Transition systems \(\cong\) extended digraphs

The semantic model yields a formal representation of:

- the states of the system \(\leftarrow\) nodes
- the stepwise behaviour \(\leftarrow\) transitions
- the initial states
- additional information on
  - communication \(\leftarrow\) actions
  - state properties \(\leftarrow\) atomic proposition
Transition system (TS)

A transition system is a tuple

\[ \mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L) \]

- \( S \) is the state space, i.e., set of states,
- \( \text{Act} \) is a set of actions,
- \( \rightarrow \subseteq S \times \text{Act} \times S \) is the transition relation, i.e., transitions have the form \( s \xrightarrow{\alpha} s' \) where \( s, s' \in S \) and \( \alpha \in \text{Act} \)
- \( S_0 \subseteq S \) the set of initial states,
- \( AP \) a set of atomic propositions,
- \( L : S \rightarrow 2^{AP} \) the labeling function
Transition system for beverage machine

Diagram:
- Pay
- Select
- Coke
- Sprite

Transition arrows from:
- Pay to Select
- Select to Coke
- Select to Sprite
Transition system for beverage machine

state space \( S = \{\text{pay, select, coke, sprite}\} \)
set of initial states: \( S_0 = \{\text{pay}\} \)
Transition system for beverage machine

state space \( S = \{ \text{pay, select, coke, sprite} \} \)

set of initial states: \( S_0 = \{ \text{pay} \} \)
Transition system for beverage machine

state space \( S = \{ \text{pay}, \text{select}, \text{coke}, \text{sprite} \} \)

set of initial states: \( S_0 = \{ \text{pay} \} \)

set of atomic propositions: \( AP = \{ \text{pay}, \text{drink} \} \)

labeling function:
\[
L(\text{coke}) = L(\text{sprite}) = \{ \text{drink} \}
L(\text{pay}) = \{ \text{pay} \}, \quad L(\text{select}) = \emptyset
\]
Transition system for beverage machine

actions:  
  coin  
  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 $s$
possible behaviours of a TS result from:

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

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 \( \alpha \) and put \( s := s' \)

OD

executions: maximal “transition sequences”

\[ s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots \] with \( s_0 \in S_0 \)
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 \( \alpha \) and put \( s := s' \)

OD

evaluations: 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:

\( \text{Reach}(T) = \) 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. \[ \begin{align*}
\text{action } \alpha & : \quad x & := x + 1 \\
\text{action } \beta & : \quad y & := y - 3
\end{align*} \]

\(\alpha, \beta \) independent

parallel execution of dependent actions
Transition system for parallel actions

parallel execution of independent actions

e.g. $x := x + 1 \ || \ || \ y := y - 3$  \hspace{1cm} \alpha, \beta \text{ independent}

parallel execution of dependent actions

e.g. $x := x + 1 \ || \ || \ y := 2 \times x$  \hspace{1cm} \alpha, \beta \text{ dependent}
Transition system for parallel actions

parallel execution of independent actions $\leftarrow$ interleaving

e.g. $x := x+1 \parallel\parallel y := y-3$  $\alpha, \beta$ independent

parallel execution of dependent actions $\leftarrow$ competition

e.g. $x := x+1 \parallel\parallel y := 2\times x$  $\alpha, \beta$ dependent
parallel execution of independent actions \[\xrightarrow{\text{interleaving}}\]

\[
\begin{align*}
\text{action } \alpha & : \quad x := x + 1 \\
\text{action } \beta & : \quad y := y - 3
\end{align*}
\]
parallel execution of independent actions \leftarrow \text{interleaving}

parallel execution of dependent actions \leftarrow \text{competition}
parallel execution of independent actions \[\xrightarrow{\text{interleaving}}\]

parallel execution of dependent actions \[\xrightarrow{\text{competition}}\]
Implementation freedom

... modelled by nondeterminism
Implementation freedom

sender

fax

unknown receiver

email
Implementation freedom

realization by a TS:

sender → fax → unknown receiver

email

generate message

send_fax   send_email
Implementation freedom

sender → fax → unknown receiver

email

realization by a TS:

generate message

send_fax

send_email

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

realization by a TS:

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

realization by a TS:

```
generate message
  ▼
   ▼
  send_fax  send_email
```

refined TS:

```
generate message
  ▼
   ▼
  send_fax
```

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

produce message

try to send

lost
delivered
Underspecification

at a future refinement step the nondeterminism is replaced with probabilism
Incomplete information on the environment

Resolution of the nondeterministic choices by a human user
Modelling of sequential circuits by TS

input bits \( x_1, \ldots, x_n \) → circuit → output bits \( y_1, \ldots, y_m \)

register \( r_1, \ldots, r_k \)
Modelling of sequential circuits by TS

- **Input bits**: \(x_1, \ldots, x_n\)
- **Circuit**
- **Register**: \(r_1, \ldots, r_k\)
- **Transition functions**: \(\delta_1, \ldots, \delta_k\)
- **Output functions**: \(\lambda_1, \ldots, \lambda_m\)
Modelling of sequential circuits by TS

\[ x_1, \ldots, x_n \rightarrow \text{circuit} \rightarrow y_1, \ldots, y_m \]

\[ \text{register } r_1, \ldots, r_k \]

\[ \delta_j, \lambda_i \equiv \text{switching functions } \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\} \]
Modelling of sequential circuits by TS

input bits $x_1, \ldots, x_n$ → circuit → output functions $y_1, \ldots, y_m$

delta_j, lambda_i \equiv \text{switching functions } \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}

input values $a_1, \ldots, a_n$

for the input variables

+ current values $c_1, \ldots, c_k$

of the registers

output value $\lambda_i(\ldots)$

for output variable $y_i$

next value $\delta_j(\ldots)$

for register $r_j$
Modelling of sequential circuits by TS

Input bits $x_1, \ldots, x_n$ → Circuit

Output functions $y_1, \ldots, y_m$

Transition functions $\delta_1, \ldots, \delta_k$

Register $r_1, \ldots, r_k$

Initial register evaluation $[r_1 = c_{01}, \ldots, r_k = c_{0k}]$
Modelling of sequential circuits by TS

input bits $x_1, \ldots, x_n$ \[\xrightarrow{\text{circuit}}\] output functions $y_1, \ldots, y_m$

transition functions $\delta_1, \ldots, \delta_k$

register $r_1, \ldots, r_k$

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

input bits $x_1, \ldots, x_n$ → circuit → output functions $y_1, \ldots, y_m$

transition functions $\delta_1, \ldots, \delta_k$

register $r_1, \ldots, r_k$

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

- **Input bits**: \( x_1, \ldots, x_n \)
- **Circuit**
- **Register**: \( r_1, \ldots, r_k \)
- **Output functions**: \( y_1, \ldots, y_m \)
- **Transition functions**: \( \delta_1, \ldots, \delta_k \)

**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

input bits $x_1, \ldots, x_n$ → circuit → output functions $y_1, \ldots, y_m$

transition functions $\delta_1, \ldots, \delta_k$

register $r_1, \ldots, r_k$

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:  \( \delta_r = x \lor r \)
Example: TS for sequential circuit

output function
\[ \lambda_y = \neg(x \oplus r) \]

transition function
\[ \delta_r = x \lor r \]
Example: TS for sequential circuit

output function
\[ \lambda_y = \neg(x \oplus r) \]

transition function
\[ \delta_r = x \lor r \]
Example: TS for sequential circuit

output function
\[
\lambda_y = \neg(x \oplus r)
\]

transition function
\[
\delta_r = x \lor r
\]

transition system

\[
\begin{align*}
x=0 & \quad r=0 \\
x=1 & \quad r=0 \\
x=0 & \quad r=1 \\
x=1 & \quad r=1
\end{align*}
\]
Example: TS for sequential circuit

output function
\[ \lambda_y = \neg(x \oplus r) \]

transition function
\[ \delta_r = x \lor r \]

transition system
- \( x=0 \) r=0
- \( x=1 \) r=0
- \( x=0 \) r=1
- \( x=1 \) r=1

initial register evaluation: \( r=0 \)
Example: TS for sequential circuit

**Output function**

\[ \lambda_y = \neg(x \oplus r) \]

**Transition function**

\[ \delta_r = x \lor r \]

**Transition system**

- Initial register evaluation: \( r = 0 \)
Example: TS for sequential circuit

output function
\[ \lambda_y = \neg(x \oplus r) \]

transition function
\[ \delta_r = x \lor r \]

transition system

initial register evaluation: \( r=0 \)
Example: TS for sequential circuit

output function\n\[ \lambda_y = \neg(x \oplus r) \]

transition function\n\[ \delta_r = x \lor r \]

transition system

initial register evaluation: \( r=0 \)
Example: TS for sequential circuit

transition system

\{y\}

\[
\begin{align*}
\text{x=0 } & \quad \text{r=0} \\
\text{x=1 } & \quad \text{r=0} \\
\text{x=0 } & \quad \text{r=1} \\
\text{x=1 } & \quad \text{r=1}
\end{align*}
\]

output function

\[
\lambda_y = \neg (x \oplus r)
\]

transition function

\[
\delta_r = x \lor r
\]

initial register evaluation: \( r=0 \)
Example: TS for sequential circuit

Output function
\[ \lambda_y = \neg(x \oplus r) \]

Transition function
\[ \delta_r = x \lor r \]

Transition system

Initial register evaluation: \( r = 0 \)
Example: TS for sequential circuit

output function
\[ \lambda_y = \neg (x \oplus r) \]

transition function
\[ \delta_r = x \lor r \]

transition system

initial register evaluation: \( r = 0 \)
Example: TS for sequential circuit

output function
\[ \lambda_y = \neg(x \oplus r) \]

transition function
\[ \delta_r = x \lor r \]

transition system

initial register evaluation: \( r=0 \)
Data-dependent systems

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

\[
\text{if } x > 0 \quad \text{if } x \leq 0
\]

\[
\ldots \quad \text{...}
\]
Data-dependent systems

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

if $x > 0$  

if $x \leq 0$

\[ \ldots \quad \ldots \]

**example:** sequential program

```plaintext
WHILE $x > 0$ DO
    $x := x - 1$;
    $y := y + 1$
OD
\[ \ldots \]
```
**Data-dependent systems**

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

if $x > 0$ \quad \quad if $x \leq 0$

\[ \ldots \quad \ldots \]

**example:** sequential program

```
WHILE $x > 0$ DO
    $x := x - 1$;
    $y := y + 1$
OD
\[ \ldots \]
```

\[ y := y + 1 \]

if $x \leq 0$

\[ l_1 \rightarrow l_2 \rightarrow l_3 \]

if $x > 0$ then

\[ x := x - 1 \]
**Data-dependent systems**

.problem: TS-representation of conditional branchings?

\[
\begin{align*}
\text{if } x &> 0 \quad \text{if } x \leq 0 \\
&\quad \ldots \quad \ldots \\
\end{align*}
\]

.example: sequential program

\[
\text{WHILE } x > 0 \text{ DO} \\
\quad x := x - 1; \\
\quad y := y + 1 \\
\text{OD} \\
\ldots
\]

program graph

\[
\begin{align*}
\text{if } x &\leq 0 \\
&y := y + 1 \\
\text{if } x &> 0 \text{ then} \\
&x := x - 1
\end{align*}
\]
**Data-dependent systems**

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

```
if x > 0

if x ≤ 0

... ...
```

**example:** sequential program

```
\[ \ell_1 \rightarrow \text{WHILE } x > 0 \text{ DO } \]

\[ x := x - 1; \]

\[ \ell_2 \rightarrow \text{OD} \]

\[ y := y + 1 \]

\[ \ell_3 \rightarrow \ldots \]
```

\[ \ell_1, \ell_2, \ell_3 \text{ are locations, i.e., control states} \]
Data-dependent systems

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

\[
\begin{align*}
\text{if } x > 0 & \quad \text{if } x \leq 0 \\
\ldots & \quad \ldots
\end{align*}
\]

**example:** sequential program

\[
\begin{align*}
\ell_1 & \rightarrow \quad \text{WHILE } x > 0 \text{ DO } \\
& \quad x := x - 1; \\
\ell_2 & \rightarrow \quad y := y + 1 \\
\ell_3 & \rightarrow \quad \ldots
\end{align*}
\]

states of the transition system:

**locations + relevant data** (*here: values for \(x\) and \(y\))
Example: TS for sequential program

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

$l_1 \rightarrow \text{WHILE } x > 0 \text{ DO}$

- $x := x - 1$

$l_2 \rightarrow \text{ y := y + 1}$

$\text{OD}$

$l_3 \rightarrow \ldots$

program graph
Example: TS for sequential program

initially: \( x = 2, y = 0 \)

\( \ell_1 \rightarrow \text{WHILE } x > 0 \text{ DO} \)

\( x := x - 1 \)

\( \ell_2 \rightarrow y := y + 1 \)

\( \ell_3 \rightarrow \ldots \)

program graph

\( y := y + 1 \) if \( x \leq 0 \)

\( \ell_2 \) if \( x > 0 \) then

\( x := x - 1 \)
Example: TS for sequential program

Initially: \( x = 2, \ y = 0 \)

\( \ell_1 \rightarrow \text{WHILE } x > 0 \text{ DO} \)

\( x := x - 1 \quad \text{← action } \alpha \)

\( \ell_2 \rightarrow \quad y := y + 1 \quad \text{← action } \beta \)

\( \ell_3 \rightarrow \ldots \)

Program graph

if \( x \leq 0 \) then \( \text{loop_exit} \)

if \( x > 0 \) then \( \alpha \)