T-79.179 Rinnakkaiset ja hajautetut digitaaliset järjestelmät

Prosessialgebra

Marko Mäkelä

24. maaliskuuta 2003

Taustaa

Rinnakkaiset ja hajautetut järjestelmät koostuvat usein melko irrallisista prosesseista, jotka kommunikoivat silloin tällöin keskenään viestien tai jaetun muistin välityksellä.

Petri-verkoilla on näppärä esittää jaettua muistia (siirtymien yhteisinä paikkoina), mutta perusmuodossaan ne eivät mahdollista prosessien erottamista toisistaan.* Verkon paikoista ja siirtymistä ei suoraan näe, ovatko ne jonkin prosessin sisäisiä (ja jos ovat, niin minkä prosessin) vai liittyvätkö ne kommunikointiin. Osittaisjärjestysmenetelmät pyrkivät hankkimaan tämän tiedon jälkikäteen, mikä ei aina onnistu tehokkaasti.

Prosessialgebra keskittyy nimenomaan siirtymien esittämiseen ja *kompositionaaliseen mallintamiseen*, jossa prosessit mallinnetaan erikseen ja niitä kytketään toisiinsa. Prosessialgebrat perustuvat siirtymäjärjestelmiin ja samankaltaisuus- eli *ekvivalenssirelaatioihin*, joiden suhteen siirtymäjärjestelmiä voidaan verrata toisiinsa tai kutistaa.

*Modulaariset verkot ovat eräs vaihtoehto.

CCS ja CSP

Kaksi tunnetuinta prosessialgebraa lienevät Robin Milnerin CCS *(Calculus of Communicating Systems)* ja Tony Hoaren CSP *(Communicating Sequential Processes).*

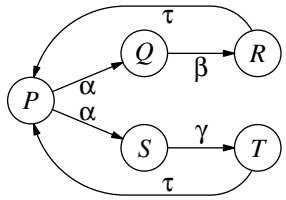
CCS erottelee kolmenlaisia tapahtumia: prosessin eli *agentin* sisäisiä (τ), vastaanottoa (α) ja lähetystä ($\overline{\alpha}$). Kommunikointi on aina kahden agentin välistä.

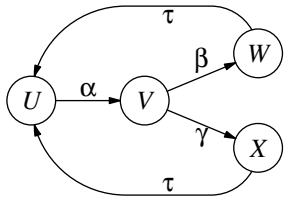
Alun perin (1978) Hoare esitti CSP:n laajentamalla Dijkstran esittämää *guarded command* -kieltä lähetys- ja vastaanottokäskyillä, mutta nykymuotoinen CSP (1985) ei erottele lähetystä ja vastaanottoa, vaan tapahtumaan voi osallistua monta agenttia.

Rinnankytkentäsäännöissä tahdistukset voidaan määritellä monella tavalla. Voidaan määritellä vaikkapa niin, että prosessi voi suorittaa tapahtuman ainoastaan samanaikaisesti kaikkien samannimisen tapahtuman sisältävien prosessien kanssa. Jos osallistuminen tahdistukseen määritellään vapaaehtoiseksi, rinnankytkennän koko kasvaa helposti.

Tämä luento keskittyy CCS-prosessialgebraan.

Prosessialgebran ja siirtymäjärjestelmien vastaavuus





$P ::= \alpha \,.\, Q + \alpha \,.\, S$
$Q ::= \beta \,.\, R$
$R ::= \tau \,.\, P$
$S ::= \gamma \,.\, T$
$T ::= \tau \,.\, P$

$P ::= \alpha \,.\, \beta \,.\, \tau \,.\, P + \alpha \,.\, \gamma \,.\, \tau \,.\, P$

$U ::= \alpha \,.\, V$
$V ::= \beta \,.\, W + \gamma \,.\, X$
$W ::= \tau \,.\, U$
$X ::= \tau \,.\, U$

$U ::= \alpha \,.\, (\beta \,.\, \tau \,.\, U + \gamma \,.\, \tau \,.\, U)$

Prosessialgebrojen perusteita

- Jokaisella agentilla on nimi. Matemaattisessa tekstissä nimiä merkitään tavallisesti suuraakkosilla $P, Q, R, \ldots$; työkalut saattavat käyttää esimerkiksi lukuja $0, 1, 2, \ldots$.
- *Näkymätön tapahtuma* τ kuvaa agentin sisäistä laskentaa. Niitä muut agentit eivät voi havaita, eikä niiden suoritukseen voi vaikuttaa.
- *Näkyviä tapahtumia* merkitään matemaattisessa tekstissä kreikkalaisilla aakkosilla $\alpha, \beta, \gamma, \ldots$. Käytetyt rinnankytkentäsäännöt määräävät, miten agentit voivat tahdistua suorittamalla näkyviä tapahtumia samanaikaisesti.
- Prosessialgebra keskittyy tapahtumiin. Merkintä $P \xrightarrow{\alpha} Q$ tarkoittaa, että agentti P käyttäytyy kuten agentti Q suoritettuaan tapahtuman α.

CCS:n erityispiirteitä

- Jokaisella näkyvällä tapahtumalla $\alpha \neq \tau$ on *komplementtitapahtuma* $\overline{\alpha}$. Komplementtitapahtuman komplementtitapahtuma on tapahtuma itse: $\overline{\overline{\alpha}} = \alpha$.
- Tapahtuman ja sen komplementtitapahtuman voidaan ajatella tarkoittavan saman sanoman vastaanottoa ja lähetystä.
- Rinnankytkentäsääntö koskee kahta agenttia kerrallaan: jos Q kykenee ottamaan sanoman γ vastaan ja R lähettämään sen tai päinvastoin ($Q \xrightarrow{\gamma} Q', R \xrightarrow{\overline{\gamma}} R'$), niin $(Q \| R) \xrightarrow{\tau} (Q' \| R')$. Agentit voivat myös edetä yksitellen: jos $Q \xrightarrow{\beta} Q''$, niin $(Q \| R) \xrightarrow{\beta} (Q'' \| R)$.* Rinnankytkentä on symmetrinen: $(Q \| R) = (R \| Q)$.

*Tämä pätee myös, vaikka R voisi edetä $\overline{\beta}$:lla.

CCS:n kielioppi ja semantiikka (1/2)

Seuraavassa esitämme CCS-prosessialgebran kieliopin ja laskusäännöt.

Tyhjä agentti (Nil) ei kykene suorittamaan mitään tapahtumaa: $\nexists \alpha, P : Nil \xrightarrow{\alpha} P$.

Tapahtuman liittäminen *(Action Prefixing)* Olkoon Q agentti ja olkoon α (näkymätön tai näkyvä) tapahtuma. Tällöin $P ::= \alpha\,.\,(Q)$ on agentti ja $P \xrightarrow{\alpha} Q$. *Agentti P kykenee suorittamaan tapahtuman α ja käyttäytymään sen jälkeen kuten agentti Q.*

Valinta *(Choice)* Olkoot Q ja R agentteja ja α ja β tapahtumia. $P ::= (Q + R)$ on agentti. Jos $Q \xrightarrow{\alpha} Q'$, niin $P \xrightarrow{\alpha} Q'$. Jos $R \xrightarrow{\beta} R'$, niin $P \xrightarrow{\beta} R'$. *Agentti P voi "alkutilassaan" valita, käyttäytyykö se niin kuin Q vai niin kuin R.*

Rajoitus *(Restriction)* Olkoon Q agentti ja Σ joukko näkyviä tapahtumia, $\tau \notin \Sigma$. Tällöin $P ::= (Q) \setminus \Sigma$ on agentti. Jos $Q \xrightarrow{\alpha} Q'$ ja $\alpha \notin \Sigma, \overline{\alpha} \notin \Sigma$, niin $((Q) \setminus \Sigma) \xrightarrow{\alpha} ((Q') \setminus \Sigma)$. *Agentti P on muuten kuten agentti Q, mutta se ei kykene suorittamaan joukon Σ tapahtumia eikä niiden komplementtitapahtumia.*

CCS:n kielioppi ja semantiikka (2/2)

Uudelleennimeäminen *(Relabeling)* Olkoon Q agentti ja Σ sen näkyvien tapahtumien joukko, $\tau \notin \Sigma$. Olkoon Σ' tapahtumien joukko ja $m : \Sigma \cup \{\tau\} \to \Sigma' \cup \{\tau\}$ siten, että $m(\tau) = \tau$ ja $\forall \alpha \neq \tau : m(\overline{\alpha}) = \overline{m(\alpha)}$. Tällöin $P ::= Q[m]$ on agentti. Jos $Q \xrightarrow{\alpha} Q'$, niin $Q[m] \xrightarrow{m(\alpha)} Q'[m]$. *Agentti P on muuten kuin Q, mutta sen tapahtumat on saatu Q:n tapahtumista kuvauksen m mukaan.* Erikoistapaus: piilotus ($m(\alpha) = m(\overline{\alpha}) = \tau$).

Rinnankytkentä *(Parallel Composition)* Olkoot Q ja R agentteja. $(Q \| R)$ on agentti.

1. Jos $Q \xrightarrow{\gamma} Q'$ ja $R \xrightarrow{\overline{\gamma}} R'$, niin $(Q \| R) \xrightarrow{\tau} (Q' \| R')$.
2. Jos $Q \xrightarrow{\gamma} Q''$, niin $(Q \| R) \xrightarrow{\gamma} (Q'' \| R)$.
3. Jos $R \xrightarrow{\gamma} R''$, niin $(Q \| R) \xrightarrow{\gamma} (Q \| R'')$.

Prosessialgebrojen kieliopista ja semantiikasta

Eri prosessialgebroissa on määritelty hieman erilaisia rakenteita ja laskentasääntöjä. Niiden sisäistäminen voi viedä aikansa, sillä ne esitetään usein formaalin tiiviisti ja rekursiivisessa muodossa, kuten edellä.

Luettavuuden parantamiseksi jätetään usein sulkuja pois. Sulkujen kanssa kirjoitettaisiin esimerkiksi $U ::= \alpha\,.\,((\beta\,.\,(\tau\,.\,(U)) + \gamma\,.\,(\tau\,.\,(U))))$.

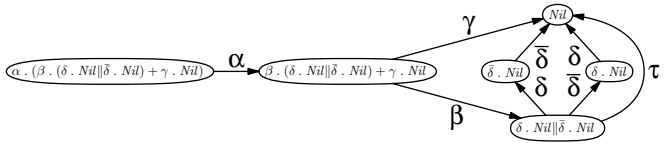
Kysymys: onko määritys $U ::= \alpha\,.\,(\beta\,.\,\tau + \gamma\,.\,\tau)\,.\,U$ esittämiemme sääntöjen mukainen, jos $\alpha, \beta, \gamma, \tau$ ovat tapahtumien nimiä? Entä $U ::= \alpha\,.\,(\beta + \gamma)\,.\,\tau\,.\,U$?

Usein prosessialgebrassa esitetään erilaisia lyhennysmerkintöjä. Esimerkiksi $.\,Nil$ voidaan jättää pois. Muuttuvatko äskeiset määritykset sääntöjen mukaisiksi, jos niissä on käytetty tätä lyhennysmerkintää?

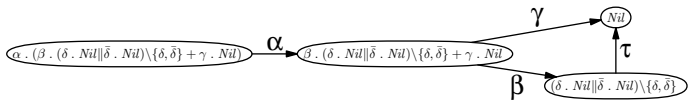
Marko Mäkelä

Rinnankytkentäesimerkki

Seuraava kuva esittää agentin $\alpha\,.\,(\beta\,.\,(\delta\,.\,Nil \| \overline{\delta}\,.\,Nil) + \gamma\,.\,Nil)$ käyttäytymistä:



Kuvan siirtymäjärjestelmä on muodostettu soveltamalla äskeisiä laskusääntöjä. Sen kuhunkin tilaan on merkitty siinä mahdollista käyttäytymistä vastaavan agentin lauseke. Agentin $\delta\,.\,Nil \| \overline{\delta}\,.\,Nil$ käyttäytyminen sisältää tapahtumien δ ja $\overline{\delta}$ lomitetut suoritukset. Monesti rinnankytkentäsäännössä sallitaan vain samanaikainen tahdistuminen:



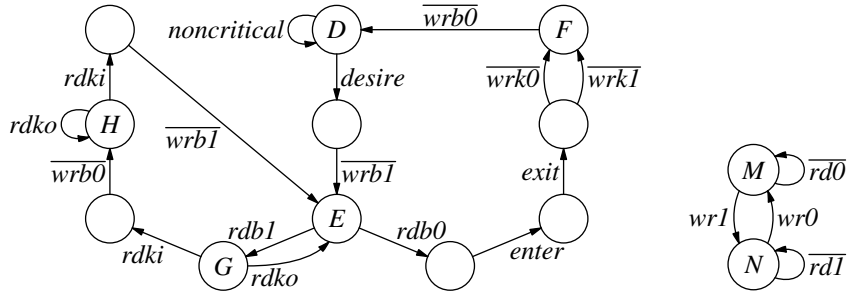
Olkoon $P|Q$ tällainen rajoitettu rinnankytkentä. Harjoitustehtävä: määrittele se formaalisti.

Marko Mäkelä

Esimerkki: Dekkerin algoritmi (1/6)

Kuvataan Dekkerin kahden prosessin keskinäisen poissulkevuuden algoritmi CCS:llä.*

Koska prosessialgebrat ja siirtymäjärjestelmät eivät tunne muistin käsitettä, se on mallinnettava erikseen. Määritellään aluksi perusprosessit: ohjelmakoodi ja yksibittinen muisti:



*Sivulla http://www.tcs.hut.fi/Software/maria/tools/tvt/tvt.html on laajempi, CSP-tyylistä rinnankytkentää käyttävä esimerkki, jossa on tarkastettu algoritmin reiluutta.

Marko Mäkelä

Esimerkki: Dekkerin algoritmi (2/6)

Tekstimuodossa perusprosessit voidaan kirjoittaa muun muassa seuraavasti:

$$
\begin{aligned}
M &::= \overline{rd0}\,.\,M + wr1\,.\,N \\
N &::= \overline{rd1}\,.\,N + wr0\,.\,M \\
D &::= noncritical\,.\,D + desire\,.\,\overline{wrb1}\,.\,E \\
E &::= rdb0\,.\,enter\,.\,exit\,.\,(\overline{wrk0}\,.\,F + \overline{wrk1}\,.\,F) + rdb1\,.\,G \\
F &::= \overline{wrb0}\,.\,D \\
G &::= rdko\,.\,E + rdki\,.\,\overline{wrb0}\,.\,H \\
H &::= rdko\,.\,H + rdki\,.\,\overline{wrb1}\,.\,E
\end{aligned}
$$

Marko Mäkelä

Esimerkki: Dekkerin algoritmi (3/6)

Dekkerin algoritmissa on kaksi prosessia ja kolme jaettua muistipaikkaa. Muodostetaan ne perusprosesseista uudelleennimeämisen avulla. CSP-rinnankytkentää sovellettaessa kaksiporttinen muisti K voisi olla kuin yksiporttiset muistit B_1 ja B_2, mutta CCS:ssä se on määriteltävä eri tavalla. Harjoitustehtävä: Miksi? Miten?

$$
\begin{array}{rcl}
B_1 & ::= & M[c(\{rd0 \mapsto b^1_\bot, rd1 \mapsto b^1_\top, wr0 \mapsto b^1_{\top\to\bot}, wr1 \mapsto b^1_{\bot\to\top}\})] \\
B_2 & ::= & M[c(\{rd0 \mapsto b^2_\bot, rd1 \mapsto b^2_\top, wr0 \mapsto b^2_{\top\to\bot}, wr1 \mapsto b^2_{\bot\to\top}\})] \\
P_1 & ::= & D[c(\{wrb1 \mapsto b^1_{\bot\to\top}, wrb0 \mapsto b^1_{\top\to\bot}, wrk0 \mapsto k^1_{2\to1}, wrk1 \mapsto k^1_{1\to2}, \\
 & & \quad rdb0 \mapsto b^2_\bot, rdb1 \mapsto b^2_\top, rdko \mapsto k^1_2, rdki \mapsto k^1_1\})] \\
P_2 & ::= & D[c(\{wrb1 \mapsto b^2_{\bot\to\top}, wrb0 \mapsto b^2_{\top\to\bot}, wrk0 \mapsto k^2_{2\to1}, wrk1 \mapsto k^2_{1\to2}, \\
 & & \quad rdb0 \mapsto b^1_\bot, rdb1 \mapsto b^1_\top, rdko \mapsto k^2_1, rdki \mapsto k^2_2\})]
\end{array}
$$

Yllä kuvaus $c(\Sigma) = \bigcup_{\alpha\in\Sigma}\{\alpha, \overline{\alpha}\}$ selkeyttää uudelleennimeämisjoukkoja, koska komplementtitapahtumia ei tarvitse luetella.

Marko Mäkelä

Esimerkki: Dekkerin algoritmi (4/6)

Kokonaiskäyttäytymisen esittää $Dekker ::= K|B_1|B_2|P_1|P_2$. Näiden agenttien rakenteen ja CCS:n rinnankytkennän symmetrisyyden perusteella rinnankytkentäjärjestyksellä ei ole väliä. Lasketaan $BP_1 ::= B_1|P_1$, josta saadaan $BP_2 ::= B_2|P_2$ uudelleennimeämisellä, jonka jälkeen voitaisiin laskea $Dekker = K|BP_1|BP_2$. Avataan ensin B_1 ja P_1:

$$B_1 ::= \overline{b^1_\bot} \,.\, B_1 + b^1_{\bot\to\top} \,.\, B'_1 \qquad\qquad B'_1 ::= \overline{b^1_\top} \,.\, B'_1 + b^1_{\top\to\bot} \,.\, B_1$$

$$
\begin{array}{rcl}
P_1 & ::= & noncritical \,.\, P_1 + desire \,.\, \overline{b^1_{\bot\to\top}} \,.\, P'_1 \\
P'_1 & ::= & b^2_\bot \,.\, enter \,.\, exit \,.\, (\overline{k^1_{2\to1}} \,.\, P''_1 + \overline{k^1_{1\to2}} \,.\, P''_1) + b^2_\top \,.\, P'''_1 \\
P''_1 & ::= & \overline{b^1_{\top\to\bot}} \,.\, P_1 \\
P'''_1 & ::= & k^1_2 \,.\, P'_1 + k^1_1 \,.\, \overline{b^1_{\top\to\bot}} \,.\, P''''_1 \\
P''''_1 & ::= & k^1_2 \,.\, P''''_1 + k^1_1 \,.\, \overline{b^1_{\bot\to\top}} \,.\, P'_1
\end{array}
$$

Marko Mäkelä

Esimerkki: Dekkerin algoritmi (5/6)

Lasketaan $BP_1 ::= B_1|P_1$. Nyt tapahtumia ei voi liittää kuin yhden kerrallaan, sillä kuhunkin agenttiin johtaa monta tapahtumaa:

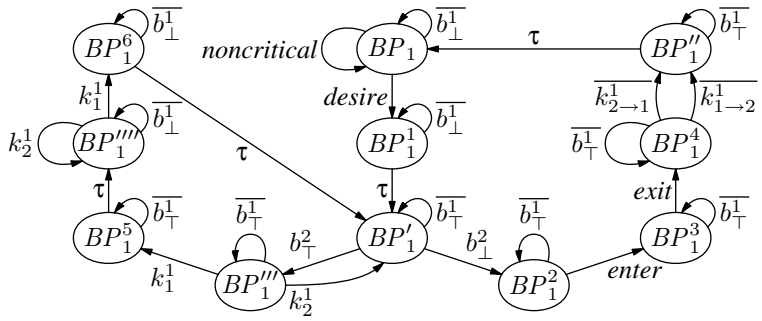
$$
\begin{array}{rcl}
BP_1 & ::= & \overline{b^1_\bot}.BP_1 + noncritical \,.\, BP_1 \\
 & & +desire \,.\, BP^1_1 \\
BP^1_1 & ::= & \overline{b^1_\bot}.BP^1_1 + \tau \,.\, BP'_1 \\
BP'_1 & ::= & \overline{b^1_\top}.BP'_1 \\
 & & +b^2_\bot \,.\, BP^2_1 + b^2_\top \,.\, BP'''_1 \\
BP^2_1 & ::= & \overline{b^1_\bot}.BP^2_1 + enter \,.\, BP^3_1 \\
BP^3_1 & ::= & \overline{b^1_\top}.BP^3_1 + exit \,.\, BP^4_1
\end{array}
\qquad
\begin{array}{rcl}
BP^4_1 & ::= & \overline{b^1_\top}.BP^4_1 \\
 & & +\overline{k^1_{2\to1}} \,.\, BP''_1 + \overline{k^1_{1\to2}} \,.\, BP''_1 \\
BP''_1 & ::= & \overline{b^1_\top}.BP''_1 + \tau \,.\, BP_1 \\
BP'''_1 & ::= & \overline{b^1_\top}.BP'''_1 + k^1_2 \,.\, BP'_1 + k^1_1 \,.\, BP^5_1 \\
BP^5_1 & ::= & \overline{b^1_\top}.BP^5_1 + \tau \,.\, BP''''_1 \\
BP''''_1 & ::= & \overline{b^1_\bot}.BP''''_1 + k^1_2 \,.\, BP''''_1 + k^1_1 \,.\, BP^6_1 \\
BP^6_1 & ::= & \overline{b^1_\bot}.BP^6_1 + \tau \,.\, BP'_1
\end{array}
$$

BP_1 muistuttaa kovasti P_1:tä, koska B_1 pystyy osallistumaan kaikkiin P_1:n "haluamiin" tahdistuksiin. Suurin ero on $\overline{b^1_\bot}$- tai $\overline{b^1_\top}$-silmukka jokaisessa tilassa.

Marko Mäkelä

Esimerkki: Dekkerin algoritmi (6/6)

Agenttia BP_1 vastaavaa siirtymäjärjestelmää on hieman helpompi lukea:



$K|BP_1$ käsittää jo 22 tilaa ja 92 siirtymää, $BP_1|BP_2$ 105 ja 289 ja $Dekker$ 193 ja 428. Voisikohan agentteja yksinkertaistaa poistamalla τ-tapahtumat (ja kätkemällä sitä ennen mielenkiinnottomat tapahtumat, esimerkiksi $noncritical \mapsto \tau$)?

Marko Mäkelä

Siirtymäjärjestelmien ekvivalenssit

Ekvivalenssin eli samankaltaisuuden käsite mahdollistaa agenttien vertaamisen toisiinsa ja kutistamisen. Ekvivalensseja on määritelty lukuisia erilaisia. Esittelemme tässä kaksi.

Jälkiekvivalenssi *(Trace Equivalence)* Jos agenttien P ja Q äärelliset suoritusjäljet eli näkyvät tapahtumaketjut ovat samat, ne ovat jälkiekvivalentit, $P \approx_{\mathrm{tr}} Q$. *Esimerkkejä:* $\alpha \,.\, \tau \,.\, R \approx_{\mathrm{tr}} \alpha \,.\, R$ *ja* $\alpha \,.\, (\beta \,.\, R + \gamma \,.\, R) \approx_{\mathrm{tr}} \alpha \,.\, \beta \,.\, R + \alpha \,.\, \gamma \,.\, R$. *Olkoon* $S ::= \tau \,.\, S$. *Pillastuma ei erotu lukkiumasta:* $S \approx_{\mathrm{tr}} Nil$.

Hylkäysekvivalenssi *(Failure Equivalence)* Olkoon R agentti, α näkyvä tapahtuma ja Σ joukko näkyviä tapahtumia. Jos $R \xrightarrow{\alpha} R'$ ja $\forall \beta \in \Sigma : \nexists R'' : R' \xrightarrow{\beta} R''$, niin $\langle \alpha, \Sigma \rangle$ on R:n *hylkäys*. Agentit P ja Q ovat hylkäysekvivalentit, $P \approx_{\mathrm{f}} Q$, jos niiden hylkäykset ovat samat ja kaikille $\alpha \neq \tau, P', Q'$ s.e. $P \xrightarrow{\tau} \cdots \xrightarrow{\alpha} \xrightarrow{\tau} \cdots P'$ ja $Q \xrightarrow{\tau} \cdots \xrightarrow{\alpha} \xrightarrow{\tau} \cdots Q'$ pätee, että $P' \approx_{\mathrm{f}} Q'$. *Esimerkki:* $\alpha \,.\, (\beta \,.\, R + \gamma \,.\, R) \not\approx_{\mathrm{f}} \alpha \,.\, \beta \,.\, R + \alpha \,.\, \gamma \,.\, R$.

Simulaatio, bisimulaatio ja ekvivalenssin valinta

Olkoon S kaikkien agenttien joukko ja $P \in S, Q \in S$. Agentti P simuloi agenttia Q, jos on olemassa jokin simulaatiorelaatio $R \subseteq S \times S$ siten, että

1. $\langle P, Q \rangle \in R$ ja
2. jos $\langle P', Q' \rangle \in R$ ja $P' \xrightarrow{\alpha} P''$, niin on Q'' siten, että $Q' \xrightarrow{\alpha} Q''$ ja $\langle P'', Q'' \rangle \in R$.

Simulaatio ei ole ekvivalenssi, koska P voi simuloida Q:ta vaikka Q ei simuloisikaan P:tä,* mutta bisimulaatio (P simuloi Q:ta ja Q P:tä) on. TVT-työkalu osaa verrata ja kutistaa siirtymäjärjestelmiä vahvan bisimulaatiorelaation mukaan.

Sovellettava ekvivalenssi kannattaa valita tarkastettavan ominaisuuden mukaan. Esimerkiksi jälkiekvivalenssi on usein liian voimakas, sillä se ei säilytä LTL-kaavojen totuusarvoja. Jos tarkastetaan $\bigcirc$-konnektiivia sisältämättömiä LTL-kaavoja, osittaisjärjestysmenetelmien lähisukulaiset CFFD ja NDFD ovat tehokkaita.

*Ekvivalenssi on refleksiivinen, symmetrinen ja transitiivinen.