## Programação Concorrente - Exercícios 6

CCS com passagem de valores

1. Para cada um dos seguintes processos $P$ calcula o LTS $[\![P]\!]$
   (a) $(a!21.0|a?x.b!(x*2).0)\backslash\{a\}$
   (b) $(a!42.0|a?x.Sender[x])\backslash\{a\}$

2. Considera o seguinte programa em $CCS_{vp}$ e calcula o seu LTS usando as regras.

$$\begin{aligned}
Sender &:= put?x.send!x.Sending[x] \\
Sending[x] &:= receiveAck?.Sender + receiveNAck?.send!x.Sending[x] \\
Receiver &:= receive?x.get!x.sendAck!.Receiver + \\
&\quad gargled?.sendNAck!.Receiver \\
Medium &:= send?x.(receive!x.Medium + i.garbled!.Medium) \\
AckMedium &:= sendAck?.receiveAck!.AckMedium + \\
&\quad sendNAck?.receivedNAck!.AckMedium \\
DupMedium &:= Medium|AckMedium \\
Protocol &:= (Sender \mid Receiver \mid DupMedium)\backslash \\
&\quad \{send, receive, sendAck, receiveAck, \\
&\quad receiveNAck, sendNAck, garbled\}
\end{aligned}$$

   Calcula $[\![(Protocol|put!1.put!2.put!3.put!4.0)\backslash\{put\}]\!]$. Restringe os valores $rangeR := 0..9$.

3. Dado

$$\begin{aligned}
Fac[n,j] &:= when(j>0)i.Fac[n*j, j-1] \\
&\quad + when(j==0)println!n.0
\end{aligned}$$

   Calcular $[\![Fac[1,5]]\!]$.

4. Considera a seguinte máquina de vendas que aceita moedas de 1, 2 e 5 euros. Quando o valor é suficiente para tirar um café a máquina dá o café e dá troco se existir. Cada café custa 5 euros.

$$\begin{aligned}
Machine[b] &:= when(b<5)coin?c.Machine[b+c] + when(b\geq 5)coffee!.ReturnMachine[b-1] \\
ReturnMachine[b] &:= when(b>0)change!.ReturnMachine[b-1] + Machine[0] \\
User &:= coin!2.coin!2.coin!2.coffee?.change?.0
\end{aligned}$$

   Calcula $[\![(Machine[0]|User)\backslash\{coin, change, coffee\}]\!]$.

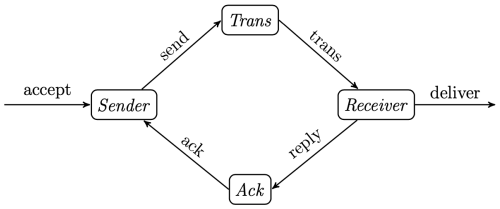
5. Dado

$$\begin{aligned}
Cell[rd,wr,x] &:= rd!x.Cell[rd,wr,x] + wr?y.Cell[rd,wr,y] \\
Cells &:= Cell[rdA, wrA, 0]|Cell[rdB, wrB, 0] \\
Serve &:= mult?.rdA?x:R.rdB?y:R.IterMult[0,x,y] \\
IterMult[z,x,y] &:= when(x>0)i.IterMult[z+y, x-1, y] \\
&\quad + when(x==0)println!z.Serve \\
Use &:= wrA!7.wrB!5.mult!.0
\end{aligned}$$

Calcular $[\![(Cells|Serve|Use)\backslash\{rdA, wrA, rdB, wrB, mult\}]\!]$.

6. Considera o exercício 9 da folha 5 com $CCS_{vp}$.

7. (Alternating bit protocol) Alternating-Bit Protocol (ABP) é um protocolo entre um emissor (Sender) e um receptor (Receiver) sobre um canal não confiável. Isto é conseguido com a retransmissão de mensagens se necessário.




Funciona do seguinte modo. Para o envio da mensagem :

- Aceita mensagem para envio
- Envia-a com o bit $b$ pelo Trans e activa um temporizador
- Quando este atinge "timeout" reenvia $b$
- Ao receber a confirmação $b$ pelo canal Ack, continua com uma nova mensagem e $1 - b$.
- Ignora qualquer confirmação no canal Ack de $1 - b$..

Para o receptor é semelhante:

- Recebe e entrega a mensagem
- Envia confirmação com o bit $b$ e activa um temporizador
- Quando atinge "timeout" reenvia $b$
- Na recepção de uma nova mensagem $1 - b$ entrega-a e prossegue com $1 - b$Receiving a message $1 - b$ delivers it and continues with $1 - b$.
- Ignora mensagens etiquetadas com $b$Ignores any message in Trans with $b$

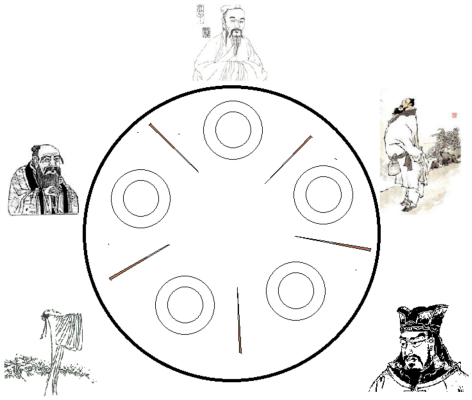
Os processos Sender e Receiver obtêm-se pela composição paralela com o temporizador (Timer) e ignorando as ações do Timer (internas) Os processos Trans e Ack têm de ser não deterministicos e simular o meio não confiável.

(a) Implementa ABP em CCS considerando apenas o bit de controlo (e não os dados).

(b) Escreve uma especificação de comunicação e compara com o ABP.

(c) Verifica se existem deadlocks

8. (Jantar dos Filósofos)Cinco filósofos estão sentados à volta duma mesa com uma taça de arroz no meio e um pauzinho colocado entre cada um dos filósofos. Não falam e apenas têm duas actividades: pensar e comer, alternadamente. Cada filósofo precisa de dois pauzinhos para comer arroz e só pode usar os adjacentes se estiverem livres.



(a) Escreve um processo CCS para modelar o comportamento dos filósofos e os pauzinhos. Podes começar por considerar só 2. .

(b) Verifica se existe deadlock.

(c) Os pauzinhos podem ser ordenados de modo que um filósofo tenha que pegar num pauzinho de orem menor antes de um de ordem maior, se disponível. Escreve um processo CCS para este caso..