

# Aula 3

## Paralelismo e comunicação

Normalmente um sistema informático será modelado por sistemas de transições que funcionam em sequência ou, mais frequentemente, em paralelo.

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

Existem muitas maneiras de modelar o paralelismo, algumas possibilidades são:

- Processos intercalados (*interleaving*) (assíncronos). Ex: processos independentes associados a semáforos de tráfego em ruas distintas.
- Comunicação por variáveis partilhadas
- *Handshaking* (uma ação sincroniza os processos)
- Comunicação por canais (filas de mensagens, etc.)

## Concorrência e *interleaving*

- ações de uma componente alternam não-deterministicamente com as restantes componentes. Se  $P$  e  $Q$  são suas componentes

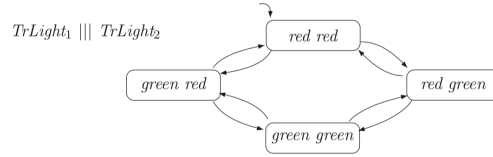
$$PPQQPQQPPPPQQP \dots$$

- um processador executa diversos processos que não comunicam
- deve haver um escalador com uma dada estratégia
- a alternância deve ser justa (*fair*)

## Semáforos de ruas paralelas



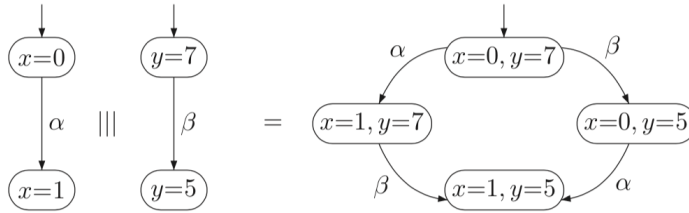
### Sistema de transição $TL_1 ||| TL_2$



- $|||$  operador de intercalagem (*interleaving*)
- ; execução sequencial
- + escolha não determinística
- $Effect(\alpha ||| \beta, \eta) = Effect((\alpha; \beta) + (\beta; \alpha), \eta)$

### Exemplo

Se  $\alpha$  é  $x := x + 1$ ,  $\beta$  é  $y := y - 2$  e  $\eta = \langle x = 0, y = 7 \rangle$  desenhar o diagrama de  $\alpha ||| \beta$ .



Para programas dependentes a ordem interessa, p.e.  $x := x + 1 ||| x := 2x$ .

### Intercalagem de sistemas de transição

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

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

onde a relação de transição é definida por

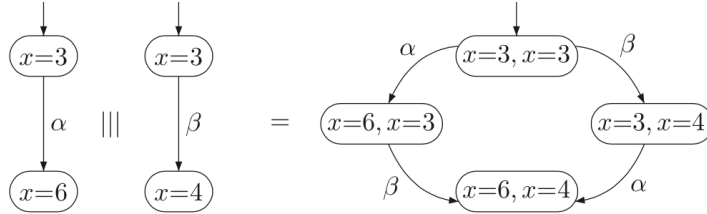
$$\frac{s_1 \xrightarrow{\alpha}_1 s'_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s_2 \rangle} \quad \frac{s_2 \xrightarrow{\alpha}_2 s'_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s'_2 \rangle}$$

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

Para  $PG_i$  com  $Var_1 \cap Var_2 = \emptyset$ ,  $T(PG_1) ||| T(PG_2)$  é o sistema de transições para execução simultânea.

### Comunicação por variáveis partilhadas

Para  $x := x + 1 \parallel x := 2x$  e  $x = 3$  teríamos



tem 3 estados que são inconsistentes!!!

*Solução:* intercalar os grafos de programa e não os sistemas de transição ( $PG_1 \parallel PG_2$ ).

### Intercalagem de grafos de programa

$PG_i = (Loc_i, Act_i, Effect_i, \hookrightarrow_i, Loc_{0,i}, g_{0,i})$  sobre  $Var_i$  para  $i = 1, 2$

$PG_1 \parallel PG_2 = (Loc_1 \times Loc_2, Act_1 \cup Act_2, Effect, \hookrightarrow, Loc_{0,1} \times Loc_{0,2}, g_{0,1} \wedge g_{0,2})$

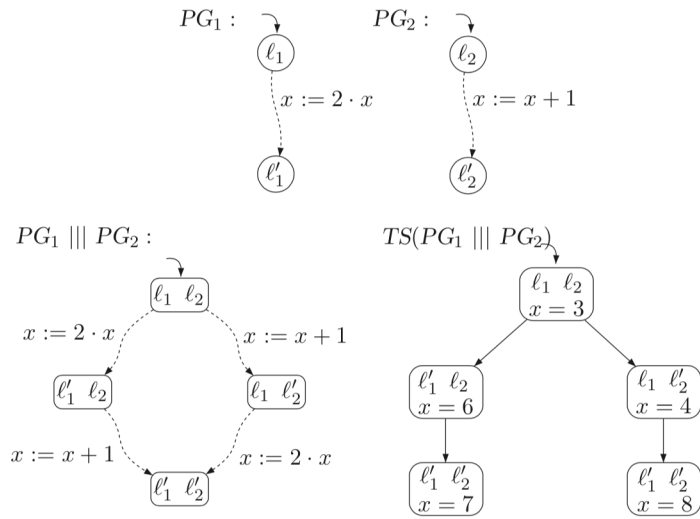
onde  $\hookrightarrow$  é definido por

$$\frac{\ell_1 \xrightarrow{g:\alpha}_1 \ell'_1}{\langle \ell_1, \ell_2 \rangle \xrightarrow{g:\alpha} \langle \ell'_1, \ell_2 \rangle} \quad \frac{\ell_2 \xrightarrow{(g:\alpha)}_2 \ell'_2}{\langle \ell_1, \ell_2 \rangle \xrightarrow{g:\alpha} \langle \ell_1, \ell'_2 \rangle}$$

e  $Effect(\alpha, \eta) = Effect_i(\alpha, \eta)$  se  $\alpha \in Act_i$ .

Notar que as variáveis em  $Var_1 \cap Var_2$  são *variáveis partilhadas*. Sendo *locais* as de  $Var_1 \setminus Var_2$  e  $Var_2 \setminus Var_1$ .

### Exemplo



### Ações críticas

- ações que actuam em variáveis partilhadas designam-se por *críticas*.
- nas partes internas dos processos podemos usar escolhas não-determinísticas
- as ações críticas não podem ser executadas em paralelo, tendo de existir algum tipo de estratégia/escalonamento.

### Atomicidade

- as ações  $\alpha \in Act$  representadas num sistema de transições têm de ser indivisíveis.

### Exemplo

Se o seguinte for uma ação  $\alpha$ , não pode ser divisível

```

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

```

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

### Concorrência e atomicidade

```

proc Inc = while true do if x < 200 then x := x + 1 fi od
proc Dec = while true do if x > 0 then x := x - 1 fi od
proc Reset = while true do if x = 200 then x := 0 fi od

```

Os valores de  $x$  são sempre entre 0 e 200? Não, depende do escalonador:

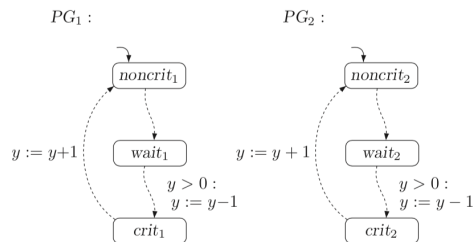
- suponhamos  $x = 200$
- o processo Dec testa  $x$  e passa o controlo ao processo Reset
- O processo Reset testa  $x$  e faz  $x = 0$
- O controlo volta ao processo Dec e fica  $x = -1$

### Exclusão Mútua com semáforos

```

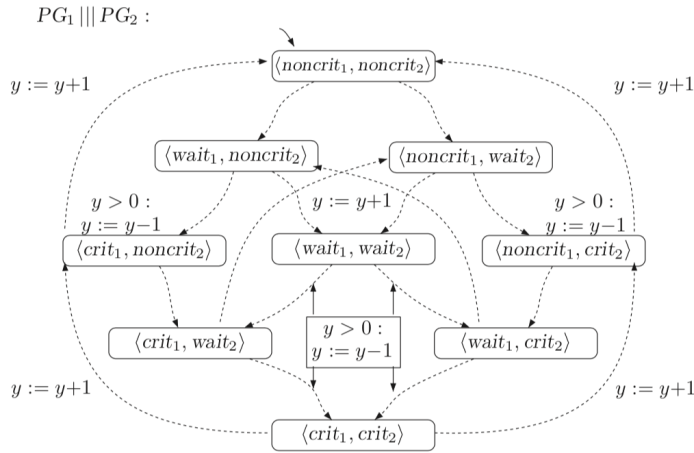
Pi loop forever
  ⋮ (* noncritical actions *)
  request
  critical section
  release
  ⋮ (* noncritical actions *)
end loop

```



$y$  semáforo binário, variável partilhada. Se  $y = 0$  um dos processos está na zona crítica; se  $y = 1$  o semáforo está livre.

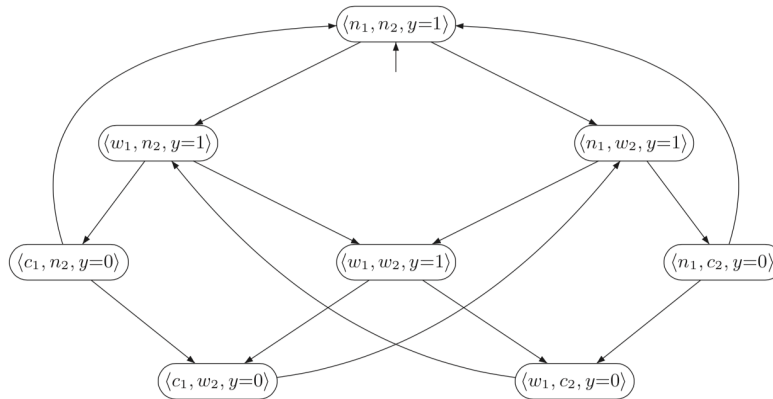
$PG_1 ||| PG_2$



Localização não desejada  $\langle crit_1, crit_2 \rangle$

$T(PG_1 ||| PG_2)$

Muitos estados são não atingíveis...inclusive  $\langle crit_1, crit_2, y = \dots \rangle$  pelo que satisfaz a propriedade de *exclusão mútua*.



Mas, como decidir como sair de  $\langle wait_1, wait_2, y = 1 \rangle$

**Quem entra na zona crítica? Algoritmo de Peterson**

- $P_1, P_2$  processos
- variáveis partilhadas  $b_1, b_2$  e  $x$ , sendo  $dom(b_i) = \{0, 1\}$ .

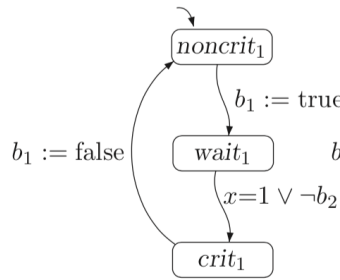
- se ambos os processos querem entrar na zona crítica, isto é  $\langle wait_1, wait_2 \rangle$ ,  $x$  decide qual deles deve entrar
- se  $x = i$  então  $P_i$  pode entrar
- quando  $P_1$  entra em  $wait_1$  faz  $x = 2$  (dá o privilégio a  $P_2$ )
- e simetricamente para  $P_2$
- $b_i = wait_i \vee crit_i$
- acções indivisíveis entre  $\langle \rangle$

```

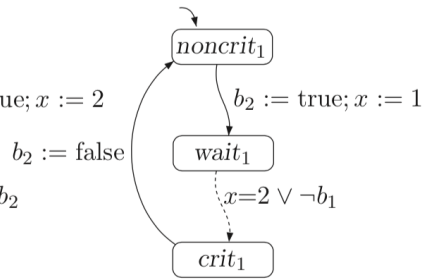
P1 loop forever
  :
  <b1 := true; x := 2> (* noncritical actions *)
  (* request *)
  wait until (x = 1 ∨ ¬b2)
  do critical section od
  b1 := false (* release *)
  :
  (* noncritical actions *)
end loop

```

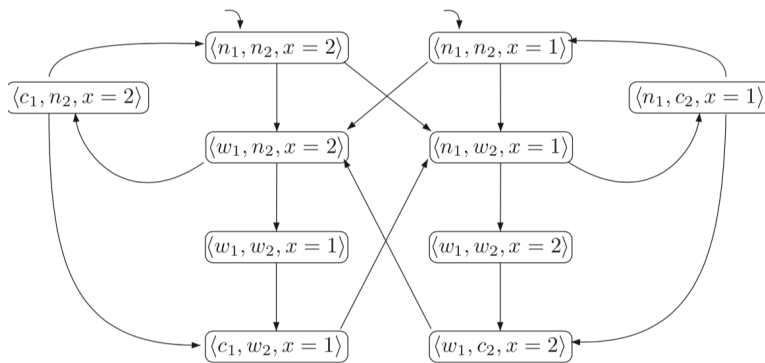
$PG_1 :$



$PG_2 :$



Sistema de transição  $T(PG_1 ||| PG_2)$



- 10 estados at ngiveis, da forma  $\langle loc_1, loc_2, b_1, b_2, x \rangle$  (total 72 estados)
- $b_i$  n o representados
- no in cio  $b_1 = b_2 = 0$ .
- verifica a propriedade de exclus o m tua

### **Handshaking-passagem de mensagens s ncrona**

- processos concorrentes interagem de forma s ncrona
- a informa o trocada pode ser complexa (mas vamos omitir isso)
- ambos os processos partilham um conjunto de a o es  $H \subseteq Act_i$  (*handshake*) e t m de executar uma mesma a o   $\alpha \in H$  simultaneamente
- as restantes a o es podem ser executadas intercaladas (independentes)

### **Handshaking-passagem de mensagens s ncrona**

$T_i = (S_i, Act_i, \longrightarrow_i, I_i, AP_i, L_i)$  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 $||_H$**

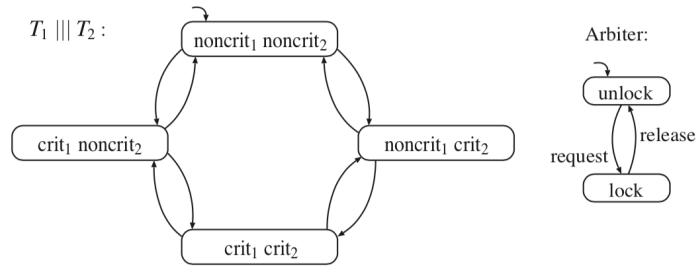
- se  $H = Act_1 \cap Act_2$ , usa-se  $T_1 || T_2$
- $T_1 ||_{\emptyset} T_2 = T_1 || T_2$
- $T_1 ||_H T_2 = T_2 ||_H T_1$  (comutativo)
- em geral  $T_1 ||_H (T_2 ||_{H'} T_3) \neq (T_1 ||_H T_2) ||_{H'} T_3$  (n o associativo)
- Mas se  $H = H'$    associativo

$$T = T_1 ||_H T_2 ||_H T_3 \cdots ||_H T_n \text{ com } H \subseteq Act_1 \cap \cdots \cap Act_n$$

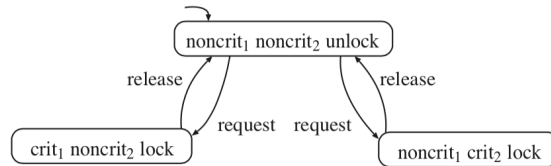


- modela *broadcasting* onde um processo pode enviar informação para vários processos em simultâneo.
- $\parallel_H$  pode-se generalizar a  $T_1 \parallel T_2 \dots \parallel T_n$ , com  $H_{i,j} = Act_i \cap Act_j$  e  $H_{i,j} \cap Act_k = \emptyset$  para  $k \notin \{i, j\}$  (verifica!).

### Exclusão mútua com árbitro

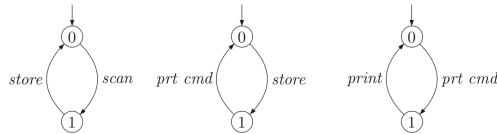


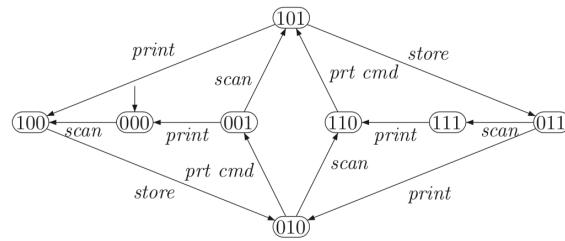
$(T_1 \parallel T_2) \parallel \text{Arbiter}$ :



### Máquina registadora

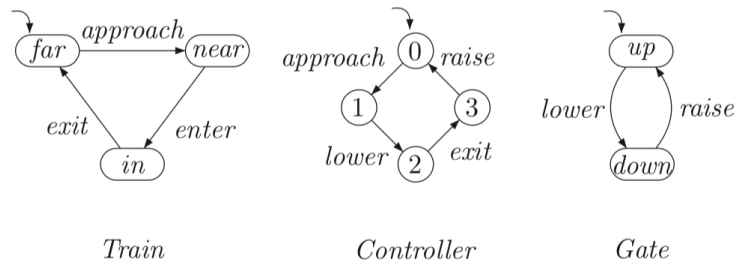
- Leitor de código de barras (código do produto) (BCR)
- Programa de Registo (preço) (BP)
- Impressão de recibo (Printer)
- $BCR \parallel BP \parallel Printer$



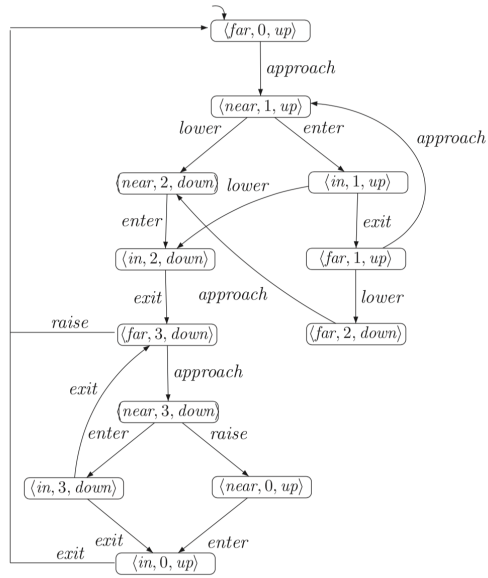


### Passagem de nível

- 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?



## Referências