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.)

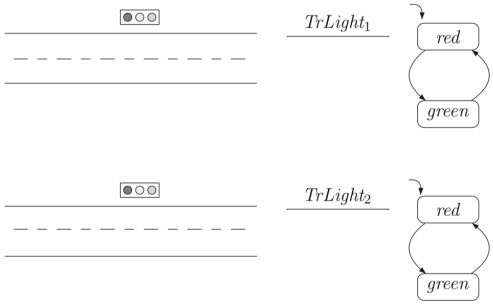
Concorrência e *interleaving*

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

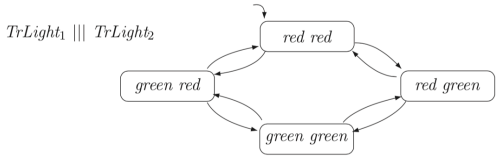
$$PPQQPQQPPPQQP\ldots$$

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

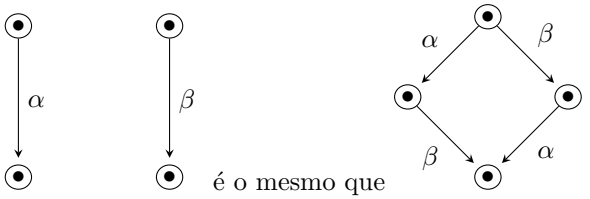
Semáforos de ruas paralelas



Sistema de transição $TL_1|||TL_2$

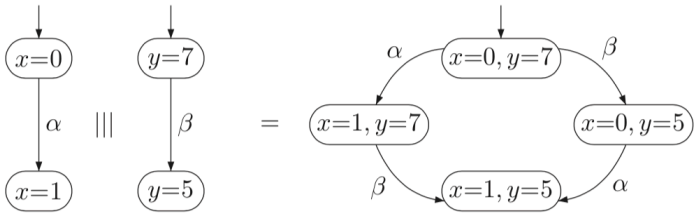


- $|||$ operador de intercalagem (*interleaving*)
- ; execução sequencial
- + escolha não determinística
- $Effect(\alpha|||\beta, \eta) = Effect((\alpha;\beta)+(\beta;\alpha), \eta)$



Exemplo

Se α é $x := x + 1$, β 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

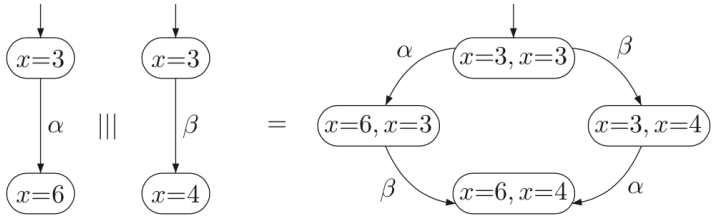
$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2\rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1, s_2'\rangle}$$

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 $x := x + 1|||x := 2x$ e $x = 3$ teríamos



tem 3 estados que são inconsistentes!!!

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

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 $i = 1, 2$

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

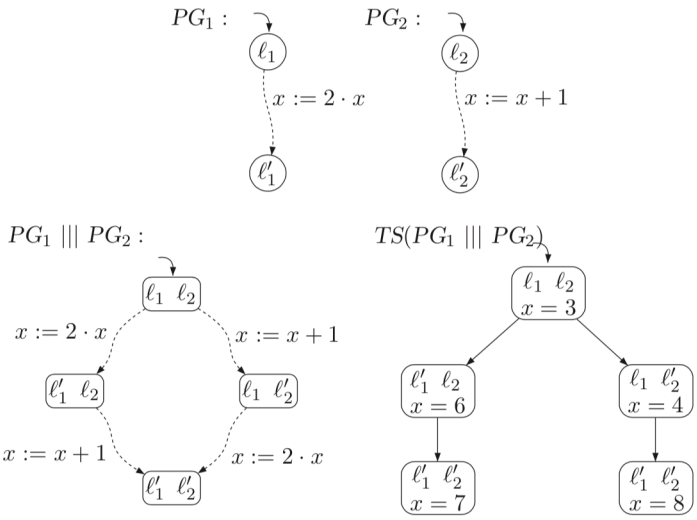
onde $\hookrightarrow$ é definido por

$$\frac{\ell_1 \overset{g:\alpha}{\hookrightarrow}_1 \ell_1'}{\langle \ell_1, \ell_2\rangle \overset{g:\alpha}{\hookrightarrow} \langle \ell_1', \ell_2\rangle} \qquad \frac{\ell_2 \overset{(g:\alpha)}{\hookrightarrow}_2 \ell_2'}{\langle \ell_1, \ell_2\rangle \overset{g:\alpha}{\hookrightarrow} \langle \ell_1, \ell_2'\rangle}$$

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-deterministicas
- 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 α, 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

$$
\begin{aligned}
\textbf{proc } \mathsf{Inc} &= \textbf{while } \text{true } \textbf{do if } x < 200 \textbf{ then } x := x+1 \textbf{ fi od} \\
\textbf{proc } \mathsf{Dec} &= \textbf{while } \text{true } \textbf{do if } x > 0 \textbf{ then } x := x-1 \textbf{ fi od} \\
\textbf{proc } \mathsf{Reset} &= \textbf{while } \text{true } \textbf{do if } x = 200 \textbf{ then } x := 0 \textbf{ fi od}
\end{aligned}
$$

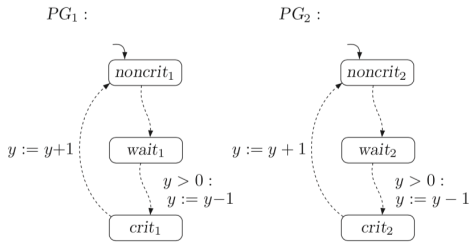
Os valores de x são sempre entre 0 e 200? Não, depende do escalonador:

- 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

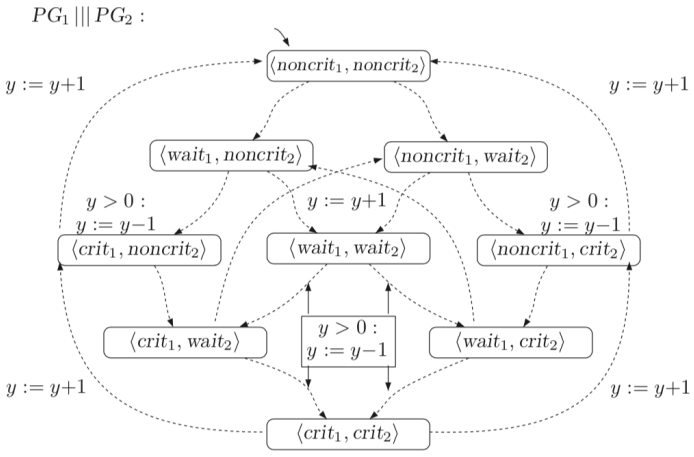
while True **do**
noncricital actions
await $y > 0$ **do** y:=y-1 **od**
critical actions
$y := y + 1$

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



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

$PG_1|||PG_2$



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

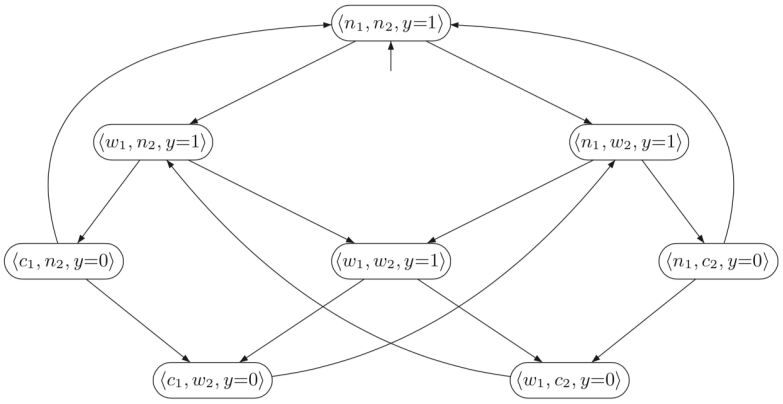
$T(PG_1|||PG_2)$

Só 8 estados são atingiveis:

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

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

$T(PG_1|||PG_2)$

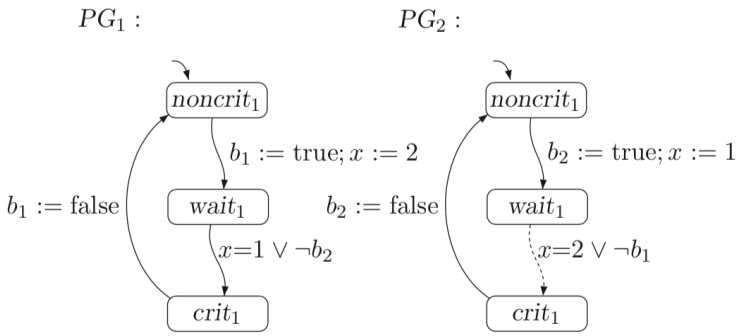


Mas, como decidir como sair de $\langle wait_1, wait_2, y = 1\rangle$

Quem entra na zona crítica? Algoritmo de Peterson

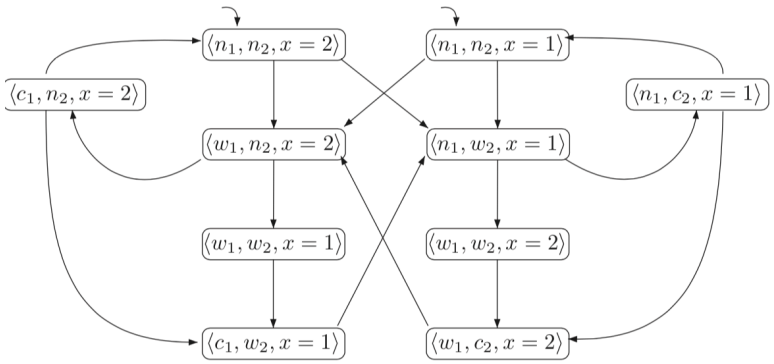
- P_1, P_2 processos
- variáveis partilhadas b_1, b_2 e x, sendo $dom(b_i) = \{0, 1\}$ e $x \in \{1, 2\}$.
- se ambos os processos querem entrar na zona crítica, isto é $\langle wait_1, wait_2\rangle$, x decide qual deles deve entrar
- se $x = i$ então P_i pode entrar
- quando P_1 entra em $wait_1$ faz $x = 2$ (dá o privilégio a P_2)
- e simétricamente para P_2
- b_i indica a localização de P_i, i.e
- $b_i = wait_i \vee crit_i$
- $b_i = true$ quando P_i começa a esperar (wait) e só volta a $b_i = false$ quando P_i sai da zona crítica.
- acções indivísiveis entre $\langle\rangle$

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



$P_1|||P_2$ tem as mesmas localizações

Sistema de transição $T(PG_1|||PG_2)$



- 10 estados atíngiveis, da forma $\langle loc_1, loc_2, b_1, b_2, x\rangle$ (total 72 estados)
- b_i não representados ($= wait_1 \vee crit_1$)
- no inicio $b_1 = b_2 = 0$.
- verifica a propriedade de exclusão mútua

Referências