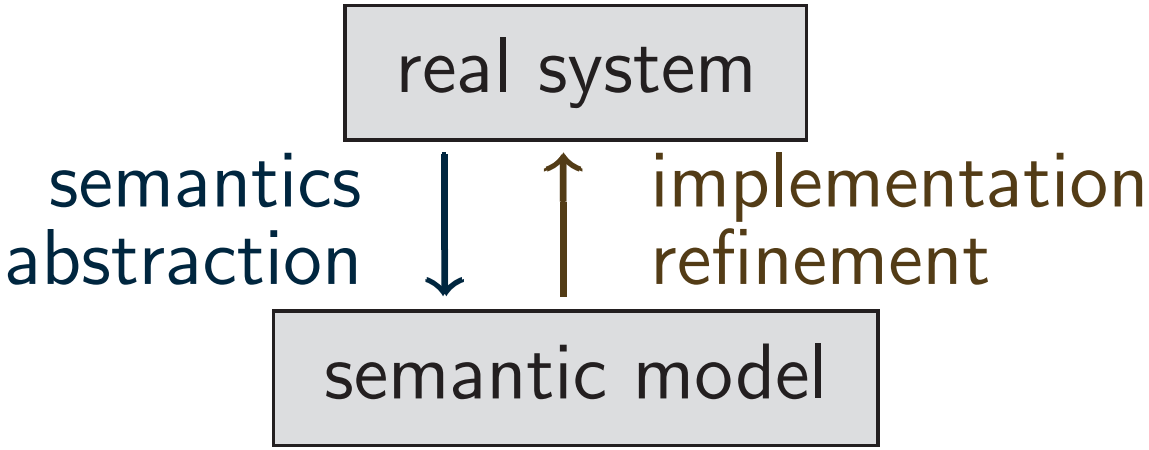
Transition systems $\widehat{=}$ extended digraphs



The semantic model yields a formal representation of:

- the states of the system ⟵ **nodes**
- the stepwise behaviour ⟵ **transitions**
- the initial states
- additional information on
  - communication ⟵ **actions**
  - state properties ⟵ **atomic proposition**

Transition system (TS)

A transition system is a tuple

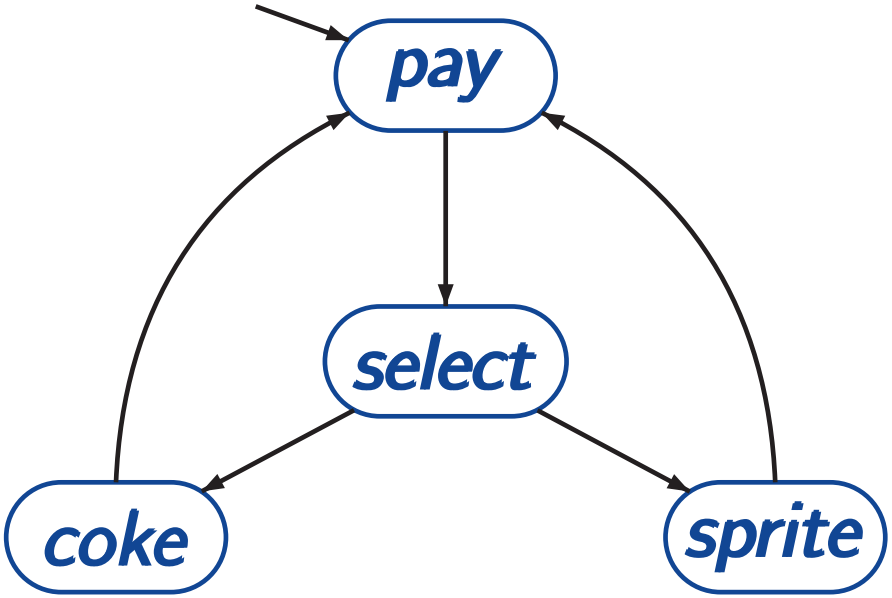
$$\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$$

- 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,

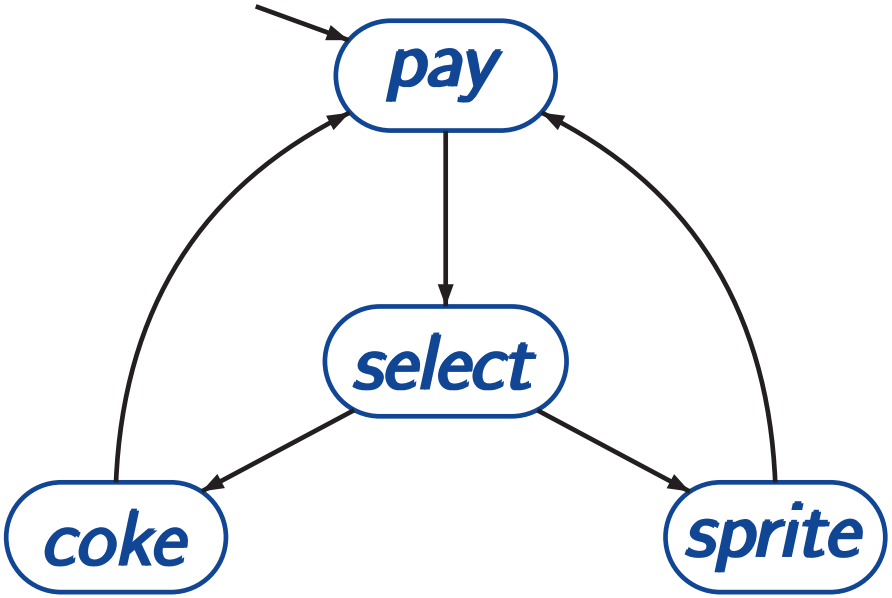
  i.e., transitions have the form $s \xrightarrow{\alpha} s'$
  where $s, s' \in S$ and $\alpha \in Act$

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

Transition system for beverage machine



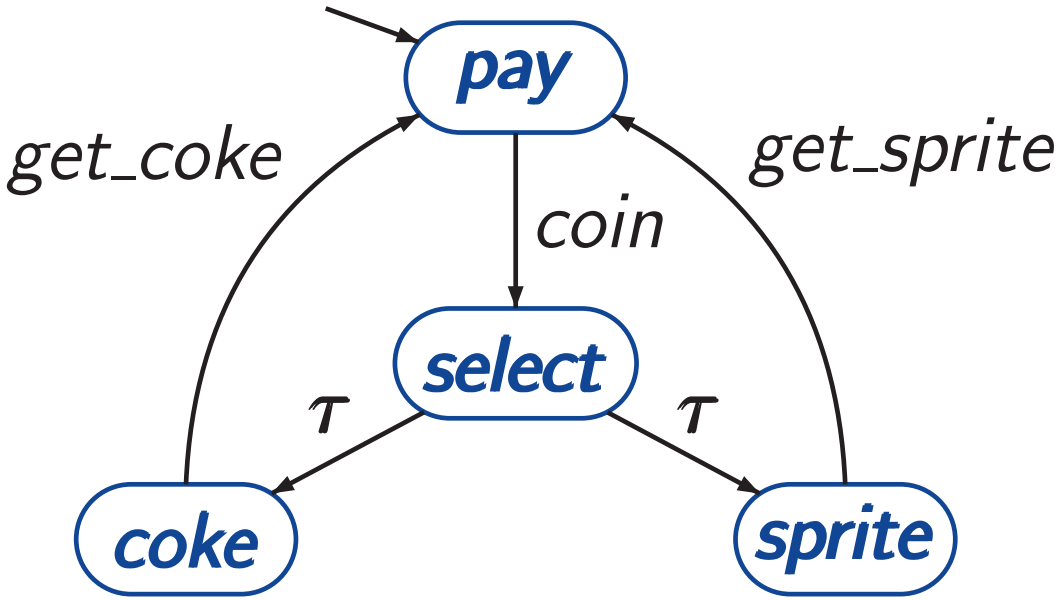
Transition system for beverage machine



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

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

Transition system for beverage machine

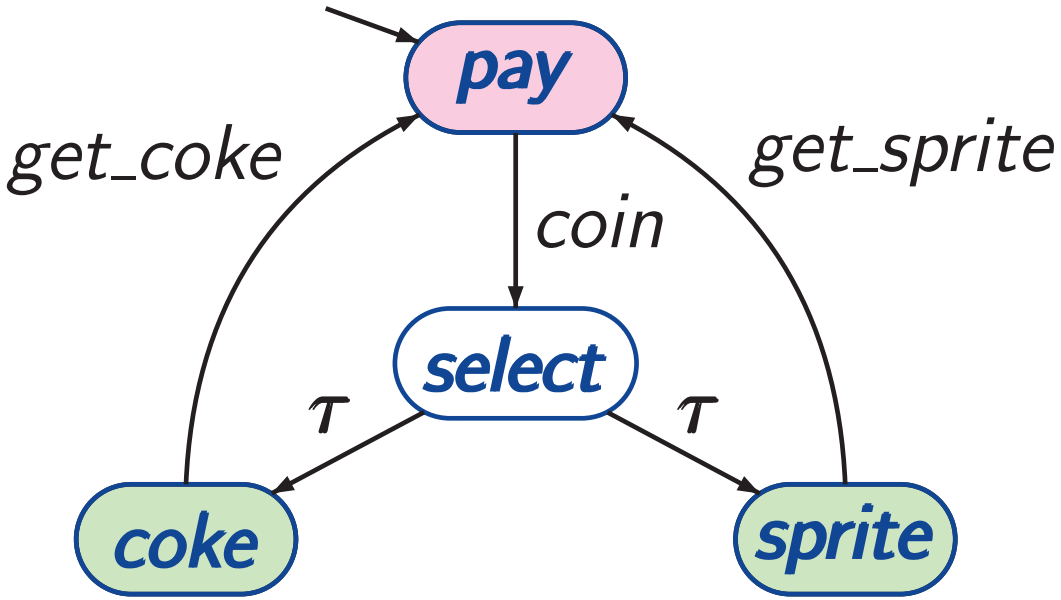


actions:
- *coin*
- τ
- *get_sprite*
- *get_coke*

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

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

Transition system for beverage machine



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

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

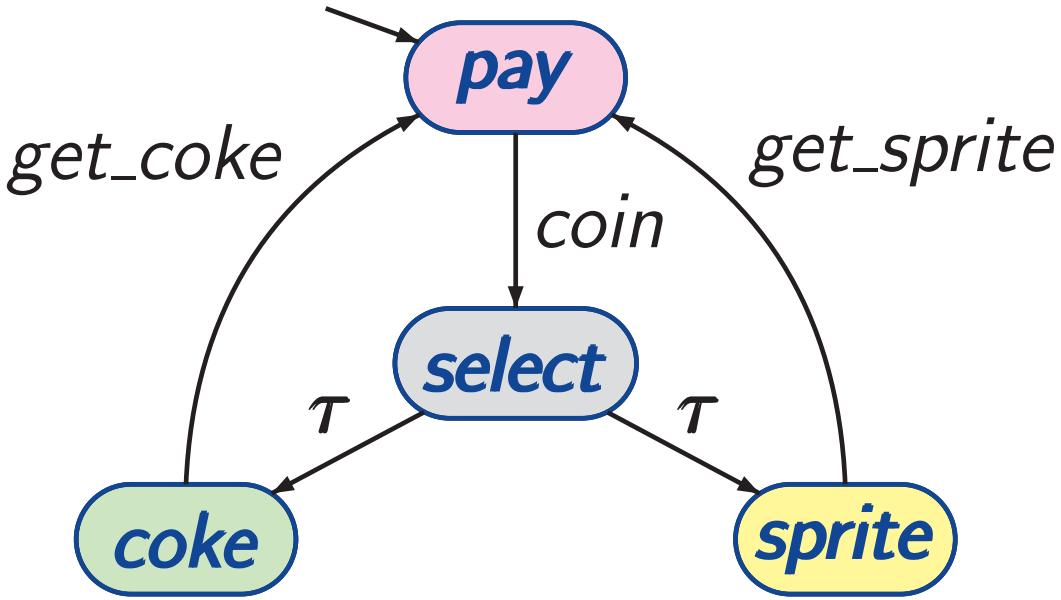
set of initial states: $S_0 = \{pay\}$

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

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

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

Transition system for beverage machine



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

"Behaviour" of transition systems

possible behaviours of a TS result from:

select nondeterministically an initial state $s \in S_0$

WHILE s is non-terminal DO

select nondeterministically a transition $s \xrightarrow{\alpha} s'$

execute the action α and put $s := s'$

OD

"Behaviour" of transition systems

possible behaviours of a TS result from:

select nondeterministically an initial state $s \in S_0$

WHILE s is non-terminal DO

select nondeterministically a transition $s \xrightarrow{\alpha} s'$

execute the action α and put $s := s'$

OD

executions: maximal "transition sequences"

$$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots \text{ with } s_0 \in S_0$$

"Behaviour" of transition systems

possible behaviours of a TS result from:

select nondeterministically an initial state $s \in S_0$
WHILE s is non-terminal DO
    select nondeterministically a transition $s \xrightarrow{\alpha} s'$
    execute the action α and put $s := s'$
OD

executions: maximal "transition sequences"

$$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots \text{ with } s_0 \in S_0$$

reachable fragment:

$\mathit{Reach}(\mathcal{T})$ = set of all states that are reachable from an initial state through some execution

Possible meanings of nondeterminism in TS

- (true) concurrency modeled by interleaving
- competition of parallel dependent actions
- implementational freedom, underspecification
- incomplete information on system environment

Transition system for parallel actions

parallel execution of independent actions

parallel execution of dependent actions

Transition system for parallel actions

parallel execution of independent actions

e.g. $\underbrace{x := x+1}_{\text{action } \alpha} \;|||\; \underbrace{y := y-3}_{\text{action } \beta}$ α, β independent

parallel execution of dependent actions

Transition system for parallel actions

parallel execution of independent actions

e.g. $\underbrace{x := x+1}_{\text{action } \alpha} \;|||\; \underbrace{y := y-3}_{\text{action } \beta}$ α, β independent

parallel execution of dependent actions

e.g. $\underbrace{x := x+1}_{\text{action } \alpha} \;|||\; \underbrace{y := 2*x}_{\text{action } \beta}$ α, β dependent

Transition system for parallel actions

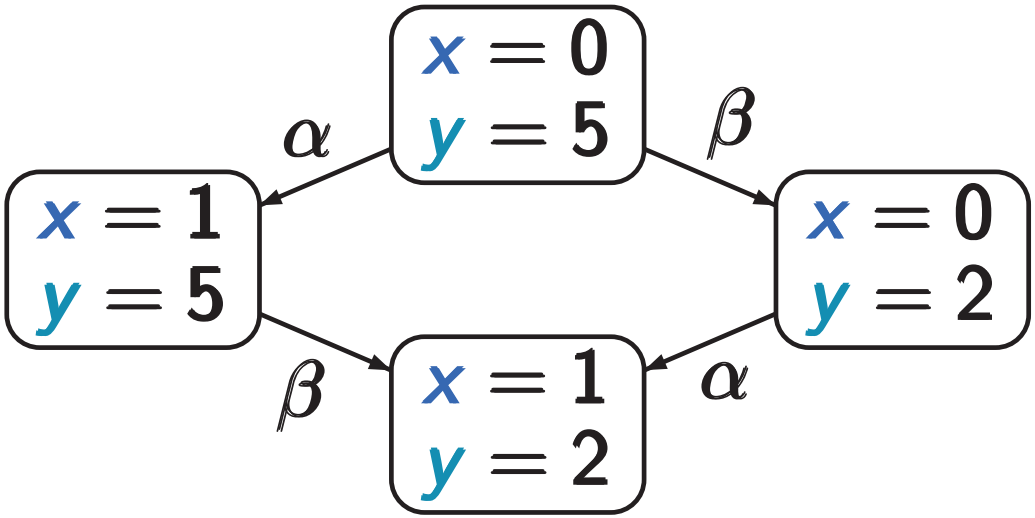
parallel execution of independent actions ← interleaving

e.g. $\underbrace{x := x+1}_{\text{action } \alpha} \;|||\; \underbrace{y := y-3}_{\text{action } \beta}$ α, β independent

parallel execution of dependent actions ← competition

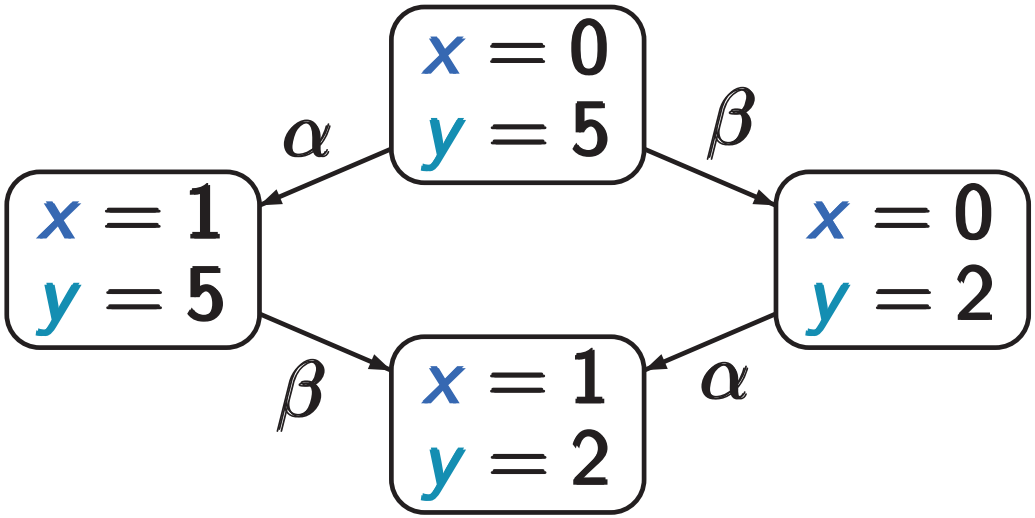
e.g. $\underbrace{x := x+1}_{\text{action } \alpha} \;|||\; \underbrace{y := 2*x}_{\text{action } \beta}$ α, β dependent

parallel execution of independent actions ← interleaving



$\underbrace{x := x+1}_{\text{action } \alpha} \;|||\; \underbrace{y := y-3}_{\text{action } \beta}$

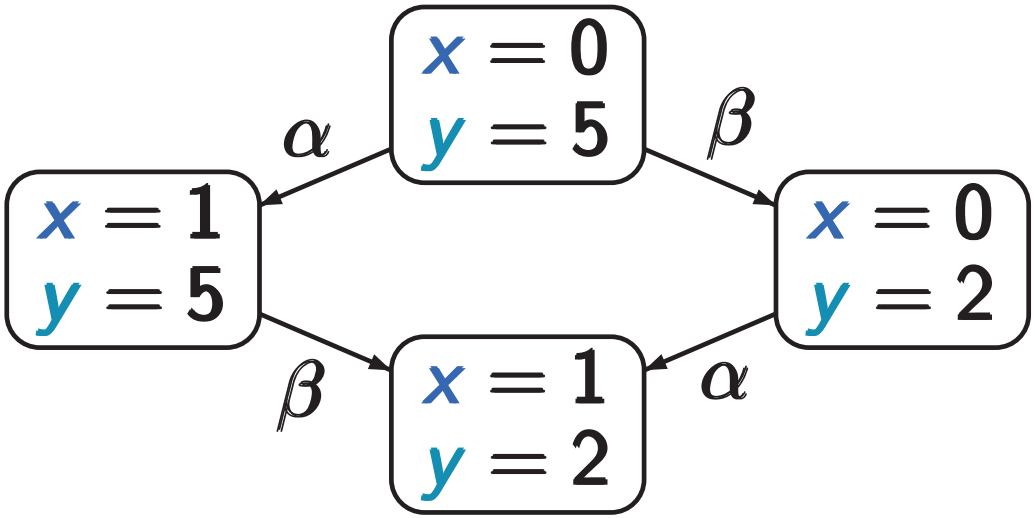
parallel execution of independent actions ← interleaving



$\underbrace{x := x+1}_{\text{action } \alpha} \;|||\; \underbrace{y := y-3}_{\text{action } \beta}$

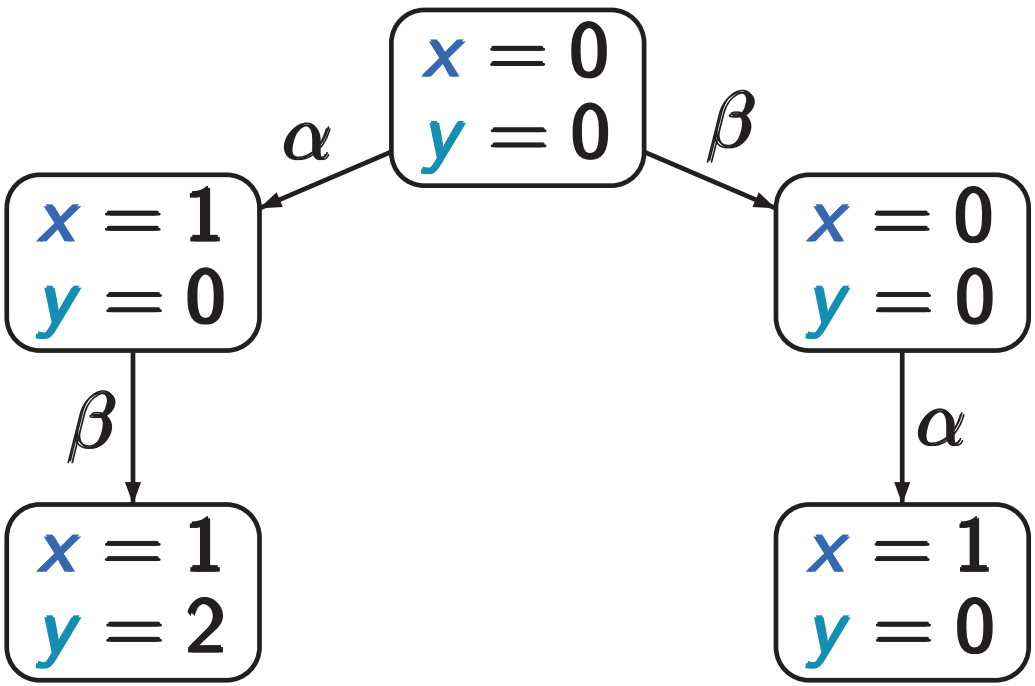
parallel execution of dependent actions ← competition

parallel execution of independent actions ← interleaving



$\underbrace{x := x+1}_{\text{action } \alpha} \;|||\; \underbrace{y := y-3}_{\text{action } \beta}$

parallel execution of dependent actions ← competition

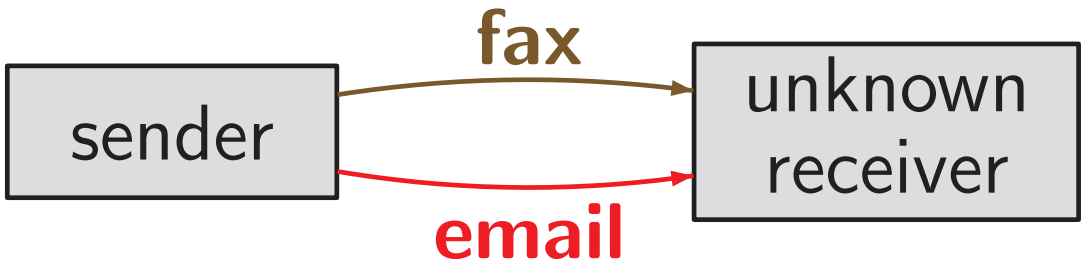


$\underbrace{x := x+1}_{\text{action } \alpha} \;|||\; \underbrace{y := 2*x}_{\text{action } \beta}$

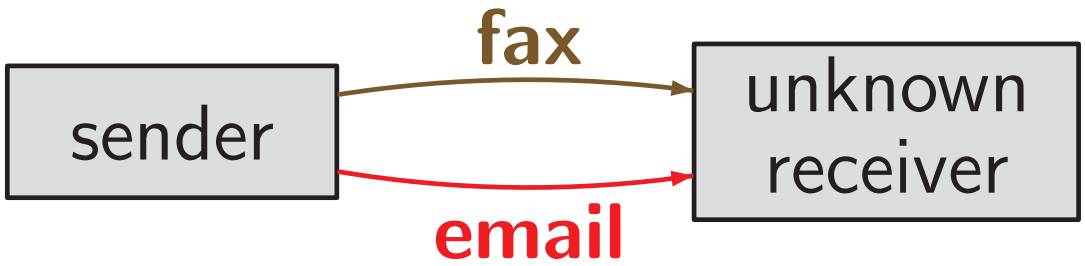
Implementation freedom

... modelled by nondeterminism

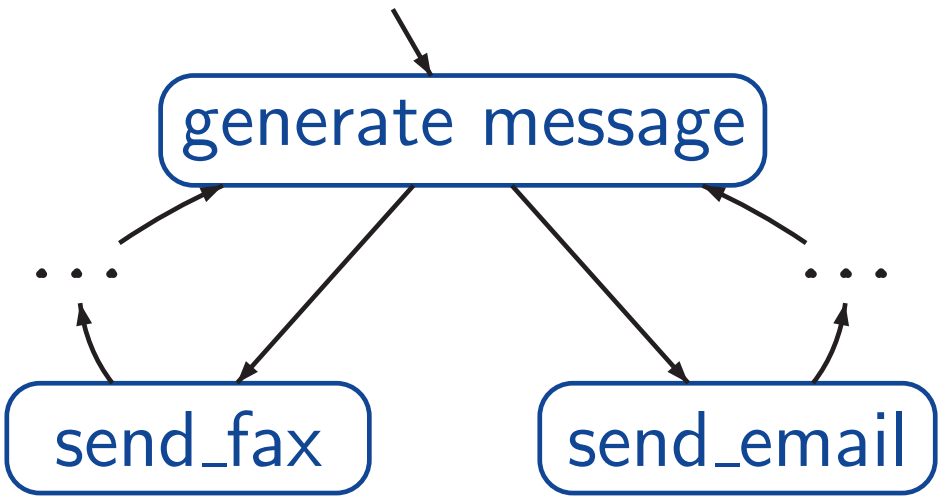
Implementation freedom



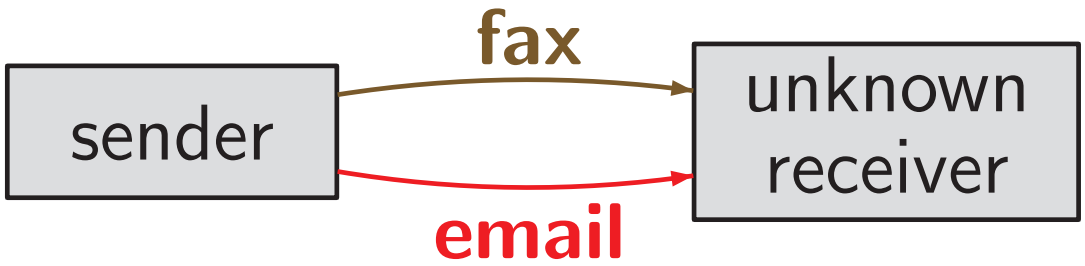
Implementation freedom



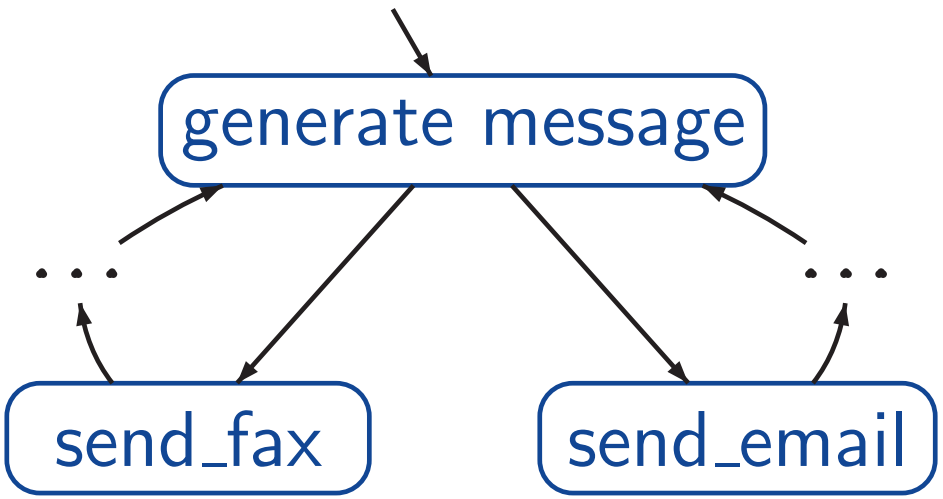
realization by a TS:



Implementation freedom

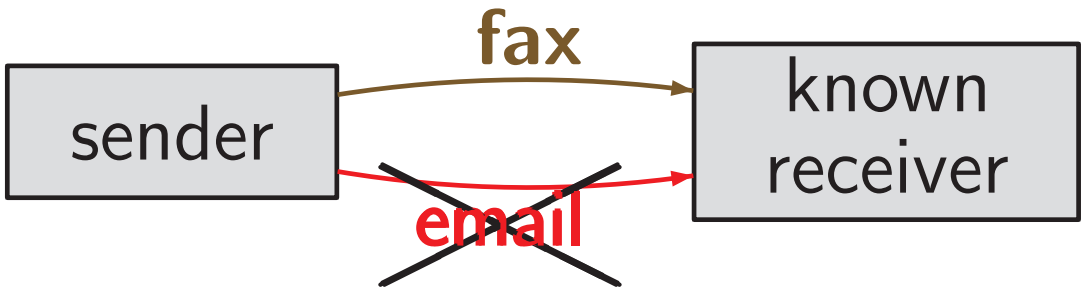


realization by a TS:



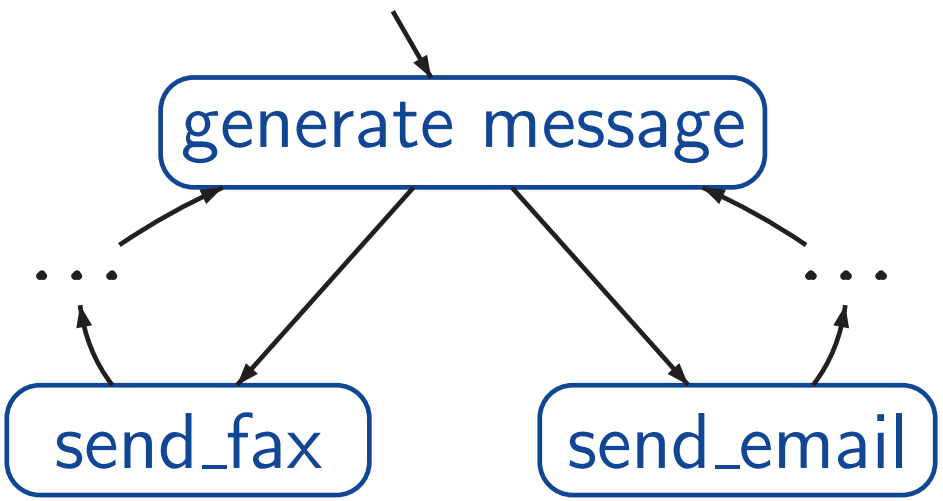
at a future refinement step the nondeterminism is replaced with one of the alternatives

Implementation freedom



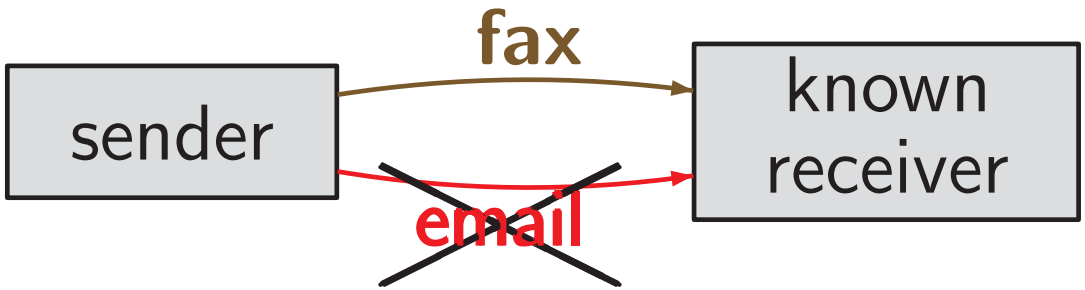
without
email access

realization by a TS:



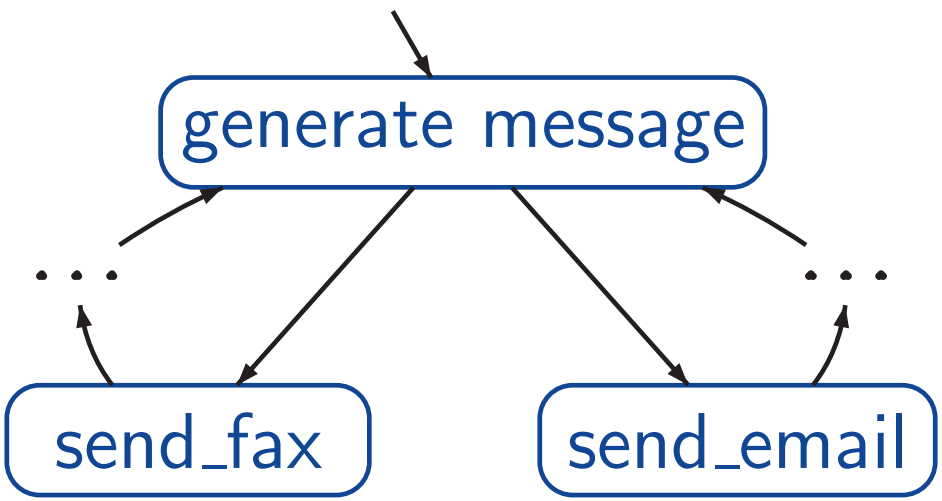
at a future refinement step the nondeterminism
is replaced with one of the alternatives

Implementation freedom

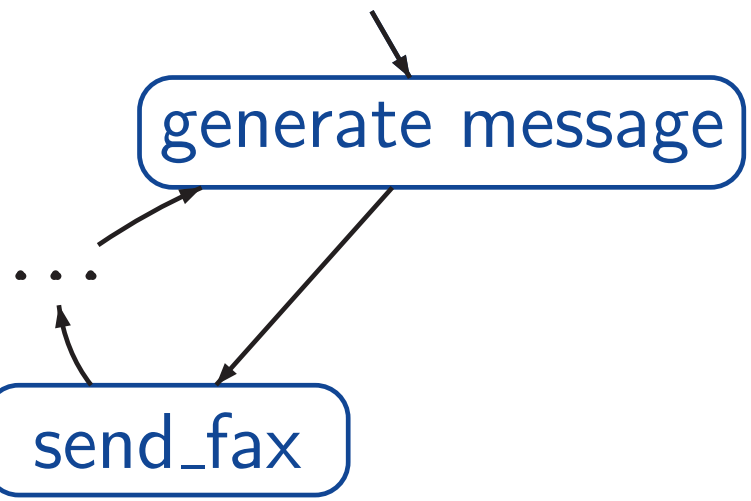


without
email access

realization by a TS:



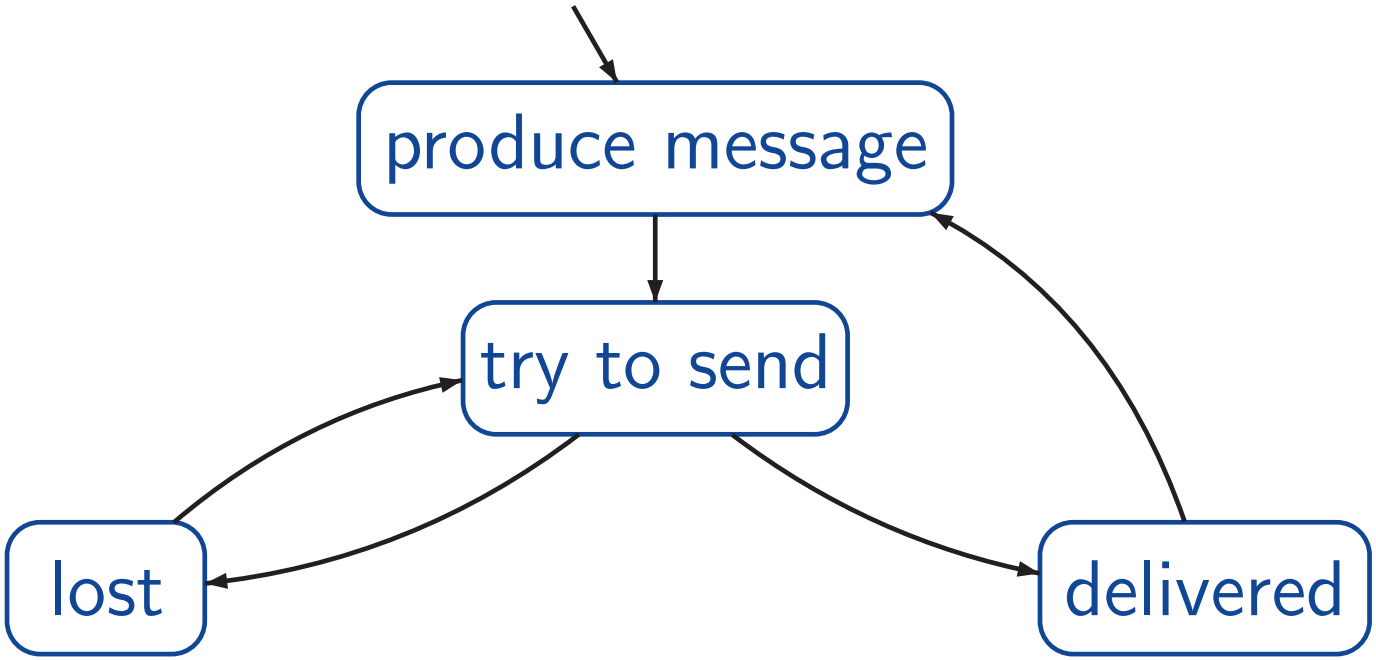
refined TS:



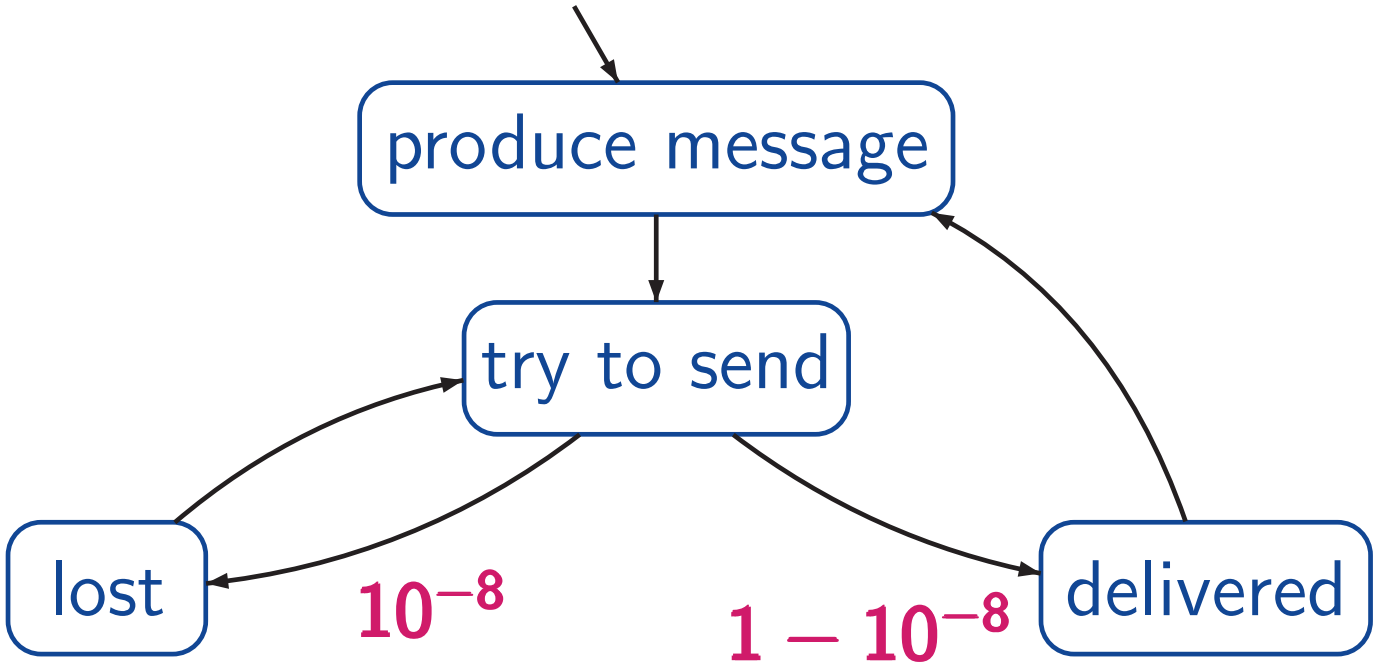
at a future refinement step the nondeterminism
is replaced with one of the alternatives

Underspecification

Underspecification

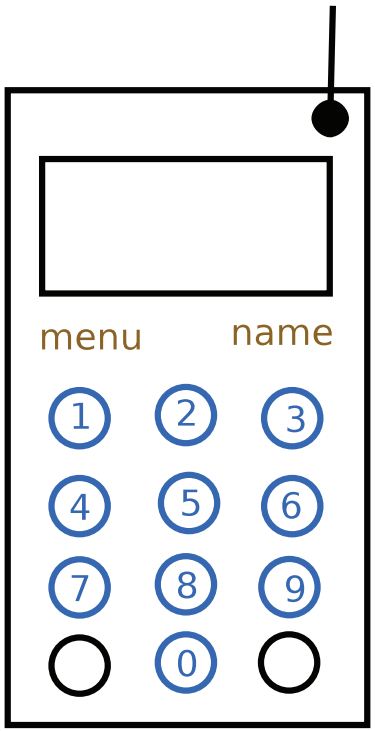


Underspecification

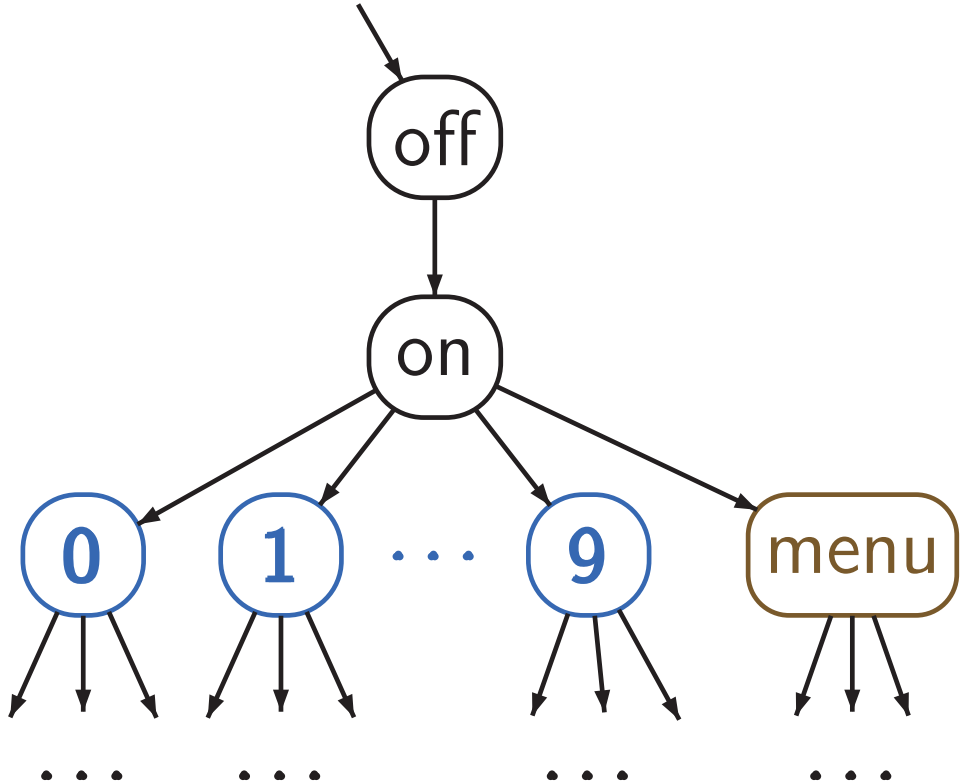


at a future refinement step the nondeterminism is replaced with probabilism

Incomplete information on the environment

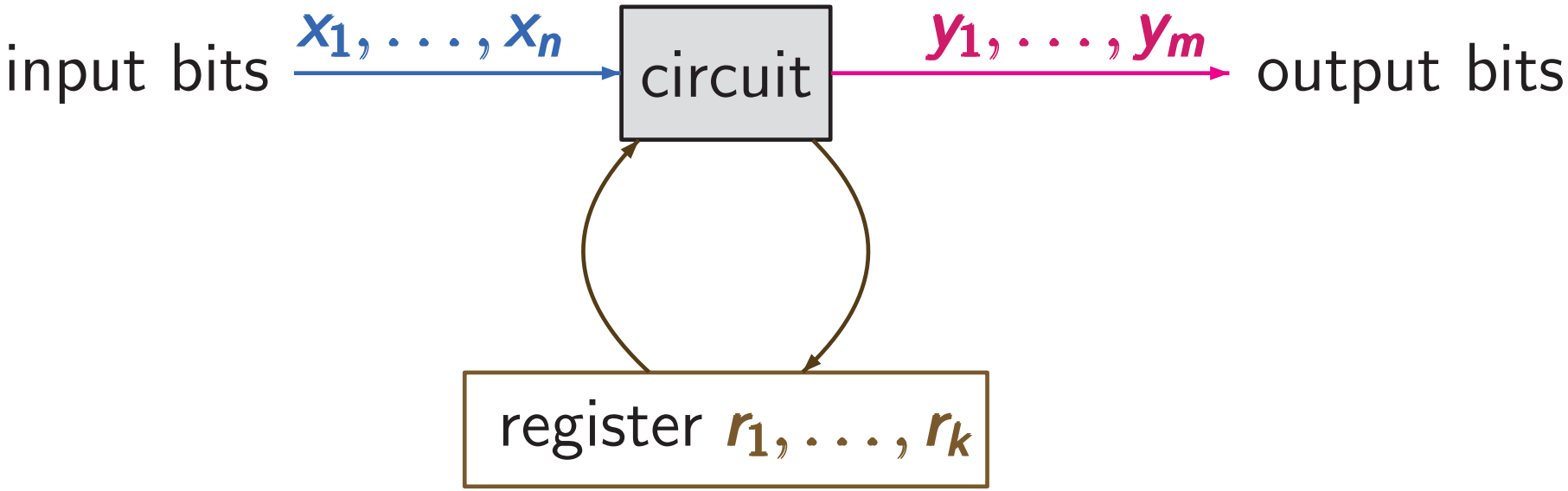


mobile phone

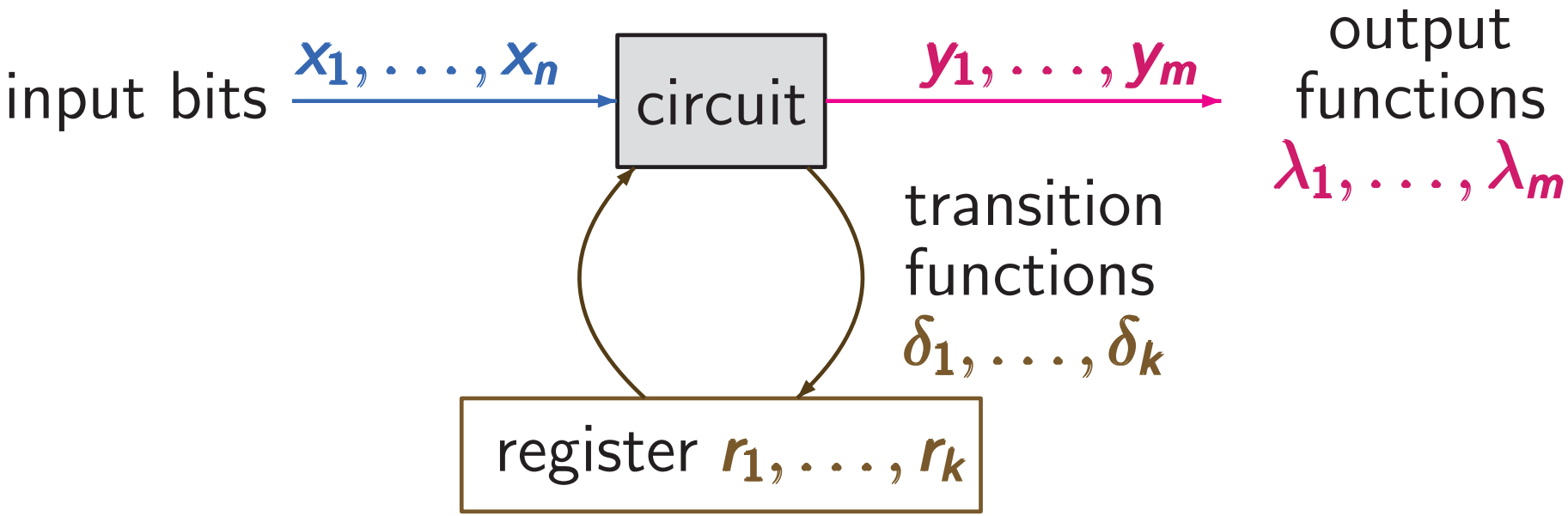


resolution of the nondeterministic choices
by a human user

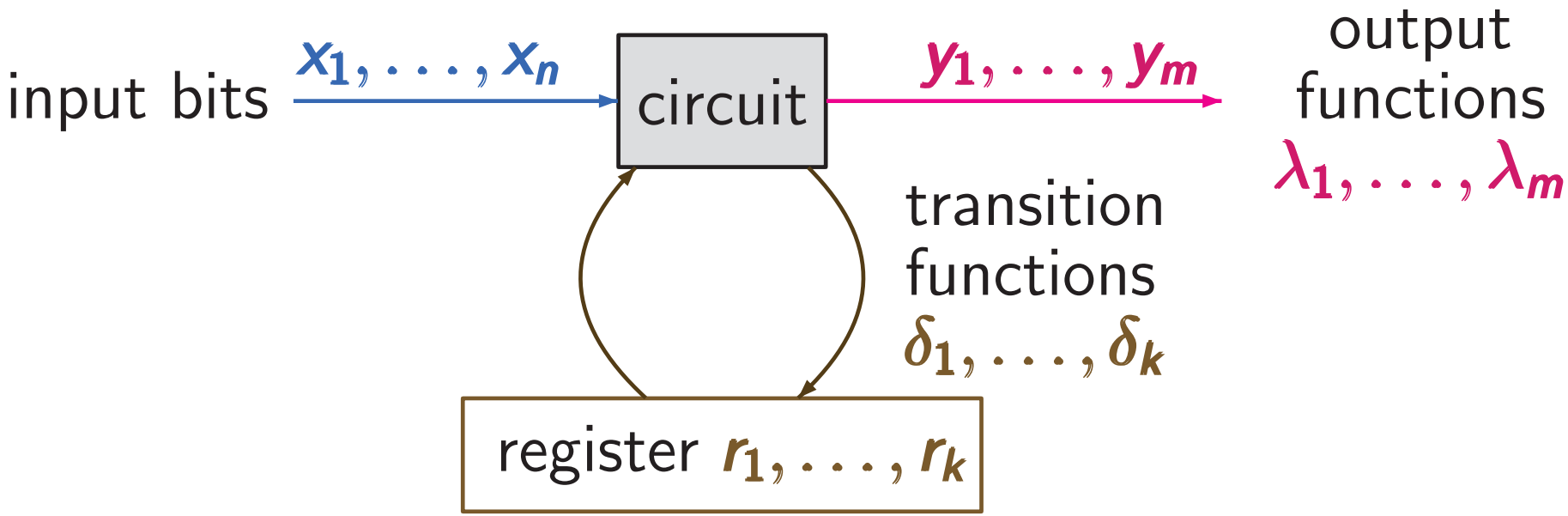
Modelling of sequential circuits by TS



Modelling of sequential circuits by TS

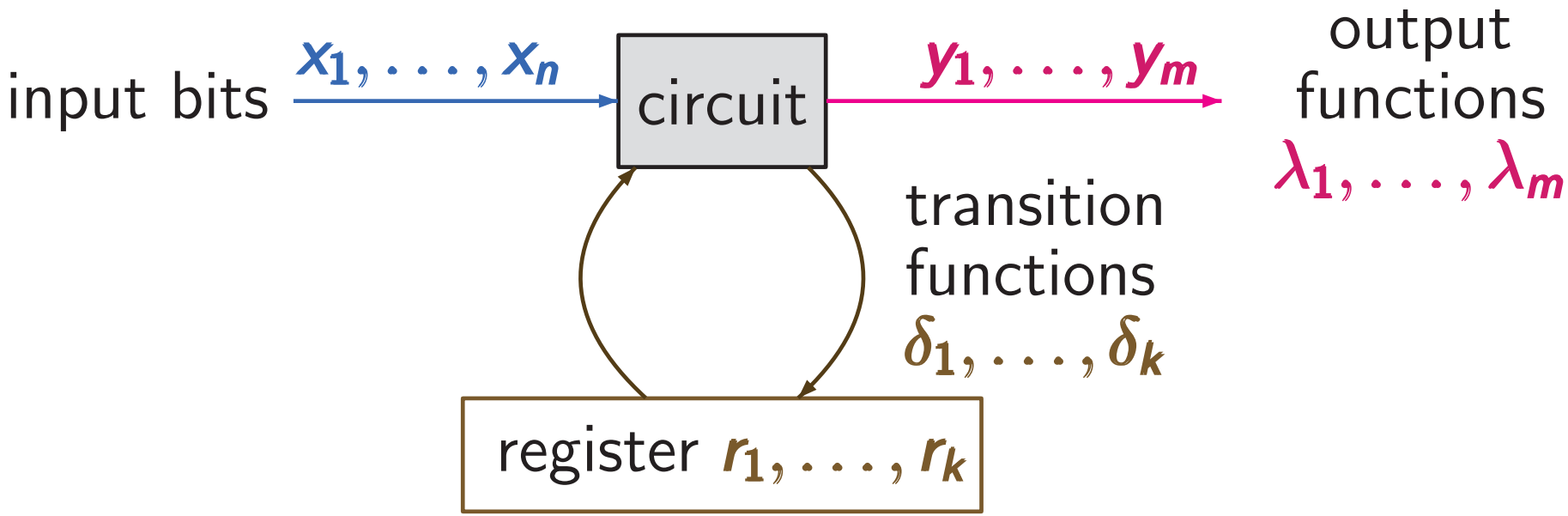


Modelling of sequential circuits by TS



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

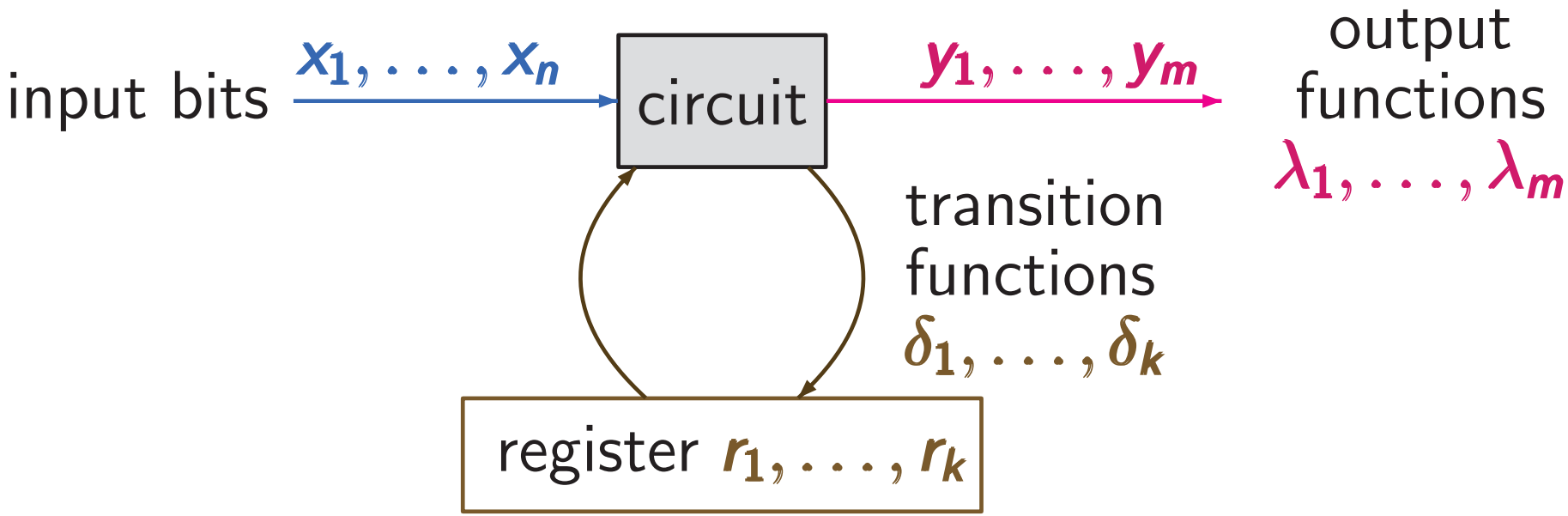
Modelling of sequential circuits by TS



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

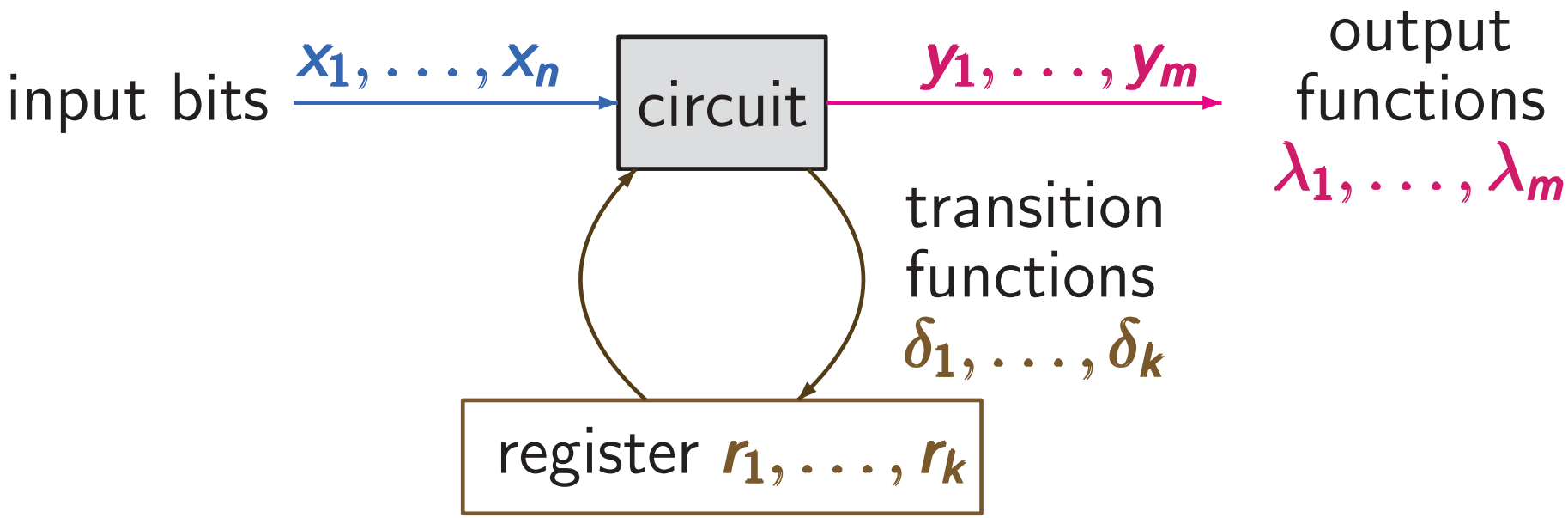
input values $a_1, \ldots, a_n$ for the input variables	$\longmapsto$	output value $\lambda_i(\ldots)$ for output variable y_i
$+$ current values $c_1, \ldots, c_k$ of the registers		next value $\delta_j(\ldots)$ for register r_j

Modelling of sequential circuits by TS



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

Modelling of sequential circuits by TS

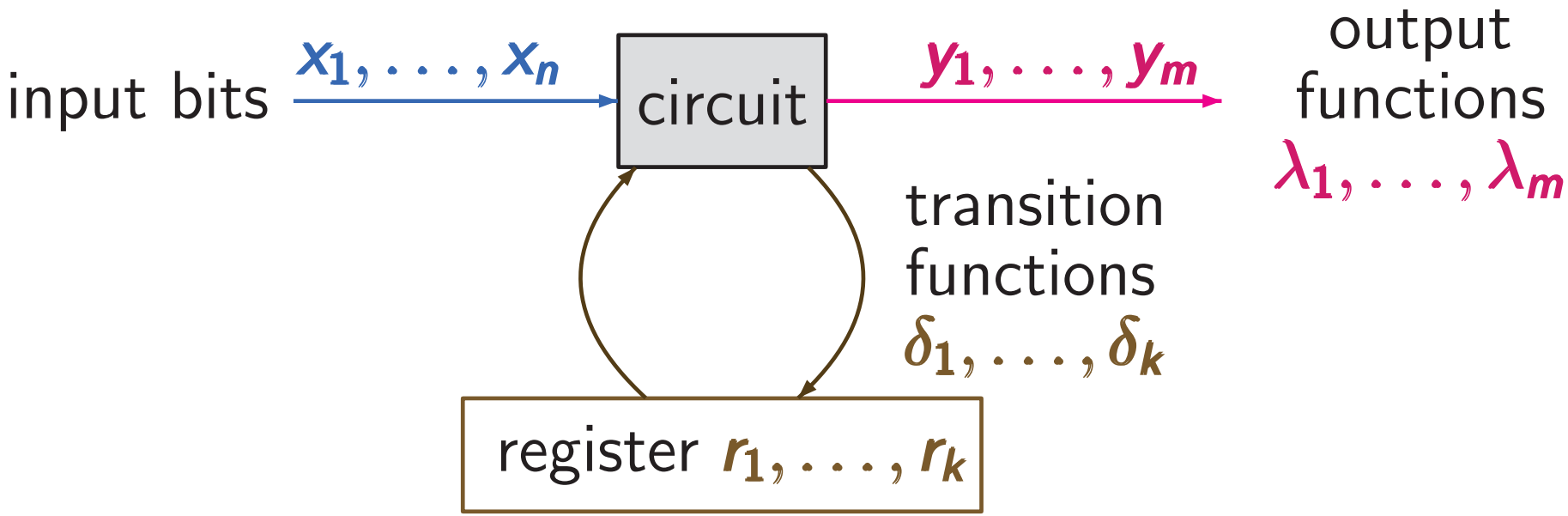


initial register evaluation $[r_1 = c_{01}, \ldots, r_k = c_{0k}]$

transition system:

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

Modelling of sequential circuits by TS

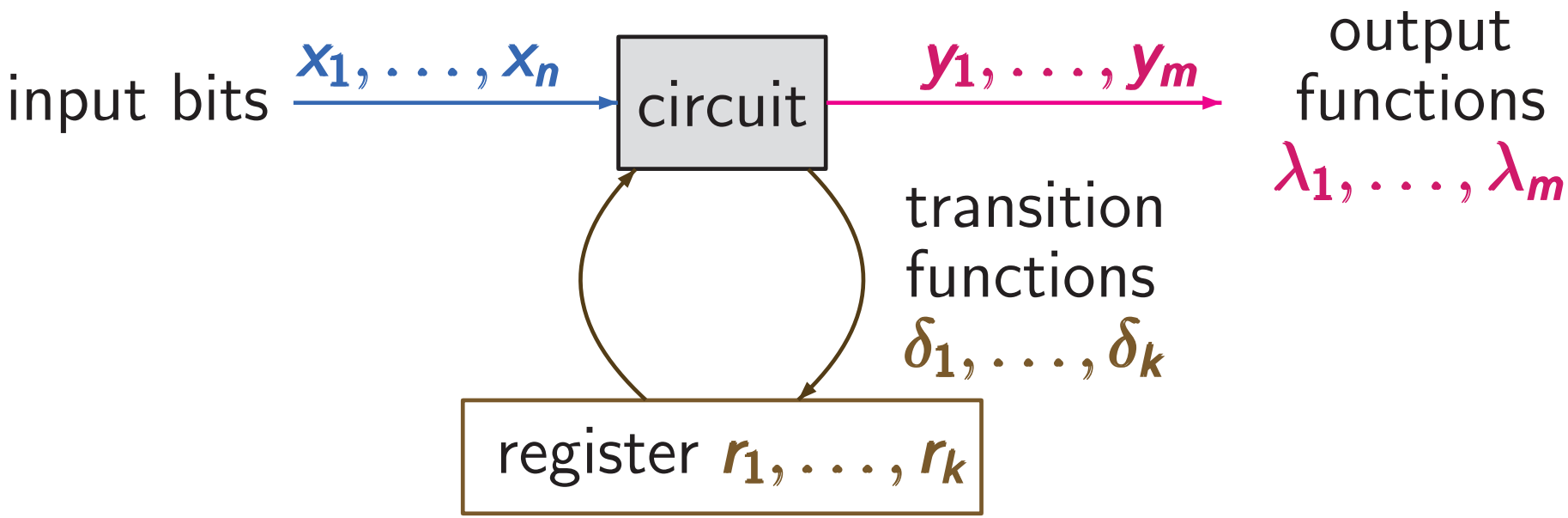


initial register evaluation $[r_1 = c_{01}, \ldots, r_k = c_{0k}]$

transition system:

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

Modelling of sequential circuits by TS

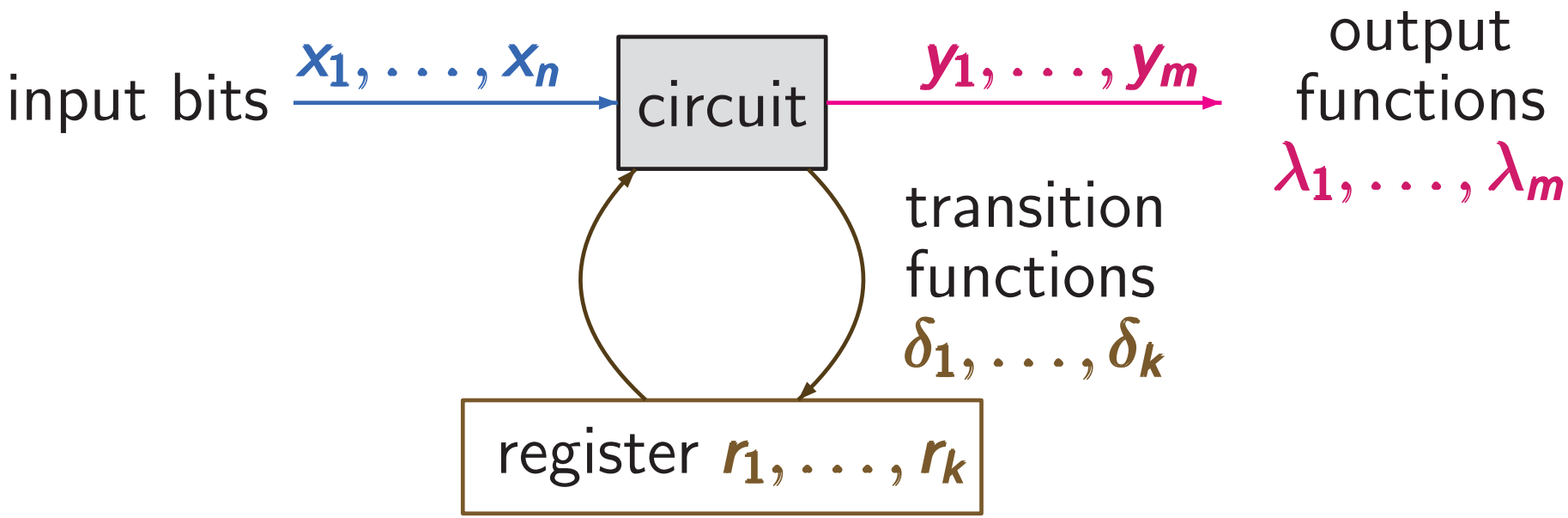


initial register evaluation $[r_1 = c_{01}, \ldots, r_k = c_{0k}]$

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

Modelling of sequential circuits by TS

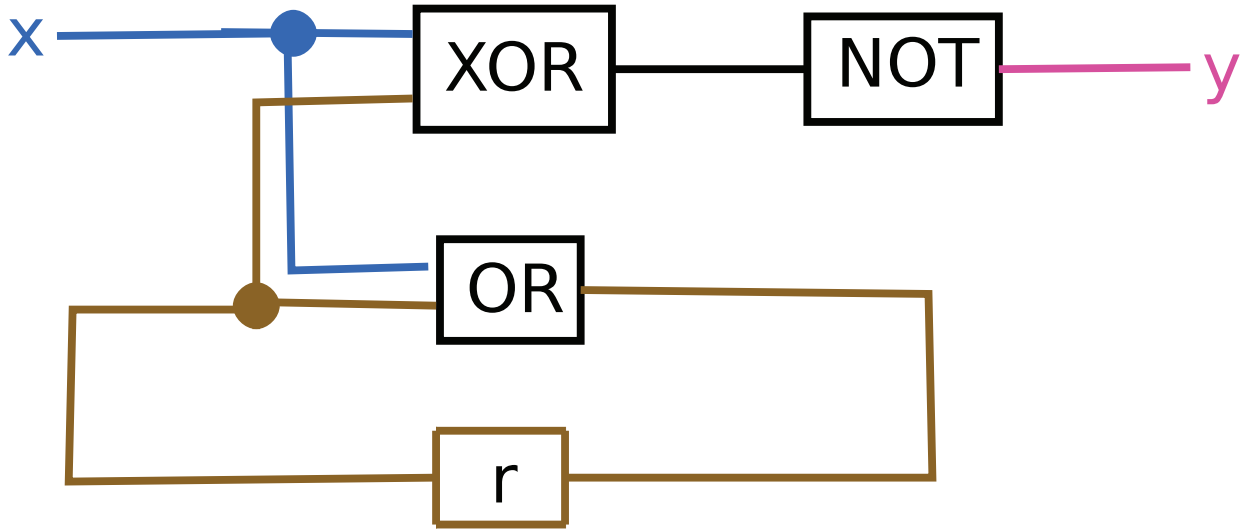


initial register evaluation $[r_1 = c_{01}, \ldots, r_k = c_{0k}]$

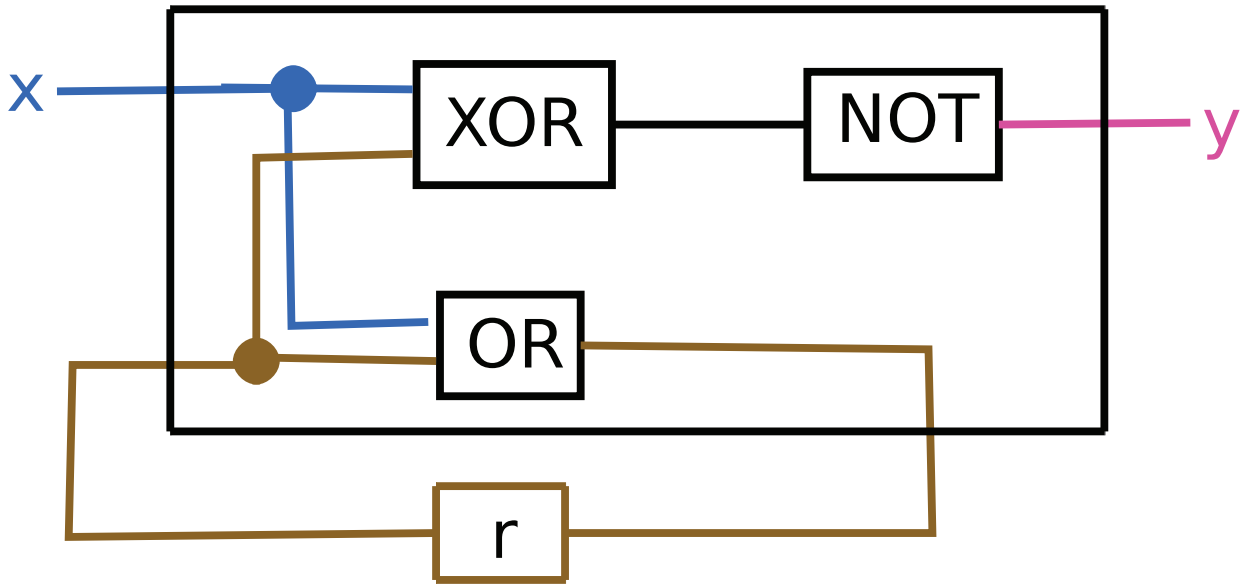
transition system:

- 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: $x_1, \ldots, x_n, y_1, \ldots, y_m, r_1, \ldots, r_k$

Example: sequential circuit



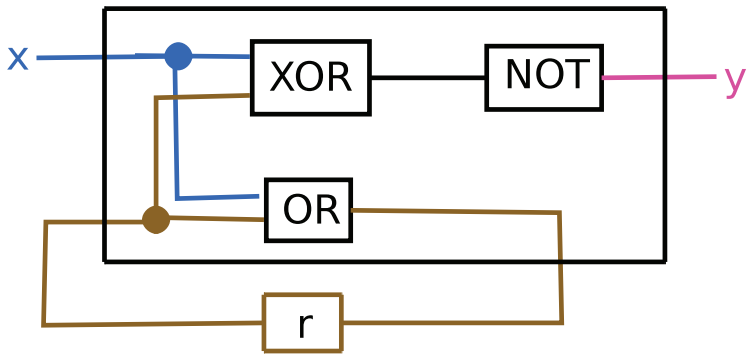
Example: sequential circuit



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

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

Example: TS for sequential circuit



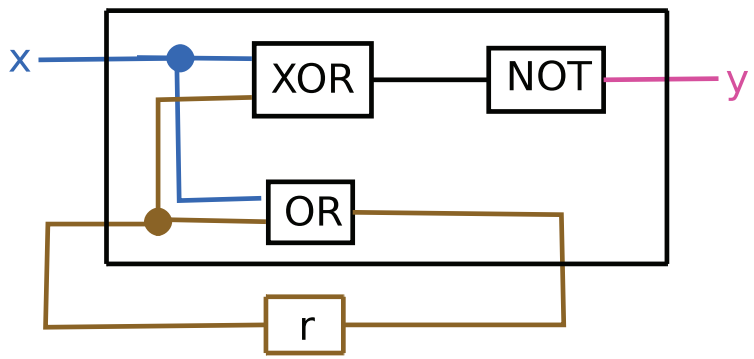
output function

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

transition function

$$\delta_r = x \vee r$$

Example: TS for sequential circuit



transition system

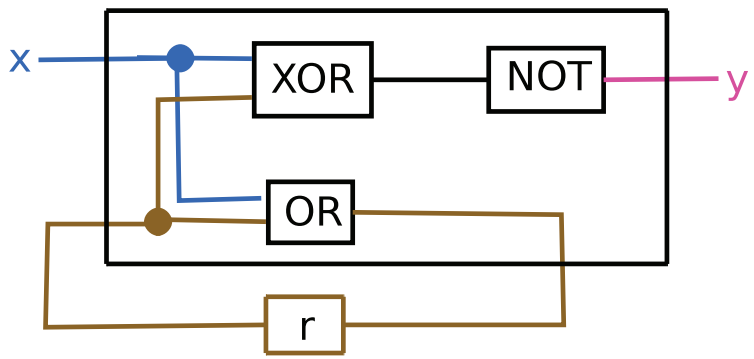
output function

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

transition function

$$\delta_r = x \vee r$$

Example: TS for sequential circuit



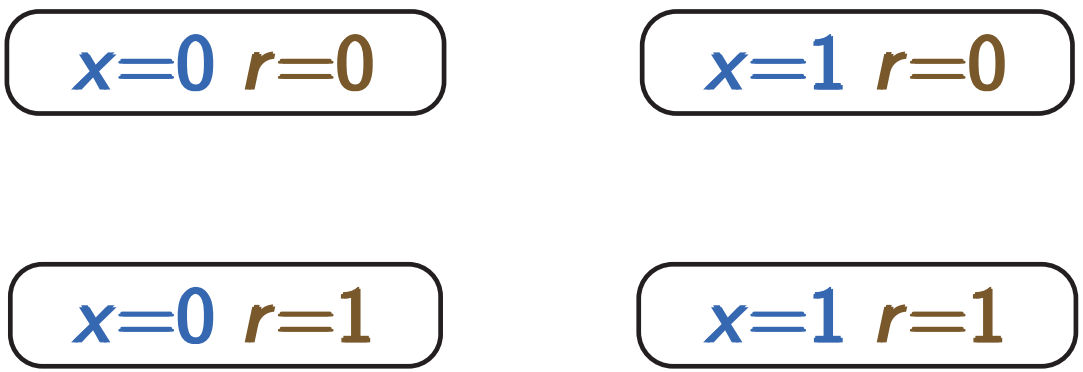
output function

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

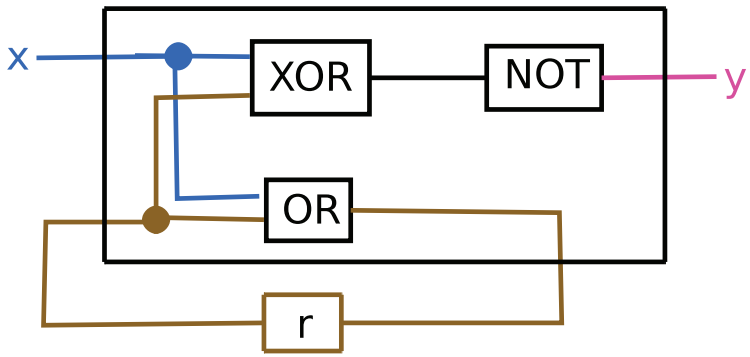
transition function

$$\delta_r = x \vee r$$

transition system



Example: TS for sequential circuit



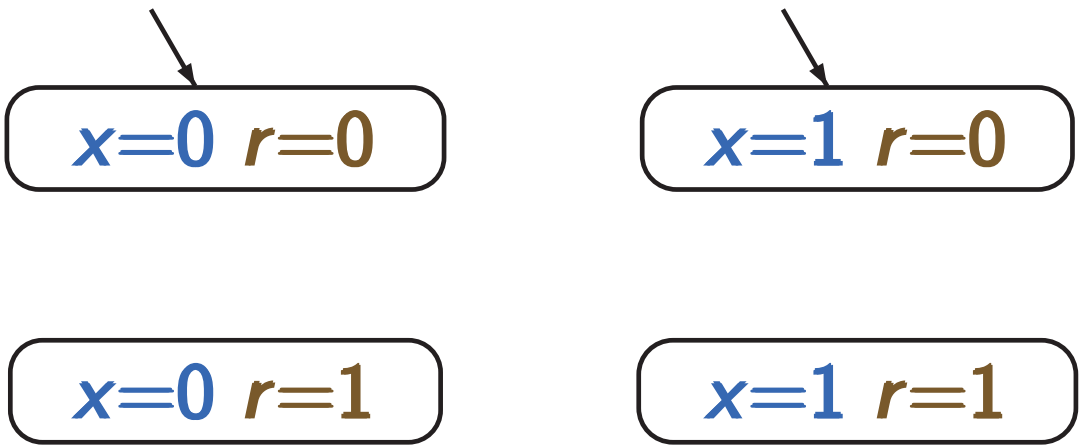
output function

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

transition function

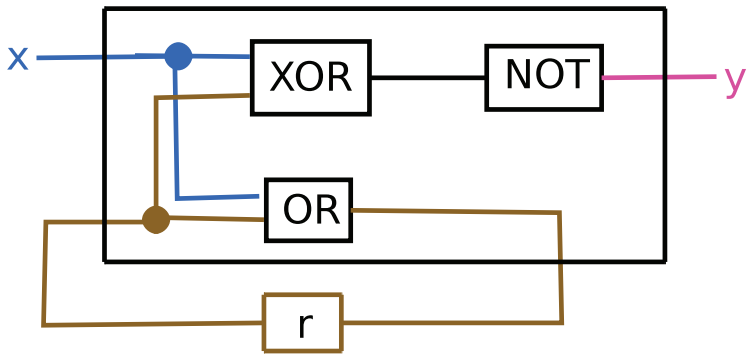
$\delta_r = x \vee r$

transition system



initial register evaluation: $r=0$

Example: TS for sequential circuit



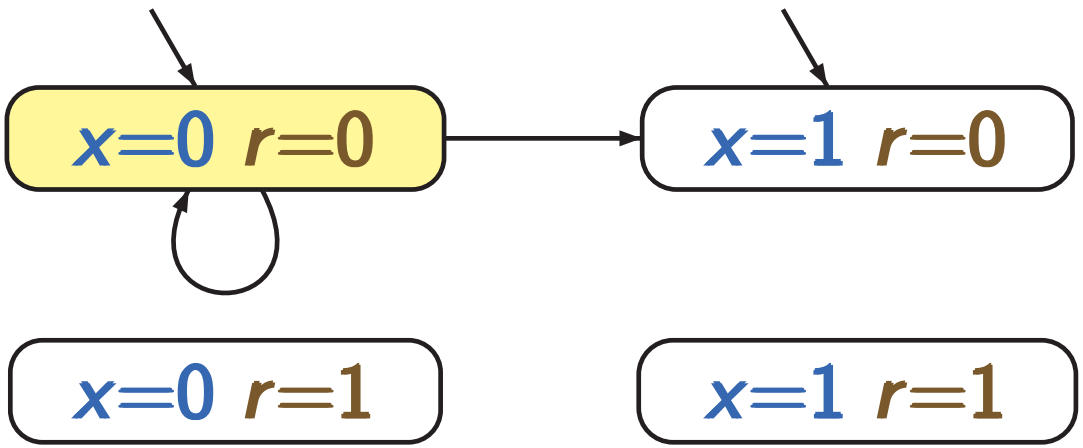
output function

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

transition function

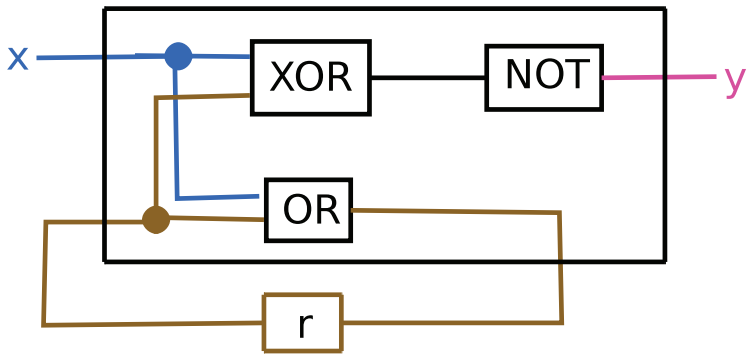
$\delta_r = x \vee r$

transition system



initial register evaluation: $r=0$

Example: TS for sequential circuit



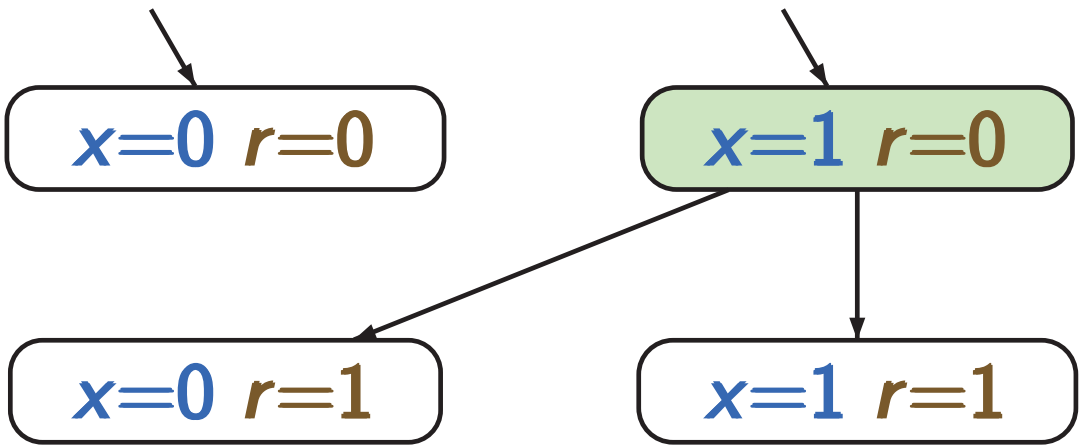
output function

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

transition function

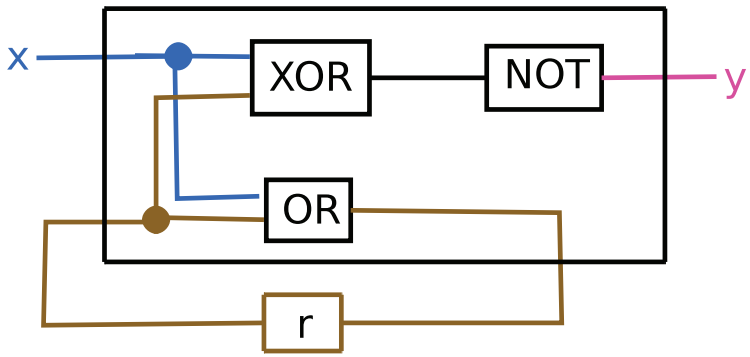
$\delta_r = x \vee r$

transition system



initial register evaluation: $r=0$

Example: TS for sequential circuit



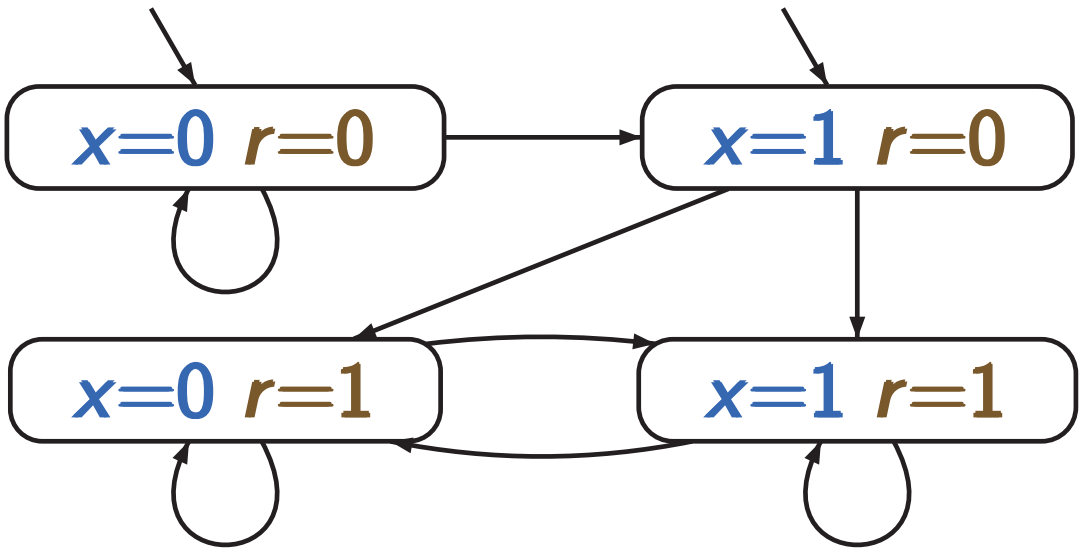
output function

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

transition function

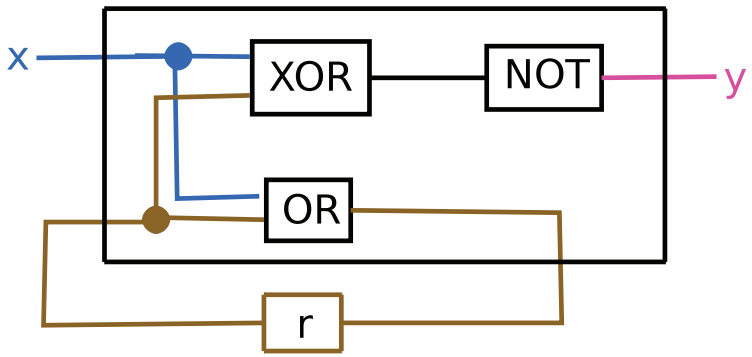
$\delta_r = x \vee r$

transition system



initial register evaluation: $r=0$

Example: TS for sequential circuit



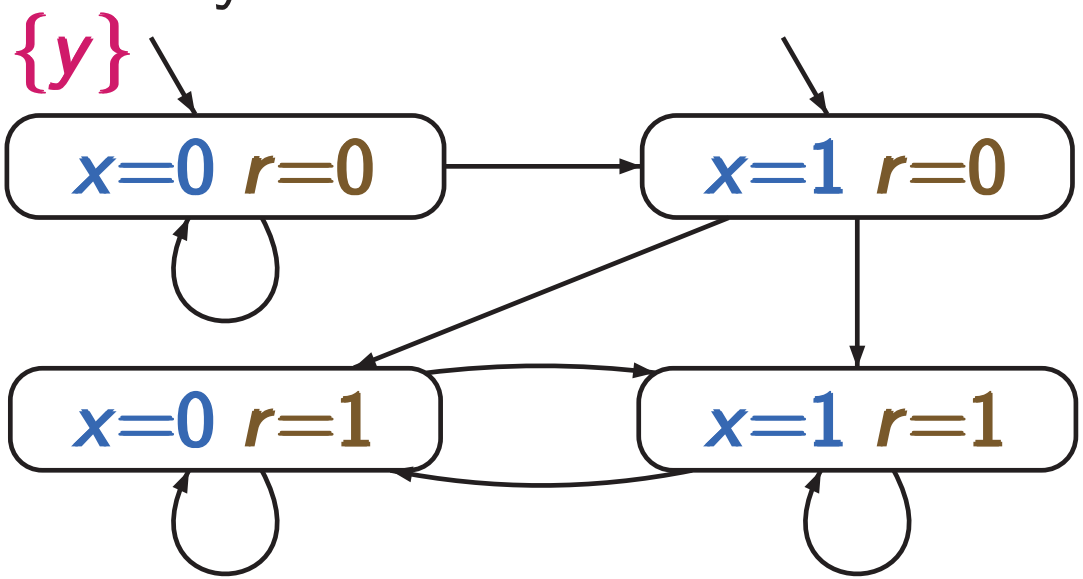
output function

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

transition function

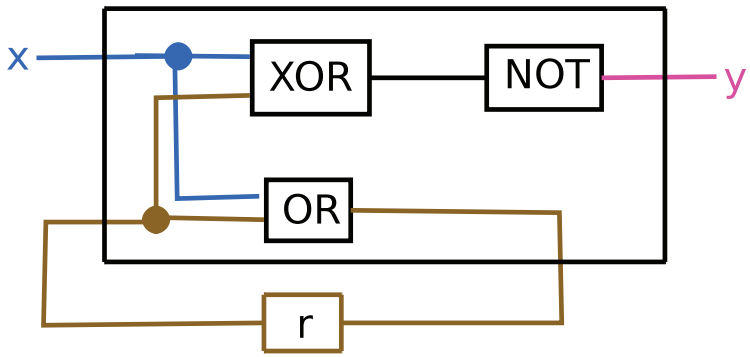
$\delta_r = x \vee r$

transition system



initial register evaluation: $r=0$

Example: TS for sequential circuit



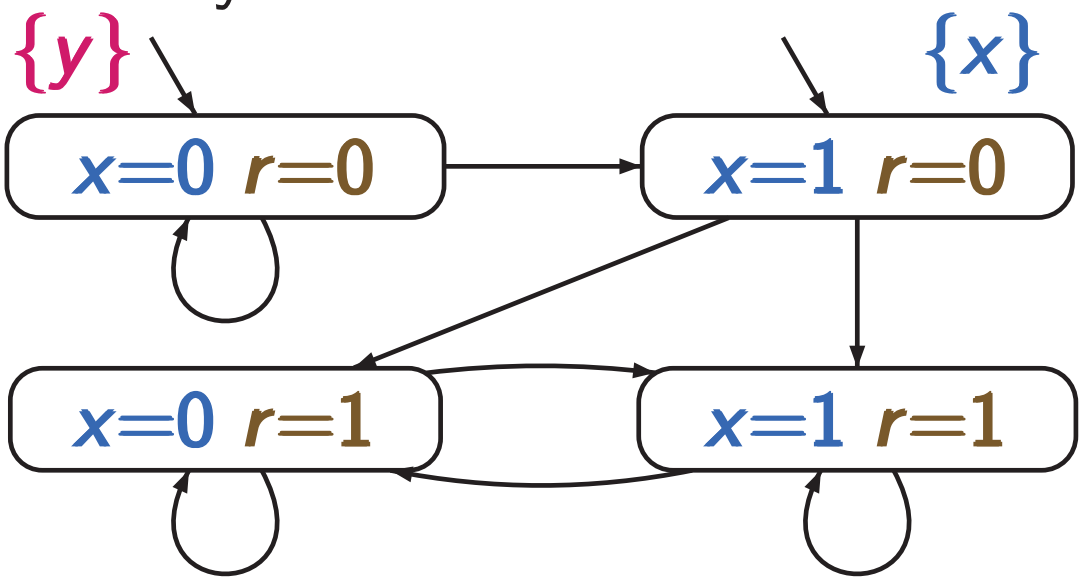
output function

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

transition function

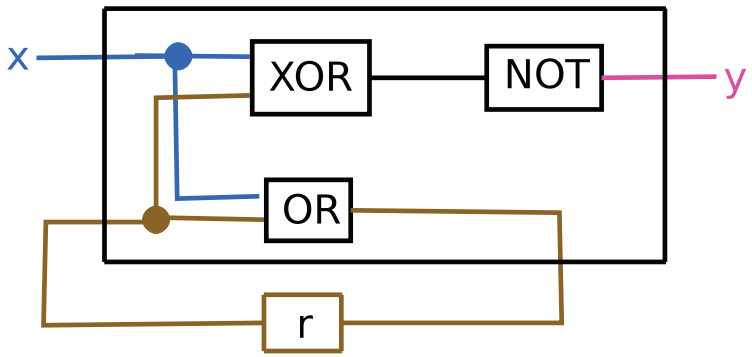
$\delta_r = x \vee r$

transition system



initial register evaluation: $r=0$

Example: TS for sequential circuit



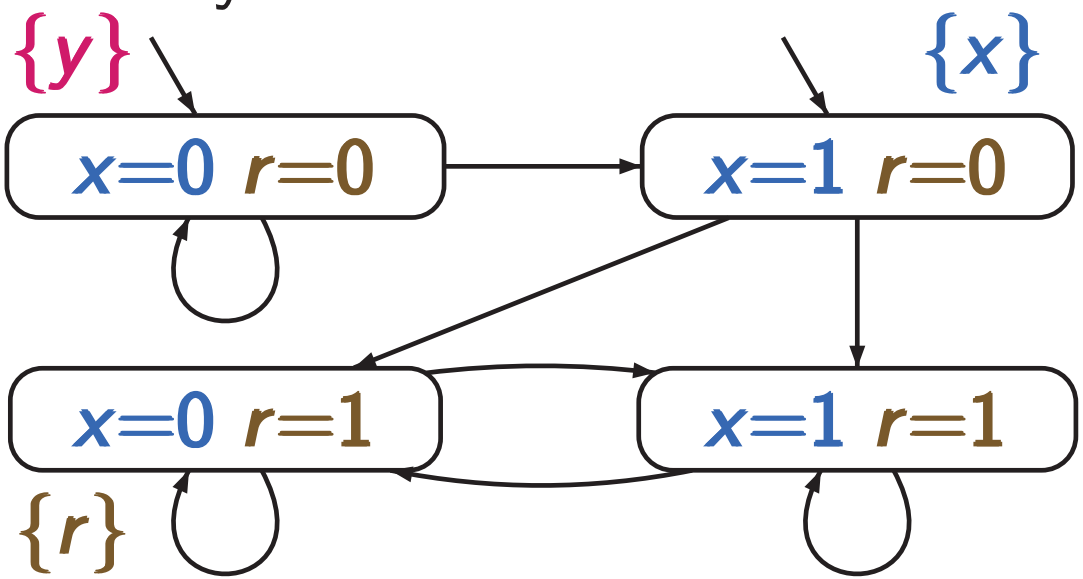
output function

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

transition function

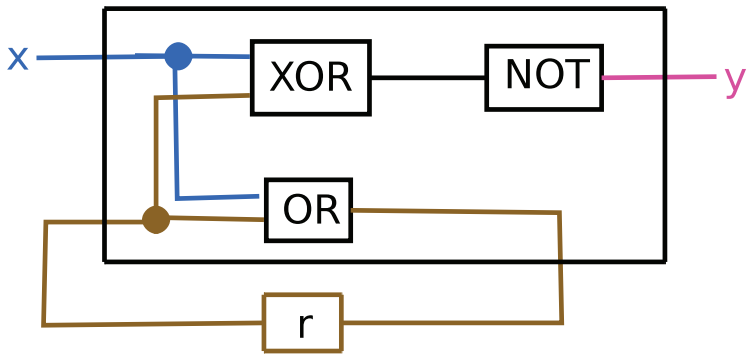
$\delta_r = x \vee r$

transition system



initial register evaluation: $r=0$

Example: TS for sequential circuit



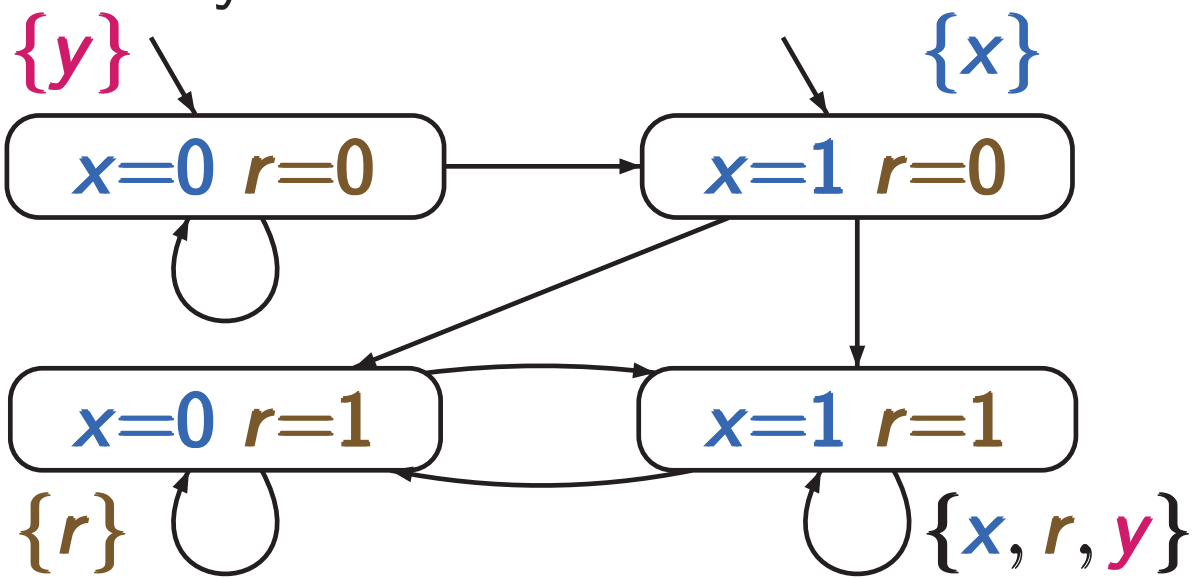
output function

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

transition function

$\delta_r = x \vee r$

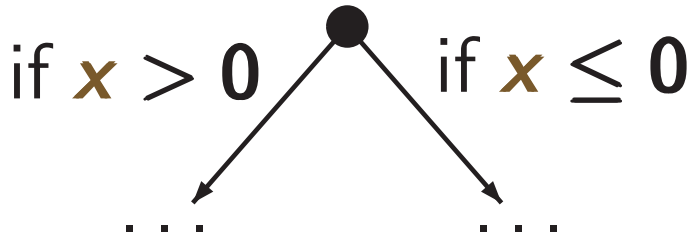
transition system



initial register evaluation: $r{=}0$

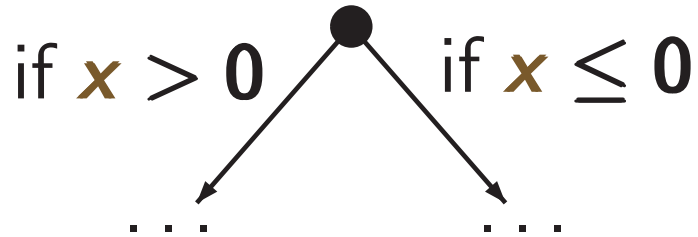
Data-dependent systems

problem: TS-representation of conditional branchings **?**



Data-dependent systems

problem: TS-representation of conditional branchings **?**

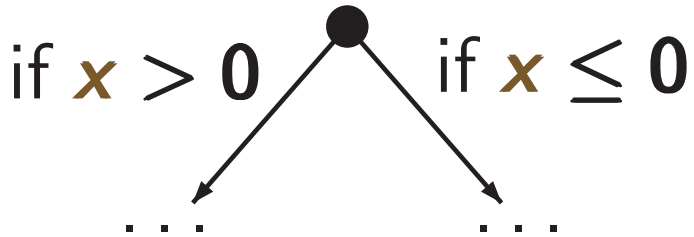


example: sequential program

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

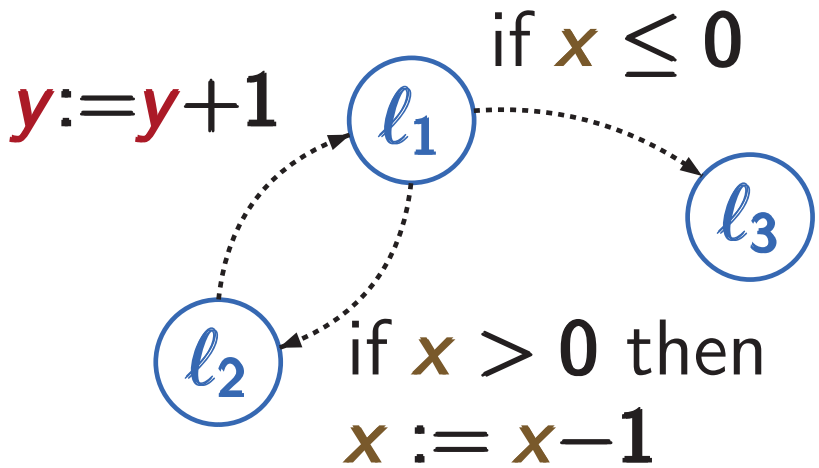
Data-dependent systems

problem: TS-representation of conditional branchings ?



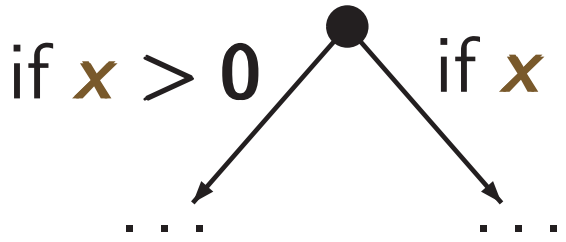
example: sequential program

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



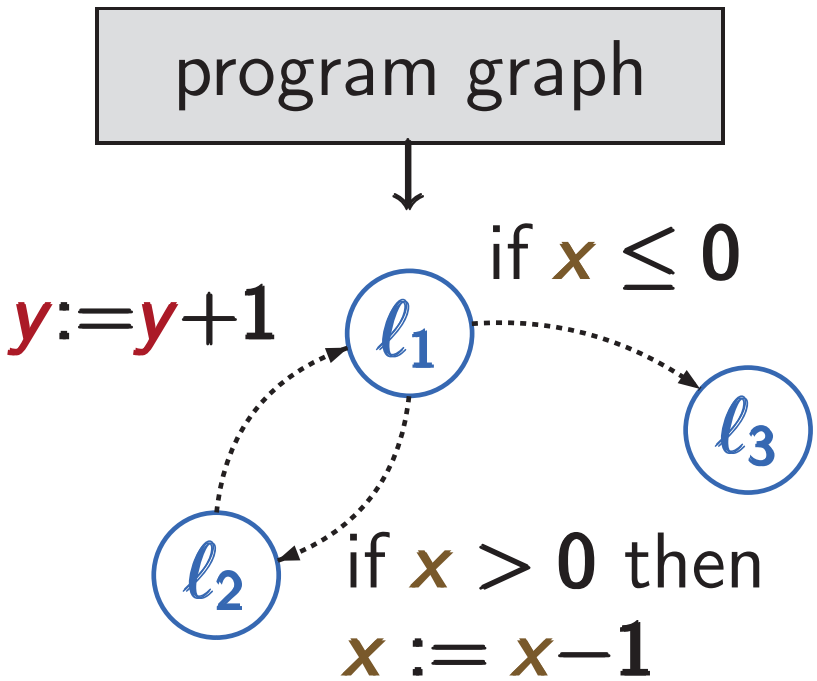
Data-dependent systems

problem: TS-representation of conditional branchings ?



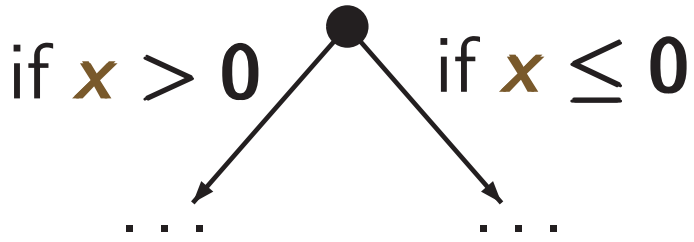
example: sequential program

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



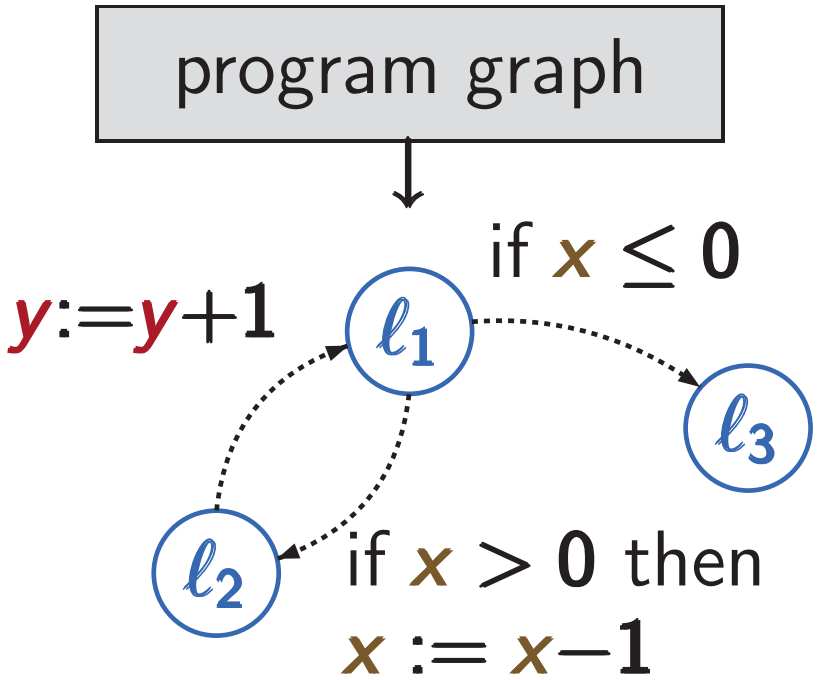
Data-dependent systems

problem: TS-representation of conditional branchings ?



example: sequential program

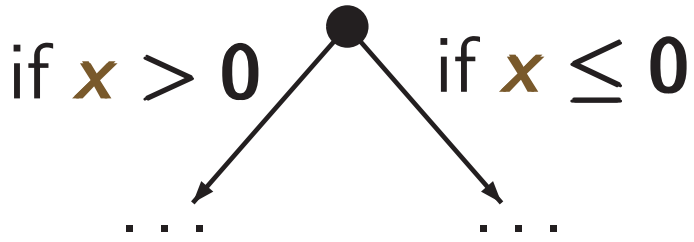
$\ell_1 \rightarrow$ WHILE $x > 0$ DO
$x := x-1$;
$\ell_2 \rightarrow$ $y := y+1$
OD
$\ell_3 \rightarrow$ ...



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

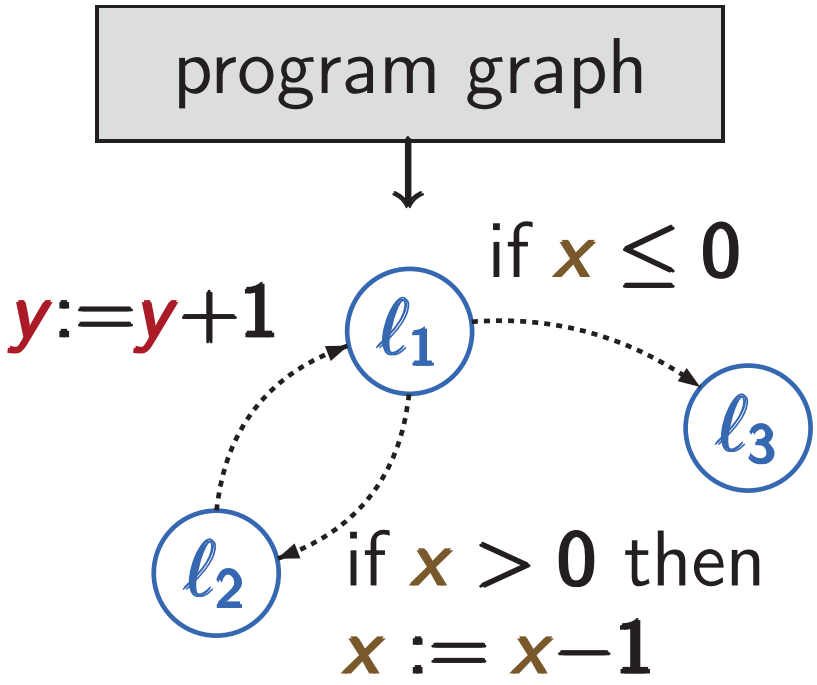
Data-dependent systems

problem: TS-representation of conditional branchings ?



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)

Example: TS for sequential program

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

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

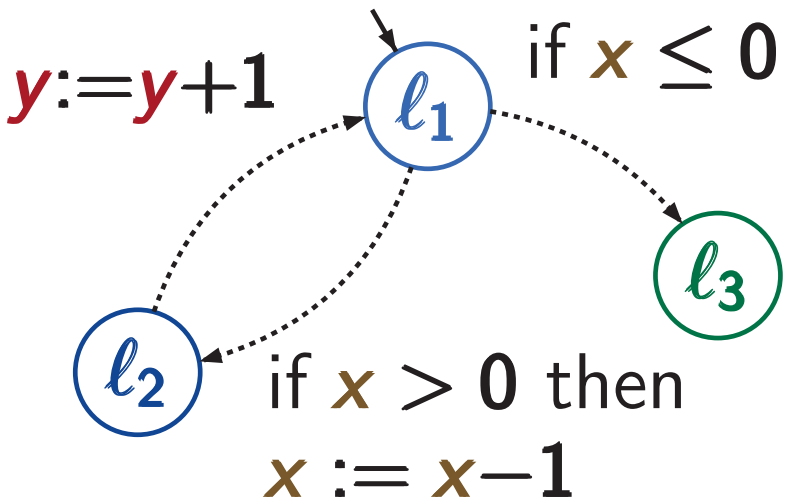
$x := x-1$

$\ell_2 \rightarrow$ $y := y+1$

OD

$\ell_3 \rightarrow$ ...

program graph



Example: TS for sequential program

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

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

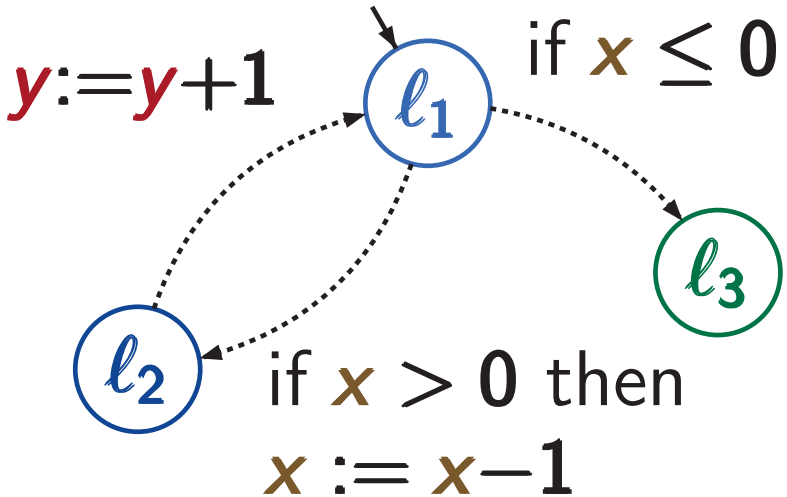
$x := x-1$

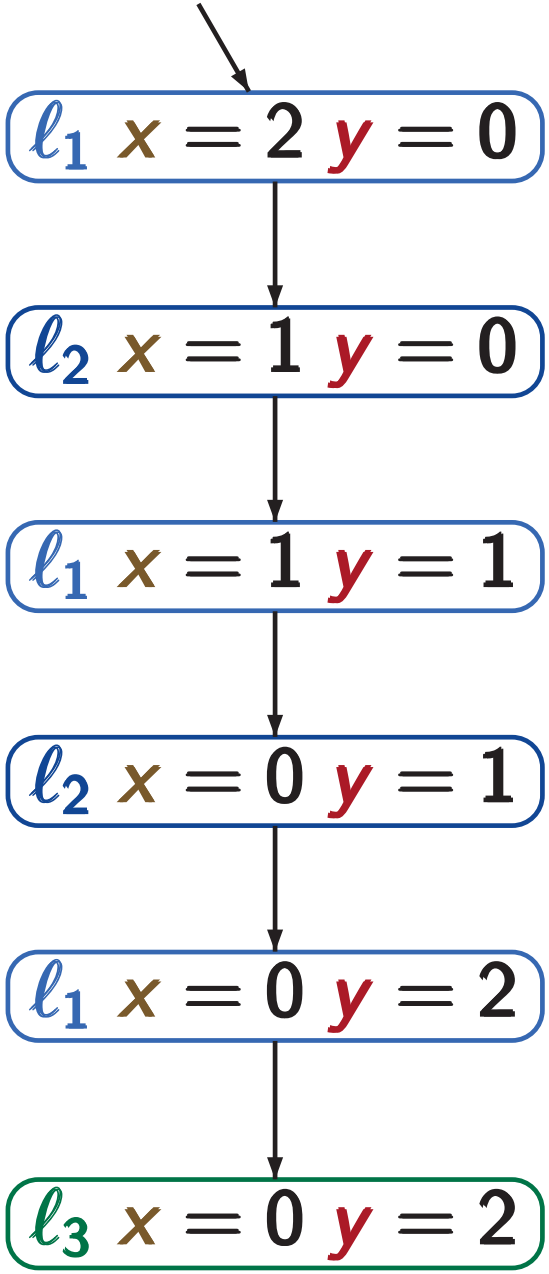
$\ell_2 \rightarrow$ $y := y+1$

OD

$\ell_3 \rightarrow$ ...

program graph





Example: TS for sequential program

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