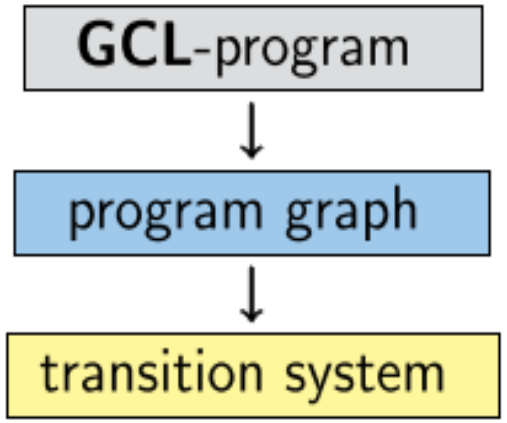
به نام خدا

روش‌های رسمی در مهندسی نرم‌افزار

مرجان عظیمی
Azimi.marjan@gmail.com

Guarded Command Language(GCL)

- توسط دایکسترا معرفی شده است.
- یک زبان مدلسازی تقریبا سطح بالاست.



Guarded Command Language(GCL)

guarded command $g \Rightarrow$ ***stmt*** ← enabled if g is true

g	: guard, i.e., Boolean condition on the program variables
stmt	: statement

Guarded Command Language(GCL)

guarded command $g \Rightarrow$ ***stmt*** ← enabled if g is true

g	: guard, i.e., Boolean condition on the program variables
stmt	: statement

repetitive command/loop:

DO :: $g \Rightarrow$ ***stmt*** OD ← WHILE g DO ***stmt*** OD

Guarded Command Language(GCL)

guarded command $g \Rightarrow$ ***stmt*** ← enabled if g is true

g	: guard, i.e., Boolean condition on the program variables
stmt	: statement

repetitive command/loop:

`DO` :: $g \Rightarrow$ ***stmt*** `OD` ← `WHILE` g `DO` ***stmt*** `OD`

conditional command:

```
IF :: g ⇒ stmt1
   :: ¬g ⇒ stmt2
FI
```

←

```
IF g THEN stmt1
     ELSE stmt2
FI
```

برنامه GCL برای ماشین نوشابه

uses two variables $\#sprite, \#coke \in \{0, 1, \ldots, max\}$
for the number of available drinks (sprite or coke)

برنامه GCL برای ماشین نوشابه

uses two variables $\#sprite, \#coke \in \{0, 1, \ldots, max\}$
for the number of available drinks (sprite or coke)

uses the following actions:

	enabled	effect
get_coke	if $\#coke > 0$	$\#coke := \#coke - 1$
get_sprite	if $\#sprite > 0$	$\#sprite := \#sprite - 1$
refill	any time	$\#sprite := max$ $\#coke := max$
insert_coin	any time	no effect on variables
return_coin	if machine is empty and user has entered a coin (no effect on variables)	

برنامه GCL برای ماشین نوشابه

```
DO  ::  true ⇒ insert_coin;
        IF  ::  #sprite = #coke = 0 ⇒ return_coin
            ::  #coke  > 0 ⇒ #coke := #coke − 1
            ::  #sprite > 0 ⇒ #sprite := #sprite−1
        FI
    ::  true ⇒ #sprite := max; #coke := max
OD
```

برنامه GCL برای ماشین نوشابه

```
DO  ::  true ⇒ insert_coin; (* user inserts a coin *)
        IF  ::  #sprite = #coke = 0 ⇒ return_coin
                                (* no beverage available *)
            ::  #coke  > 0 ⇒ #coke := #coke − 1
                                (* user selects coke *)
            ::  #sprite > 0 ⇒ #sprite := #sprite−1
                                (* user selects sprite *)
        FI
    ::  true ⇒ #sprite := max; #coke := max
                                (* refilling of the machine *)
OD
```

برنامه GCL برای ماشین نوشابه

```
DO  ::  true ⇒ insert_coin;
        IF  ::  #sprite = #coke = 0 ⇒ return_coin
            ::  #coke  > 0 ⇒ get_coke
            ::  #sprite > 0 ⇒ get_sprite
        FI
    ::  true ⇒ refill
OD
```

برنامه GCL برای ماشین نوشابه

```
DO  ::  true ⇒ insert_coin;
        IF  ::  #sprite = #coke = 0
                        ⇒ return_coin
            ::  #coke  > 0 ⇒ get_coke
            ::  #sprite > 0 ⇒ get_sprite
        FI
    ::  true ⇒ refill
OD
```

... yields a program graph with

- two variables $\#sprite, \#coke \in \{0, 1, \ldots, max\}$

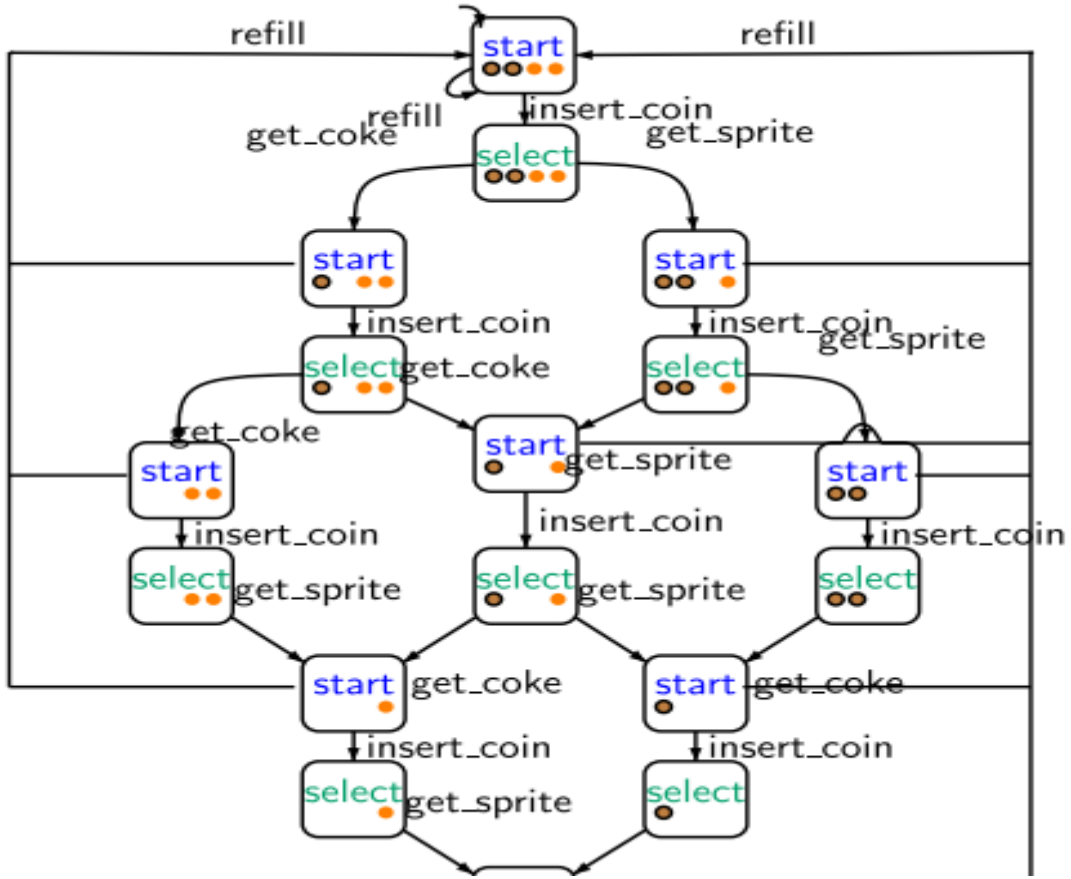
برنامه GCL برای ماشین نوشابه

```
start → DO  ::  true ⇒ insert_coin;
select→         IF  ::  #sprite = #coke = 0
                            ⇒ return_coin
                    ::  #coke  > 0 ⇒ get_coke
                    ::  #sprite > 0 ⇒ get_sprite
                FI
            ::  true ⇒ refill
        OD
```

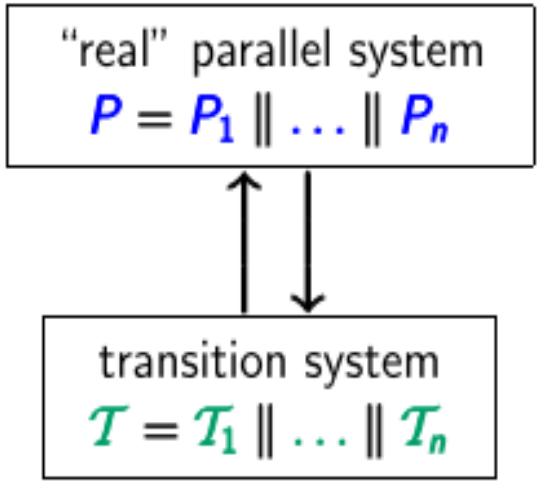
... yields a program graph with

- two variables $\#sprite, \#coke \in \{0, 1, \ldots, max\}$
- two locations *start* and *select*

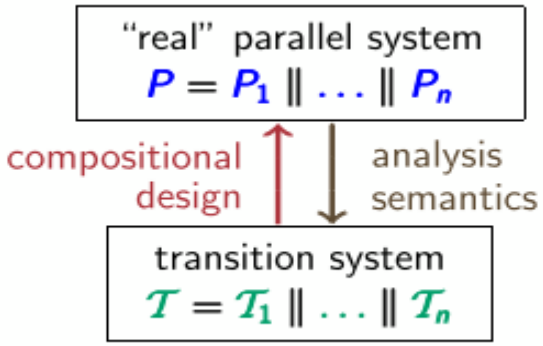
برنامه GCL برای ماشین نوشابه



موازی‌سازی و همروندی



عملگر interleaving برای سیستم گذار



هدف: مدل‌سازی اجرای موازی فرآیندها با TS یا PG است.

عملگر interleaving برای سیستم گذار

- کارهای غیر وابسته در سیستم گذار با interleaving پیاده‌سازی می‌شوند.
- Interleaving توسط انتخاب غیر قطعی مدل‌سازی می‌شود.

$$effect(\alpha|||\beta) = effect(\alpha;\beta+\beta;\alpha)$$

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

عملگر interleaving برای سیستم گذار

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

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

عملگر interleaving برای سیستم گذار

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

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

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:

عملگر interleaving برای سیستم گذار

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

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

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:

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

عملگر interleaving برای سیستم گذار

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

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

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

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

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

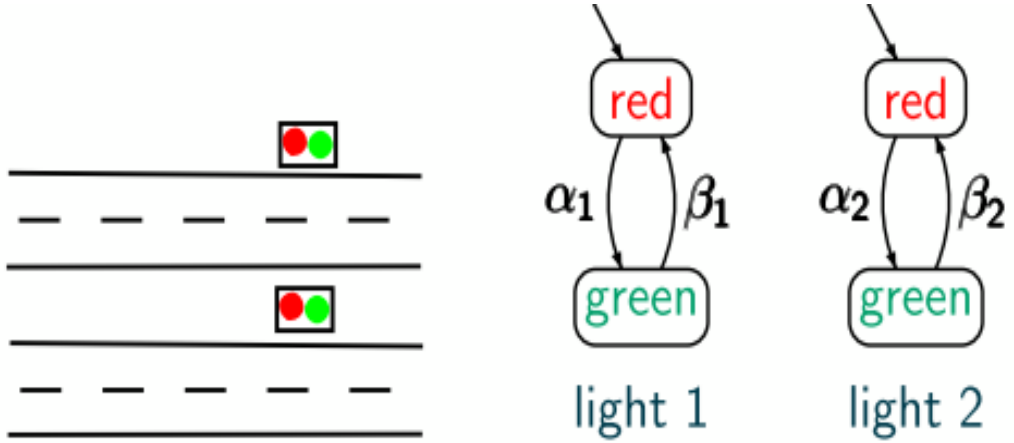
$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2\rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1, s_2'\rangle}$$

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

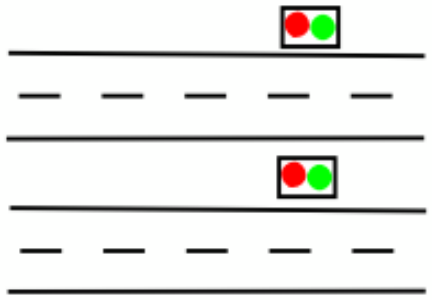
(1) If $s_1 \xrightarrow{\alpha}_1 s_1'$, then $\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2\rangle$

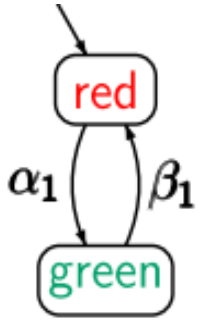
(2) If $s_2 \xrightarrow{\alpha}_2 s_2'$, then $\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1, s_2'\rangle$

مثال چراغ راهنما برای خیابان غیر متقاطع

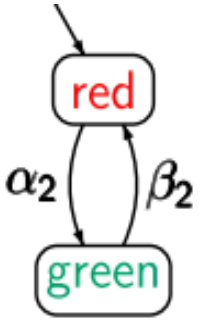


مثال چراغ راهنما برای خیابان غیر متقاطع



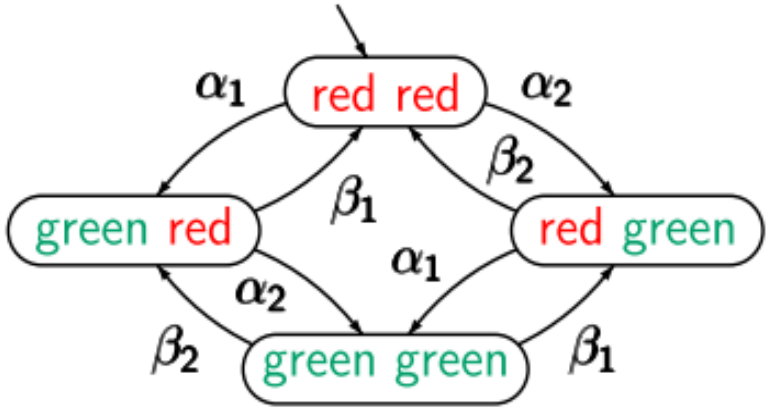


light 1



light 2

light 1 ||| light 2



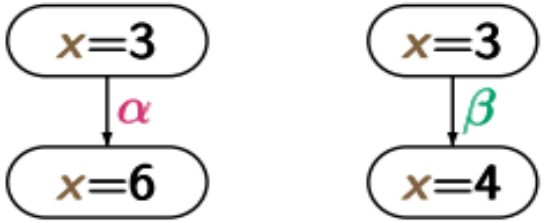
فرآیندهای وابسته

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

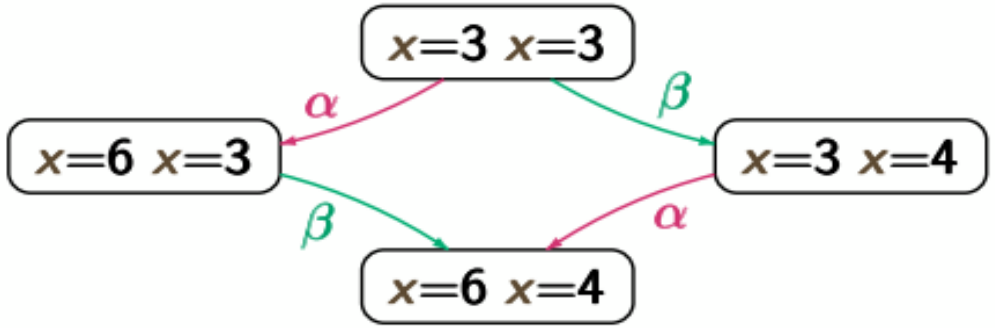
representations in transition systems

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

representations in transition systems

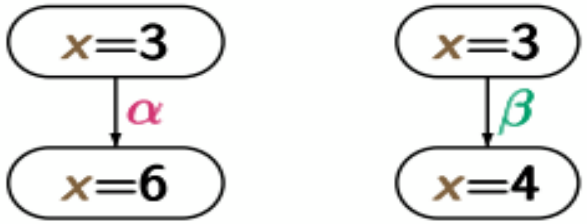


interleaving operator |||

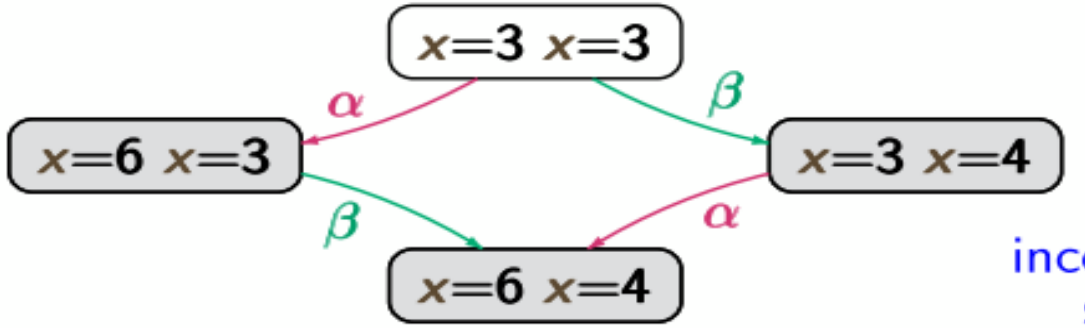


dependent actions $\alpha \mathrel{\widehat{=}} x{:=}2x$ and $\beta \mathrel{\widehat{=}} x{:=}x{+}1$

representations in transition systems



interleaving operator ||| for transition systems "fails"



inconsistent states

Interleaving for program graph

- برای مدلسازی سیستم‌های موازی که زیر برنامه‌هایش از متغیر مشترک استفاده می‌کنند،استفاده می‌شود.

Interleaving for program graph

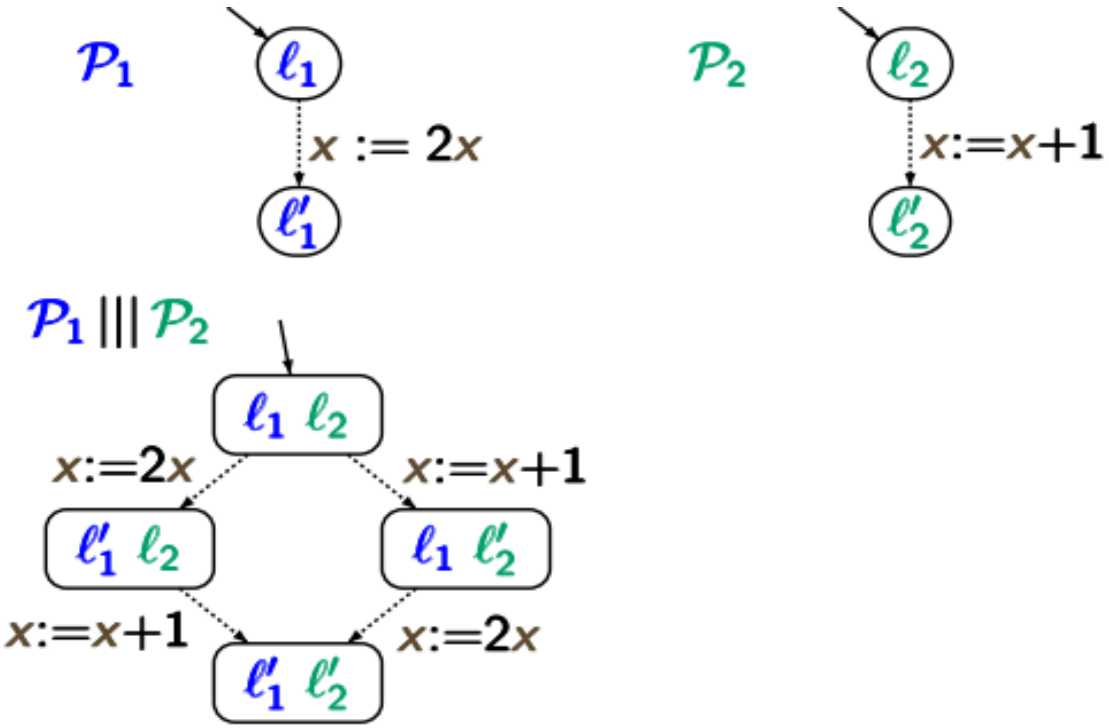
program graph $\mathcal{P}_1$	program graph $\mathcal{P}_2$
$(Loc_1, \ldots, \hookrightarrow_1, \ldots)$	$(Loc_2, \ldots, \hookrightarrow_2, \ldots)$

interleaving operator

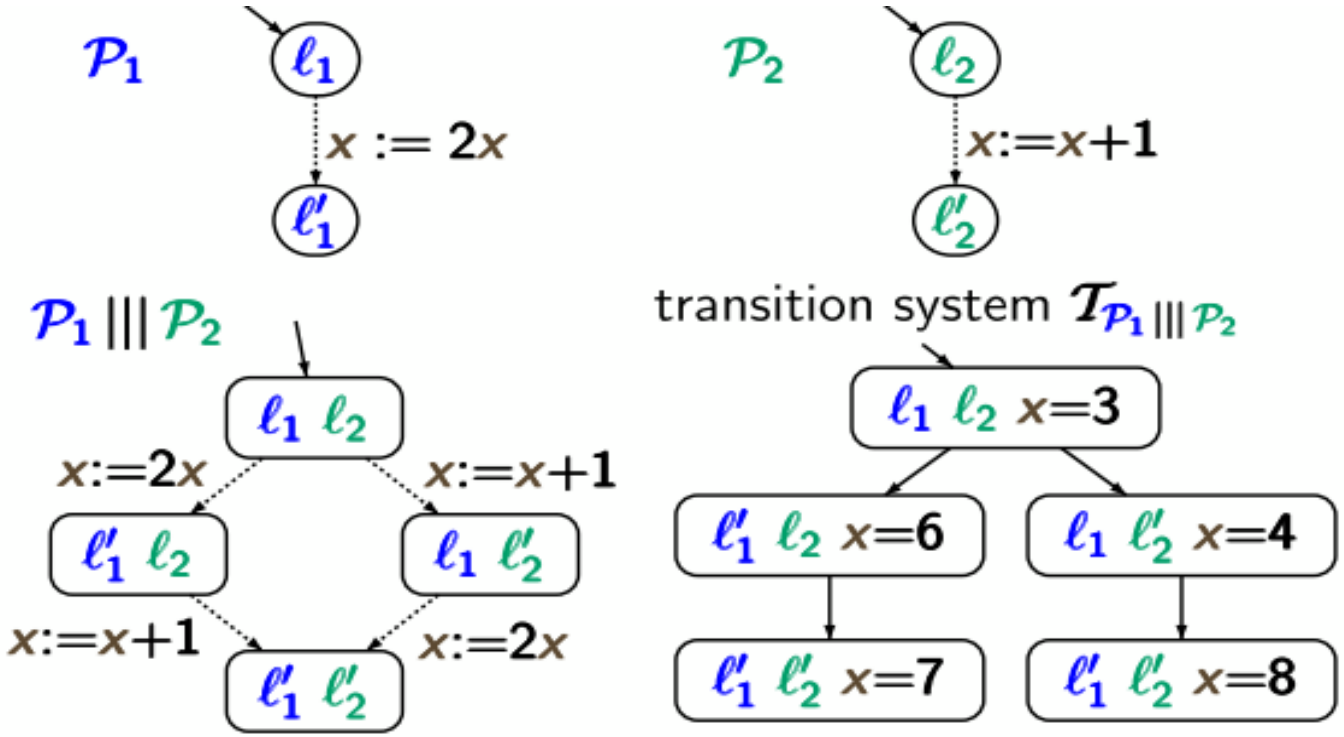
$$\mathcal{P}_1 \,|||\, \mathcal{P}_2 \;=\; (Loc_1 \times Loc_2, \ldots, \hookrightarrow, \ldots)$$

$$\frac{\ell_1 \overset{g:\alpha}{\hookrightarrow}_1 \ell_1'}{\langle \ell_1, \ell_2 \rangle \overset{g:\alpha}{\hookrightarrow} \langle \ell_1', \ell_2 \rangle} \qquad \frac{\ell_2 \overset{g:\alpha}{\hookrightarrow}_2 \ell_2'}{\langle \ell_1, \ell_2 \rangle \overset{g:\alpha}{\hookrightarrow} \langle \ell_1, \ell_2' \rangle}$$

Interleaving for program graph

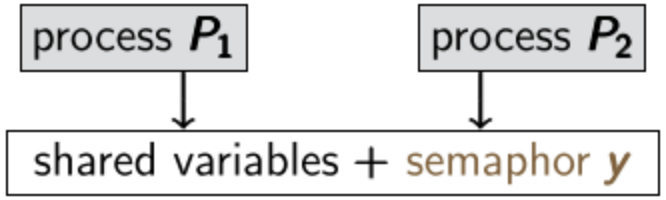


Interleaving for program graph

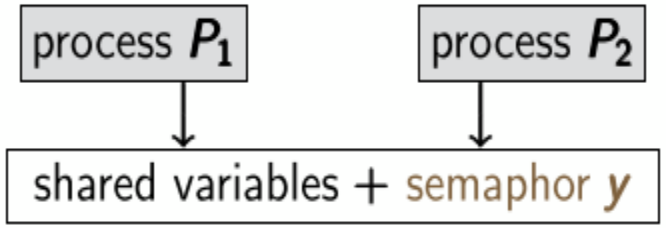


note: $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2} \neq \mathcal{T}_{\mathcal{P}_1} ||| \mathcal{T}_{\mathcal{P}_2}$

Mutual exclusion with semaphore



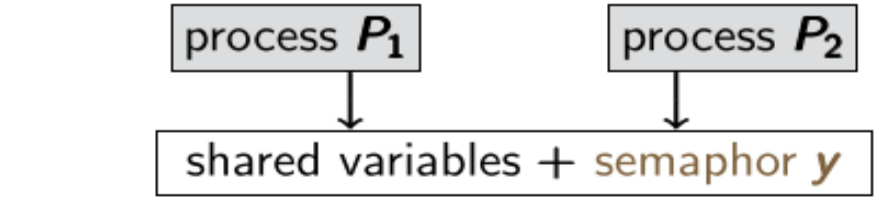
Mutual exclusion with semaphore



protocol for process P_i

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

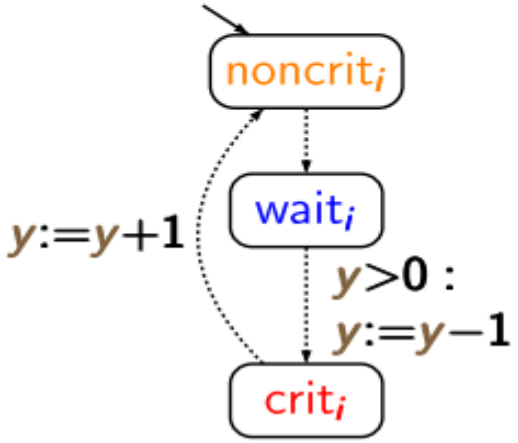
Mutual exclusion with semaphore



protocol for process P_i

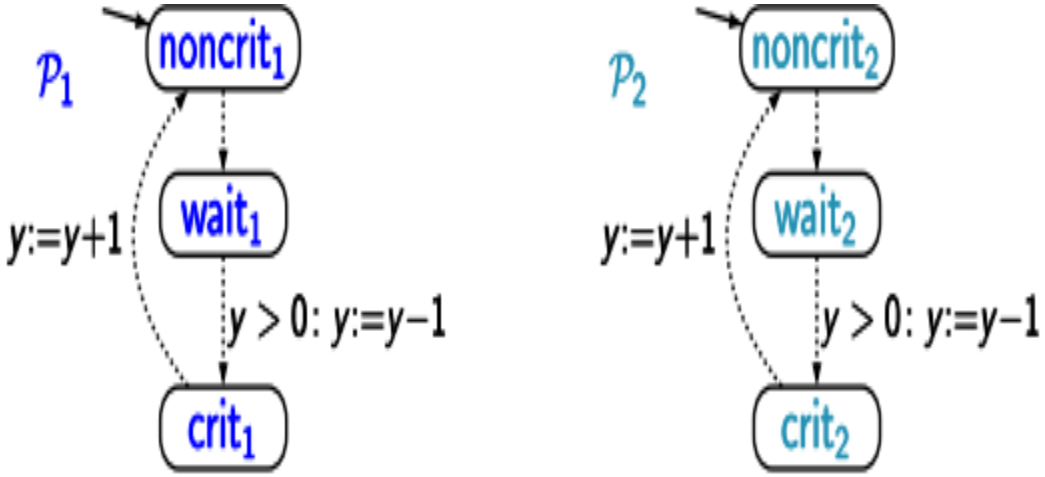
```
LOOP FOREVER
  noncritical actions;
  AWAIT y > 0 DO
        y:=y−1
  OD
  critical actions;
  y:=y+1
END LOOP
```

program graph $\mathcal{P}_i$

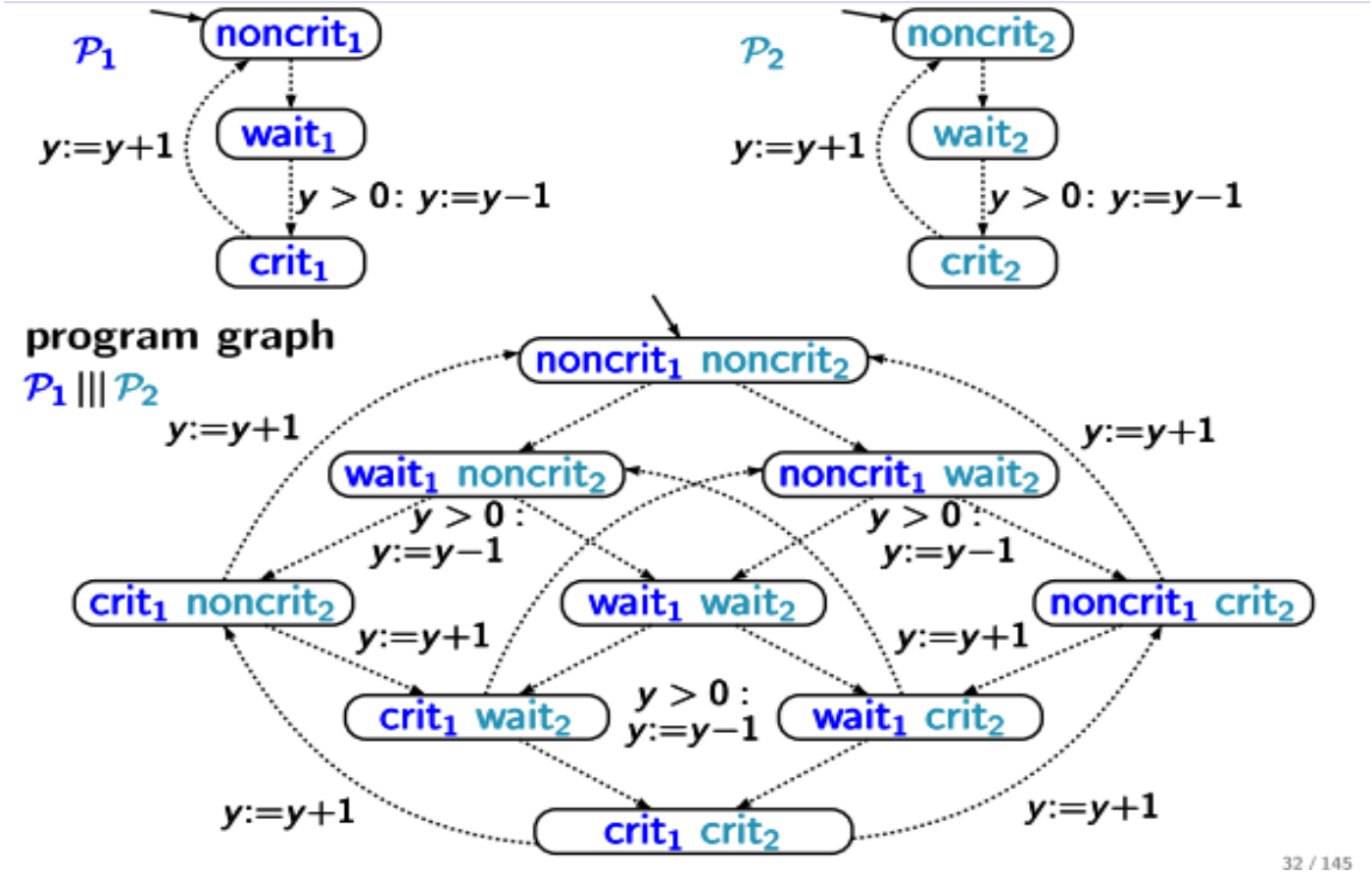


30 / 145

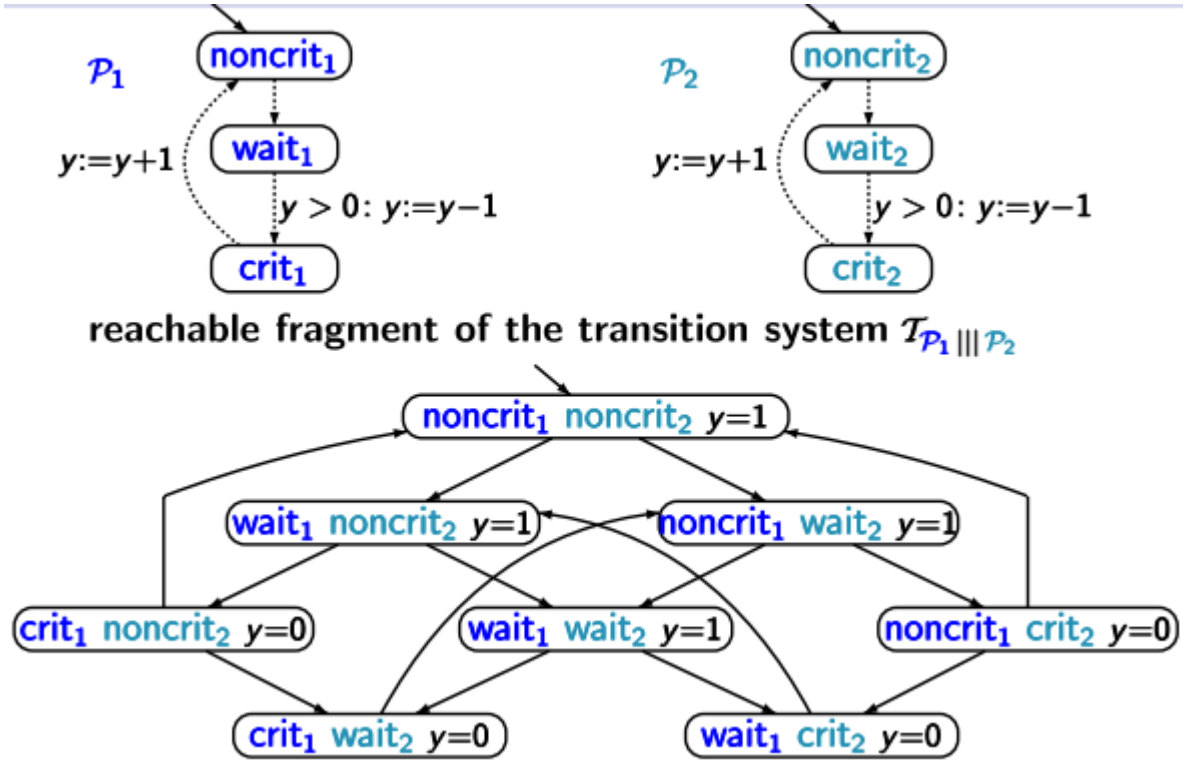
Mutual exclusion with semaphore



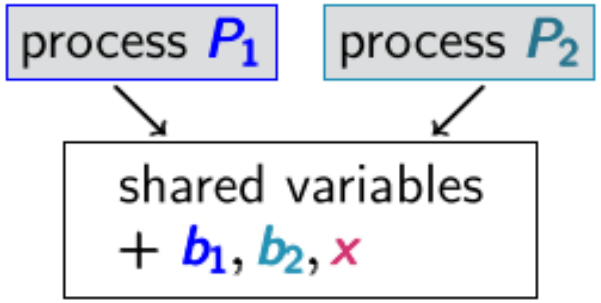
Mutual exclusion with semaphore



TS for Mutual Exclusion with Semaphore

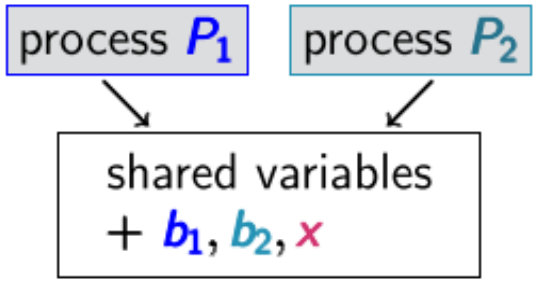


Peterson Algorithm for mutual exclusion



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

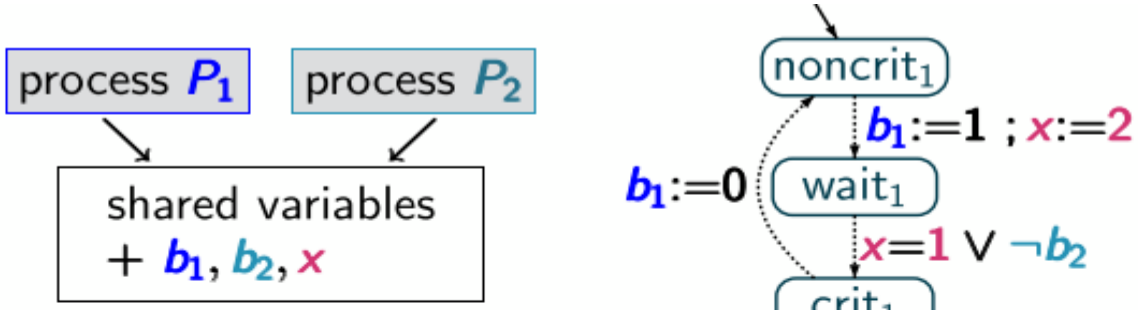
Peterson Algorithm for mutual exclusion



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
```

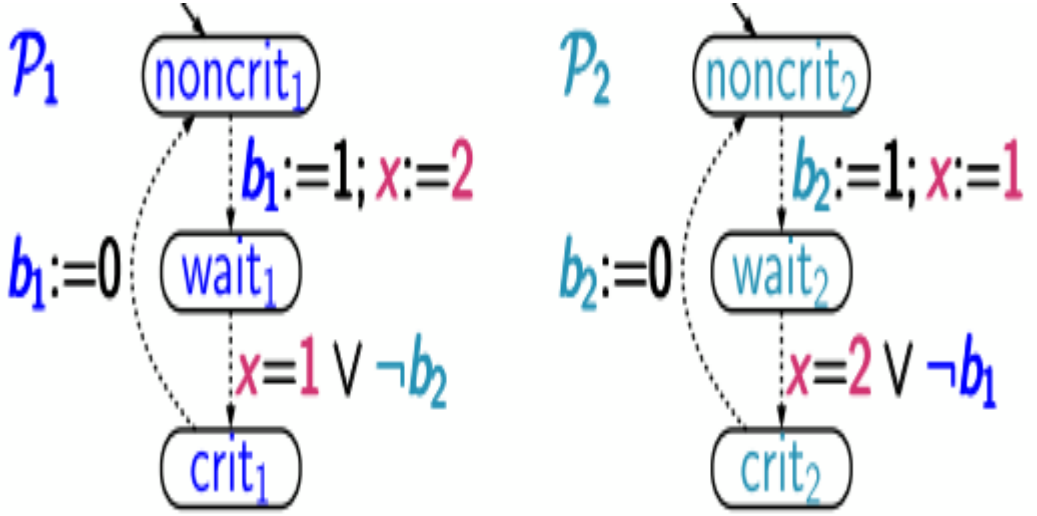
Peterson Algorithm for mutual exclusion



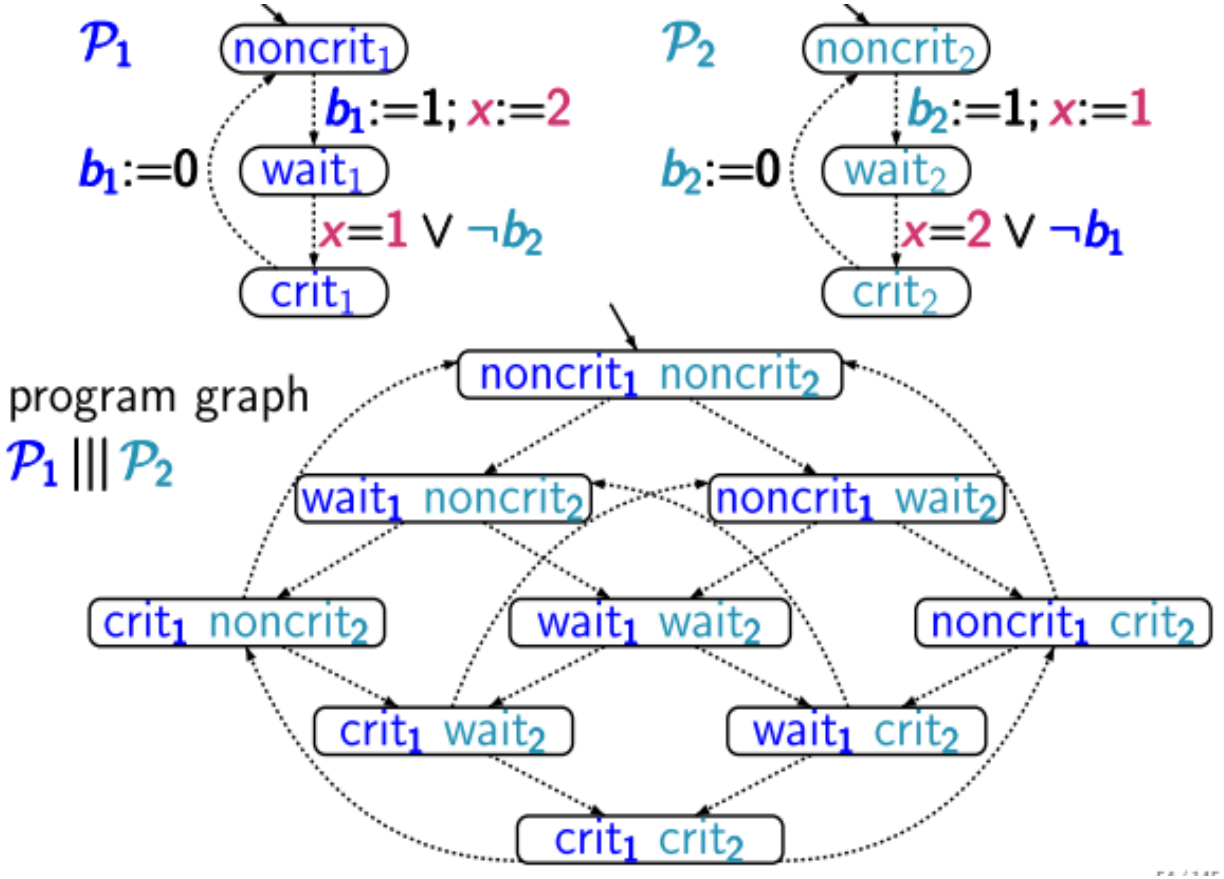
b_1, b_2 Boolean variables, $x \in \{1, 2\}$

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

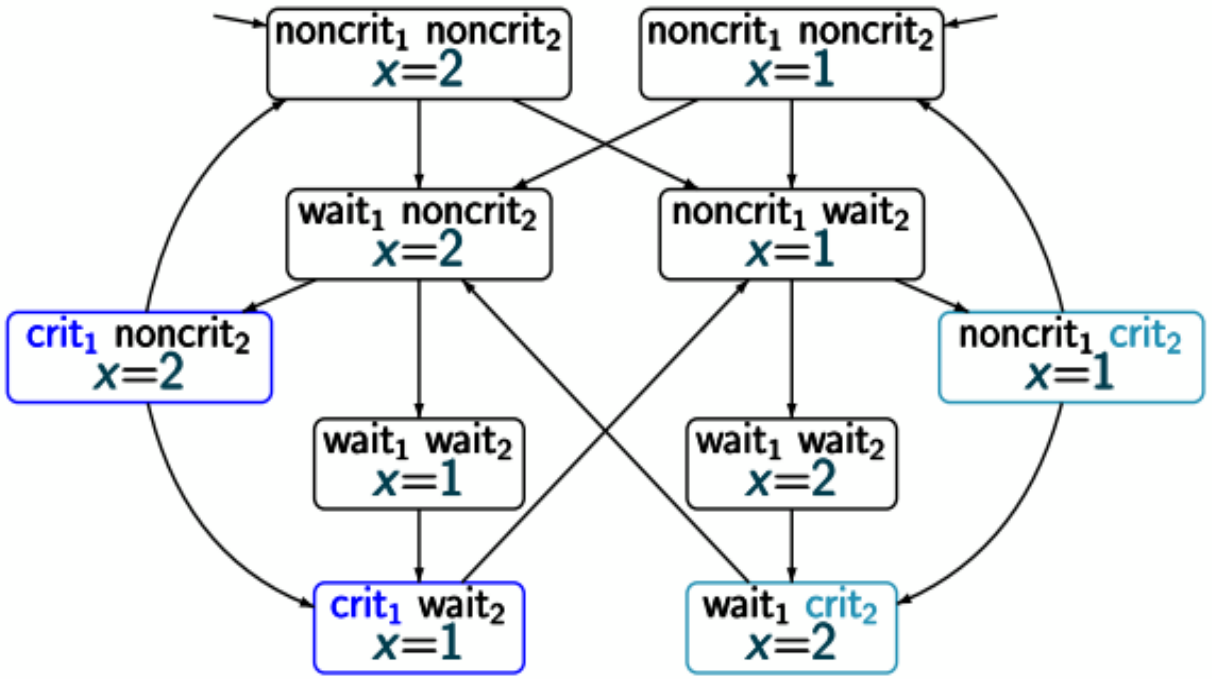
Program Graph for Peterson Algorithm



Program Graph for Peterson Algorithm



TS for the Peterson Algorithm



Handshaking

- مکانیزمی که فرآیندهای همروند از طریق آن با یکدیگر تعامل دارند.
- فرآیندهایی که می‌خواهند با هم تعامل داشته باشند، باید در مد همگام‌سازی باشند.
- یک مجموعه عملگر H تحت عنوان handshake action در نظر گرفته می‌شود. تمام عملگرهای خارج از H مستقل از یکدیگر می‌باشند.

Handshaking(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 $\alpha \in Syn$:

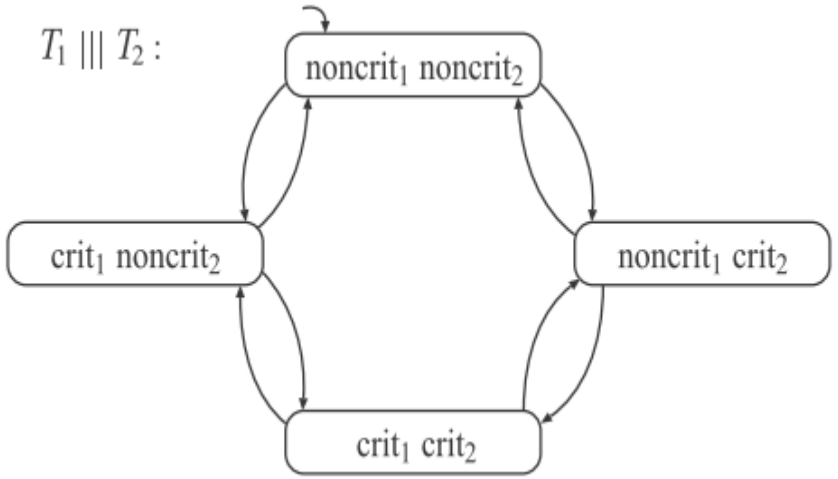
$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \quad \wedge \quad s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2'\rangle}$$

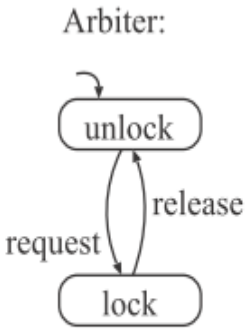
ناحیه انحصاری با عبور پیام همزمان

protocol for process P_i

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

ناحیه انحصاری با عبور پیام همزمان





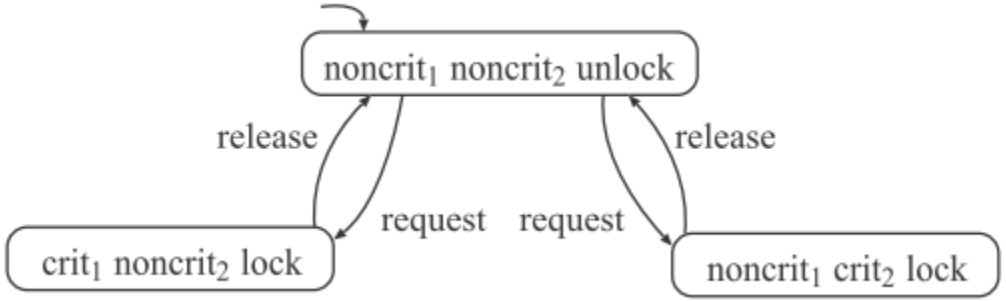
ناحیه انحصاری با عبور پیام همزمان

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

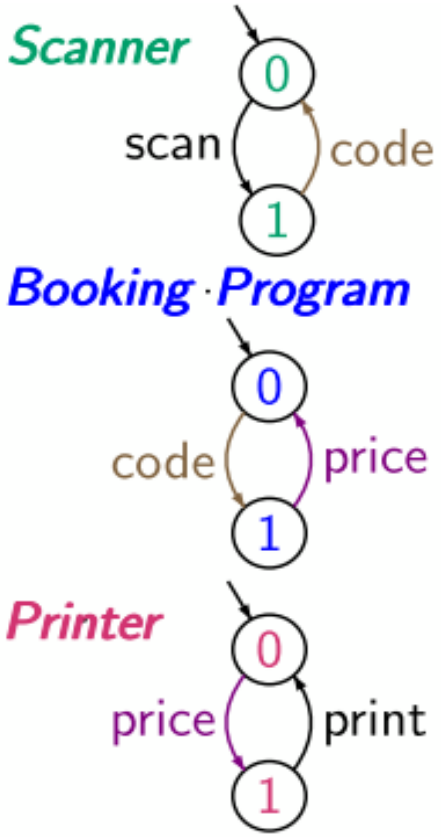
"pure" interleaving for TS

handshaking for actions request and release

ناحیه انحصاری با عبور پیام همزمان

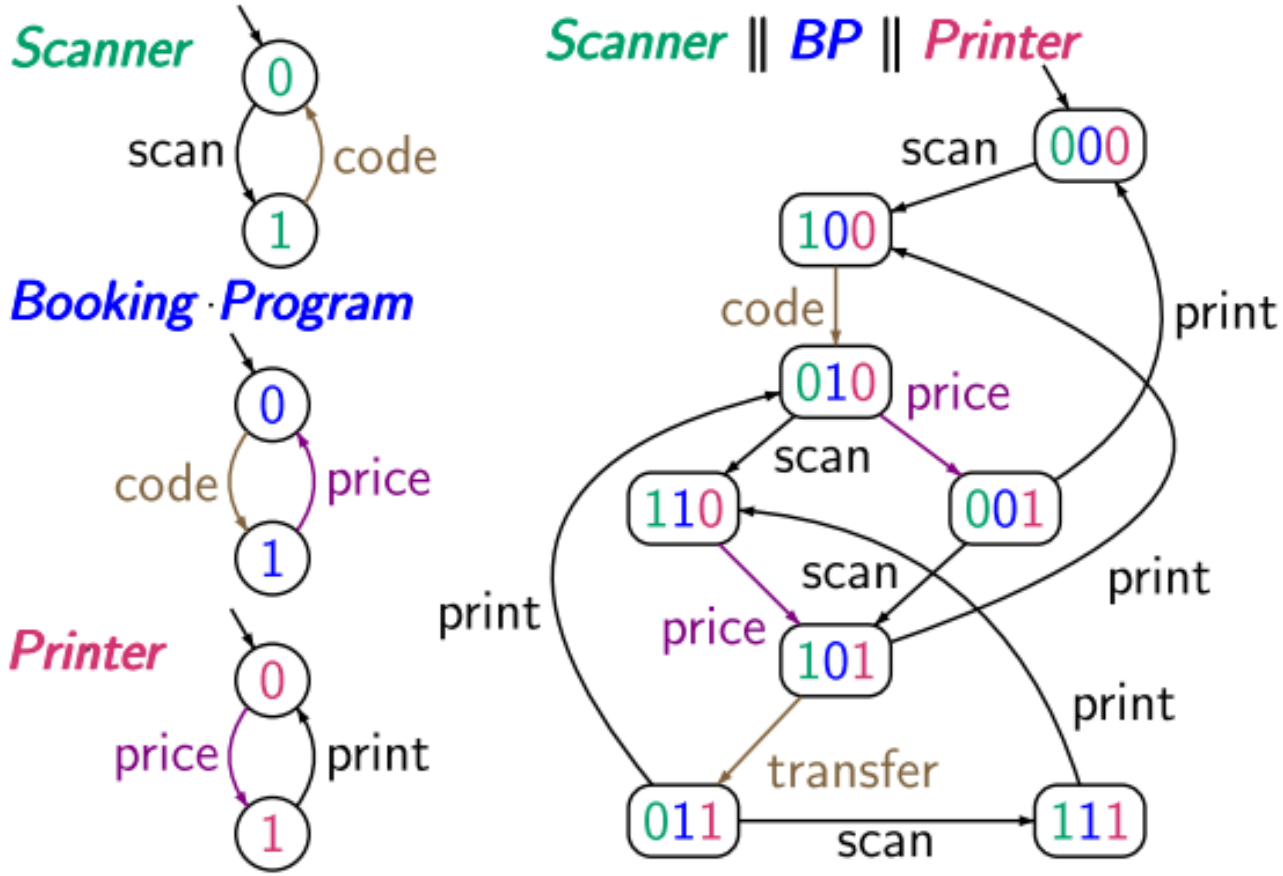


سیستم ثبت در سوپرمارکت

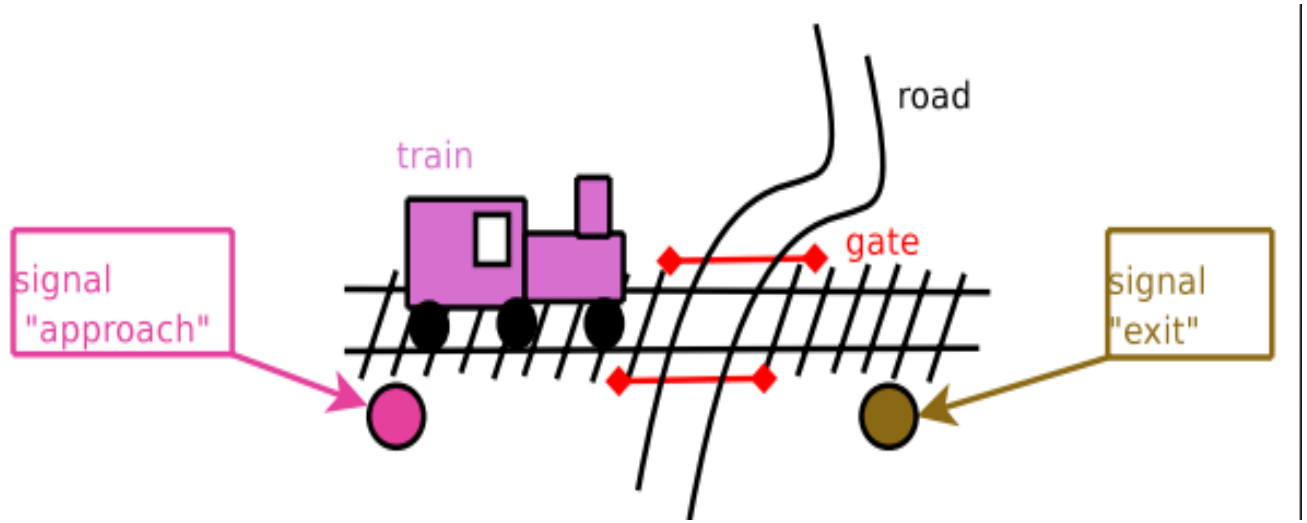


Scanner || BP || Printer

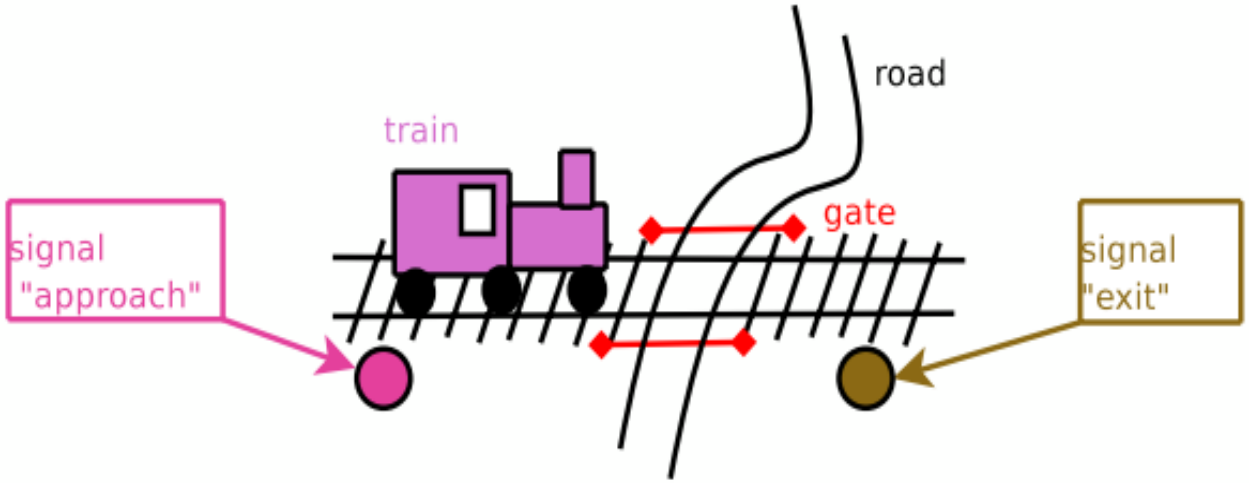
سیستم ثبت در سوپرمارکت



تقاطع ریل قطار



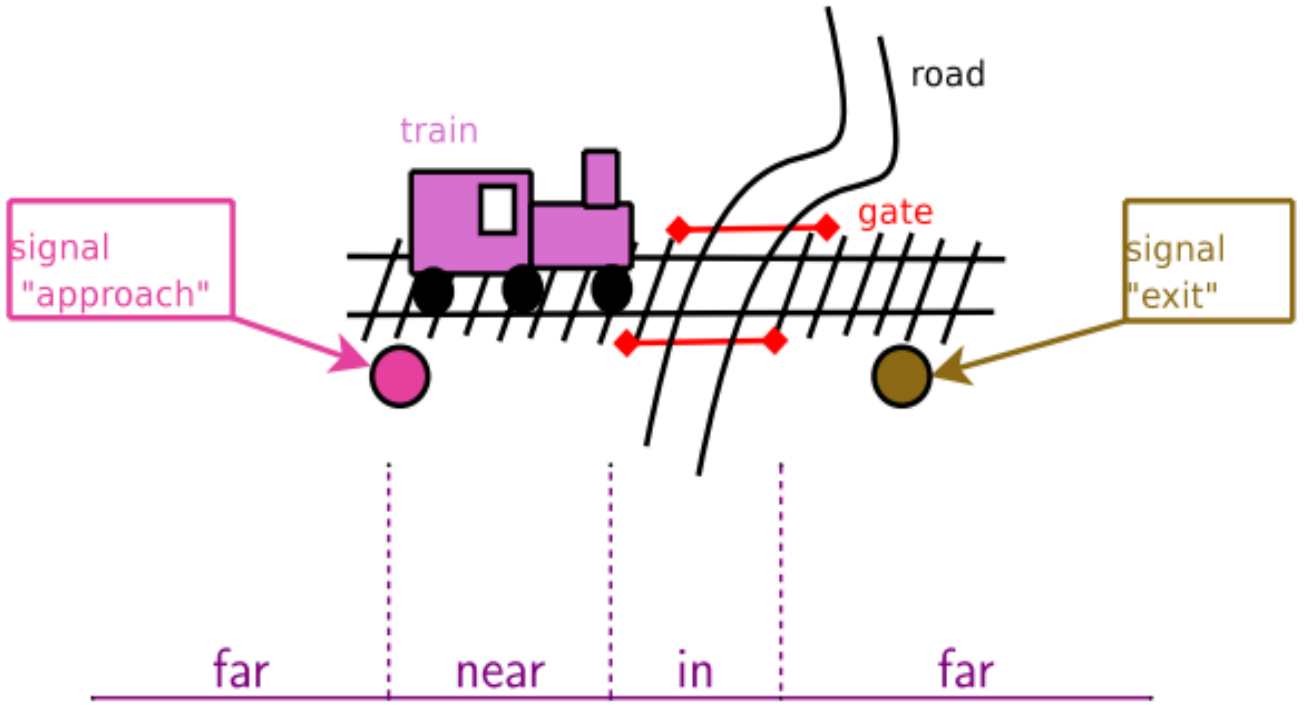
تقاطع ریل قطار



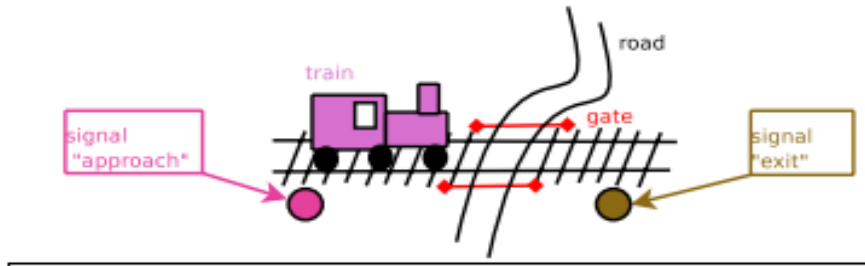
modeling by a transition system with **3** processes:

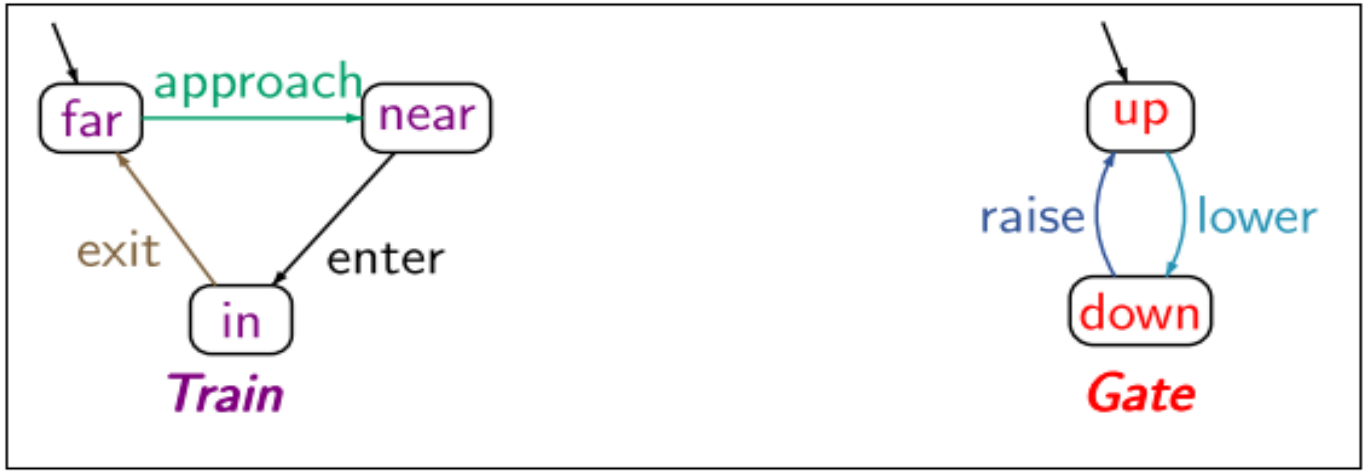
Train || *Controller* || *Gate*

تقاطع ریل قطار

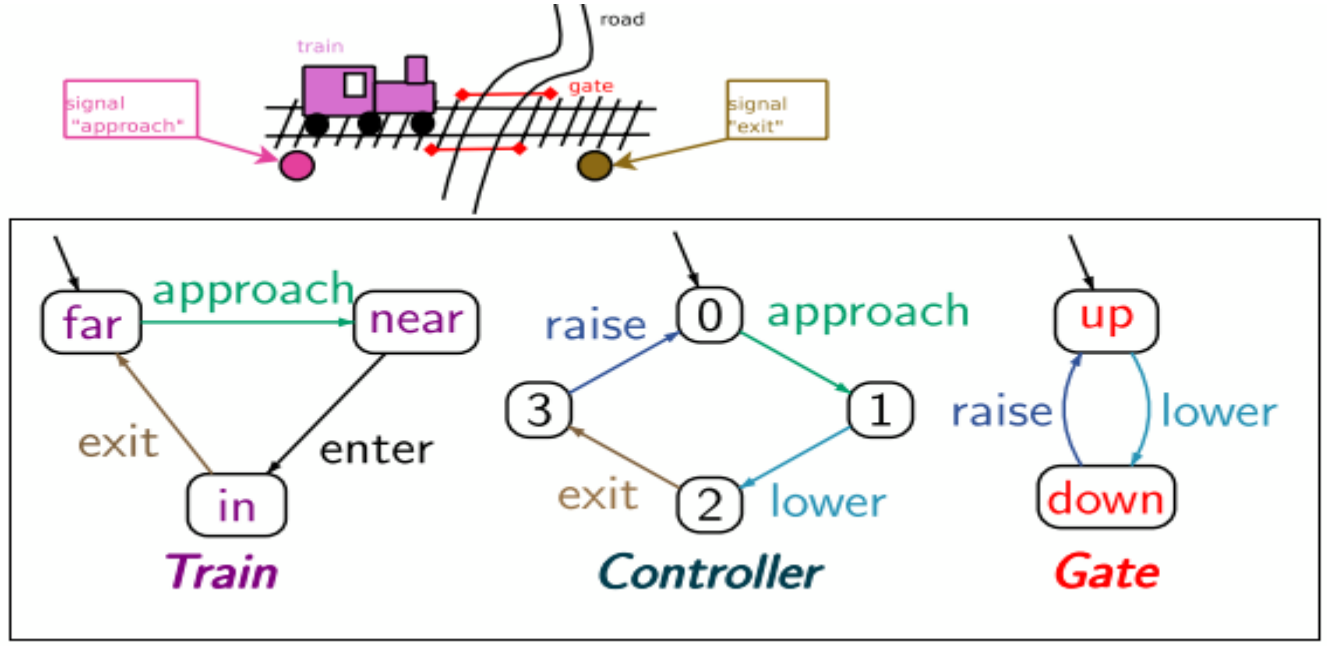


تقاطع ریل قطار

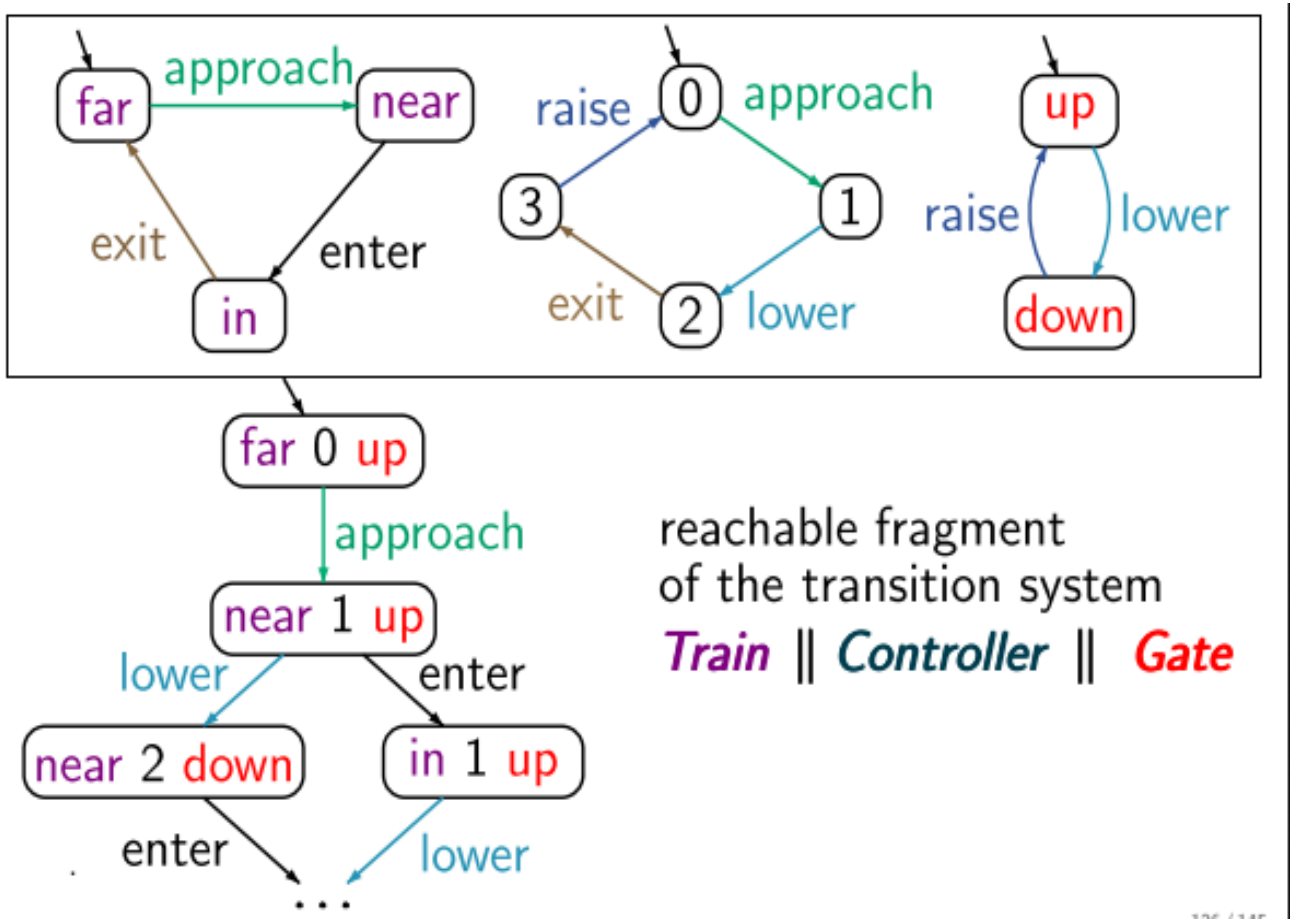




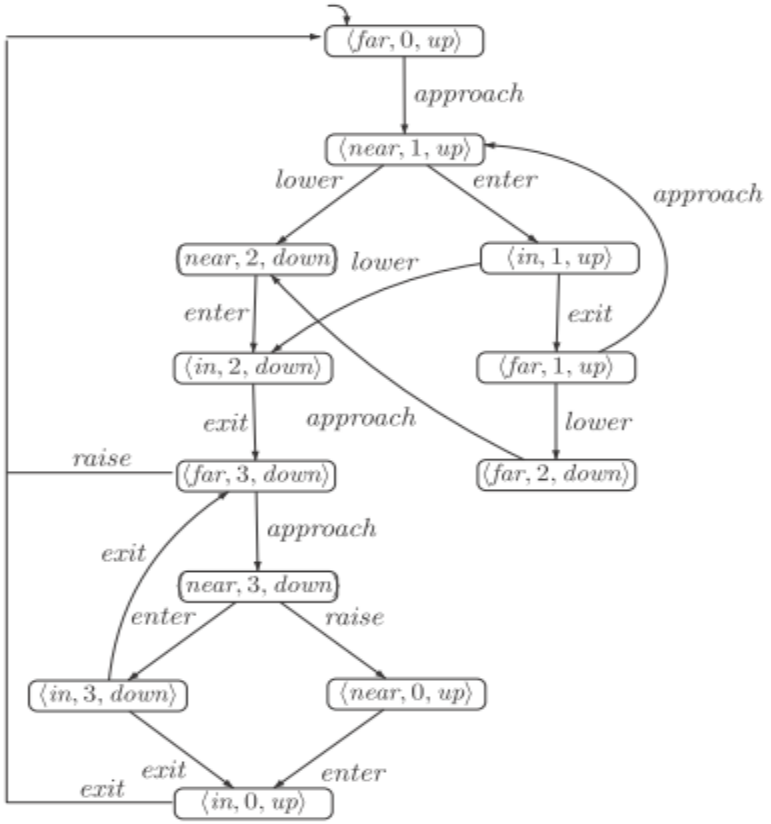
تقاطع ریل قطار



تقاطع ریل قطار



تقاطع ریل قطار



Channel Systems

- یک‌سری فرآیندهای موازی داریم که از طریق کانال با یکدیگر ارتباط دارند. می‌توان فرض کرد، کانال همان بافر است که شامل پیام است و از رویکرد FiFo برای پردازش پیام‌ها استفاده می‌کند.

Channel Systems

- نمایشی است برای سیستم‌های موازی که وابستگی داده‌ای دارند:
  - انتقال پیام همزمان(ظرفیت کانال صفر)
  - انتقال پیام غیر همزمان (ظرفیت کانال مخالف صفر)

- ظرفیت برابر است با تعداد خانه‌های بافر.

Channel system

- اگر بخواهیم این مدل را به صورت فرمال نمایش دهیم:
- فرض کنیم که n فرآیند $p_1 \dots p_n$ داریم که هر فرآیند را با program graph نشان می‌دهیم.
- Action ها را به دو دسته action های مستقل و action های ارتباطی تقسیم می‌کنیم.

Communication action

- C!V: مقدار v را به کانال ارسال می‌کند.
- C?X: پیام از کانال c دریافت و در متغیر x قرار داده می‌شود.
- $communication\ action = \{c!\,v, c?\,x | c \in chan, v \in dom(c), x \in var\ with\ dom(x) \supseteq dom(c)\}$

Channel system

- $cap(c) \varepsilon N \cup \{\infty\}$
- Dom(c)

هر کانال با دو مشخصه نمایش داده می‌شود: ظرفیت کانال و دامنه کانال که ظرفیت کانال می‌تواند متناهی و یا نامتناهی باشد. دامنه کانال نیز مشخص می‌کند که چه چیزهایی می‌تواند در کانال قرار بگیرند.

Channel system(cs)

$[\mathcal{P}_1 \mid \mathcal{P}_2 \mid \ldots \mid \mathcal{P}_n]$ where $\mathcal{P}_i$ are program graphs over a pair $(Var, Chan)$

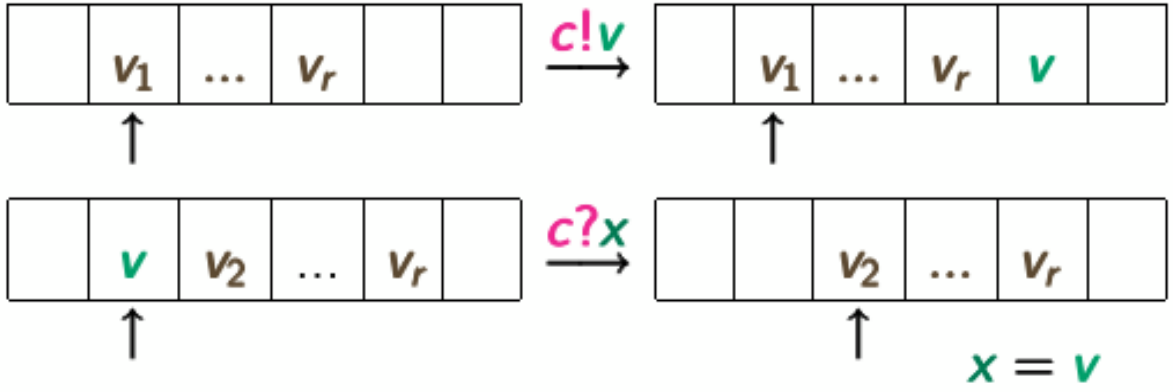
Var	set of typed variables
$Chan$	set of typed channels with capacities $cap(\cdot)$ and domains $Dom(\cdot)$

program graphs $\mathcal{P}_i = (Loc_i, Act_i, Effect_i, \hookrightarrow_i, Loc_{0,i}, g_0)$ with conditional transitions

- $\ell \xrightarrow{g:\alpha}_i \ell'$ guarded command
- $\ell \xrightarrow{c!v}_i \ell'$ sending value v via channel c
- $\ell \xrightarrow{c?x}_i \ell'$ receiving a value for variable x via channel c

تاثیر communication action در channel system

	enabled if ...	effect
sending $c!v$	channel c not full	$add(c, v)$
receiving $c?x$	channel c not empty $v = front(c)$	$x := v$ $remove(c)$



TS of channel system

channel system over $(Var, Chan)$

$$\mathcal{C} = [\mathcal{P}_1 | \ldots | \mathcal{P}_n]$$

$\Downarrow$

transition system $\mathcal{T}_{\mathcal{C}}$

states of $\mathcal{T}_{\mathcal{C}}$ have the form

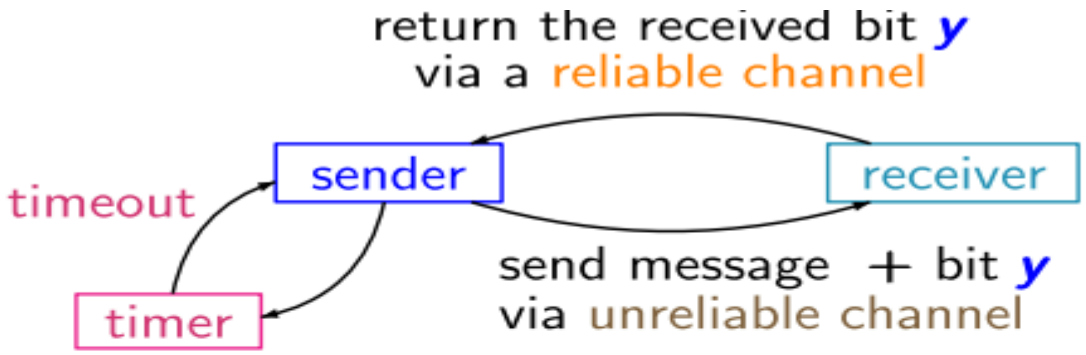
$$\langle \ell_1, \ldots, \ell_n, \eta, \xi \rangle$$

- $\ell_1, \ldots, \ell_n$: locations of $\mathcal{P}_1, \ldots, \mathcal{P}_n$
- η: variable valuation
- ξ: channel evaluation

مثال: ارسال بیت متناوب

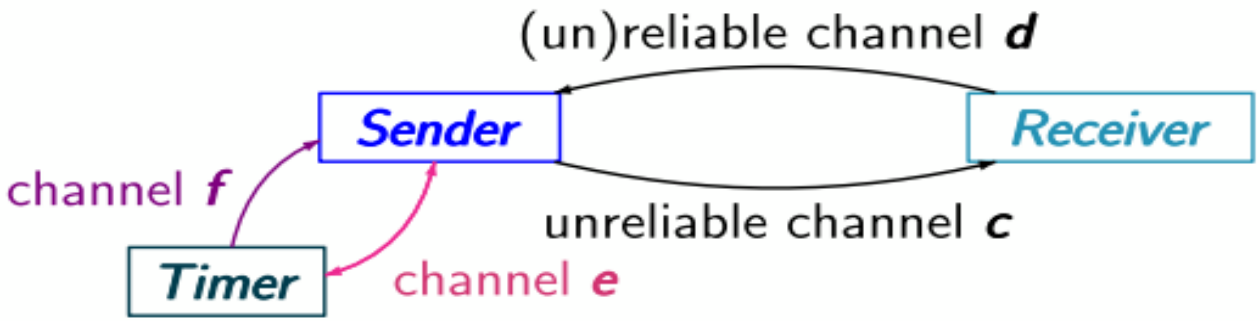
- سیستمی را در نظر بگیرید که یک ارسال کننده s و یک دریافت کننده R دارد. این دو جز از طریق دو کانال c, d با یکدیگر ارتباط دارند. ظرفیت هر دو کانال نیز نامتناهی است. هدف طراحی یک پروتکل ارتباطی است که مطمن شود پیام فرستاده شده از طرف s حتما به R رسیده است. پیام‌ها یکی یکی ارسال می‌شوند. ارسال کننده زمانی پیام جدید را ارسال می‌کند که مطمن باشد پیام قبلی تحویل داده شده است.

مثال: ارسال بیت متناوب




```
LOOP FOREVER
(1)  send message + bit y and activate timer
(2)  AWAIT timeout or acknowledgement DO
         IF timeout THEN goto (1)
                    ELSE turn off timer; y:=¬y
         FI
     OD
```

مثال: ارسال بیت متناوب

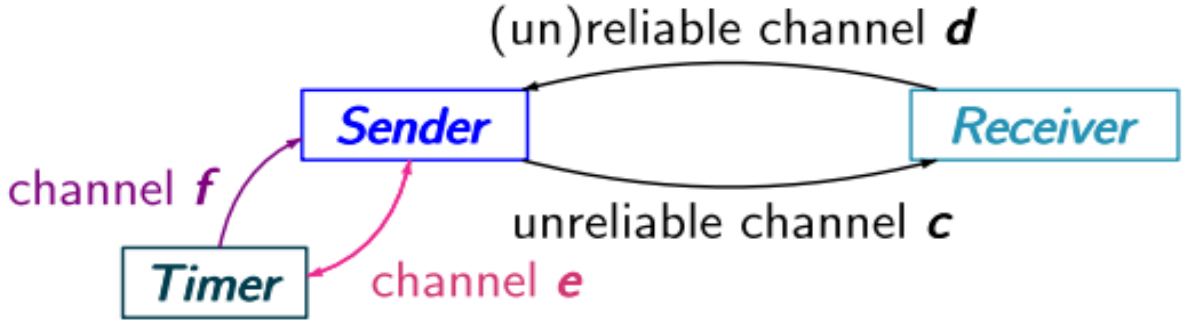


channel system: [*Sender* | *Timer* | *Receiver*]

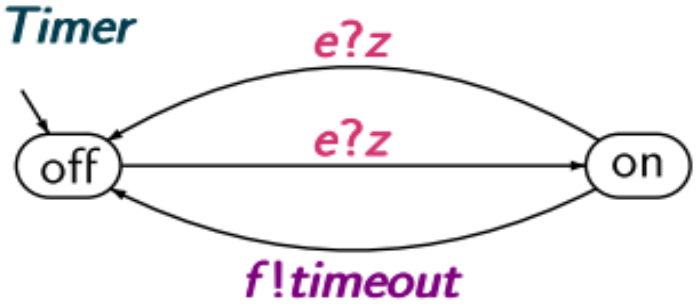
synchronous message passing between
Timer and *Sender* ← channels *e* and *f*

asynchronous message passing between
Receiver and *Sender* ← channels *c*, *d*

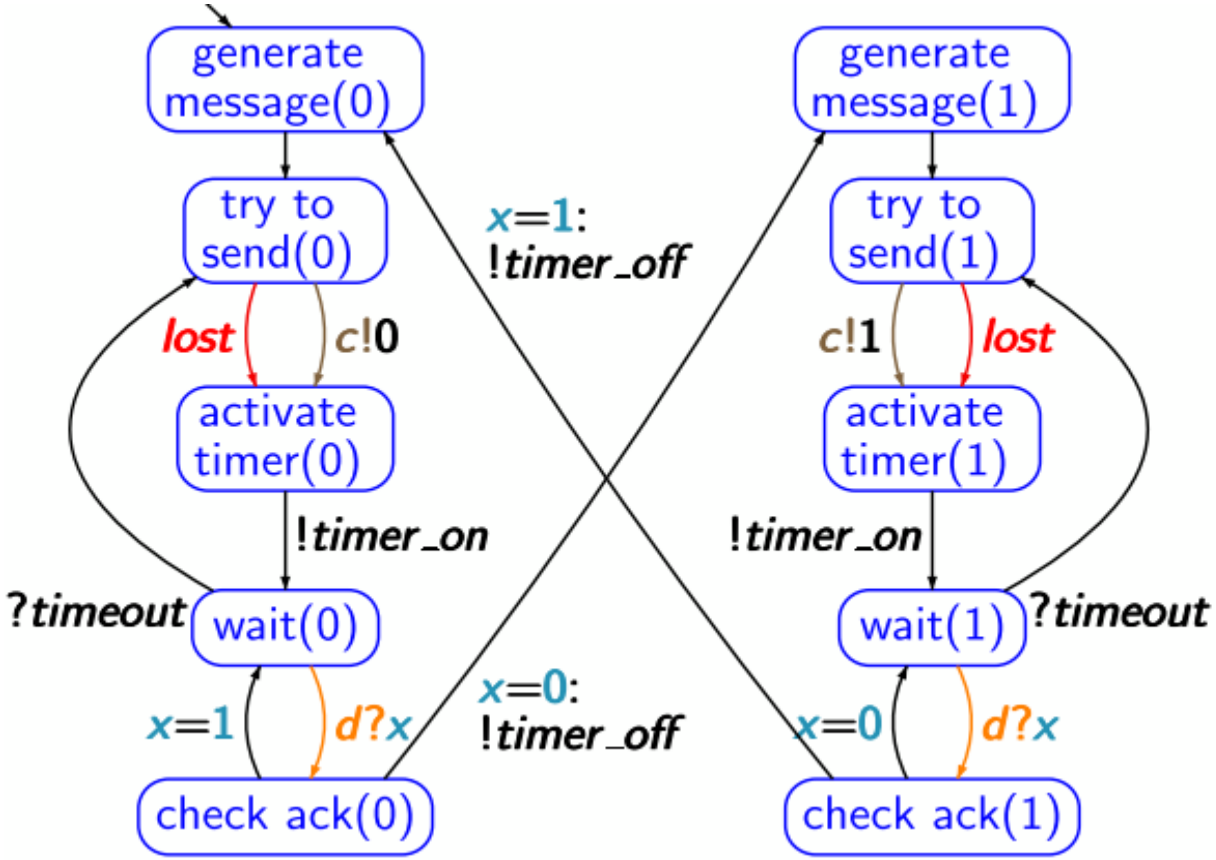
Program graph for timer



channel system: [*Sender* | *Timer* | *Receiver*]



Program graph for sender



Program graph for receiver