Contents

1 Parallelism and Communication

Parallelism and Communication

In general, hardware and software systems are parallel and may communicate among them. We want to define the system

$$T_1||T_2\cdots||T_n$$

Parallelism can be modelled in several ways:

- Interleaving processes (asynchronous).
- Communication by shared variables
- Synchronous product
- *Handshaking* (actions allow to synchronise processes)
- Message passing - communication by channels

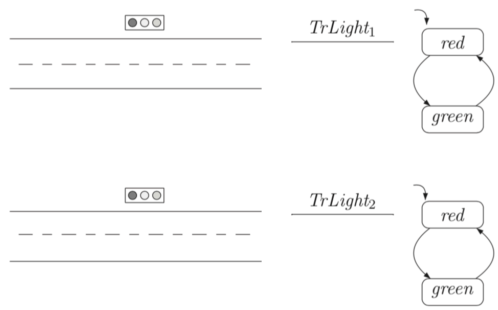
Concurrence and Interleaving

- nondeterministic alternation of actions of each component, Let P and Q be two components

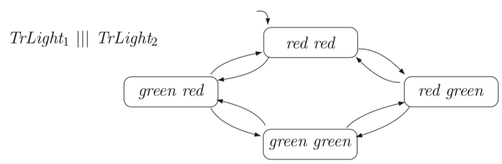
$$PPQQPQQPPPQQP\ldots$$

- one processor executes several processes that do not interact
- there is a scheduler with a given strategy
- interleaving must be *fair*

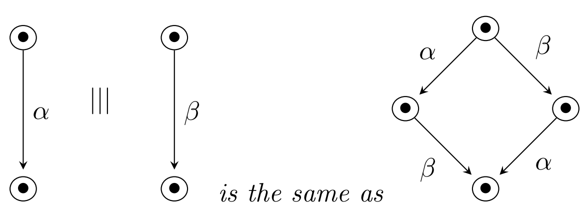
Traffic lights in parallel streets



Transition System $TL_1|||TL_2$

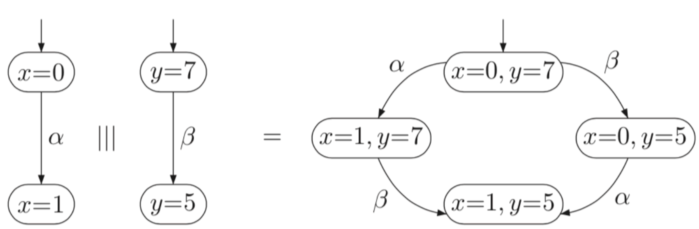


- ||| interleaving operator
- ; sequential execution
- + nondeterministic choice $Effect(\alpha|||\beta, \eta) = Effect((\alpha;\beta) + (\beta;\alpha), \eta)$



Example

Let α be $x \leftarrow x + 1$, β be $y \leftarrow y - 2$ and $\eta = \langle x = 0, y = 7\rangle$ then the diagram of $\alpha|||\beta$ is



Note: This works because there is no shared variables. Otherwise the order matters and one need to use program graphs, e.g. $x \leftarrow x + 1 ||| x \leftarrow 2x$.

Interleaving of Transition Systems

$T_i = (S_i, Act_i, \longrightarrow_i, I_i, AP_i, L_i)$, $i = 1, 2$

$$T_1 ||| T_2 = (S_1 \times S_2, Act_1 \cup Act_2, \longrightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$$

where the transition relations is defined by

$$\frac{s_1 \overset{\alpha}{\longrightarrow}_1 s_1'}{\langle s_1, s_2 \rangle \overset{\alpha}{\longrightarrow} \langle s_1', s_2 \rangle} \quad \frac{s_2 \overset{\alpha}{\longrightarrow}_2 s_2'}{\langle s_1, s_2 \rangle \overset{\alpha}{\longrightarrow} \langle s_1, s_2' \rangle}$$

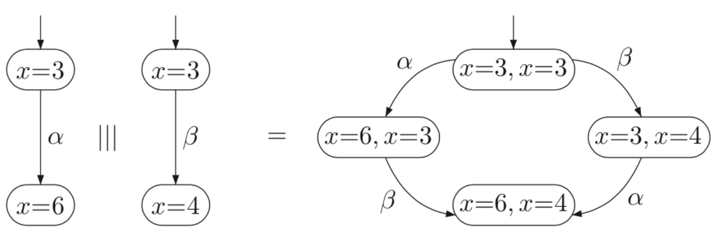
and $L(\langle s_1, s_2 \rangle) = L(s_1) \cup L(s_2)$.

For PG_i with $Var_1 \cap Var_2 = \emptyset$, $T(PG_1) ||| T(PG_2)$ is the transition system for simultaneous execution.

Note: corresponds to a cartesian product.

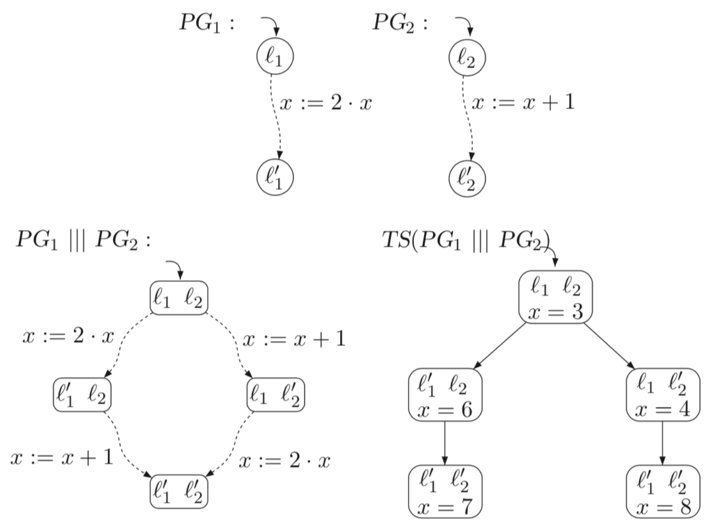
Communication with Shared Variables

For $x \leftarrow x + 1 \ ||| \ x \leftarrow 2x$ and $x = 3$



3 *states are inconsistent!* *Solution*: interleaving the program graphs and not their transition systems ($PG_1 ||| PG_2$).

Example: let $\langle x = 3 \rangle$



Critical Actions

- Actions that act on shared variables are called *critical*.
- Processes internal actions can use nondeterministic choices
- Critical actions cannot be execute in parallel, and there must exist some scheduler strategy.

Atomicity

- $\alpha \in Act$ represented in a transitions system must be indivisible.
- E.g. if α is a below it cannot be divisible (i.e corresponds only to one transition)

If the following is a α action it cannot be splitetd by a scheduller.

$x \leftarrow x + 1;$
$y \leftarrow 2x + 1;$
if $x \leq 12$ **then**
  $z \leftarrow (x - z)^2 \times y$

$$\begin{aligned}
Effect(\alpha,\eta)(x) &= \eta(x)+1\\
Effect(\alpha,\eta)(y) &= 2(\eta(x)+1)+1\\
Effect(\alpha,\eta)(z) &= \begin{cases}(\eta(x)-\eta(z))^2 \times \eta(y), & \text{se } \eta(x)\leq 12\\ \eta(z), \text{caso contrário}\end{cases}
\end{aligned}$$

Mutual Exclusion

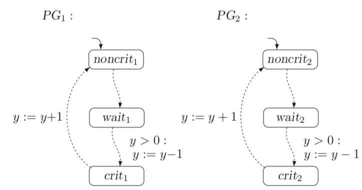
- Whenever concurrent processes share a resource it may be necessary to ensure that they do not have access to it at the same time. That is called the *critical section*
- Examples:
  - shared variables: simultaneous update cannot occur
  - access to devices: e.g. a printer
- Problem: find a protocol for determining which process is allowed to enter its critical section
- Expected properties:
- *Safety*: Only one process is in its critical section at any time
- *Liveness*: Whenever any process requests to enter in its critical section will eventually be permitted to enter
- *Non-blocking*: A process can always request to enter its critical section

Mutual Exclusion with Semaphores

Two processes P_1 and P_2 want to access a critical section. P_i:

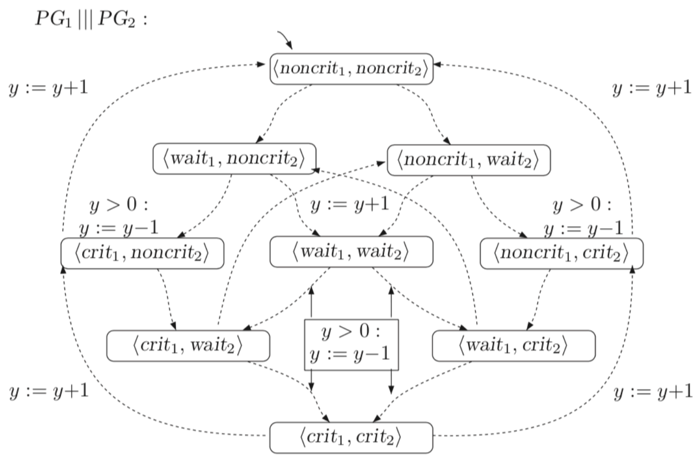
while True **do**
  non critical actions
  await $y > 0$ **do** $y \leftarrow y - 1$ **od**
  critical actions
  $y \leftarrow y + 1$

```
P_i  loop forever
       ⋮                 (* noncritical actions *)
       request
       critical section
       release
       ⋮                 (* noncritical actions *)
     end loop
```



Shared variable y is a binary semaphore: if $y = 0$ one of the processes is in the critical zone; if $y = 1$ the critical zone is free.

$PG_1|||PG_2$



Forbidden Location $\langle crit_1, crit_2 \rangle$

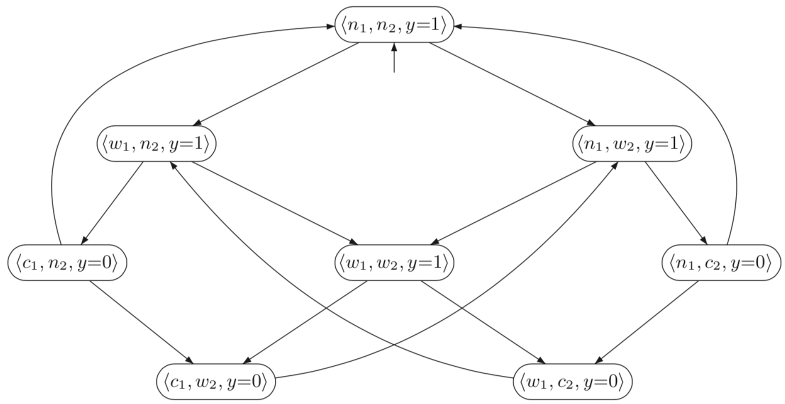
$T(PG_1|||PG_2)$

Initially $y = 1$. From 18 states, only 8 states are reachable:

$$
\begin{array}{rl}
\langle noncrit_1, noncrit_2, y = 1\rangle & \langle noncrit_1, wait_2, y = 1\rangle \\
\langle wait_1, noncrit_2, y = 1\rangle & \langle wait_1, wait_2, y = 1\rangle \\
\langle noncrit_1, crit_2, y = 0\rangle & \langle crit_1, noncrit_2, y = 0\rangle \\
\langle wait_1, crit_2, y = 0\rangle & \langle crit_1, wait_2, y = 0\rangle
\end{array}
$$

Many states are not reachable, including $\langle crit_1, crit_2, y = \ldots\rangle$ thus it satisfies the *mutual exclusion* property.

$T(PG_1 ||| PG_2)$



How to decide how to leave the state$\langle wait_1, wait_2, y = 1\rangle$?

Synchronous Product

$T_i = (S_i, Act_i, \longrightarrow_i, I_i, AP_i, L_i)$ for $i = 1, 2$

$$\begin{array}{rcl} * : Act_1 \times Act_2 & \rightarrow & Act \\ (\alpha, \beta) & \mapsto & \alpha * \beta \end{array}$$

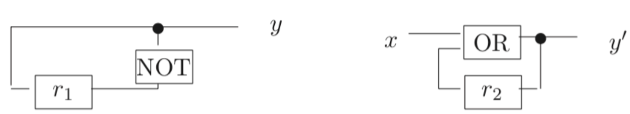
$$T_1 \otimes T_2 = (S_1 \times S_2, Act, \longrightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$$

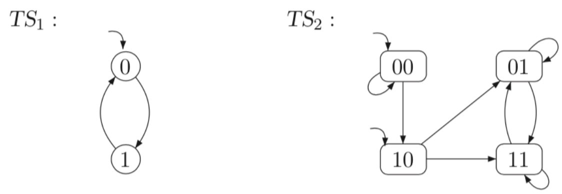
where the transition functions is defined by

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

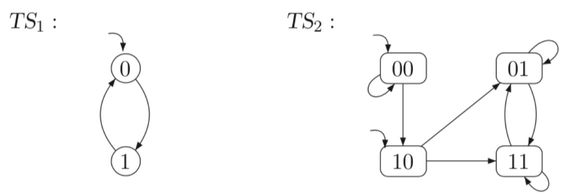
and $L(\langle s_1, s_2\rangle) = L(s_1) \cup L(s_2)$.

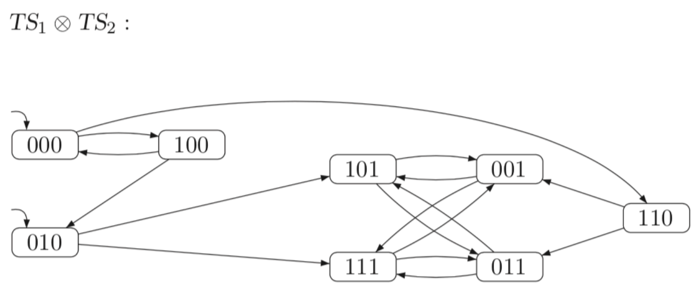
Synchronous Product of Circuits



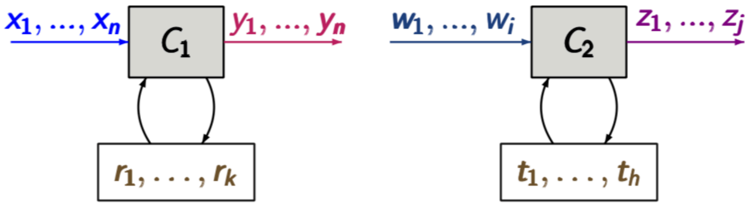


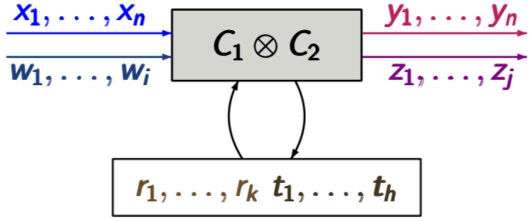
Synchronous Product of Circuits





Synchronous Product of Circuits(general)



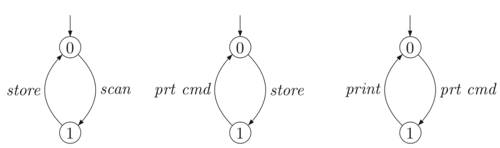


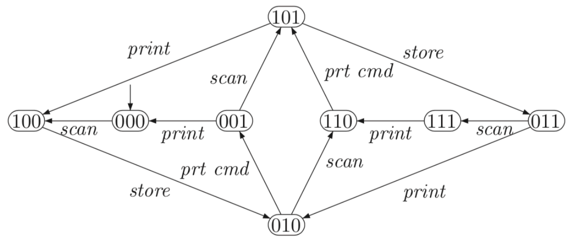
Handshaking: Synchronous Message Passing

- concurrent processes interact in a synchronous fashion
- both processes share a set of actions $H \subseteq Act_i$ (*handshake*) and has to execute the same action $\alpha \in H$ simultaneously
- other actions may be executed in a interleaved fashion
- corresponds also to a channel of size 0 (synchronous message passing)

Booking System- at a supermarket cashier

- Bar code reader (product code) (BCR)
- Booking program (price) (BP)
- receipt printer (Printer)
- $BCR||BP||Printer$





Handshaking-Synchronous Message Passing

$T_i = (S_i, Act_i, \longrightarrow_i, I_i, AP_i, L_i)$ for $i = 1, 2$ e $H \subseteq Act_1 \cap Act_2$ and $\tau \notin H$.

$$T_1||_H T_2 = (S_1 \times S_2, Act_1 \cup Act_2, \longrightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$$

where

- $L(\langle s_1, s_2 \rangle) = L(s_1) \cup L(s_2)$
- Se $\alpha \notin H$,
$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \quad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$
- se $\alpha \in H$
$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \wedge s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2' \rangle}$$

Properties of $||_H$

- if $H = Act_1 \cap Act_2$, $T_1||T_2$
- $T_1||_\emptyset T_2 = T_1|||T_2$
- $T_1||_H T_2 = T_2||_H T_1$ (commutative)
- in general $T_1||_H(T_2||_{H'}T_3) \neq (T_1||_H T_2)||_{H'}T_3)$ (is not associative)
- If $H = H'$ it is associative
  $T = T_1||_H T_2||_H T_3 \cdots ||_H T_n$ with $H \subseteq Act_1 \cap \cdots \cap Act_n$
- models *broadcasting* where one process commuticates with several processes at the same time
- $||_H$ generalizes to $T_1||T_2 \ldots ||T_n$, with $H_{i,j} = Act_i \cap Act_j$ and $H_{i,j} \cap Act_k = \emptyset$ for $k \notin \{i, j\}$.

Mutual Exclusion with Arbiter

Process P_i

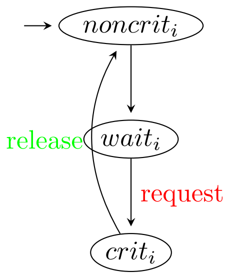
while True **do**

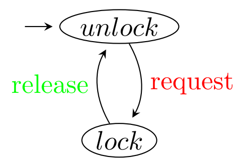
noncricital actions

request

critical actions

release

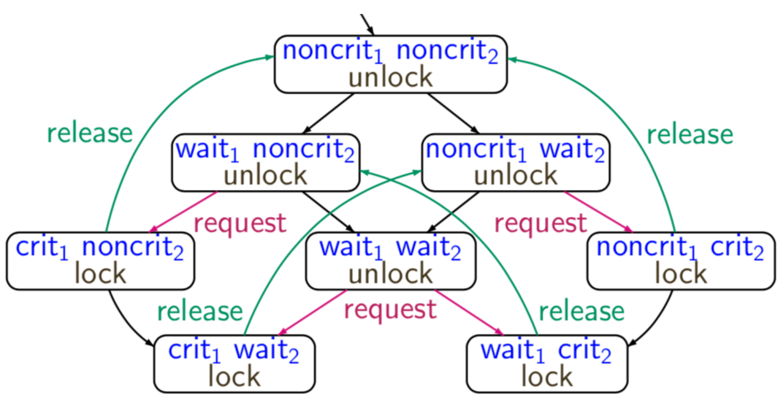




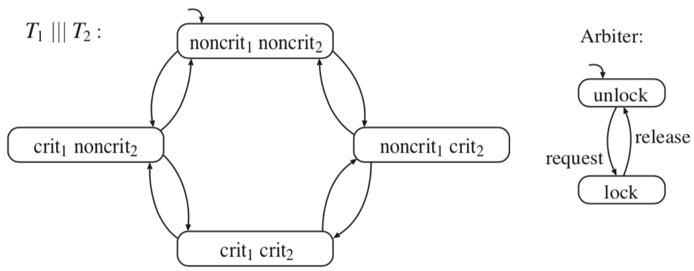
Process *Arbiter* selects P_1 or P_2 nondeterministically

$$(T_1|||T_2)||_{Syn}Arbiter$$

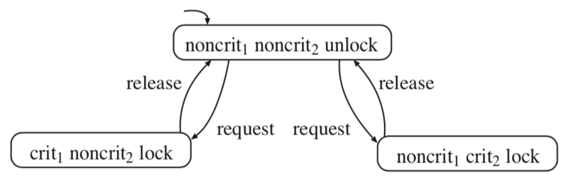
where $Syn = \{request, release\}$



Mutual Exclusion with Arbiter (simplified without *wait*)

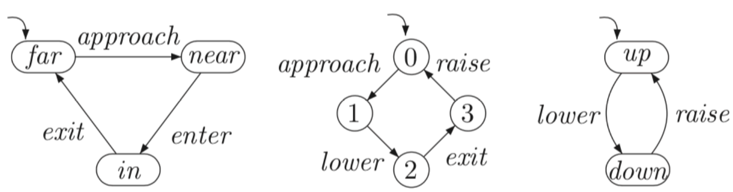


$(T_1 ||| T_2) ||$ Arbiter :



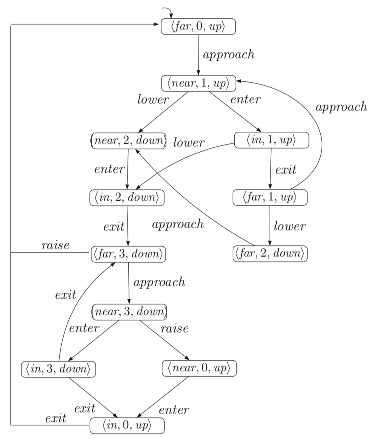
Railroad Crossing

- when a train approaches sends a signal for the gate to close
- the gate opens after the train sends an exit signal
- we want that the gates ar closed when the train is passing
- $Train||Gate||Controler$



Train *Controller* *Gate*

Is the railroad crossing safe?



No, we need to have real time restrictions.

See more:[BKL08, Chap. 2.2.1-2.2.3,2.2.6]

References

[BKL08] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen. *Principles of Model Checking*. MIT Press, 2008.