1 PROSESSIALGEBRA

- Seuraavassa on tarkoitus kuvata erästä Petri-verkoille vaihtoehtoista rinnakkaisten ja hajautettujen digitaalisten järjestelmien määrittelymenetelmää.
- Prosessialgebra on yleisnimitys joukolle rinnakkaisten ja hajautettujen järjestelmien spesifiontimenetelmille. Tällä kurssilla prosessialgebrana tulee olemaan CCS (Calculus of Communicating Systems). Muita prosessialgebroja ovat CSP, ACP, ESTELLE, LOTOS.
- Algebra on matemaatinen rakenne joka koostuu alkioista ja operaatioista, sekä alkioiden ja operaatioden välisistä yhtälöistä.
- Prosessi on tapahtumien joukko.
- Prosessialgebra koostuu siis tapahtumista ja operaatioista joilla tapahtumia yhdistellään.

©Johan Lilius 1997, Keijo Heljanko 1998

©Johan Lilius 1997, Keijo Heljanko 1998

Johdanto

- Petri-verkko kuvaa järjestelmän kausaalisia riippuvuuksia.
- Prosessialgebrassa lähtökohta on erilainen:
  - Sekventiaalinen ohjelma voidaan mieltää matemaattiseksi funktioksi joka kuvaa jonkin muistin tilan jollekin toiselle. Oletuksena tällä ns. funktionaaliselle lähestymistavalla on, että jokainen ohjelma hallitsee itse omaa muistiavaruutta.
  - Jos sallitaan rinnakkaisia ohjelmia, jotka jakavat muistin, tilanne muttuu oleellisesti:

    (1) `X:=1` (2) `X:=0;X:=X+1`
  - Jos ohjelmat toimivat täysin riippumattomasti on niiden vaikutus sama.
  - Jos taas on olemassa toinen ohjelma joka minä hetkenä hyvänsä voi kirjoittaa `X:=1` ohjelmilla (1) ja (2) ei enää ole samaa vaikutusta.

JOHDANTO

- Prosessialgebran keskeinen primitiivi on

  KOMMUNIKAATIO.

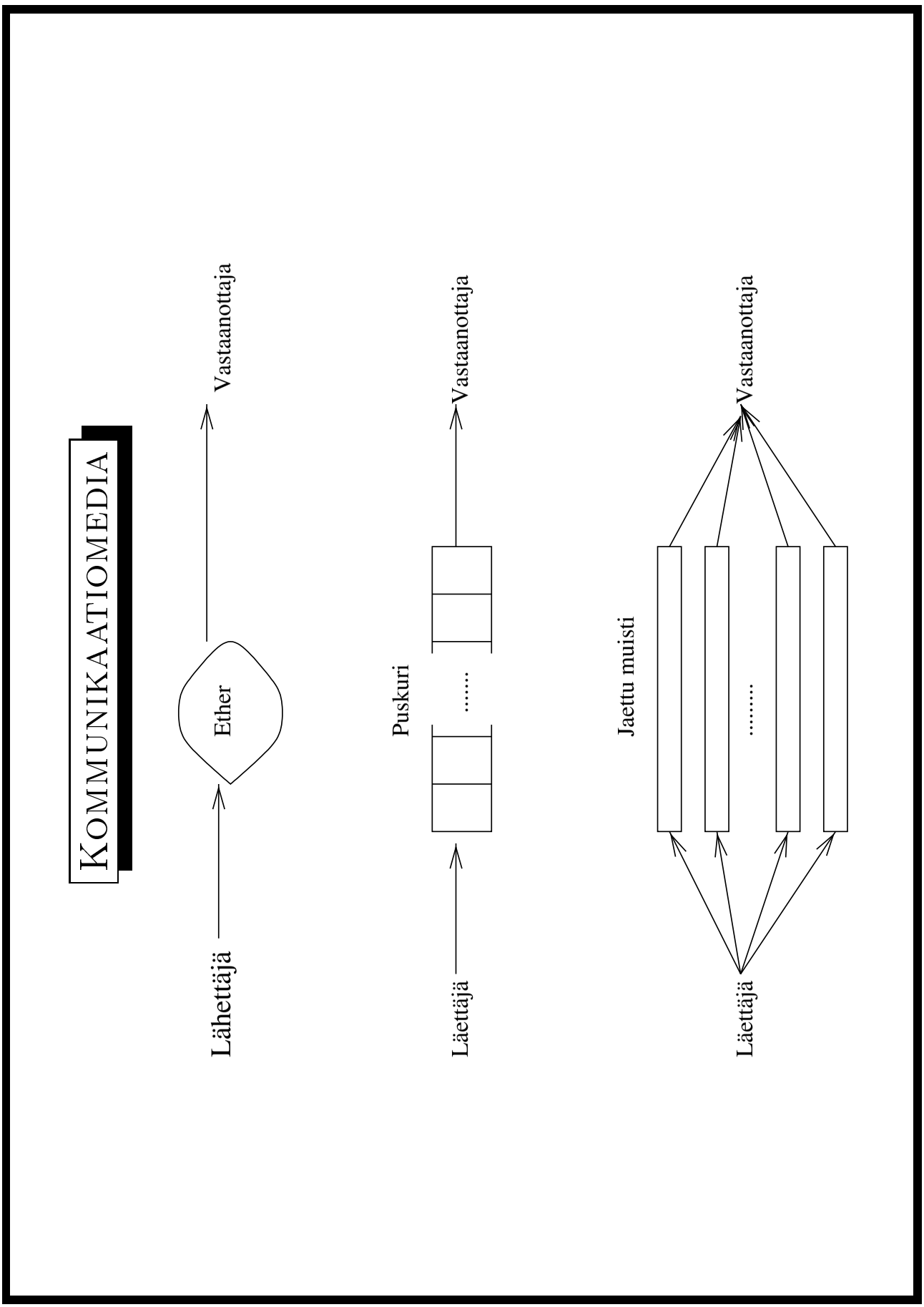
  Tutkitaan siis järjestelmien tai järjestelmän osien välistä kommunikaatiota tai interaktioita.
- Prosessialgebrassa on operaatiot kommunikaatiotapahtumien yhdistämiselle.
- Eräs Petri-verkkojen puute on, että ei ole olemassa operaatioita joiden avulla verkkoja voisi yhdistellä

©Johan Lilius 1997, Keijo Heljanko 1998

KOMMUNIKAATIOTAPAHTUMA

- Kommunikaatio ja rinnakkaisuus täydentävävät toisiaan käsitteinä:
  - Järjestelmässä on *moninaisuutta*: se koostuu eri osista jotka toimivat rinnakkaisesti.
  - Järjestelmässä on *ykseyttä*: sen eri osat kommunikoivat keskenään.
- Järjestelmän jokaisella osalla on oma identiteetti, se on *AGENTTI*.
- Järjestelmän käyttäymistä tutkitaan havaitsemalla sen tapahtumia.
- Tapahtumat havaitaan identifioimalla ne agentit jotka osallistuvat siihen.
- Agentti voi siis osallistua muiden agenttien kanssa tapahtumaan jolloin se on *KOMMUNIKAATIOTAPAHTUMA*,
- tai, agenttin toiminto voi olla riippumaton muiden agenttien toiminnoista jolloin se on *RINNAKKAINENTAPAHTUMA*.
- Itse asiassa voidaan kuitenkin olettaa, että tällainen riippumaton toiminto koostuu agentin sisäisitä kommunikaatiotapahtumista.
- Oleellista on siis tehdä ero näkyvien eli *ULKOISTEN* kommunikaatiotapahtumien ja näkymättömien eli *SISÄISTEN* kommunikaatiotapahtumien välillä.

©Johan Lilius 1997, Keijo Heljanko 1998



©Johan Lilius 1997, Keijo Heljanko 1998

KOMMUNIKAATIOMEDIA

- Mitä yhteistä on erillaisilla kommunikaatiotapahtumilla.
  - Lähettäjä
  - Media
  - Vastaanottaja
  - Kommunikaatio tapahtuu viestien kautta.
- Viestin lähettäminen, sen kulku median kautta ja sen vastaanotto voidaan mieltää yhdeksi ajallisesti jakamattomaksi tapahtumaksi.
- Media voidaan siis mieltää agentin tapaiseksi osallistujaksi.
- Prosessialgebrassa ei erikseen käsitellä mediaa. Jos mallintaja haluaa erikseen käsitellä mediaa, voi hän mallintaa sen yhtenä agenttina.
- Kommunikaatio on siis kahden agentin samanaikaisesti kokema ajallisesti jakamaton tapahtuma. Tällaista tapahtumaa kutsutaan myös kättelyksi.

$$S \longrightarrow R$$

©Johan Lilius 1997, Keijo Heljanko 1998

Järjestelmän oikeellisuus

- Miten varmistua järjestelmän toimivuudesta?
- Järjestelmän käyttäytyminen on se mitä siitä voidaan havaita.
- Järjestelmän havaitseminen edellyttää sen kanssa kommunikointia.
- Järjestelmä toimii siis oikein jos sen ulkoiset kommunikaatiotapahtumat vastavat spesifikaation kommunikaatiotapahtumia.
- Tällaista oikeellisuuden käsitettä kutsutaan havaintoekvivalenssiksi.

©Johan Lilius 1997, Keijo Heljanko 1998

CCS: Syntaksi

- Määritellään syntaksi jolla voidaan kuvata agentteja.
- Agentit mallintavat prosesseja.
- Agentit kommunikoivat porttien kautta.
- Portit ovat joko sisäänmeno- tai ulostulo-portteja.
- Jokaisella portilla on tunnus (label).
- Ulostulo-portin tunnusta merkitään yläviivalla (esim. $\overline{in}$).
- Oletetaan siis annetuksi joukko $\mathcal{L}$ tunnuksia joka koostuu joukosta sisäänmeno tunnuksia $\mathcal{I}$ ja ulostulo tunnuksia $\mathcal{O}$. l, k ovat mielivaltaisia tunnuksia. Jos l on tunnus on sen komplementti $\bar{l}$.

©Johan Lilius 1997, Keijo Heljanko 1998

Syntaksi

Nil

- Agentti *Nil* ei tee mitään.
- Agenttia *Nil* käytetään prosessin päättymisen kuvaamiseen.

Prefixing

- Jos l on tunnus ja P on agentti on $l.P$ se agentti jota vastaava prosessi voi ensin suoritta toiminnon l ja sen jälkeen P:n toiminnot.
- Esim. tulitikku voidaan sytyttää ja sen jälkeen se ei voi enää tehdää mitään: $Syty.Nil$.

©Johan Lilius 1997, Keijo Heljanko 1998

Syntaksi

Määrittely

- Agentti voidaan määritellä määrittelylausekkeen avulla:
  $Tulitikku \stackrel{\text{def}}{=} Syty.Nil$
- Määrittely voi myös olla rekursiivinen:
  $Clock \stackrel{\text{def}}{=} tick.Clock$
- Yksittäinen tapahtuma voi välittää parametrin:
  $Addone \stackrel{\text{def}}{=} in(x).\overline{out}(x+1).Addone$
- Agentti voidaan määritellä useamman lausekken avulla:
  $Addone \stackrel{\text{def}}{=} in(x).Holding(x)$
  $Holding(y) \stackrel{\text{def}}{=} \overline{out}(y+1).Addone$

©Johan Lilius 1997, Keijo Heljanko 1998

©Johan Lilius 1997, Keijo Heljanko 1998

Syntaksi

Summa

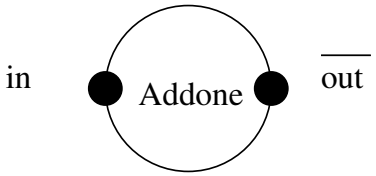
- Seuraava lauseke esittää konetta joka vaihtaa 50 penniä kahteen 20 pennin ja yhteen 10 pennin kolikkoon:
  $Change \stackrel{\text{def}}{=} fifty.\overline{out}(20).\overline{out}(20).\overline{out}(10).Change$
- Lajennetaan laitetta niin että se vaihtaa markan kolikon kahteen 50 pennin kolikkon:
  $Change1 \stackrel{\text{def}}{=} hundred.\overline{out}(50).\overline{out}(50).Change1$
- Uuden laitteen toimintaa voidaan kuvata summa-operaation avulla:

$$\begin{aligned} NewChange \quad &\stackrel{\text{def}}{=} \quad fifty.\overline{out}(20).\overline{out}(20).\overline{out}(10).NewChange \\ &+ \quad hundred.\overline{out}(50).\overline{out}(50).NewChange \end{aligned}$$

SYNTAKSI

Kommunikaatio ja Rinnakkainen kompositio

- Prosessien välisen kommunikaation kuvaaminen on helpompaa jos käytetään kuvaa apuna:

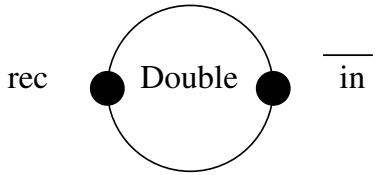


esittää agenttia $Addone$.

©Johan Lilius 1997, Keijo Heljanko 1998

SYNTAKSI

- Agenttia
  $Double \stackrel{\text{def}}{=} rec(x).\overline{in}(2x).Double$
  esittää kuva



©Johan Lilius 1997, Keijo Heljanko 1998

SYNTAKSI

- Kahden agentin rinnakkainen kompositio kuvataan operaattorilla $|$. $P|Q$ on siis agentti jossa P ja Q voivat toimia itsenäisesti tai halutessaan kommunikoida.
- Agenttin $DA \stackrel{\text{def}}{=} Double|Addone$ kuva on :

©Johan Lilius 1997, Keijo Heljanko 1998

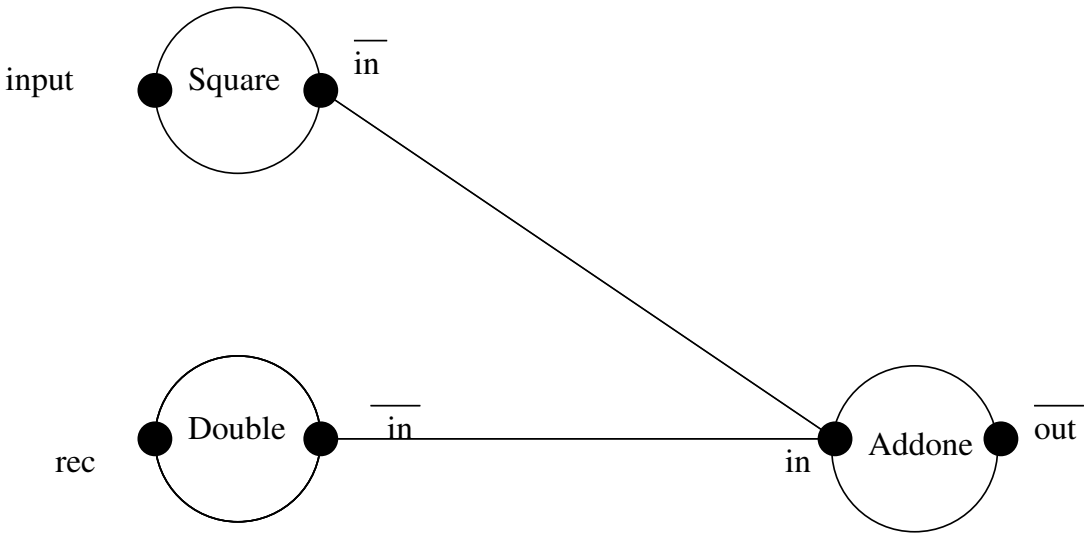
Syntaksi

- DA voidaan yhdistää agenttiin

$$Square \stackrel{\text{def}}{=} input(y).\overline{in}(y^2).Square$$

jolloin saadaan

$$SDA \stackrel{\text{def}}{=} Square|DA$$



- Ongelma: Kaikki tunnukset toistaiseksi globaaleja.

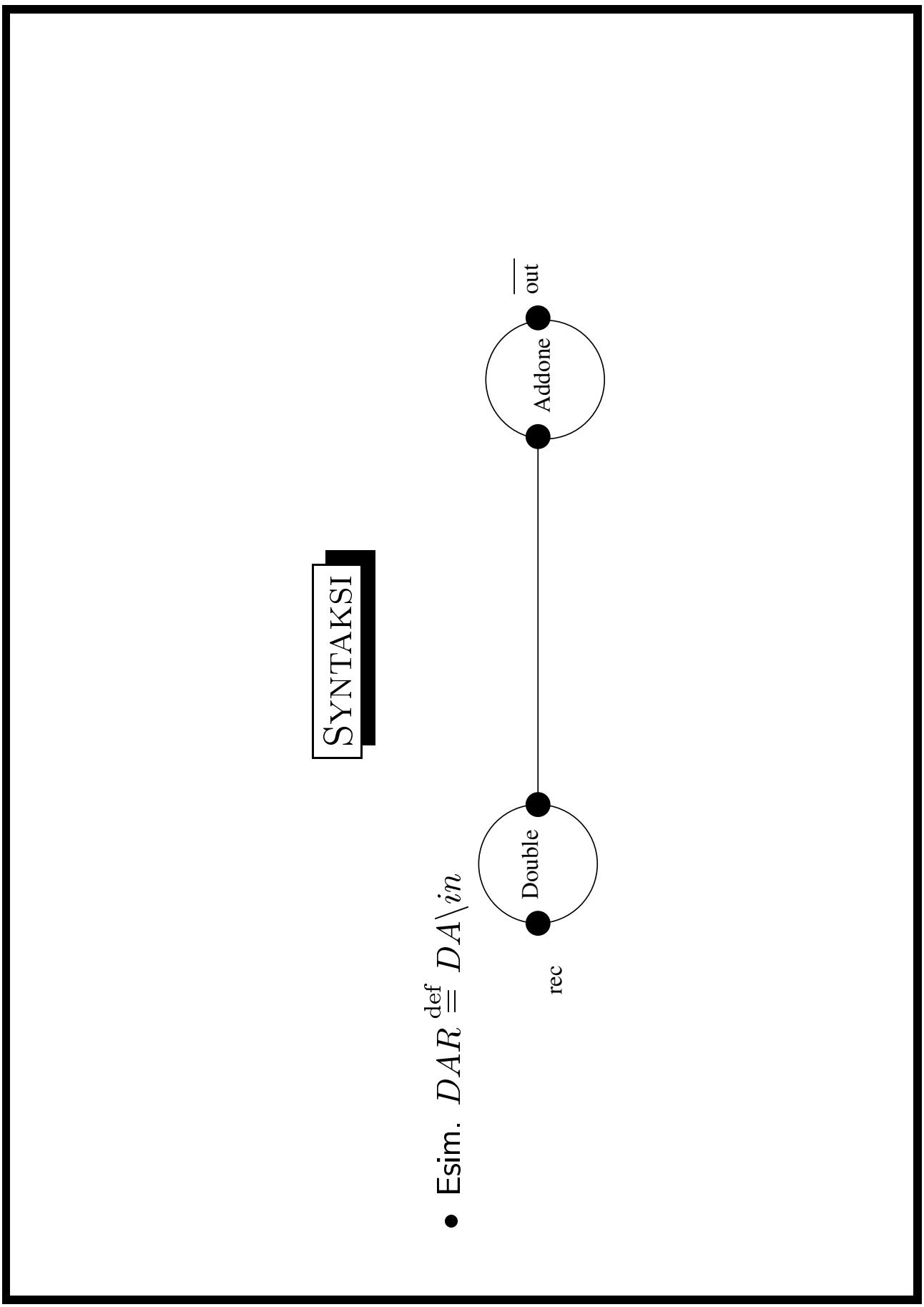
©Johan Lilius 1997, Keijo Heljanko 1998

SYNTAKSI

Rajoitus

- Usein on tarvetta rajoittaa tunnuksen näkyvyttä.
- Tätä varten on olemassa rajoitus operaattori $\backslash$.
- $P\backslash l$ on agentti joka muuten on sama kuin P, mutta ei voi kommunikoida porttien $l, \bar{l}$ kautta.

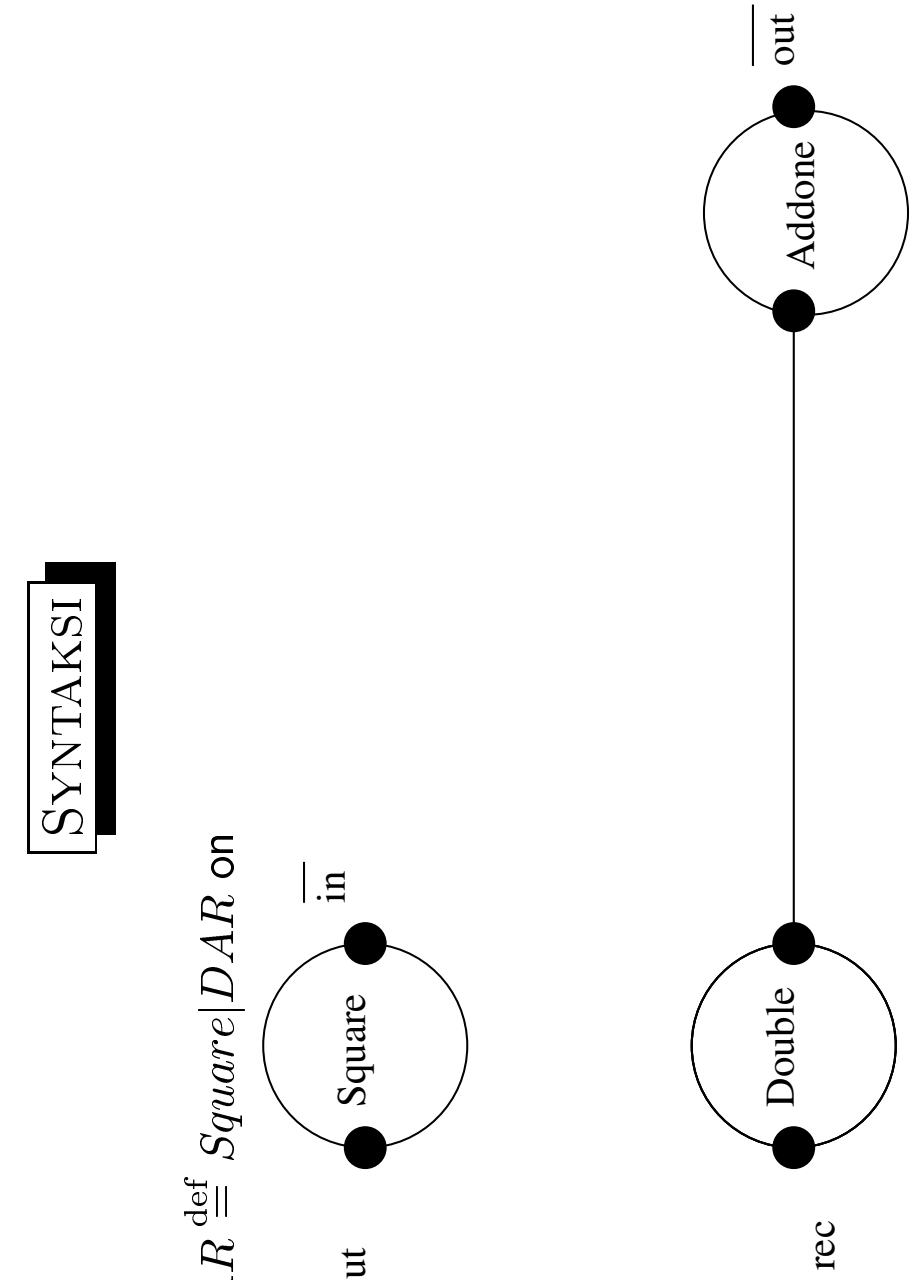
©Johan Lilius 1997, Keijo Heljanko 1998



©Johan Lilius 1997, Keijo Heljanko 1998

SYNTAKSI

- Nyt $SDAR \stackrel{\text{def}}{=} Square|DAR$ on



©Johan Lilius 1997, Keijo Heljanko 1998

©Johan Lilius 1997, Keijo Heljanko 1998

Syntaksi

Valinta

- Olkoon annettu seuraavat agentit:

$$
\begin{aligned}
User1 &\stackrel{\text{def}}{=} in1(x).(\overline{send1}(x).rec1(y).\overline{out1}(y).User.1 \\
&\quad + \overline{send2(x)}.rec2(y).\overline{out1}(y).User1) \\
User2 &\stackrel{\text{def}}{=} in2(x).\overline{send2}(x).rec2(y).\overline{out2}(y).User2 \\
Test1 &\stackrel{\text{def}}{=} send1(z).if\ \ z = 0\ \ then\ \ \overline{rec1}(\mathbf{T}).Test1\ \ else\ \ \overline{rec1}(\mathbf{F}).Test1 \\
Test2 &\stackrel{\text{def}}{=} send2(z).if\ \ z = 0\ \ then\ \ \overline{rec2}(\mathbf{T}).Test2\ \ else\ \ \overline{rec2}(\mathbf{F}).Test2
\end{aligned}
$$

- Agentti $\text{if}\, e\, \text{then}\, P\, \text{else}\, Q$ on agentti joka toimii kuten P jos e on tosi ja kuten Q jos e on epätosi.
- Näistä agenteista voidaan konstruoida järjestelmät:

$$
\begin{aligned}
SystemA &\stackrel{\text{def}}{=} ((User2 \mid Test2) \mid (User1 \mid Test1))\backslash\{send1, send2, rec1, rec2\} \\
SystemB &\stackrel{\text{def}}{=} ((User2 \mid Test2)\backslash\{send2, rec2\}) \mid ((User1 \mid Test1)\backslash\{send1, rec1\})
\end{aligned}
$$

©Johan Lilius 1997, Keijo Heljanko 1998

Syntaksi

Uudelleennimeäminen

- Olkoon annettu funktio $f : \mathcal{L} \to \mathcal{L}$ tällaista funktiota kutsutaan uudelleennimeämisfunktioksi.
- $P[f]$ on agentti joka toimii muuten kuin P, mutta sen kommunikaatio tapahtuu funktion f uudelleen nimeämien porttien kautta.
- Tarkastellaan agenttia:
  $Addone \stackrel{\text{def}}{=} in(x).\overline{out}(x+1).Addone$
  sekä funktioita f_1 ja f_2
  $f_1(out) = mid, f_1(\overline{out}) = \overline{mid}, f_1(l) = l$ muuten,
  $f_2(in) = mid, f_2(\overline{in}) = \overline{mid}, f_2(l) = l$ muuten.

Syntaksi

- Nyt voidaan määritellä
  $Addone1 \stackrel{\text{def}}{=} Addone[f_1]$ ja
  $Addone2 \stackrel{\text{def}}{=} Addone[f_2]$.
  Usein on tapan kirjoittaa:
  $Addone1 \stackrel{\text{def}}{=} Addone[\overline{mid}/\overline{out}]$ ja
  $Addone2 \stackrel{\text{def}}{=} Addone[mid/in]$.
- Nyt voidaan muodostaa agentti
  $Addtwo \stackrel{\text{def}}{=} (Addone1|Addone2)\backslash mid$.

©Johan Lilius 1997, Keijo Heljanko 1998

Yhteenveto

- CCS:n syntaksi koostuu seuraavista piirteistä:
- Agentti: Nil.
- Prefixing: $l.P$.
- Summa: $P + Q$.
- Kompositio: $P|Q$.
- Rajoitus: $P\backslash L$.
- Uudelleennimeäminen: $P[f]$.
- Valinta: $\text{if } e \text{ then } P \text{ else } Q$
- Rekursiivinen määritelmä: $agentti \stackrel{\text{def}}{=} lauseke$.

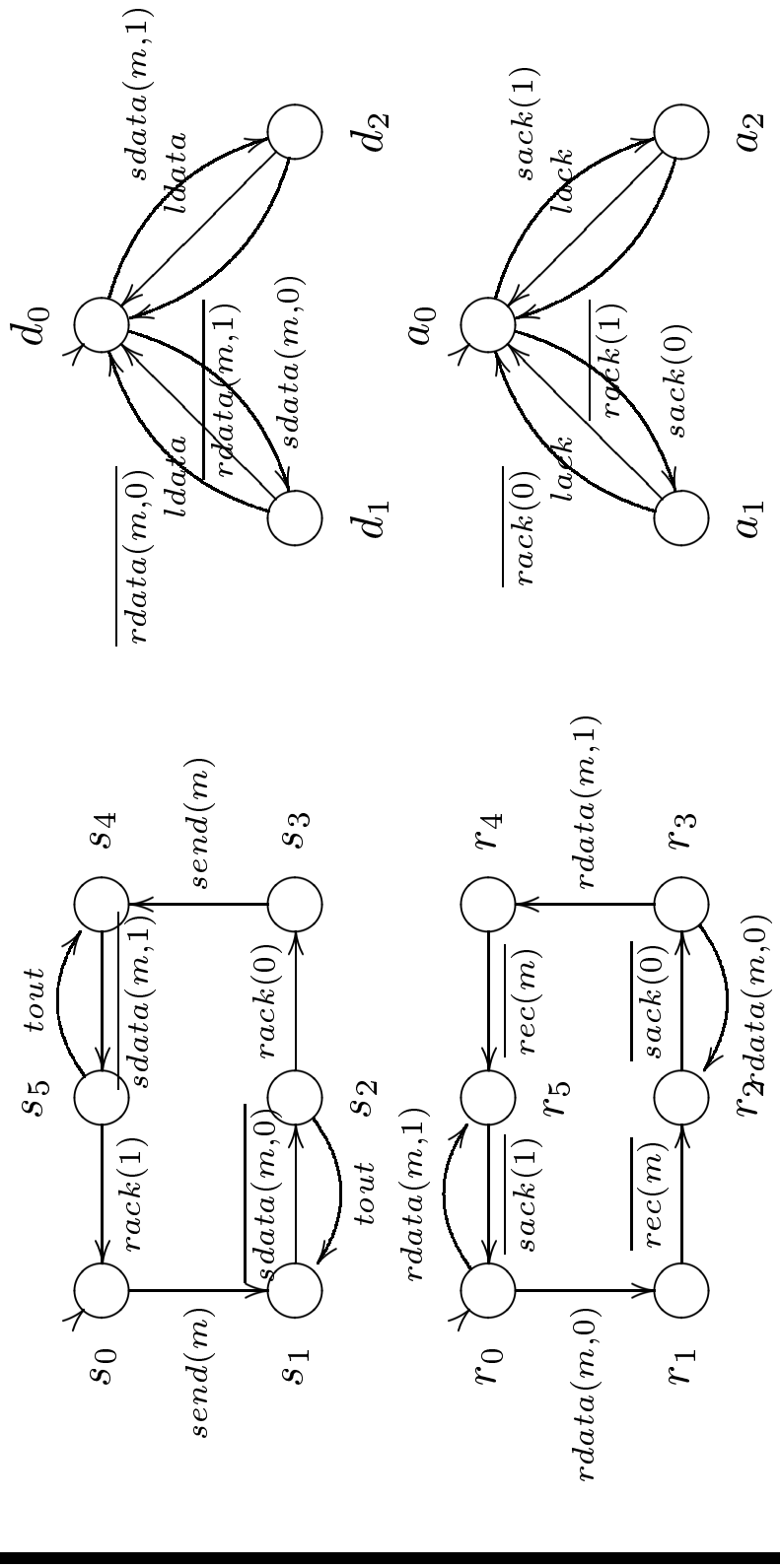
©Johan Lilius 1997, Keijo Heljanko 1998

CCS ja Siirtymäkaaviot

- CCS:n ja siirtymäkaavioiden välillä on selkeä yhteys:
  - Siirtymäkaavion jokainen automaatti määrittelee agentin. Määrittelemällä mitkä kaavion tapahtumista ovat input- ja mitkä output-tapahtumia, saadaan koko siirtymäkaaviota kuvaava agentti automaatteja kuvaavien agenttien rinnankytkentänä. Lisäksi joudutaan vielä suorittamaan rajoitus input- ja output-tapahtumien yli.
  - Vastaavasti jokainen CCS-agentti määrittelee automaatin.
- Tätä yhteyttä on tarkoitus seuraavassa tarkastella tarkemmin.

©Johan Lilius 1997, Keijo Heljanko 1998

CCS JA SIIRTYMÄKAAVIOT



©Johan Lilius 1997, Keijo Heljanko 1998

CCS JA SIIRTYMÄKAAVIOT

- Lähettäjä vastaava agenttilauseke on:

$$\begin{aligned} Sender &\stackrel{\text{def}}{=} send(m).Sending_1 \\ Sending_1 &\stackrel{\text{def}}{=} \overline{sdata(m,0)}.(tout.Sending_1 + rack(0).Sending_2) \\ Sending_2 &\stackrel{\text{def}}{=} send(m).Sending_3 \\ Sending_3 &\stackrel{\text{def}}{=} \overline{sdata(m,1)}.(tout.Sending_3 + rack(1).Sender) \end{aligned}$$

- Agentit $Receiver$, $Data$, ja Ack voidaan konstruoida helposti (harjoitustehtävä).
- AB-protokollaa vastaava agenttilauseke on siten:

$$AB \stackrel{\text{def}}{=} (Sender|Data|Receiver|Ack)\backslash\{sdata, rdata, sack, rack\}$$

©Johan Lilius 1997, Keijo Heljanko 1998

Operationaalinen semantiikka

- Seuraavassa tarkastellaan tarkemmin, mitä agentin käyttäytymisellä tarkoitetaan.
- On hyödyllistä ajatella että prosessi on tietyissä *tiloissa*, ja että se vaihtaa tilaa joidenkin *tapahtumien* johdosta joita kutsutaan *tilasiirtymiksi*.
- Tarkastellaan agenttia:
  $Change \stackrel{\text{def}}{=} fifty.\overline{out}(20).\overline{out}(20).\overline{out}(10).Change.$
- Määritelmä voidaan kirjoittaa myös muotoon:
  $Change \stackrel{\text{def}}{=} fifty.Change'$
  $Change' \stackrel{\text{def}}{=} .\overline{out}(20).Change''$
  $Change'' \stackrel{\text{def}}{=} \overline{out}(20).Change'''$
  $Change''' \stackrel{\text{def}}{=} \overline{out}(10).Change$
- Agenttien $Change, Change', Change'', Change'''$ voidaan ajatella edustavan järjestelmän tiloja. Tällöin toiminnot $fifty$, $\overline{out}(20)$, $\overline{out}(10)$ ovat tilasiirtymiä.
- Jatkossa puhutaan agentista, tilasta ja agentin tilasta synonyymeinä.
- Tilasiirtymiä kuvataan seuraavasti:

$$P \stackrel{a}{\rightarrow} Q$$

©Johan Lilius 1997, Keijo Heljanko 1998

©Johan Lilius 1997, Keijo Heljanko 1998

Kommunikaatio

- Mikä on kommunikaatiotapahtuman formaali määritelmä?
- Oletetaan, että agentti:
  $Person \stackrel{\text{def}}{=} \overline{fifty}.Person'$
  $Person' \stackrel{\text{def}}{=} \ldots$
  kuvaa henkilöä, joka haluaa vaihtaa 50 penniä pienemmäksi.
- Agentti voi suorittaa tilasiirtymän: $Person \stackrel{\overline{fifty}}{\rightarrow} Person'$
- Vastaavasti: $Change \stackrel{fifty}{\rightarrow} Change'$
- Eli on mahdollista suoritta samanaikaisesti toiminnot $fifty$ ja $\overline{fifty}$, eli prosessit voivat kommunikoida.
- $Change|Person \stackrel{?}{\rightarrow} Change'|Person'$.
- Minkalainen tapahtuma on ? ?
- CCS määrittelee kommunikaatiotapahtuman näkymättömäksi tapahtumaksi. Kommunikaatiotapahtumaan osaallistuvien agenttien ympäristö ei pysty havaitsemaan ko. tapahtumaa. Näkymätöntä tapahtumaa merkitään τ.

SÄÄNNÖT

- Nil: Nil agentille ei ole sääntöä.
- Prefixing: $a.P$ on agentti joka voi ensin tehdä a ja sen jälkeen P.
  - $a.P \xrightarrow{a} P$ (Act)
  - $l(x).P \xrightarrow{l_v} P\{v/x\}$ (Act')
  - $\overline{l}(x).P \xrightarrow{l_v} P$ (Act")
- Summa: Agentti $P + Q$ voi tehdä mitä P ja Q voivat tehdä.
  - $P \xrightarrow{a} P' \Longrightarrow P + Q \xrightarrow{a} P'$ (Sum1)
  - $Q \xrightarrow{a} Q' \Longrightarrow P + Q \xrightarrow{a} Q'$ (Sum2)
- Kompositio: Agentti $P|Q$ toimii siten, että agentit P ja Q voivat toimia riippumattomasti tai siten, että ne kommunikoivat:
  - $P \xrightarrow{a} P' \Longrightarrow P|Q \xrightarrow{a} P'|Q$ (Com1)
  - $Q \xrightarrow{a} Q' \Longrightarrow P|Q \xrightarrow{a} P|Q'$ (Com2)
  - $P \xrightarrow{l} P'$ ja $Q \xrightarrow{\overline{l}} Q' \Longrightarrow P|Q \xrightarrow{\tau} P'|Q'$ (Com3)
- Rajoitus: Agentti $P\backslash L$ toimii kuten agentti P, poissulkien toiminnot $l, \overline{l} \in L$:
  - Jos $a, \overline{a} \notin L$ niin $P \xrightarrow{a} P' \Rightarrow P\backslash L \xrightarrow{a} P'\backslash L$ (Res)

©Johan Lilius 1997, Keijo Heljanko 1998

Säännöt

- Kompositio: Agentti $P|Q$ toimii siten, että agentit P ja Q voivat toimia riippumattomasti tai siten, että ne kommunikoivat:
  - $P \xrightarrow{a} P' \Longrightarrow P|Q \xrightarrow{a} P'|Q$ (Com1)
  - $Q \xrightarrow{a} Q' \Longrightarrow P|Q \xrightarrow{a} P|Q'$ (Com2)
  - $P \xrightarrow{l} P'$ ja $Q \xrightarrow{\overline{l}} Q' \Longrightarrow P|Q \xrightarrow{\tau} P'|Q$ (Com3)
- Rajoitus: Agentti $P\backslash L$ toimii kuten agentti P, poissulkien toiminnot $l, \overline{l} \in L$:
  - Jos $a, \overline{a} \notin L$ niin $P \xrightarrow{a} P' \Rightarrow P\backslash L \xrightarrow{a} P'\backslash L$ (Res)
- Uudelleennimeäminen: Agentti $P[f]$ on agentti jonka toiminta on sama kuin P:n mutta jossa kaikki nimet on nimetty uudelleen funktion f avulla:
  - $P \xrightarrow{a} P' \Rightarrow P[f] \xrightarrow{f(a)} P'[f]$ (Rel)

  Lisäksi vaaditaan, että $f(\tau) = \tau$.

©Johan Lilius 1997, Keijo Heljanko 1998

Säännöt

- Valinta: Agentti $\text{if } e \text{ then } P \text{ else } Q$ toimii kuten P tai Q riippuen lausekkeen e arvosta.
  - Jos e on tosi niin:
    $P \xrightarrow{a} P' \Rightarrow \text{ if } e \text{ then } P \text{ else } Q \xrightarrow{a} P'$ (Cond1)
  - Jos e on epätosi niin:
    $Q \xrightarrow{a} Q' \Rightarrow \text{ if } e \text{ then } P \text{ else } Q \xrightarrow{a} Q'$ (Cond2)
- Määrittelylauseke: $P \stackrel{\text{def}}{=} A$ tarkoitta, että P:n käyttäytyminen on määritelty A:n avulla:
  - $A \xrightarrow{a} P' \Rightarrow P \xrightarrow{a} P'$ (Rec)

©Johan Lilius 1997, Keijo Heljanko 1998

Semantiikka

- Sääntöjen avulla tehtyä päättelyä merkitään usein seuraavasti:

$$\frac{P_1 \stackrel{a_1}{\rightarrow} P_1' \ldots P_n \stackrel{a_n}{\rightarrow} P_n'}{P \stackrel{a}{\rightarrow} P'} \text{(X)}$$

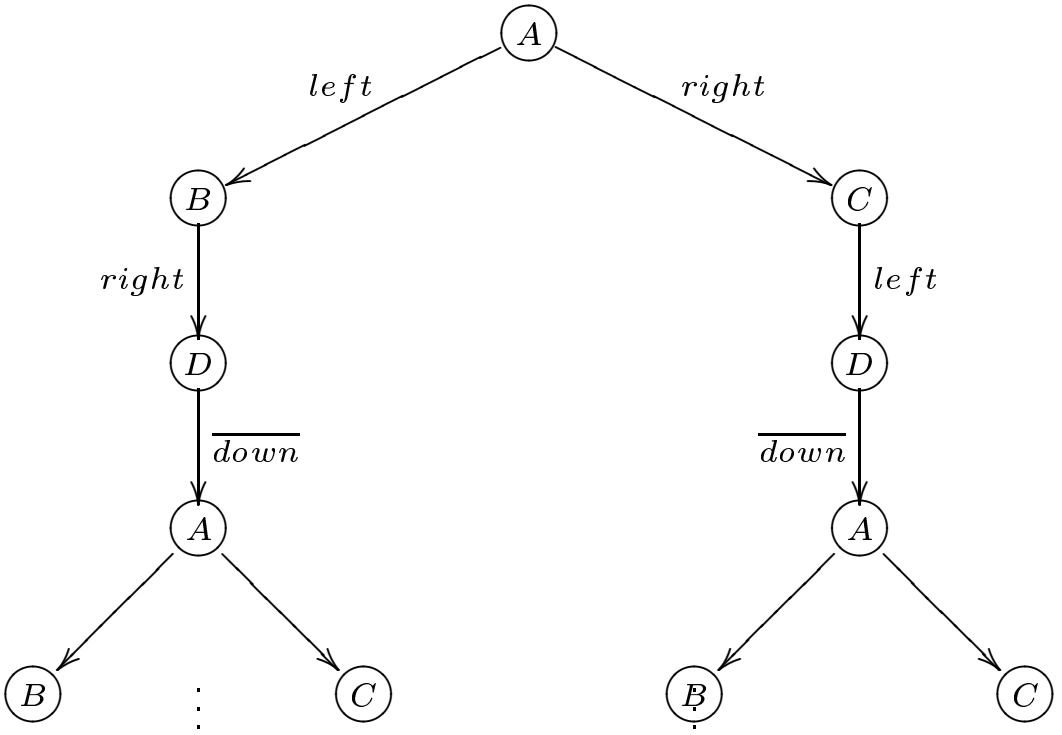
- Lauseke $Change|Person \stackrel{\tau}{\rightarrow} Change'|Person'$ voidaan nyt johtaa seuraavasti:

$$\frac{\dfrac{}{Person \stackrel{\overline{fifty}}{\rightarrow} Person'} \text{(Act)} \quad \dfrac{}{Change \stackrel{fifty}{\rightarrow} Change'} \text{(Act)}}{Change|Person \stackrel{\tau}{\rightarrow} Change'|Person'} \text{(Com3)}$$

©Johan Lilius 1997, Keijo Heljanko 1998

Semantiikka

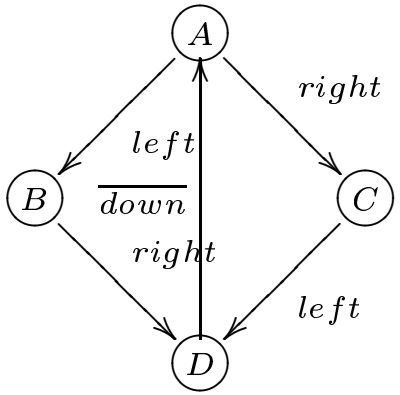
- Usein käytetään johtojen sijaan transitiopuita tai transitiodiagrammeja.
- Agentin A
  $A \stackrel{\text{def}}{=} left.right.\overline{down}.A + right.left.\overline{down}.A$
  transitiopuu on:



©Johan Lilius 1997, Keijo Heljanko 1998

Semantiikka

- vastaava transitiodiagrammi on:



©Johan Lilius 1997, Keijo Heljanko 1998

Ekvivalentti käyttäytyminen

- Mitä tarkoitetaan kun sanotaan, että kahden järjestelmän P ja Q käyttäytyminen on sama ?
- Pyritään muodostamaan relaatio järjestelmien tilojen ja tapahtumien välille.
- *Havaintoekvivalenssi*: Kahden järjestelmän ulkoisten kommunikointitapahtumien välinen ekvivalenssirelaatio.
- *Havaintokongruenssi*: Havaintoekvivalenssi joka on yhteensopiva prosessialgebran operaatioiden kanssa.
- Relaatio ei saa olla liian vahva, eikä liian heikko.
- Havaintokongruenssin tulisi olla sellainen, että:
  - $P = Q$ jos Q on P:n tarkennus (refinement),
  - jos $P = Q$ ja P' on P:n osajärjestelmä jolle pätee $P' = R$ niin $P[R/P'] = Q$

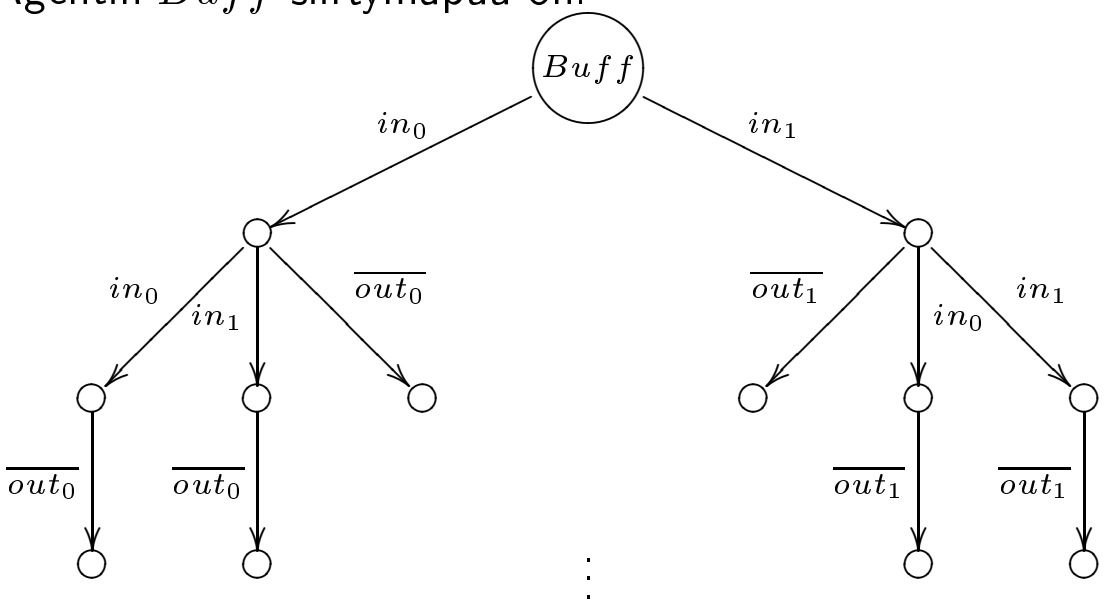
©Johan Lilius 1997, Keijo Heljanko 1998

Ekvivalentti käyttäytyminen

Tarkastellaan agentteja:

$$
\begin{aligned}
Buff &\stackrel{\text{def}}{=} in(x).Buff'(x) \\
Buff'(x) &\stackrel{\text{def}}{=} in(y).Buff''(x,y) + \overline{out(x)}.Buff \\
Buff''(x,y) &\stackrel{\text{def}}{=} \overline{out}(x).Buff'(y)
\end{aligned}
$$

Agentin $Buff$ siirtymäpuu on:



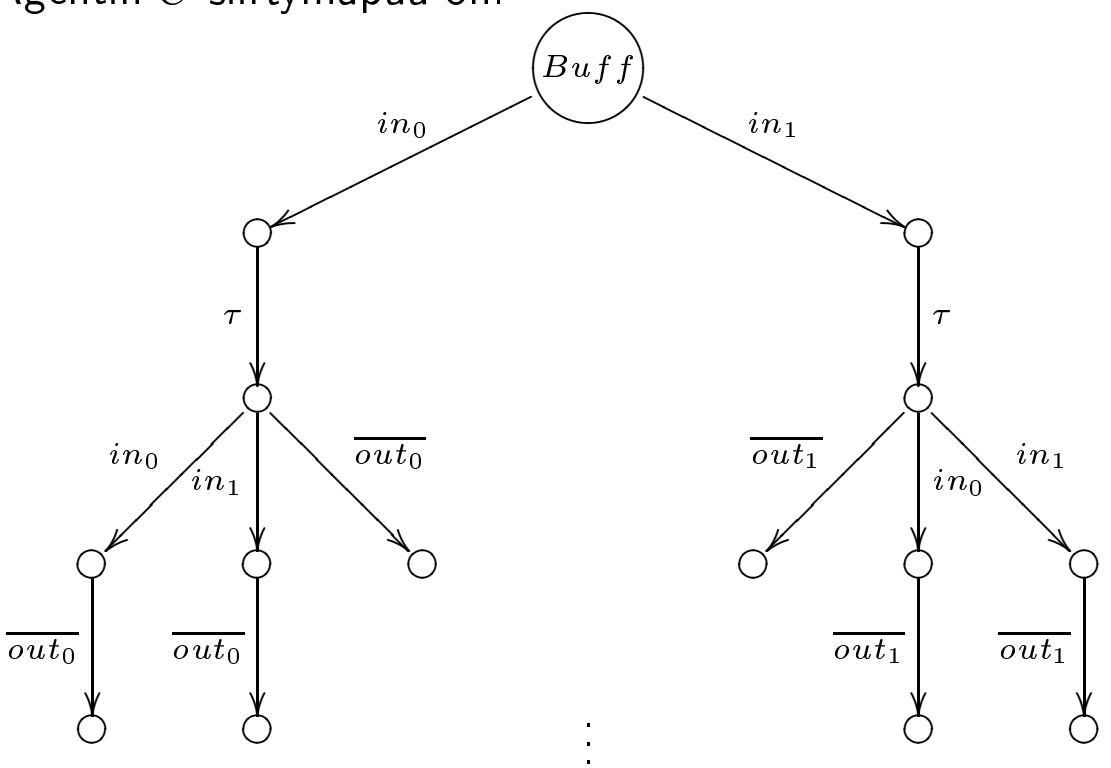
©Johan Lilius 1997, Keijo Heljanko 1998

Ekvivalentti käyttäytyminen

Tarkastellaan agentteja:

$$
\begin{aligned}
Cell &\stackrel{\text{def}}{=} in(x).Cell'(x) \\
Cell'(x) &\stackrel{\text{def}}{=} \overline{out}(x).Cell \\
C1 &\stackrel{\text{def}}{=} Cell[\overline{mid}/\overline{out}] \\
C2 &\stackrel{\text{def}}{=} Cell[mid/in] \\
C &\stackrel{\text{def}}{=} (C1 \mid C2)\backslash\{mid\}
\end{aligned}
$$

Agentin C siirtymäpuu on:



©Johan Lilius 1997, Keijo Heljanko 1998

Ekvivalentti käyttäytyminen

- Koska τ-toiminnot ovat näkymättömiä voidaan sanoa, että agenteilla C ja $Buff$ on sama havaittavissa oleva käyttäytyminen.
- Tästä seuraa, että agentti C voidaan missä tahansa järjestelmässä korvata agentilla $Buff$ ja päinvastoin.
- Korvavuus voidaan formalisoida relaation $=$ välillä.
- Relaation tulisi toteuttaa seuraavat ehdot:

(1) $P = P$,

(2) jos $P = Q$ niin $Q = P$,

(3) jos $P = Q$ ja $Q = R$ niin $P = R$,

(4) jos $P = Q$ niin

$$a.P = a.Q$$

$$P + R = Q + R$$

$$P|R = Q|R$$

$$P\backslash L = Q\backslash L$$

$$\text{if } e \text{ then } P \text{ else } R = \text{ if } e \text{ then } Q \text{ else } R$$

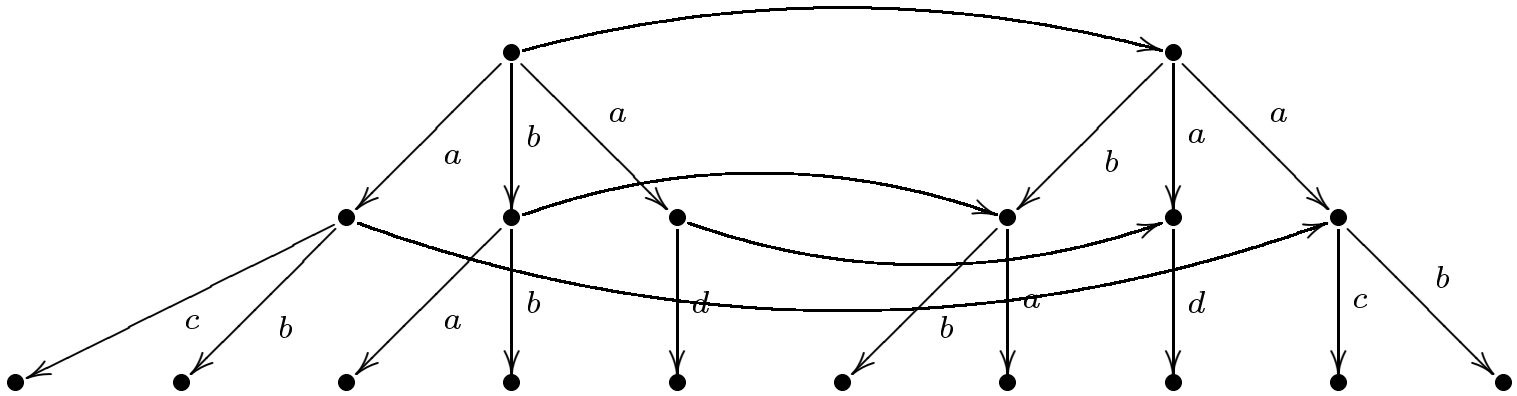
jos $A \stackrel{\text{def}}{=} P$ ja $B \stackrel{\text{def}}{=} Q$ niin $A = B$.

- Relaatio joka toteuttaa ehdot (1)-(3) on **HAVAINTOEKVIVALENSSI**.
- Relaatio joka toteuttaa ehdot (1)-(4) **HAVAINTOKONGRUENSSI**

©Johan Lilius 1997, Keijo Heljanko 1998

Ekvivalentti käyttäytyminen

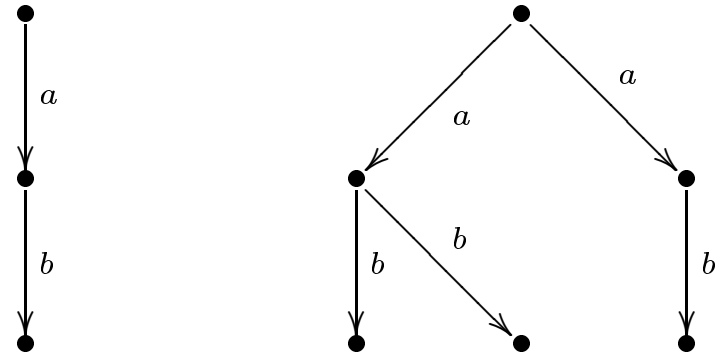
- Millainen on sitten sopiva relaatio?
  - Syntaktinen identiteetti ?
  - Liian vahva: C ja $Buff$ ei olisi kongruentteja.
  - Siirtymäpuiden isomorfia ?



©Johan Lilius 1997, Keijo Heljanko 1998

Ekvivalentti käyttäytyminen

- Liian vahva koska puut



eivät ole ekvivalentteja, mutta agentien $a.b.Nil$ ja $a.b.Nil + a.(b.Nil + b.Nil)$ haluttaisiin olevan.

- Agentit ovat ekvivalentteja jos ne voivat suorittaa samat tilasiirtymäketjut ?

$$P = Q \text{ jos } P \xrightarrow{s} P' \text{ joss } Q \xrightarrow{s} Q'$$

Nyt em. puut ovat ekvivalentteja.

©Johan Lilius 1997, Keijo Heljanko 1998

Ekvivalentti käyttäytyminen

- Ekvivalenssi on kuitenkin liian heikko:

$$a.(b.Nil + c.Nil) = a.b.Nil + a.c.Nil$$

  oikeanpuoleisen agentin on kuitenkin mahdollista suorittaa a siten, että c ei ole mahdollinen seuraaja.
- Ekvivalenssin tulisi säilyttää valinnan "ajankohta".
- SIIS:
  - Ekvivalenttien agenttien täytyy pystyä suorittamaan samat toiminnot.
  - Tietyn toiminnon suorituksen jälkeisien tilojen tulee olla "ekvivalentteja".

©Johan Lilius 1997, Keijo Heljanko 1998

Ekvivalentti käyttäytyminen

- Määritelmä (*): P ja Q ovat ekvivalentteja joss jokaiselle toiminnolle a:
  - jos P' s.e. $P \stackrel{a}{\rightarrow} P'$ niin on olemassa Q' s.e. $Q \stackrel{a}{\rightarrow} Q'$ ja P' ja Q' ovat ekvivalentteja, tai
  - jos Q' s.e. $Q \stackrel{a}{\rightarrow} Q'$ niin on olemassa P' s.e. $P \stackrel{a}{\rightarrow} P'$ ja Q' ja P' ovat ekvivalentteja, tai
- Määritelmä huono: ekvivalenssi määritellään ekvivalenssiksi.
- Osoittautuu kuitenkin, että on olemassa relaatioita jotka toteuttavat määritelmän (*), ja näiden joukossa on olemassa "suurin" joka vielä on kongruenssi.
- Määritelmä (*) unohtaa kuitenkin τ-toiminnot.
- Esimerkiksi agentit $\tau.Nil$ ja Nil sekä agentit $a.Nil$ ja $a.\tau.Nil$ eivät ole ekvivalentteja.

©Johan Lilius 1997, Keijo Heljanko 1998

Ekvivalentti käyttäytyminen

- Notaatio: $P \stackrel{a}{\Rightarrow} Q$ joss $P \stackrel{<\tau>^m<a><\tau>^n}{\rightarrow} Q$
- Esim: jos $P \stackrel{\text{def}}{=} \tau.a.\tau.Nil$ niin $P \stackrel{a}{\Rightarrow} Nil$
- Huom: $P \stackrel{\tau}{\Rightarrow} Q$ joss P voi muuntua Q:ksi suorittamalla *vähiintään* yhden τ-toiminnon.
- Määritelmä (**): P ja Q ovat ekvivalentteja joss jokaiselle toiminnolle a:
  - jos P' s.e. $P \stackrel{a}{\rightarrow} P'$ niin on olemassa Q' s.e. $Q \stackrel{a}{\Rightarrow} Q'$ ja P' ja Q' ovat ekvivalentteja, tai
  - jos Q' s.e. $Q \stackrel{a}{\rightarrow} Q'$ niin on olemassa P' s.e. $P \stackrel{a}{\Rightarrow} P'$ ja Q' ja P' ovat ekvivalentteja, tai
- Ei kuitenkaan haluttu, koska esimerkiksi agentit $\tau.Nil$ ja Nil sekä agentit $a.Nil$ ja $a.\tau.Nil$ eivät vieläkään ole ekvivalentteja. (Miksi?)

©Johan Lilius 1997, Keijo Heljanko 1998

Ekvivalentti käyttäytyminen

- Määritelmä (***): P ja Q ovat ekvivalentteja joss jokaiselle toiminnolle a:
  - jos P' s.e. $P \xrightarrow{a} P'$ niin *joko* (1) on olemassa Q' s.e. $Q \stackrel{a}{\Rightarrow} Q'$ ja P' ja Q' ovat ekvivalentteja, *tai* (2) $a = \tau$ ja P' ja Q ovat ekvivalentteja, sekä vastaavasti
  - jos Q' s.e. $Q \xrightarrow{a} Q'$ niin *joko* (1) on olemassa P' s.e. $P \stackrel{a}{\Rightarrow} P'$ ja Q' ja P' ovat ekvivalentteja, *tai* (2) $a = \tau$ ja Q' ja P ovat ekvivalentteja.
- Tämä määritelmä vastaa parhaiten intuitiota siitä milloin kaksi agenttia ovat ekvivalentteja.

©Johan Lilius 1997, Keijo Heljanko 1998

©Johan Lilius 1997, Keijo Heljanko 1998

Ekvivalentti käyttäytyminen

- Määritelmää (***) ei voida suoraan ottaa havaintoekvivalenssin määritelmäksi.
- Osoittautuu, että on olemassa *suurin* relaatio joka toteuttaa määritelmän (***). Merkitään sitä $\approx$.
- Se voidaan määritellä seuraavasti:
- Määritelmä: Relaatio $\approx$ on **HAVAINTOEKVIVALENSSI** silloin kuin $P \approx Q$ joss jokaiselle toiminnolle a:
  - jos P' s.e. $P \stackrel{a}{\rightarrow} P'$ niin *joko* (1) on olemassa Q' s.e. $Q \stackrel{a}{\Rightarrow} Q'$ ja $P' \approx Q'$, *tai* (2) $a = \tau$ ja $P' \approx Q$, sekä vastaavasti
  - jos Q' s.e. $Q \stackrel{a}{\rightarrow} Q'$ niin *joko* on olemassa P' s.e. $P \stackrel{a}{\Rightarrow} P'$ ja $Q' \approx P'$, *tai* (2) $a = \tau$ ja $Q' \approx P$.

  Lisäksi $\approx$ on suurin tällainen relaatio, s.e. jos $\mathcal{R}$ on jokin toinen relaatio joka toteuttaa määritelmän (***) ja jos $P\mathcal{R}Q$ niin $P \approx Q$.

Ekvivalentti käyttäytyminen

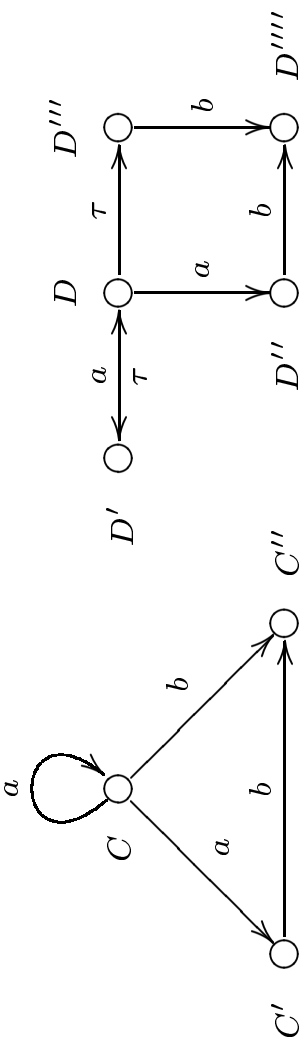
- $\approx$ on "melkein" havaintokongruenssi.
- Esim. $P \approx Q \Rightarrow P + R \approx Q + R$ ei päde koska, $\tau.a.Nil \approx a.Nil$, mutta $\tau.a.Nil + l.Nil \not\approx a.Nil + l.Nil$
- Havaintokongruenssi on suurin relaatio joka sisältyy havaintoekvivalenssiin ja on agenttien kongruenssi.
- Merkitään $P = Q$.

©Johan Lilius 1997, Keijo Heljanko 1998

ESIMERKKI

$$C \stackrel{\text{def}}{=} a.C + a.b.Nil + b.Nil$$

$$D \stackrel{\text{def}}{=} a.\tau.D + \tau.b.Nil + a.b.Nil$$



$C \cong D$?

©Johan Lilius 1997, Keijo Heljanko 1998

Havaintoekvivalenssin ominaisuuksia

- Määritellään lakeja joiden avulla kahden agentin lauseketta voidaan yksinkertaistaa.
- Neljä ryhmää:
  - *Dynaamiset lait* määrittelevät dynaamisten operaattoreiden ominaisuuksia:
    - prefixing ja
    - summa.
  - *Staattiset lait* määrittelevät staattisten operaattoreiden ominaisuuksia:
    - rinnnakkainen kompositio,
    - rajoitus ja
    - uudelleennimeäminen.
  - *Ekspansioperiaate*, jolla määritellään dynaamisten ja staattisten lakien yhteys.
  - *Rekursiosäännöt*, joilla määritellään rekursion ominaisuuksia.

©Johan Lilius 1997, Keijo Heljanko 1998

Dynaamiset lait

1.

$$P + Q = Q + P \quad (1)$$

$$P + (Q + R) = (P + Q) + R \quad (2)$$

$$P + P = P \quad (3)$$

$$P + Nil = P \quad (4)$$

2.

$$a.\tau.P = a.P \quad (5)$$

$$P + \tau.P = \tau.P \quad (6)$$

$$a.(P + \tau.Q) + a.Q = a.(P + \tau.Q) \quad (7)$$

$$P + \tau.(P + Q) = \tau.(P + Q) \quad (8)$$

©Johan Lilius 1997, Keijo Heljanko 1998

Non-laws

- Seuraavat ehkä intuitiivisesti oikeilta tuntuvat säännöt *EIVÄT* päde:
  - $$P \stackrel{?}{=} \tau.P$$
  - "distributiivisuus"
    $$a.(P + Q) \stackrel{?}{=} a.P + a.Q$$

©Johan Lilius 1997, Keijo Heljanko 1998

Staattiset lait

1.

$$P|Q = Q|P \quad (9)$$

$$P|(Q|R) = (P|Q)|R \quad (10)$$

$$P|Nil = P \quad (11)$$

2.

$$P\backslash L = P \text{ joss } sort(P) \cap (L \cup \overline{L}) = \emptyset \quad (12)$$

$$((P\backslash L)\backslash K) = P\backslash(L \cup K) \quad (13)$$

$$(P[f])\backslash L = (P\backslash f^{-1}(L))[f] \quad (14)$$

$$(P|Q)\backslash L = (P)\backslash L)|(Q)\backslash L) \quad (15)$$

$$\text{joss } sort(P) \cap \overline{sort(Q)} \cap (L \cup \overline{L}) = \emptyset \quad (16)$$

©Johan Lilius 1997, Keijo Heljanko 1998

Staattiset lait

1.

$$P[Id_L] = P \quad (17)$$

$$P[f] = P[g] \text{ jos } f(sort(P)) = g(sort(P)) \quad (18)$$

$$(P[f])[g] = P[g \circ f] \quad (19)$$

$$(P|Q)[f] = (P[f])|(Q[f]) \quad (20)$$

$$\text{jos } f \text{ on injektiivinen joukolla } sort(P|Q) \cup \overline{sort(P|Q)} \quad (21)$$

2.

$$P[k/l] = P \text{ jos } l, \overline{l} \notin sort(P) \quad (22)$$

$$P\backslash l = (P[k/l])\backslash k \text{ jos } k, \overline{k} \notin sort(P) \quad (23)$$

$$(P\backslash L)[k/l] = (P[k/l])\backslash L \text{ jos } k, l \notin L \cup \overline{L} \quad (24)$$

(25)

©Johan Lilius 1997, Keijo Heljanko 1998

Ekspansioperiaate

- Järjestelmä on usein helpointa kuvata osiensa rinnakkaisena kompositiona ja rajoituksena eli muodossa $(P_1|P_2|\ldots|P_n)\backslash L$
- Ekspansioperiaate kertoo miten staattiset operaattorit voidaan purkaa auki dynaamisiksi operaattoreiksi.
- Dynaamisten lakien avulla voidaan sitten johtaa ekvivalentti agentti.
- Ekspansioperiaate antaa prosessialgebralle **lomitusemantiikan**:

  PA "concurrency = choice + sequential composition"

  PV "concurrency = causal independence" ("true" concurrency)

©Johan Lilius 1997, Keijo Heljanko 1998

Ekspansioperiaate

$$
\begin{aligned}
P_1|\ldots|P_n \quad = \quad & \Sigma\{a.(P_1|\ldots|P_i'|\ldots|P_n) \mid 1 \leq i \leq n,\ P_i \xrightarrow{a} P_i' \ \} \\
& +\Sigma\{\tau.(P_1|\ldots|P_i'|\ldots|P_j'|\ldots|P_n) \\
& |1 \leq i \leq j \leq n, \exists l:\ P_i \xrightarrow{l} P_i' \ ,\ P_j \xrightarrow{\bar{l}} P_j' \ \} \qquad (26)
\end{aligned}
$$

$$
\begin{aligned}
(P_1[f_1]|\ldots|P_n[f_n])\backslash L \quad = \quad & \Sigma\{f_i(a).(P_1[f_1]|\ldots|P_i'[f_i]|\ldots|P_n[f_n])\backslash L \\
& |1 \leq i \leq n,\ P_i \xrightarrow{a} P_i' \ , f_i(a) \notin L \cup \overline{L}\} \\
& +\Sigma\{\tau.(P_1[f_1]|\ldots|P_i'[f_i]|\ldots|P_j'[f_j]|\ldots|P_n[f_n])\backslash L \\
& |1 \leq i < j \leq n, \exists l, k:\ P_i \xrightarrow{l} P_i' \ ,\ P_j \xrightarrow{k} P_j' \ \}, \\
& \overline{f_i(l)} = f_j(k)\} \qquad (27)
\end{aligned}
$$

©Johan Lilius 1997, Keijo Heljanko 1998

Rekursiolait

- Jos $A \stackrel{\text{def}}{=} P$ niin $A = P$.
- Miten helpottaa rekursiivisesti määriteltyjen agenttien analyysiä?
- Siis: Miten todistetaan, että agenteilla

$$\begin{aligned} A &\stackrel{\text{def}}{=} l.A \\ B &\stackrel{\text{def}}{=} l.l.B \end{aligned}$$

  on sama käyttäytyminen.
- Otetaan käyttöön agenttimuuttujia.
- Oletetaan, että meillä on annettuna yhtälö $X = E$, missä X on agenttimuuttuja ja E ei sisällä muita agenttimuutujia kuin X.
- Pyritään osoittamaan, että:
  1. Aina kun P ja Q ovat yhtälön ratkaisuja, niin $P = Q$, ja
  2. A ja B ovat yhtälön $X = E$ ratkaisuja.

©Johan Lilius 1997, Keijo Heljanko 1998

Rekursiolait

- Kohta 2. on todettavissa em. lakien avulla.
- Kohta 1. edellyttää, että yhtälölle pätevät tietyt ehdot:
  - Yhtälössä $X = E$ agenttimuuttujaa X edeltää näkyvä toiminto (X is guarded). Esim.

    $$l.X + Y$$

    mutta ei

    $$l.X + X.$$

  - Jotta ratkaisut olisivat yksikäsitteisiä vaaditaan vielä sekventiaalisuus. X on sekventiaalinen E:ssä joss mikään X:n esiintymä ei sijaitse staattisen operaattorin "sisällä". Esim.

    $$X + a.Nil, a.X + (Nil|b.Y)$$

    mutta ei

    $$a.X|b.Nil, a.X + (a.X|Y)\backslash b$$

©Johan Lilius 1997, Keijo Heljanko 1998

Rekursiolait

- Laki 9B
  Olkoon I indeksointijoukko, $\{X_i|i \in I\}$ agenttimuuttujien joukko ja $\{E_i|i \in I\}$ agenttilausekkeiden joukko se. $Var(\{E_i|i \in I\}) = \{X_i|i \in I\}$. Oletetaan lisäksi, että jokaista muuttuja X_i edeltää näkyvä toiminto ja jokainen muuttuja X_i on sekventiaalinen lausekkeessa E_j. Jos nyt $\{P_i|i \in I\}$ ja $\{Q_i|i \in I\}$ ovat agenttijoukoja se. jokaiselle i sekä P_i, että Q_i ovat yhtälön $X_i = E_i$ ratkaisuja, niin jokaiselle $i \in I$ $P_i = Q_i$.

©Johan Lilius 1997, Keijo Heljanko 1998

Rekursiolait

- Tarkastellaan agentteja:

$$\begin{aligned} Buff &\stackrel{\text{def}}{=} in_0.B_0 + in_1.B_1 \\ B_i &\stackrel{\text{def}}{=} in_0.B_{i0} + in_1.B_{i1} + \overline{out_i}.Buff \\ B_ij &\stackrel{\text{def}}{=} \overline{out_i}.Buff_j \end{aligned}$$

ja

$$\begin{aligned} A &\stackrel{\text{def}}{=} in_0.A_0 + in_1.A_1 \\ A_i &\stackrel{\text{def}}{=} \overline{mid_i}.A \\ A' &\stackrel{\text{def}}{=} mid_0.A'_0 + mid_1.A'_1 \\ A'_i &\stackrel{\text{def}}{=} \overline{out_i}.A \\ C &\stackrel{\text{def}}{=} (A \mid A')\backslash\{mid_0, mid_1\} \end{aligned}$$

©Johan Lilius 1997, Keijo Heljanko 1998

Rekursiolait

- Osoittakseemme, että $C = Buff$ täytyy meidän osoitaa, että sekä C että $Buff$ ovat seuraavan yhtälöryhmän ratkaisuja:

$$\begin{aligned}
X_1 &= in_0.X_2 + in_1.X_3 \\
X_2 &= in_0.X_4 + in_1.X_5 + \overline{out_0}.X_1 \\
X_3 &= in_0.X_6 + in_1.X_7 + \overline{out_1}.X_1 \\
X_4 &= \overline{out_0}.X_2 \\
X_5 &= \overline{out_0}.X_3 \\
X_6 &= \overline{out_1}.X_2 \\
X_7 &= \overline{out_1}.X_3
\end{aligned}$$

- Sijoittamalla $Buff, B_0, \ldots$ yhtälöön nähdään, että $Buff$ on yhtälöryhmän ratkaisu.
- Vastaavasti soveltamalla expansioperiaatetta iteratiivisesti saadaan agentti C muotoon, josta on helppo nähdä, että sekin on yhtälöryhmän ratkaisu.

©Johan Lilius 1997, Keijo Heljanko 1998

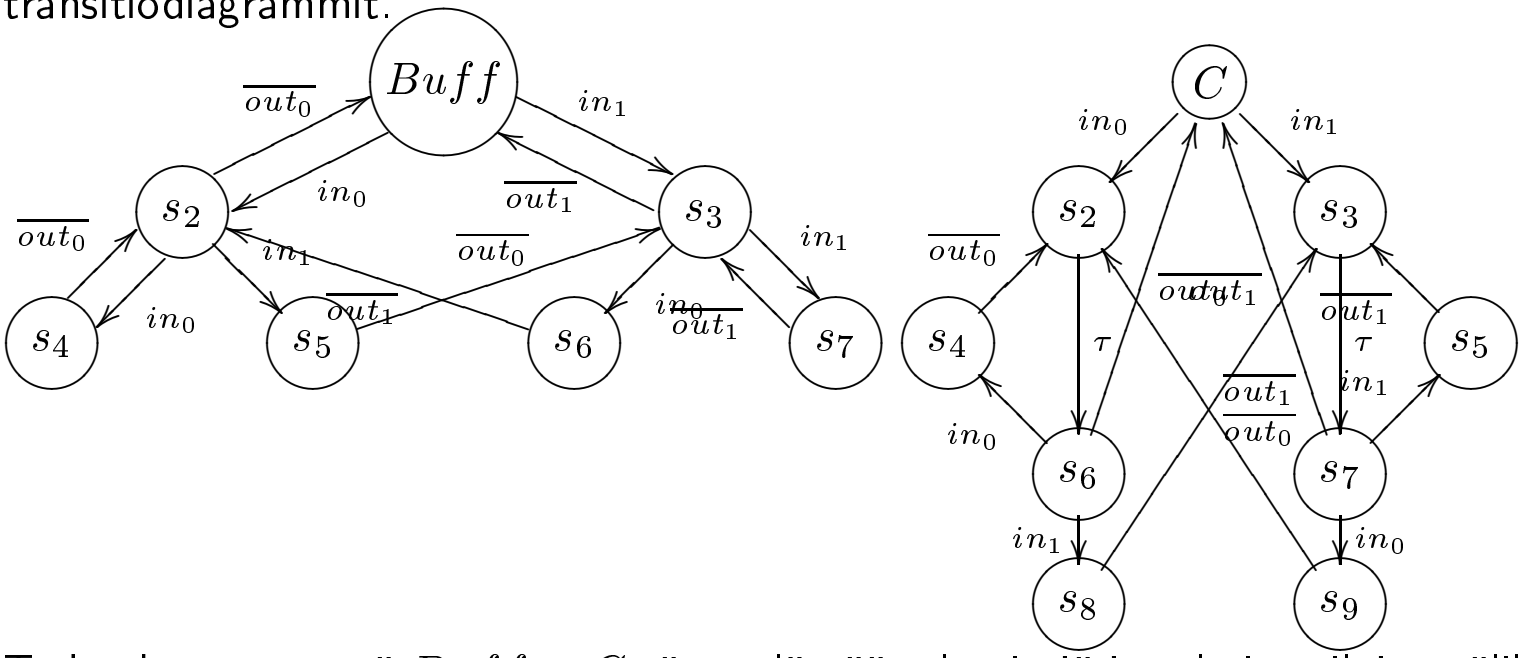
©Johan Lilius 1997, Keijo Heljanko 1998

Bisimulaatio

- Bisimulaatio on menetelmä jonka avulla voidaan tarkastaa ovatko kaksi agenttia havaintoekvivalentteja
- Yhdessä havaintokongruenssilakien kanssa saadaan menetelmä jolla agenttien käyttäytymistä voidaan analysoida.
- (***) Havaintoekvivalenssi $\approx$ on suurin relaatio, siten että $P \approx Q$ joss jokaiselle toiminnolle a:
  - jos P' s.e. $P \xrightarrow{a} P'$ niin *joko*
    1. on olemassa Q' s.e. $Q \stackrel{a}{\Rightarrow} Q'$ ja $P' \approx Q'$, *tai*
    2. $a = \tau$ ja $P' \approx Q$, sekä vastaavasti
  - jos Q' s.e. $Q \xrightarrow{a} Q'$ niin *joko*
    1. on olemassa P' s.e. $P \stackrel{a}{\Rightarrow} P'$ ja $Q' \approx P'$, *tai*
    2. $a = \tau$ ja $Q' \approx P$.

Bisimulaatio

- Tarkastellaan jälleen agentteja $Buff$ ja C. Niiden johtopuista saadaan seuraavat transitiodiagrammit.



- Todetaksemme, että $Buff \approx C$ täytyy löytää relaatio järjestelmien tilojen välillä.

©Johan Lilius 1997, Keijo Heljanko 1998

Bisimulaatio

- Kun aloitetaan olettamalla, että $(Buff, C)$ on tällainen pari saadaan soveltamalla määritelmää (***) seuraavaat vastaavuudet:

$$\begin{aligned} R \;=\; & \{(Buff, C), \\ & (B_0, (A_0|A')\backslash L), (B_0, (A|A'_0)\backslash L), \\ & (B_1, (A_1|A')\backslash L), (B_1, (A|A'_1)\backslash L), \\ & (B_{00}, (A_0|A'_0)\backslash L), (B_{01}, (A_1|A'_0)\backslash L), \\ & (B_{10}, (A_1|A'_0)\backslash L), (B_{10}, (A_0|A'_1)\backslash L), \\ & (B_{11}, (A_1|A'_1)\backslash L)\} \end{aligned}$$

- Tämä vastaavuus muodostaa osan koko havaintoekvivalenssista.

©Johan Lilius 1997, Keijo Heljanko 1998

Bisimulaatio

- Sille pätee:

  Jos $(P, Q) \in R$ niin jokaiselle toiminnolle a:

  - jos P' s.e. $P \xrightarrow{a} P'$ niin *joko*
    1. on olemassa Q' s.e. $Q \stackrel{a}{\Rightarrow} Q'$ ja $(P', Q') \in R$, *tai*
    2. $a = \tau$ ja $(P', Q) \in R$, sekä vastaavasti
  - jos Q' s.e. $Q \xrightarrow{a} Q'$ niin *joko*
    1. on olemassa P' s.e. $P \stackrel{a}{\Rightarrow} P'$ ja $(Q', P') \in R$, *tai*
    2. $a = \tau$ ja $(Q', P) \in R$.
- Tällainen relaatio on *BISIMULAATIO*.
- Bisimulaatio toteuttaa ainostaan ehdon (***) vasemmalta oikealle.
- Se voidaan kuitenkin yksikäsitteisesti täydentää havaintoekvivalenssiksi.

©Johan Lilius 1997, Keijo Heljanko 1998

Bisimulaatio

- Milloin $P \approx Q \Rightarrow P = Q$?
- Syy siihen, että aina ei $P \approx Q \Rightarrow P = Q$ on τ-toiminnon poissulkeva vaikutus.
- Jos P voi aloittaa tekemällä näkymättömän toiminnon, voi se siten estää muita mahdollisia käyttäytymisiä.
- Nyt $Buff$, ja C eivät kumpikaan voi aloittaa τ-toiminnolla.
- Tällainen agentti on *STABIILI*.
- Stabiilisuusperiaate:
  Jos P ja Q ovat stabiileja, ja $P \approx Q$ niin $P = Q$.

©Johan Lilius 1997, Keijo Heljanko 1998

Bisimulaatio

- Stabiilisuusperiaate kuvaa havaintoekvivalenssin ja havaintokongruenssin eroa.
- Havaintokongruenssi voidaan myös määritellä eksplisiittisesti:
  $P = Q$ joss jokaiselle toiminnolle a:
  - jos P' s.e. $P \xrightarrow{a} P'$ niin on olemassa Q' s.e. $Q \stackrel{a}{\Rightarrow} Q'$ ja $P' \approx Q'$, sekä vastaavasti
  - jos Q' s.e. $Q \xrightarrow{a} Q'$ niin on olemassa P' s.e. $P \stackrel{a}{\Rightarrow} P'$ ja $Q' \approx P'$.

©Johan Lilius 1997, Keijo Heljanko 1998

Bisimulaatio

- Tarkastellaan agentteja:

$$
\begin{aligned}
Sem0 &\stackrel{\text{def}}{=} get.Sem1 \\
Sem1 &\stackrel{\text{def}}{=} get.Sem2 + put.Sem0 \\
Sem2 &\stackrel{\text{def}}{=} get.Sem3 + put.Sem1 \\
Sem3 &\stackrel{\text{def}}{=} put.Sem2
\end{aligned}
$$

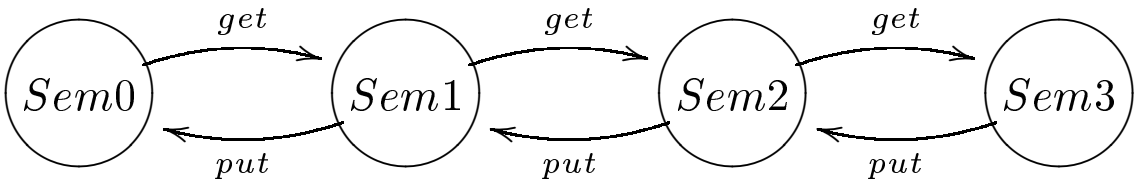
sekä

$$
\begin{aligned}
S &\stackrel{\text{def}}{=} get.S' \\
S' &\stackrel{\text{def}}{=} put.S \\
T &\stackrel{\text{def}}{=} S|S|S
\end{aligned}
$$

©Johan Lilius 1997, Keijo Heljanko 1998

BISIMULAATIO

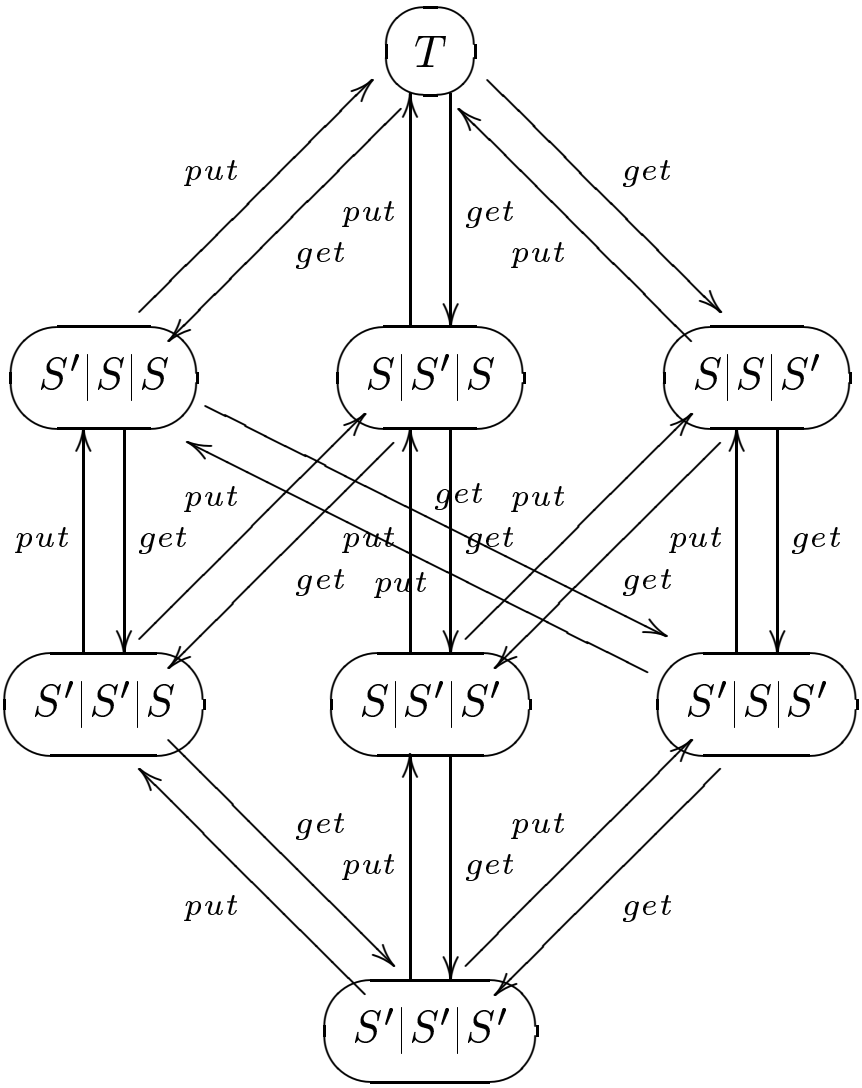
- $Sem0$ kuvaa ternäärisemaforia. Sen transitiodiagrammi on:



©Johan Lilius 1997, Keijo Heljanko 1998

Bisimulaatio

- Vastaavasti T:n transitiodiagrammi on:



©Johan Lilius 1997, Keijo Heljanko 1998

Bisimulaatio

- Osoitetaan, että $Sem0 \approx T$.
- Haetaan siis bisimulaatiota joka sisältää parin $(Sem0, T)$.
- Täällainen on:

$$\begin{aligned} R \quad = \quad & \{(Sem0, T), \\ & (Sem1, S'|S|S), (Sem1, S|S'|S), \\ & (Sem1, S|S|S'), (Sem2, S'|S'|S), \\ & (Sem2, S'|S|S'), (Sem2, S|S'|S'), \\ & (Sem3, S'|S'|S')\} \end{aligned}$$

- Nyt kuitenkin voidaan osoitaa, että

$$S'|S|S \approx S|S'|S \approx S|S|S'$$

ja

$$S'|S'|S \approx S'|S|S' \approx S|S'|S'$$

- Eli onko tehty turhaa työtä?

©Johan Lilius 1997, Keijo Heljanko 1998

Bisimulaatio

- Bisimulaation määritelmää voidaan muuttaa siten, että sitä muodostettaessa voidaan hyödyntää havaintokongruenssin lakeja:
- Jos $(P, Q) \in R$ niin jokaiselle toiminnolle a:
  - jos P' s.e. $P \stackrel{a}{\rightarrow} P'$ niin *joko*
    1. on olemassa P'', Q', Q'' s.e. $Q \stackrel{a}{\Rightarrow} Q'$ ja $P' \approx P'', Q' \approx Q''$ ja $(P'', Q'') \in R$, *tai*
    2. $a = \tau$ ja $(P', Q) \in R$, sekä vastaavasti
  - jos Q' s.e. $Q \stackrel{a}{\rightarrow} Q'$ niin *joko*
    1. on olemassa P' s.e. $P \stackrel{a}{\Rightarrow} P'$ ja $Q' \approx Q'', P' \approx P''$ $(P'', Q'') \in R$, *tai*
    2. $a = \tau$ ja $(Q', P) \in R$.

©Johan Lilius 1997, Keijo Heljanko 1998

Esimerkki: Alternating-bit protokolla

- Tarkastellaan jälleen alternating-bit protokollaa.
- Protokollan oikeaksi todistaminen:
  - Protokollan tarkoituksena on toimia puskurina.
  - Siis:
    1. Rakennetaan agentti AB joka vastaa protokollaa.
    2. Rakennetaan agentti $Buff$ joka vastaa puskuria.
    3. Haetaan bisimulaatio jonka avulla osoitettaan, että $AB \approx Buff$.

©Johan Lilius 1997, Keijo Heljanko 1998

Esimerkki: Alternating-bit protokolla

- Viestit lähetetään biteillä 0 ja 1 "taggattuna".
- Lähettäjäprosessi: Saatuaan toimeksiannon, viesti lähetetään yhdessä bitin b kanssa.
  - Jos viesti on hävinnyt saa prosessi time-outin ja viesti lähetetään uudestaan bitin b kanssa.
  - Jos prosessi vastaannottaa bitin b kättelynä se lähettää uuden viestin bitin $\hat{b} = 1 - b$ kanssa.
  - Jos prosessi vastaannottaa bitin $\hat{b}$ kättelynä se unohdetaan.
- Vastaannottajaprosessi: prosessi vastaannottaa viestin yhdessä bitin b kanssa. Bitti b lähetetään kättelynä takaisin.
  - Jos kättely on hävinnyt saa prosessi time-outin ja kättely uusitaan.
  - Jos prosessi vastaannottaa uuden viestin ja bitin $\hat{b}$ tämä kuitataan.
  - Jos prosessi vastaannottaa uudestaan edellisen viestin yhdessä bitin b kanssa, se unohdetaan.
- Huom: Tässä myös vastaanottajaprosessi voi tehdä time-outin (vrt. ensimmäisen luennon ABP-malli).

©Johan Lilius 1997, Keijo Heljanko 1998

Esimerkki: Alternating-bit protokolla

- Agenttien lausekkeet ovat:

$$
\begin{aligned}
Send(b) &\stackrel{\text{def}}{=} \overline{\texttt{send}_b}.\overline{\texttt{time}}.Sending(b) \\
Sending(b) &\stackrel{\text{def}}{=} \texttt{timeout}.Send(b) + \texttt{ack}_b.\texttt{timeout}.Accept(\hat{b}) \\
&\quad + \texttt{ack}_{\hat{b}}.Sending(b) \\
Accept(b) &\stackrel{\text{def}}{=} \texttt{accept}.Send(b) \\
Reply(b) &\stackrel{\text{def}}{=} \overline{\texttt{reply}_b}.\overline{\texttt{time}}.Replying(b) \\
Replying(b) &\stackrel{\text{def}}{=} \texttt{timeout}.Reply(b) + \texttt{trans}_{\hat{b}}.\texttt{timeout}.Deliver(\hat{b}) \\
&\quad + \texttt{trans}_b.Replying(b) \\
Deliver(b) &\stackrel{\text{def}}{=} \overline{\texttt{deliver}}.Reply(b) \\
Timer &\stackrel{\text{def}}{=} \texttt{time}.\overline{\texttt{timeout}}.Timer
\end{aligned}
$$

©Johan Lilius 1997, Keijo Heljanko 1998

Esimerkki: Alternating-bit protokolla

- Linjat on helpompi kuvata transitiosäännöillä:

$$Ack(bs) \overset{\overline{\mathtt{ack}_b}}{\rightarrow} Ack(s) \qquad Trans(sb) \overset{\overline{\mathtt{trans}_b}}{\rightarrow} Trans(s)$$

$$Ack(s) \overset{\mathtt{reply}_b}{\rightarrow} Ack(sb) \qquad Trans(s) \overset{\mathtt{send}_b}{\rightarrow} Trans(bs)$$

$$Ack(sbt) \overset{\tau}{\rightarrow} Ack(st) \qquad Trans(tbs) \overset{\tau}{\rightarrow} Trans(ts)$$

$$Ack(sbt) \overset{\tau}{\rightarrow} Ack(sbbt) \qquad Trans(tbs) \overset{\tau}{\rightarrow} Trans(tbbs)$$

- Nyt kannattaa suorittaa seuraavat yksinkertaistukset:

$$Send'(b) \overset{\text{def}}{=} (Send(b)|Timer)\backslash\{\mathtt{time}, \mathtt{timeout}\}$$

$$Reply'(b) \overset{\text{def}}{=} (Reply(b)|Timer)\backslash\{\mathtt{time}, \mathtt{timeout}\}$$

©Johan Lilius 1997, Keijo Heljanko 1998

Esimerkki: Alternating-bit protokolla

- Agenttilausekkeet saadaan nyt muotoon:

$$
\begin{aligned}
Send(b) &\stackrel{\text{def}}{=} \overline{\texttt{send}_b}.Sending(b) \\
Sending(b) &\stackrel{\text{def}}{=} \tau.Send(b) + \texttt{ack}_b.Accept(\hat{b}) + \texttt{ack}_{\hat{b}}.Sending(b) \\
Accept(b) &\stackrel{\text{def}}{=} \texttt{accept}.Send(b) \\
Reply(b) &\stackrel{\text{def}}{=} \overline{\texttt{reply}_b}.Replying(b) \\
Replying(b) &\stackrel{\text{def}}{=} \tau.Reply(b) + \texttt{trans}_{\hat{b}}.Deliver(\hat{b}) + \texttt{trans}_b.Replying(b) \\
Deliver(b) &\stackrel{\text{def}}{=} \overline{\texttt{deliver}}.Reply(b)
\end{aligned}
$$

- Protokolla voidaan nyt määritellä seuraavasti:

$$AB = Accept(\hat{b})|Trans(\epsilon)|Ack(\epsilon)|Reply(b)$$

©Johan Lilius 1997, Keijo Heljanko 1998

Esimerkki: Alternating-bit protokolla

- Protokollan tulee toteuttaa seuraava spesifikaatio:

$$Buff \stackrel{\text{def}}{=} \texttt{accept}.Buff'$$
$$Buff' \stackrel{\text{def}}{=} \overline{\texttt{deliver}}.Buff$$

- Tämän todistamiseksi on löydettävä bisimulaatio se.:

$$AB \approx Buff$$

- On huomattava, että

$$AB \neq Buff$$

  koska AB on epästabiili.

©Johan Lilius 1997, Keijo Heljanko 1998

Esimerkki: Alternating-bit protokolla

- Tila-avaruus voidaan jakaa siten, että saadaan seuraava bisimulaatio ehdokas:

$$Accept(\hat{b})|Trans(b^n)|Ack(b^p)| \left\{ \begin{array}{ll} Reply(b) & \\ Replying(b) & \end{array} \right. \quad Buff$$

$$\left. \begin{array}{l} Send(\hat{b}) \\ Sending(\hat{b}) \end{array} \right\} |Trans(\hat{b}^m b^n)|Ack(b^p)| \left\{ \begin{array}{ll} Reply(b) & \\ Replying(b) & \end{array} \right. \quad Buff'$$

$$\left. \begin{array}{l} Send(\hat{b}) \\ Sending(\hat{b}) \end{array} \right\} |Trans(\hat{b}^m)|Ack(b^p)|Deliver(\hat{b}) \quad Buff'$$

$$\left. \begin{array}{l} Send(\hat{b}) \\ Sending(\hat{b}) \end{array} \right\} |Trans(\hat{b}^m)|Ack(b^p\hat{b}^q)| \left\{ \begin{array}{ll} Reply(\hat{b}) & \\ Replying(\hat{b}) & \end{array} \right. \quad Buff$$

©Johan Lilius 1997, Keijo Heljanko 1998

Esimerkki: Alternating-bit protokolla

- Tarkastellaan esim. paria:

$$(Sending(\hat{b})|Trans(\hat{b}^m b^n)|Ack(b^p)|Reply(b), Buff')$$

- Sen vasen alkio voi:
  - tehdä τ-toimintoja jotka vastaavat $Trans$ tai Ack toimintojen toistoa. Kaikki vastaavat seuraajatilat kuuluvat samaan ryhmään joten voimme valita $Buff'$ tilan omaksi τ-seuraajaksi.
  - τ-toiminto

$$Sending(\hat{b}) \xrightarrow{\tau} Send(\hat{b})$$

    vie meidät toiseen ryhmään, joka on myös relaatiossa $Buff'$:n kanssa.
  - Jos $Sending(b)$ tekee $\overline{\mathtt{ack}_b}$ toiminnon, voi $Ack(b^p)$ vastaavasti tehdä $\mathtt{ack}_b$ toiminnon, jolloin syntyvä τ-toiminto vie meidät samaan ryhmään.
  - Lopuksi, jos $Reply(b)$ tekee $\overline{\mathtt{reply}_b}$ toiminnon voi $Ack(b^p)$ vastaavasti tehdä $\mathtt{reply}_b$ toiminnon, jolloin syntyvä τ-toiminto vie meidät eri ryhmään, joka kuitenkin on relaatiossa $Buff'$:n kanssa.

©Johan Lilius 1997, Keijo Heljanko 1998

Esimerkki: Alternating-bit protokolla

- Toisaalta $Buff'$ voi ainoastaan tehdä

$$Buff' \overset{\overline{\texttt{deliver}}}{\rightarrow} Buff.$$

Vastaavasti protokolla voi tehdä $\overline{\texttt{reply}_b}$, $\texttt{trans}_b$ (n kertaa), sekä $\texttt{trans}_{\hat{b}}$ jotka tuottavat τ-toimintoja, sekä lopuksi $\overline{\texttt{deliver}}$-toiminnon. Saatu tila on

$$Sending(\hat{b})|Trans(\hat{b}^{m-1})|Ack(b^{p+1})|Reply(\hat{b}),$$

joka on relaatiossa $Buff$ agentin kanssa.

©Johan Lilius 1997, Keijo Heljanko 1998