Transition systems

A **transition system** TS is a tuple $(S, Act, \rightarrow, I, AP, L)$ where

- S is a set of states
- Act is a set of actions
- $\longrightarrow \subseteq S \times Act \times S$ is a transition relation
- $I \subseteq S$ is a set of initial states
- AP is a set of atomic propositions
- $L : S \rightarrow 2^{AP}$ is a labeling function

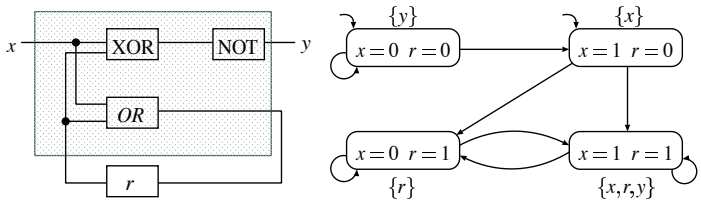
Notation: $s \xrightarrow{\alpha} s'$ for $(s, \alpha, s') \in \longrightarrow$

Transition systems $\neq$ automata

As opposed to automata, in a transition system:

- there are **no** accepting states
- set of states and actions may be **infinite**
- may have **infinite branching**
- actions may be subject to **synchronization**
- **nondeterminism** has a different role

Modeling hardware: sequential circuits



Transition system representation of a simple hardware circuit:

- **input** variable x, **output** variable y, and **register** r
- output function $\neg(x \oplus r)$
- register evaluation function $x \vee r$

Modeling software: program graphs

A **program graph** PG over set Var of typed variables is a tuple

$$(Loc, Act, Effect, \longrightarrow, Loc_0, g_0) \quad \text{where}$$

- Loc is a set of locations with initial locations $Loc_0 \subseteq Loc$
- Act is a set of actions
- $Effect : Act \times Eval(Var) \to Eval(Var)$ is the effect function
- $\longrightarrow \subseteq Loc \times (\underbrace{Cond(Var)}_{\text{Boolean condition over } Var} \times Act) \times Loc$ is the transition relation
- $g_0 \in Cond(Var)$ is the initial condition.

Notation: $\ell \xrightarrow{g:\alpha} \ell'$ denotes $(\ell, g, \alpha, \ell') \in \longrightarrow$

Beverage vending machine

- $Loc = \{ start, select \}$ with $Loc_0 = \{ start \}$
- $Act = \{ bget, sget, coin, ret\_coin, refill \}$
- $Var = \{ nsprite, \ \ nbeer \}$ with domain $\{ 0, 1, \ldots, max \}$
- *Effect*:

$$
\begin{array}{lcl}
Effect(coin, \eta) & = & \eta \\
Effect(ret\_coin, \eta) & = & \eta \\
Effect(sget, \eta) & = & \eta[nsprite := nsprite - 1] \\
Effect(bget, \eta) & = & \eta[nbeer := nbeer - 1] \\
Effect(refill, \eta) & = & \eta[nsprite := max, nbeer := max]
\end{array}
$$

- $g_0 = (nsprite = max \ \wedge \ nbeer = max)$

From program graphs to transition systems

- Basic strategy: unfolding
  - **state** = location (current control) ℓ + data valuation η
  - **initial state** = initial location satisfying the initial condition g_0
- Propositions and labeling
  - **propositions:** "ℓ" and "$x \in D$" for $D \subseteq dom(x)$
  - $\langle \ell, \eta \rangle$ is **labeled with** "ℓ" and all conditions that hold in η
- if $\ell \xrightarrow{g:\alpha} \ell'$ and g holds in η, then $\langle \ell, \eta \rangle \xrightarrow{\alpha} \langle \ell', Effect(\alpha, \eta) \rangle$

Structured operational semantics

- We describe the operational semantics using inference rules of the form

$$\frac{\text{premise}}{\text{conclusion}}$$

  The **notation** means:
  If the premise holds, then the conclusion holds

- If the premise is a tautology, it may be omitted
- In this case, the rule is also called an axiom

Transition systems for program graphs

The transition system $TS(PG)$ of program graph

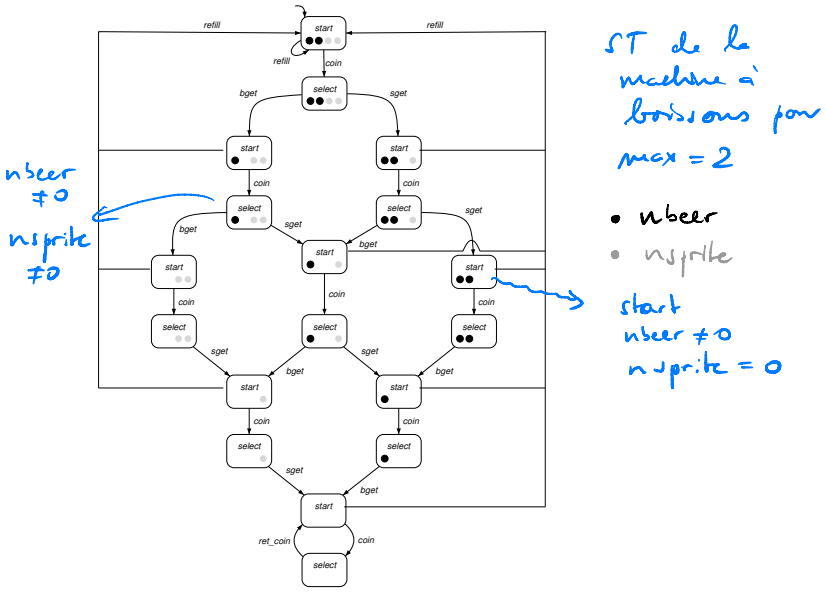
$$PG = (Loc, Act, Effect, \hookrightarrow, Loc_0, g_0)$$

over set Var of variables is the tuple $(S, Act, \longrightarrow, I, AP, L)$ where

- $S = Loc \times Eval(Var)$
- $\longrightarrow \subseteq S \times Act \times S$ is defined by the rule:

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

- $I = \{\langle \ell, \eta \rangle \mid \ell \in Loc_0, \eta \models g_0\}$
- $AP = Loc \cup Cond(Var)$ and
  $L(\langle \ell, \eta \rangle) = \{\ell\} \cup \{g \in Cond(Var) \mid \eta \models g\}$.



Synchronous composition

Let $TS_i = (S_i, Act, \rightarrow_i, I_i, AP_i, L_i)$ and
$Act \times Act \rightarrow Act$, $(\alpha, \beta) \rightarrow \alpha * \beta$

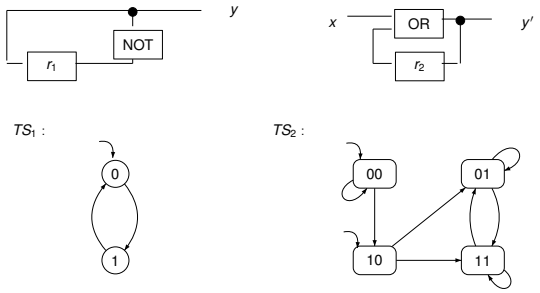
$$TS_1 \otimes TS_2 = (S_1 \times S_2, Act, \rightarrow, I_1 \times I_2, AP_1 \uplus AP_2, L)$$

with $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$ and $\rightarrow$ is defined by the following rule:

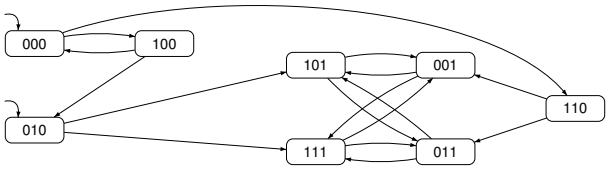
$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \quad \wedge \quad s_2 \xrightarrow{\beta}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha * \beta} \langle s_1', s_2' \rangle}$$

typically used for synchronous hardware circuits

Example



$TS_1 \otimes TS_2$:



Exclusion mutuelle

$P_1 \,||\, P_2 \,||\, \dots \,||\, P_n$

Composition by interleaving

- Actions of independent processes are interleaved for example if
  - a single processor is available
  - that takes turns in processing the actions of the processes
- No assumptions are made on the order of processes
  - possible orders for non-terminating independent processes P and Q:

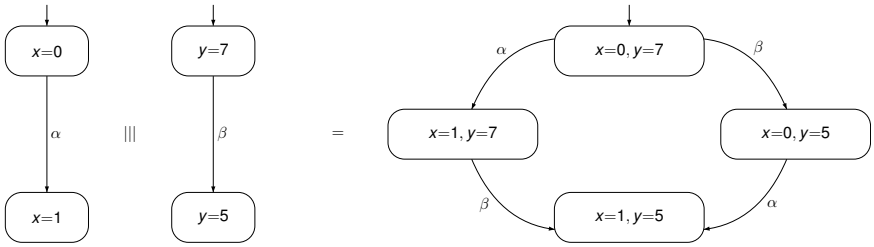
P	Q	P	Q	P	Q	Q	Q	P	...
P	P	Q	P	P	Q	P	P	Q	...
P	Q	P	P	Q	P	P	P	Q	...
...									

  - assumption: there is a scheduler with an a-priori **unknown** strategy

$P \,||\, Q$ une exécution de P est entrelacée avec une exécution de Q

Interleaving

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



Interleaving of transition systems → pour des ST qui ne se synchronisent pas

Let $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP_i, L_i)$ i=1,2, be two transition systems.

Transition system

$$TS_1 \;|||\; TS_2 = (S_1 \times S_2, Act_1 \uplus Act_2, \rightarrow, I_1 \times I_2, AP_1 \uplus AP_2, L)$$

where $L(\langle s_1, s_2\rangle) = L_1(s_1) \cup L_2(s_2)$ and the transition relation $\rightarrow$ is defined by the rules:

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

$(s_1, s_2) \xrightarrow{\alpha} (s_1', s_2)$ si $s_1 \xrightarrow{\alpha}_1 s_1'$

$(s_1, s_2) \xrightarrow{\beta} (s_1, s_2')$ si $s_2 \xrightarrow{\beta}_2 s_2'$

Interleaving of program graphs

For program graphs PG_1 (on Var_1) and PG_2 (on Var_2) **without** shared variables, i.e., $Var_1 \cap Var_2 = \emptyset$,

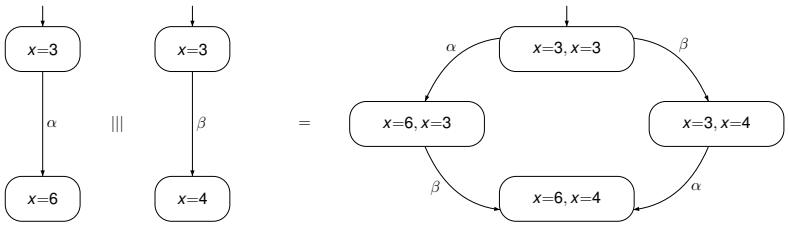
$$TS(PG_1) \;|||\; TS(PG_2)$$

faithfully describes the concurrent behavior of PG_1 and PG_2

what if they have variables in common?

Shared variable communication

$\underbrace{x := 2{\cdot}x}_{\text{action } \alpha} \;|||\; \underbrace{x := x + 1}_{\text{action } \beta}$ with initially $x = 3$



$\langle x{=}6, x{=}4\rangle$ is an **inconsistent** state!

$\Rightarrow$ no faithful model of the concurrent execution of α and β

Idea: first interleave, then unfold

Interleaving of program graphs

Let $PG_i = (Loc_i, Act_i, Effect_i, \longrightarrow_i, Loc_{0,i}, g_{0,i})$
over variables Var_i.

Program graph $PG_1 \;|||\; PG_2$ over $Var_1 \cup Var_2$ is defined by:

$$(Loc_1 \times Loc_2, Act_1 \uplus Act_2, Effect, \longrightarrow, Loc_{0,1} \times Loc_{0,2}, g_{0,1} \wedge g_{0,2})$$

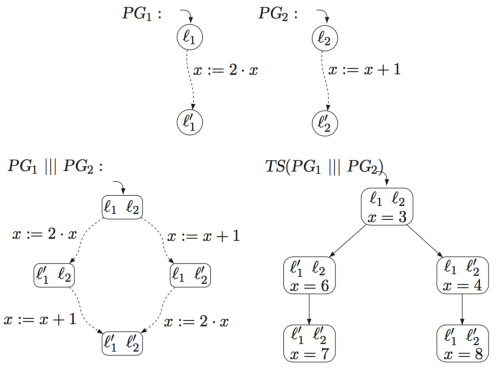
where $\longrightarrow$ is defined by the inference rules:

$$\frac{\ell_1 \xrightarrow{g:\alpha}_1 \ell_1'}{\langle \ell_1, \ell_2 \rangle \xrightarrow{g:\alpha} \langle \ell_1', \ell_2 \rangle} \quad \text{and} \quad \frac{\ell_2 \xrightarrow{g:\alpha}_2 \ell_2'}{\langle \ell_1, \ell_2 \rangle \xrightarrow{g:\alpha} \langle \ell_1, \ell_2' \rangle}$$

and $Effect(\alpha, \eta) = Effect_i(\alpha, \eta)$ if $\alpha \in Act_i$.

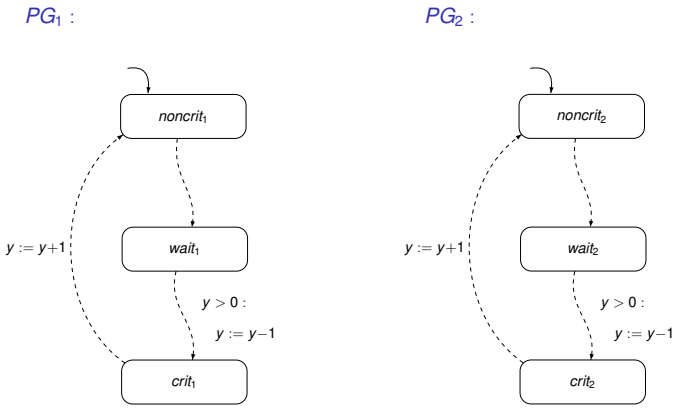
Example

$\underbrace{x := 2{\cdot}x}_{\text{action } \alpha} \;|||\; \underbrace{x := x + 1}_{\text{action } \beta}$ with initially $x = 3$



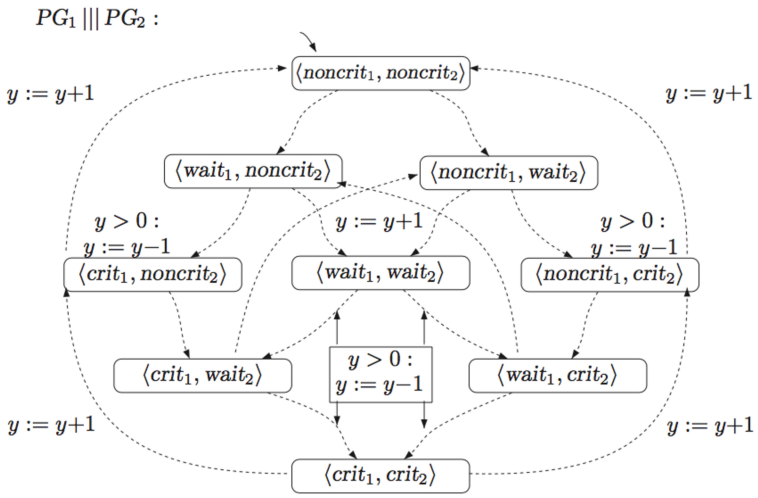
note that $TS(PG_1) \;|||\; TS(PG_2) \neq TS(PG_1 \;|||\; PG_2)$

Semaphore-based mutual exclusion

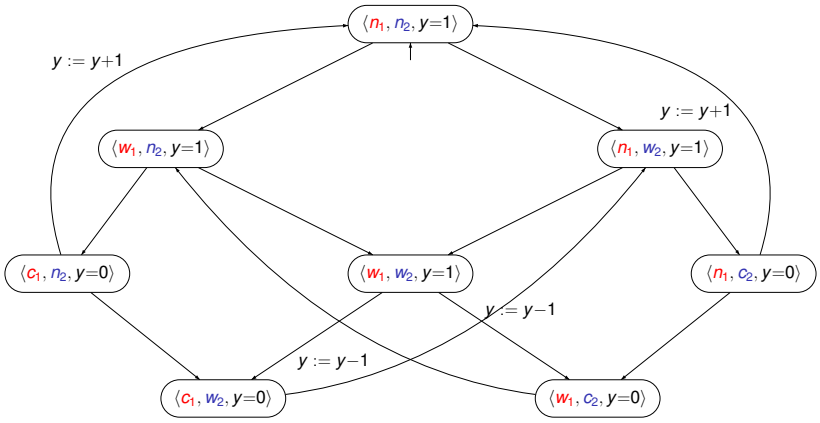


y=0 means “lock is currently possessed”; y=1 means “lock is free”

Program graph $PG_1 \,|||\, PG_2$



Transition system $TS(PG_1 \,|||\, PG_2)$



Composition by handshaking

(Renduit - vous)

ST_1 ST_2

$H \subseteq Act_1 \cap Act_2$

- H is a set of handshake actions
- actions **outside** H are independent and are **interleaved**
- actions **in** H are **synchronized**
- the interacting processes “shake hands”

Handshaking

Let $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP_i, L_i)$, i=1,2 and $H \subseteq Act_1 \cap Act_2$.

$TS_1 \parallel_H TS_2 \;=\; (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \uplus AP_2, L)$

where $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$ and with $\rightarrow$ defined by:

- **interleaving** for $\alpha \notin H$:

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

- **handshaking** for $\alpha \in H$:

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

note that $TS_1 \parallel_H TS_2 = TS_2 \parallel_H TS_1$ but
$(TS_1 \parallel_{H_1} TS_2) \parallel_{H_2} TS_3 \neq TS_1 \parallel_{H_1} (TS_2 \parallel_{H_2} TS_3)$

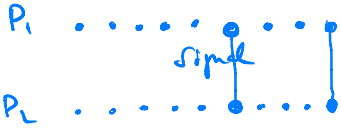
exemple

processus qui communiquent par signaux

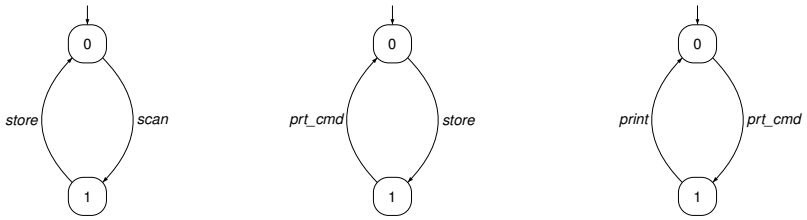
$P_1 ! P_2$ P_1 envoie signal à P_2

$P_2 ? P_1$ P_2 reçoit signal de P_1

$H = \{ signal(P_1, P_2) \}$



A booking system



BCR || *BP* || *Printer*

|| is a shorthand for $\parallel_H$ with $H = Act_1 \cap Act_2$

Handshaking

Let $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP_i, L_i)$, i=1,2 and $H \subseteq Act_1 \cap Act_2$.

$TS_1 \parallel_H TS_2 \;=\; (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \uplus AP_2, L)$

where $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$ and with $\rightarrow$ defined by:

- **interleaving** for $\alpha \notin H$:

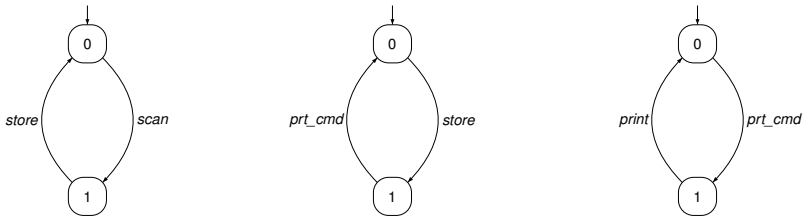
$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

- **handshaking** for $\alpha \in H$:

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

note that $TS_1 \parallel_H TS_2 = TS_2 \parallel_H TS_1$ but
$(TS_1 \parallel_{H_1} TS_2) \parallel_{H_2} TS_3 \neq TS_1 \parallel_{H_1} (TS_2 \parallel_{H_2} TS_3)$

A booking system



BCR $\parallel$ *BP* $\parallel$ *Printer*

$\parallel$ is a shorthand for $\parallel_H$ with $H = Act_1 \cap Act_2$