5. Bissimulação fraca e Congruência Observável

José Proença
(slides mainly from Nelma Moreira)

Concurrent programming (CC3040) 2025/2026

CISTER – U.Porto, Porto, Portugal

https://fm-dcc.github.io/cp2526

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

CISTER - Research Centre in
Real-Time & Embedded
Computing Systems

Bissimilaridade

Seja $TS = (S, \longrightarrow, s_0)$. Uma relação $R \subseteq S \times S$ é uma bissimulação se sempre que $(s,t) \in R$ (ou $s\ R\ t$) e $\alpha \in Act$:

- se $s \xrightarrow{\alpha} s'$ então existe t' tal que $t \xrightarrow{\alpha} t'$ e $(s',t') \in R$
- se $t \xrightarrow{\alpha} t'$ então existe s' tal que $s \xrightarrow{\alpha} s'$ e $(s',t') \in R$.

Dois estados s e t são bissimilares $s \sim t$, se existe uma bissimulação R tal que $(s,t) \in R$.

Isto é

$$\sim = \bigcup_{\text{R é bissimulação}} R$$

A relação $\sim$ chama-se bissimilaridade

- relação de equivalência
- relação de congruência
- mesmo conjunto de traços
- seja sensível a deadlocks

$$
\begin{aligned}
P + Q &\sim Q + P \\
P|Q &\sim Q|P \\
P + 0 &\sim P \\
P|0 &\sim P \\
(P + Q) + R &\sim P + (Q + R) \\
(P|Q)|R &\sim P|(Q|R) \\
\alpha.(P + Q) &\not\sim \alpha.P + \alpha.Q
\end{aligned}
$$

Mas como calcular a relação $\sim$

Dado $TS = (S, \longrightarrow, s_0)$, como o maior ponto fixo de uma função (FR) monótona definida num reticulado completo $(2^{S \times S}, \subseteq)$), i.e., $FR(B) = B$. Sem prova: sendo $R \subseteq S \times S$

$$FR(R) = \{(s,t) \mid s \xrightarrow{\alpha} s' \implies \exists t'\ t \xrightarrow{\alpha} t' \wedge (s',t') \in R \\ \wedge\ t \xrightarrow{\alpha} t' \implies \exists s's \xrightarrow{\alpha} s' \wedge (s',t') \in R\}$$

$$FR(S \times S) \supseteq FR^2(S \times S) \supseteq \cdots$$

ou ver algoritmo em Pseuco Book

$$Buffer \quad := \quad put?.get!.Buffer$$

$$Buffer \quad := \quad put?.get!.Buffer$$

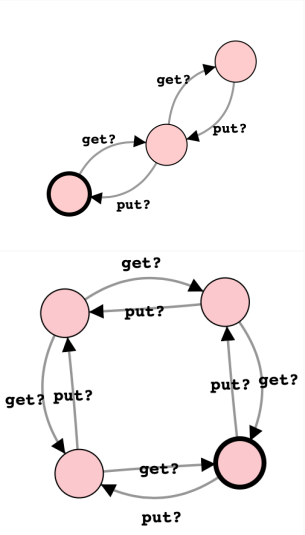
$$\begin{aligned} Buffer0 &:= put?.Buffer1 \\ Buffer1 &:= put?.Buffer2 + get!.Buffer0 \\ Buffer2 &:= get?.Buffer1 \end{aligned}$$

Será que $Buffer0 \sim (Buffer|Buffer)$?

$$Buffer \quad := \quad put?.get!.Buffer$$

$$\begin{aligned} Buffer0 &:= put?.Buffer1 \\ Buffer1 &:= put?.Buffer2 + get!.Buffer0 \\ Buffer2 &:= get?.Buffer1 \end{aligned}$$

Será que $Buffer0 \sim (Buffer|Buffer)$?

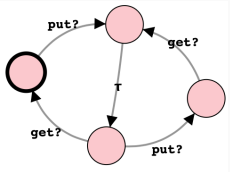


$$Buffer := put?.get!.Buffer$$

$$BufferL := put?.pass!.BufferL$$
$$BufferR := pass?.get!.BufferR$$

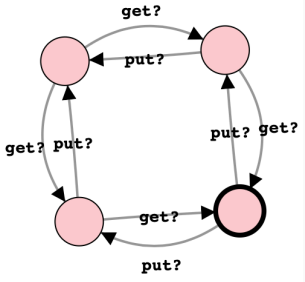
Será que $(BufferL|BufferR)\backslash\{pass!, pass?\} \sim (Buffer|Buffer)$?

$$Buffer := put?.get!.Buffer$$



$$BufferL := put?.pass!.BufferL$$
$$BufferR := pass?.get!.BufferR$$

Será que $(BufferL|BufferR)\backslash\{pass!, pass?\} \sim (Buffer|Buffer)$?



Bissimulação Fraca

Ignora transições por τ.

Seja $TS = (S, \longrightarrow, s_0)$. Uma relação $R \in S \times S$ é uma bissimulação fraca se $(s, t) \in R$ (ou $s\ R\ t$) implica para todo $\alpha \in Act$:

- se $s \xrightarrow{\alpha} s'$ então existe t' tal que $t \stackrel{\alpha}{\Rightarrow} t'$ e $(s', t') \in R$
- se $t \xrightarrow{\alpha} t'$ então existe s' tal que $s \stackrel{\alpha}{\Rightarrow} s'$ e $(s', t') \in R$.

Bissimulação Fraca

Seja $TS = (S, \longrightarrow, s_0)$. Uma relação $R \in S \times S$ é uma bissimulação fraca se $(s,t) \in R$ (ou $s\ R\ t$) implica para todo $\alpha \in Act$:

- se $s \xrightarrow{\alpha} s'$ então existe t' tal que $t \stackrel{\alpha}{\Rightarrow} t'$ e $(s', t') \in R$
- se $t \xrightarrow{\alpha} t'$ então existe s' tal que $s \stackrel{\alpha}{\Rightarrow} s'$ e $(s', t') \in R$.

Transições fracas

- $s \stackrel{\tau}{\Rightarrow} s'$ sse $\exists n \geq 0 : s \stackrel{\tau^n}{\rightsquigarrow} s'$
- $s \stackrel{a}{\Rightarrow} s'$ sse $\exists s'', s''' \in S : s \stackrel{\tau}{\Rightarrow} s'' \xrightarrow{a} s''' \stackrel{\tau}{\Rightarrow} s'$

Nota que

- $\longrightarrow \subseteq \Rightarrow$
- na definição de b.f. podemos usar sempre $\Rightarrow$.
- $s \stackrel{\tau}{\Rightarrow} s$ para todo o s

Seja $TS = (S, \longrightarrow, s_0)$. Uma relação $R \in S \times S$ é uma bissimulação fraca se $(s, t) \in R$ (ou $s\ R\ t$) implica para todo $\alpha \in Act$:

- se $s \xrightarrow{\alpha} s'$ então existe t' tal que $t \stackrel{\alpha}{\Rightarrow} t'$ e $(s', t') \in R$
- se $t \xrightarrow{\alpha} t'$ então existe s' tal que $s \stackrel{\alpha}{\Rightarrow} s'$ e $(s', t') \in R$.

Bissimilaridade fraca

Dois estados s e t são fracamente bissimilares, e escreve-se $s \approx t$, se existe uma bissimulação fraca R que contêm (s, t), i.e., $(s, t) \in R$.

Exemplo

Mostrar que

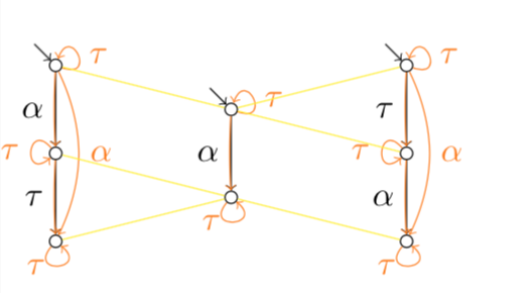
- $a.\tau.0 \approx a.0 \approx \tau.a.0$
- $\tau.(a.0 + \tau.a.0) \approx a.0$

Exemplo

Mostrar que

- $a.\tau.0 \approx a.0 \approx \tau.a.0$
- $\tau.(a.0 + \tau.a.0) \approx a.0$

$a.\tau.0 \approx a.0 \approx \tau.a.0$



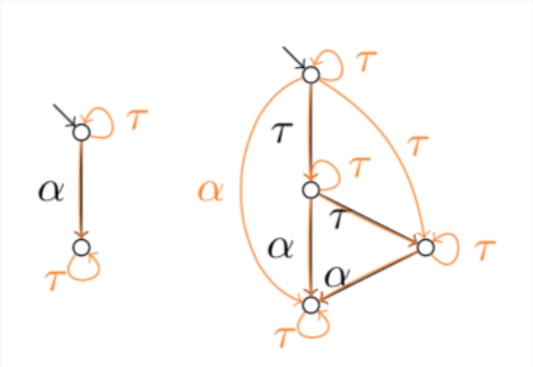
A laranja a relação $\Rightarrow$ e a amarelo a bissimulação R.

Exemplo

Mostrar que

- $a.\tau.0 \approx a.0 \approx \tau.a.0$
- $\tau.(a.0 + \tau.a.0) \approx a.0$

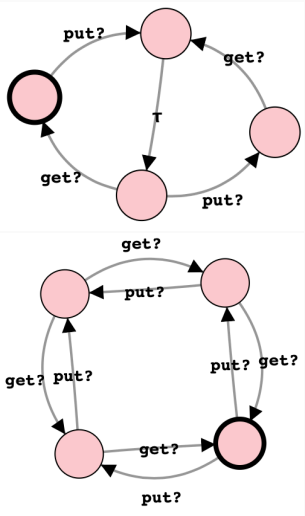
$\tau.(a.0 + \tau.a.0) \approx a.0$



Exemplo dos Buffers

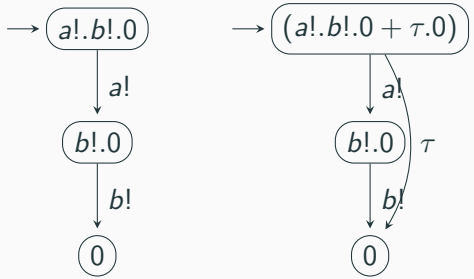
$$
\begin{aligned}
Buffer &:= put?.get?.Buffer \\
BufferL &:= put?.pass!.BufferL \\
BufferR &:= pass?.get?.BufferR
\end{aligned}
$$

$$(BufferL|BufferR)\backslash\{pass!, pass?\} \approx (Buffer|Buffer)$$



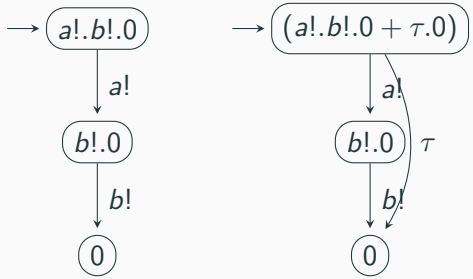
Processos não fracamente bissimilares

$$(a!.b!.0) \not\approx (a!.b!.0 + \tau.0)$$



Processos não fracamente bissimilares

$(a!.b!.0) \not\approx (a!.b!.0 + \tau.0)$



- A: $a!.b!.0 + \tau.0 \xrightarrow{\tau} 0$
- D: $a!.b!.0 \stackrel{\tau}{\Rightarrow} a!.b!.0$
- A: $a!.b!.0 \xrightarrow{a!} b!.0$
- D: não pode jogar com $a!$ porque 0 não tem transição fraca com $a!$.

Temos que

$$\approx = \bigcup_{\text{R é bissimulação fraca}} R$$

Teorema

A relação $\approx$ é de equivalência.

Propriedades de $\approx$

Temos que

$$\approx = \bigcup_{\text{R é bissimulação fraca}} R$$

Notar que se R é uma bissimulação fraca R^{-1} também é.

Temos que

$$\approx = \bigcup_{\text{R é bissimulação fraca}} R$$

Teorema

A relação $\approx$ é maior bissimulação fraca .

Mostrar que $\approx$ é uma bissimulação fraca.

Temos que

$$\approx = \bigcup_{\text{R é bissimulação fraca}} R$$

Teorema

A relação $\approx$ é maior bissimulação fraca .

Teorema

A bissimilaridade fraca é estritamente mais grosseira que a bissimilaridade forte, i.e $\sim \subsetneq \approx$.

Temos que

$$\approx = \bigcup_{\text{R é bissimulação fraca}} R$$

Teorema

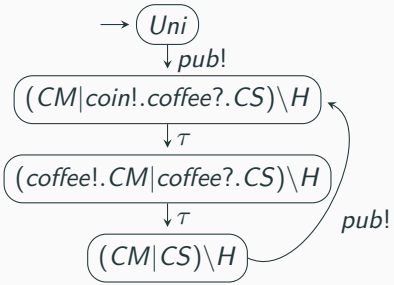
A bissimilaridade fraca é estritamente mais grosseira que a bissimilaridade forte, i.e $\sim \subsetneq \approx$.

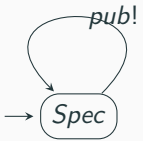
Cada bissimulação forte é uma bissimulação fraca. E vimos exemplos de processos que não bissimilares fortes mas que são bissimilares fracos. A bissimilaridade fraca também é chamada de *equivalência observável*.

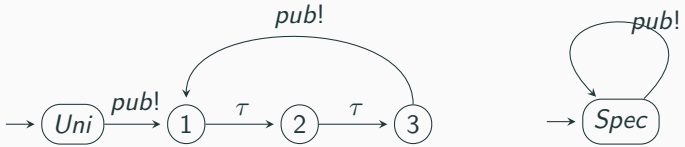
Será que $Spec \approx Uni$?

Seja $Spec := pub!.Spec$ e

$$\begin{aligned} CM &:= coin?.coffee!.CM \\ CS &:= pub!.coin!.coffee?.CS \\ Uni &:= (CM|CS)\backslash\{coin, coffee\} \end{aligned}$$







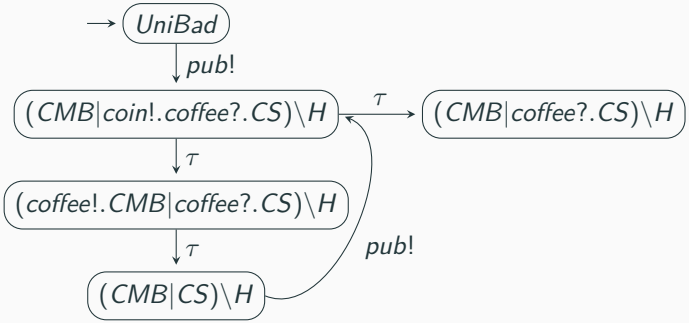
Jogo onde o defesa ganha. Configuração inicial $(Uni, Spec)$

- A: $Spec \xrightarrow{pub!} Spec$
- D: $Uni \stackrel{pub!}{\Rightarrow} 3$ $(3, Spec)$
- A: $3 \xrightarrow{pub!} 1$
- D: $Spec \stackrel{pub!}{\Rightarrow} Spec$ $(1, Spec)$
- A: $1 \xrightarrow{\tau} 2$
- D: $Spec \stackrel{\tau}{\Rightarrow} Spec$ $(2, Spec)$
- A: $2 \xrightarrow{\tau} 3$
- D: $Spec \stackrel{\tau}{\Rightarrow} Spec$ $(3, Spec)$ (ciclo detectado)

Será que *Uni* $\approx$ *Unibad*?

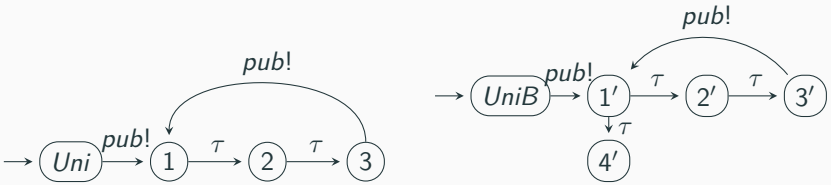
Seja

$$
\begin{array}{rcl}
CM & := & coin?.coffee!.CM \\
CMB & := & coin?.coffee!.CMB + coin?.CMB \\
CS & := & pub!.coin!.coffee?.CS \\
Uni & := & (CM|CS)\backslash\{coin, coffee\} \\
UniBad & := & (CMB|CS)\backslash\{coin, cofee\}
\end{array}
$$

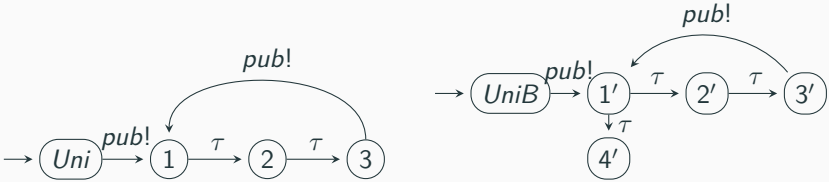


Neste caso $Uni \not\approx UniBad$, o que é bom porque *UniBad* entra em deadlock e *Uni* não.

Jogo de bissimilaridade fraca ($Uni, UniBad$)



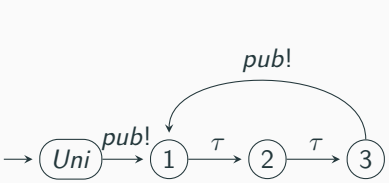
Jogo de bissimilaridade fraca $(Uni, UniBad)$

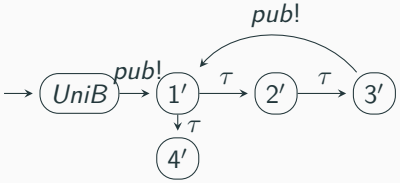


O atacante ganha. Configuração inicial $(Uni, UniBad)$

- A: $UniBad \xrightarrow{pub!} 1'$
- D: $Uni \stackrel{pub!}{\Rightarrow} 3 \qquad (3, 1')$
- A: $1' \xrightarrow{\tau} 4'$
- D: $3 \stackrel{\tau}{\Rightarrow} 3 \qquad (3, 4')$
- A: $3 \xrightarrow{pub!} 1$
- D: não pode jogar e perde

Jogo de bissimilaridade fraca $(Uni, UniBad)$





O atacante ganha. Configuração inicial $(Uni, UniBad)$

- A: $UniBad \xrightarrow{pub!} 1'$
- D: $Uni \overset{pub!}{\Rightarrow} 3 \qquad (3, 1')$
- A: $1' \xrightarrow{\tau} 4'$
- D: $3 \overset{\tau}{\Rightarrow} 3 \qquad (3, 4')$
- A: $3 \xrightarrow{pub!} 1$
- D: não pode jogar e perde

Mas neste caso temos que ver também o qual a estratégia do atacante quando o defesa na primeira jogada considera outras alternativas (1 ou 2). Mostra que neste casos o atacante também consegue ganhar.

A Bissimilaridade Fraca não é uma Congruência

Para todo $P, Q \in CCS$ se $P \approx Q$ e $\alpha \in Act$, $R \in CCS$ e $H \subseteq Com$,

$$\begin{aligned} \alpha.P &\approx \alpha.Q \\ P|R &\approx Q|R \\ R|P &\approx R|Q \\ P\backslash H &\approx Q\backslash H \end{aligned}$$

A Bissimilaridade Fraca não é uma Congruência

Para todo $P, Q \in CCS$ se $P \approx Q$ e $\alpha \in Act$, $R \in CCS$ e $H \subseteq Com$,

$$\begin{aligned} \alpha.P &\approx \alpha.Q \\ P|R &\approx Q|R \\ R|P &\approx R|Q \\ P\backslash H &\approx Q\backslash H \end{aligned}$$

mas não implica que

$$\begin{aligned} P + R &\approx Q + R \\ R + P &\approx R + Q \end{aligned}$$

A Bissimilaridade Fraca não é uma Congruência

Para todo $P, Q \in CCS$ se $P \approx Q$ e $\alpha \in Act$, $R \in CCS$ e $H \subseteq Com$,

$$\begin{aligned} \alpha.P &\approx \alpha.Q \\ P|R &\approx Q|R \\ R|P &\approx R|Q \\ P\backslash H &\approx Q\backslash H \end{aligned}$$

mas não implica que

$$\begin{aligned} P + R &\approx Q + R \\ R + P &\approx R + Q \end{aligned}$$

Contraexemplo: $P = \tau.a!.0$, $Q = a!.0$ e $R = b!.0$.

A Bissimilaridade Fraca não é uma Congruência

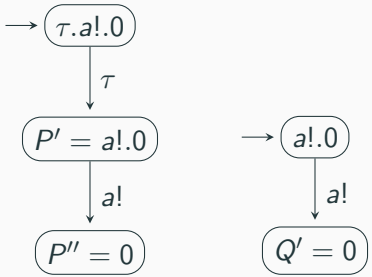
Contraexemplo: $P = \tau.a!.0$, $Q = a!.0$ e $R = b!.0$.

- Mostrar que $P \approx Q$ considerando a relação $\mathcal{R} = \{(P, Q), (P', Q), (P'', Q')\}$ onde $P' = a!.0$ e $P'' = Q' = 0$ (mas em sistemas diferentes).

A Bissimilaridade Fraca não é uma Congruência

Contraexemplo: $P = \tau.a!.0$, $Q = a!.0$ e $R = b!.0$.

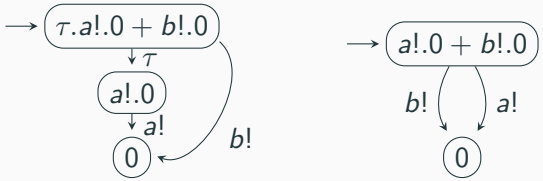
- Mostrar que $P \approx Q$ considerando a relação $\mathcal{R} = \{(P, Q), (P', Q), (P'', Q')\}$ onde $P' = a!.0$ e $P'' = Q' = 0$ (mas em sistemas diferentes).



A Bissimilaridade Fraca não é uma Congruência

Contraexemplo: $P = \tau.a!.0$, $Q = a!.0$ e $R = b!.0$.

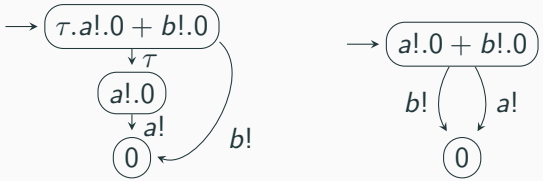
- Mostrar que $P + R \not\approx Q + R$.



A Bissimilaridade Fraca não é uma Congruência

Contraexemplo: $P = \tau.a!.0$, $Q = a!.0$ e $R = b!.0$.

- Mostrar que $P + R \not\approx Q + R$.

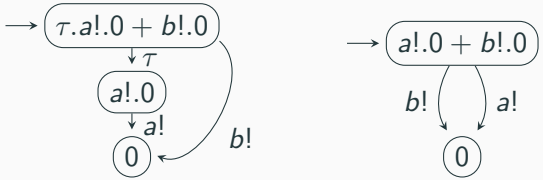


- A: $\tau.a!.0 + b!.0 \xrightarrow{\tau} a!.0$
- D: $a!.0 + b!.0 \stackrel{\tau}{\Rightarrow} a!.0 + b!.0$
- A: $a!.0 + b!.0 \xrightarrow{b!} 0$
- D: não pode jogar porque não há nenhuma transição fraca de $a!.0$ por $b!$

A Bissimilaridade Fraca não é uma Congruência

Contraexemplo: $P = \tau.a!.0$, $Q = a!.0$ e $R = b!.0$.

- Mostrar que $P + R \not\approx Q + R$.



O problema está nos processos não guardados!

Congruência para a Escolha

Dois processos $P, Q \in CCS$ são congruentes para a escolha, e escreve-se $P \approx^{+} Q$ sse para todos os $R \in CCS$

$$P + R \approx Q + R$$

.

Congruência para a Escolha

Dois processos $P, Q \in CCS$ são congruentes para a escolha, e escreve-se $P \approx^{+} Q$ sse para todos os $R \in CCS$

$$P + R \approx Q + R$$

.

Pode-se provar que

- $\approx^{+} \subsetneq \approx$ (tomando $R = 0$)
- $\approx^{+}$ é uma congruência e é a mais grosseira contida em $\approx$

Congruência para a Escolha

Dois processos $P, Q \in CCS$ são congruentes para a escolha, e escreve-se $P \approx^{+} Q$ sse para todos os $R \in CCS$

$$P + R \approx Q + R$$

.

Pode-se provar que

- $\approx^{+} \subsetneq \approx$ (tomando $R = 0$)
- $\approx^{+}$ é uma congruência e é a mais grosseira contida em $\approx$

Embora seja esta a noção de igualdade de processos que se pretende, não é fácil de usar por causa de *para todos os* $R \in CCS$. Vamos ver uma definição que é equivalente mas mais perto das outras definições de bissimulação.

Congruência Observável

Dois processos $P, Q \in CCS$ são congruentes para a observação, e escreve-se $P \simeq Q$ sse para todos $\alpha \in Act$ verifica-se que:

- se $P \xrightarrow{\alpha} P'$ então existe Q' tal que $Q \stackrel{\tau}{\Rightarrow} \xrightarrow{\alpha} \stackrel{\tau}{\Rightarrow} Q'$ e $P' \approx Q'$.
- se $Q \xrightarrow{\alpha} Q'$ então existe P' tal que $P \stackrel{\tau}{\Rightarrow} \xrightarrow{\alpha} \stackrel{\tau}{\Rightarrow} P'$ e $P' \approx Q'$.

Congruência Observável

Dois processos $P, Q \in CCS$ são congruentes para a observação, e escreve-se $P \simeq Q$ sse para todos $\alpha \in Act$ verifica-se que:

- se $P \xrightarrow{\alpha} P'$ então existe Q' tal que $Q \stackrel{\tau}{\Rightarrow} \xrightarrow{\alpha} \stackrel{\tau}{\Rightarrow} Q'$ e $P' \approx Q'$.
- se $Q \xrightarrow{\alpha} Q'$ então existe P' tal que $P \stackrel{\tau}{\Rightarrow} \xrightarrow{\alpha} \stackrel{\tau}{\Rightarrow} P'$ e $P' \approx Q'$.

A diferença entre $\approx$ e $\simeq$ está que os passos internos iniciais não podem ser simulados por não haver movimento (lacete de τ).

Congruência Observável

Dois processos $P, Q \in CCS$ são congruentes para a observação, e escreve-se $P \simeq Q$ sse para todos $\alpha \in Act$ verifica-se que:

- se $P \xrightarrow{\alpha} P'$ então existe Q' tal que $Q \stackrel{\tau}{\Rightarrow} \xrightarrow{\alpha} \stackrel{\tau}{\Rightarrow} Q'$ e $P' \approx Q'$.
- se $Q \xrightarrow{\alpha} Q'$ então existe P' tal que $P \stackrel{\tau}{\Rightarrow} \xrightarrow{\alpha} \stackrel{\tau}{\Rightarrow} P'$ e $P' \approx Q'$.

A diferença entre $\approx$ e $\simeq$ está que os passos internos iniciais não podem ser simulados por não haver movimento (lacete de τ).

Teorema

Para todos os $P, Q \in CCS$ $P \simeq Q$ sse $P \approx^{+} Q$.

Exemplos

- $\tau.a.0 \not\approx a.0$
- $P|\tau.Q \not\approx \tau.(P|Q)$

Igualdade de comportamento

- Dois processos $P, Q \in CCS$ tem o mesmo comportamento sempre que $P \simeq Q$.

Igualdade de comportamento

- Dois processos $P, Q \in CCS$ tem o mesmo comportamento sempre que $P \simeq Q$.
- É chamada a igualdade de comportamento

Igualdade de comportamento

- Dois processos $P, Q \in CCS$ tem o mesmo comportamento sempre que $P \simeq Q$.
- É chamada a igualdade de comportamento
- as propriedades pretendidas:

Igualdade de comportamento

- Dois processos $P, Q \in CCS$ tem o mesmo comportamento sempre que $P \simeq Q$.
- É chamada a igualdade de comportamento
- as propriedades pretendidas:
  - relação de equivalência

Igualdade de comportamento

- Dois processos $P, Q \in CCS$ tem o mesmo comportamento sempre que $P \simeq Q$.
- É chamada a igualdade de comportamento
- as propriedades pretendidas:
  - relação de equivalência
  - relação de congruência

Igualdade de comportamento

- Dois processos $P, Q \in CCS$ tem o mesmo comportamento sempre que $P \simeq Q$.
- É chamada a igualdade de comportamento
- as propriedades pretendidas:
  - relação de equivalência
  - relação de congruência
  - ter o mesmo conjunto de traços fracos

Igualdade de comportamento

- Dois processos $P, Q \in CCS$ tem o mesmo comportamento sempre que $P \simeq Q$.
- É chamada a igualdade de comportamento
- as propriedades pretendidas:
  - relação de equivalência
  - relação de congruência
  - ter o mesmo conjunto de traços fracos
  - seja sensível a deadlocks

Igualdade de comportamento

- Dois processos $P, Q \in CCS$ tem o mesmo comportamento sempre que $P \simeq Q$.
- É chamada a igualdade de comportamento
- as propriedades pretendidas:
  - relação de equivalência
  - relação de congruência
  - ter o mesmo conjunto de traços fracos
  - seja sensível a deadlocks
- $\sim \subsetneq \simeq \subseteq \approx$

Igualdade de comportamento

- Dois processos $P, Q \in CCS$ tem o mesmo comportamento sempre que $P \simeq Q$.
- É chamada a igualdade de comportamento
- as propriedades pretendidas:
  - relação de equivalência
  - relação de congruência
  - ter o mesmo conjunto de traços fracos
  - seja sensível a deadlocks
- $\sim \subsetneq \simeq \subseteq \approx$
- em vez de $\simeq$ usa-se $=$.

Regras algébricas de $=$ (τ-Leis de Milner)

Para além das satisfeitas por $\sim$ temos

$$\begin{aligned} \alpha.\tau.P &= \alpha.P \\ P + \tau.P &= \tau.P \\ \alpha.(P + \tau.Q) &= \alpha.(P + \tau.Q) + \alpha.Q \end{aligned}$$

Nota: as regras algébricas permitem provar a igualdade de processos pela aplicação das regras (em especial em CCS_0) ou considerá-las axiomas num sistema de dedução.

Representantes Minimais

- Para cada LTS podemos considerar o representante minimal o menor LTS cujo comportamento é igual ao LTS dado.

Representantes Minimais

- Para cada LTS podemos considerar o representante minimal o menor LTS cujo comportamento é igual ao LTS dado.
- O representante minimal é único a menos de isomorfismo se for finito por estados (regular).

Representantes Minimais

- Para cada LTS podemos considerar o representante minimal o menor LTS cujo comportamento é igual ao LTS dado.
- O representante minimal é único a menos de isomorfismo se for finito por estados (regular).
- É obtido pelo quociente induzido pela relação de equivalência ($\simeq$).

Representantes Minimais

- Para cada LTS podemos considerar o representante minimal o menor LTS cujo comportamento é igual ao LTS dado.
- O representante minimal é único a menos de isomorfismo se for finito por estados (regular).
- É obtido pelo quociente induzido pela relação de equivalência ($\simeq$).
- Cada classe de equivalência é um estado

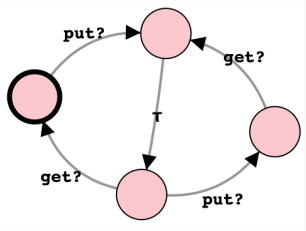
Como obter o representante minimal

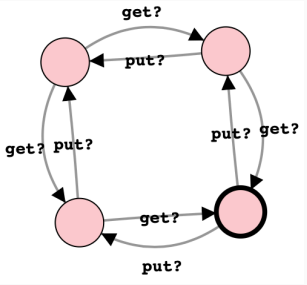
Dado um processo P do CCS estados-finito:

- Construir o LTS atíngivel de P.
- Calcular a relação $\simeq$
- Considerar cada classe de equivalência um estado: $S = \{[s]_{\simeq} \mid s \in Reach(P)\}$
- Adicionar transições entre duas classes sempre que uma transição exista entre os elementos da respectiva classe.
- Podemos apagar as transições que seriam adicionadas por $\Rightarrow$ e não estavam em $\longrightarrow$.
- Adicionar um lacete com τ se P tem uma τ-transição para a sua classe $[P]_{\simeq}$.

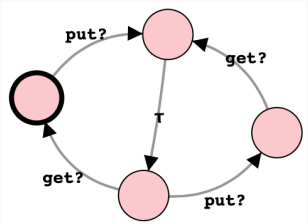
$$\begin{aligned} Buffer &:= put?.get?.Buffer \\ BufferL &:= put?.pass!.BufferL \\ BufferR &:= pass?.get?.BufferR \end{aligned}$$

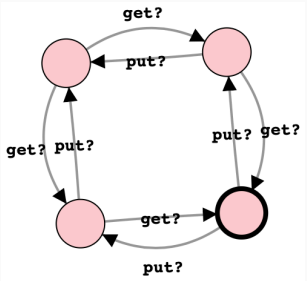
Será que $(BufferL|BufferR)\backslash\{pass!, pass?\} \sim (Buffer|Buffer)$?



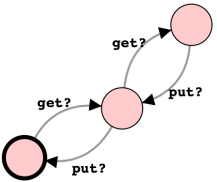


Será que $(BufferL|BufferR)\backslash\{pass!, pass?\} \sim (Buffer|Buffer)$?





Sim e o LTS mínimo equivalente é



Expressividade do CCS

Algoritmo de Peterson para a Exclusão Mútua

- P_1, P_2 processos
- variáveis partilhadas b_1, b_2 e k, sendo
- se $k = i$ então P_i pode entrar
- P_1 faz $k = 2$ (dá o privilégio a P_2)
- e simétricamente para P_2
- $b_i = \mathit{true}$ quando P_i espera
- $b_i = \mathit{false}$ quando P_i sai da zona crítica.

Processo P_i

while true do
  noncricital actions
  $b_i \leftarrow$ **true;**
  $k \leftarrow j$**;**
  while $b_j \wedge k = j$ **do**
    skip;
  critical actions
  $b_i \leftarrow$ **false**;

Algoritmo de Peterson em CCS

$$
\begin{aligned}
B_{1f} &:= b1rf!.B_{1f} + b1wf?.B_{1f} + b1wt?.B_{1t} \\
B_{1t} &:= b1rt!.B_{1t} + b1wf?.B_{1f} + b1wt?.B_{1t} \\
B_{2f} &:= b2rf!.B_{2f} + b2wf?.B_{2f} + b2wt?.B_{2t} \\
B_{2t} &:= b2rt!.B_{2t} + b2wf?.B_{2f} + b2wt?.B_{2t} \\
K_1 &:= kr1!.K_1 + kw1?.K_1 + kw2?.K_2 \\
K_2 &:= kr2!.K_2 + kw1?.K_1 + kw2?.K_2 \\
P_1 &:= b1wt!.kw2!.P_{11} \\
P_{11} &:= b2rf?.P_{12} + b2rt?.(kr2?.P_{11} + kr1?.P_{12}) \\
P_{12} &:= enter1.exit1.b1wf!.P_1 \\
P_2 &:= b2wt!.kw1!.P_{21} \\
P_{21} &:= b1rf?.P_{22} + b1rt?.(kr1?.P_{21} + kr2?.P_{22}) \\
P_{22} &:= enter2.exit2.b2wf!.P_2
\end{aligned}
$$

Se $k = 1$ no início

$$Peterson := (P_1|P_2|B_{1f}|B_{2f}|K_1)\backslash L$$

onde L são todas as ações excepto as de entrada e saída.

Ficheiro no Pseco.com:

https://pseuco.com/#/edit/remote/tkxz8fpu1t8ke56bj9uo

Exclusão mútua em CCS

Se se modelar a entrada e a saída da zona crítica fica

$$MutexSpec \quad := \quad enter1.exit1.MutexSpec + enter2.exit2.MutexSpec$$

Será que $MutexSpec \approx Peterson$?

Exclusão mútua em CCS

Se se modelar a entrada e a saída da zona crítica fica

$$MutexSpec \quad := \quad enter1.exit1.MutexSpec + enter2.exit2.MutexSpec$$

Será que $MutexSpec \approx Peterson$?

Não. Verifica que depois de

$$Peterson \stackrel{\tau}{\Rightarrow} (P_{12}|P_{21}|B_{1t}|B_{2t}|K_1)\backslash L$$

o estado atingido não tem transição por $enter_2$.

Exclusão mútua em CCS

Se se modelar a entrada e a saída da zona crítica fica

$$MutexSpec \quad := \quad enter1.exit1.MutexSpec + enter2.exit2.MutexSpec$$

Será que $MutexSpec \approx Peterson$?

Contudo pode-se provar que são equivalentes por traços fracos.

Exercício

Seja $P := acc?.del!.P$ que corresponde a um canal em que se envia e recebe mensagens. Seja a seguinte implementação

$$
\begin{array}{lcl}
S & := & acc?.send!.W \\
W & := & ack?.S + error?.send!.W \\
R & := & trans?.del!.ack!.R \\
M & := & send?.(\tau.E + trans!.M) \\
E & := & error!.M \\
I & := & (S|M|R)\backslash\{send,error,trans,ack\}
\end{array}
$$

1. *Desenha os diagramas dos processos $[\![P]\!]$ e $[\![I]\!]$.*
2. *Indica, justificando, se $I \sim P$*
3. *Indica, justificando, se $I \approx P$.*

$\diamond$

Exercício

Codifica em CCS o seguinte protocolo Sem para a exclusão mútua de dois processos P_1 e P_2 usando um semáforo y. O código genérico para um processo P_i, com $i = 1, 2$ é:

P_i: ***while*** **true do**
    while $y = 0$ **do**
        skip;
    $y \leftarrow 0$; **crit$_i$;** $y \leftarrow 1$;

1. *Define processos Y_1 e Y_0 que correspondem aos valores 1 e 0 da variável y (global). Considera ações de leitura e de escrita (as ações podem ser de entrada (?) ou de saída (!)).*
2. *Define os processos P_i para $i = 1, 2$, que apenas devem diferir no valor da ação critica: $crit_1$ e $crit_2$ respectivamente.*
3. *Define a execução paralela de P_1, P_2 e do processo que corresponde ao valor inicial 1 para y, e considerando observáveis apenas as acções $crit_1$ e $crit_2$ (todas as outras devem ser restritas).*
4. *Verifica se o protocolo resultante é fracamente bissimilar com MutexSpec.*