Tik-79.179 Rinnakkaiset ja hajautetut digitaaliset järjestelmät

Luennoitsija:	DI Esa Kettunen
	Esa.Kettunen@hut.fi
	451 5456 (t.), 803 52995 (k.), 040-727 4547
Assistentit:	Jukka Honkola
	Jukka.Honkola@hut.fi
Kotitehtävät:	Annikka Aalto
	amaalto@cc.hut.fi

Suoritus:	Tentti
	Pakolliset kotitehtävät
	(joilla voi myös parantaa arvosanaa)
Esitiedot:	Tik-79.144, Tik-79.148 tai vastavat

Tik-79.231 is the same course as this one (except that the teaching language is English).

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Kirjallisuus: *Johan Lilius, Keijo Heljanko, Esa Kettunen*,
Opetusmonisteet (kalvot),
Mikko Tiusanen,
Petri-verkkojen perusteet
Kimmo Varpaaniemi,
PROD Reference Manual
David Walker,
Introduction to a Calculus of
Communicating Systems,

Oheismateriaali: Tadao Murata,
Petri Nets: Properties, Analysis and Applications
Robin Milner,
Communication and Concurrency

Luennot ovat saatavilla kurssin kotisivulta.

(Materiaalin saatavuudesta ilmoitetaan myöhemmin.)

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Lisäinfoa

- WWW: kurssin kotisivu `http://www.tcs.hut.fi/Teaching/Tik-79.179/`
  - Kurssiesite
  - Aikataulu
  - Harjoitustehtävät
  - Kotitehtävät
  - Linkkejä
- Otaxin uutisryhmä: opinnot.tik.rhj
  - pikaiset ilmoitusasiat

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Kurssin tavoite

Antaa valmiuksia rinnakkaisten ja hajautettujen järjestelmien spesifiointiin, verifiointiin sekä validointiin.

Opettaa:

- Petri-verkkojen, temporaalilogiikan, ja prosessialgebran perusteet, sekä
- järjestelmien mallintamista ja analysointia.

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Muut aiheeseen liittyvät opintojaksot

- Tik-79.186 Reaktiiviset järjestelmät
- Tik-79.190 Rinnakkaisohjelmistojen testaus
- Tik-79.193 Rinnakkaisohjelmistojen määrittelymenetelmät
- Tik-76.164 Sulautetut järjestelmät
- Tik-76.166 Rinnakkaisohjelmointi
- Tik-79.146 Logiikka tietotekniikassa: Erityiskysymyksiä I
- (Tik-79.185 Verifionti)

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Rinnakkainen ja hajautettu järjestelmä

- Rinnakkaisuus: tapahtumien samanaikaisuus (concurrency).
  - Tällä kurssilla ei kuitenkaan käsitellä tieteelliseen laskentaan kehitettyjä rinnakkaisprosessoreita, ja niihin liittyvää problematiikkaa.
- Hajautus: järjestelmää ei kontroloida keskitetysti.

  Hajautettu järjestelmä on järjestelmä jossa sellaisen koneen hajoaminen, jonka olemassaolosta sinulla ei ollut aavistustakaan, tekee oman koneesi käyttäkelvottomaksi. (L. Lamport)
- Asynkronisuus: eri osajärjestelmiä ei ole tahdistettu keskenään, vaan ne voivat edetä eri nopeuksin.
- Reaktiivisuus: Järjestelmä on jatkuvasti toiminnassa ja sen toimintaa säätelevät ulkoiset ärsykkeet joihin se reagoi.

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Rinnakkainen ja hajautettu järjestelmä

Suunnitteluongelmia:

- korkeasta epädeterministisyydestä johtuva *tilojen suuri määrä*,
- eri osajärjestelmien keskinäiset riippuvuudet vaikeasti hallittavissa ja ymmärrettävissä,

$\Rightarrow$ tärkeiden haluttujen ominaisuuksien toteaminen vaikeaa.

Minkä takia rinnakkaisia ja hajautettuja järjestelmiä?

- Tehonlisäys
- Redundanssi
- Modulaarisuus
- Laitteiden maantieteellinen erotus

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Ominaisuuksia

- Turvallisuusominaisuuksia:
  "Mitään pahaa ei tapahdu"
  - lukkiutumattomuus (deadlock-freedom)
  - toistensa poissulkevuus (mutual exclusion)
  - tietoturvallisuus,
  - yl. järjestelmä ei koskaan mene tietyyn tilaan
- Elävyysominaisuuksia:
  "Jotain hyvää tapahtuu joskus"
  - reiluus
  - lähetetty viesti menee perille
  - jokainen palvelupyyntä suoritetaan
- Muita:
  - P tapahtuu äärettömän usein
  - P:n jälkeen Q tapahtuu n:n askeleen/sekunnin jälkeen
  - joka tilassa pätee C

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Menetelmiä

Kurssilla opetettavat menetelmät:

- Petri-verkot
- Temporaalilogiikka
- PROD
- Prosessialgebra
- Ekvivalenssit

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Menetelmien käyttökohteita

- Tietoliikenne
  - Protokollien verifiointi
  - Suorituskykyanalyysi
  - Protokollien testaus
- Turvallisuuskriittiset järjestelmät
  - Rautatieylikäytävien ohjauslogiikka
  - Lentokoneen ohjausjärjestelmien logiikka
  - Lennonjohtojärjestelmät
- Hardissuunnittelu
  - Mikroprosessorit
- Reaktiiviset järjestelmät yleensä

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

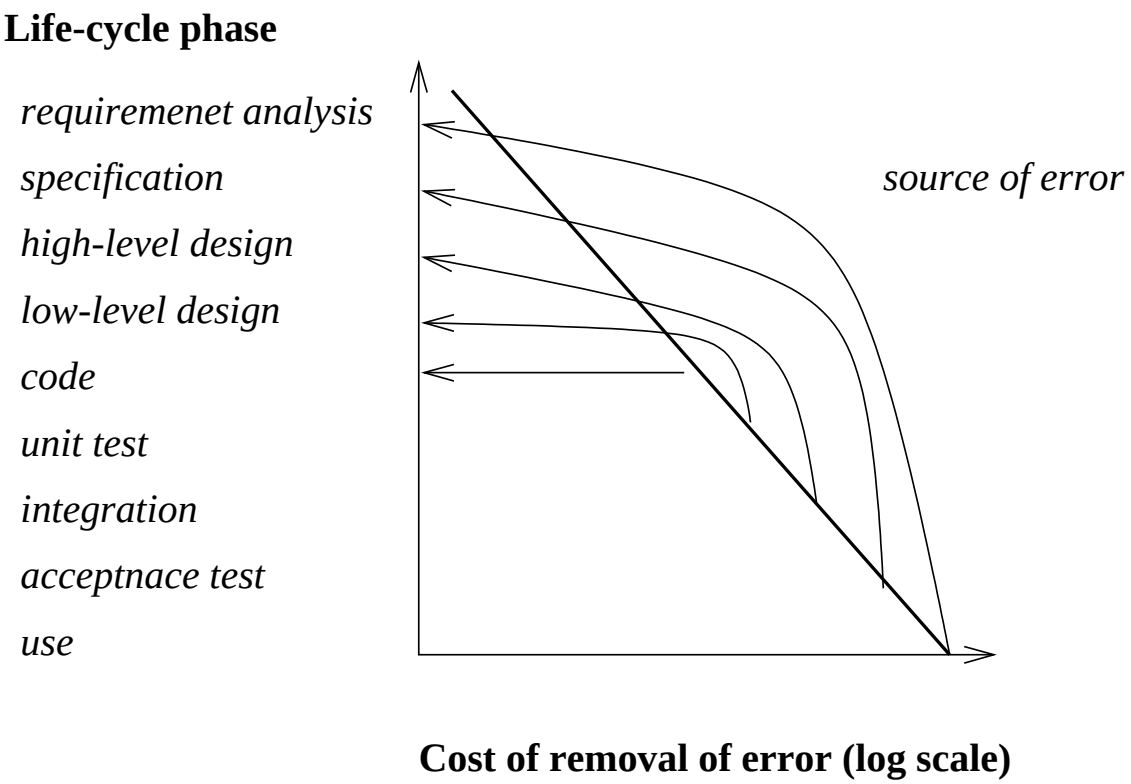
Ohjelmistokriisi

- Miten suunnitella ja toteuttaa ohjelmisto siten, että:
  - se toimitetaan ajallaan,
  - sen kustannukset pysyvät ennalta määrätyissä rajoissa,
  - toimitusajankohtana siinä ei ole enää virheitä,
  - asiakas saa mitä alkujaan oli pyytänyt,
  - ohjelmisto on helposti laajennettavissa,
  - ...
- Ohjelmien tekeminen on pitkälle käsityötä.
- Software Engineering.
- Ns. formaalit kuvausmenetelmät (Formal Methods) pyrkivät tarjoamaan ratkaisuja osaan ym. ongelmista.

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Ohjelmistokriisi

- Ohjelmien koko kasvaa -> hinta kasvaa -> virheet tulevat kalliimiksi:
  - AT&T:n vaihteet menivät solmuun: puolet USA:sta puhelinmotissa
  - Ariane 5 räjähti ohjelmistovirheen takia
  - Pentium FPDIV bugi



©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Mitä on spesifiointi?

- Spesifiointi on järjestelmän toimintojen oletuksien ja vaatimusten kuvausta. Kuvauksen tulisi olla:
  - yksikäsitteinen
  - ei välttämättä täydellinen
- Spesifiointi on sallittujen laskentojen kuvaus
  - laskenta ymmärretään mahdollisimman laajasti
- Spesifikaatiota voidan pitää asiakkaan ja toimittajan välisenä **SOPIMUKSENA** siitä mitä ohjelma tekee.
- Tyypillisesti kun puhutaan spesifioinnista tarkoitetaan spesifiointia jonkin formaalin menetelmän avulla.

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Spesifioinnin etuja

- Oikeellisuus: Automaattiset työkalut tarkastavat spesifikaation.
  - Validointi
  - Verifiointi
- Täydellisyys: Spesifikaatio toimii muistilistana.
- Konsistenssi: Spesifikaation avulla voidaan varhaisessa vaiheessa havaita ristiriitaisia tai kohtuuttomia vaatimuksia.
- Reusability: Software Factory.
- Formaali kuvausmenetelmä, spesifiointimenetelmä = spesifiointi, validointi ja verifiointi.
- SA, ER, eri OO menetelmät, UML etc. . . , eivät ole tässä mielessä spesifiointimenetelmiä.

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Spesifiointi osana SE-prosessia

- Requirements analysis
  - spesifiointimentelmän avulla voidaan selventää asiakkaan vaatimuksia
- Järjestelmän suunnittelu
  - monet menetelmät tukevat dekompositiota ja tarkennusta ("refinement")
- Järjestelmän verifiointi
  - verifiointi on mahdotonta ilman formaaleja menetelmiä
- Järjestelmän validointi
  - spesifikaatiosta voidaan usein suoraan johtaa joukko testejä joiden avulla järjestelmä voidaan validoida
- Järjestelmän dokumentointi
  - spesifikaatio on järjestelmän kuvaus
- Järjestelmän analysointi ja evaluointi
  - useat spesifikaatio menetelmien sovelluksista koostuvat jo olemassaolevan järjestelmän mallintamisesta ja verifioinnista: reverse-engineering.

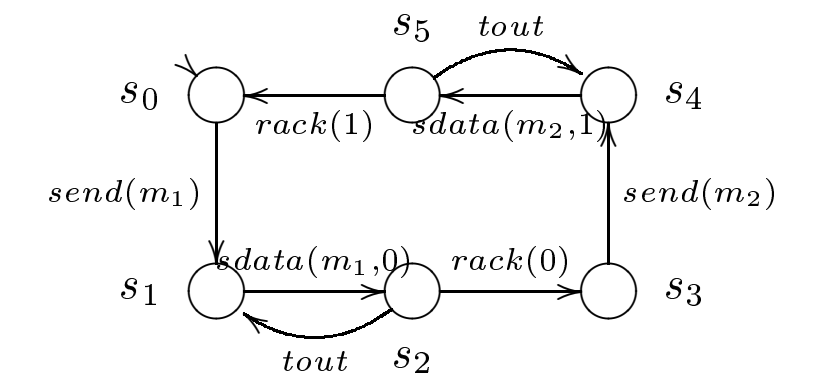
©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

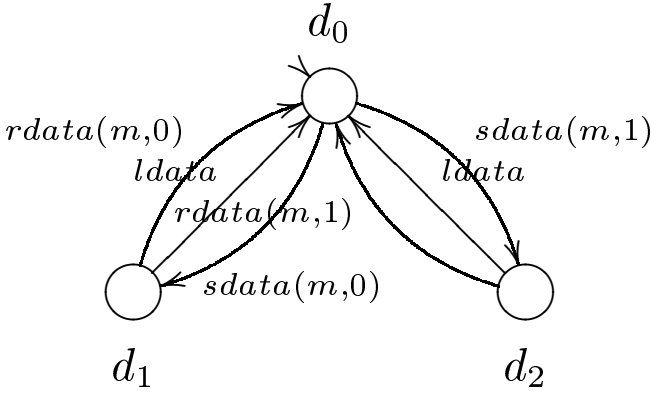
Esimerkki: Alternating-bit protokolla

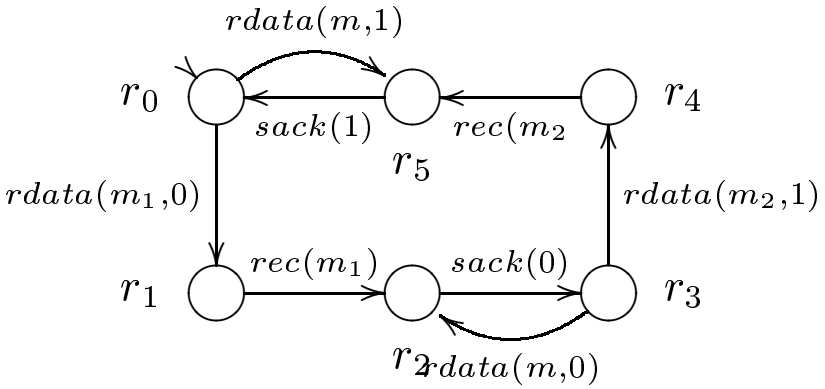
- AB protokolla on yksinkertainen tietoliikenneprotokolla,
- Viestit lähetetään biteillä 0 ja 1 "taggattuna".
  - Lähettäjäprosessi: Saatuaan toimeksiannon, viesti lähetetään yhdessä bitin b kanssa.
    - Jos viesti on hävinnyt saa prosessi time-outin ja viesti lähetetään uudestaan bitin b kanssa.
    - Jos prosessi vastaanottaa bitin b kättelynä se lähettää uuden viestin bitin $\hat{b} = 1 - b$ kanssa.
  - Vastaanottajaprosessi: prosessi vastaanottaa viestin yhdessä bitin b kanssa. Bitti b lähetetään kättelynä takaisin.
    - Jos prosessi vastaanottaa väärän bitin se unohdetaan.
    - Jos prosessi vastaanottaa uuden viestin ja bitin $\hat{b}$ tämä kuitataan.
  - Protokolla siis lähettää ja vastaanottaa viestejä vuorotelleen.
- Väitös: Protokolla pystyy toipumaan jos viesti häviää korkeintaan äärellisen monta kertaa.

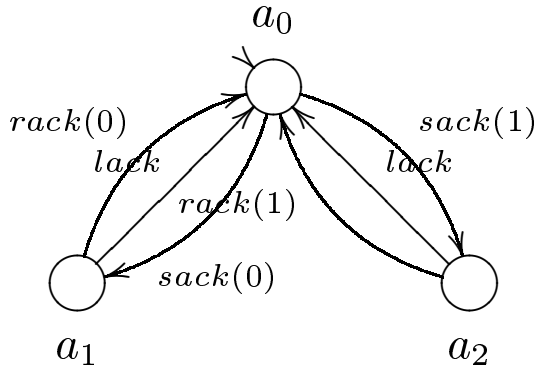
©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Siirtymäkaaviot (kommunikoivat automaatit)







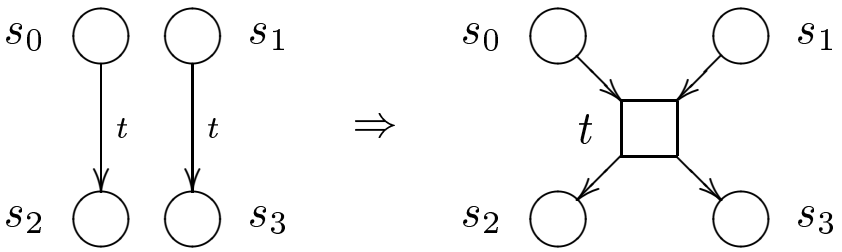


- Jokaisella osajärjestelmällä oma tila! Alkutilaa merkitään $>$.
- Samannimiset tapahtumat tapahtuvat samanaikaisesti (kommunikointi, synkronointi)

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Kuvaus Petri-verkkona

- Siirtymäkaavioiden ongelma on rinnakkaisuuden *granulaarisuus* (eli jyväkoko):
  - automaatit täysin sekventiaalisia,
  - rinnakkaisuutta esiintyy vain automaatien välillä.
- Petri-verkot voidaan nähdä siirtymäkaavioiden yleistyksenä.
- Perusidea: Kuvataan synkronointi eksplisiittisesti.

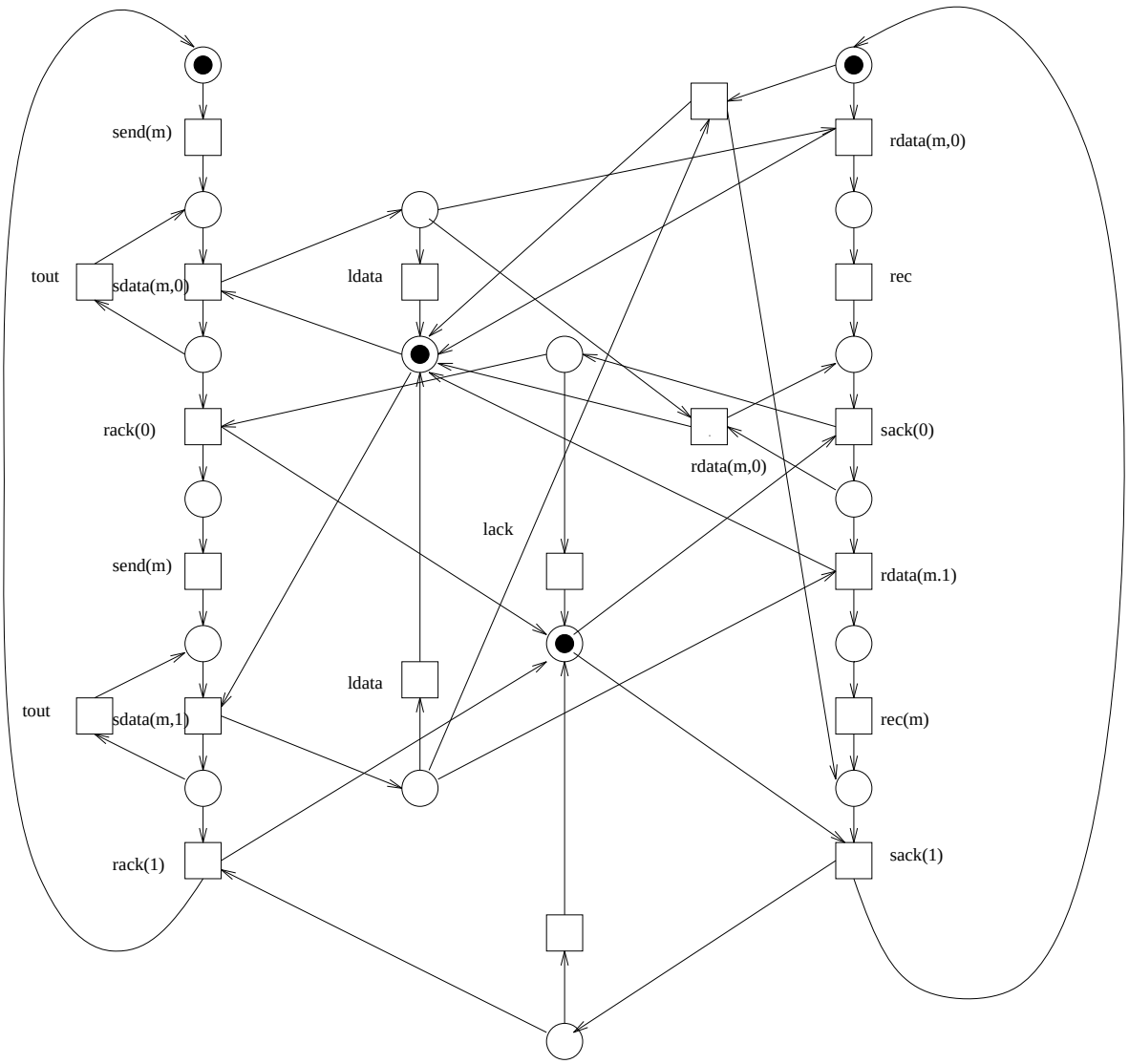


Petri-verkon (perus-)osat:

- Paikka: p ◯
- Transitio: t □
- Suunnattu kaari: ⟶
- Petri verkossa:
  - Järjestelmän globaali tila koostuu useasta lokaalista tilasta.
  - Järjestelmän tila on *hajautettu*
  - Järjestelmän osat toimivat *rinnakkain*

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

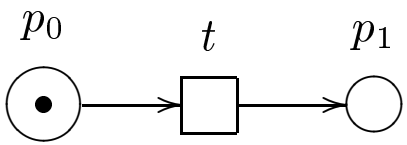
Kuvaus Petri-verkkona



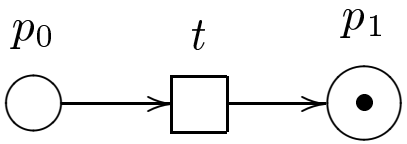
©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Token-peli

- Kuvataan järjestelmän tilaa asettamalla paikkoihin tokeneita (täpliä).
- Järjestelmä vaihtaa tilaa laukaisemalla transitioita.
- Transitio laukeaa jos kaikissa siihen kytketyissä esi-paikoissa on token ja kaikki jälki-paikat ovat tyhjiä (tätä ehtoa tarkennetaan myöhemmin).
- Järjestelmän uusi tila saadaan poistamalla esi-paikoista täplät ja lisäämällä jälki-paikkoihin täplät.
- t vireessä:



- t laukaistaan:



©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000

Token-peli

- Järjestelmän käyttäytymistä voidaan nyt kuvata tilasiirtymäketjuilla.
  - esim: viestin onnistunut lähetys: $send(m) \rightarrow sdata(m,0) \rightarrow rdata(m,0) \rightarrow rec(m) \rightarrow sack(0) \rightarrow rack(0) \rightarrow send(m) \ldots$
- Tilasiirtymäketjuista voidaan muodostaa suunnattu graafi, joka kuvaa kaikkia järjestelmän mahdollisia tilasiirtymiä:
  - Tila-avaruus: järjestelmän kaikki tilat.
  - Saavutettavuusgraafi: graafi jonka solmut ovat tiloja, ja kaaret tilasiirtymiä
- Pelaamalla täpläpeliä käsin on tässä tapauksessa "melko" helppoa varmistua abp-protokollan oikeellisuudesta.

©Johan Lilius 1997, Keijo Heljanko 1998, Esa Kettunen 2000