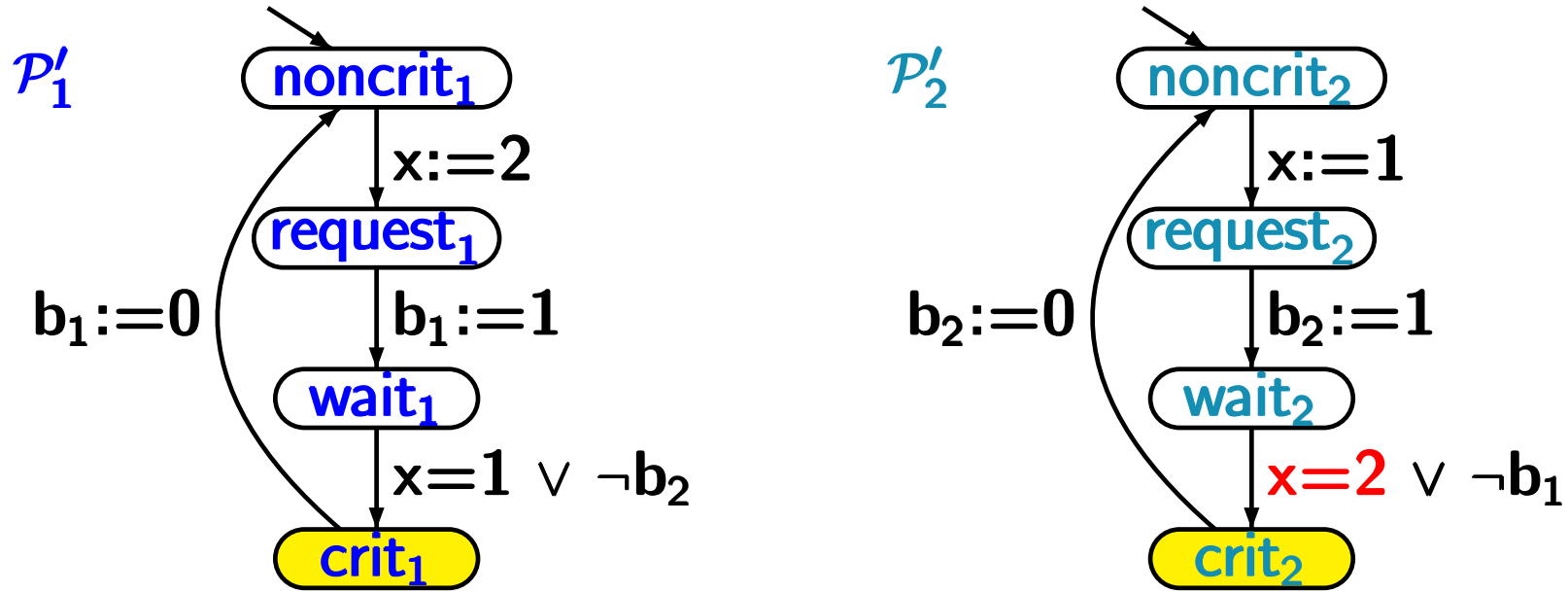
Variant of Peterson algorithm



possible executions

noncrit$_1$	noncrit$_2$	x=1	¬b$_1$	¬b$_2$
noncrit$_1$	request$_2$	x=1	¬ b$_1$	¬b$_2$
request$_1$	request$_2$	x=2	¬b$_1$	¬b$_2$
wait$_1$	request$_2$	x=2	b$_1$	¬b$_2$
crit$_1$	request$_2$	x=2	b$_1$	¬b$_2$
crit$_1$	wait$_2$	x=2	b$_1$	b$_2$
crit$_1$	crit$_2$	x=2	b$_1$	b$_2$

Variant of Peterson algorithm incorrect!



possible executions

noncrit$_1$	noncrit$_2$	x=1	¬b$_1$	¬b$_2$
noncrit$_1$	request$_2$	x=1	¬ b$_1$	¬b$_2$
request$_1$	request$_2$	x=2	¬b$_1$	¬b$_2$
wait$_1$	request$_2$	x=2	b$_1$	¬b$_2$
crit$_1$	request$_2$	x=2	b$_1$	¬b$_2$
crit$_1$	wait$_2$	x=2	b$_1$	b$_2$
crit$_1$	crit$_2$	x=2	b$_1$	b$_2$

How many states ...?

Given n processes by program graphs $\mathcal{P}_1, \ldots, \mathcal{P}_n$

How many states ...?

Given n processes by program graphs $\mathcal{P}_1, \ldots, \mathcal{P}_n$

- with 2 locations each
- over the set of variables $Var = \{x_1, \ldots, x_m\}$ with $Dom(x_i) = \{0, 1\}$

How many states has the transition system $\mathcal{T}_{\mathcal{P}_1 \,|||\, \ldots \,|||\, \mathcal{P}_n}$?

How many states ...?

Given n processes by program graphs $\mathcal{P}_1, \ldots, \mathcal{P}_n$

- with 2 locations each
- over the set of variables $Var = \{x_1, \ldots, x_m\}$ with $Dom(x_i) = \{0, 1\}$

How many states has the transition system $\mathcal{T}_{\mathcal{P}_1 \,|||\, \ldots \,|||\, \mathcal{P}_n}$?

answer: $2^n \cdot 2^m$

State explosion problem

Given n processes by program graphs $\mathcal{P}_1, \ldots, \mathcal{P}_n$

- with 2 locations each
- over the set of variables $Var = \{x_1, \ldots, x_m\}$ with $Dom(x_i) = \{0, 1\}$

How many states has the transition system $\mathcal{T}_{\mathcal{P}_1 \,|||\, \ldots \,|||\, \mathcal{P}_n}$?

answer: $2^n \cdot 2^m$

state explosion: size of transition systems grows

- exponentially in the number of parallel processes
- exponentially in the number of variables

Operators for parallelism and communication

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS (no communication, no dependencies)

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS (no communication, no dependencies)
- communication via shared variables
  - description of subsystems by program graphs
  - interleaving $|||$ for program graphs
  - TS is obtained by "unfolding"

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS (no communication, no dependencies)
- communication via shared variables
  - description of subsystems by program graphs
  - interleaving $|||$ for program graphs
  - TS is obtained by "unfolding"
- synchronous message passing

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS (no communication, no dependencies)
- communication via shared variables
  - description of subsystems by program graphs
  - interleaving $|||$ for program graphs
  - TS is obtained by "unfolding"
- synchronous message passing
  - operator $\|_{Syn}$ for TS
  - interleaving for independent actions
  - synchronization over actions in Syn

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS (no communication, no dependencies)
- communication via shared variables
  - description of subsystems by program graphs
  - interleaving $|||$ for program graphs
  - TS is obtained by "unfolding"
- synchronous message passing $\longleftarrow$ data abstract
  - operator $\|_{Syn}$ for TS
  - interleaving for independent actions
  - synchronization over actions in Syn

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS
  (no communication, no dependencies)
- communication via shared variables
  - description of subsystems by program graphs
  - interleaving $|||$ for program graphs
  - TS is obtained by "unfolding"
- synchronous message passing $\longleftarrow$ data abstract
  - operator $\|_{Syn}$ for TS
  - interleaving for independent actions
  - synchronization over actions in Syn
- channel systems
  communication via shared variables + via channels
- synchronous product

Synchronous message passing

Synchronous message passing

$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, \ldots)$, $\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$ TS

$Syn \subseteq Act_1 \cap Act_2$ set of synchronization actions

Synchronous message passing

$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, \ldots)$, $\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$ TS

$Syn \subseteq Act_1 \cap Act_2$ set of synchronization actions

composite transition system:

$$\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \ldots)$$

for modeling the concurrent execution of $\mathcal{T}_1$ and $\mathcal{T}_2$ with synchronization over all actions in Syn

Synchronous message passing

$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, \ldots)$, $\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$ TS

$Syn \subseteq Act_1 \cap Act_2$ set of synchronization actions

composite transition system:

$$\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \ldots)$$

interleaving for all actions $\alpha \in Act_i \setminus Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

Synchronous message passing

$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, \ldots)$, $\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$ TS

$Syn \subseteq Act_1 \cap Act_2$ set of synchronization actions

composite transition system:

$$\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \ldots)$$

interleaving for all actions $\alpha \in Act_i \setminus Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

handshaking (rendezvous) for all $\alpha \in Syn$:

Synchronous message passing

$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, \ldots)$, $\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$ TS

$Syn \subseteq Act_1 \cap Act_2$ set of synchronization actions

composite transition system:

$$\mathcal{T}_1 \,\|_{Syn}\, \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \ldots)$$

interleaving for all actions $\alpha \in Act_i \setminus Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2\rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1, s_2'\rangle}$$

handshaking (rendezvous) for all $\alpha \in Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \quad \wedge \quad s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2'\rangle}$$

Mutual exclusion

by synchronous message passing

Mutual exclusion

by synchronous message passing using an arbiter

Mutual exclusion with an arbiter

protocol for process P_i

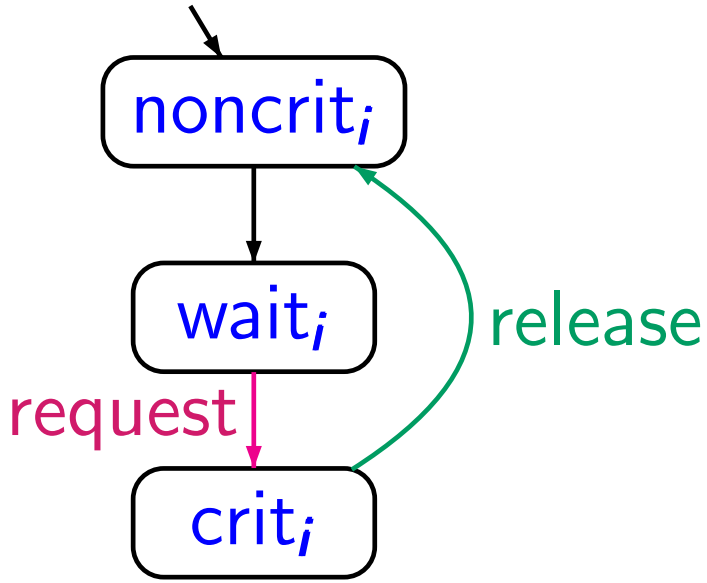
```
LOOP FOREVER DO
     noncritical actions
     request
     critical section
     release
     noncritical actions
OD
```

Mutual exclusion with an arbiter

protocol for process P_i

```
LOOP FOREVER DO
      noncritical actions
      request
      critical section
      release
      noncritical actions
OD
```

transition system $\mathcal{T}_i$

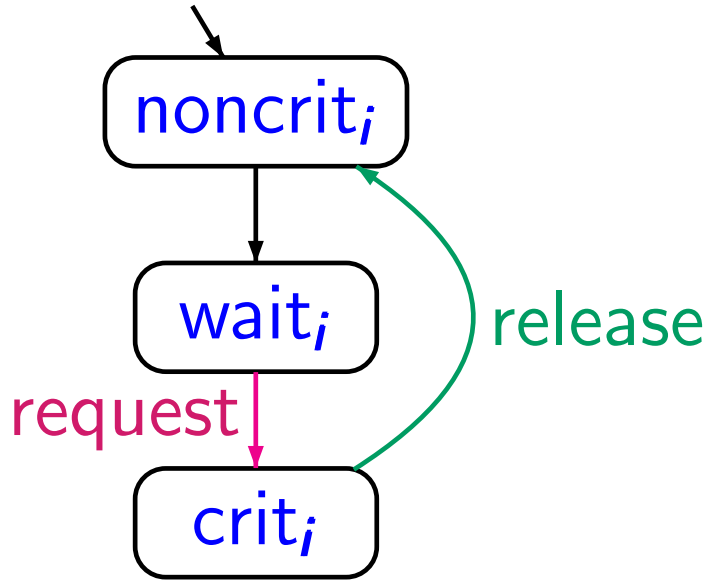


Mutual exclusion with an arbiter

protocol for process P_i

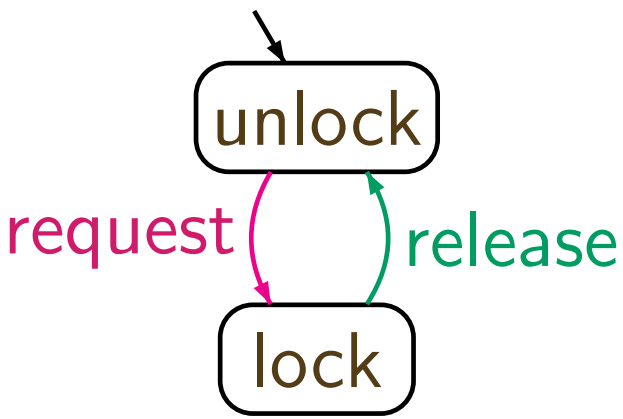
```
LOOP FOREVER DO
      noncritical actions
      request
      critical section
      release
      noncritical actions
OD
```

transition system $\mathcal{T}_i$



Arbiter:

selects nondeterministically
a synchronization partner
$\mathcal{T}_1$ or $\mathcal{T}_2$



Mutual exclusion with an arbiter

$$(\mathcal{T}_1 \;|||\; \mathcal{T}_2) \;\|_{Syn}\; \mathit{Arbiter} \text{ where } \mathit{Syn} = \{\text{request}, \text{release}\}$$

Mutual exclusion with an arbiter

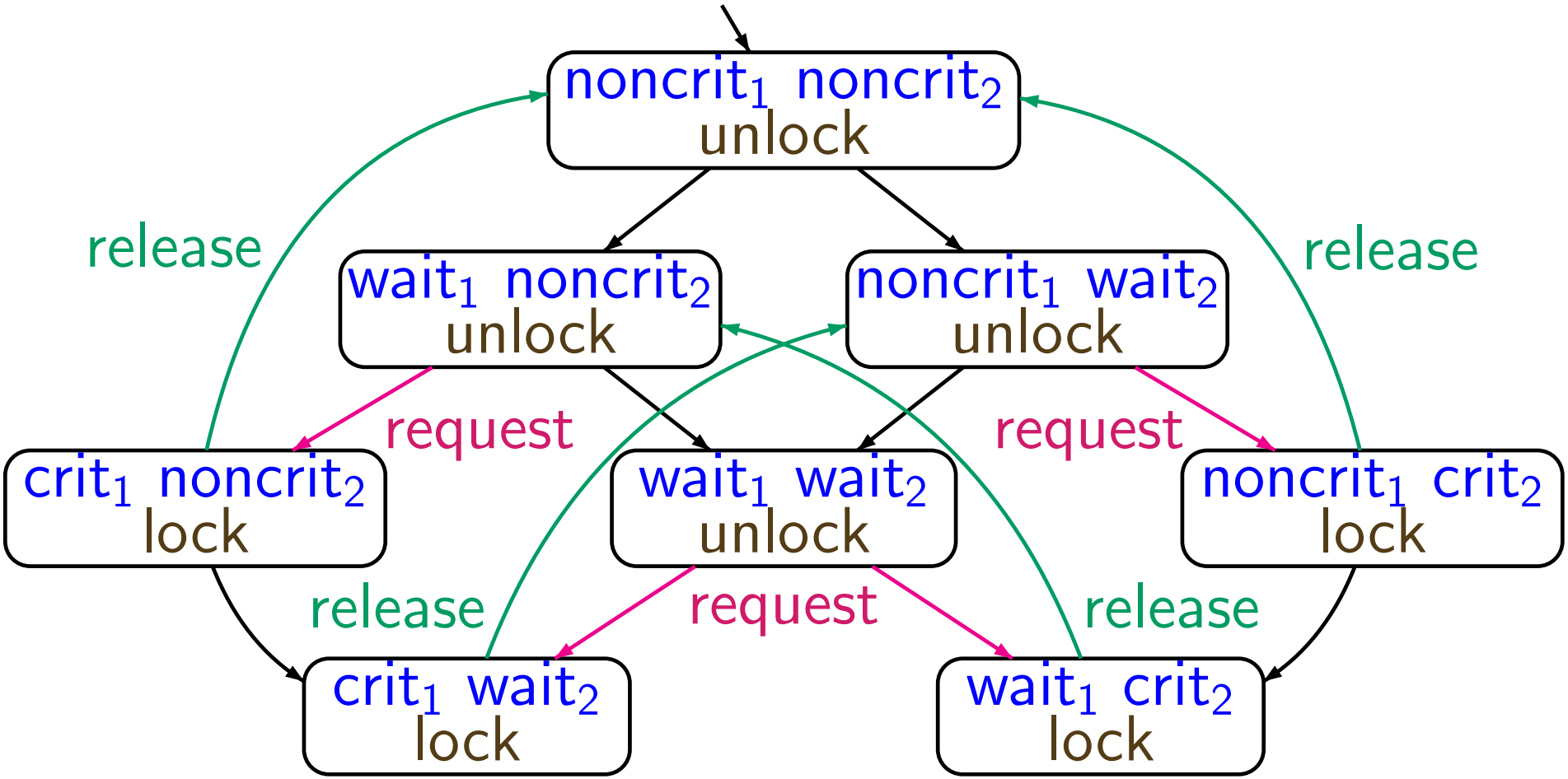
$$(\mathcal{T}_1 \;|||\; \mathcal{T}_2) \;\|_{Syn}\; \mathit{Arbiter} \text{ where } \mathit{Syn} = \{\text{request}, \text{release}\}$$

"pure" interleaving for TS

handshaking for actions request and release

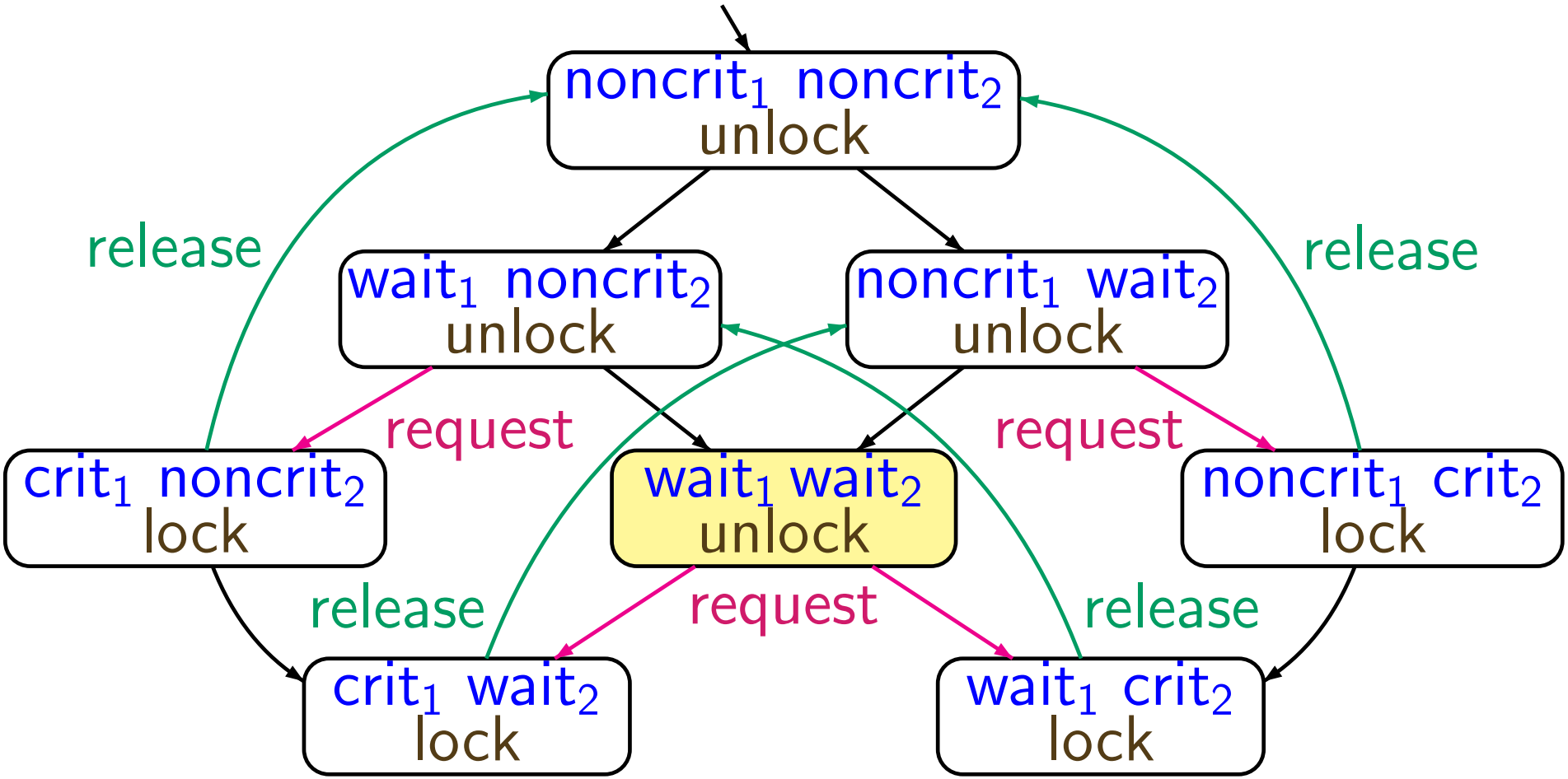
Mutual exclusion with an arbiter

$(\mathcal{T}_1 \,|||\, \mathcal{T}_2) \,\|_{Syn}\, \mathit{Arbiter}$ where $Syn = \{\text{request}, \text{release}\}$



Mutual exclusion with an arbiter

$(\mathcal{T}_1 \,|||\, \mathcal{T}_2) \,\|_{Syn}\, \mathit{Arbiter}$ where $Syn = \{\text{request}, \text{release}\}$



nondeterministic choice: who enters the critical section?

Synchronous message passing

synchronization operator $\|_{Syn}$ for

three or more processes

Synchronous message passing

$$\left.\begin{array}{lcl} \mathcal{T}_1 & = & (S_1, Act_1, \rightarrow_1, \ldots) \\ \mathcal{T}_2 & = & (S_2, Act_2, \rightarrow_2, \ldots) \\ \mathcal{T}_3 & = & (S_3, Act_3, \rightarrow_3, \ldots) \\ \mathcal{T}_4 & = & (S_4, Act_4, \rightarrow_4, \ldots) \\ \vdots & & \vdots \end{array}\right\} \text{transition systems}$$

Synchronous message passing

$$\left.\begin{array}{rcl} \mathcal{T}_1 & = & (S_1, Act_1, \rightarrow_1, \ldots) \\ \mathcal{T}_2 & = & (S_2, Act_2, \rightarrow_2, \ldots) \\ \mathcal{T}_3 & = & (S_3, Act_3, \rightarrow_3, \ldots) \\ \mathcal{T}_4 & = & (S_4, Act_4, \rightarrow_4, \ldots) \\ \vdots & & \vdots \end{array}\right\} \text{ transition systems}$$

for $Syn \subseteq Act_1 \cup Act_2 \cup Act_3 \cup Act_4 \cup \ldots$

$$\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2 \parallel_{Syn} \mathcal{T}_3 \parallel_{Syn} \mathcal{T}_4 \parallel_{Syn} \ldots \stackrel{\text{def}}{=}$$

$$\Big(\big((\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2) \parallel_{Syn} \mathcal{T}_3\big) \parallel_{Syn} \mathcal{T}_4\Big) \parallel_{Syn} \ldots$$

Synchronous message passing

$$\left.\begin{array}{rcl} \mathcal{T}_1 & = & (S_1, Act_1, \rightarrow_1, \ldots) \\ \mathcal{T}_2 & = & (S_2, Act_2, \rightarrow_2, \ldots) \\ \mathcal{T}_3 & = & (S_3, Act_3, \rightarrow_3, \ldots) \\ \mathcal{T}_4 & = & (S_4, Act_4, \rightarrow_4, \ldots) \\ \vdots & & \vdots \end{array}\right\} \text{ transition systems}$$

for $Syn \subseteq Act_1 \cup Act_2 \cup Act_3 \cup Act_4 \cup \ldots$

$$\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2 \parallel_{Syn} \mathcal{T}_3 \parallel_{Syn} \mathcal{T}_4 \parallel_{Syn} \ldots \stackrel{\text{def}}{=}$$

$$\Big(\big((\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2) \parallel_{Syn} \mathcal{T}_3\big) \parallel_{Syn} \mathcal{T}_4\Big) \parallel_{Syn} \ldots$$

or any other order of paranthesis

Synchronous message passing

$$\left.\begin{array}{rcl} \mathcal{T}_1 & = & (S_1, Act_1, \rightarrow_1, \ldots) \\ \mathcal{T}_2 & = & (S_2, Act_2, \rightarrow_2, \ldots) \\ \mathcal{T}_3 & = & (S_3, Act_3, \rightarrow_3, \ldots) \\ \mathcal{T}_4 & = & (S_4, Act_4, \rightarrow_4, \ldots) \\ \vdots & & \vdots \end{array}\right\} \text{ transition systems}$$

for $Syn \subseteq Act_1 \cup Act_2 \cup Act_3 \cup Act_4 \cup \ldots$

$$\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2 \parallel_{Syn} \mathcal{T}_3 \parallel_{Syn} \mathcal{T}_4 \parallel_{Syn} \ldots \stackrel{\text{def}}{=}$$

$$\Big(\big((\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2) \parallel_{Syn} \mathcal{T}_3\big) \parallel_{Syn} \mathcal{T}_4\Big) \parallel_{Syn} \ldots$$

if Syn be a subset of Act1, Act2, ..., then the synchronous communication is associative

where, e.g., $\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2 \stackrel{\text{def}}{=} \mathcal{T}_1 \parallel_H \mathcal{T}_2$

with $H = Syn \cap Act_1 \cap Act_2$

Parallel operator $\|$

$$\begin{array}{lcl} \mathcal{T}_1 & = & (S_1, Act_1, \rightarrow_1, \ldots) \\ \mathcal{T}_2 & = & (S_2, Act_2, \rightarrow_2, \ldots) \\ \mathcal{T}_3 & = & (S_3, Act_3, \rightarrow_3, \ldots) \\ \mathcal{T}_4 & = & (S_4, Act_4, \rightarrow_4, \ldots) \\ \vdots & & \vdots \end{array}$$

transition systems s.t.

$Act_i \cap Act_j \cap Act_k = \emptyset$

if i, j, k are pairwise distinct

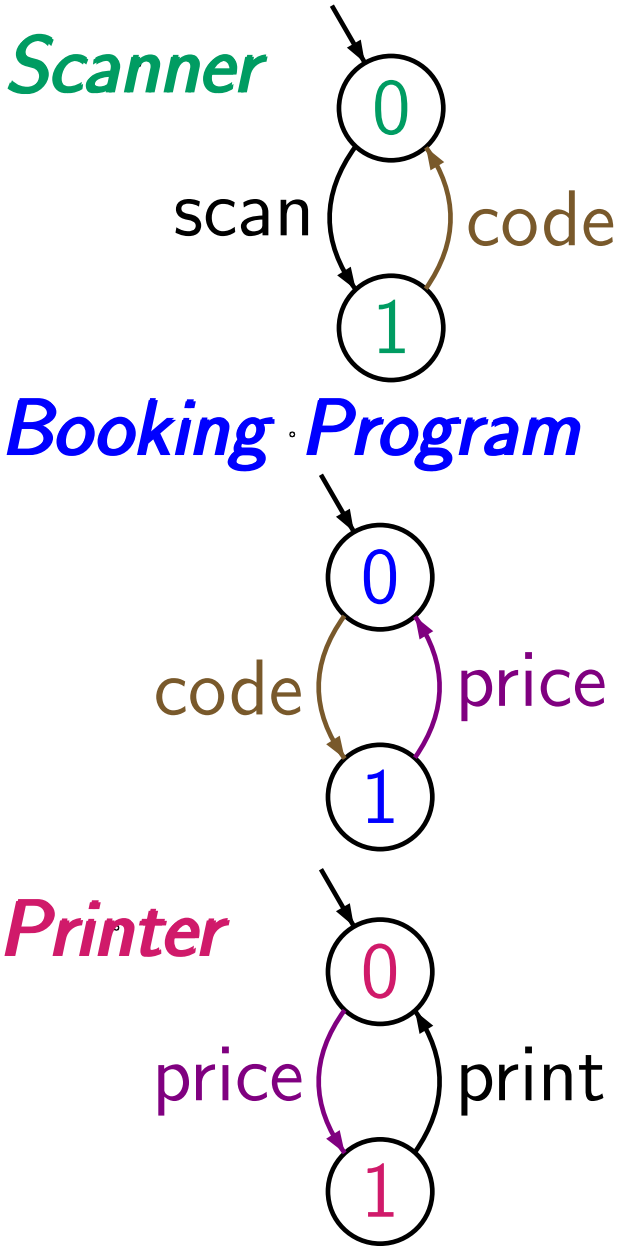
Write SOS rules for the parallel operator ?

$$\mathcal{T}_1 \| \mathcal{T}_2 \| \mathcal{T}_3 \| \mathcal{T}_4 \| \ldots \stackrel{\text{def}}{=} (((\mathcal{T}_1 \|_{Syn_{1,2}} \mathcal{T}_2) \|_{Syn_{1,2,3}} \mathcal{T}_3) \|_{Syn_{1,2,3,4}} \mathcal{T}_4) \ldots$$

where

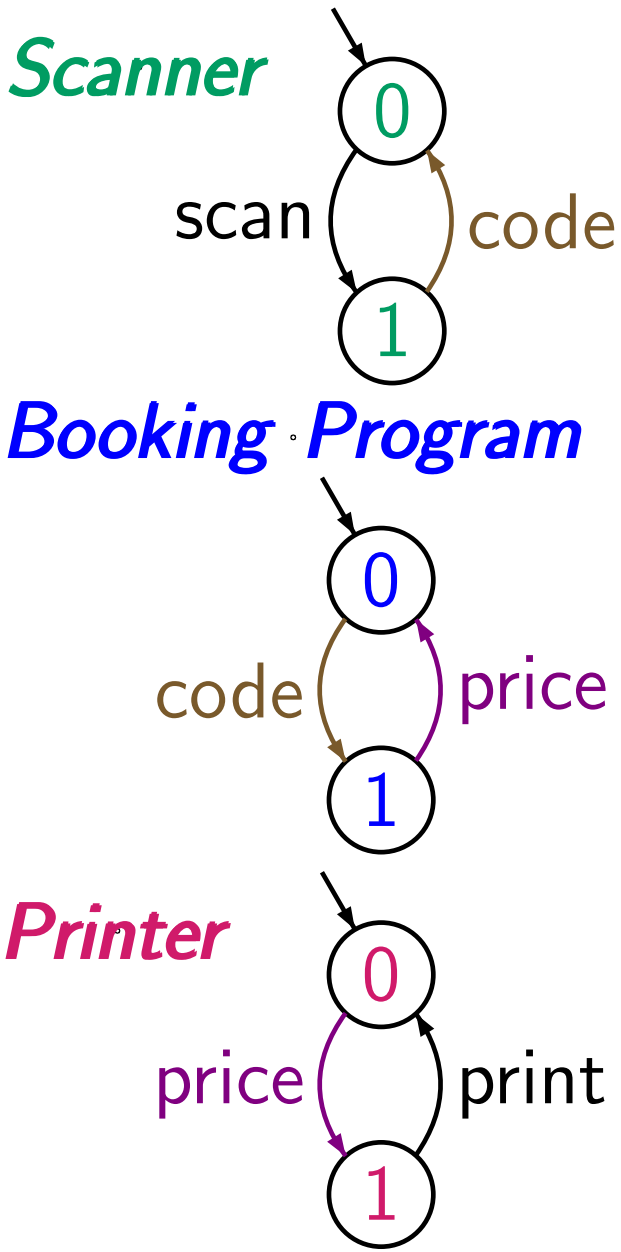
$$\begin{array}{lcl} Syn_{1,2} & = & Act_1 \cap Act_2 \\ Syn_{1,2,3} & = & (Act_1 \cup Act_2) \cap Act_3 \\ Syn_{1,2,3,4} & = & (Act_1 \cup Act_2 \cup Act_3) \cap Act_4 \\ \vdots & & \vdots \end{array}$$

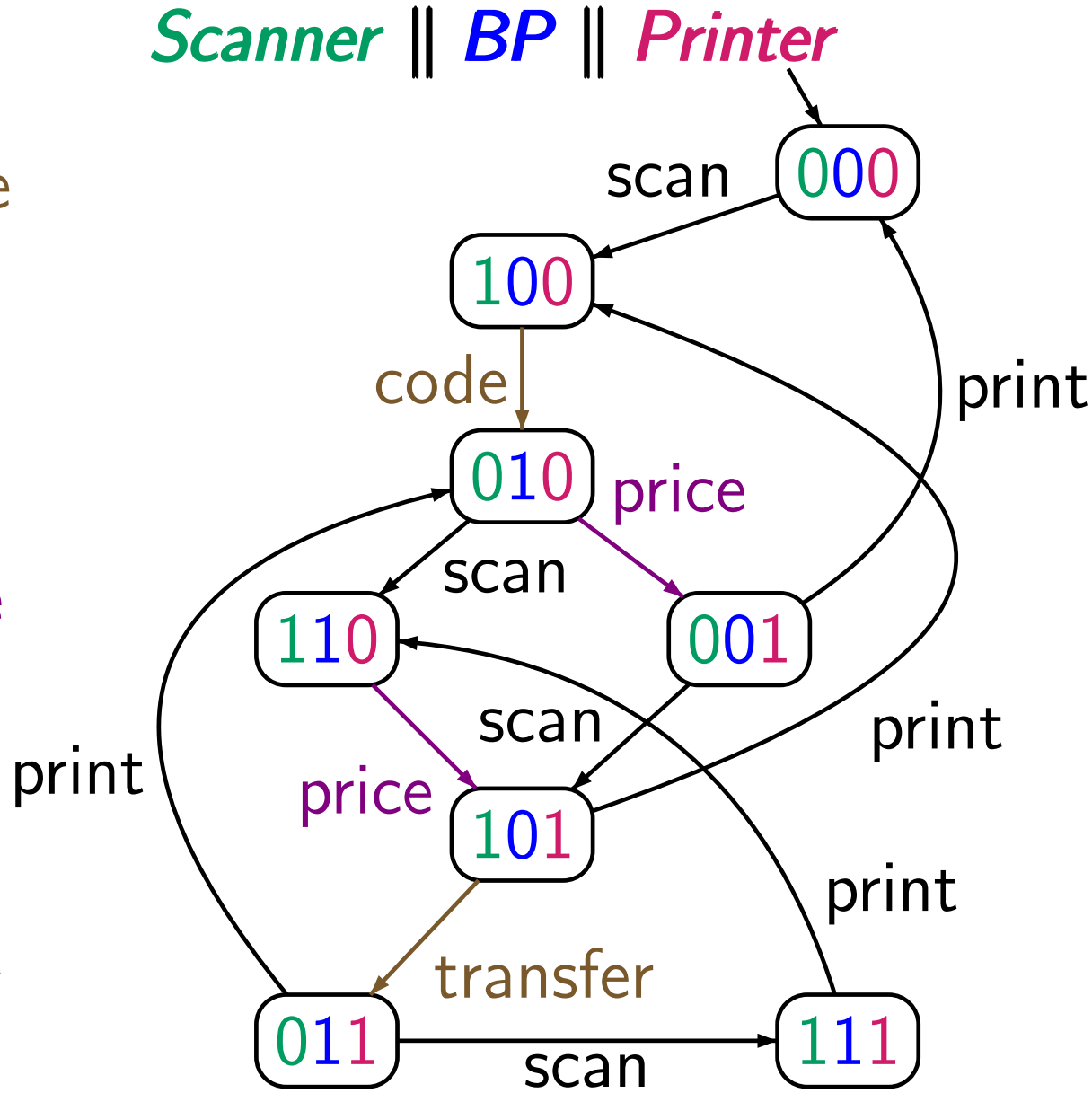
Booking system in supermarket



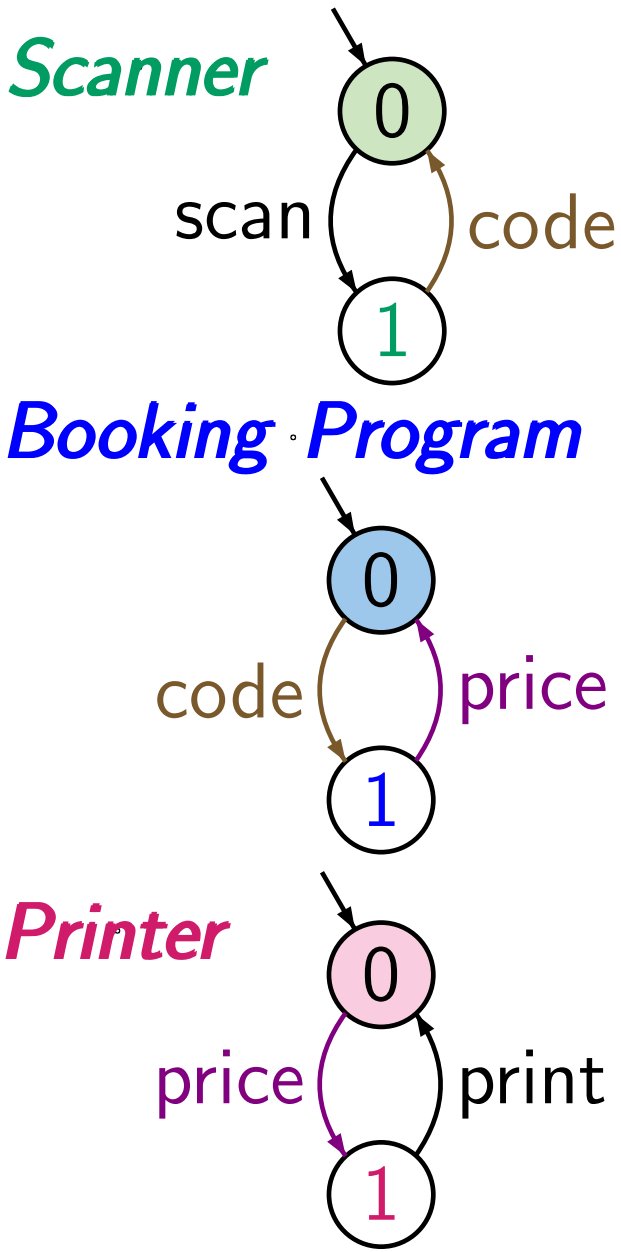
Scanner ‖ BP ‖ Printer

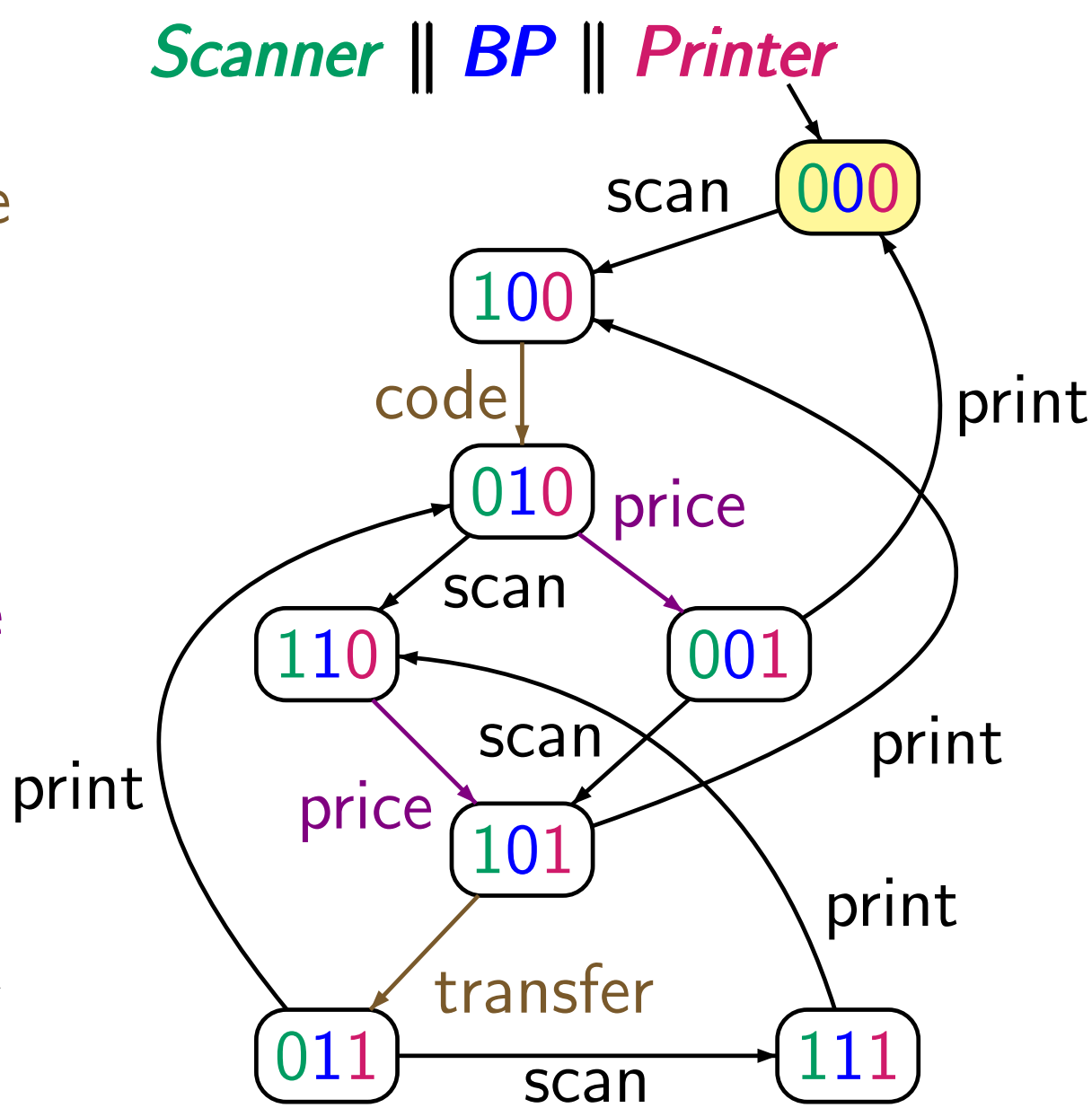
Booking system in supermarket



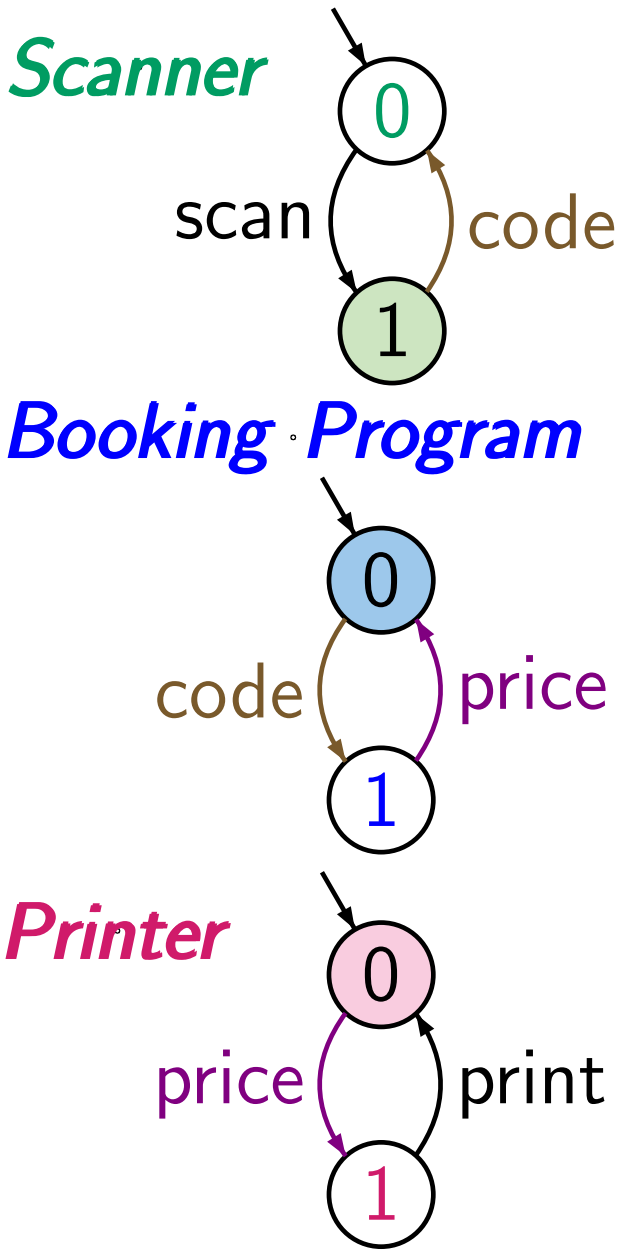


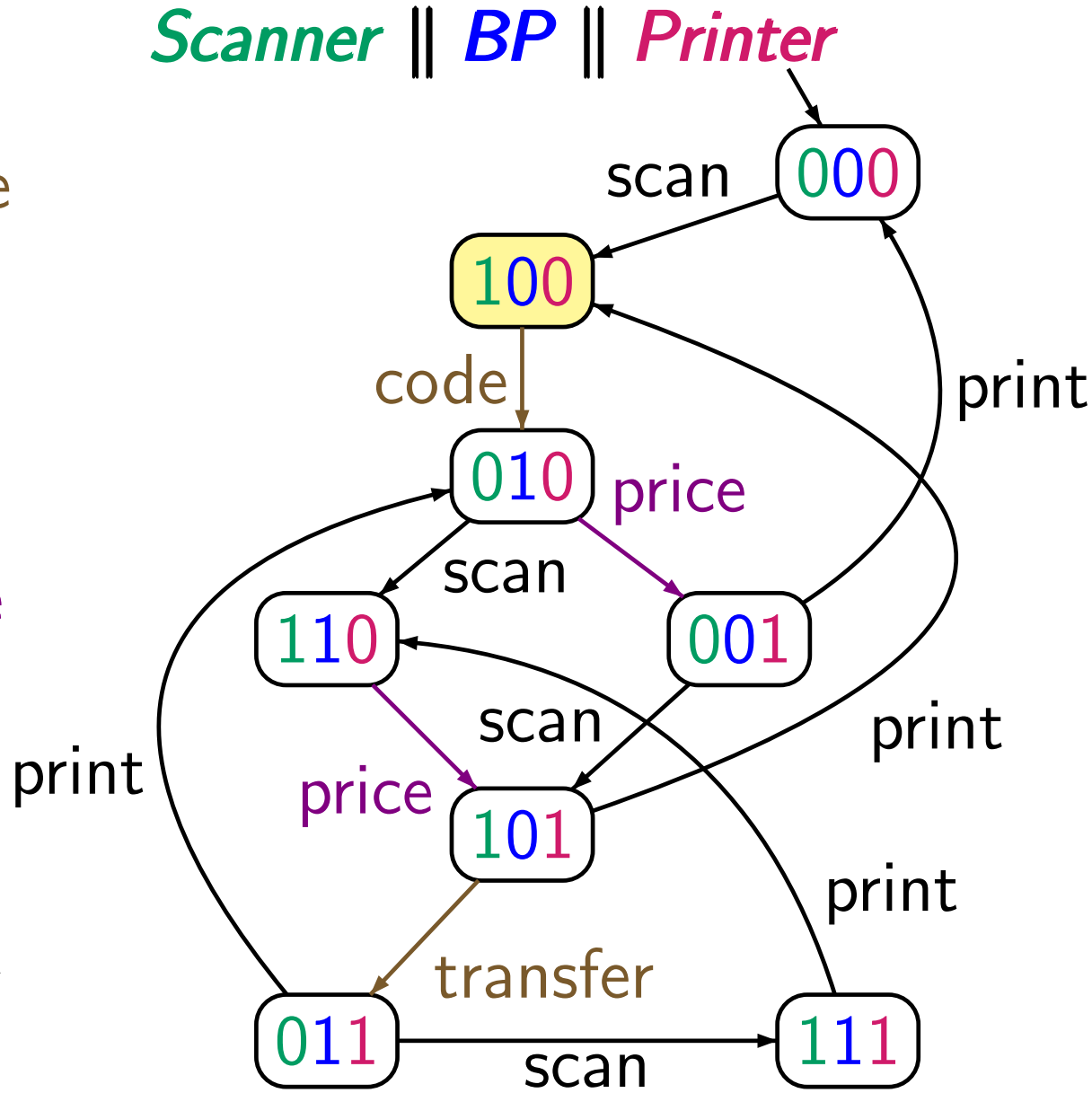
Booking system in supermarket



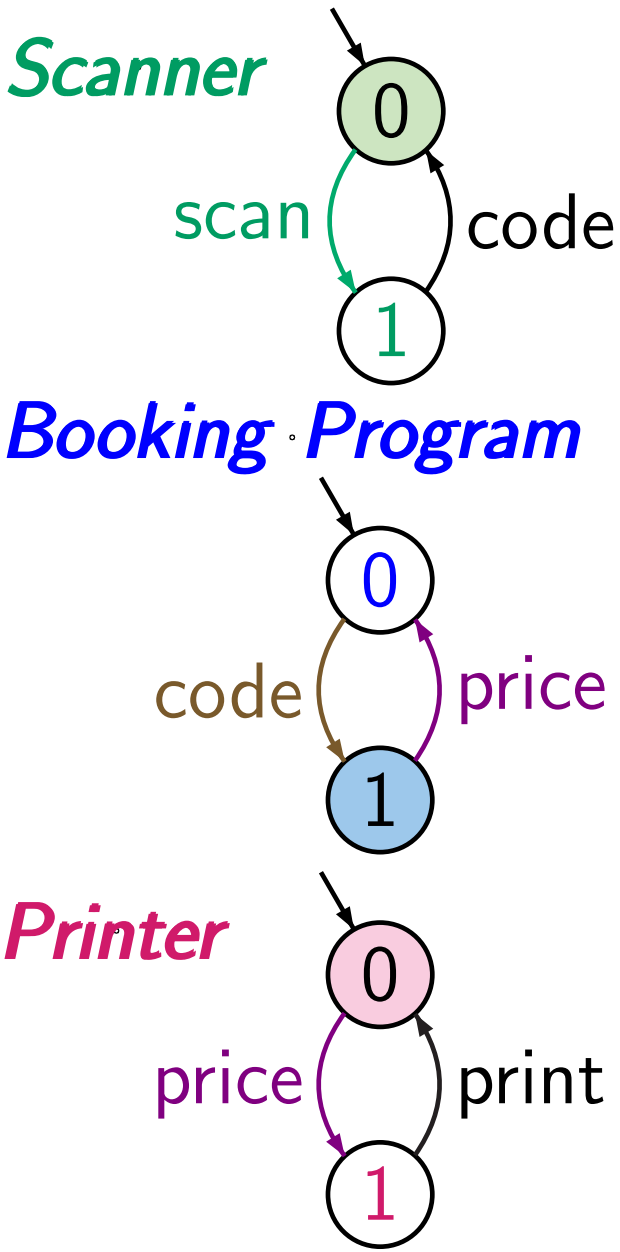


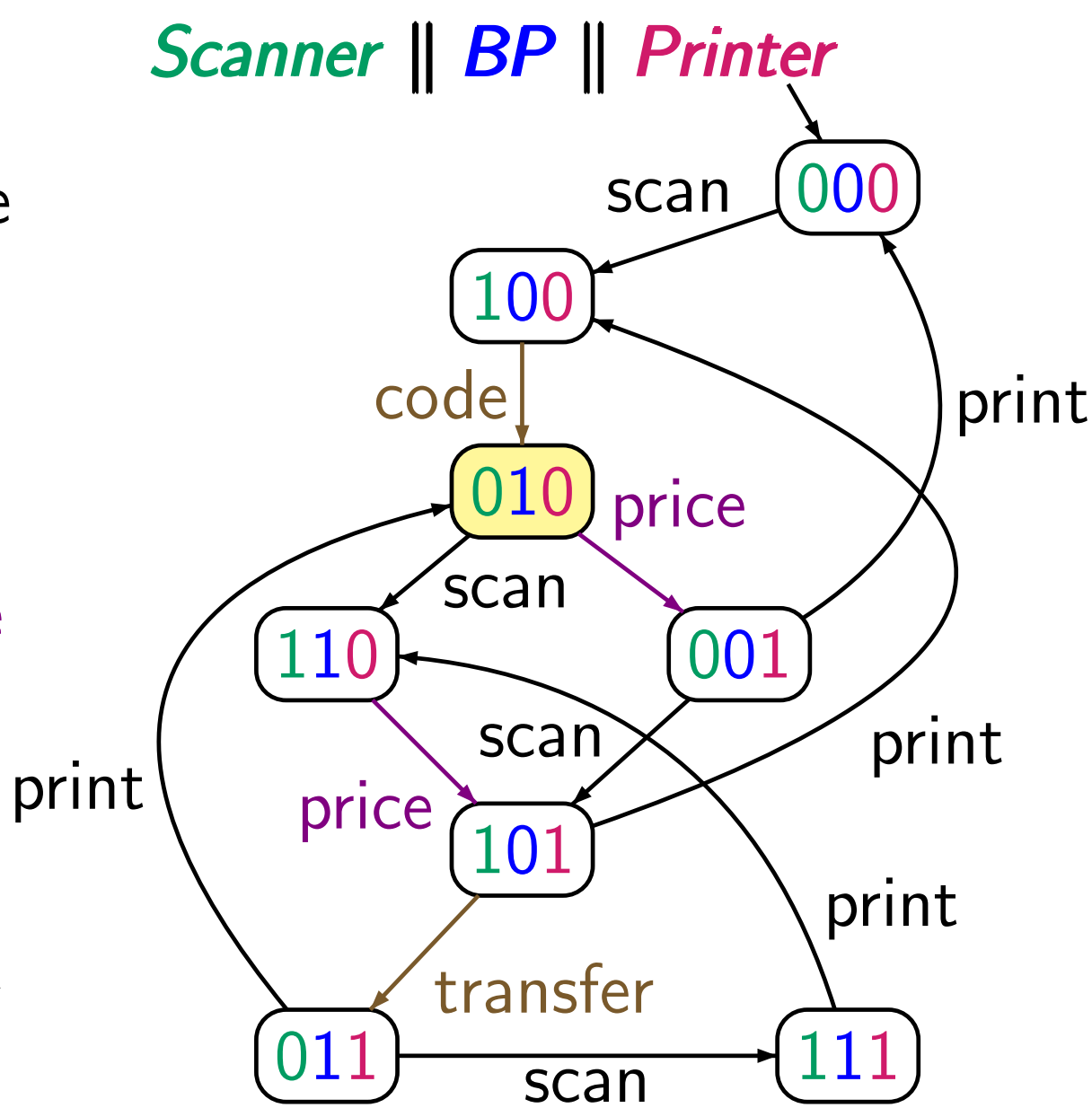
Booking system in supermarket



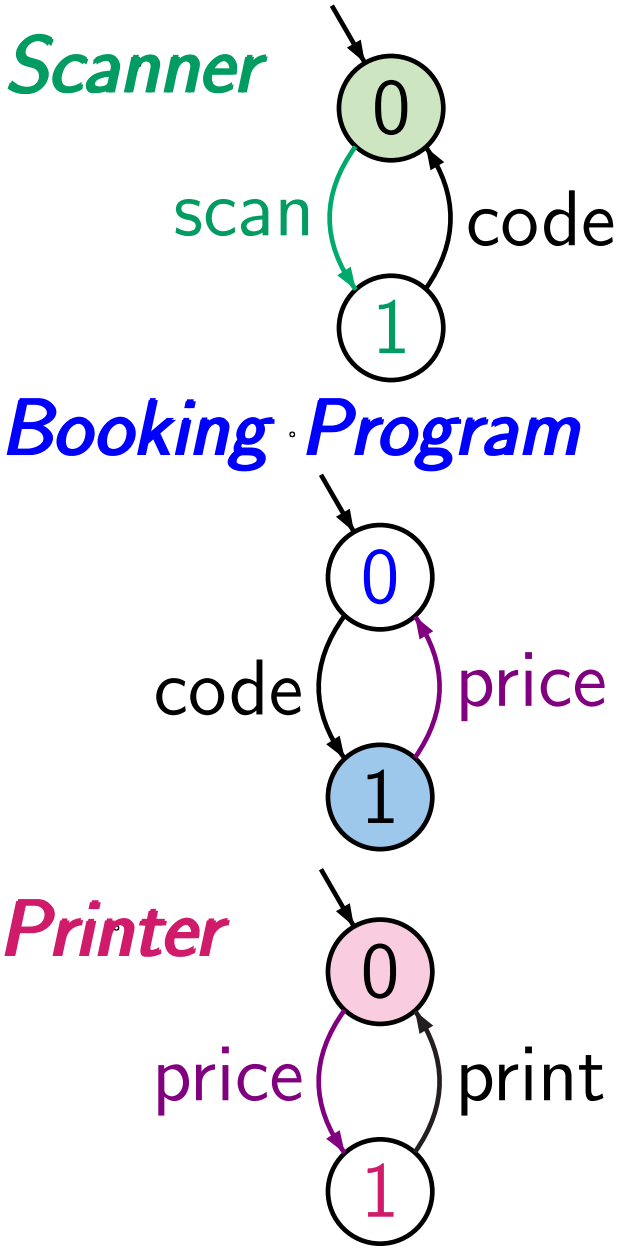


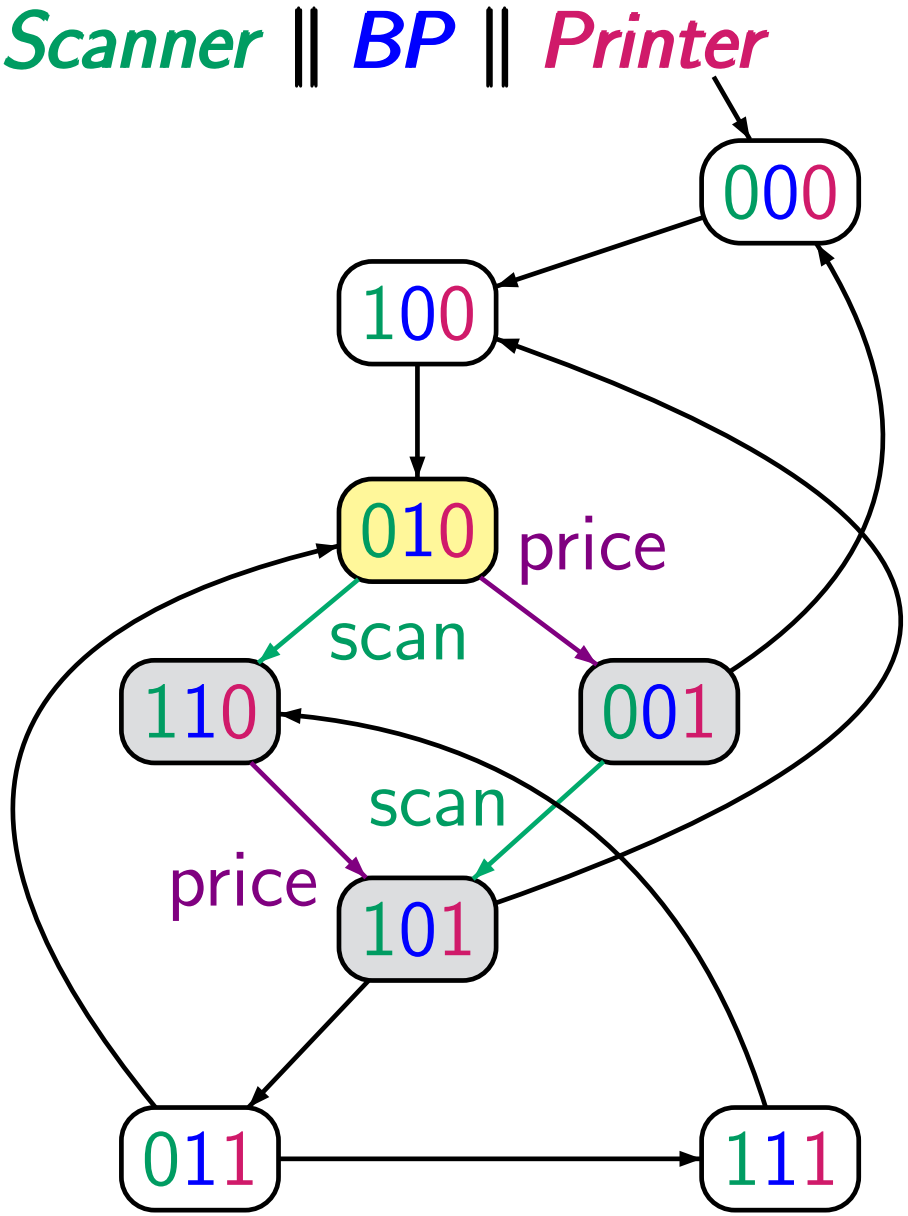
Booking system in supermarket



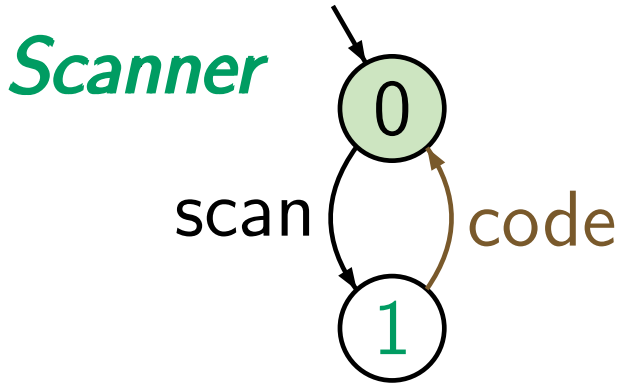


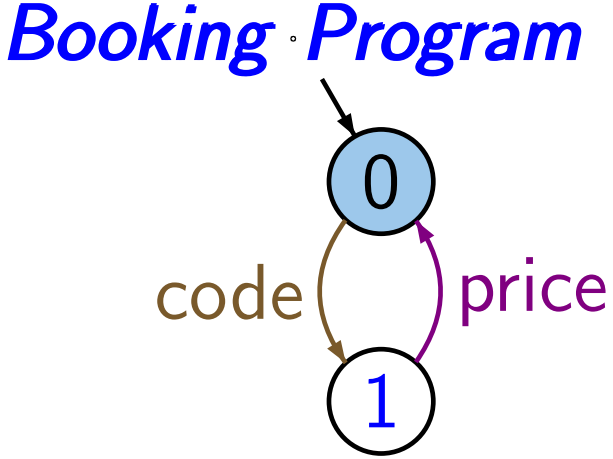
Interleaving

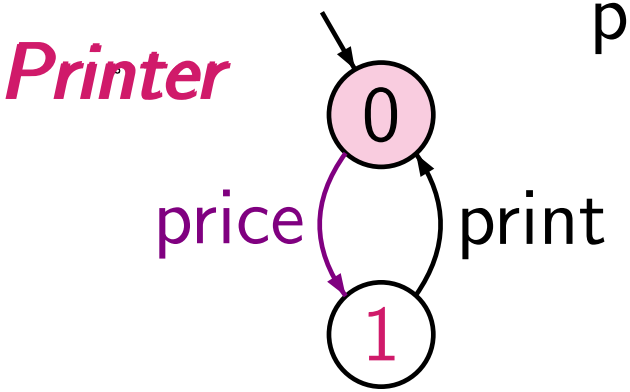


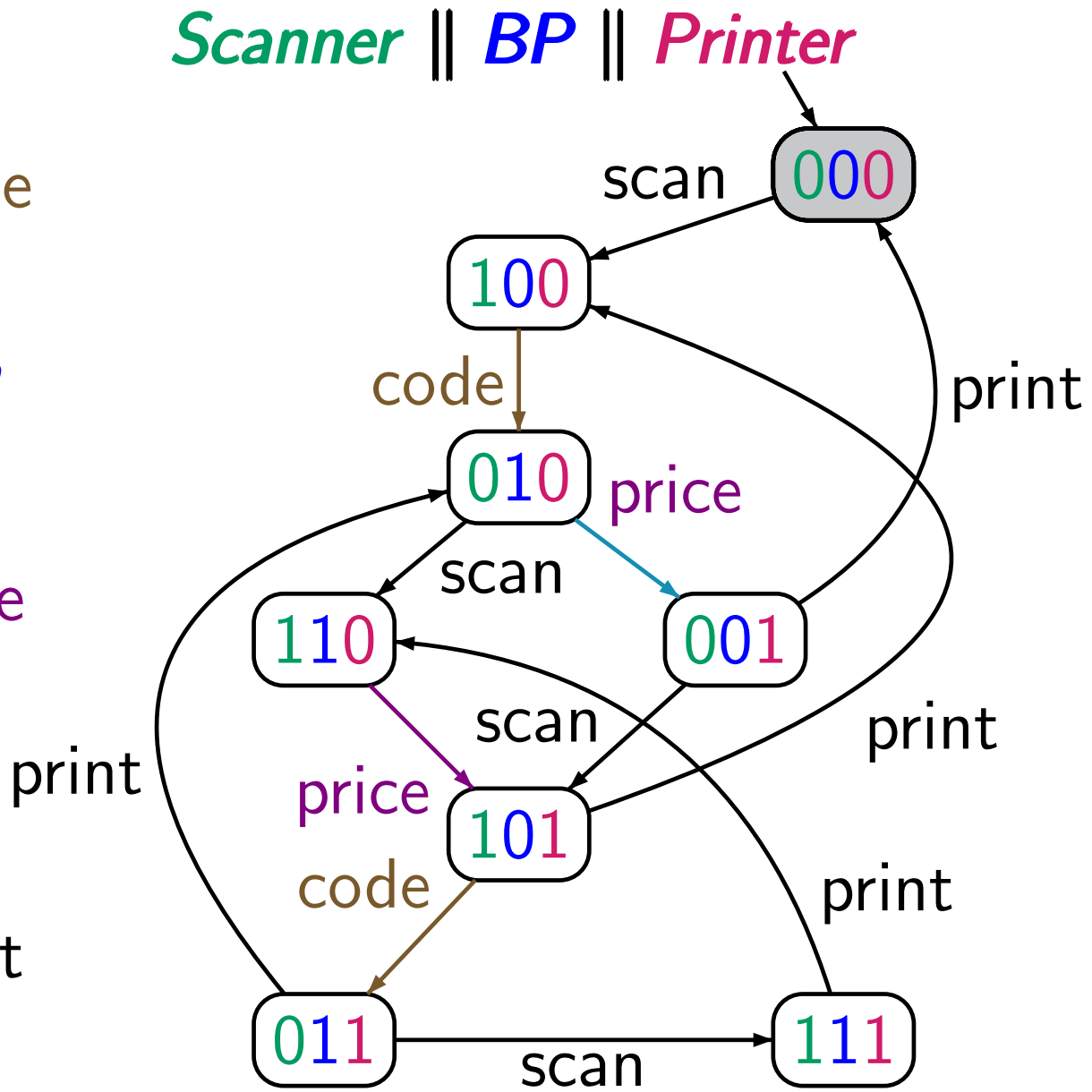


Booking system in supermarket

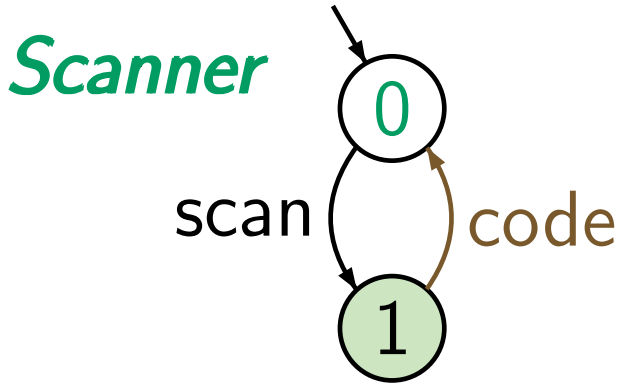


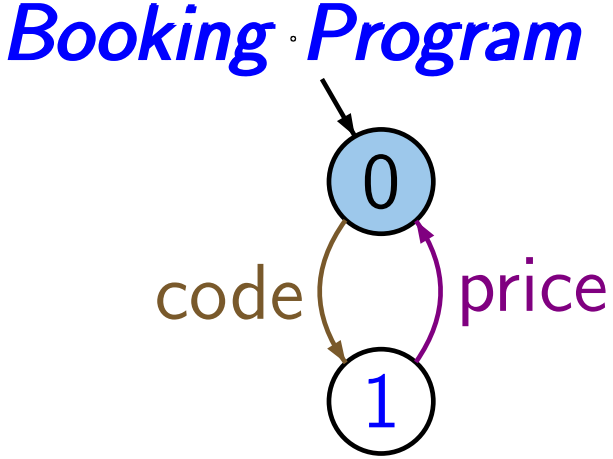


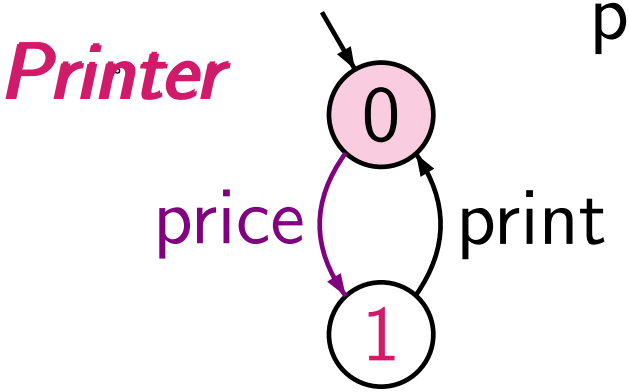


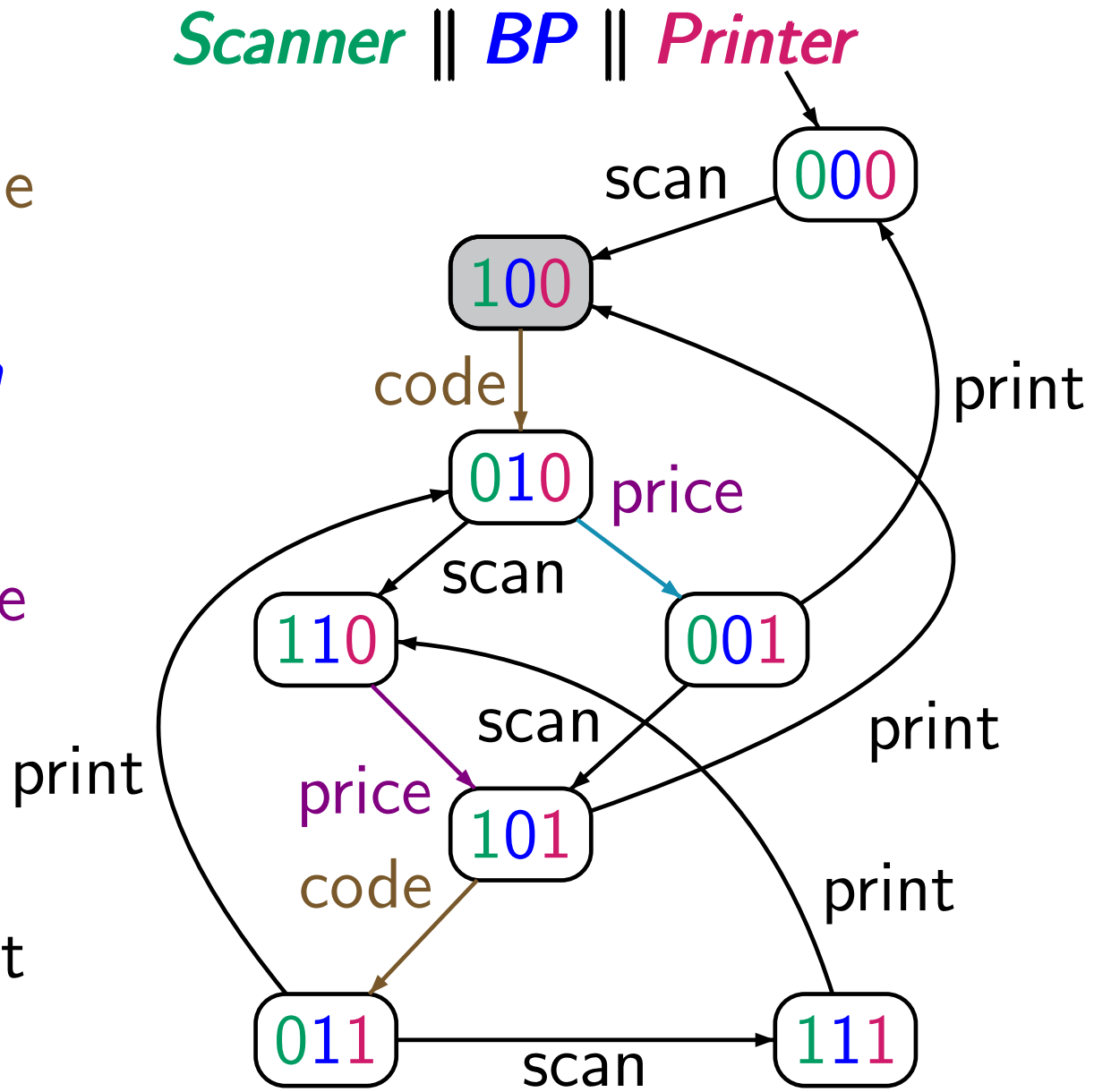


Booking system in supermarket

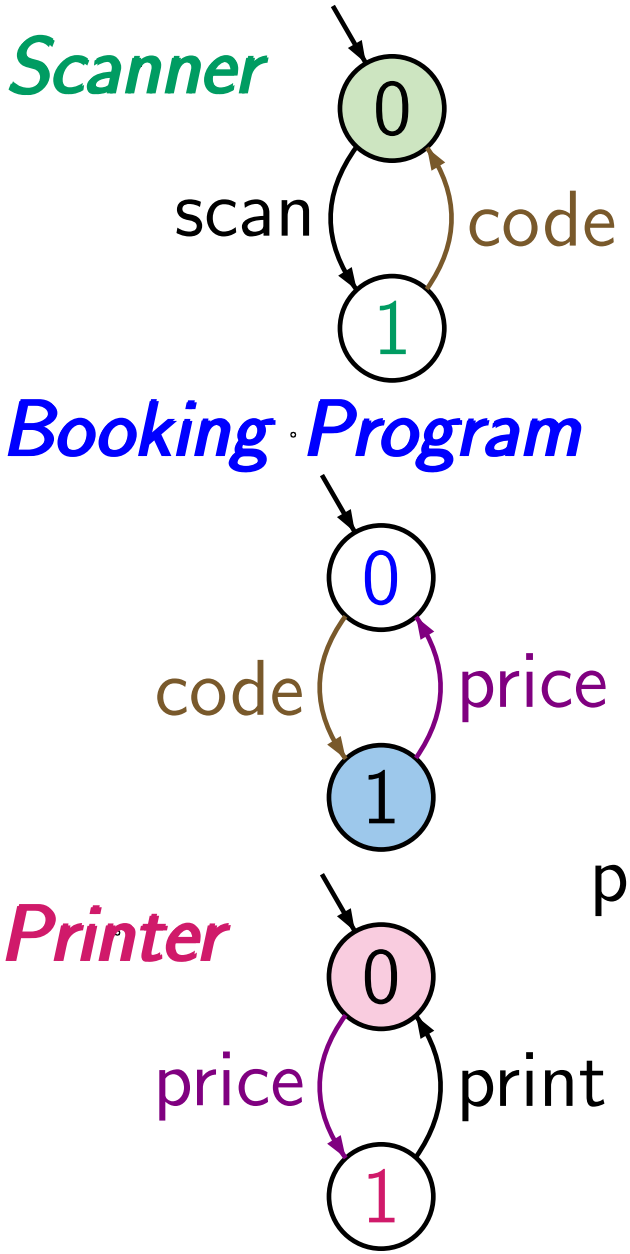


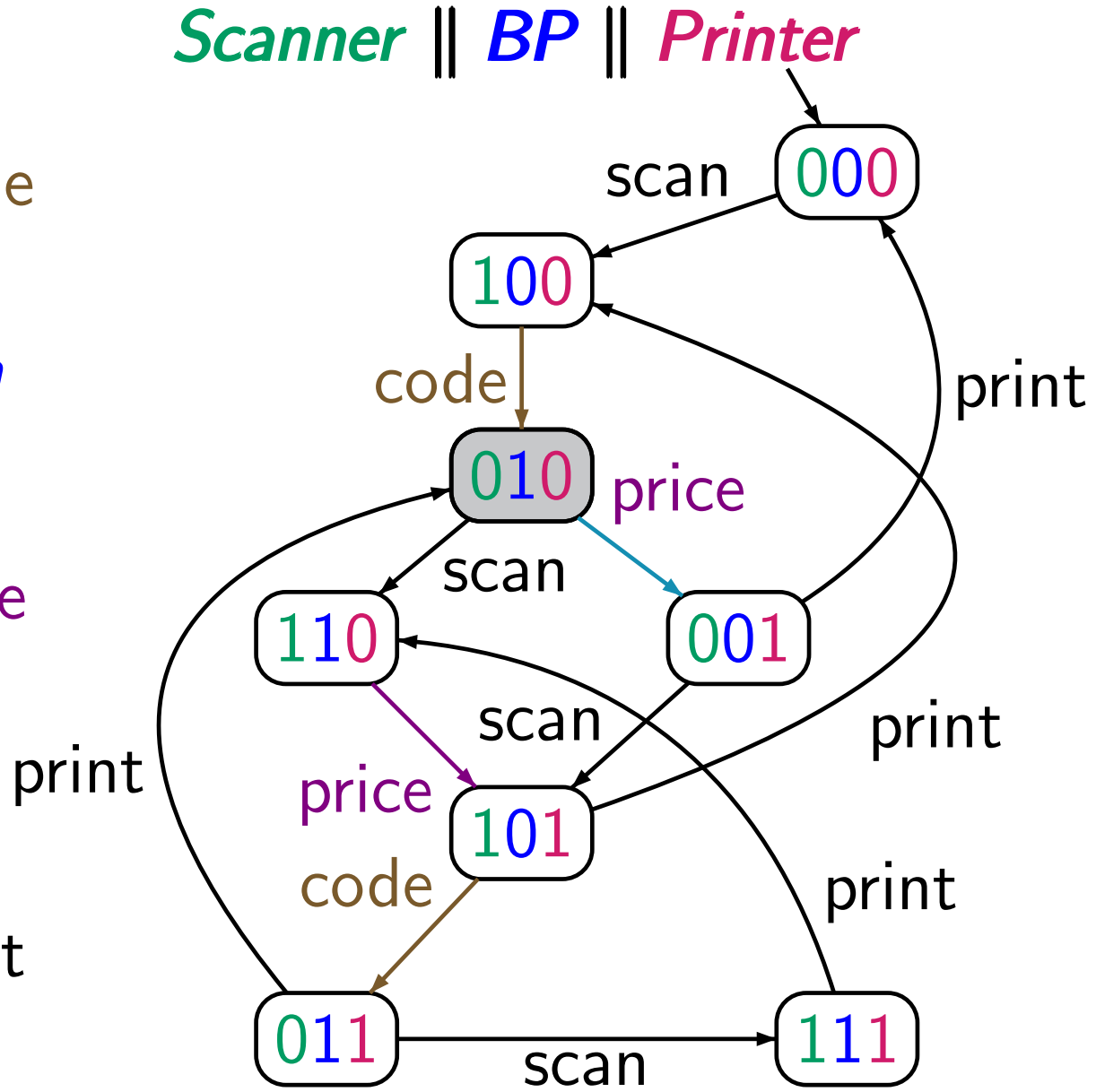




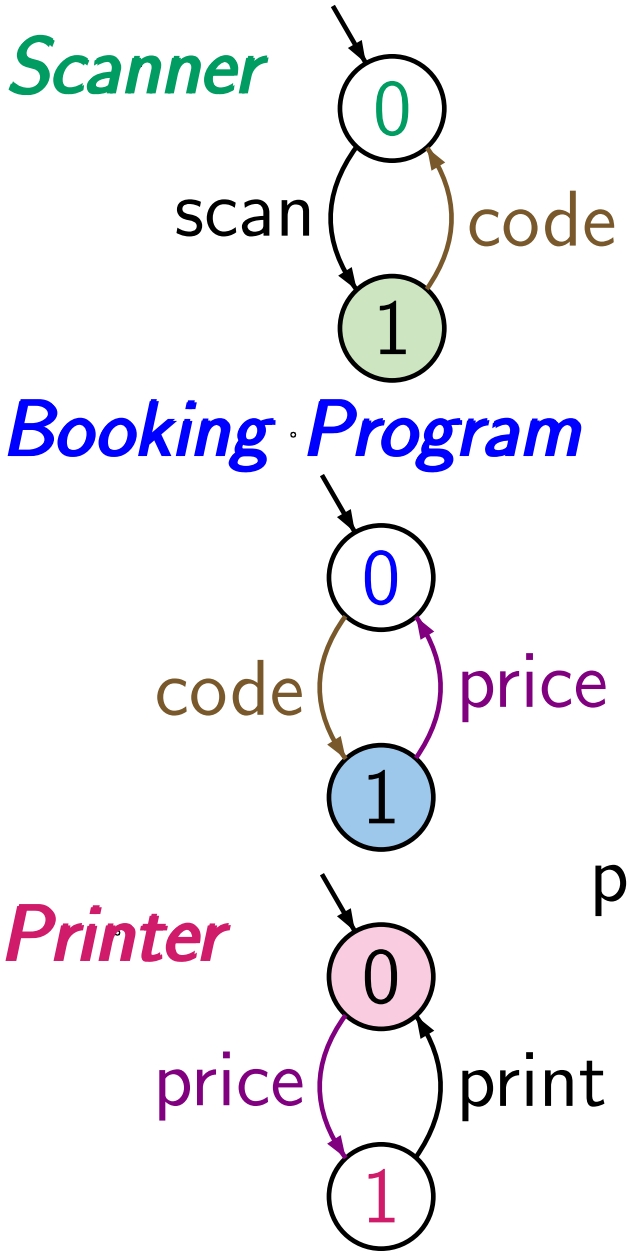


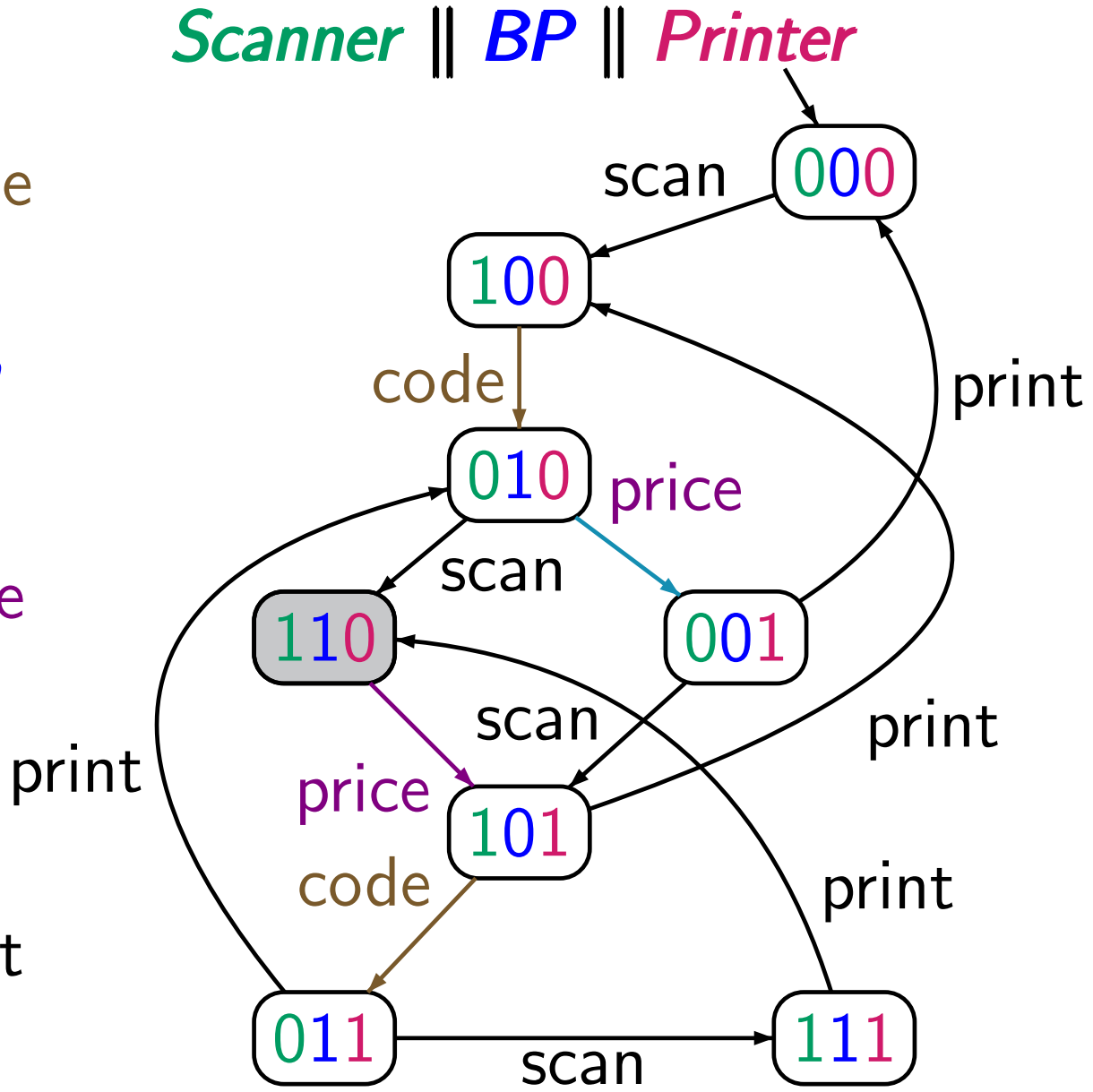
Booking system in supermarket



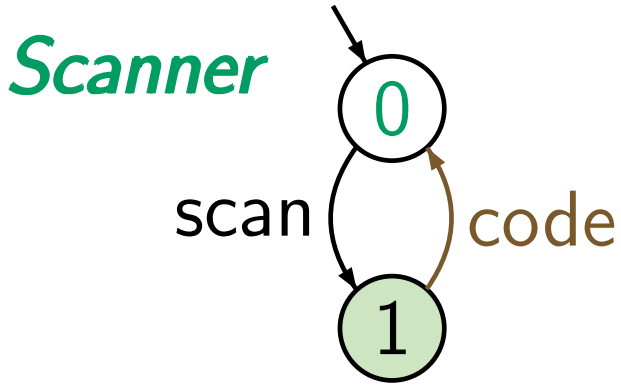


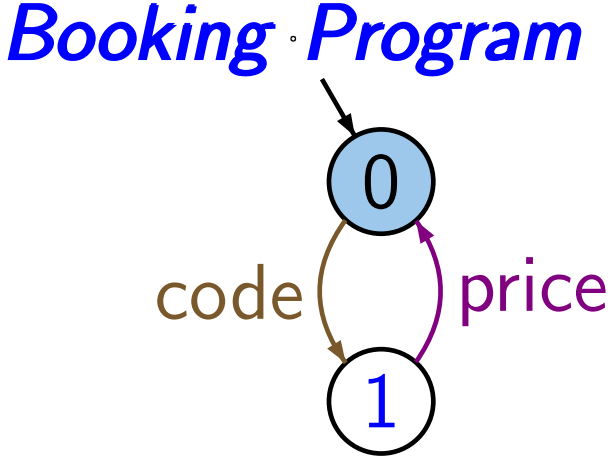
Booking system in supermarket

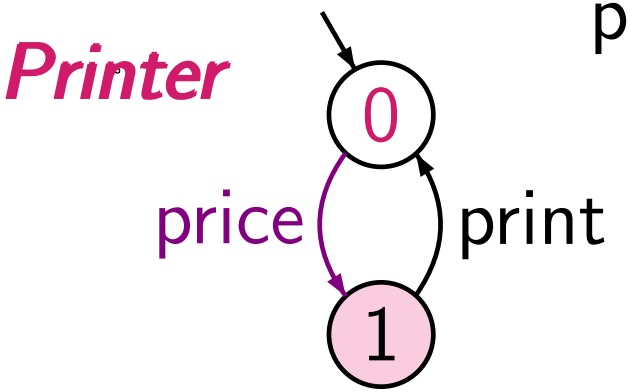


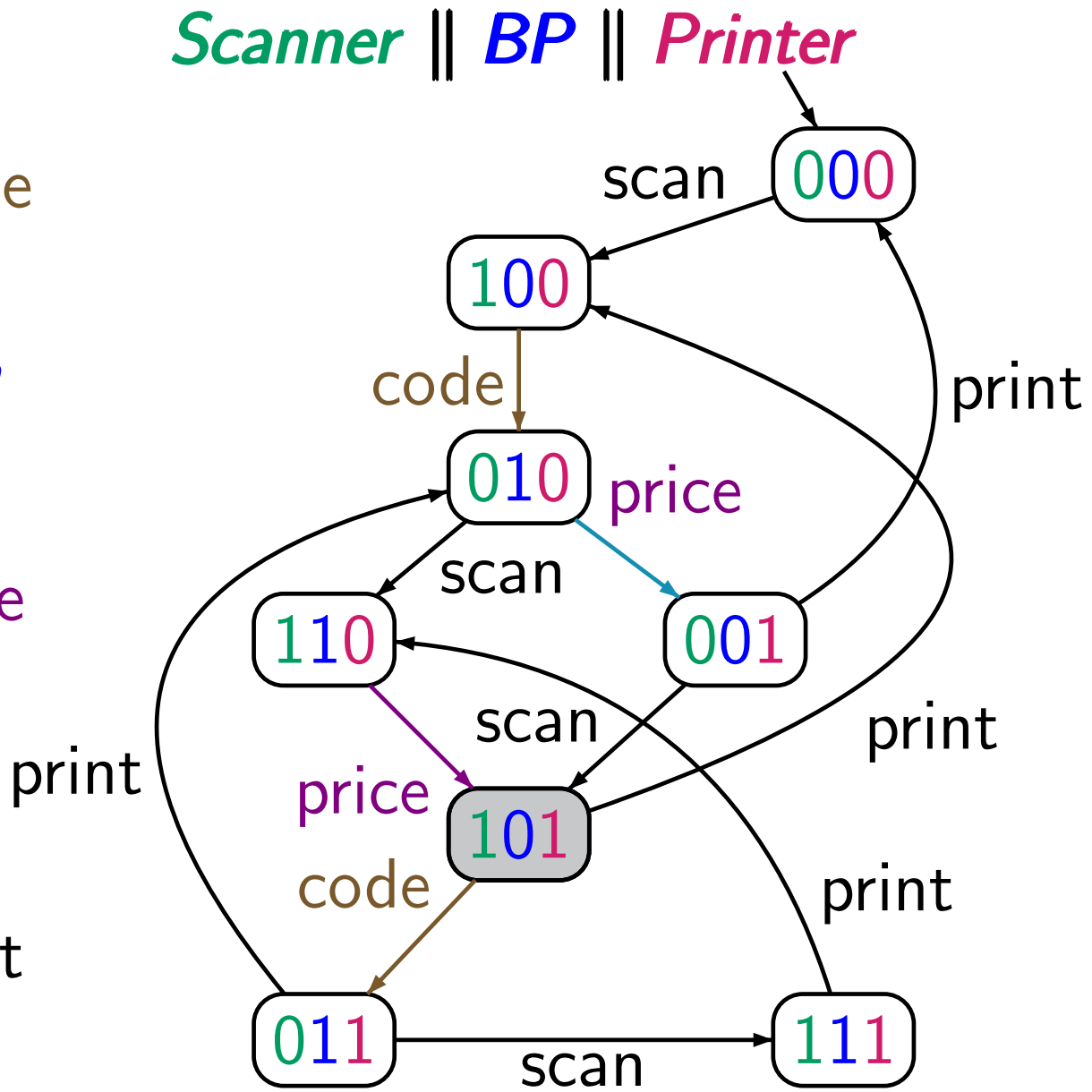


Booking system in supermarket

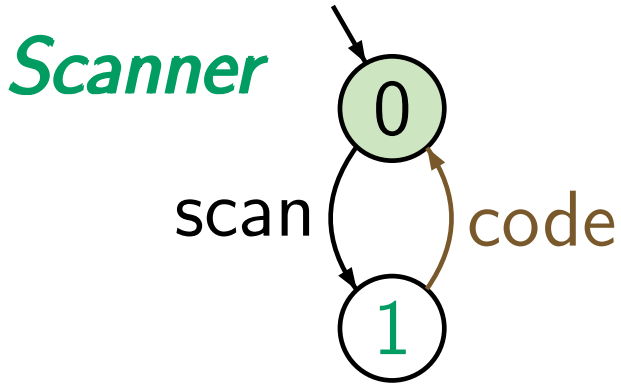


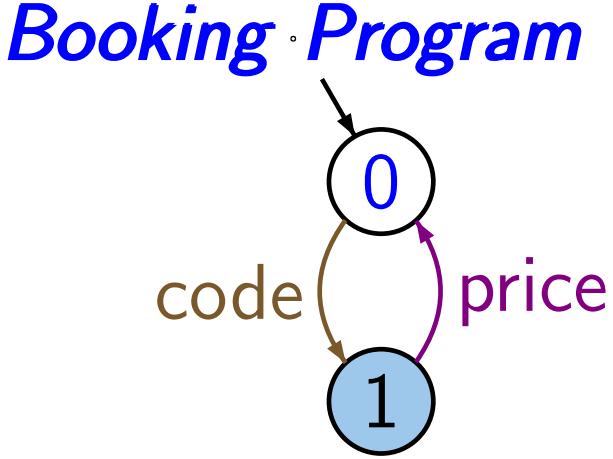


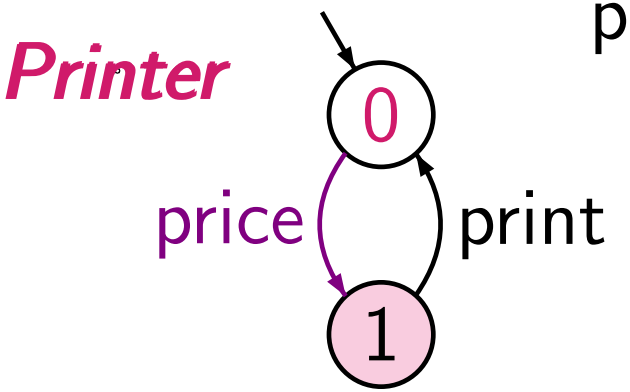


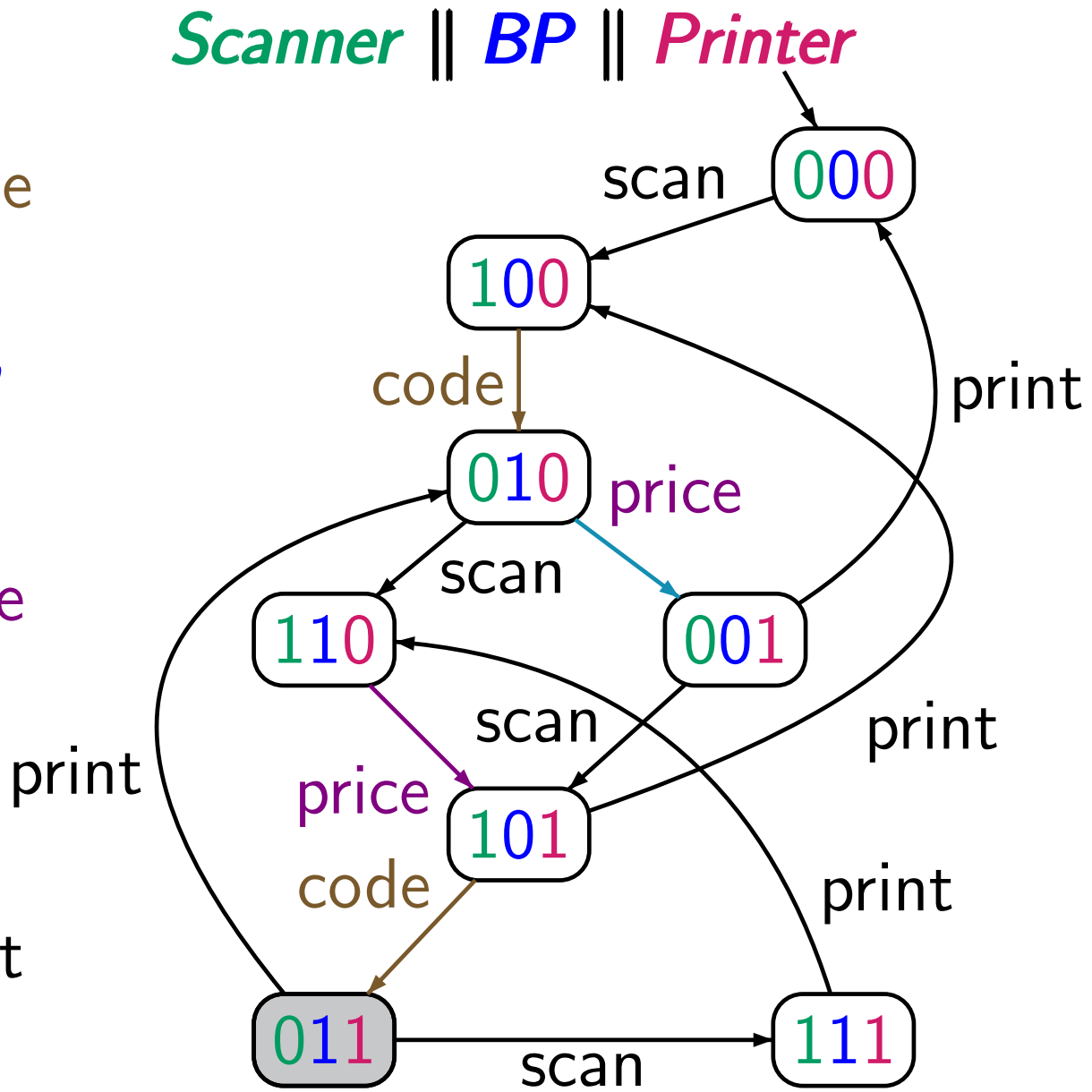


Booking system in supermarket

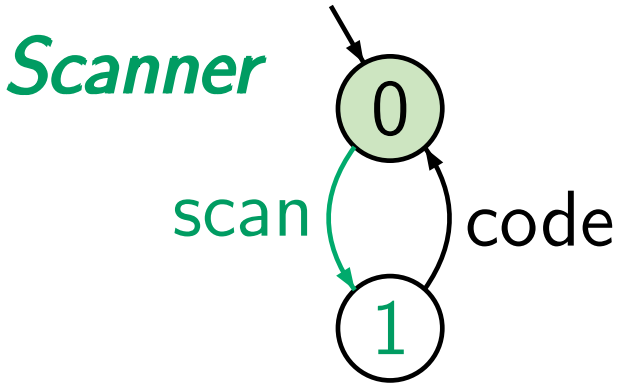


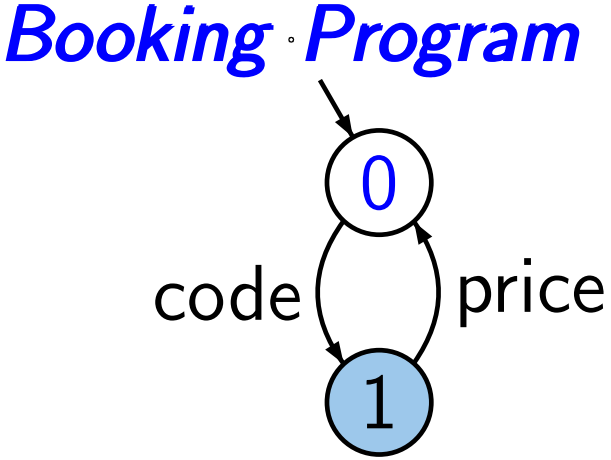


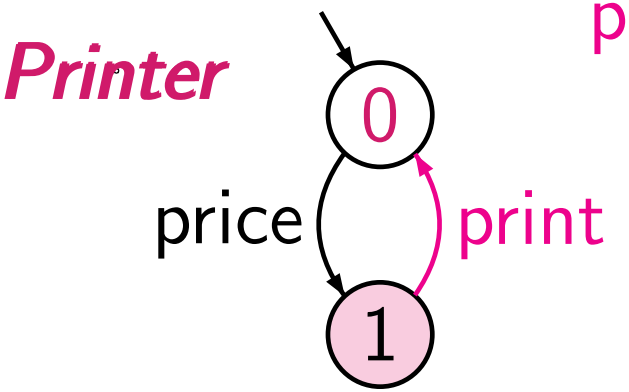


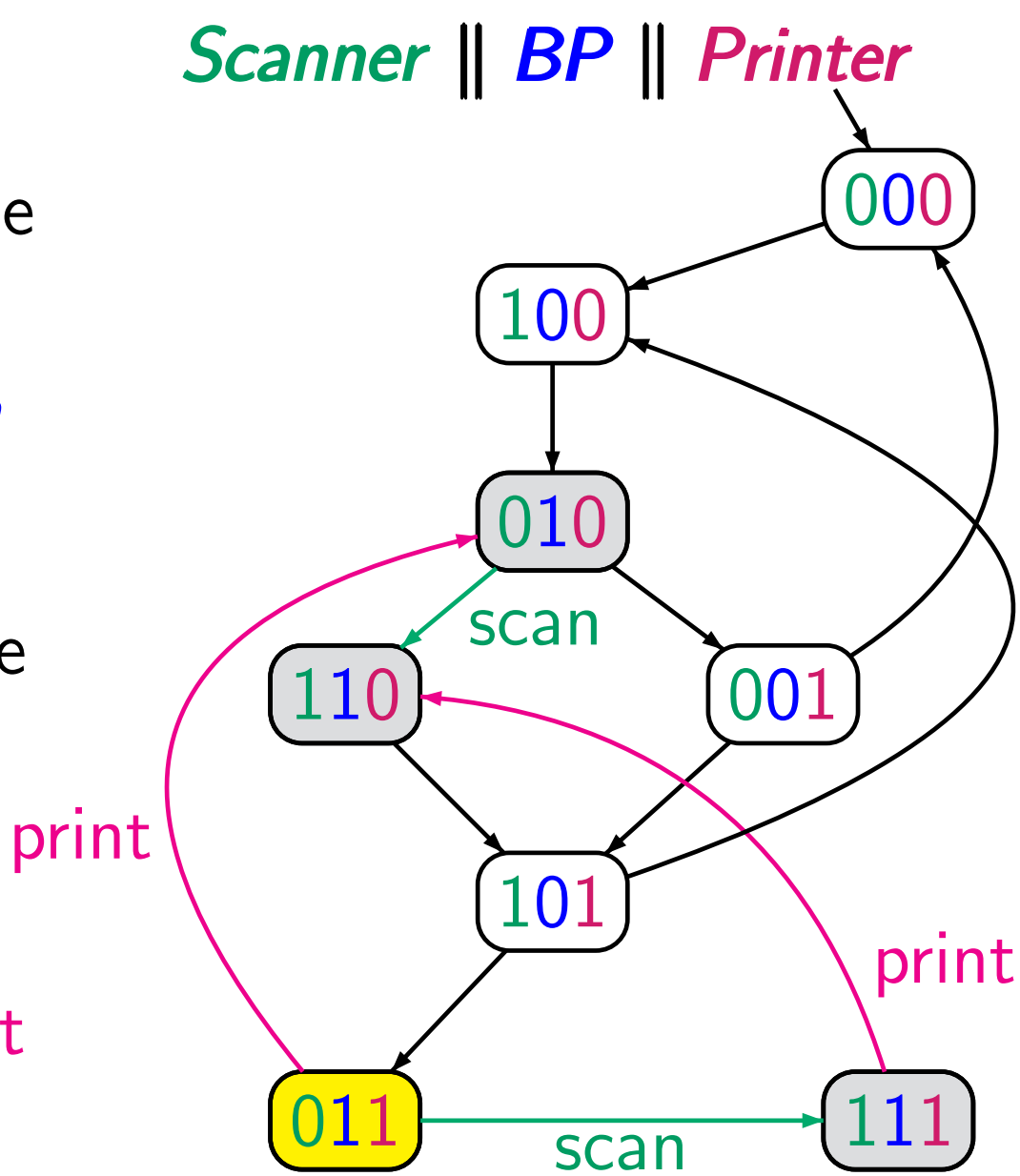


Interleaving

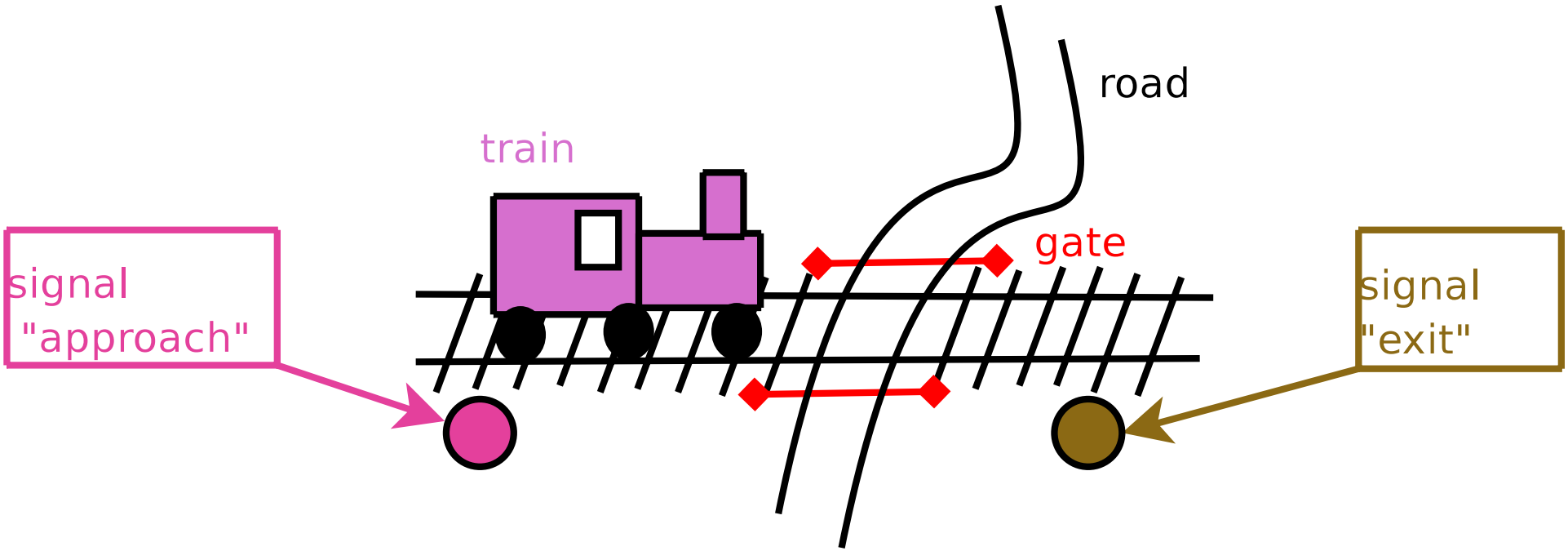




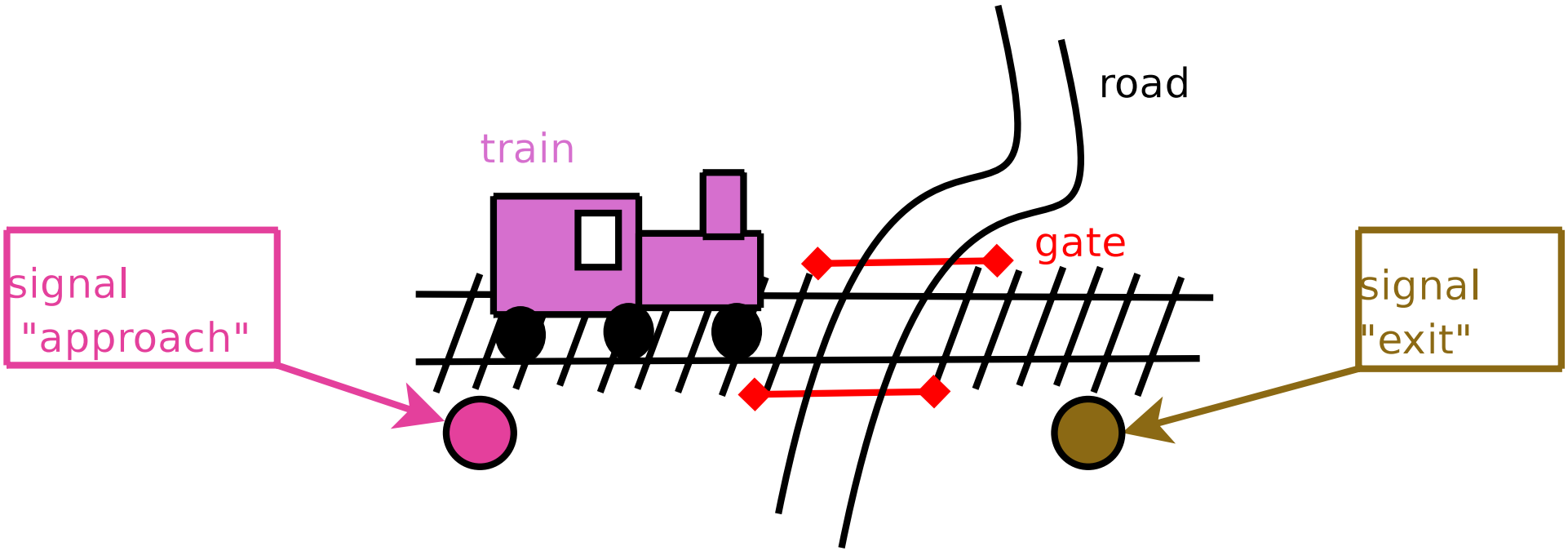




Railroad crossing



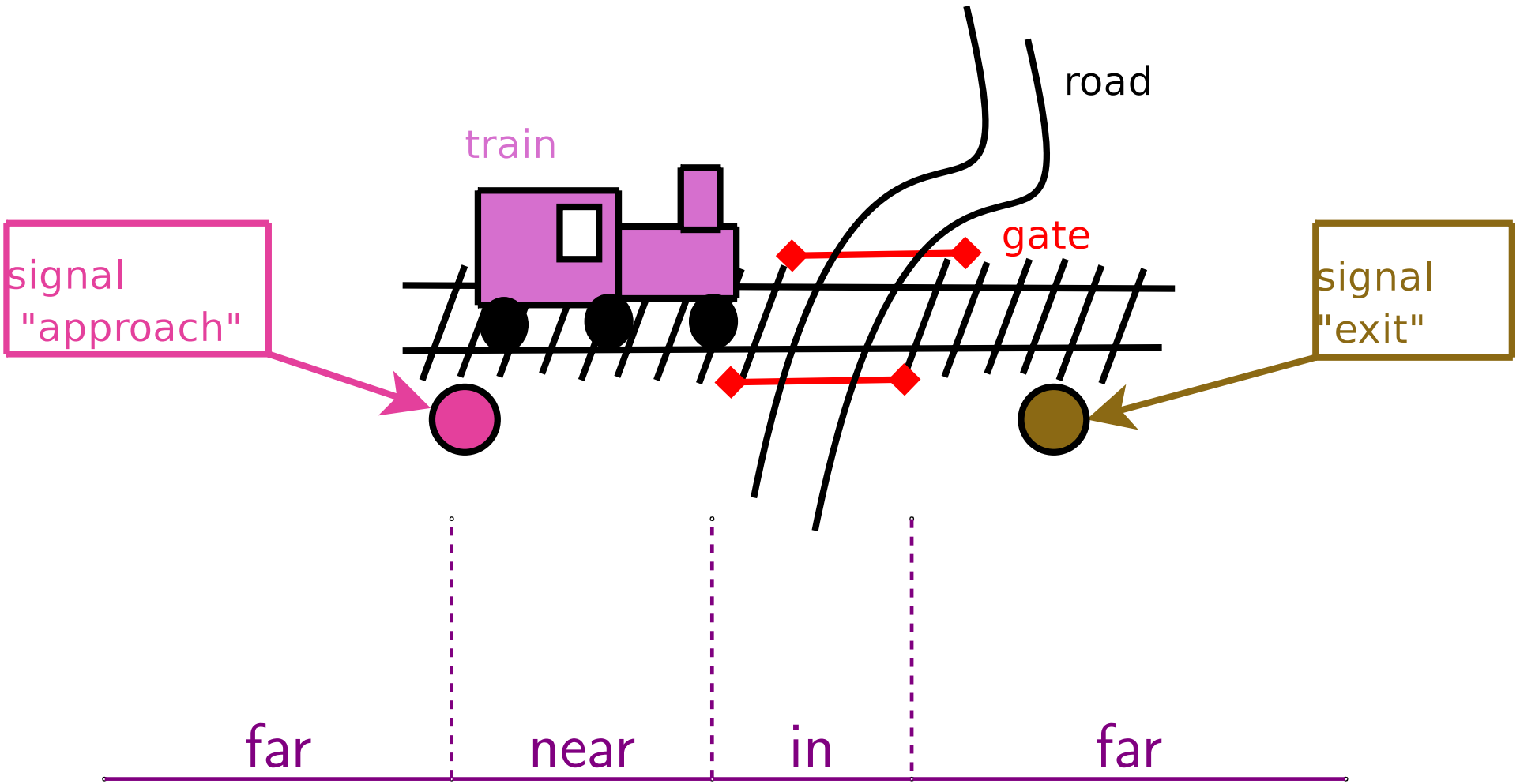
Railroad crossing



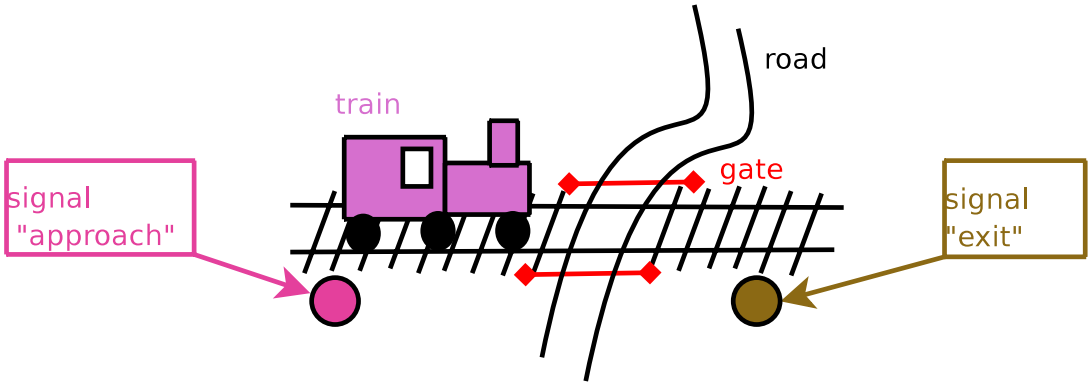
modeling by a transition system with **3** processes:

Train ‖ ***Controller*** ‖ ***Gate***

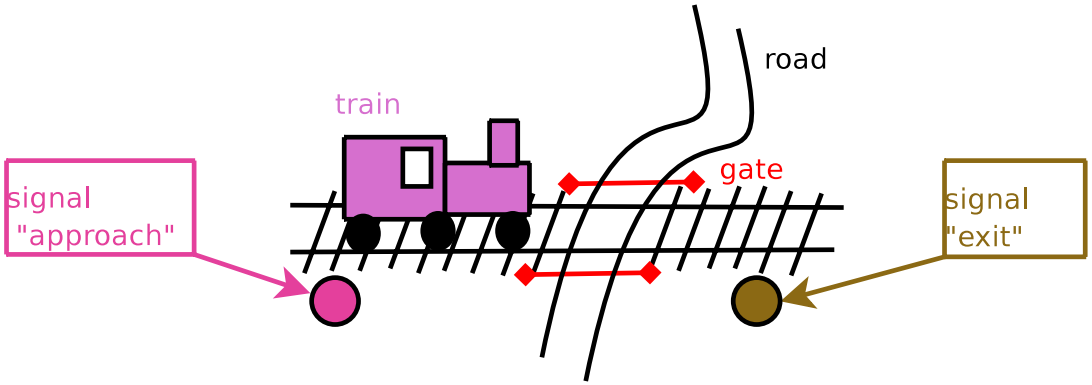
Modeling the behavior of trains



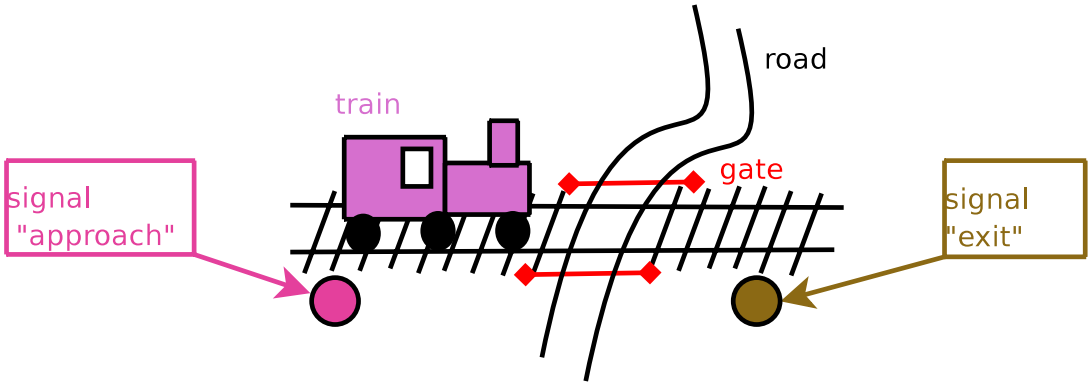
TS for the trains

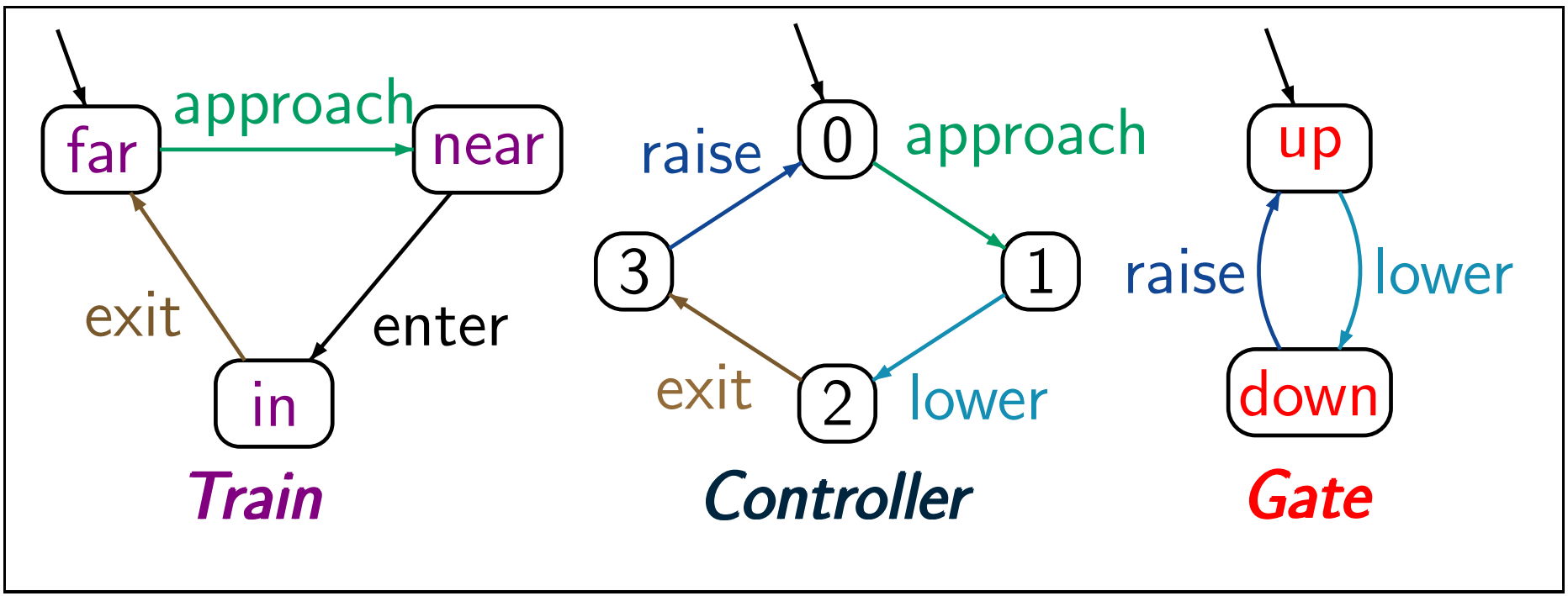


TS for the trains and gate

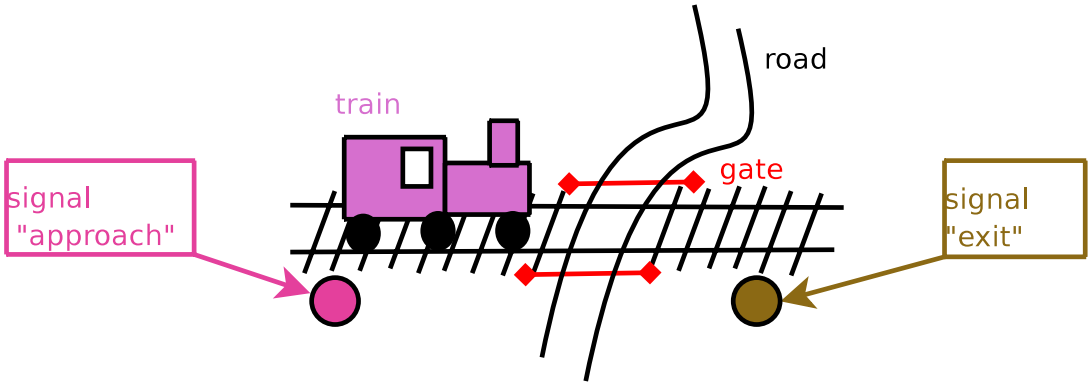


TS for the trains, controller and gate

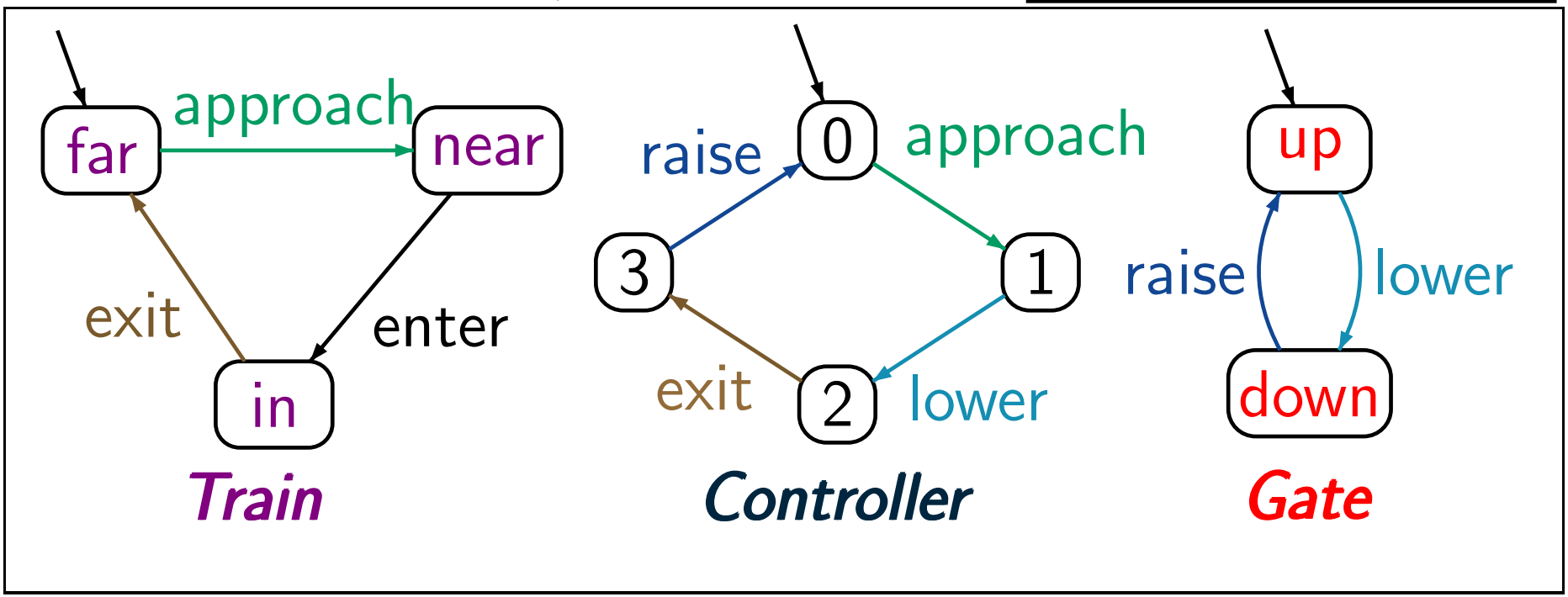




TS for the trains, controller and gate

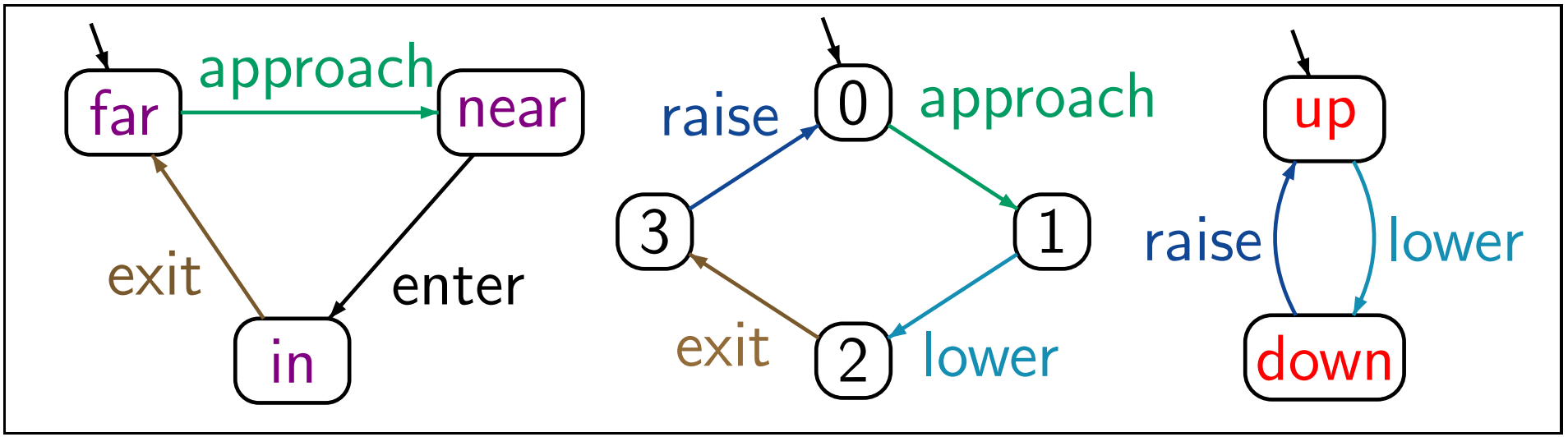


The controller must guarantee that the gate is always closed when the train is crossing



transition system *Train* ‖ *Controller* ‖ *Gate*

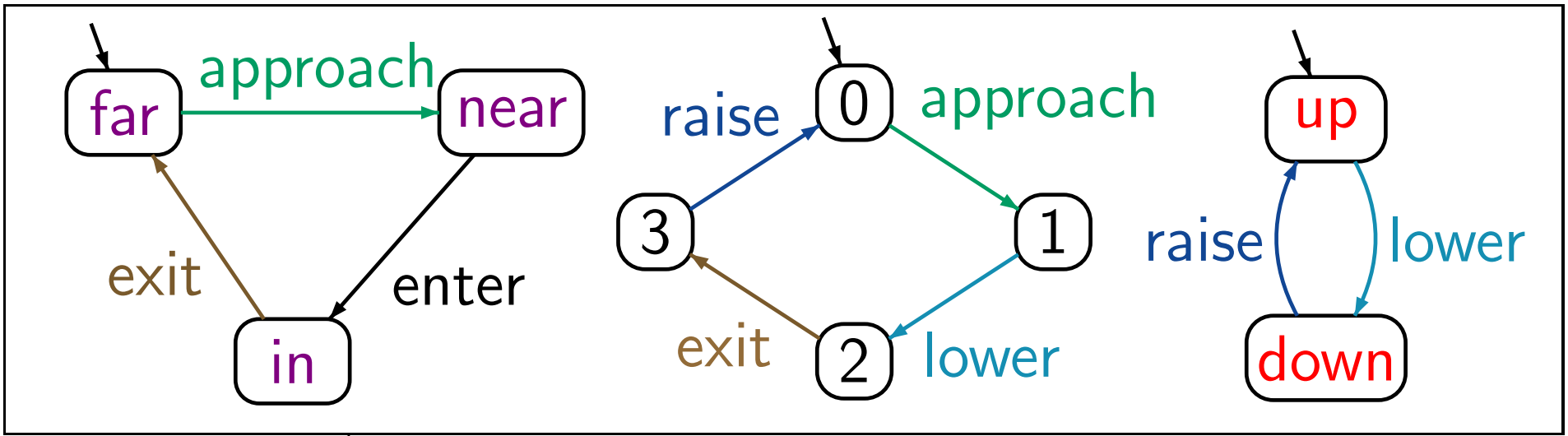
TS for railroad crossing

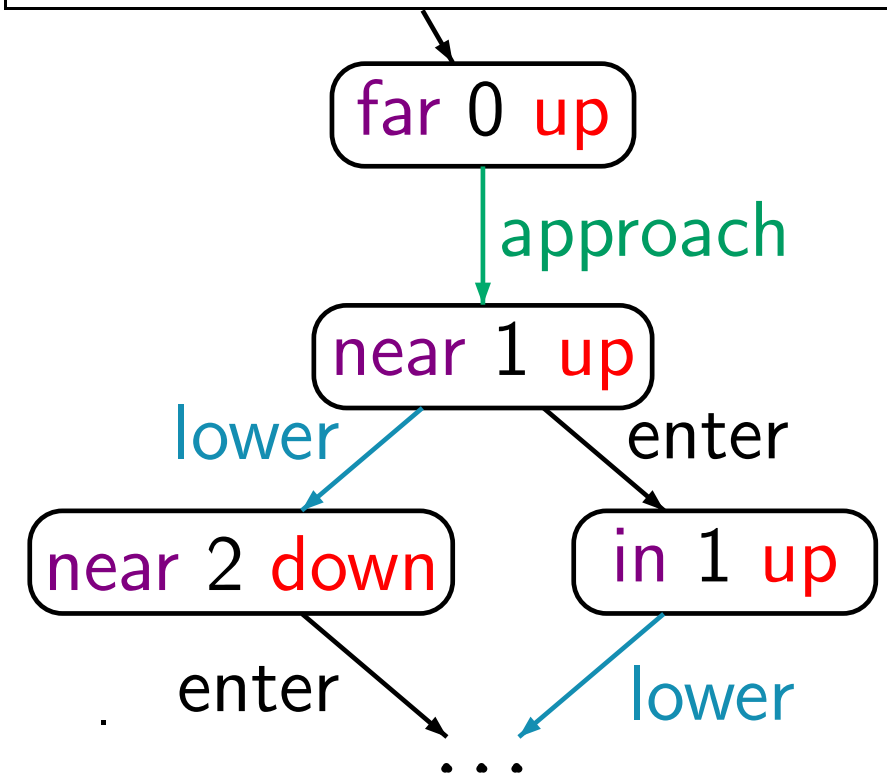


reachable fragment
of the transition system

Train ‖ ***Controller*** ‖ ***Gate***

TS for railroad crossing

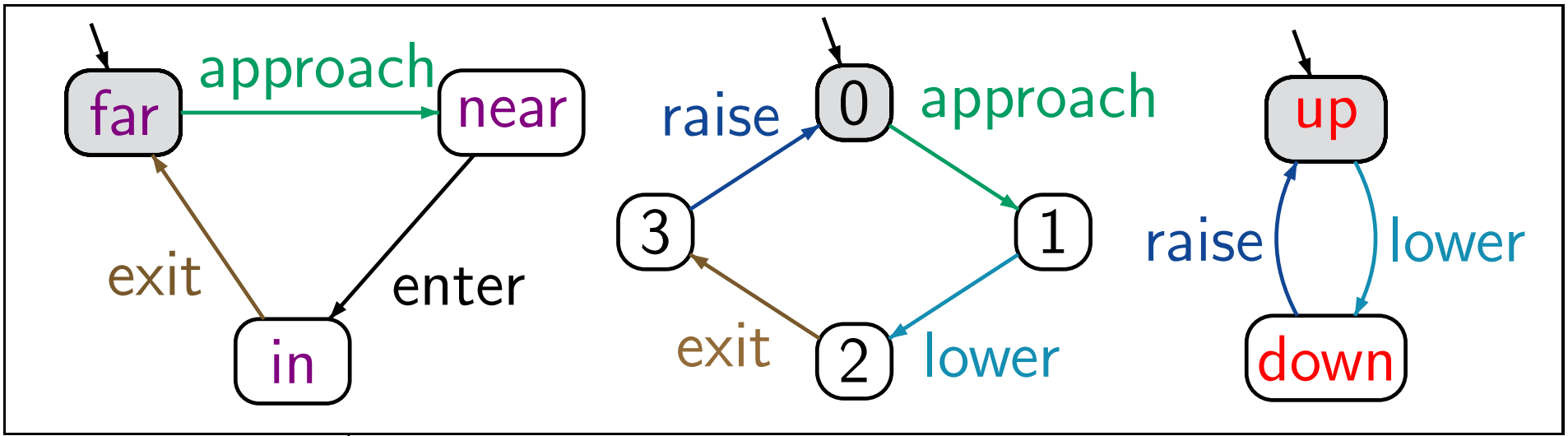


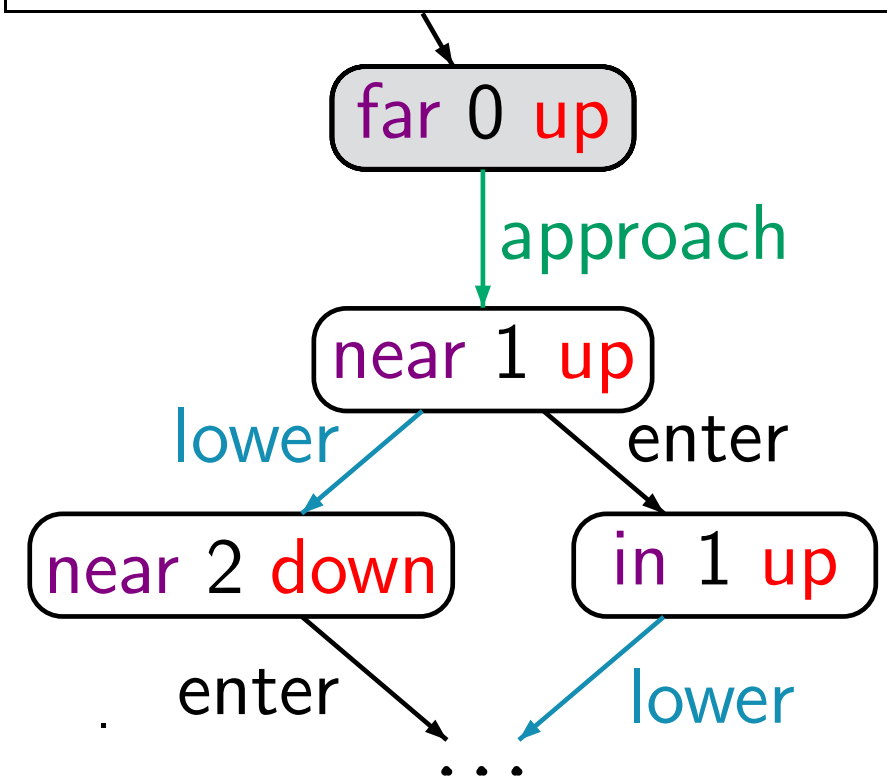


reachable fragment
of the transition system

Train ‖ ***Controller*** ‖ ***Gate***

TS for railroad crossing

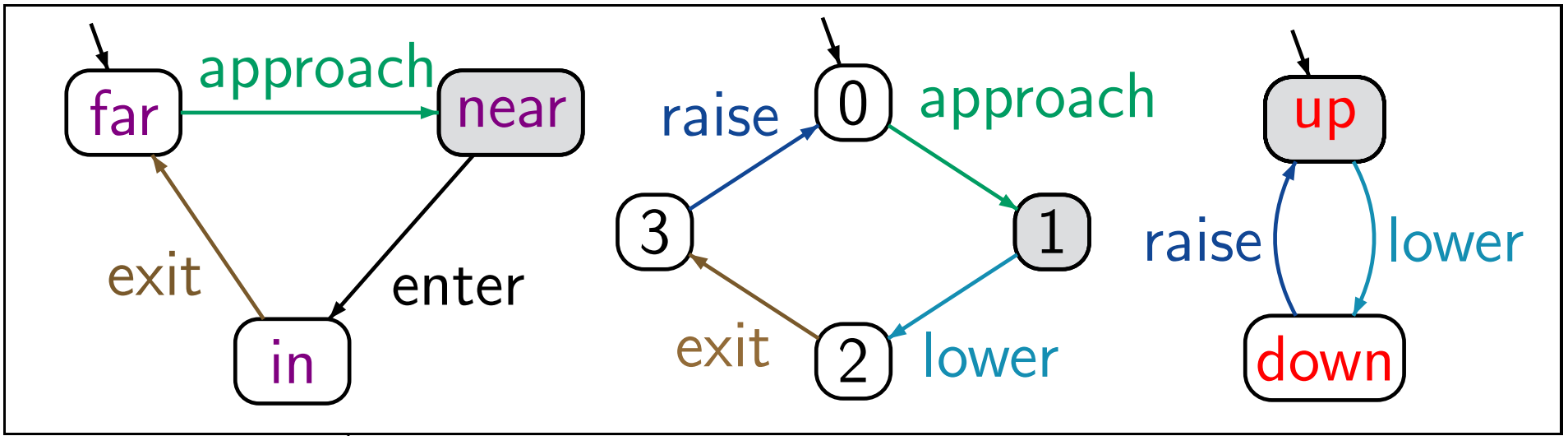


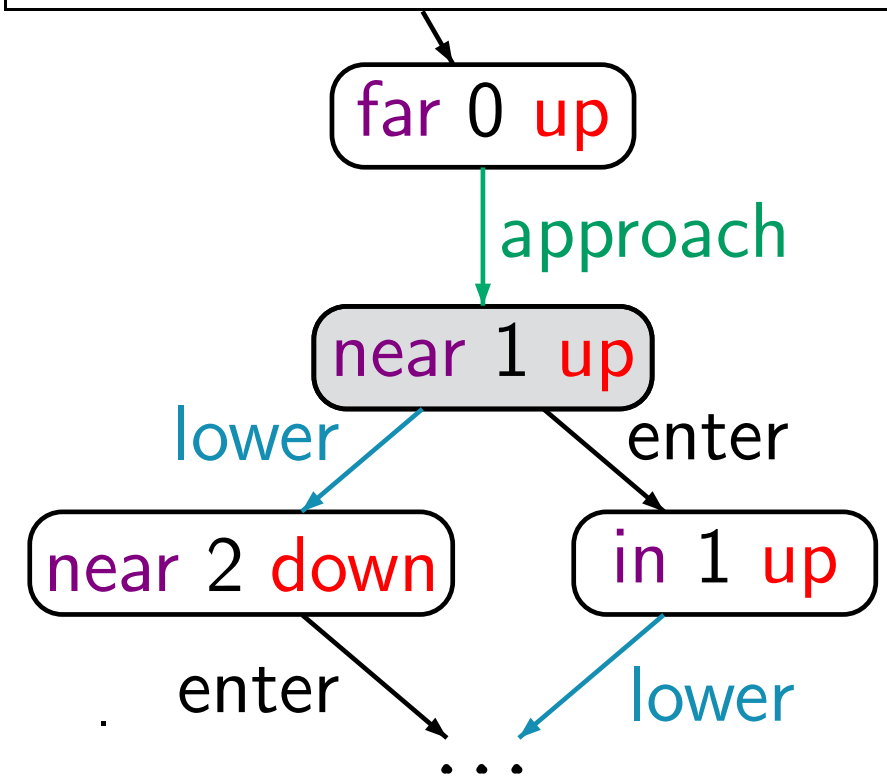


reachable fragment
of the transition system

Train ‖ ***Controller*** ‖ ***Gate***

TS for railroad crossing

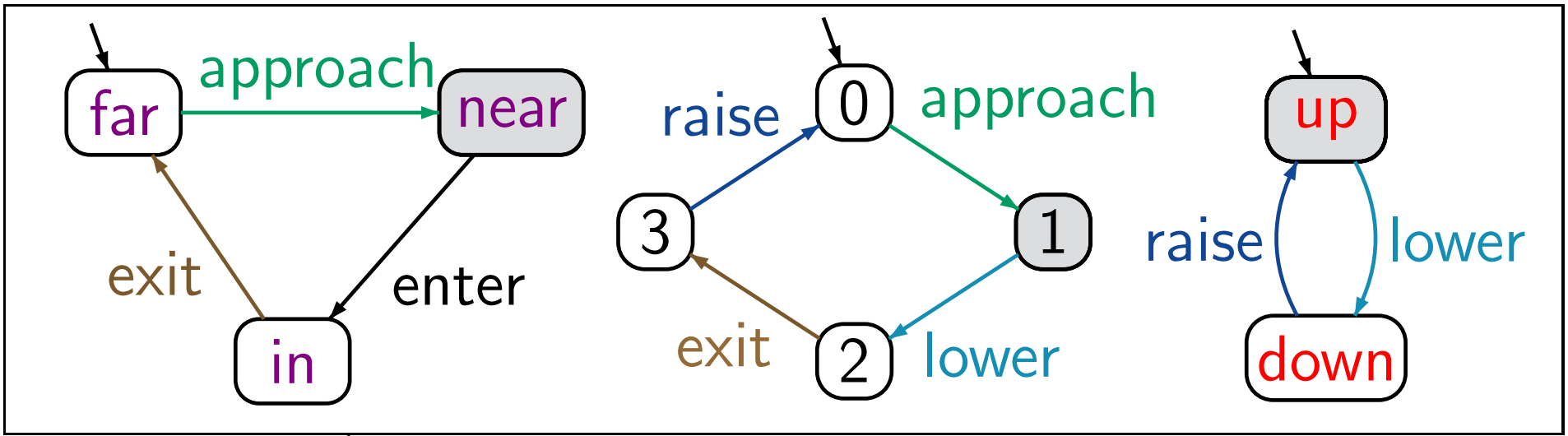


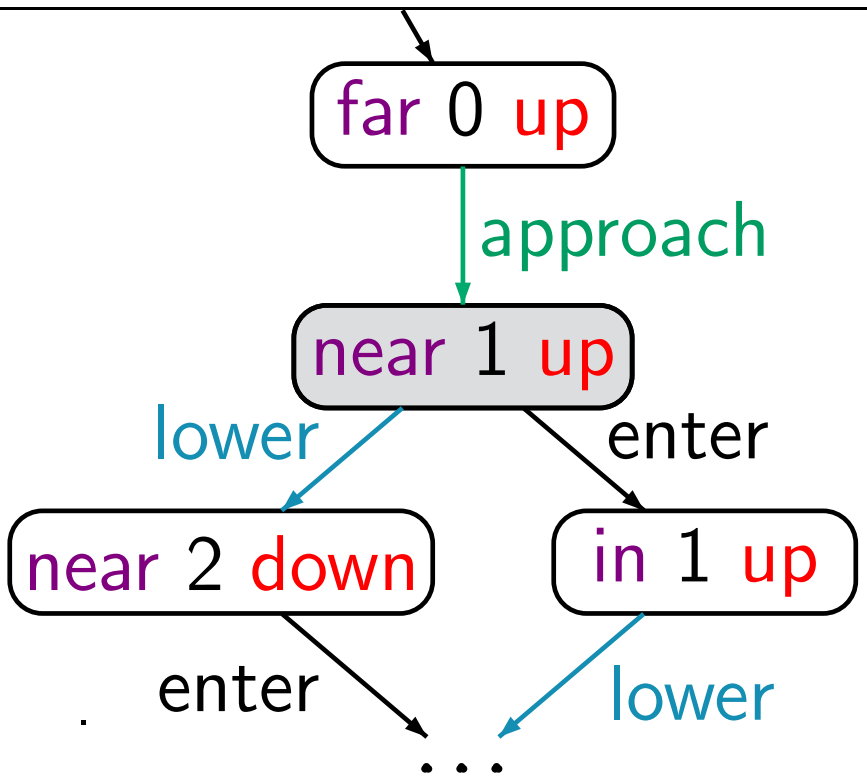


reachable fragment
of the transition system

Train ‖ ***Controller*** ‖ ***Gate***

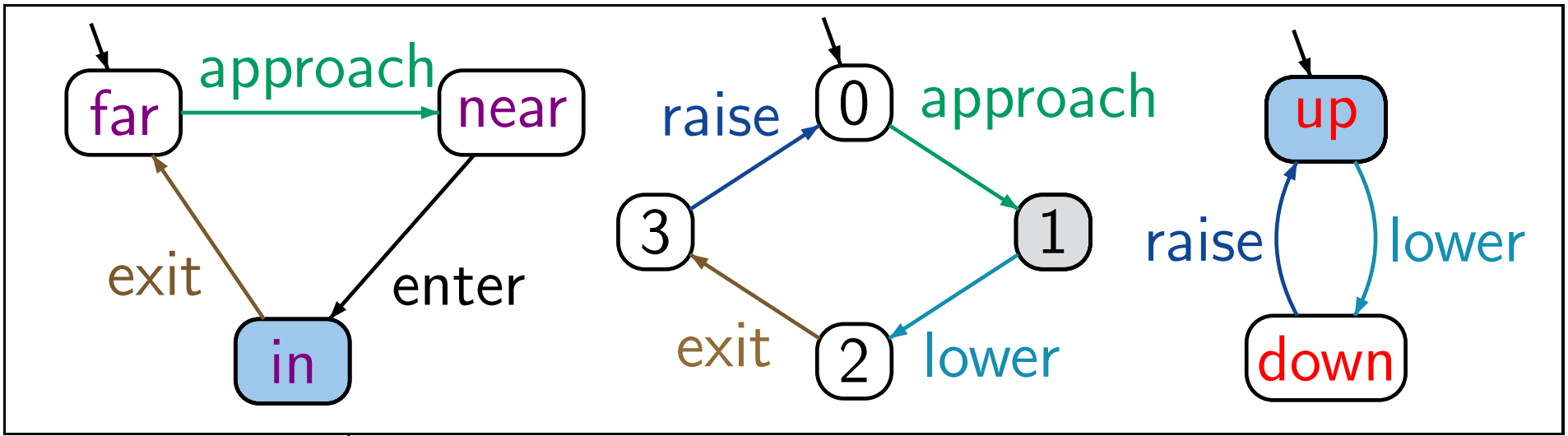
TS for railroad crossing

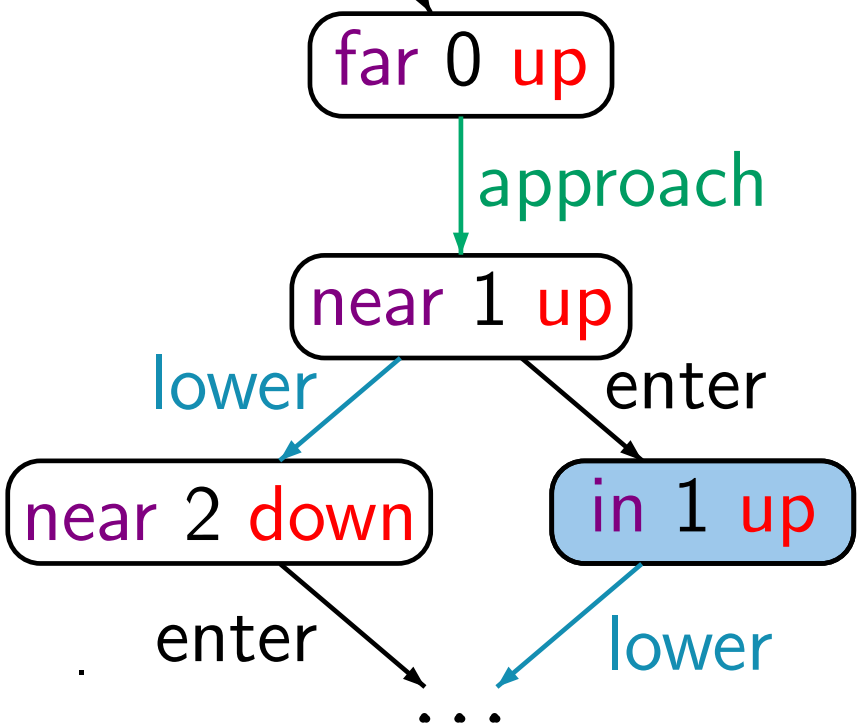




race between train and gate

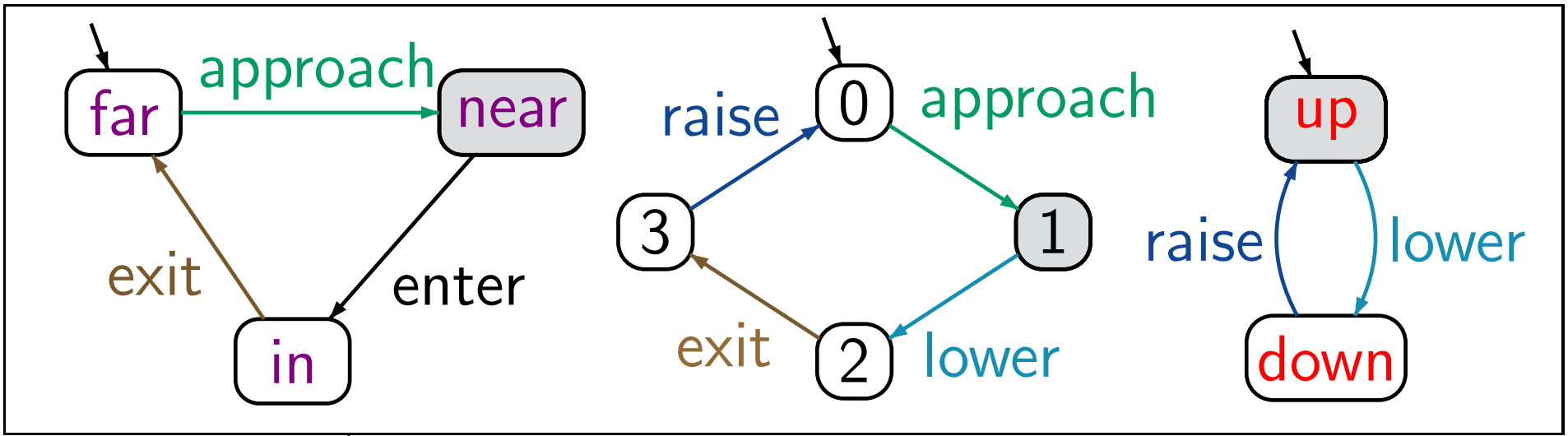
TS for railroad crossing

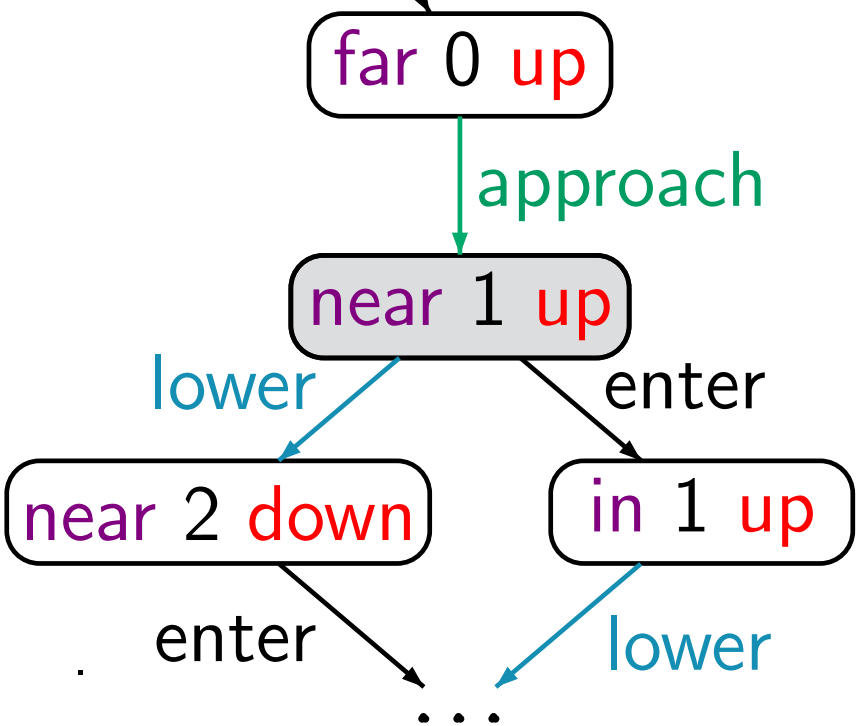




gate is open, while train is crossing the road

TS for railroad crossing





interleaving is time-abstract