Aula 5

Operadores paralelos (resumo)

- *concorrência pura* $T_1|||T_2$
- passagem de mensagens síncrona $T_1||_{Syn}T_2$
- *comunicação por variáveis partilhadas:*
  - descrição dos sistemas por grafos de programa
  - intercalagem de grafos de programa $P_1|||P_2$;
  - TS por expansão
- Sistemas de canais $[P_1|\ldots P_2]$ com
  - concorrentes, variáveis partilhadas
  - passagem de mensagens síncrona
  - passagem de mensagens assíncrona
- Paralelismo síncrono: (p.e. para circuitos sequenciais, sincronia sempre mesmo com ações diferentes –com um relógio
-)

***Handshaking*-passagem de mensagens síncrona**

- processos concorrentes interagem de forma síncrona
- a informação trocada pode ser complexa (mas vamos omitir isso)
- ambos os processos partilham um conjunto de ações $H \subseteq Act_i$ (*handshake*) e têm de executar uma mesma ação $\alpha \in H$ simultaneamente
- as restantes ações podem ser executadas intercaladas (independentes)

***Handshaking*-passagem de mensagens síncrona**

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

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

onde

- $L(\langle s_1, s_2\rangle) = L(s_1) \cup L(s_2)$
- 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 $\alpha \in H$

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

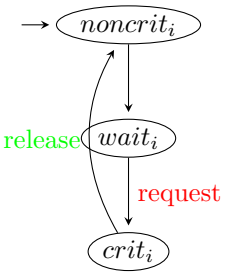
Propriedades de $||_H$

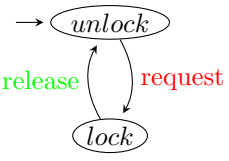
- se $H = Act_1 \cap Act_2$, usa-se $T_1||T_2$
- $T_1||_\emptyset T_2 = T_1|||T_2$
- $T_1||_H T_2 = T_2||_H T_1$ (comutativo)
- em geral $T_1||_H(T_2||_{H'}T_3) \neq (T_1||_H T_2)||_{H'}T_3)$ (não associativo)
- Mas se $H = H'$ é associativo
  $T = T_1||_H T_2||_H T_3 \cdots ||_H T_n$ com $H \subseteq Act_1 \cap \cdots \cap Act_n$
- modela *broadcasting* onde um processo pode enviar informação para vários processos em simultâneo.
- $||_H$ pode-se generalizar a $T_1||T_2 \ldots ||T_n$, com $H_{i,j} = Act_i \cap Act_j$ e $H_{i,j} \cap Act_k = \emptyset$ para $k \notin \{i, j\}$ (verifica!).

Exclusão mútua com árbitro

Para o processo P_i

while True **do**
noncricital actions
request
critical actions
release

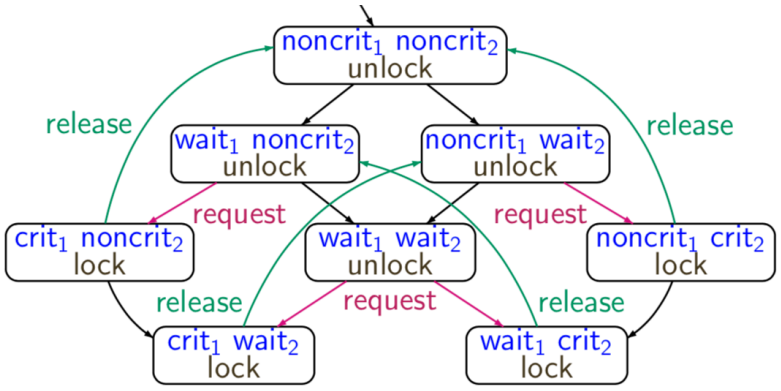




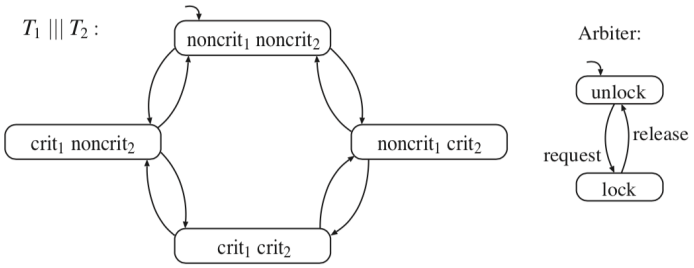
O processo árbitro *Arbiter* seleciona P_1 ou P_2 não deterministicamente

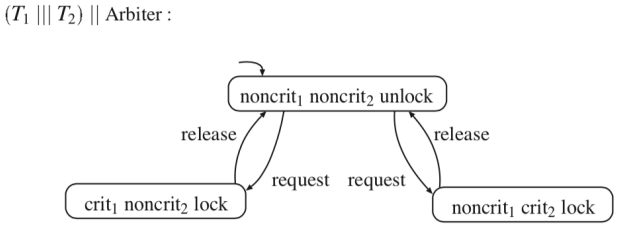
$$(T_1|||T_2)||_{Syn}Arbiter$$

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



Exclusão mútua com árbitro (simplificada sem *wait*)

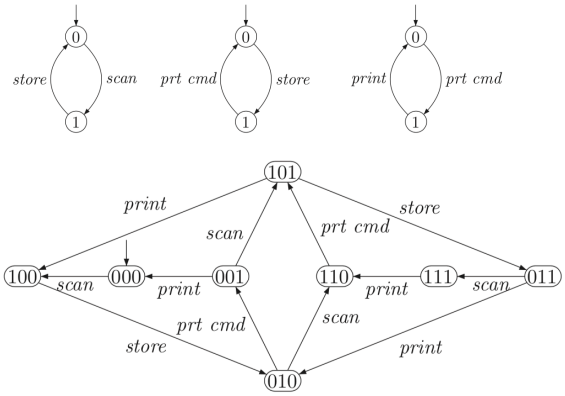




Máquina registadora

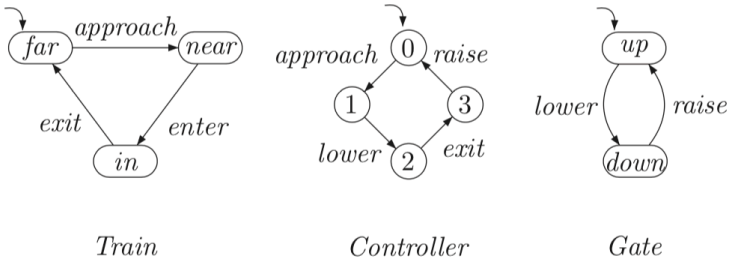
- Leitor de código de barras (código do produto) (BCR)
- Programa de Registo (preço) (BP)

- Impressão de recibo (Printer)
- $BCR||BP||Printer$

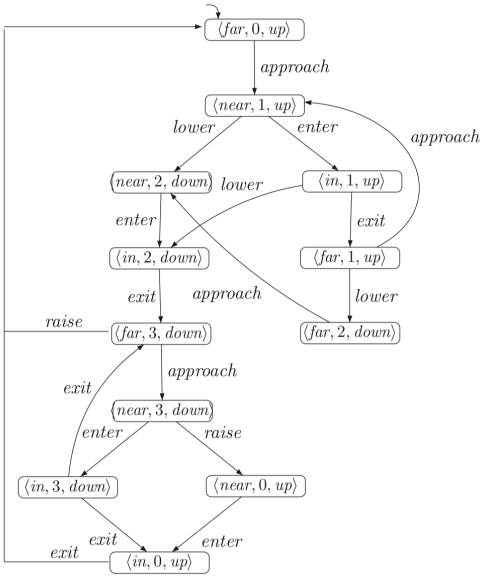


Passagem de nível

- quando o comboio se aproxima envia um sinal para a cancela fechar
- a cancela abre depois do comboio enviar um sinal que já passou
- pretende-se que as cancelas estejam fechadas quando o comboio passa.
- $Train||Gate||Controler$



A passagem de nível é segura?



Não, mas seria se houvessem restrições de tempo real.

Comunicação por canais

- Um canal pode ser um buffer FIFO (variável partilhada)
- são usados em protocolos de comunicação
- um *sistema de canais*
- tem n processos $P_1,\ldots,P_n$, onde cada um tem um grafo PG_i com
- transições condicionais $g:\alpha$ ou *ações de comunicação*:
- $g:c!v$ transmitir a mensagem v pelo canal c
- $g:c?x$ receber uma mensagem pelo canal c e guardar na variável x.
- $\ell \overset{g:\alpha}{\hookrightarrow} \ell'$, $\ell \overset{g:c!v}{\hookrightarrow} \ell'$, ou $\ell \overset{g:c?x}{\hookrightarrow} \ell'$.
- podem ser síncronos ou assíncronos

Canais

- Seja c um buffer.
- $c!v$ coloca v no fim do buffer c
- $c?x$ vai buscar o elemento no topo do buffer c e guarda-o em x
- *capacidade do canal*,
$$cap(c) \in \mathbb{N} \cup \{\infty\},$$
indica o número máximo de mensagens que c pode conter (pode ser finito ou infinito)
- *tipo do canal*, indica o tipo das mensagens que se pode transmitir sobre c, $dom(c)$.
- Seja $Chan$ um conjunto de canais, o conjunto das *ações de comunicação* é
$$\begin{aligned} Comm \quad = \quad & \{c!v, c?x \mid c \in Chan \wedge v \in dom(c) \wedge \\ & x \in Var \wedge dom(x) \subseteq dom(c)\} \end{aligned}$$

Síncronos e Assíncronos

- Ex: um canal c que transmite bits tem $dom(c) = \{0, 1\}$
- Se $cap(c) = 0$ o sistema corresponde a comunicação por *Handshaking*: transmissão e recepção simultânea, *síncrona*
- Se $cap(0) > 0$ há um atraso na transmissão e na recepção da mensagem: *passagem de mensagens assíncrona*

Sistema de canais (CS)

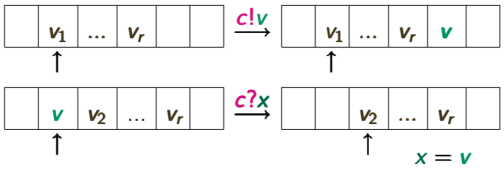
$CS = [PG_1|PG_2|\cdots|PG_n]$ sobre $(Var, Chan)$ onde PG_i são grafos de programa sobre $(Var_i, Chan)$

- $Var = \bigcup_{1 \leq i \leq n} Var_i$ conjunto de variáveis tipificadas
- $Chan$ conjunto de canais tipificados com capacidades $cap(\cdot)$ e domínios $dom(\cdot)$
- $PG_i = (Loc_i, Act_i, Effect_i, \hookrightarrow_i, Loc_{0,i}, g_{0,i})$
$$\hookrightarrow_i \subseteq Loc_i \times (Cond(Var_i) \times (Act_i \cup Comm_i)) \times Loc_i$$
- $\ell \overset{g:\alpha}{\hookrightarrow}_i \ell'$, g guarda

- $\ell \overset{g:c!v}{\hookrightarrow}_i \ell'$, envia o valor v via o canal c
- $\ell \overset{g:c?x}{\hookrightarrow}_i \ell'$, recebe um valor que guarda na variável x via o canal c
- Em geral, se $g = True$ omitimos g : nas ações de comunicação.

Comunicação se $cap(c) > 0$

- P_i pode executar a transição $\ell_i \overset{c!v}{\hookrightarrow}_i \ell'_i$ se e só se *o canal não está cheio* e v é guardado no fim de c $(add(c, v))$
- P_j pode executar $\ell_j \overset{c?x}{\hookrightarrow}_j \ell'_j$ se *o canal c não está vazio* $(v = front(c); x := v; remove(c))$



Comunicação se $cap(c) = 0$ **(*rendezvous*)**

- o processo P_i pode executar

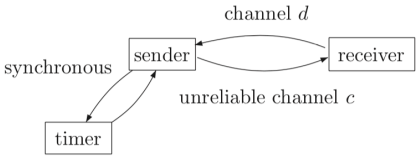
$$\ell_i \overset{c!v}{\hookrightarrow}_i \ell'_i$$

se simultaneamente existe um processo P_j que oferece a ação complementar

$$\ell_j \overset{c?x}{\hookrightarrow}_j \ell'_j$$

- sendo o resultado equivalente a $x := v$ (sincronia).

***Alternating bit Protocol* (ABP)**



- o canal c não é perfeito e pode perder mensagens enviadas
- o canal d é perfeito e envia "acknowledgment"

- Pretende-se protocolo de comunicação que
- assegure que dados distintos de S são entregues a R.
- Para tal S tem de retransmitir mensagens
- e uma nova mensagem só é enviada quando houver garantia que a anterior foi recebida (*send and wait*))

Alternating bit Protocol

- S envia a mensagem, um bit y extra e ativa o *timer*
- se houver *timout* volta a enviar a mesma mensagem
- se R enviou y então S reinicia o timer e faz $y = \neg y$ (tornando a enviar uma mensagem).
- Não considerando o tempo-real, o "timeout"é considerado de modo não deteministico.

Sender

while True **do**
(1) send message + bit y and activate timer
(2) **await** timeout or ack x **do**
if timout **then**
goto (1)
else if x=y **then**
turn off timer; y=-y
else
ignore x
od

Sender

while True **do**
(1) send message + bit y and activate timer
(2) **await** timeout or ack x **do**
if timout **then**
goto (1)
else if x=y **then**
turn off timer; y=-y
else
ignore x
od

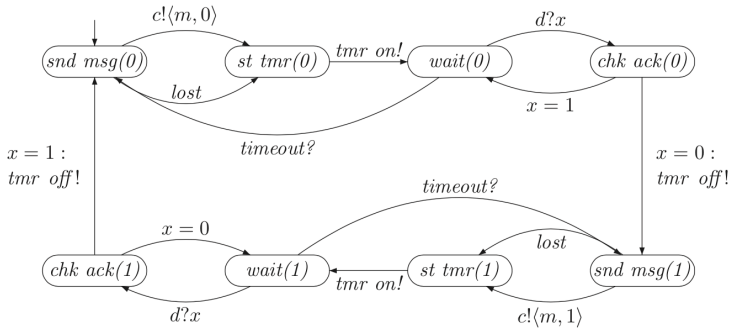
Alternating bit Protocol

- S envia mensagens sobre c

  $$\langle m_0, b_0\rangle, \langle m_1, b_1\rangle, \ldots$$

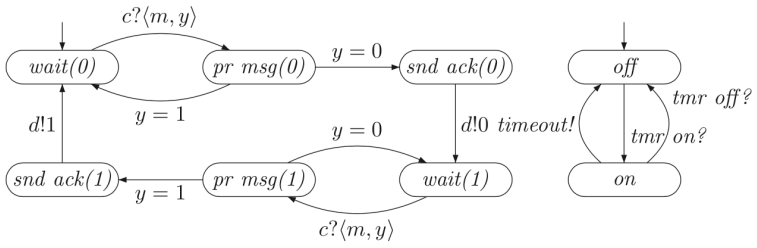
  e $b_0 = 0$, $b_1 = 1$, $b_2 = 0$, ...
- Quando R recebe $\langle m, b\rangle$ envia o bit de controlo b que recebeu pelo canal d
- Quando S recebe b, S transmite uma nova mensagem m' com o bit $\neg b$.
- Mas se S tiver de esperar muito a receber a mensagem de R, S retransmite $\langle m, b\rangle$ (aqui é a simulação é feita usano o nãodeterminismo).

PG **para o *Sender***



$$\begin{aligned} Chan &= \{c, d, tmr\_on, tmr\_off, timeout\} \\ Var &= \{x, y, m_i\} \end{aligned}$$

PG **para *Receiver* e *Timer***



$$ABP \quad = \quad [S|Timer|R]$$

- passagem de mensagens síncrona: entre S e $Timer$
- passagem de mensagens asíncrona: entre S e R

Sistemas de transições para um CS

Seja $CS = [PG_1|PG_2|\cdots|PG_n]$ sobre $(Var, Chan)$.

$$PG_i = (Loc_i, Act_i, Effect_i, \hookrightarrow_i, Loc_{0,i}, g_{0,i})$$

- estados $\langle \ell_1, \ldots, \ell_n, \eta, \zeta \rangle$
- ℓ_i localização em PG_i
- $\eta \in Eval(Var)$ atribuição de valores às variáveis
- $\zeta : Chan \to \bigcup_{c \in Chan} dom(c)^\star$ atribuição de valores dos canais
- para $c \in Chan$, $\zeta(c) \in dom(c)^\star$
- e $len(\zeta(c)) \leq cap(c)$
- $Eval(Chan)$ é o conjunto de todos os ζ.

Sistemas de transições para um CS

Seja $CS = [PG_1|PG_2|\cdots|PG_n]$ sobre $(Var, Chan)$.

$$PG_i = (Loc_i, Act_i, Effect_i, \hookrightarrow_i, Loc_{0,i}, g_{0,i})$$

- estados iniciais: componentes $\ell_i \in Loc_{0,i}$
- inicialmente todos os canais estão vazios ($\zeta_0(c) = \varepsilon,\ c \in Chan$) e $len(\varepsilon) = 0$.
- $\zeta(c) = v_1 v_2 \cdots v_k$, com v_1 o *topo* do canal
- $len(\zeta) = k$
- $\zeta[c := w_1, w_2, \ldots, w_k]$ é a atribuição igual a ζ mas com $\zeta(c) = w_1 w_2 \cdots w_k$

$$\zeta[c := w_1, w_2, \ldots, w_k](c') = \begin{cases} \zeta(c') \text{ se } c' \neq c \\ w_1 w_2 \cdots w_k \text{se } c' = c \end{cases}$$

$T(CS)$

$$T(CS) = (S, Act, \longrightarrow, I, AP, L)$$

- $S = (Loc_1 \times \cdots \times Loc_n) \times Eval(Var) \times Eval(Chan)$
- $Act = \bigoplus_{0<1\leq n} Act_i \oplus \{\tau\}$, união disjunta
- $I = \{\ \langle \ell_1, \ldots, \ell_n, \eta, \zeta_0\rangle \mid \forall 0 < i \leq n(\ell_i \in Loc_{0,i} \wedge \eta \models g_{0,i})\ \}$
- $AP = \bigoplus_{0<1\leq n} Loc_i \oplus Cond(Var)$
- $L(\langle \ell_1, \ldots, \ell_n, \eta, \zeta\rangle) = \{\ell_1, \ldots, \ell_n\} \cup \{\ g \in Cond(Var) \mid \eta \models g\ \}$
- relação de transição $\longrightarrow$ tem regras para ações $\alpha \in Act_i$ e de passagem de mensagens.

Concorrência/intercalagem para $\alpha \in Act_i$

$$\frac{\ell_i \overset{g:\alpha}{\hookrightarrow}_i \ell_i' \wedge \eta \models g}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_n, \eta, \zeta\rangle \overset{\alpha}{\longrightarrow} \langle \ell_1, \ldots, \ell_i', \ldots, \ell_n, \eta', \zeta\rangle}$$

com $\eta' = Effect(\alpha, \eta)$.

Passagem de mensagens assíncrona para $c \in Chan$ **e** $cap(c) > 0$

- receber um valor em c e guardar em x

$$\frac{\ell_i \overset{g:c?x}{\hookrightarrow}_i \ell_i' \wedge \eta \models g \wedge \zeta(c) = v_1 \cdots v_k \wedge k > 0}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_n, \eta, \zeta\rangle \overset{\tau}{\longrightarrow} \langle \ell_1, \ldots, \ell_i', \ldots, \ell_n, \eta', \zeta'\rangle}$$

  com $\eta' = \eta[x := v_1]$ e $\zeta' = \zeta[c := v_2 \cdots v_k]$.

- Transmitir um valor $v \in dom(c)$ sobre c

$$\frac{\ell_i \overset{g:c!v}{\hookrightarrow}_i \ell_i' \wedge \eta \models g \wedge \zeta(c) = v_1 \cdots v_k \wedge k < cap(c)}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_n, \eta, \zeta\rangle \overset{\tau}{\longrightarrow} \langle \ell_1, \ldots, \ell_i, \ldots, \ell_n, \eta', \zeta'\rangle}$$

  com $\zeta' = \zeta[c := v_1 \cdots v_k v]$.

Passagem de mensagens síncrona para $c \in Chan$ **e** $cap(c) = 0$

$$\frac{\ell_i \overset{g_1:c?x}{\hookrightarrow}_i \ell_i' \wedge \eta \models g_1 \wedge \eta \models g_2 \wedge \ell_j \overset{g_2:c!v}{\hookrightarrow}_j \ell_j' \wedge i \neq j}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_j, \ldots \ell_n, \eta, \zeta\rangle \overset{\tau}{\longrightarrow} \langle \ell_1', \ldots, \ell_i', \ldots, \ell_j', \ldots \ell_n', \eta', \zeta\rangle}$$

com $\eta' = \eta[x := v]$.

Quantos estados tem um sistema de transições...

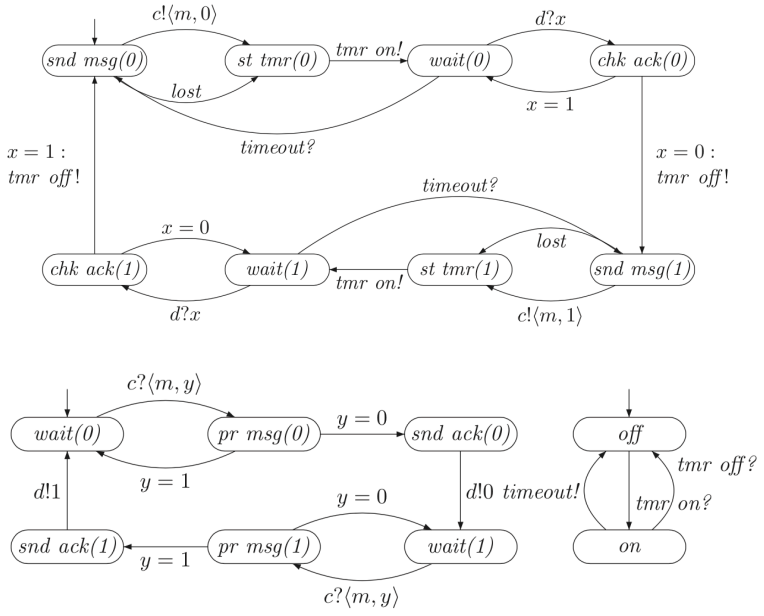
de um sistema de canais com:

- 2 processos com 2 localizações
- 2 variáveis Booleanas
- 2 canais de capacidade 10 de valores Booleanos
- ?

$$2 \times 2 \times 2 \times 2 \times (1 + 2 + 2^2 + \cdots + 2^{10}) = 2^4(2^{11} - 1)^2 > 2^{24}$$

se os canais não forem limitados, $cap(c) = \infty$, o número de estados também é ∞.

ABP



$T(ABP)$

- como o Timer pode fazer *timeout* de modo arbitrário o número de mensagens em c pode ser infinito,
- pelo que $T(ABP)$ pode ser infinito

- fragmento de execução onde a mensagem é perdida

sender S	timer	receiver R	channel c	channel d	event
$snd\ msg(0)$	off	$wait(0)$	$\varnothing$	$\varnothing$	
$st\ tmr(0)$	off	$wait(0)$	$\varnothing$	$\varnothing$	loss of message
$wait(0)$	on	$wait(0)$	$\varnothing$	$\varnothing$	
$snd\ msg(0)$	off	$wait(0)$	$\varnothing$	$\varnothing$	timeout
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Ignorar retransmissões

sender S	timer	receiver R	channel c	channel d	event
$snd\ msg(0)$	off	$wait(0)$	$\varnothing$	$\varnothing$	
$st\ tmr(0)$	off	$wait(0)$	$\langle m,0\rangle$	$\varnothing$	message with bit 0 transmitted
$wait(0)$	on	$wait(0)$	$\langle m,0\rangle$	$\varnothing$	
$snd\ msg(0)$	off	$wait(0)$	$\langle m,0\rangle$	$\varnothing$	timeout
$st\ tmr(0)$	off	$wait(0)$	$\langle m,0\rangle\ \langle m,0\rangle$	$\varnothing$	retransmission
$st\ tmr(0)$	off	$pr\ msg(0)$	$\langle m,0\rangle$	$\varnothing$	receiver reads first message
$st\ tmr(0)$	off	$snd\ ack(0)$	$\langle m,0\rangle$	$\varnothing$	
$st\ tmr(0)$	off	$wait(1)$	$\langle m,0\rangle$	0	receiver changes into mode-1
$st\ tmr(0)$	off	$pr\ msg(1)$	$\varnothing$	0	receiver reads retransmission
$st\ tmr(0)$	off	$wait(1)$	$\varnothing$	0	and ignores it
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Operadores paralelos (resumo)

- concorrência pura $T_1|||T_2$
- passagem de mensagens síncrona $T_1||_{Syn}T_2$
- comunicação por variáveis partilhadas:
  - descrição dos sistemas por grafos de programa
  - intercalagem de grafos de programa $P_1|||P_2$;
  - TS por expansão
- Sistemas de canais $[P_1|\ldots P_2]$ com
  - concorrentes, variáveis partilhadas
  - passagem de mensagens síncrona
  - passagem de mensagens assíncrona
- Paralelismo síncrono: (p.e. para circuitos sequenciais, sincronia sempre mesmo com ações diferentes –com um relógio
-)

Produto Síncrono

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

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

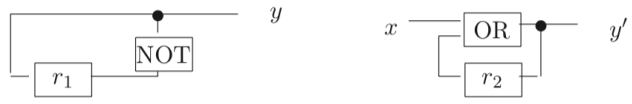
$$T_1 \otimes T_2 = (S_1 \times S_2, Act, \longrightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$$

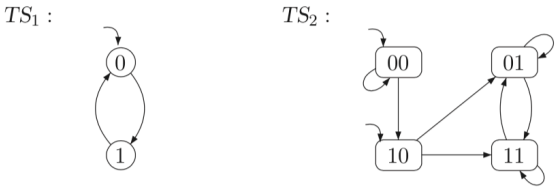
onde a relação de transição é definida pela regra

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

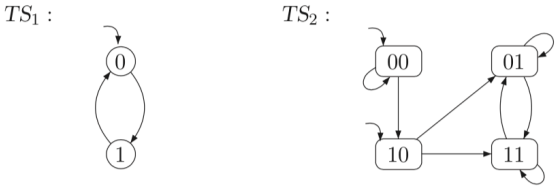
e $L(\langle s_1, s_2 \rangle) = L(s_1) \cup L(s_2)$.

Produto Síncrono de Circuitos

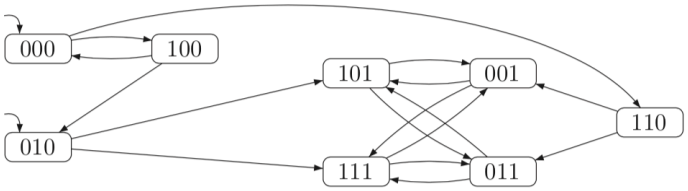




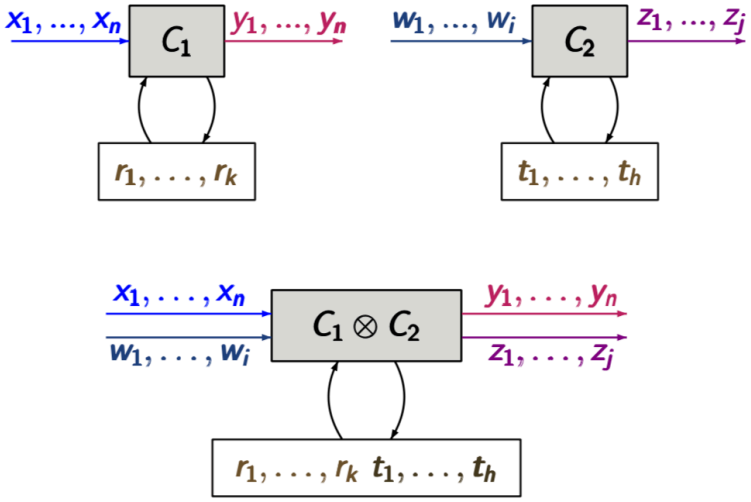
Produto Síncrono de Circuitos



$TS_1 \otimes TS_2$:



Produto Síncrono de Circuitos (geral)



Referências