MODELY KOMUNIKUJÍCÍCH PROCESŮ

CSP, π – kalkul, OCCAM

CSP – Communicating Sequential Processes

- Communicating Sequential Processes (CSP) je formální jazyk pro modelování paralelních systémů komunikujících předáváním zpráv.
- C. A. R. HOARE 1978
- Výchozí pro další systémy, jako jsou napríkad π-kalkul nebo OCCAM

OCCAM

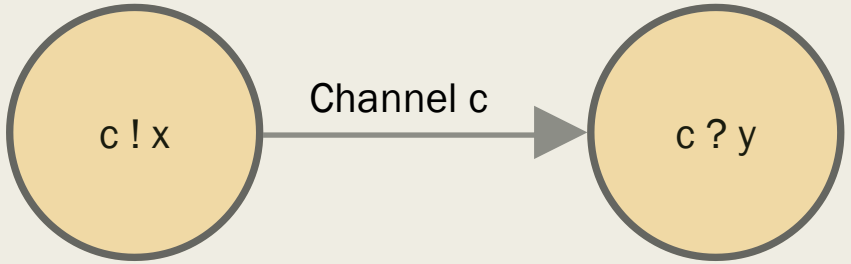
- Původně založen na Hoareho CSP, později inspirován a rozšířen o prvky z PI-kalkulu (OCCAM PI)
- Navržen tak, aby měl formální sémantikou vhodnou pro automatickou transformaci programu
- Transformace programů zapsaných v OCCAMu přímo na hardware (transputery, SW-HW codesign)
- Řada jazyků má k dispozici OCCAM – like rozšíření pro paralelní procesy, včetně JAVA

OCCAM, základní primitiva

- Occam primitivum je *proces* a je jednoho z následujících pěti druhů:

Přiřazení	`x := y + 2`
Vstup	`keyboard ? char`
Výstup	`screen ! char`
Skip	`SKIP – NOP, které se ukončí`
Stop	`STOP -- NOP, které se nikdy neukončí`

- Kanály poskytují možnost komunikace mezi procesy, nebufferovanou point-to-point synchronní komunikaci
- Kanály mají definovaný typ zpráv, které přenášejí



Sekvenční procesy

- SEQ executes sub-processes sequentially

```
SEQ
      keyboard ? char -- read char from keyboard
      screen ! char -- write char to screen
```

Můžeme replikovat sekvence

```
SEQ i = 0 FOR array.size
      stream ! data.array[i]
```

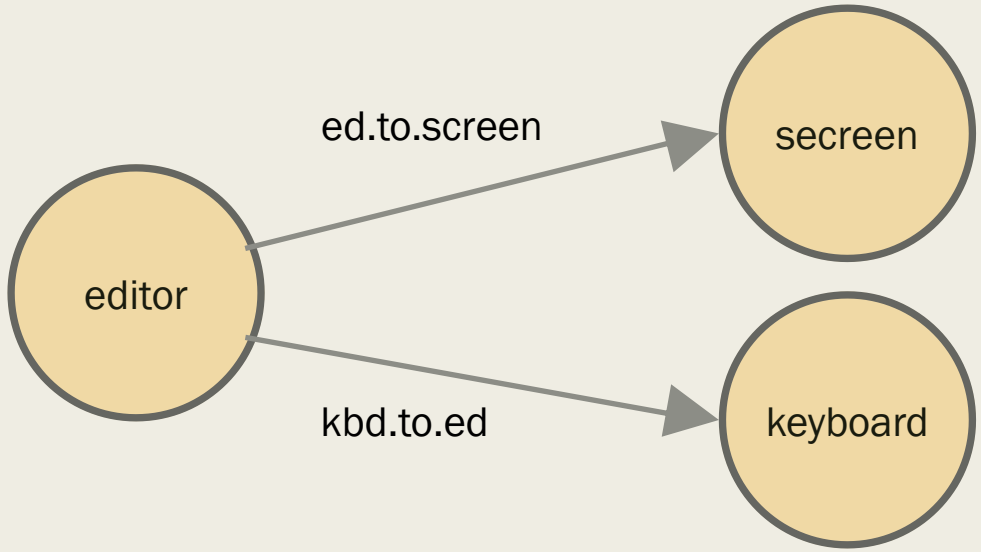
... je stejné jako

```
SEQ
      stream ! data.array[0]
      stream ! data.array[1]
      ...
```

Kompozice paralelních procesů

- PAR spustí vykonávání paralelních procesů

```
PAR
        keyboard(kbd.to.ed)
        editor(kbd.to.ed,ed.to.screen)
        screen(ed.to.screen)
```



PAR pro paralelní vykonávání procesů

- Příklad: Square pipe

```
WHILE next <> EOF
      SEQ
            x := next
            PAR
                  in ? next
                  out ! x * x
```

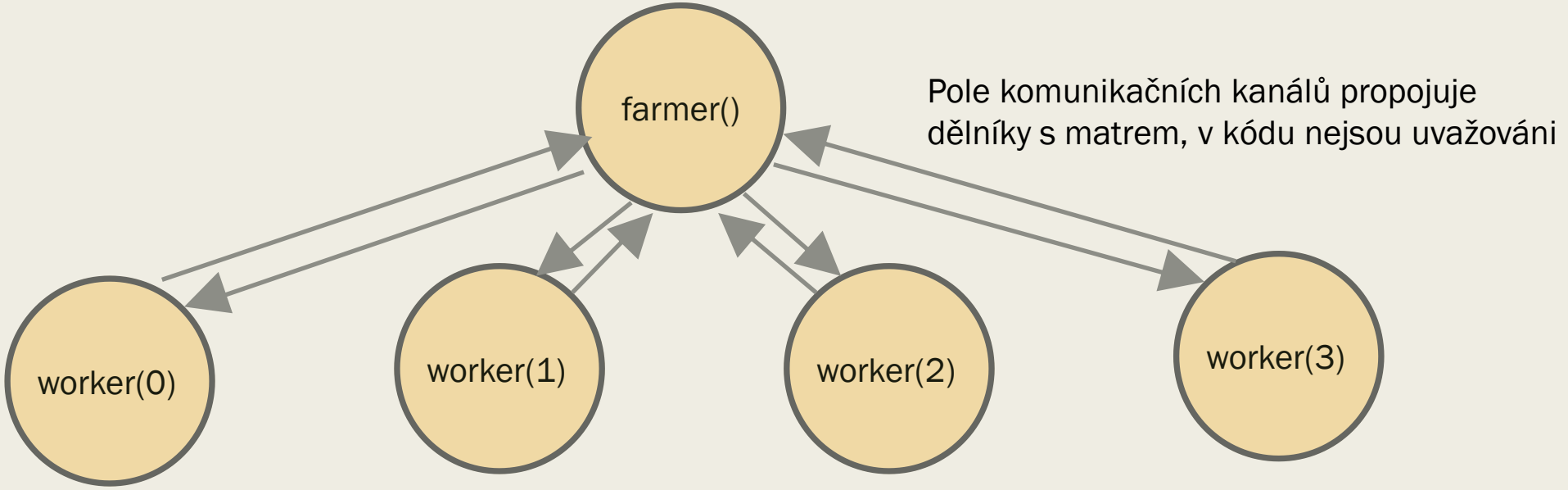
- Omezení přístupu paralelních procesů k datům
  - *Variables modified in one arm of PAR cannot be read or written in other parts of PAR, e.g., PAR -- this PAR is invalid*

```
SEQ
      mice := 42 -- assigns to mice
      c ! 42
      c ? mice -- assigns to mice
```

Replikovaný PAR

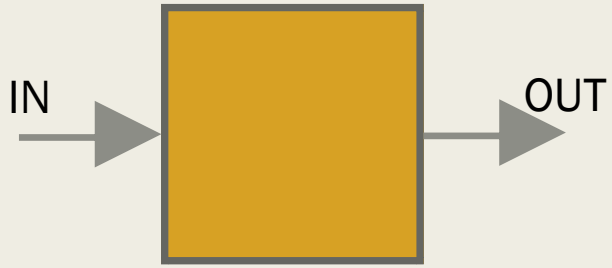
- Replikovaný PAR může být použit pro vytvoření pole paralelních procesů

```
PAR
        farmer()
        PAR i = 0 FOR 4 -- count must be constant
              worker(i)
```



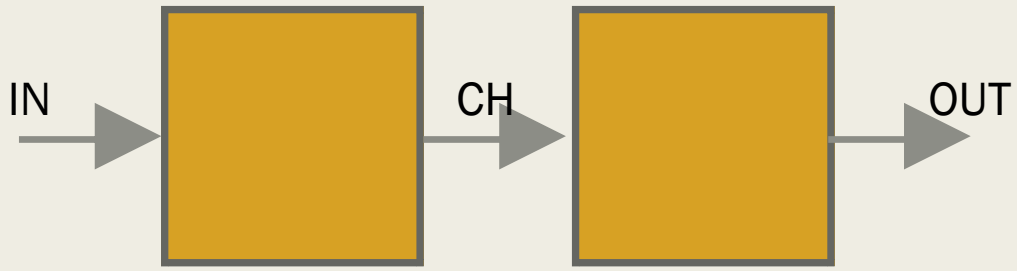
Pole komunikačních kanálů propojuje dělníky s matrem, v kódu nejsou uvažováni

Vyrovnávací paměť (Buffer)



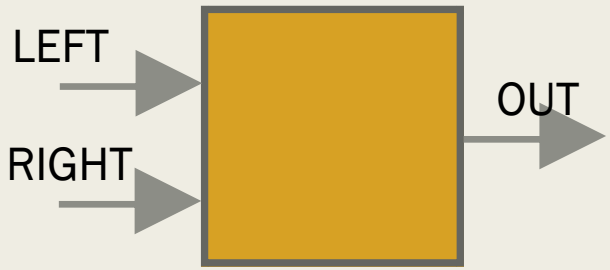

```
WHILE TRUE
    BYTE b:
    SEQ
        in ? b
        out ! b
```

Dvojitý Buffer




```
CHAN OF BYTE ch:
PAR
    WHILE TRUE
        BYTE b:
        SEQ
            in ? b
            ch ! b
    WHILE TRUE
        BYTE b:
        SEQ
            ch ? b
            out ! b
```

Dvojitý Buffer



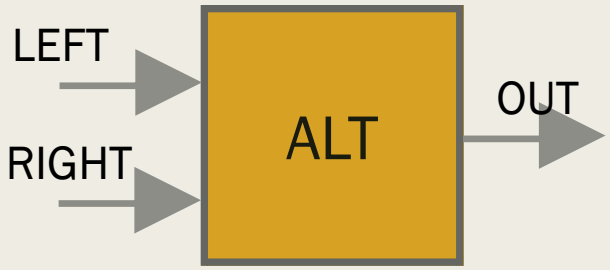

```
WHILE TRUE
    left ? packet
    out ! Packet
    right ? packet
    out ! packet
```

```
WHILE TRUE
    PAR
        left ? packet
            out ! Packet
        right ? packet
            out ! packet
```

Alternace

- ALT combines a number of processes only one of which is executed
- Každý proces má strážce:
  - *vstup input on channel*
  - *čeká na časovač*
  - *Čeká na splnění boolovské podmínky*

```
ALT
        left ? packet -- guard input statement
              out ! packet
        right ? packet -- guard input statement
              out ! packet
```



Alternace

```
ALT
        strazce1
                proces1
…
        strazcen
                proces
```

Strážci mohou čekat na

1, vstup

```
        in ? data
```

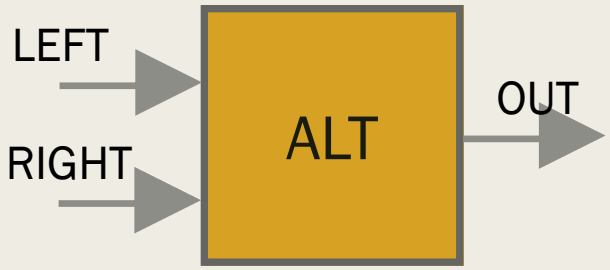
2, platnost boolovské podmínky & vstup

```
        not.empty  &  in ? data
```

2, platnost boolovské podmínky & SKIP

```
        not.empty  &  SKIP
```

```
WHILE TRUE
    ALT
        left ? packet
            out ! packet
        right ? packet
            out ! packet
```



OCCAM – deklarace procedur

```
-- other parts to program
        PROC buff(CHAN OF BYTE in, out)
                WHILE TRUE
                        BYTE x:
                        SEQ
                                in ? x
                                out ! x :
-- end of buff
        CHAN OF BYTE comms, buffer.in, buffer.out:
                PAR
                        buff(buffer.in, comms)
                        buff(comms, buffer.out)
```

Π-KALKUL

π - kalkul

- Formální nástroj pro modelování paralelních a **mobilních** procesů
- Jazyk + axiomy + redukční pravidla
- Syntax:
  - *Předpona π je v jednom z následujících tvarů* $\pi = \bar{x}y \,|x(z) \mid \tau \mid [x{=}y]$
  - *Proces je pak zapsán jako*

$$P ::= M \mid P|P' \mid \vartheta z P \mid !\,P$$

$$M ::= \mathbf{0} \mid \pi.P \mid M + M'$$

π – kalkul, procesy

$\mathbf{0}$	-	prázdný (ukončený) proces
$\pi.P$	–	předpona
$[x = y]P$	-	test
$P + P'$	-	nedeterministický výběr
$P \mid P'$	-	paralelní kompozice
$(\vartheta z)P$	-	omezení jména
$!\,P$	-	replikace

Strukturální kongruence

V relaci strukturální kongruence jsou procesy, které jsou stejné, pouze jinak zapsané.
V π–kalkulu je strukturální kongruence nejmenší kongruencí, která splňuje axiomy =>

$$M + \mathbf{0} \equiv M$$

$$\mathrm{M}_1 + \mathrm{M}_2 \equiv \mathrm{M}_2 + \mathrm{M}_1$$

$$\mathrm{M}_1 + (\mathrm{M}_2 + \mathrm{M}_3) \equiv (\mathrm{M}_1 + \mathrm{M}_2 + \mathrm{M}_3)$$

$$P|\mathbf{0} \equiv P$$

$$P_1|P_2 \equiv P_2|P_1$$

$$(P_1|P_2)|P_3 \equiv P_1|(P_2|P_3)$$

$$!\,P \equiv P|!\,P$$

$$(\vartheta \mathrm{a})\mathbf{0} \equiv \mathbf{0}$$

$$(\vartheta a)(\vartheta b)P \equiv (\vartheta b)(\vartheta a)P$$

$$[\mathrm{x} = \mathrm{x}]\,\pi.\,\mathrm{P} \equiv \pi.\,\mathrm{P}$$

$$\mathrm{P}_1|(\vartheta a)P_2 \equiv (\vartheta a)(\mathrm{P}_1|P_2)$$

$$pokud\ a \in fn(P_1)$$

π – kalkul, redukční pravidla

Je možné přejmenovávat volná jména v procesu, nebo

(COMM) $\overline{\bar{x}z.P|x(y).Q \rightarrow P|Q[y/z]}$

(aplikuje substituci na všechna volná jména v Q)

(R-PAR) $\frac{P \rightarrow Q}{P|R \rightarrow Q|R}$

(R-RES) $\frac{P \rightarrow Q}{(\vartheta x)P \rightarrow (\vartheta x)Q}$

(R-STRUCT) $\frac{P \equiv P' \rightarrow Q \equiv Q'}{P \rightarrow Q'}$

(RESTRUCT) $\overline{\tau.P + M \rightarrow P}$

(MATCH) $\frac{P \rightarrow Q}{[x=x]P \rightarrow Q}$

π – kalkul, komunikace

- Synchronní přes pojmenovaný komunikační kanál

$$x(y).P|\bar{x}a.P'$$

Tj. Procesy komunikují přes komunikační kanál kanál *x*

$$x(y).P|\bar{x}a.P' \rightarrow P[y/a] \mid P'$$

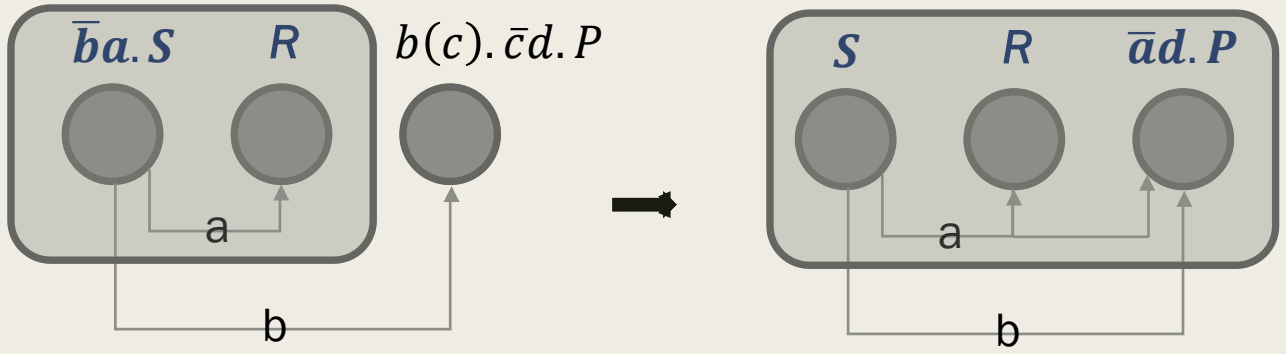
- Jiný příklad ukazuje mobilitu komunikujících kanálů*:*

$$\bar{x}b.\mathbf{0}|x(y).\bar{y}a.\mathbf{0}|b(w).P|c(z).P' \rightarrow$$
$$\mathbf{0}\big|\bar{b}a.\mathbf{0}\big|b(w).P|c(z).P' \equiv$$
$$\bar{b}a.\mathbf{0}|b(w).P|c(z).P' \rightarrow$$
$$\mathbf{0}|P[w/a]|c(z).P' \equiv P[w/a]|c(z).P'$$

π – kalkul, omezené komunikační kanály

- Omezení se používá pro soukromé – skryté kanály nějaké skupiny procesů
- Na rozdíl od CSP, kde omezení nemohlo být rozšířeno, v π – kalkulu mohou procesy v dosahu omezení poslat jméno skrytého kanálu vnějšímu procesu a rozšířit tak o něj toto omezení … (Scope Extrusion)

$$(\vartheta a)(\boldsymbol{\bar{b}a.S}|\boldsymbol{R})|b(c).\bar{c}d.P \rightarrow (\vartheta a)(\boldsymbol{S}|\boldsymbol{R}|\boldsymbol{\bar{a}d.P})$$



π – kalkul, sématika se štítkovanými přechody

- Provedení akce znázorňujeme štítkovanou přechodovou relací $\rightarrow^{\alpha}$ kde α je akce
- α může být:

xy	vstupní akce
$\bar{x}y$	otevřená výstupní akce
$\bar{x}(z)$	výstupní akce s omezeným jménem ‘z’
τ	tichá akce

Příklad:

$$\bar{s}a.\bar{s}b.\mathbf{0} \rightarrow^{\bar{s}a} \bar{s}b.\mathbf{0}$$

$$a(y).Q \rightarrow^{az} Q[y/z]$$

- Tiché akce τ nejsou pozorovatelné (například i komunikace v rámci procesu, nedeterministický výběr apod.)

Příklad:

$$\bar{s}a.\bar{s}b.\mathbf{0}|z(y).\mathbf{0}|s(v).s(u).\bar{v}u.\mathbf{0} \rightarrow^{\tau} \bar{s}b.0|z(y).\mathbf{0}|s(u).\bar{a}u.\mathbf{0}$$

Simulace

Relace simulace **S,** procesy **P** a **Q**

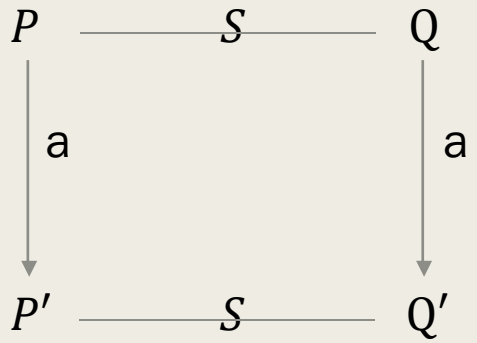
Silná simulace

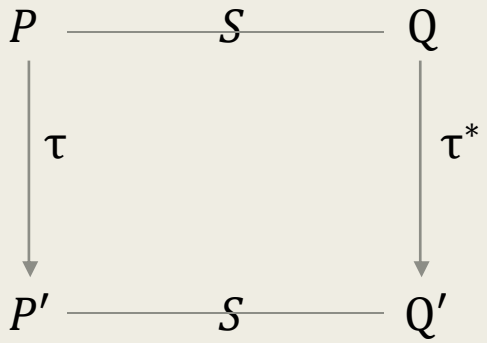
Pokud $P \; \boldsymbol{S} \; Q$ a $P \xrightarrow{a} P'$ *pak existuje* Q'
takové, že $Q \xrightarrow{a} Q'$ a $P' \boldsymbol{S} \; Q'$

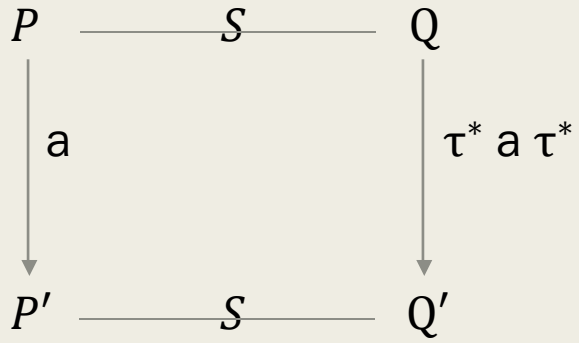
Slabá simulace

Pokud $P \; \boldsymbol{S} \; Q$ a $P \xrightarrow{\tau} P'$ *pak existuje* Q'
takové, že $Q \xrightarrow{\tau*} Q'$ a $P' \boldsymbol{S} \; Q'$

Pokud $P \; \boldsymbol{S} \; Q$ a $P \xrightarrow{a} P'$ *pak existuje* Q'
takové, že $Q \xrightarrow{\tau*a\tau*} Q'$ a $P' \boldsymbol{S} \; Q'$



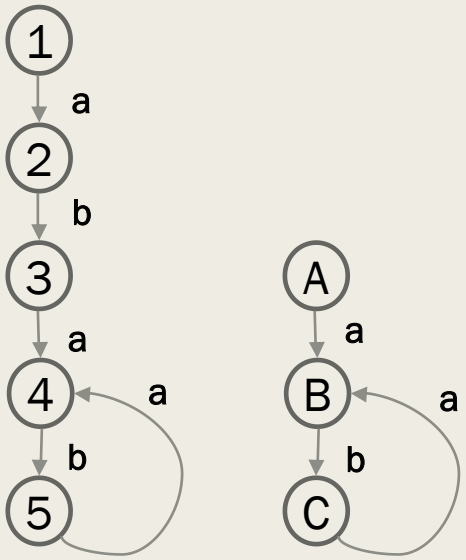




Bisimulace, bisimilarita

Bisimulace je relace taková, že procesy se mohou simulovat navzájem

Bisimilarita je relace mezi dvěma procesy, pro které existuje (alespoň jedna) relace simulace, je sjednocením všech bisimulací



Bisimulace

S1={(1,A),(2,B),(3,C),(4,B),(5,C),(A,1),(B,2),(C,3),(B,4),(C,5)}
S2={(1,C),(2,B),(3,C),(4,B),(5,C),(C,1),(B,2),(C,3),(B,4),(C,5)}
S3={(2,B),(3,C),(4,B),(5,C),(B,2),(C,3),(B,4),(C,5)}
S4={(3,A),(4,B),(5,C),(A,3),(B,4),(C,5)}
S5={(3,C),(4,B),(5,C),(3,C),(B,4),(C,5)}
S6={(4,B),(5,A),(5,C),(B,4),(A,5),(C,5)}

Bisimilarita

R={(1,A),(1,C),(2,B),(3,A),(3,C),(4,B),(5,A),(5,C),
(A,1),(C,1),(B,2),(A,3),(C,3),(B,4),(A,5),(C,5)}

π – kalkul, kontext, kongruence

- Kontext zapisujeme s uvedením místa [.] na místo **0** procesu, který není degenerující (první na některé pozici nedeterministického výběru)
- Proces *P*, který má být proveden v **kontextu** *C* pak *C*[*P*]
- Procesy jsou v relaci kongruence, pokud jsou v této relaci pro libovolný kontext

$$C_0 : (\vartheta z)([.] \mid !\, z(w).\bar{w}a.\mathbf{0})$$
$$C_1 : x(z).!\,(\vartheta w)(z(w).[.] + y(v).\mathbf{0})$$

- Aplikací procesu

$$C_0[!\,\bar{z}b.\mathbf{0}] = (\vartheta z)\,(!\,\bar{z}b.\mathbf{0} \mid !\, z(w).\bar{w}a.\mathbf{0}\,)$$
$$C_1[!\,\bar{z}b.\mathbf{0}] = x(z).!\,(\vartheta w)(z(w).!\,\bar{z}b.\mathbf{0} + y(v).\mathbf{0})$$

- **Kongruence** je relace S, kde $(\mathrm{P}, \mathrm{Q}) \in S \Rightarrow (C[P], C[Q]) \in S$ pro jakýkoliv kontext C

π – kalkul, kongruence

- ‚Nesmí být degenerující' – příklad

$$\mathrm{P}_1 = \tau.A.\mathbf{0}\ \mathrm{P}_2 = A.\mathbf{0}$$ (kongruence)

$$[.] + \mathrm{P}_3 : \tau.\mathrm{A} + \mathrm{P}_3 \text{ vs. } \mathrm{A} + \mathrm{P}_3$$

- Kongruence je ekvivalence, která platí pro aplikaci jakékoliv substituce

Například procesy

$$\mathrm{P}_1 = \mathrm{a(a)}|\bar{b}b \quad \text{a} \quad \mathrm{P}_2 = \mathrm{a(a)}.\bar{b}b + \bar{b}b.a(a)$$

nejsou kongruencí, protože pro $x(a)[.]|\bar{x}b$ a substituci $\sigma = \{[b/a]\}$

$$\mathrm{P}_1\sigma = b(b)|\bar{b}b \quad \text{a} \quad \mathrm{P}_2\sigma = b(b).\bar{b}b + \bar{b}b.b(b)$$

Kongruencí jsou procesy

$$\mathrm{P}_1 = \mathrm{a(a)}|\bar{b}b \quad \text{a} \quad \mathrm{P}_2 = \mathrm{a(a)}.\bar{b}b + \bar{b}b.a(a) + [a = b]\,\tau$$

π – kalkul, pozorování

- Pozorování se vztahují ke komunikačním kanálů, které nejsou soukromé pro nějakou skupinu procesů
- Zápis $\boldsymbol{P \downarrow x}$ – process P má jméno x jako vstup
- Zápis $\boldsymbol{P \downarrow \overline{x}}$ - process P má jméno x jako výstup
- Procesy jsou pozorovatelem neroznatelné, pokud mají stejné chování na výstupech
- Příklad:

$$\mathrm{P} = (\vartheta z)(\bar{s}a.\bar{t}b.\mathbf{0}|z(y).\mathbf{0}|q(v).t(u).\bar{v}u.\mathbf{0}|\tau.w(z).\mathbf{0})$$

$\mathrm{P} \downarrow a$??? – ne, je subjekt komunikace, ne komunikační kanál (vstup / výstup)

$\mathrm{P} \downarrow \bar{s}$??? – ano, agent může poslat jméno po kanálu s

$\mathrm{P} \downarrow \bar{t}$??? – ne, agent nyní nemůže posílat jméno po kanálu t

$\mathrm{P} \downarrow q$??? – ano, agent může přijmout jméno po kanálu q

$\mathrm{P} \downarrow t$??? – ne, agent nyní nemůže přijímat jméno po kanálu t

$\mathrm{P} \downarrow z$??? - ne, jméno z je privátní procesu a není pozorovatelné z vnějšku

$\mathrm{P} \downarrow w$??? – ne, agent nyní nemůže použít w, ale platí $\mathrm{P} \Downarrow w$, viz dále

π – kalkul, ‘**barbed**’ simulace, bisimulace

Centrem zájmu v oblasti procesních algeber je podobnost procesů

Jeden proces může simulovat jiný proces (relace **simulace**), založena na pozorováních -> **barbed** simulace

Silná ‘barbed’ bisimulace ->

Procesy *P* a *Q* jsou v relaci silné bisimulace $S(P, Q)$ pokud

$P \downarrow x$ implikuje Q $\downarrow x$

$P \rightarrow^{\tau} P'$ implikuje $Q \rightarrow^{\tau} Q'$pro nějaké Q' a $S(P', Q')$

$Q \downarrow x$ implikuje $P \downarrow x$

$Q \rightarrow^{\tau} Q'$implikuje $P \rightarrow^{\tau} P'$pro nějaké Q' a $S(P', Q')$

Slabá ‘barbed’ bisimulace -> tiché akce nejsou pozorovatelné, proto použijeme relaci pro pozorovatelné akce $\Rightarrow^{\alpha}$

Jednou nebo více tichými akcemi přejde proces do stavu dle relace.

$$\Rightarrow \stackrel{\text{def}}{=} (\rightarrow^{\tau})^{*}$$

Pak pro akci α

$$\Rightarrow^{\alpha} \stackrel{\text{def}}{=} \Rightarrow \rightarrow^{\alpha} \Rightarrow$$

Zápis $\boldsymbol{P \Downarrow x}$ značí, že proces P může tichými akcemi přejít ⇒ do stavu, kde $\boldsymbol{P \downarrow x}$

π – kalkul, silná simulace, příklad

Pozor! Mohou být procesy P a Q v relaci silné bisimulace?

$$P = \tau.(a(x)b(y) + a(x)c(z))$$
$$Q = \tau.a(x)b(y) + \tau.a(x)c(z)$$

Může Q simulovat krok?

$$P \rightarrow^{\tau} P'$$

$$\tau.(a(x)b(y) + c(x)c(z)) \rightarrow^{\tau} a(x)b(y) + a(x)c(z)$$

Nemůže, protože

$$Q \rightarrow^{\tau} Q'$$

buď 1, $\tau.a(x)b(y) + \tau.a(x)c(z) \rightarrow^{\tau} a(x)b(y)$

nebo 2, $\tau.a(x)b(y) + \tau.a(x)c(z) \rightarrow^{\tau} a(x)c(z)$

Pak v případě 1 $P' \rightarrow^{a(x)} P'' = c(z)$ nemůže $Q' \rightarrow^{a(x)} Q'' = b(y)$,

protože $\neg(P'' \downarrow c \rightarrow Q'' \downarrow c)$ a $\neg(Q'' \downarrow b \rightarrow P'' \downarrow b)$

V případě 2 $P' \rightarrow^{a(x)} P'' = b(y)$ nemůže $Q' \rightarrow^{a(x)} Q'' = c(z)$

protože $\neg(P'' \downarrow b \rightarrow Q'' \downarrow b)$ a $\neg(Q'' \downarrow c \rightarrow P'' \downarrow c)$

π – kalkul, ‘late’ / ’early’ simulace

- ‘pozdní’ late či ‘včasná’ early simulace, založená na štítkovaných přechodech
- Liší se v tom, jak jsou chápány vstupní akce a to se týká převážně procesů s nedeterministickým výběrem.
- Pro obě late/early simulace S:

  Pokud $P \xrightarrow{\alpha} P'$, akce není vstupní $\alpha \neq x(y)$ a vázané jméno v α není volné v P nebo Q $(bn(\alpha) \notin fn(P,Q))$, pak pro nějaké Q', $Q \xrightarrow{\alpha} Q'$ jsou tyto procesy v relaci late nebo early simulace $P' S\ Q'$

  Vstupní akce pro late simulaci

  Pokud $P \xrightarrow{x(y)} P'$, y $\notin fn(P,Q)$, pak pro nějaký proces Q' , $Q \xrightarrow{\alpha} Q'$ a pro jakékoliv jméno w jsou tyto procesy v relaci late simulace $P'[w/y]\ S\ Q'[w/y]$

  Vstupní akce pro early simulaci

  Pokud $P \xrightarrow{x(y)} P'$, y $\notin fn(P,Q)$, pak pro všecha jména w existuje nějaké Q' , $Q \xrightarrow{\alpha} Q'$ a tyto procesy jsou v relaci early simulace $P'[w/y]\ S\ Q'[w/y]$

π – kalkul, ‘late’/’early’ simulace, příklad

Uvažujme dva procesy

P= x(y).τ.**0** + x(y).**0**

Q= **P** + x(y).[x=y].τ.**0**

Jsou tyto procesy v relaci late bisimulace?

Nejsou, při late bisimulaci nedokáže proces P simulovat krok x(y) v případě, že proces Q vybere x(y).[x=y].τ.**0** podproces. Příjem jména ***x*** za vstupní ***y*** způsobí jiné chování (provedení τ) než příjem jiného jména za ***y*** (způsobí zablokování procesu, tedy **0**). Tedy P nedokáže jedním z možných podprocesů x(y).τ.**0** či x(y).**0** nedokáže simulovat všechna možná přijímaná jména.

Jsou tyto procey v relace early bisimulace?

Jsou, v tomto případě je potřeba, aby pro každé jméno přijímané přes x(y) existoval podproces, který ‘odpovídá’ podprocesu druhého procesu. Tedy pokud by pro přijímané jméno proces Q (nedeterministicky) zvolil podproces s testem, tak P může reagovat na základě jména, jestli odpovídající jeho podproces je ***τ.0*** (v případě, že přijímané jméno je x a test tedy projde), nebo ***0***

π – kalkul, příklady

- **Forwarder** $\mathrm{FW}(\mathrm{a}, b) = a(z).\bar{b}z$

$$\bar{a}x \mid \mathrm{FW}(\mathrm{a}, b) = \bar{a}x \mid a(z)\,.\bar{b}z \rightarrow \bar{b}x$$

$$\bar{a}x|(\vartheta b)(\mathrm{FW}(\mathrm{a}, b)|\mathrm{FW}(b, c)) = \bar{a}x|(\vartheta b)\left(a(z).\bar{b}z\middle|b(z).\bar{c}z\right) \rightarrow$$
$$(\vartheta b)\left(\bar{b}x\middle|b(z).\bar{c}z\right) \rightarrow \bar{c}x$$

- **Duplikátor** $\mathrm{D}(a, b, c) = a(d).(\bar{b}d|\bar{c}d)$

$$\bar{a}x|\mathrm{D}(a, b, c) = \bar{a}x|a(d).(\bar{b}d|\bar{c}d) \rightarrow \bar{b}x|\bar{c}x$$

- **Generátor nového jména** $NN(a) = a(x).(\vartheta b)(\bar{x}b)$

$$\bar{\imath}\mathrm{o}|NN(i) = \bar{\imath}o|i(o).(\vartheta b)(\bar{o}b) \rightarrow (\vartheta b)(\bar{o}b)$$

π – kalkul, přepínání stanic

$Car(talk) = \overline{talk} < data >.Car(talk)$
$StationA(talk, sw) = talk(x).Process(x).StationA(talk, sw) + \overline{sw} < talk >.\mathbf{0}$
$StationB(sw) = sw(t).BaseStation(t)$
$BaseStation(t) = t(x).Process(x).BaseStation(t)$

$\boldsymbol{Car(talk)|StationA(talk, sw)|StationB(sw)}$

$\overline{talk} < ahoj >.Car(talk) \mid talk(x).Process(x).StationA(talk, sw) + \overline{sw} < talk >.\mathbf{0}|StationB(sw)$

$Car(talk) \mid Process(ahoj).StationA(talk, sw)|StationB(sw)$

$\overline{talk} < prl >.Car(talk)| StationA(talk, sw)|StationB(sw)$

$\overline{talk} < prl >.Car(talk) \mid talk(x).Process(x).StationA(talk, sw) + \overline{sw} < talk >.\mathbf{0} |StationB(sw)$

$\overline{talk} < prl >.Car(talk) \mid \overline{sw} < talk >.\mathbf{0} \mid sw(t).BaseStation(t)$

$\overline{talk} < prl >.Car(talk) |BaseStation(talk)$

$\overline{talk} < prl >.Car(talk) |talk(x).Process(x).BaseStation(talk)$

π – kalkul, příklad, neomezený buffer

Definujeme procesy modelující neomezený buffer

$\boldsymbol{B}(\mathrm{i}, o) = i(x).\boldsymbol{C}(x, i, o)$

$\boldsymbol{C}(x, i, o) = \bar{o}x.\boldsymbol{B}(i, o) + i(y).(\boldsymbol{C}(y, i, o) \sim \boldsymbol{C}(x, i, o))$

$\boldsymbol{C}(\mathrm{y}, \mathrm{i}, \mathrm{o}) \sim \boldsymbol{C}(\mathrm{x}, \mathrm{i}, \mathrm{o}) = (\vartheta m)(\boldsymbol{C}(y, i, m) | \boldsymbol{C}(x, m, o))$

Prozkoumáme chování procesu

$\bar{\imath}a.\bar{\imath}b.o(x).o(y).\bar{\imath}c.o(z) \mid \boldsymbol{B}(\mathrm{i}, o)$

π – kalkul, příklad, neomezený buffer

$$\boldsymbol{B}(\mathrm{i}, o) = i(x).\boldsymbol{C}(x, i, o)$$
$$\boldsymbol{C}(x, i, o) = \bar{o}x.B(i, o) + i(y).(\boldsymbol{C}(y, i, o) \sim \boldsymbol{C}(x, i, o))$$
$$\boldsymbol{C}(\mathrm{y}, \mathrm{i}, \mathrm{o}) \sim \boldsymbol{C}(\mathrm{x}, \mathrm{i}, \mathrm{o}) = (\vartheta m)(\boldsymbol{C}(y, i, m) | \boldsymbol{C}(x, m, o))$$

$$\bar{\imath}a.\bar{\imath}b.o(x).o(y).\bar{\imath}c.o(z) \mid \boldsymbol{B}(\mathrm{i}, o) =$$

$$\bar{\imath}a.\bar{\imath}b.o(x).o(y).\bar{\imath}c.o(z) \mid i(x).\boldsymbol{C}(x, i, o) \rightarrow$$
$$\bar{\imath}b.o(x).o(y).\bar{\imath}c.o(z) \mid \boldsymbol{C}(a, i, o) =$$

$$\bar{\imath}b.o(x).o(y).\bar{\imath}c.o(z) \mid \bar{o}a.\boldsymbol{B}(i, o) + i(y).(\boldsymbol{C}(y, i, o) \sim \boldsymbol{C}(a, i, o)) \rightarrow$$

$$o(x).o(y).\bar{\imath}c.o(z) \mid (\boldsymbol{C}(b, i, o) \sim \boldsymbol{C}(a, i, o)) =$$

$$o(x).o(y).\bar{\imath}c.o(z) \mid (\vartheta m)(\boldsymbol{C}(b, i, m) | \boldsymbol{C}(a, m, o)) =$$
$$o(x).o(y).\bar{\imath}c.o(z) \mid$$

$$(\vartheta m)\left(\bar{m}b.\boldsymbol{B}(i, m) + i(y).\left(\boldsymbol{C}(y, i, m) \sim \boldsymbol{C}(b, i, m)\right) \middle| \bar{o}a.\boldsymbol{B}(m, o) + m(y).\left(\boldsymbol{C}(y, m, o) \sim \boldsymbol{C}(a, m, o)\right)\right.$$

π – kalkul, příklad, neomezený buffer

$$\boldsymbol{B}(\mathrm{i},o) = i(x).\boldsymbol{C}(x,i,o)$$
$$\boldsymbol{C}(x,i,o) = \bar{o}x.B(i,o) + i(y).(\boldsymbol{C}(y,i,o) \sim \boldsymbol{C}(x,i,o))$$
$$\boldsymbol{C}(\mathrm{y},\mathrm{i},\mathrm{o}) \sim \boldsymbol{C}(\mathrm{x},\mathrm{i},\mathrm{o}) = (\vartheta m)(\boldsymbol{C}(y,i,m) | \boldsymbol{C}(x,m,o))$$

$$o(x).o(y).\bar{\imath}c.o(z) \mid$$
$$(\vartheta m)\left(\bar{m}b.\boldsymbol{B}(i,m) + i(y).\left(\boldsymbol{C}(y,i,m) \sim \boldsymbol{C}(b,i,m)\right)\middle| \bar{o}a.\boldsymbol{B}(m,o) + m(y).\left(\boldsymbol{C}(y,m,o) \sim \boldsymbol{C}(a,m,o)\right) \rightarrow\right.$$
$$o(y).\bar{\imath}c.o(z) \mid (\vartheta m)\left(\bar{m}b.\boldsymbol{B}(i,m) + i(y).\left(\boldsymbol{C}(y,i,m) \sim \boldsymbol{C}(b,i,m)\right)\middle| \boldsymbol{B}(m,o) =\right.$$
$$o(y).\bar{\imath}c.o(z) \mid (\vartheta m)\left(\bar{m}b.\boldsymbol{B}(i,m) + i(y).\left(\boldsymbol{C}(y,i,m) \sim \boldsymbol{C}(b,i,m)\right)\middle| m(x).\boldsymbol{C}(x,m,o) \rightarrow\right.$$
$$o(y).\bar{\imath}c.o(z) \mid (\vartheta m)(\boldsymbol{B}(i,m) | \boldsymbol{C}(b,m,o) =$$
$$o(y).\bar{\imath}c.o(z) \mid (\vartheta m)(i(x).\boldsymbol{C}(x,i,m) \mid \bar{o}b.\boldsymbol{B}(m,o) + i(y).(\boldsymbol{C}(y,m,o) \sim \boldsymbol{C}(b,m,o)) \rightarrow$$
$$\bar{\imath}c.o(z) \mid (\vartheta m)(i(x).\boldsymbol{C}(x,i,o) \mid \boldsymbol{B}(m,o))$$

π – kalkul, příklad, neomezený buffer

$$\boldsymbol{B}(\mathrm{i}, o) = i(x).\boldsymbol{C}(x, i, o)$$
$$\boldsymbol{C}(x, i, o) = \bar{o}x.B(i, o) + i(y).(\boldsymbol{C}(y, i, o) \sim \boldsymbol{C}(x, i, o))$$
$$\boldsymbol{C}(\mathrm{y}, \mathrm{i}, \mathrm{o}) \sim \boldsymbol{C}(\mathrm{x}, \mathrm{i}, \mathrm{o}) = (\vartheta m)(\boldsymbol{C}(y, i, m) | \boldsymbol{C}(x, m, o))$$

$$\bar{\imath}c.o(z) | (\vartheta m)(i(x).\boldsymbol{C}(x, i, o) \mid \boldsymbol{B}(m, o)) \rightarrow$$
$$o(z) \left| (\vartheta m)(\boldsymbol{C}(c, i, o) | \boldsymbol{B}(m, o)) \right. =$$

$$o(z) \left| (\vartheta m)(\bar{o}c.\boldsymbol{B}(i, o) + i(y).(\boldsymbol{C}(y, i, o) \sim \boldsymbol{C}(c, i, o)) \mid \boldsymbol{B}(m, o)) \right. \rightarrow \boldsymbol{O} | (\vartheta m)(\boldsymbol{B}(i, o) | \boldsymbol{B}(m, o))$$