به نام خدا

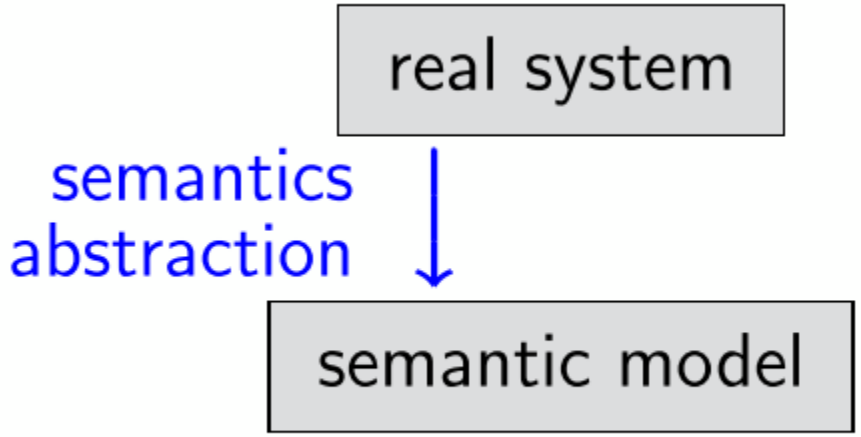
روش‌های رسمی در مهندسی نرم‌افزار

مرجان عظیمی
Azimi.marjan@gmail.com

فصل دوم

- مدل‌سازی سیستم‌های موازی
  - سیستم‌های گذار
  - مدل‌سازی سیستم‌های نرم‌افزاری و سخت‌افزاری
  - موازی‌سازی و ارتباطات

سیستم‌های گذار(Transition System)



سیستم‌های گذار

- گذارها در علم کامپیوتر برای توصیف رفتار سیستم استفاده می‌شوند.
- حالت(state): توصیف کننده برخی اطلاعات در مورد سیستم در یک زمان خاص است.

نود

- گذار(transition): چطور سیستم از یک حالت به حالت دیگر می‌رود.

یال

- اطلاعات اضافی در مورد:
  - ارتباطات بین حالات
  - ویژگی حالات(واقعیاتی را در مورد سیستم بیان می‌کند.)

سیستم‌های گذار

A transition system is a tuple

$$\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$$

- S is the state space, i.e., set of states,
- Act is a set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ is the transition relation,

  i.e., transitions have the form $s \xrightarrow{\alpha} s'$
  where $s, s' \in S$ and $\alpha \in Act$

- $S_0 \subseteq S$ the set of initial states,
- AP a set of atomic propositions,
- $L : S \to 2^{AP}$ the labeling function

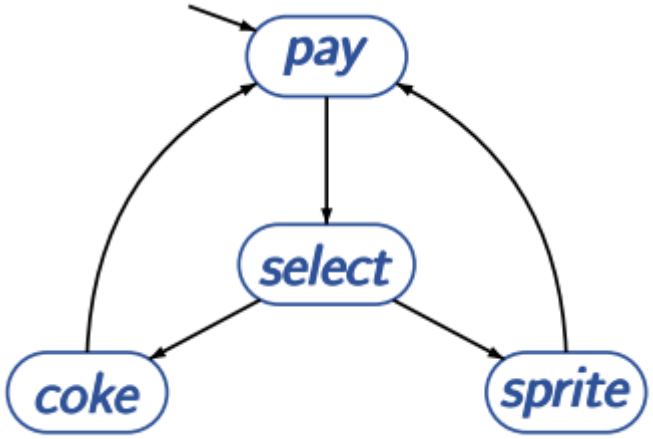
ماشین گذار

- رفتار شهودی ماشین گذار:
  - از یکسری حالات اولیه شروع می‌شود. در گیر یکسری گذارها می‌گردد(→). در گذار $s \xrightarrow{\alpha} s'$ عمل α انجام می‌شود و از حالت s به s' تغییر می‌کنیم. این انتقال ادامه می‌یابد تا در نهایت به حالتی می‌رسیم که از آن حالت گذار خروجی نداریم.

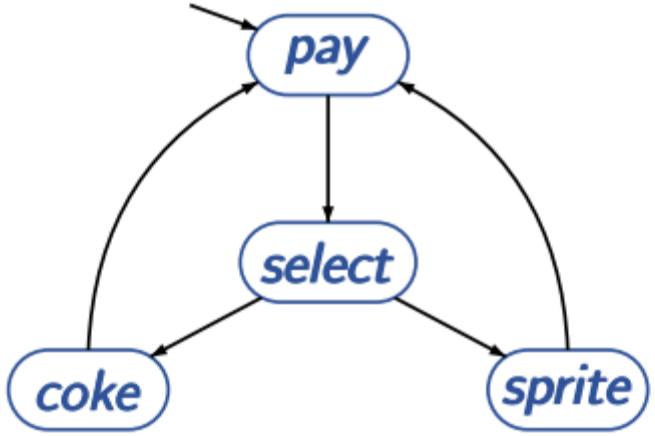
  - مجموعه حالات اولیه ممکن است تهی باشد که در این صورت ماشین گذار کاری انجام نمی‌دهد.

  - اگر حالات اولیه بیش از یک حالت باشد،یک حالت به صورت غیر قطعی انتخاب می‌شود.

مثال



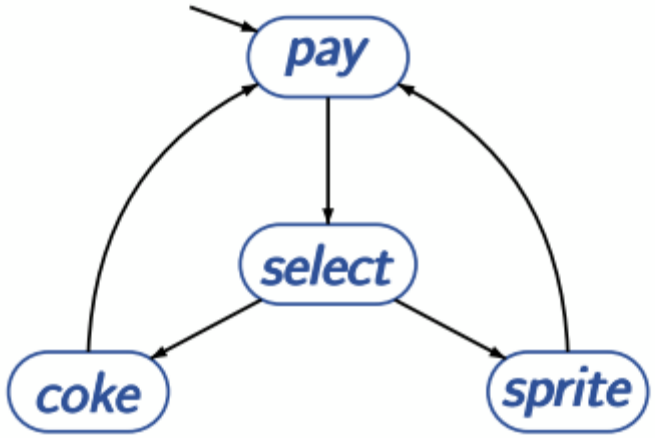
مثال



state space $S = \{pay, select, coke, sprite\}$

set of initial states: $S_0 = \{pay\}$

مثال

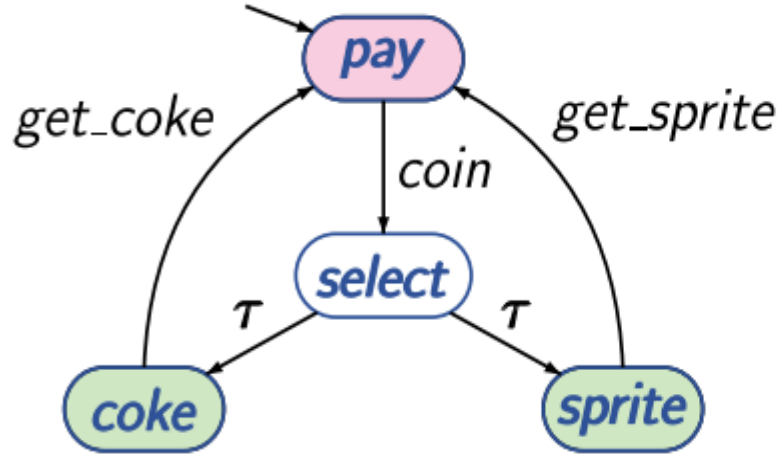


actions:
- *coin*
- τ
- *get_sprite*
- *get_coke*

state space $S = \{pay, select, coke, sprite\}$

set of initial states: $S_0 = \{pay\}$

مثال



actions:
- *coin*
- τ
- *get_sprite*
- *get_coke*

state space $S = \{pay, select, coke, sprite\}$

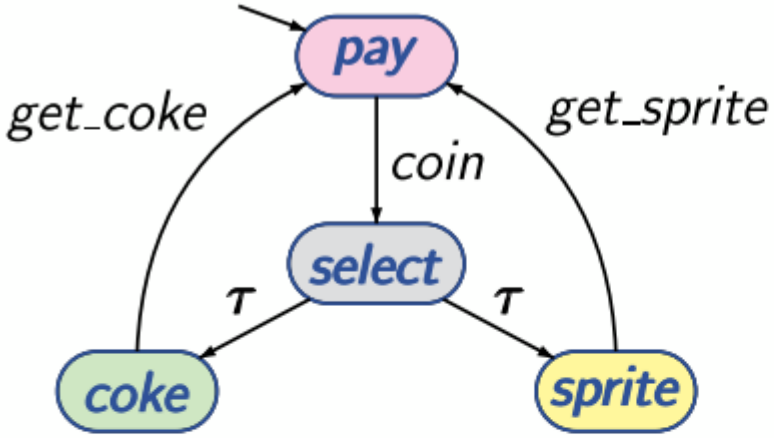
set of initial states: $S_0 = \{pay\}$

set of atomic propositions: $AP = \{pay, drink\}$

labeling function: $L(coke) = L(sprite) = \{drink\}$

$L(pay) = \{pay\}$, $L(select) = \emptyset$

مثال



actions:
- *coin*
- τ
- *get_sprite*
- *get_coke*

state space $S = \{pay, select, coke, sprite\}$

set of initial states: $S_0 = \{pay\}$

set of atomic propositions: $AP = S$

labeling function: $L(s) = \{s\}$ for each state s

تعاریف

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system. For $s \in S$ and $\alpha \in Act$, the set of direct α-*successors* of s is defined as:

$$Post(s, \alpha) = \left\{ s' \in S \mid s \xrightarrow{\alpha} s' \right\}, \quad Post(s) = \bigcup_{\alpha \in Act} Post(s, \alpha).$$

The set of α-*predecessors* of s is defined by:

$$Pre(s, \alpha) = \left\{ s' \in S \mid s' \xrightarrow{\alpha} s \right\}, \quad Pre(s) = \bigcup_{\alpha \in Act} Pre(s, \alpha).$$

تعاریف

- حالت پایانه(Terminal State)
  - حالاتی هستند که هیچ گذار خروجی ندارند.

1. *TS* is called *action-deterministic* if $|I| \leqslant 1$ and $|Post(s,\alpha)| \leqslant 1$ for all states s and actions α.

2. *TS* is called *AP-deterministic* if $|I| \leqslant 1$ and $|Post(s) \cap \{s' \in S \mid L(s') = A\}| \leqslant 1$ for all states s and $A \in 2^{AP}$.

اجرا

- اجرای متناهی

$$\varrho \;=\; s_0\,\alpha_1\,s_1\,\alpha_2\ldots\alpha_n\,s_n \text{ such that } s_i \xrightarrow{\alpha_{i+1}} s_{i+1} \text{ for all } 0 \leqslant i < n,$$

- اجرای نامتناهی

$$\rho \;=\; s_0\,\alpha_1\,s_1\,\alpha_2\,s_2\,\alpha_3\ldots \text{ such that } s_i \xrightarrow{\alpha_{i+1}} s_{i+1} \text{ for all } 0 \leqslant i.$$

- اجرای بیشینه: یک قطعه اجرای متناهی است که خاتمه‌اش یک حالت ترمینال باشد و یا یک قطعه اجرای نامتناهی باشد.

- اجرای اولیه: یک قطعه اجرا اولیه است اگر از یکی از حالت‌های اولیه شروع شود.

مثال

- ρ_1،ρ_2 اجراهای بیشینه هستند. ϱ،ρ_1 اجراهای اولیه هستند.
- اجراهای ماشین گذار باید اولیه و بیشینه باشند.

$$\rho_1 = pay \xrightarrow{coin} select \xrightarrow{\tau} soda \xrightarrow{sget} pay \xrightarrow{coin} select \xrightarrow{\tau} soda \xrightarrow{sget} \ldots$$

$$\rho_2 = select \xrightarrow{\tau} soda \xrightarrow{sget} pay \xrightarrow{coin} select \xrightarrow{\tau} beer \xrightarrow{bget} \ldots$$

$$\varrho = pay \xrightarrow{coin} select \xrightarrow{\tau} soda \xrightarrow{sget} pay \xrightarrow{coin} select \xrightarrow{\tau} soda \ .$$

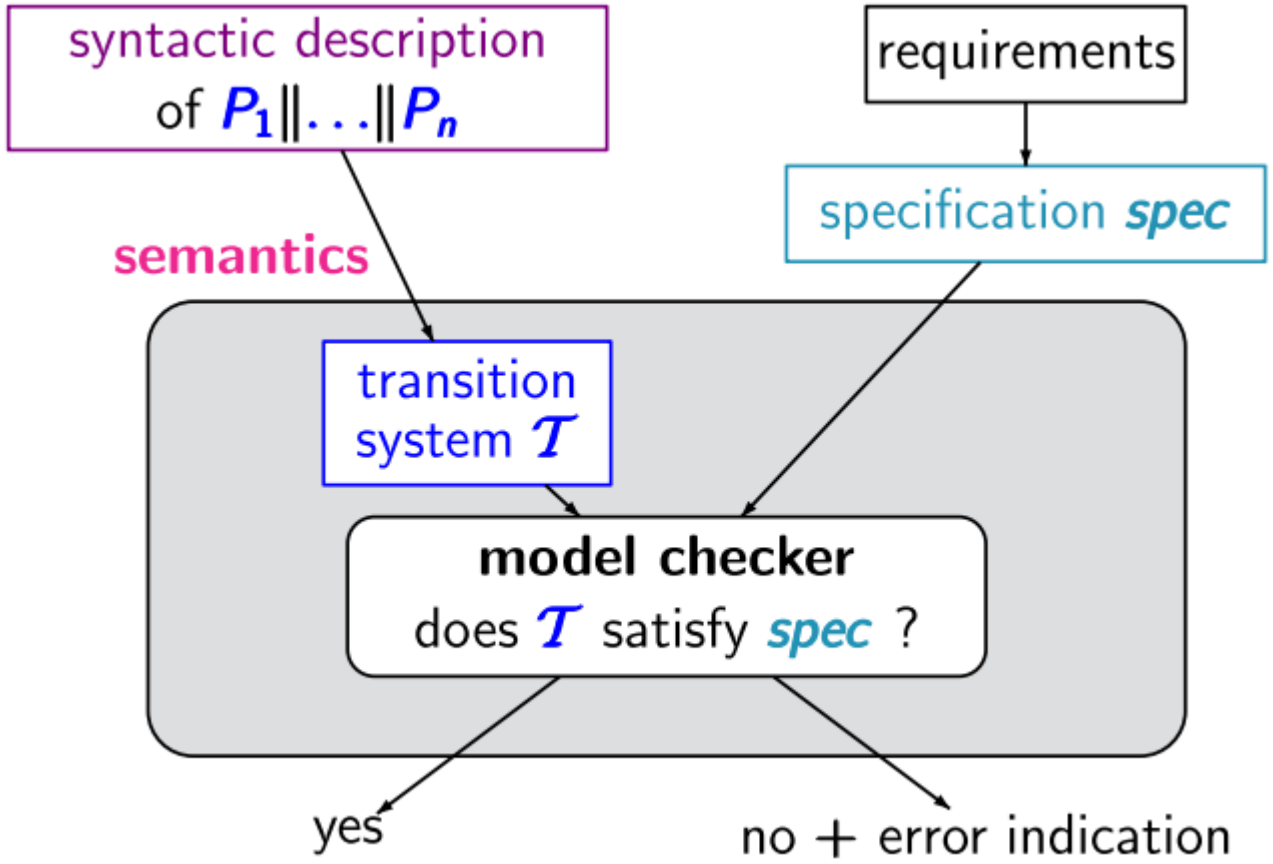
تعریف

- حالت قابل دسترسی(reachable state):
  - حالت s قابل دستیابی است اگر قطعه اجراهایی داشته باشیم که از حالات اولیه شروع شوند و به s ختم گردند.

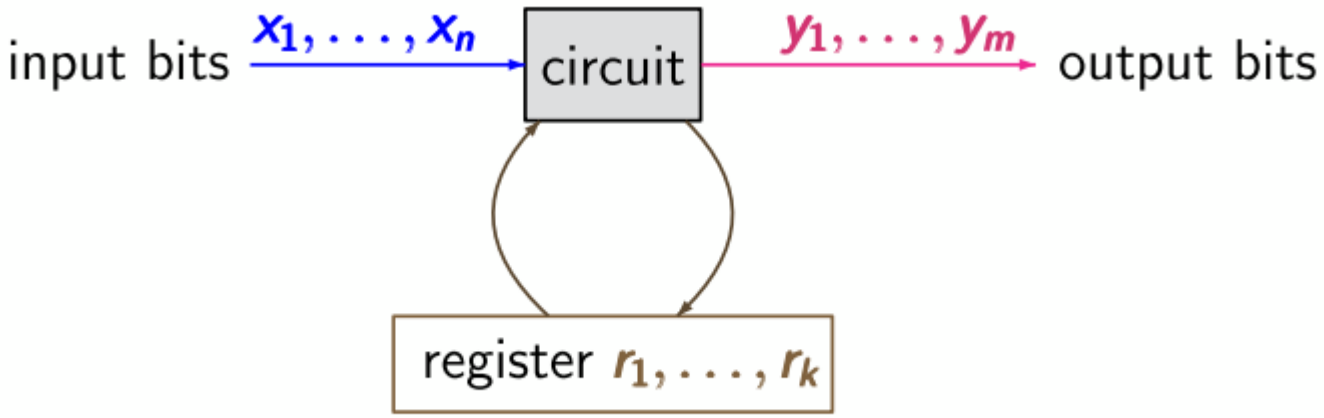
$$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} s_n = s \ .$$

- مدل کردن سیستم‌های سخت‌افزاری و نرم‌افزاری

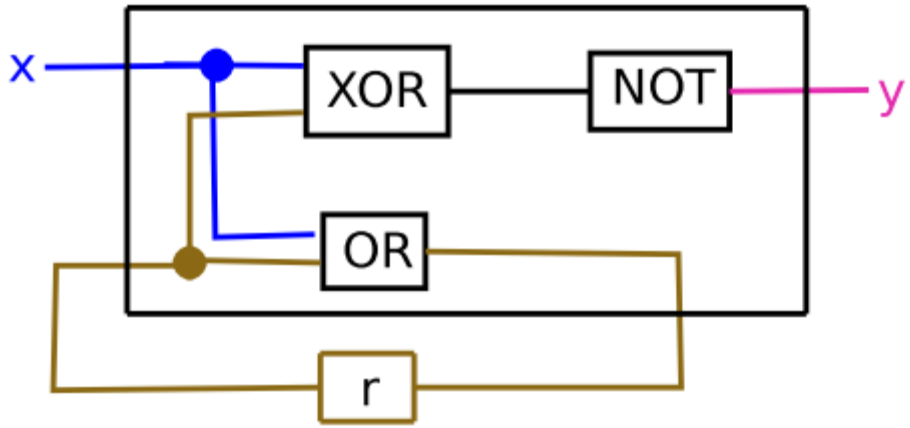
مدل کردن سیستم‌ها



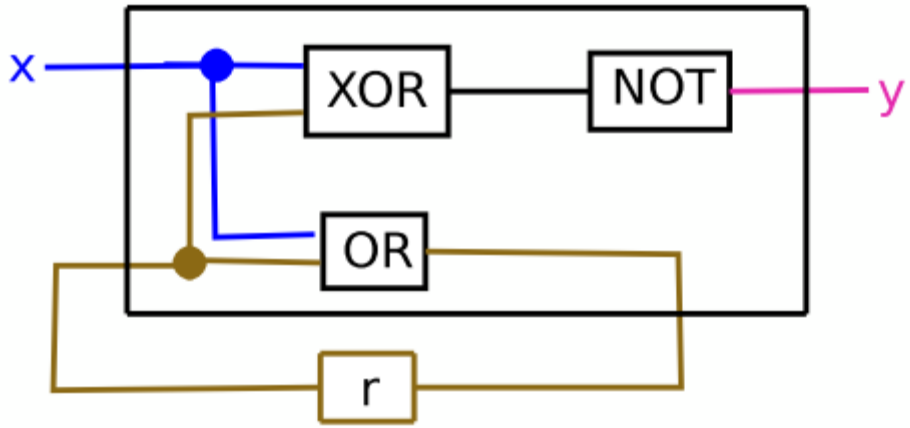
مدارهای سخت‌افزاری



مثال مدارهای سخت‌افزاری



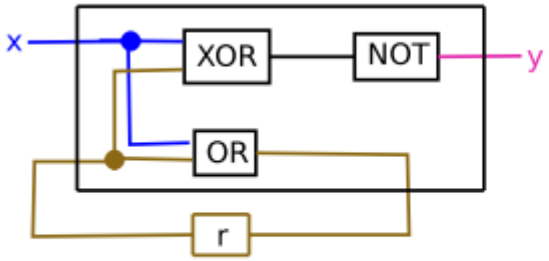
مثال مدارهای سخت‌افزاری



output function: $\lambda_y = \neg(x \oplus r)$

transition function: $\delta_r = x \vee r$

مثال مدارهای سخت‌افزاری



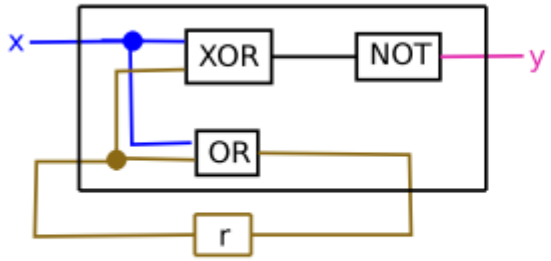
output function

$$\lambda_y = \neg(x \oplus r)$$

transition function

$$\delta_r = x \vee r$$

مثال مدارهای سخت‌افزاری



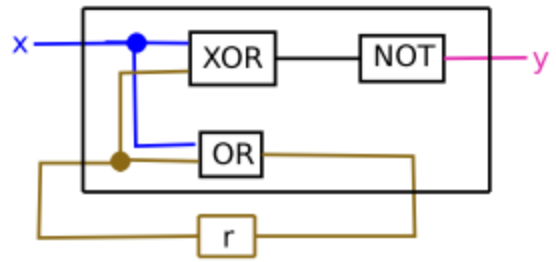
output function

$\lambda_y = \neg(x \oplus r)$

transition function

$\delta_r = x \vee r$

مثال مدارهای سخت‌افزاری



transition system

output function

$\lambda_y = \neg(x \oplus r)$

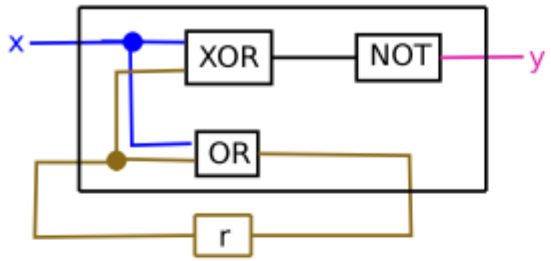
transition function

$\delta_r = x \vee r$

$x{=}0\ r{=}0$ $x{=}1\ r{=}0$

$x{=}0\ r{=}1$ $x{=}1\ r{=}1$

مثال مدارهای سخت‌افزاری



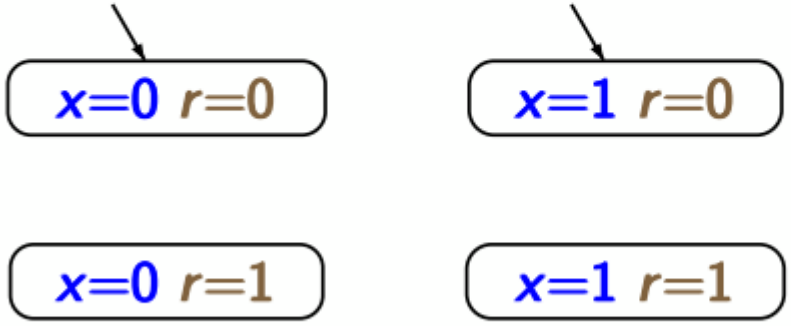
output function

$\lambda_y = \neg(x \oplus r)$

transition function

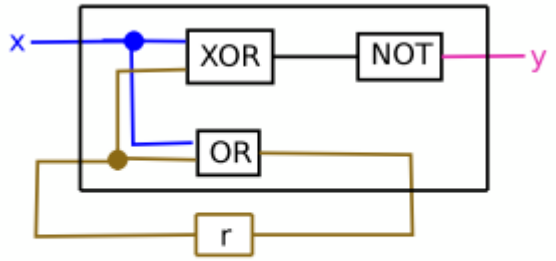
$\delta_r = x \vee r$

transition system



initial register evaluation: $r=0$

مثال مدارهای سخت‌افزاری



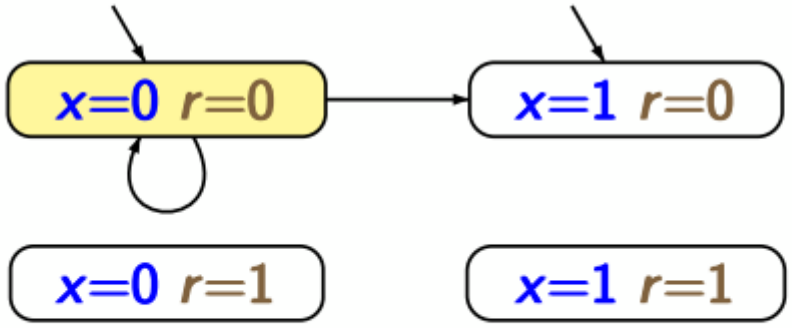
output function

$\lambda_y = \neg(x \oplus r)$

transition function

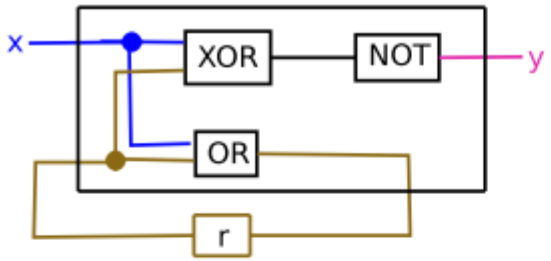
$\delta_r = x \vee r$

transition system



initial register evaluation: $r=0$

مثال مدارهای سخت‌افزاری



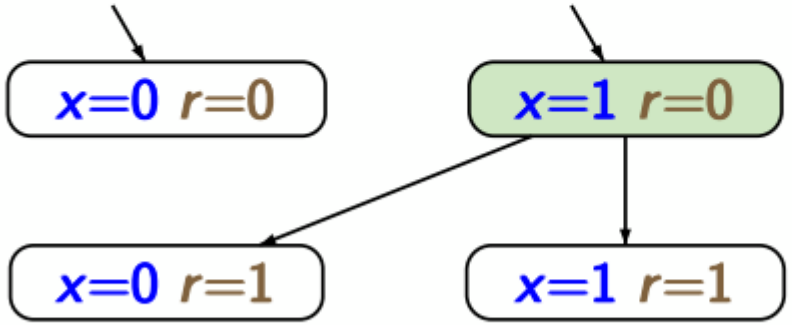
output function

$\lambda_y = \neg(x \oplus r)$

transition function

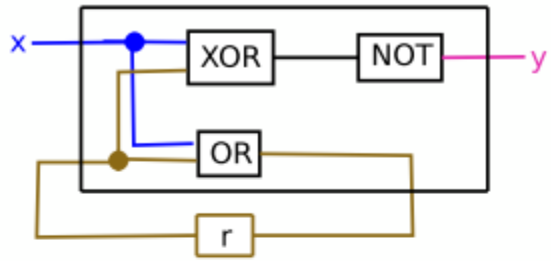
$\delta_r = x \vee r$

transition system



initial register evaluation: $r=0$

مثال مدارهای سخت‌افزاری



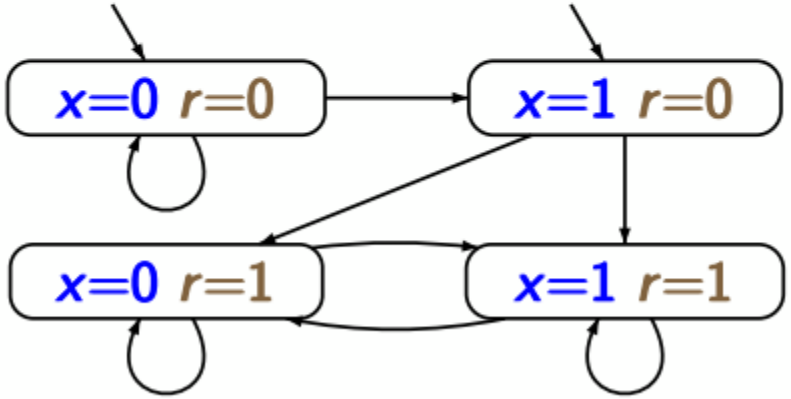
output function

$\lambda_y = \neg(x \oplus r)$

transition function

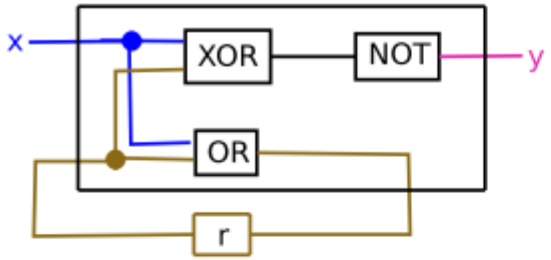
$\delta_r = x \vee r$

transition system



initial register evaluation: $r=0$

مثال مدارهای سخت‌افزاری



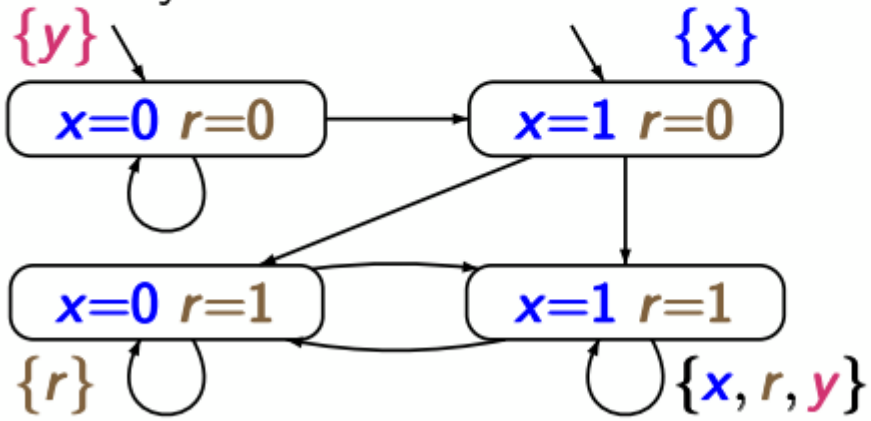
output function

$\lambda_y = \neg(x \oplus r)$

transition function

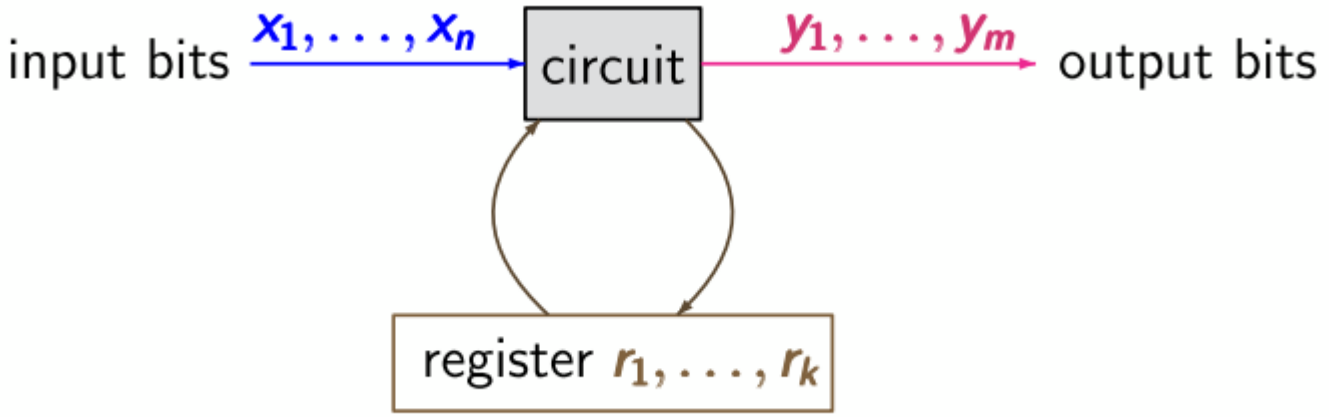
$\delta_r = x \vee r$

transition system

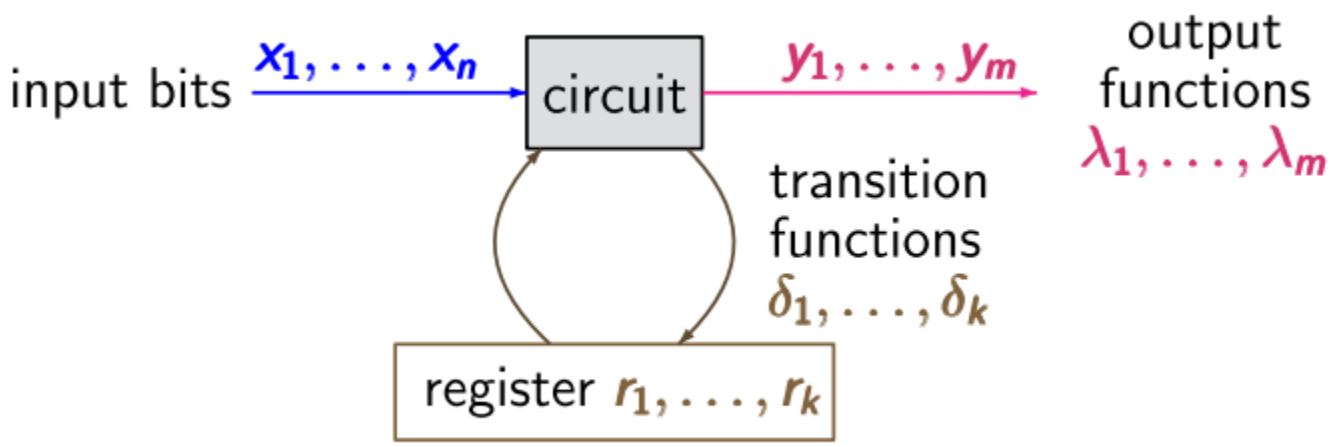


initial register evaluation: $r=0$

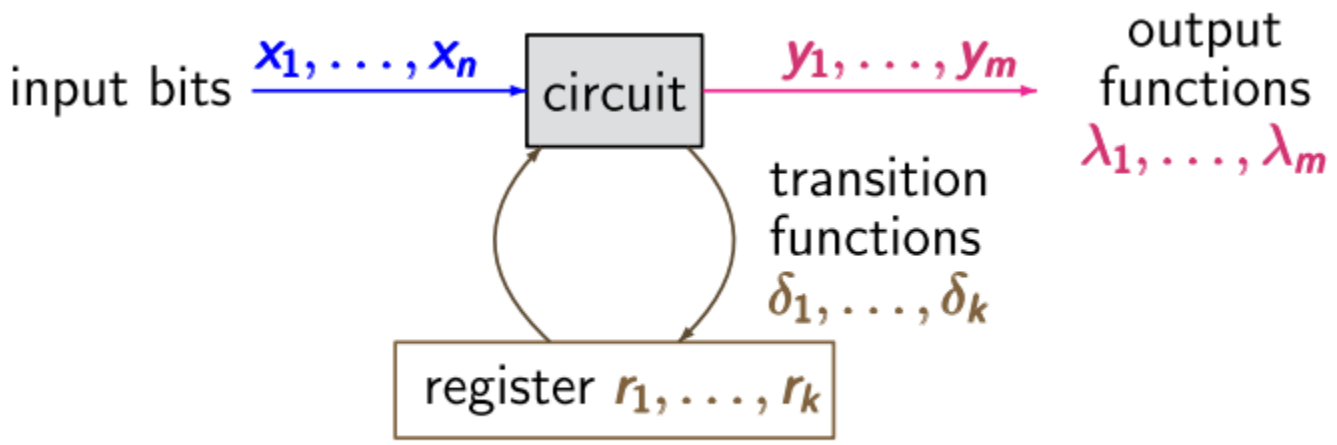
مدل کردن مدار سخت‌افزاری با سیستم گذار



مدل کردن مدار سخت‌افزاری با سیستم گذار



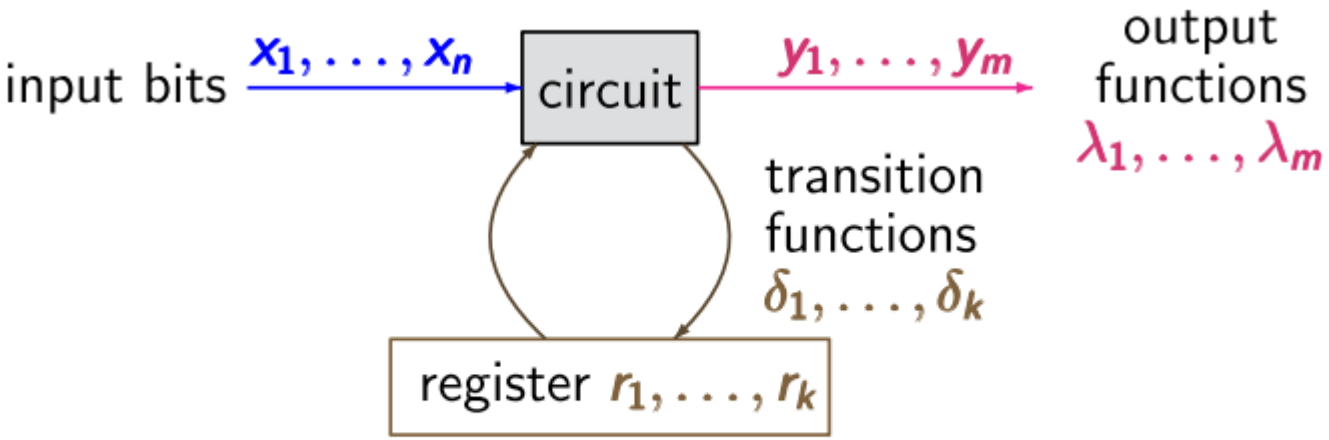
مدل کردن مدار سخت‌افزاری با سیستم گذار



$\delta_j, \lambda_i \mathrel{\widehat{=}}$ switching functions $\{0,1\}^n \times \{0,1\}^k \longrightarrow \{0,1\}$

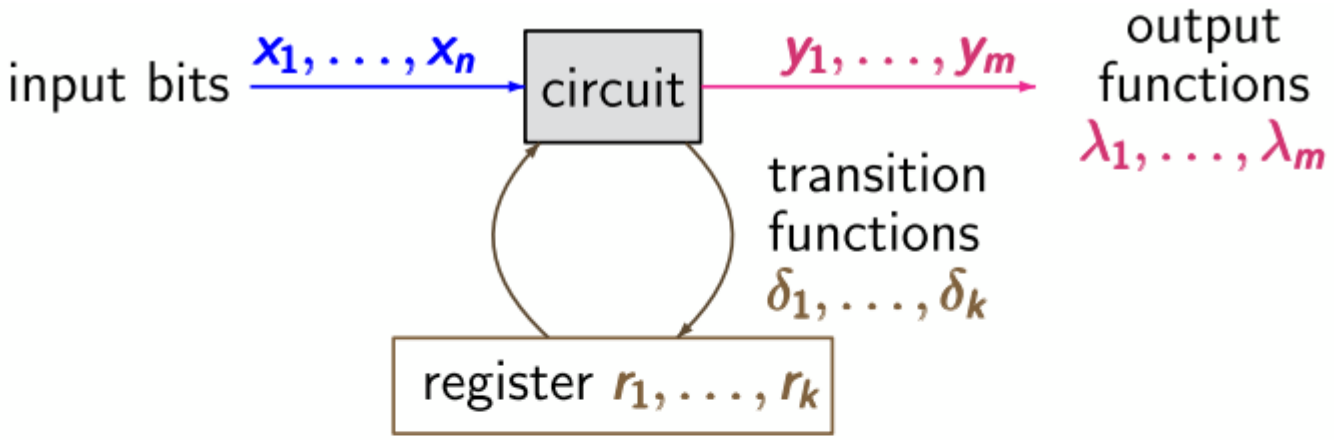
input values $a_1, \ldots, a_n$ for the input variables + current values $c_1, \ldots, c_k$ of the registers	$\mapsto$	output value $\lambda_i(\ldots)$ for output variable y_i next value $\delta_j(\ldots)$ for register r_j

مدل کردن مدار سخت‌افزاری با سیستم گذار



initial register evaluation $[r_1 = c_{01}, \ldots, r_k = c_{0k}]$

مدل کردن مدار سخت‌افزاری با سیستم گذار

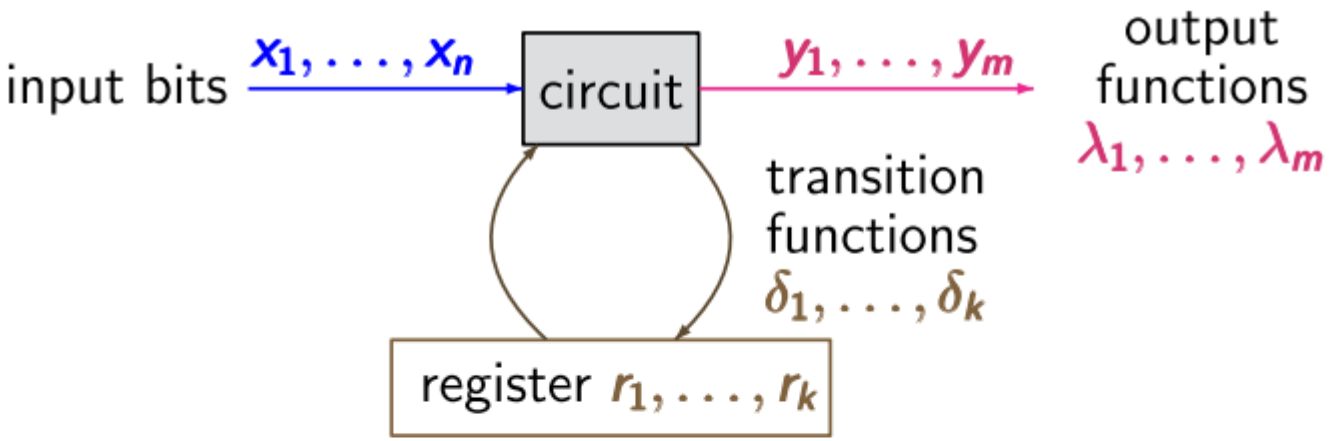


initial register evaluation $[r_1 = c_{01}, \ldots, r_k = c_{0k}]$

transition system:

- states: evaluations of $x_1, \ldots, x_n, r_1, \ldots, r_k$

مدل کردن مدار سخت‌افزاری با سیستم گذار

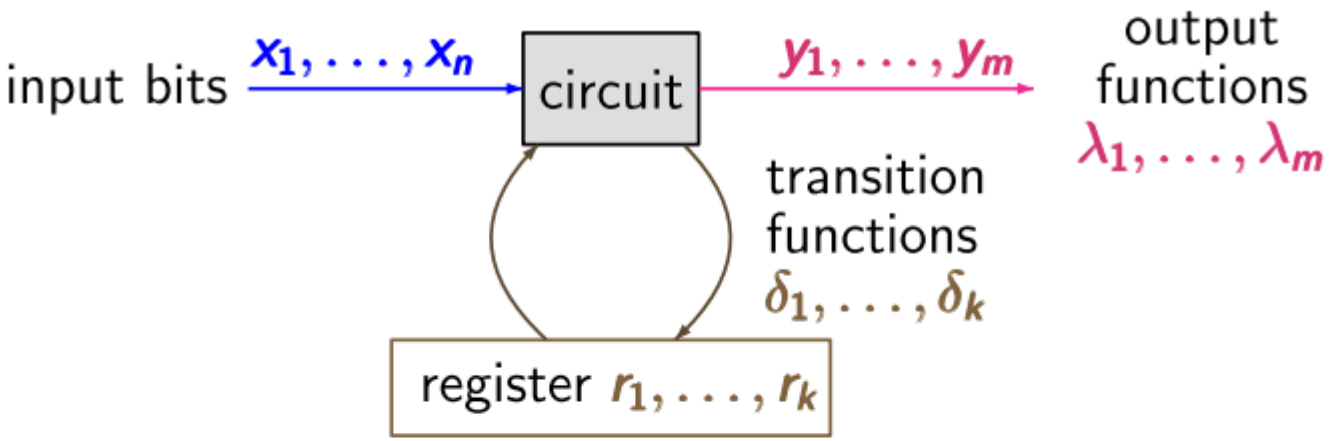


initial register evaluation $[r_1 = c_{01}, \ldots, r_k = c_{0k}]$

transition system:

- states: evaluations of $x_1, \ldots, x_n, r_1, \ldots, r_k$
- transitions represent the stepwise behavior

مدل کردن مدار سخت‌افزاری با سیستم گذار

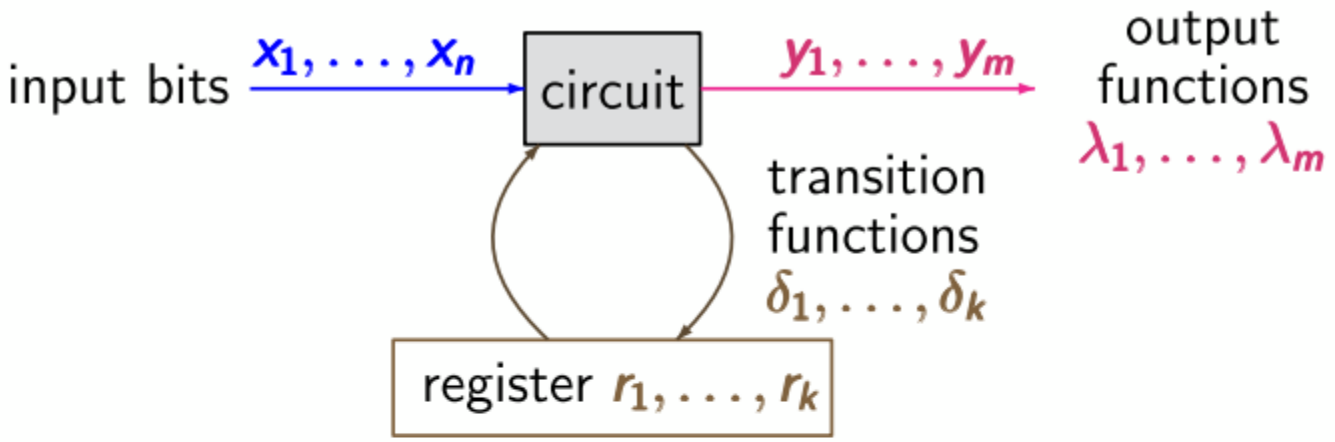


initial register evaluation $[r_1 = c_{01}, \ldots, r_k = c_{0k}]$

transition system:

- states: evaluations of $x_1, \ldots, x_n, r_1, \ldots, r_k$
- transitions represent the stepwise behavior
- values of input bits change nondeterministically

مدل کردن مدار سخت‌افزاری با سیستم گذار



initial register evaluation $[r_1 = c_{01}, \ldots, r_k = c_{0k}]$

transition system:

- states: evaluations of $x_1, \ldots, x_n, r_1, \ldots, r_k$
- transitions represent the stepwise behavior
- values of input bits change nondeterministically
- atomic propositions: $x_1, \ldots, x_n, y_1, \ldots, y_m, r_1, \ldots, r_k$

Data-Dependant-System

If x %2=1 then x=x+1 else x=2x

- زمانی که بخواهیم این قطعه کد را با سیستم گذار مدل نماییم. شرط را حذف و از غیر قطعی بودن استفاده می‌کنیم. می‌توانیم شرط‌ها را روی گذارها قرار دهیم.

مثال

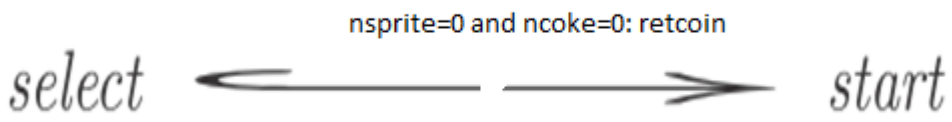
- یک تغییر در مثال قبل(دستگاه اتوماتیک فروش نوشابه) در نظر می‌گیریم. فرض کنیم تعداد sprite و coke را می‌شمارد و اگر هیچ نوشابه‌ای نداشت سکه را برمی‌گرداند.

مثال

- فرض کنیم ماشین فقط دو موقعیت start و select را دارد.
- برچسب گذارهای شرطی به فرم $g:\alpha$ هستند که g یک شرط بولین است که guard نامیده می‌شود و α عملی است که شرط g را حفظ می‌کند.

مثال

- فرض کنیم زمانی که عمل شارژ نوشابه‌ها رخ می‌دهد هر دو نوع نوشابه شارژ می‌شوند.



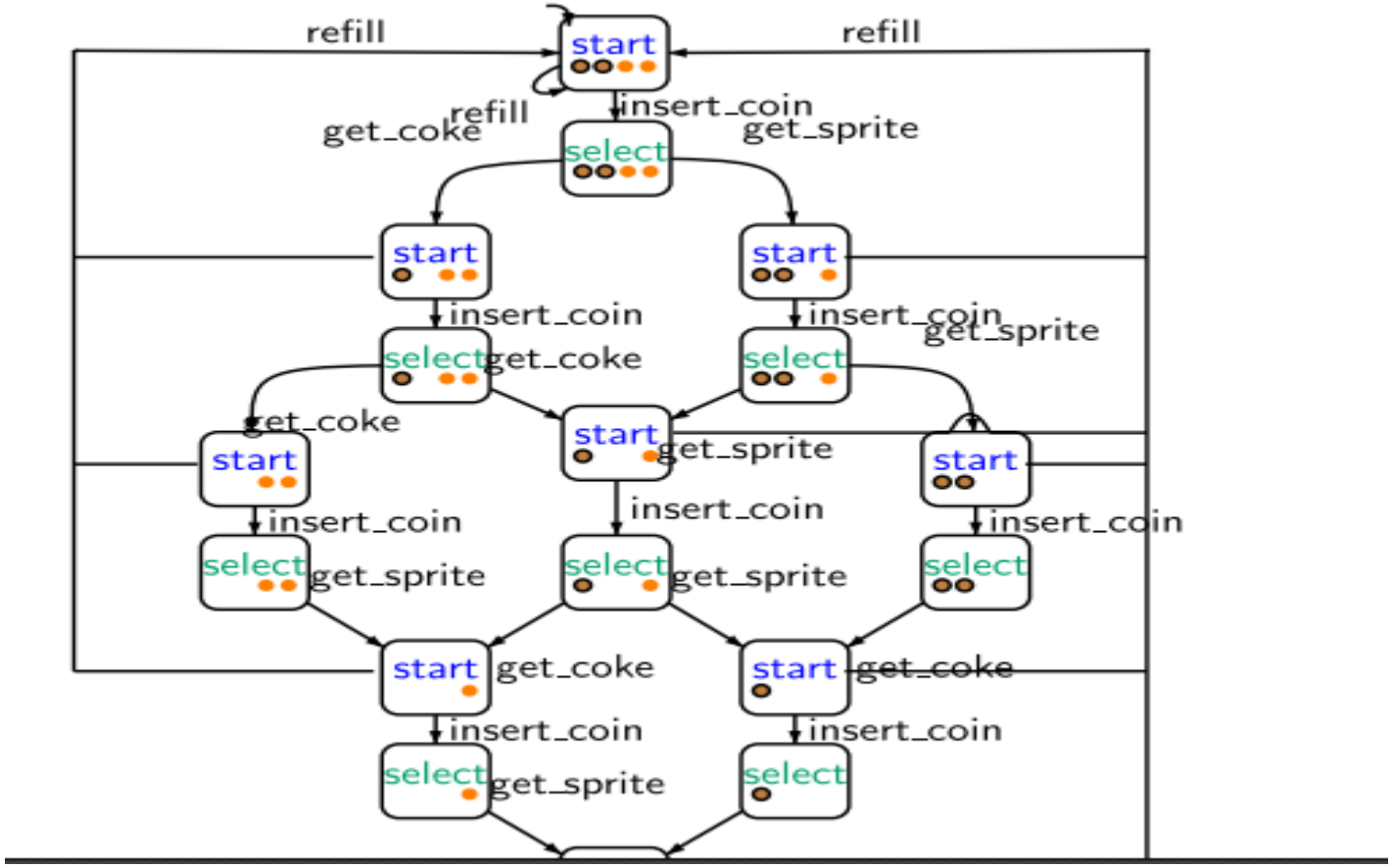
مثال

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)	

مثال

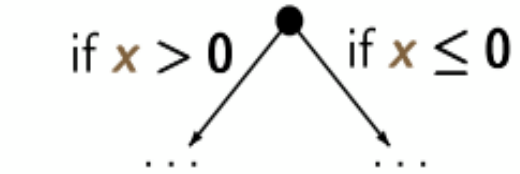

refill
start
refill
refill
get_coke
insert_coin
get_sprite
select
start
start
insert_coin
insert_coin
select
get_coke
select
get_coke
start
get_sprite
start
start
insert_coin
insert_coin
insert_coin
select
get_sprite
select
get_sprite
select
start
get_coke
start
get_coke
insert_coin
insert_coin
select
get_sprite
select

Data-Dependent System

example: sequential program

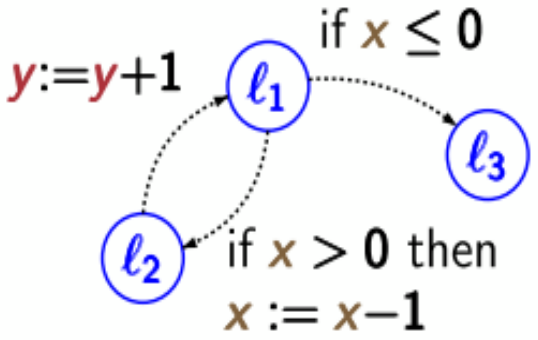
```
WHILE  x > 0 DO
       x := x−1;
       y := y+1
OD
...
```

Data-Dependent System

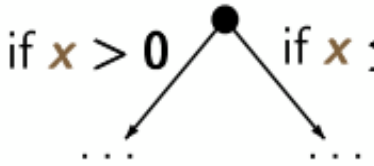


example: sequential program

```
WHILE  x > 0 DO
     x := x−1;
     y := y+1
OD
...
```



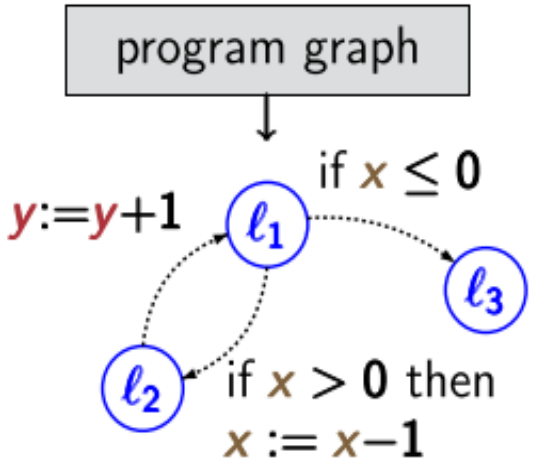
Data-Dependent System



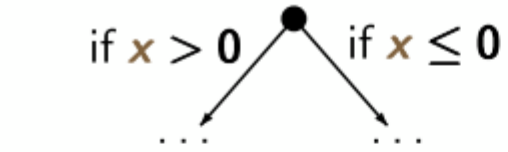
example: sequential program

$\ell_1 \rightarrow$ WHILE $x > 0$ DO
$x := x-1;$
$\ell_2 \rightarrow$ $y := y+1$
OD
$\ell_3 \rightarrow$ ...

ℓ_1, ℓ_2, ℓ_3 are locations,
i.e., control states

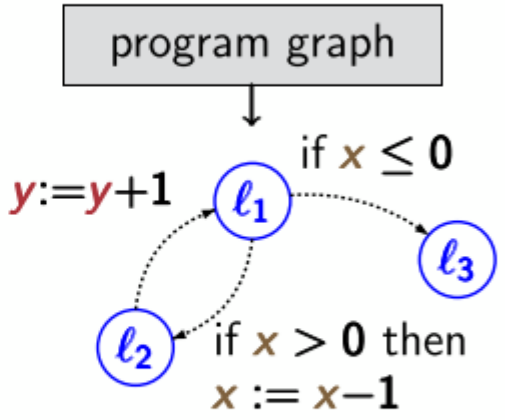


Data-Dependent System



example: sequential program

$\ell_1 \rightarrow$ WHILE $x > 0$ DO
$x := x-1$;
$\ell_2 \rightarrow$ $y := y+1$
OD
$\ell_3 \rightarrow$ ...



states of the transition system:

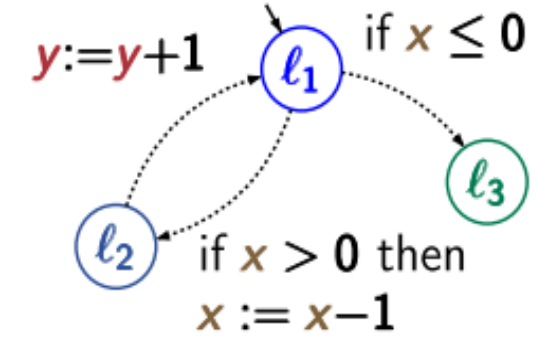
locations + relevant data (*here:* values for x and y)

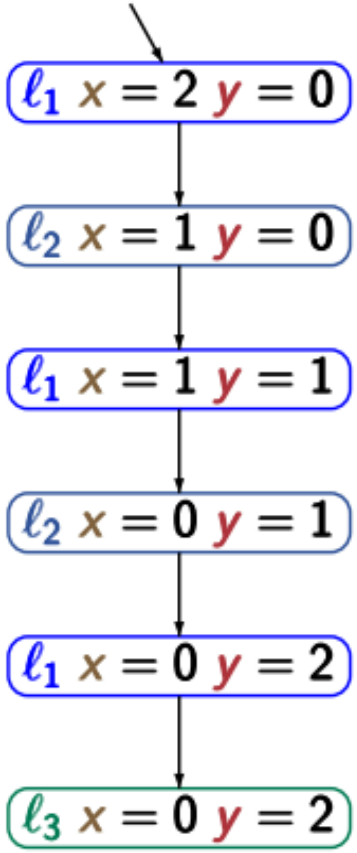
Data-Dependent System

initially: $x = 2,\ y = 0$

$\ell_1 \rightarrow$ WHILE $x > 0$ DO
$x := x-1$
$\ell_2 \rightarrow$ $y := y+1$
OD
$\ell_3 \rightarrow$ ...

program graph





Data-Dependent System

initially: $x = 2,\ y = 0$

$\ell_1 \rightarrow$ WHILE $x > 0$ DO

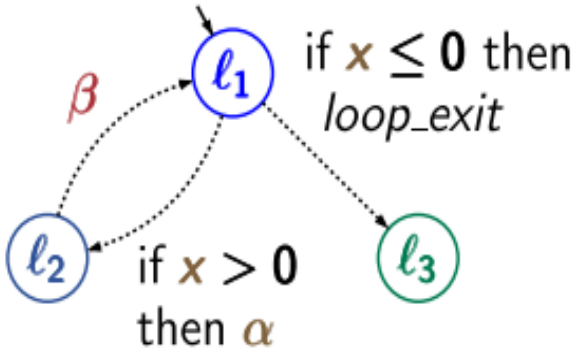
$x := x-1$ ← action α

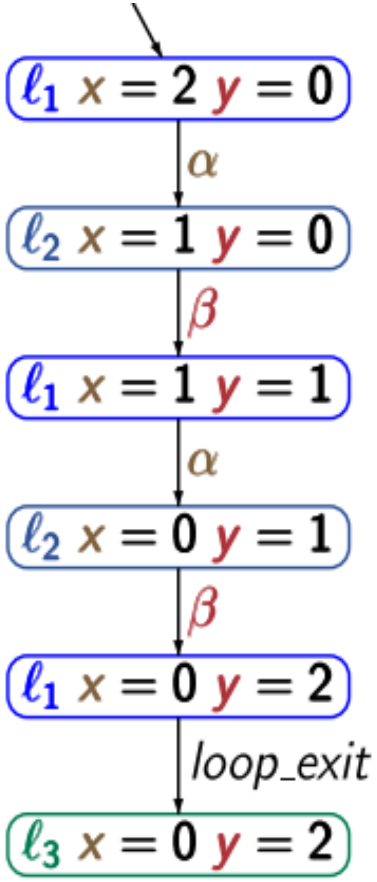
$\ell_2 \rightarrow$ $y := y+1$ ← action β

OD

$\ell_3 \rightarrow$ ...

program graph





Program Graph

- گرافی که یال‌های گذار شرطی هستند سیستم گذار نیست بلکه یک گراف برنامه(Program Graph) نامیده می‌شود.

Program Graph

Let Var be a set of typed variables.

A *program graph* over Var is a tuple

$$\mathcal{P} = (\mathit{Loc}, \mathit{Act}, \mathit{Effect}, \hookrightarrow, \mathit{Loc}_0, g_0) \text{ where}$$

Program Graph

Let Var be a set of typed variables.

A *program graph* over Var is a tuple

$$\mathcal{P} = (Loc, Act, Effect, \hookrightarrow, Loc_0, g_0) \text{ where}$$

- Loc is a (finite) set of locations, i.e., control states,
- Act a set of actions,
- $Effect : Act \times Eval(Var) \to Eval(Var)$

↑

function that formalizes the effect of the actions

example: if α is the assignment $x{:=}x{+}y$ then

$Effect(\alpha, [x{=}1, y{=}7]) = [x{=}8, y{=}7]$

Program Graph

Let Var be a set of typed variables.

A *program graph* over Var is a tuple

$$\mathcal{P} = (Loc, Act, Effect, \hookrightarrow, Loc_0, g_0) \text{ where}$$

- Loc is a (finite) set of locations, i.e., control states,
- Act a set of actions,
- $Effect : Act \times Eval(Var) \to Eval(Var)$
- $\hookrightarrow \subseteq Loc \times Cond(Var) \times Act \times Loc$

Program Graph

Let Var be a set of typed variables.

A *program graph* over Var is a tuple

$$\mathcal{P} = (Loc, Act, Effect, \hookrightarrow, Loc_0, g_0) \text{ where}$$

- Loc is a (finite) set of locations, i.e., control states,
- Act a set of actions,
- $Effect : Act \times Eval(Var) \to Eval(Var)$
- $\hookrightarrow \subseteq Loc \times Cond(Var) \times Act \times Loc$

  specifies conditional transitions of the form $\ell \xrightarrow{g:\alpha} \ell'$

ℓ, ℓ' are locations, $g \in Cond(Var)$, $\alpha \in Act$

Program Graph

Let Var be a set of typed variables.

A *program graph* over Var is a tuple

$$\mathcal{P} = (\mathit{Loc}, \mathit{Act}, \mathit{Effect}, \hookrightarrow, \mathit{Loc}_0, g_0) \text{ where}$$

- Loc is a (finite) set of locations, i.e., control states,
- Act a set of actions,
- $\mathit{Effect} : \mathit{Act} \times \mathit{Eval}(\mathit{Var}) \to \mathit{Eval}(\mathit{Var})$
- $\hookrightarrow \subseteq \mathit{Loc} \times \mathit{Cond}(\mathit{Var}) \times \mathit{Act} \times \mathit{Loc}$

  specifies conditional transitions of the form $\ell \xhookrightarrow{g:\alpha} \ell'$
- $\mathit{Loc}_0 \subseteq \mathit{Loc}$ is the set of initial locations,
- $g_0 \in \mathit{Cond}(\mathit{Var})$ initial condition on the variables.

مثال ماشین نوشابه

- Set Loc={start,select}
- Var={ncoke, nsprite}
- Act={sget,cget,insert-coin,ret-coin,refill}
- Effect(inser-coin,η)=η
- Effect(ret-coin,η)=η
- Effect(sget,η)=$\eta[\text{nsprite} = \text{nsprite} - 1]$
- Effect(cget,η)=$\eta[\text{ncoke} = \text{ncoke} - 1]$
- Effect(refill,η)=$\eta[\text{nsprite} = \max, \text{ncoke} = \max]$
- g_0=(ncoke=max and nsprite=max)

Program Graph

- هر program Graph می‌تواند در قالب یک Transition System تفسیر شود.

TS semantic of program graph

program graph $\mathcal{P}$ over Var

$\Downarrow$

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

TS semantic of program graph

program graph $\mathcal{P}$ over Var

$\Downarrow$

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

states in $\mathcal{T}_{\mathcal{P}}$ have the form

$\langle \ell, \eta \rangle$

location — variable evaluation

TS semantic of program graph

Let $\mathcal{P} = (Loc, Act, Effect, \hookrightarrow, Loc_0, g_0)$ be a PG.
The transition system of $\mathcal{P}$ is:

$$\mathcal{T}_{\mathcal{P}} = (S, Act, \longrightarrow, S_0, AP, L)$$

TS semantic of program graph

Let $\mathcal{P} = (Loc, Act, Effect, \hookrightarrow, Loc_0, g_0)$ be a PG.
The transition system of $\mathcal{P}$ is:

$$\mathcal{T}_{\mathcal{P}} = (S, Act, \longrightarrow, S_0, AP, L)$$

- state space: $S = Loc \times Eval(Var)$
- initial states: $S_0 = \{\langle \ell, \eta \rangle : \ell \in Loc_0, \eta \models g_0\}$

The transition relation $\longrightarrow$ is given by the following rule:

$$\frac{\ell \xhookrightarrow{g:\alpha} \ell' \ \wedge \ \eta \models g}{\langle \ell, \eta \rangle \xrightarrow{\alpha} \langle \ell', Effect(\alpha, \eta) \rangle}$$

TS semantic of program graph

Let $\mathcal{P} = (Loc, Act, Effect, \hookrightarrow, Loc_0, g_0)$ be a PG.

transition system $\mathcal{T}_{\mathcal{P}} = (S, Act, \longrightarrow, S_0, AP, L)$

- state space: $S = Loc \times Eval(Var)$
- initial states: $S_0 = \{\langle \ell, \eta \rangle : \ell \in Loc_0, \eta \models g_0\}$
- $\longrightarrow$ is given by the following rule:

$$\frac{\ell \xhookrightarrow{g:\alpha} \ell' \ \wedge \ \eta \models g}{\langle \ell, \eta \rangle \xrightarrow{\alpha} \langle \ell', Effect(\eta, \alpha) \rangle}$$

- atomic propositions: $AP = Loc \cup Cond(Var)$
- labeling function:

$$L(\langle \ell, \eta \rangle) = \{\ell\} \cup \{g \in Cond(Var) : \eta \models g\}$$