

OHJ-2600 TILAKONEET

5 op

Antti Valmari

Tampereen teknillinen yliopisto
Ohjelmistotekniikan laitos
PL 553, 33101 TAMPERE

huone TF 209
puh. 3115 4321

sähköposti Antti.Valmari@tut.fi
seittisivu <http://www.cs.tut.fi/~ava/>
5.12.2007

1 JOHDANTO

Kurssin aihe

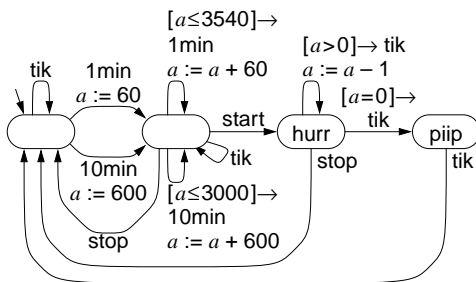
- kurssi käsittelee yhdestä tai useammasta, yhteistyössä olevasta tilakoneesta muodostettujen järjestelmien käyttäytymistä
 - tilakone on laajalti pätevä sekä rinnakkaisjärjestelmän perusosien että — kuten kurssilla havaitaan — osajärjestelmien käyttäytymisen malli
- ⇒ asiat pätevät yleisesti *rinnakkaisjärjestelmiin*
- yhteistyötä tekevästä omatoimisista osista koostuviin järjestelmiin
- kurssilla
 - ei keskitytä mihinkään tiettyyn suunnittelumenetelmään, muuhun sovelluskohteeseen tai sellaista varten suunniteltuun formalismiin, vaan
 - pyritään ymmärtämään tilakoneisiin ja rinnakkaisjärjestelmiin liittyviä ilmiöitä yleisesti
 - viitataan sovellusmahdollisuuksiin

5.12.2007

OHJ-2600 Tilakoneet 2

Tilakone

- esittää järjestelmää, jonka toiminta koostuu tiloissa tapahtuvan odotuksen ja tilasiirtymissä tapahtuvan laskemisen ja viestinnän vuorottelusta



- käytetään joka puolella tietojenkäsittelyä
 - abstrakti tietorakenne
 - olio-ohjelmoinnin olio
 - tietokanta
 - tietoliikenneprotokollan prosessi
 - reaktiivinen järjestelmä, esim. pankkiautomaatti
 - ...

Sisällöstä

- tutustutaan tilakoneen käsitteeseen ja toimintaan
- havaitaan, että muuttujallinen tilakone voidaan palauttaa (isoksi) muuttujattomaksi tilakoneeksi
- tarkastellaan järjestelmien kokoamista tilakoneista

5.12.2007

OHJ-2600 Tilakoneet 3

⇒ ns. *synkroninen vuorovaikutus* paljastuu peruskäsitteeksi, jonka avulla jokseenkin kaikki käytännössä esiintyvät vuorovaikutusmekanismit voi mallintaa

- edelleen havaitaan, että
 - järjestelmän tai sen osan käyttäytyminen voidaan tulkita muuttujattomaksi tilakoneeksi
 - muuttuja voidaan tulkita tilakoneeksi, johon muuttujan käyttäjä on kytketty rinnan

⇒ rinnakkaisjärjestelmän rakennetta voi (ainakin periaatteessa) muokata ja käyttäytymistä tutkia automaattisilla työkaluilla

& sangen yksinkertainen malli riittää teoreettisiin tarkasteluihin

- rinnakkaisjärjestelmissä on osien välisiä vuorovaikutustapahtumia, joihin järjestelmän ympäristö ei osallistu
 - epäolennaisia järjestelmän tuottaman palvelun kannalta
 - ⇒ halutaan abstrahoida pois
 - välillisiä vaikutuksia esimerkiksi lukkiutumiseksi

⇒ erilaisia abstraktioita

- seuraava aihe: *jälkisemantiikka* ja sen tuomat lisämahdollisuudet muokata järjestelmää ja tutkia käyttäytymistä
 - yksinkertaisin abstraktio

- monimutkaisempi semantiikka
 - lukkiutumien huomioonotto
 - pillastumien huomioonotto

- vaihtoehtoisia abstrakteja semantiikkoja

- matkan varrella ja lopuksi teoriaa sovelletaan muutamisiin esimerkkijärjestelmiin

Kurssin historiasta ja motivaatiosta

- on tavallista, että yliopistossa on tutkimukseen kytkeytyvä, pitkälle edistynyt, osallistujamäärältään pieni kurssi
- ko. kurssin myötä hankittavat yleisemmät valmiudet voivat olla tärkeitä niillekin, joita aihe ei itsessään kiinnosta
- tämä kurssi perustuu aiempaan rinnakkaisuuden teorian kurssiin, mutta nyt yritän muuttaa sen vähän kerrassaan tilakoneiden teorian kurssiksi
 - jotkin teorian yksityiskohdat ovat kesken
- kurssilla käytetään tietojenkäsittelyteorian tuloksia tehokkaasti hyväksi
 - ⇒ siitä saattaa saada vinkkejä muunkinlaisten tutkimusongelmien ratkaisemiseen
- kurssilla korostetaan laskutaidon kehittämistä
 - “vierihoitoa”
 - ehkä ei tenttiä
- kurssi on erityisen hyödyllinen kahdelle opiskelijaryhmälle:
 - niille, joiden tarvitsee ymmärtää tilakoneita pintaa syvemmältä
 - niille, jotka ovat suuntautumassa tutkijaksi tai jatko-opiskelijaksi
- kurssilla ei ehditä edetä niin pitkälle, että se antaisi riittävän pohjan alan tutkimiselle
 - tuskin kaikki osallistujat edes haluaisivat niin paljon teoriaa
 - hyvän alun se antaa

2.1 Tilakoneen rakenne

Tilakoneille ei ole vakiintunutta formalismia

- muuttujien ja niillä tehtävän laskennan formalisointi olisi oikeastaan pienen ohjelmointikielen määrittely
 - ei tarkoituksenmukaista tällä kurssilla
- ⇒ jätämme formalisoimatta
 - esitämme esimerkeillä
 - teoriaa varten korvaamme abstraktilla formalismilla
- muissakin yksityiskohtissa esiintyy vaihtelua
- usein on epäolennaista, miten yksityiskohta on valittu, kunhan se on valittu jotenkin

Perusrakenne

- otamme äärellisestä automaatista tuttuja perusosia
 - S = tilat (nyt niitä voi olla äärettömästi!)
 - Σ = aakkosto
 - Δ = tilasiirtymät
 - \hat{S} = alkutilat (nyt niitä voi olla monta!), $\hat{S} \neq \emptyset$
- aakkoston lisäksi käytössä on *näkymättömän tilasiirtymän* symboli τ
 - $\tau \notin \Sigma$
 - hieman eri asia kuin äärellisten automaattien ϵ , siksi eri nimi
 - joukon Σ alkiolla merkitty tilasiirtymä on näkyvä, τ :lla merkitty tilasiirtymä on näkymätön
- tilasiirtymien rakenne muuttuu monimutkaisemmaksi
 - asiaan palataan
 - piirto nuolina säilyy; nuoliin tulee vain lisää tekstiä
- tilasiirtymän suoritusta kutsutaan *tapahtumaksi*

2 TILAKONEEN RAKENNE JA TOIMINTA

Tilakone on rakenne, jossa on

- äärellisen automaatin kaltainen tila- ja tilasiirtymärakenne
- mahdollisesti tilakohtaista ulos näkyvää informaatiota
 - silloin kaarilla ei välttämättä ole nimiä
- mahdollisesti paikallisia muuttujia tms.
- mahdollisesti äärettömästi tiloja

On kehitetty lukuisia tilakonenotaatioita ja -formalismeja

- eroja muuttujien käytössä, ulos näkyvän informaation sijainnissa, vuorovaikutusmekanismeissa, ...
- toiminnan määrittelyssä usein epäselvyyksiä
 - varsinkin yhteistoiminnan

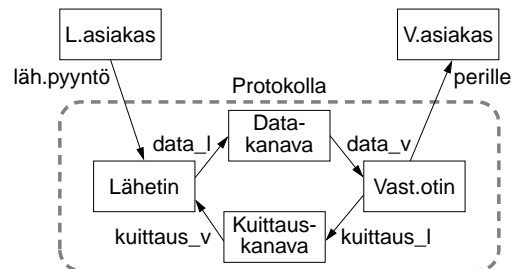
Nyt esitellään eräs mahdollinen tilakoneiden rakenne

- kattaa useimmat usein käytetyt rakennepiirteet
- käyttötymisen perusmalli

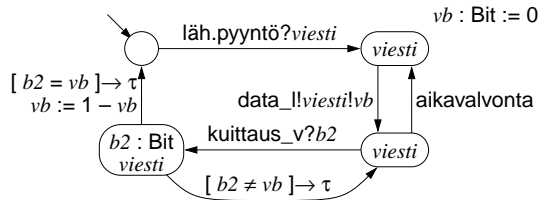
- monen alkutilan mahdollisuudesta on jatkossa hyötyä
 - joissakin tarkasteluissa voi olla jopa nolla alkutilaa
- lopputilojen joukko korvataan toisella mekanismilla, joka antaa monipuolisemmin tietoa tilan ominaisuuksista

Esimerkki: vuorottelevan bitin protokolla

- esimerkin tehtävä on havainnollistaa monia kohta esiteltäviä asioita
- ⇒ pedagogisista syistä pari yksityiskohtaa on valittu toisin kuin muutoin olisi ollut parasta
 - pikkuasioita
- vuorottelevan bitin protokolla on tietoliikenteessä käytettävä mekanismi, jonka tehtävä on toteuttaa luotettava yhteys epäluotettavien kanavien yli

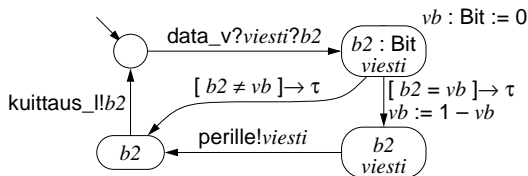


- perustoiminta
 - lähetin saa lähetysoyhtymän asiakkaaltaan
 - lähetin lähettää viestin datakanavan kautta ja jää odottamaan *kuittausta*
 - jos kuittausta ei tule määrääikaan mennessä, lähetin lähettää viestin uudelleen
 - viestissä on lisänä *vuorotteleva bitti* (arvo 0 tai 1), jonka avulla vastaanotin erottaa uudet viestit vanhojen uusinoista
 - vuorotteleva bitti tarvitaan myös kuittauksissa (syy on liian monimutkainen nyt käsiteltäväksi)
- lähetin tilakoneena

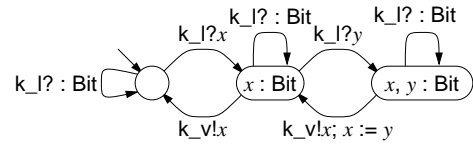


–“_” = lähetysoyhtymä, “_v” = vastaanotto kan.sta

- aikavalvonnan kestoa ei esitetä tässä esimerkissä
 - eräessä esimerkissä käytetään “tik”-tapahtumia
- vastaanotin tilakoneena



- häiriöllinen kuittauskanava tilakoneena, kapasiteetti 2
 - häiriö = kuittaus voi kadota
 - “kuittaus_” on yhennetty “k_” ja vastaavasti “k_v”
 - (kanavan koko sisällön voisi esittää myös yhtenä muuttujana, jolloin yksi tila riittäisi)



Muuttujat

- edellä osoitettiin käteväksi määrittämällä muuttujia kullekin tilalle erikseen
- matemaattisesti on helpompaa määrittää muuttujat koko tilakoneen koskeviksi
 - ⇒ oletamme, että jokaisella muuttujalla on varsinaisten arvojen lisäksi erikoisarvo “_” eli “ei käytössä”
- kuvaan tilojen ulkopuolella määritelty muuttuja on käytössä jokaisessa tilassa
 - esim. lähettimen tai vastaanottimen *vb*
 - ei koskaan saa arvoa _
- kuvaan tilan sisällä mainittu muuttuja on käytössä vain niissä tiloissa, joissa se esiintyy
 - saa arvon _ muissa tiloissa ja vain niissä
- esimerkiksi vastaanottimen oikeassa alarunkossa käytössä ovat muuttujat *b2*, *viesti* ja *vb*
- merkitään *Var* = kaikkien muuttujien joukko
 - esim. $Var_{Lähetin} = \{ b2, viesti, vb \}$
 - jatkoa varten merkitään $V = |Var|$

- kun muuttujan ja sen arvon ero on tärkeä, merkitään muuttujaa *v* ja arvoa *v*

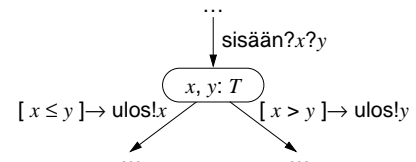
Muuttujien tyypit

- jokaisella muuttujalla *v* on *tyyppi* *Type(v)*
 - tyypit kuten ohjelmointikielissä
 - Type(v)* sisältää vain varsinaiset arvot:
 - $\perp \notin Type(v)$
 - muuttujalla täytyy voida olla varsinainen arvo:
 - $\forall v \in Var: Type(v) \neq \emptyset$
 - tyypin ilmoittaminen
 - voi ilmoittaa tilan sisällä: “: Bit”
 - voi jättää ilmoittamatta, jos se on selvä muutenkin
 - voi jättää ilmoittamatta, jos sitä ei juuri nyt tarvitse tietää: *viesti*
 - jos muuttuja esiintyy useassa tilassa, riittää ilmoittaa tyyppi kerran: *b2*
 - tyypeistä puhumisen helpottamiseksi otamme käyttöön kaikkien (olennaisten) tyyppien joukon *Types*
 - $\forall v \in Var: Type(v) \in Types$
 - tyyppiä tullaan tarvitsemaan myös kaarissa
 - tyypin järkevysvaatimukset ulotetaan kaikille tyypeille:
 - $\forall T \in Types: \perp \notin T \neq \emptyset$
 - myöhemmässä teoreettisessa käsittelyssä ei yleensä tarvita muuttujien nimiä, mutta tullaan tarvitsemaan muuttujien määrä ja kaikkien mahdollisten arvoyhdistelmien joukko
 - sijainti muuttujien luettelossa yksilöi muuttujan
- ⇒ merkitään määrää *V* ja valitaan muuttujille mielivaltaisesti jokin luettelointijärjestys v_1, v_2, \dots, v_V

- muuttujien mahdollisten arvoyhdistelmien joukko on
 - $(Type(v_1) \cup \{\perp\}) \times \dots \times (Type(v_V) \cup \{\perp\})$
 - annetaan sille nimi *VC* (value combinations)
 - ⇒ joukon *VC* jokin alioio kertoo kaikkien muuttujien senhetkiset arvot

Alkuehdot, syöttö ja tulostus tilasiirtymissä

- tilasiirtymä voi testata muuttujien arvoja alkupäässä
 - [*ehto*] ->
 - esim. “[*b2 != vb*] -> ...”
 - esim.



- tilasiirtymässä voi tulostaa muuttujien alkupään arvoista laskettuja arvoja
 - !lauseke
 - esim. “data_!viesti!vb”
 - esim. “tulos!($x^2 + y^2$)”
- tilasiirtymässä voidaan vastaanottaa arvoja loppupään muuttujille
 - ?*x* tai ?*x*: tyyppi
 - esim. “data_v?viesti?b2”
- sallitaan myös arvon vastaanotto ilman talletusta muuttujaan
 - esim. “k_!?”: Bit”

- τ -siirtymässä ei saa esiintyä "?"- eikä "!"-komentoja
 - siirtymä ei ole näkymätön, jos siinä luetaan tai kirjoitetaan arvoja
- tässä kohdassa seurailimme ISON standardoiman Lotos-kielen ratkaisuja

Tilasiirtymän loppuehto

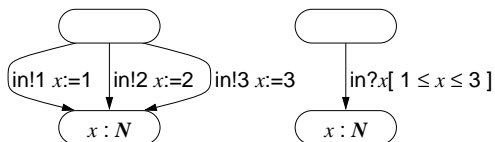
- tilasiirtymään voidaan myös liittää ehto, jota lopussa vallitsevien arvojen on noudatettava
 - esim. "lue ?x [$1 \leq x \leq 5$]"
- jos loppuehdon arvoksi tulisi **F**, tilasiirtymä ei edes käynnisty
 - ⇒ loppuehdot edellyttävät "ennustuskykyä"
 - ⇒ eivät vaikuta kovin helpolta toteuttaa!
- loppuehdoista on kuitenkin jatkossa hyötyä, kun kehitämme tilakoneiden ja niiden vuorovaikutuksen teoriaa yleisempään mutta samalla yksinkertaisempaan suuntaan
 - ⇒ auttavat ymmärtämään asioita
- määrittelykielessä saa olla asioita, joista ei heti tiedetä, miten ne voi toteuttaa
 - tukee "mitä"- ja "miten"-tasojen erottelua
- loppuehdoja ei ole pakko käyttää omista tilakoneissa, jos ei halua!
- loppuehdotkin ovat Lotoksesta

Sijoitukset tilasiirtymissä

- tilasiirtymässä voidaan laskea loppupään muuttujille arvoja alkupään muuttujista
 - *muuttuja := lauseke*
 - esim. " $vb := 1 - vb$ "

- mielivaltaisen arvon voi muodostaa lausekkeella **random** tai **random(alaraja, yläaraja)**
 - jos sama muuttuja esiintyy tilasiirtymän molemmissa päissä eikä siihen lueta eikä sijoiteta tilasiirtymässä, sen arvo säilyy tilasiirtymässä
 - muuten olisi tarpeen kirjoittaa kopioiteja $x := x$
 - jos muuttuja esiintyy vain tilasiirtymän alkupäässä, siihen sijoitetaan tilasiirtymässä automaattisesti \perp
 - jos muuttuja esiintyy vain tilasiirtymän loppupäässä, siihen täytyy lukea tai sijoittaa tilasiirtymässä
 - kaikki lausekkeet paitsi loppuehdot lasketaan alkupään muuttujilla
 - esimerkki
 - sekoile ?x !2·x; y := 1 - x; z := y
 - jos
 - ennen siirtymää $x = 1, y = 2$ ja $z = 3$ ja
 - siirtymässä luetaan 4,
 - niin
 - siirtymässä tulostetaan 2 ja
 - jälkepäin $x = 4, y = 0$ ja $z = 2$
 - samassa tilasiirtymässä saa esiintyä mielivaltaisen monta "?"- ja "!"-komentoa ja sijoitusta
- "?"- ja "!"-komennot, syöttö ja tulostus
- tähän asti olemme ajatelleet, että "?" vastaa syöttöä ja "!" tulostusta
 - ne ovat kuitenkin vain osia yleisemmästä viestintämekanismista, jota voidaan kutsua *atomiseksi vuorovaikutukseksi* tai *atomiseksi neuvotteluksi*

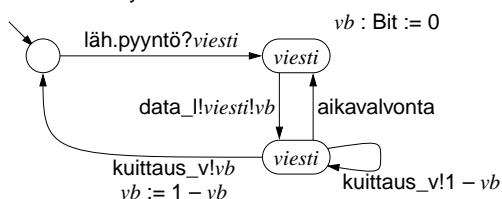
- vertaa



- molemmissa
 - tilasiirtymässä esiintyvä arvo voi olla 1, 2 tai 3
 - tilasiirtymässä esiintyvä arvo tallettuu x:ään
 - ⇒ missä ero?
 - eroa voi ajatella olevan siinä, että toisessa tilakone on lähettäjä, toisessa vastaanottaja
 - koska
 - toiminnassa ei ole eroa
 - erottelu "lähettäjä"–"vastaanottaja" ei ole mielekäs
- "sekoile ?x !2·x"-tyyppisissä tilanteissa ei eroa jatkossa tehdä

Esimerkki

- vaihtamalla "?"-komentoja "!"-komennoiksi aiempaa lähetintä voi yksinkertaistaa



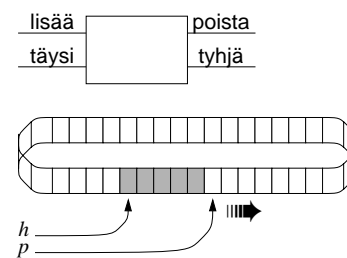
Järjestyssääntöjä

- muuttujalle annettavan arvon täytyy kuulua muuttujan tyyppiin
- samassa tilasiirtymässä samaan muuttujaan saa kohdistua yhteensä enintään yksi lukeminen (eli "?"-komento) tai sijoitus (eli ":=")

Muuttujien alkuarvot

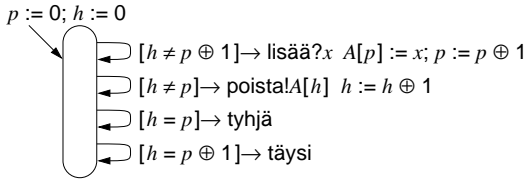
- tilojen ulkopuolella määritellylle muuttujalle voi antaa alkuarvon määritelmän yhteydessä
 - esim. " $vb : \text{Bit} := 0$ "
- mille tahansa muuttujalle voi antaa alkuarvon alkutilanuolen viereen kirjoitetulla sijoituksella
 - kumoaa tilojen ulkopuolella annetun alkuarvon
- tilasiirtymää pitkin tilaan tultaessa muuttuja saa arvon "?"-komennolla, sijoituksesta tai edellisestä tilasta, kuten edellä määriteltiin

Esimerkki: rengaspuskuri



- puskurin koon ilmaisee vakio *koko*, $koko > 0$
 - tosin vain $koko - 1$ voidaan hyödyntää (miksi?)

- muuttujat p ("pää") ja h ("häntä") ovat välillä $0, \dots, koko - 1$
- talletustilana on taulukko $A[0 \dots koko - 1]$
- kaikki laskutoimitukset ovat modulo $koko$
 - $x \oplus 1$ tarkoittaa $(x + 1) \bmod koko$



- vaikka tiloja on vain yksi, malli sallii silti jatkon teorian soveltamisen rengaspuskuriin
 - \Rightarrow rengaspuskuristakin saadaan jatkossa esitettävä kokonaistilakone

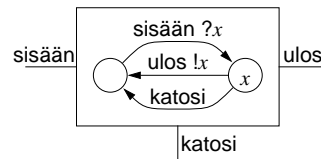
Tätä tilakoneformalismia voisi yleistää aika lailla

- voitaisiin sallia luetun arvon käyttö tulostuksessa
 - esim. "pika_neli $?x !x^2$ "
- alkuarvojen määräämisessä voitaisiin sallia mielivaltaisen predikaatti
 - esim. " $x : N := x > 0$ "
- järjestyssääntöjä voitaisiin lievittää määrittelemällä, että järjettömän tilanteen saa kirjoittaa, mutta tilasiirtymää ei silloin voi suorittaa
 - esim. "oltava_sama $?x ?x ; x := 3$ " voidaan suorittaa vain, jos molemmissa kohdissa tulee 3
- valitsemamme formalismi riittää teorian ja esimerkkien esittämiseen

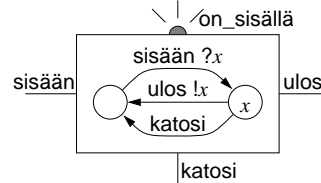
- formalismi korvataan kohta abstraktimmalla
 - \Rightarrow tässä kohti ei kannata paisuttaa formalismia

Tilainformaation näyttämisen ulos tilakoneesta

- tähän asti määritellyistä tilakoneista näkyy ulos vain tilasiirtymiä arvoineen



- joskus on hyödyllistä näyttää tiloista jotain tietoa ulos



- periaatteessa riittäisi määritellä yksi joukko

Tilan_ominaisuuden_arvot

ja yksi funktio, joka tuottaa ko. arvon tilan ja siinä käytössä olevien muuttujien arvojen perusteella

- käytännössä on osoittautunut käteväksi ottaa käyttöön
 - joukko totuusarvoisia ominaisuuksia:

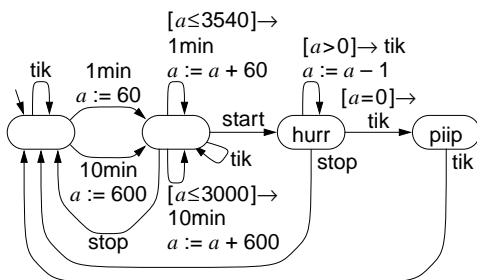
Tilapropositiot Π

- tiedot päälläolosta:

"valuaatio" funktio tai joukko $val \subseteq \Pi \times S \times VC$

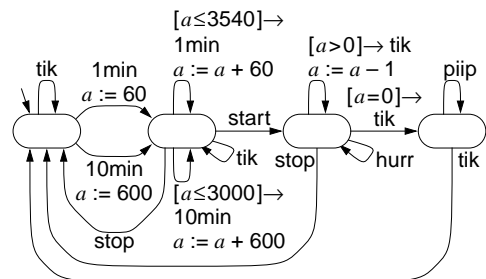
- funktio val kertoo kullekin tilapropositiolle $\pi \in \Pi$ tilan $s \in S$ ja muuttujien arvojen v_1, \dots, v_V perusteella, onko tilapropositio voimassa
 - $val(\pi, s, v_1, \dots, v_V) \in \{F, T\}$
- saman voi sanoa joukolla $(\pi, s, v_1, \dots, v_V) \in / \notin val$
- osoitamme kuvan tilassa päällä olevat tilapropositiot luettelemalla niiden nimet, ja tarvittaessa panemalla eteen "[ehto]"
- äärellisen automaatin lopputilat voidaan esittää tilapropositiolla nimeltä "on_lopputila"
 - (tosin äärellisen automaatin hyväksymän kielen matkiminen ei ole ihan helppoa)
- jos tilainformaatiota ei haluta lisätä formalismiin, niin sitä voi näyttää ulos tilaan palaavilla, sopivasti nimetyillä tilasiirtymillä

Esimerkki: mikroaaltouuni



- näppäimet: 1min, 10min, start, stop

- tilapropositiot
 - hurr: uuni lämmitää
 - piip: äänimerkki on päällä
- ajan kuluminen esitetään "tik"-tilasiirtymillä
 - aikaväli 1 sekunti
- aikaa saa kulua jokaisessa tilassa
 - \Rightarrow tik-kaaria tilasta itseensä
 - (jos tik voisi tapahtua ilman kaarta, ei voitaisi taata, että uunin kello pysyy ajassa)
- hurr ja piip tilasiirtymien niminä



Tilakoneformalismissamme on paljon osia

- ehkä kaikki ei ole edes täsmällisesti määritelty
 - \Rightarrow vaikea käyttää teorian kehittämiseen
 - \Rightarrow käytämme vain esimerkkien esittämiseen, ja kehitämme teoriaa varten yhtäpitävän, yksinkertaisemman (mutta abstraktimman) formalismin

Abstrakti datan käsittely tilasiirtymissä

- tilasiirtymille määriteltiin kuudenlaisia osia:
 - nimi $\in \Sigma \cup \{\tau\}$
 - alku- ja loppuehdot
 - "!"- ja "?"-komennot
 - sijoitukset
- alku- ja loppuehdot, "!"- ja "?"-komennot sekä sijoitukset voi korvata abstraktimmalla mekanismilla, joka kertoo vain niiden yhteisvaikutuksen: *datarelaatio*
- datarelaatio R on tilasiirtymään liitetty 3 - tai $(2V + k)$ -paikkainen[#] relaatio

$$R \subseteq VC \times (T_1 \times \dots \times T_k) \times VC$$

missä

- V on edelleenkin muuttujien määrä
- k on "!"- ja "?"-komentojen määrä tilasiirtymässä
- T_1, \dots, T_k ovat "!"- ja "?"-komentojen lausekkeiden ja vastaanottavien muuttujien tyypit

R :n alkio on muotoa

$$\langle v_1, \dots, v_V \rangle, \langle p_1, \dots, p_k \rangle, \langle v'_1, \dots, v'_V \rangle,$$

joten se koostuu kolmesta osasta, mutta kukin osa on vektori, ja kaikkiaan perusosia on $2V + k$ kappaletta

- $R(\langle v_1, \dots, v_V \rangle, \langle p_1, \dots, p_k \rangle, \langle v'_1, \dots, v'_V \rangle)$ pätee, jos ja vain jos tilakone voi siirtyä ko. tilasiirtymää pitkin tilasta s muuttujien arvojen ollessa v_1, \dots, v_V tilaan s' antaen muuttujille arvot v'_1, \dots, v'_V tilasiirtymään liittyvien data-arvojen ollessa p_1, \dots, p_k
- toisin sanoen, R pätee jos ja vain jos kaikki seuraavat pätevät
 - tässä v_i tarkoittaa i :n muuttujan nimeä, eli sen, jonka arvoksi annetaan v'_i

- muuttujien arvot v_1, \dots, v_V toteuttavat alkuehdon
- muuttujien arvot v'_1, \dots, v'_V toteuttavat loppuehdon
- jokaiselle tilasiirtymän sijoitukselle
 - $\mathbf{v}_i := \text{lauseke}$
 - pätee: *lauseke* laskettuna muuttujien arvoilla v_1, \dots, v_V tuottaa tulokseksi v'_i
- jokaiselle $i \in \{1, \dots, k\}$ pätee
 - jos tilasiirtymän i :s "!"- tai "?"-komento on muotoa "*!lauseke*", niin *lauseke* laskettuna muuttujien arvoilla v_1, \dots, v_V tuottaa tulokseksi p_k
 - jos se on muotoa "? \mathbf{v}_i ", niin $v'_i = p_k$
- jokaiselle $i \in \{1, \dots, V\}$ pätee: jos tilasiirtymässä ei ole sijoitusta $\mathbf{v}_i := \dots$ eikä "?-komentoa \mathbf{v}_i , niin $v'_i = v_i$

Jos abstrahoidemme muuttujien nimet pois tilasiirtymistä, on sama tehtävä alkuarvoille

- määrittelemme jokaiselle alkutilalle $\hat{s} \in \hat{S}$ muuttujien mahdollisten alkuarvohdistelmien joukon $Init(\hat{s})$
 - siis $Init(\hat{s}) \subseteq VC$
- vaadimme, että $\forall \hat{s} \in \hat{S}: Init(\hat{s}) \neq \emptyset$

Datarelaatioita käytettäessä tilakoneen osat ovat siis seuraavan määritelmän mukaiset

- osien $Types, V, VC, \Delta$:n R sekä $Init$:n ja val :in riippuvuus VC :stä kohdalla tärkeintä on ymmärtää, että ne ovat matemaattinen abstraktio, joka esittää saman informaation kuin edellä esitettiin ohjelmointikielimerkityksellä

Määritelmä 2.1 *Tilakone* on monikko

$$(S, Types, V, VC, \Sigma, \Delta, \hat{S}, Init, \Pi, val)$$

missä

- S tilat
- $Types$ tyypit
 - $\forall T \in Types: \perp \notin T \neq \emptyset$
- V muuttujien määrä
- VC muuttujille mahdolliset arvohdistelmät; muotoa $(T_1 \cup \{\perp\}) \times \dots \times (T_V \cup \{\perp\})$ joillekin $T_1 \in Types, \dots, T_V \in Types$
- Σ näkyvien tilasiirtymien aakkosto
 - $\tau \notin \Sigma$
- Δ tilasiirtymät: jokainen Δ :n alkio on muotoa (s, a, R, s') , missä
 - $s \in S$ ja $s' \in S$
 - $a \in \Sigma \cup \{\tau\}$
 - $\exists k: R \subseteq VC \times (T_1 \times \dots \times T_k) \times VC$, missä $T_1, \dots, T_k \in Types$
 - jos $a = \tau$, niin $k = 0$
- $\hat{S} \subseteq S$ alkutilat
- alkuarvot: $\forall \hat{s} \in \hat{S}: Init(\hat{s}) \subseteq VC \wedge Init(\hat{s}) \neq \emptyset$
- Π tilapropositiot
- tilapropositioiden arvot: $val \subseteq \Pi \times S \times VC$ □

Ylimääräisistä suluista val :n käytössä

- tarkkaan ottaen yllä määriteltyä val käytettäisiin $(\pi, s, (v_1, \dots, v_V)) \in val$
 - ylimääräiset sulut ovat kuitenkin vain esitystavasta johtuva, teorian asiasisällön kannalta tarpeeton rasite
- \Rightarrow sallimme niiden pois jättämisen
- vastaa tulkintaa
- $$A \times B \times C = (A \times B) \times C = A \times (B \times C)$$
- \Rightarrow kirjoitamme jatkossa $(\pi, s, v_1, \dots, v_V) \in val$

- matematiikassa on tavallista "väärinkäyttää" merkintöjä selkeyden vuoksi
 - asia säilyisi samana myös jos määriteltäisiin $val: \Pi \times S \times VC \rightarrow \{\mathbf{F}, \mathbf{T}\}$, koska totuusarvon tuottava funktio voidaan yhtäpitävästi tulkita lähtöjoukkonsa osajoukoksi: $val \subseteq \{(\pi, s, v_1, \dots, v_V) \mid val \rightarrow (\pi, s, v_1, \dots, v_V) = \mathbf{T}\}$
 - näinkin sama voidaan sanoa: $val_2: S \times VC \rightarrow 2^\Pi$
 - kaikki nämä ovat vain keinoja sanoa, että val kertoo, onko tilapropositio π voimassa tilassa s muuttujien arvoilla v_1, \dots, v_V vai eikö ole
- Jos $Types = \emptyset$, osa osista sievenee vakioiksi
- \Rightarrow eivät sisällä informaatiota
- on oltava $V = 0$
 - muutoin syntyy ristiriita $T_1 \in Types$ kanssa
 - $VC = \{\langle \rangle\}$
 - joukkotulon identiteettialkio
 - $R \subseteq \{\langle \rangle\} \times \{\langle \rangle\} \times \{\langle \rangle\} = \{\langle \rangle\}$
 - $\Rightarrow R = \emptyset$ tai $R = \{\langle \rangle\}$
 - $R = \emptyset$ on hyödytön tapaus, koska silloin tilasiirtymää ei voi koskaan suorittaa
 - $\Rightarrow R$ tarpeeton Δ :ssa
 - $Init(\hat{s}) \subseteq VC = \{\langle \rangle\}$ ja $Init(\hat{s}) \neq \emptyset$, joten $Init(\hat{s}) = \{\langle \rangle\}$
 - on selkeää määrittellä muuttujaton tilakone niin, että tarpeettomat osat ovat pois

Määritelmä 2.2 *Muuttujaton tilakone* on monikko
 $(S, \Sigma, \Delta, \hat{S}, \Pi, val)$

missä

- S tilat
- Σ näkyvien tilasiirtymien aakkosto
 - $\tau \notin \Sigma$
- $\Delta \subseteq S \times (\Sigma \cup \{\tau\}) \times S$
- $\hat{S} \subseteq S$ alkutilat
 - $\hat{S} \neq \emptyset$
- Π
- $val \subseteq \Pi \times S$ □

(Miksi $V = 0$ ei riitä kutistamaan tällaiseksi?)

Muuttujaton tilakone on kuten äärellinen automaatti, paitsi

- tiloja voi olla äärettömästi
- alkutiloja voi olla monta
- yhden hyväksymistilojen joukon sijasta on kokoelma tilapropositioita
 - kukin tilapropositio voidaan tulkita joukoksi
- ajatus "suoritus päättyy hyväksymistilaan" puuttuu

Datarelaatioiden käytön tässä tilakoneiden teoriassa aloitti Kokkarinen (1997, 1998)

Harjoitustehtäviä

- (a) Esitä luentojen protokollan häiriöllinen datakanava tilakoneena, kapasiteetti kaksi.
 - (b) Esitä esimerkin vastaanotin kolmetilaisena, τ -tilasiirtymättömänä tilakoneena.
- Esitä seuraava algoritmi tilakoneena. Käytä tiloja ainakin **while**- ja **if**-lauseiden esittämiseen.

```

read a; read n
x := 1
while n > 0 do
  if n mod 2 = 1 then x := x · a endif
  n := ⌊n/2⌋; a := a · a
endwhile
write x

```

- Piirrä tilakone, joka esittää puomeilla ja äänimerkillä varustettua rautatien tasoristeystä. Radan varressa on antureita, jotka havaitsevat tulevat ja menevät junat. Oleta, että risteysalueelle pyrkii korkeintaan yksi juna kerrallaan. Aluksi alue on tyhjä ja puomit ovat ylhäällä. Puomi ei laskeudu silmänräpäyksessä. Mallita aika tik-tilasiirtymillä. Puomin laskeutuminen vie yhden tik-ajan, samoin nousu. Käytä tilapropositioita ainakin äänimerkin esittämiseen.
- Määrittele $val_2: S \times VC \rightarrow 2^\Pi$ käyttäen lähtökohtana $val \subseteq \Pi \times S \times VC$ siten, että val ja val_2 sisältävät saman informaation.

2.2 Kokonaistilakone

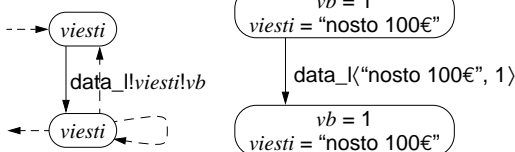
Perusaskel

- tilakoneen toiminnan perusaskel on siirtyminen tilasta s muuttujien arvojen ollessa v_1, \dots, v_V tilaan s' antaen muuttujille arvot v'_1, \dots, v'_V tilasiirtymän nimen ollessa a ja siihen liittyvien data-arvojen ollessa p_1, \dots, p_k
- tämä voidaan esittää tiiviisti näin:

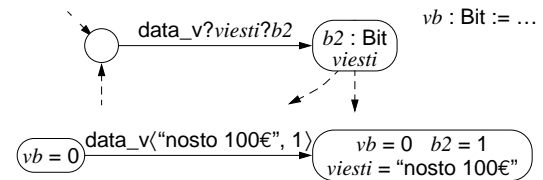
$$s(v_1, \dots, v_V) \xrightarrow{a(p_1, \dots, p_k)} s'(v'_1, \dots, v'_V)$$
- nyt
 - v_i :t ja v'_i :t eivät ole muuttujia, vaan niiden senhetkisiä arvoja
 - p_i :t ovat tilasiirtymän aikana esiintyneitä ja / tai liikkuneita data-arvoja
 - datan liikkeen suunnasta ei välitetä
 - ⇒ "!"- ja "?"-symboleita ei näytetä
 - syötönkin tapauksessa arvo on jokin yksi arvo

Pari esimerkkiä

- dataviestin lähetyksen

 $vb : \text{Bit} := \dots$ 

- dataviestin vastaanotto



Kokonaistila

- tilakoneen kokonaistila = sen hetkinen tila $s \in S$ & muuttujien arvot ko. tilassa $\langle v_1, \dots, v_V \rangle \in VC$
- edellä kuvatun perusaskelen alussa ja lopussa on kokonaistila
- kaikkien mahdollisten kokonaistilojen joukko on $S_K = S \times VC$
- kokonaistilaa pitäisi tämän mukaan merkitä vektorina $\langle s, \langle v_1, \dots, v_V \rangle \rangle$ tai parina $(s, \langle v_1, \dots, v_V \rangle)$, mutta käytämme selkeyden vuoksi merkintää $s(v_1, \dots, v_V)$
 - vrt. tilapropositiot (sivu 22)
 - s on edessä korostamassa tilan ja muuttujien erilaista luonnetta
 - matematiikassa on tavallista korvata merkintöjä selkeämmillä
- on tavallista, että järjestelmä saavuttaa tai edes voi saavuttaa toimintansa aikana vain murto-osan mahdollisista kokonaistiloistaan

- erityisesti rinnakkaisjärjestelmillä on saavuttamattomia kokonaistiloja
 - osan tila riippuu toisen osan tilasta
 - ⇒ kaikki tilayhdistelmät eivät mahdollisia

Kokonaisaakkosto

- näkyvän tilasiirtymän *kokonaisnimi* = tilasiirtymän nimi $a \in \Sigma$ & tilasiirtymän parametrit $p_1 \in U, \dots, p_k \in U$ missä U eli *universumi* on kaikkien mahdollisten data-arvojen joukko, eli $U = \cup_{T \in \text{Types}} T$
 - lisäksi voi olla näkymättömiä eil nimellä τ varustettuja tilasiirtymiä
 - niillä ei voi olla dataparametreja
 - universumiksi ei riitä muuttujien tyyppien unioni, koska lähetettävän data-arvon ei tarvitse olla muuttujien tyypeistä
 - kaikkien mahdollisten näkyvien kokonaisnimien joukko on *kokonaisaakkosto* $\Sigma_K := \Sigma \times U^*$
 - jos $U \neq \emptyset$ ja $\Sigma \neq \emptyset$, niin kokonaisaakkosto on ääretön
 - kuitenkin siitä käytetään usein vain pientä osaa
- ⇒ sallimme muotoa $(\cup_{a \in \Sigma} \{a\} \times PL(a))$ olevien joukkojen käyttämisen kokonaisaakkostoina, missä $PL(a) \subseteq U^*$, jos on (esimerkiksi järjestelmän rakenteen vuoksi) varmaa, että tilasiirtymän nimeen a voi liittyä vain joukon $PL(a)$ mukaisia dataparametrielistoja
- taas käytämme selkeämpää merkintää $a\langle p_1, \dots, p_k \rangle$ merkinnän $\langle a, \langle p_1, \dots, p_k \rangle \rangle$ sijaan

Alkukokonaistilat

- tilakoneen mahdolliset alkutilat ja muuttujien arvot niissä aloitushetkellä muodostavat joukon \hat{S}_K
- vaikka alkutiloja olisi vain yksi, voi \hat{S}_K :ssa olla silti monta alkioita, jos muuttujien alkuarvoja ei ole täysin määrätty
- jos yksi alkutila \hat{s}_K riittää, voidaan käyttää sitä joukon sijasta
- datarelaatioiden tapauksessa saadaan $\hat{S}_K = \{ \hat{s}\langle v_1, \dots, v_V \rangle \mid \hat{s} \in \hat{S} \wedge \langle v_1, \dots, v_V \rangle \in \text{Init}(\hat{s}) \}$
- joka tapauksessa $\hat{S}_K \subseteq S_K$

Tilapropositiot

- tilapropositioiden joukkoa Π ei tarvitse muuttaa
- nyt *val*:n määrittelmä voidaan kirjoittaa yksinkertaisemmin eri tavoin, esimerkiksi $val: \Pi \times S_K \rightarrow \{ \mathbf{F}, \mathbf{T} \}$
 $val \subseteq \Pi \times S_K$
 $val: S_K \rightarrow 2^\Pi$

Saimme kokonaistilakoneen, jossa on seuraavat osat:

- S_K kokonaistilat
- Σ_K kokonaisaakkosto
 - $\tau \notin \Sigma_K$
- $\Delta_K \subseteq S_K \times (\Sigma_K \cup \{ \tau \}) \times S_K$ kokonaistilasiirtymät
- $\hat{S}_K \subseteq S_K$ alkukokonaistilat
- Π tilapropositiot
- $val \subseteq \Pi \times S_K$ tilapropositioiden arvot tiloissa

Kokonaistilasiirtymät

- tilakoneen toiminnan perusaskleet ovat siis kolmikoita (s_K, a_K, s'_K) , missä
 - $s_K = s\langle v_1, \dots, v_V \rangle \in S_K$
 - $a_K = a\langle p_1, \dots, p_k \rangle \in \Sigma_K \cup \{ \tau \}$
 - $s'_K = s'\langle v'_1, \dots, v'_V \rangle \in S_K$
 - kaikkien mahdollisten perusaskelien joukkoa merkitään Δ_K
 - ⇒ $\Delta_K \subseteq S_K \times (\Sigma_K \cup \{ \tau \}) \times S_K$
 - Δ_K ei sisällä kaikkia (s_K, a_K, s'_K) -kolmikoita, vaan vain ne, jotka tilasiirtymiin kirjoitetut alku- ja loppuehdot, "!"- ja "?"-komennot sekä sijoitukset sallivat
- ⇒ Δ_K kattaa myös tilakoneen "ohjelmakoodin" sisällön
- datarelaatioiden tapauksessa saadaan
$$\Delta_K = \{ (s\langle v_1, \dots, v_V \rangle, a\langle p_1, \dots, p_k \rangle, s'\langle v'_1, \dots, v'_V \rangle) \mid \exists R: (s, a, R, s') \in \Delta \wedge R(\langle v_1, \dots, v_V \rangle, \langle p_1, \dots, p_k \rangle, \langle v'_1, \dots, v'_V \rangle) \}$$
 - myös Δ_K :n alkioille voidaan käyttää tuttuja nuolimerkintöjä
 - $s_K \xrightarrow{a_K} s'_K \Leftrightarrow (s_K, a_K, s'_K) \in \Delta_K$
 - $s\langle v_1, \dots, v_V \rangle \xrightarrow{a\langle p_1, \dots, p_k \rangle} s'\langle v'_1, \dots, v'_V \rangle \Leftrightarrow (s\langle v_1, \dots, v_V \rangle, a\langle p_1, \dots, p_k \rangle, s'\langle v'_1, \dots, v'_V \rangle) \in \Delta_K$
 - koska S_K voi sisältää lukuisia saavuttamattomia tiloja, Δ_K voi sisältää lukuisia perusaskelaita, joita ei koskaan suoriteta tai edes voida suorittaa

Näimme tällaisen rakenteen edellisen alaluvun lopussa

⇒ havainto

Tilakoneen käyttäytyminen on tilakone, jossa ei ole muuttujia.

- tämän ansiosta muuttujallille tilakoneille voidaan johtaa tuloksia, algoritmeja yms. käyttämällä muuttujattomien tilakoneiden huomattavasti yksinkertaisempaa formalismia

⇒ tulos on tärkeä!

Määrittelmä 2.3 Tilakoneen $(S, \text{Types}, V, VC, \Sigma, \Delta, \hat{S}, \text{Init}, \Pi, \text{val})$ käyttäytyminen on $(S_K, \Sigma_K, \Delta_K, \hat{S}_K, \Pi_K, \text{val}_K)$, missä

- $S_K = S \times VC$
- $\Sigma_K = (\Sigma \times U^*)$, missä $U = \cup_{T \in \text{Types}} T$
- $\Delta_K = \{ (s\langle v_1, \dots, v_V \rangle, a\langle p_1, \dots, p_k \rangle, s'\langle v'_1, \dots, v'_V \rangle) \mid \exists R: (s, a, R, s') \in \Delta \wedge R(\langle v_1, \dots, v_V \rangle, \langle p_1, \dots, p_k \rangle, \langle v'_1, \dots, v'_V \rangle) \}$
- $\hat{S}_K = \{ \hat{s}\langle v_1, \dots, v_V \rangle \mid \hat{s} \in \hat{S} \wedge \langle v_1, \dots, v_V \rangle \in \text{Init}(\hat{s}) \}$
- $\Pi_K = \Pi$
- $\text{val}_K = \text{val}$ □

- tämä on määrittelmä eikä lause, koska tämä on ensimmäinen formalisointimme sille, mitä tarkoitamme tilakoneen käyttäytymisellä

Käyttäytymisen esittäminen tämänkaltaisella rakenteella on hyvin yleistä rinnakkaisuuden tutkimuksessa

- prosessialgebrat: labelled transition system $(S_K, \Sigma_K, \Delta_K, \hat{S}_K)$
- aikalogiikat: Kripken rakenne $(S_K, \Delta_K, \hat{S}_K, \Pi_K, \text{val}_K)$

- usein lähtökohtana on tilakoneiden sijasta jokin graafinen tai tekstuaalinen järjestelmien kuvauskieli
 - Petriverkot
 - Prossialgebraaliset kielet esim. Lotos, CSP, CCS
 - Promela

Edellä tehty havainto voidaan esittää lauseena

- se, että järjestelmän osa ja järjestelmän käyttäytyminen ovat samanlainen formaali olento on yksi tämän teorian voiman lähde
- todistus on helppo, joten nimetään "pikkulauseeksi"

Pikkulause 2.4 Tilakoneen käyttäytyminen on muuttujaton tilakone. □

Koska tilakone muistuttaa äärellistä automaattia, voimme muodostaa niille samankaltaisia operaatioita

Määritelmä 2.5 Jos $TK = (S, \Sigma, \Delta, \hat{S}, \Pi, val)$ on muuttujaton tilakone, niin

- TK :n saavutettavien tilojen joukko on

$$saav_tilat(TK) = \{ s \in S \mid \exists n; n \geq 0: \exists s_0, s_1, \dots, s_n, a_1, \dots, a_n \in \Sigma \cup \{\tau\}: s_0 \in \hat{S} \wedge s = s_n \wedge \forall i; 1 \leq i \leq n: (s_{i-1}, a_i, s_i) \in \Delta \}$$
- $saav_osa(TK) = (S', \Sigma, \Delta', \hat{S}, \Pi, val')$, missä

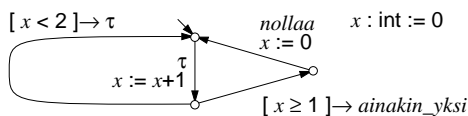
$$S' = saav_tilat(TK)$$

$$\Delta' = \{ (s, a, s') \in \Delta \mid s \in S' \}$$

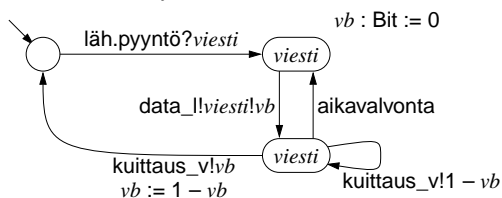
$$val' = val \cap (\Pi \times S')$$
□
- toisin sanoen,
 - tiloista otetaan mukaan vain ne, joihin on polku jostakin alkutilasta
 - poistetaan katoavista tiloista alkavat tilasiirtymät
 - val rajataan jäljelle jääviin tiloihin

Harjoitustehtäviä

1. Piirrä oheisen tilakoneen kokonaistilakoneen alkutilasta saavutettava osa.



2. (a) Piirrä oheisen tilakoneen kokonaistilakoneen alkutilasta saavutettava osa olettaen, että viesti voi saada arvot "A" ja "B".



- (b) Jos viesti voi saada n eri arvoa, niin kuinka monta alkutilasta saavutettavaa tilaa kuvan kokonaistilakoneella olisi?
3. (a) Kuinka monta alkutilasta saavutettavaa tilaa on luennolla käsitellyn rengaspuskurin kokonaistilakoneella, jos yksi lokero voi tallettaa k erisuurta arvoa?
- (b) Kuinka monta eri tilaa rengaspuskurilla on käyttäjän näkökulmasta? Esimerkiksi kaikki tilat, joissa puskurin on tyhjä ovat käyttäjän näkökulmasta samanveroisia, vaikka puskurin lokeroitten sisältö saattaa vaihdella sen mukaan mitä puskurissa on aiemmin ollut.

$saav_osa$ voidaan määrittellä samaan tapaan myös muuttujallisille tilakoneille, mutta ...

- tulos on semanttisesti "epätäydellinen"
 - tilasiirtymän R -relaatio voi olla tyhjä
- ⇒ ei ole taattua, että jokainen jäljelle jäävä tila todella voidaan saavuttaa
- ⇒ tulos ei välttämättä ole se mitä haluttiin
- tulos on ylälikiarvo "oikeasti" saavutettavista tiloista

2.3 Vahva bisimilaarisuus

Suoritukset

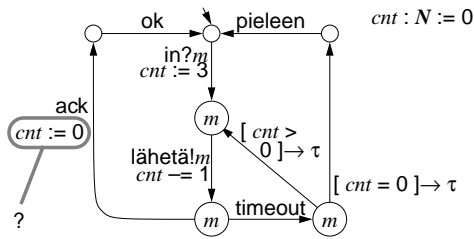
- tilakoneen *suoritus* (*execution*) on äärellinen tai ääretön jono perusaskelia siten, että
 - ensimmäinen askel alkaa jostakin alkutilasta ja sille mahdollisilla muuttujien alkuarvoilla
 - jonossa seuraava askel alkaa siitä tilasta ja niistä muuttujien arvoista, joihin edellinen askel päättyi
- sanoja "suoritus" ja "suorittaa" käytetään myös siten, että ensimmäinen tila ei välttämättä ole alkutila, vaan mainitaan erikseen
- suoritus voidaan merkitä

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} s_n$$
 tai

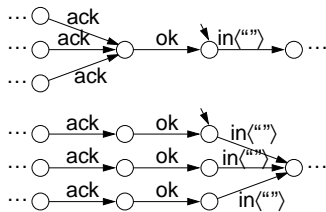
$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots$$
 - $\forall i; i > 0: (s_{i-1}, a_i, s_i) \in \Delta_K$
 - $(s_0 \in \hat{S}_K)$
- tilakoneen kokonaistila on *lukkiutunut*, jos tilakone ei voi suorittaa siitä alkaen ainuttakaan perusaskelta
- *täydellinen suoritus* joko on ääretön tai päättyi lukkiutuneeseen tilaan
- suorituksen jokainen alkuosa on suoritus
 - erityisesti kukin alkukokonaistila on suoritus
- suorituksista puhuttaessa olisi hankala huolehtia siitä, että ne jatketaan aina lukkiutuneeseen tai äärettömyyteen asti
 - ⇒ on helpompaa, että sana "suoritus" tarkoittaa täydellisen suorituksen mahdollisesti vajaata alkuosaa

Milloin kahden tilakoneen voi sanoa käyttäytyvän samoin?

- esimerkki: jos "cnt := 0" poistetaan, ...



– ... niin kokonaistilakone muuttuu



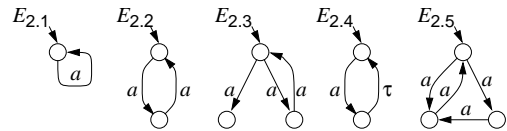
– muuttuuko käytös?

⇒ erilaiset tilakoneet voivat olla käyttäjän kannalta samanveroiset

⇒ on päätettävä, mitä käyttäjä näkee ja mitä ei

- ei tilan koko sisältöä
- tilapropositiot
- "tulevaisuudet" — tämän merkitys täsmentyy kohta

- esimerkkejä



⇒ sopivan "saman käyttäytymisen" valinta riippuu tilanteesta

Aakkostojen ja tilapropositioiden vertailu

- samoin käyttäytymiseltä mielekästä vaatia, että aakkostot ovat samat ja tilapropositioiden joukot ovat samat

$$\Sigma_1 = \Sigma_2 \wedge \Pi_1 = \Pi_2$$

- vaatimus on luonteeltaan syntaktinen tai rajapinnan rakenteeseen liittyvä
 - "tilakoneiden seinässä on samanlaiset liittimet"
 - ei aliohjelmienkaan voi väittää tuottavan saman palvelun, jos parametrien määrässä tai tyypeissä on eroa
- jos $\Sigma_1 \neq \Sigma_2$ tai $\Pi_1 \neq \Pi_2$, niin tilanteesta riippuen voi ajatella
 - tilakoneet käyttäytyvät erilailla
 - tilakoneet on tarkoitettu eri käyttöympäristöön, joten käyttäytymisen samanlaisuutta ei ole mielekästä edes kysyä

Vahva bisimilaarisuus: johdanto

- jos näkymättömät tapahtumat halutaan ottaa täysipainoisesti huomioon, on vahva bisimilaarisuus erittäin usein sopivin valinta "samoin käyttäytymisen" käsitteeksi

- matematiikan pitämiseksi selkeänä esitämme määritelmät kokonaistilakoneiden avulla
 - periaate toimii sellaisenaan muillekin tilakoneille, koska kokonaistilakonehan on vain tilakoneen käyttäytymisen esitystapa

- olkoot verrattavat tilakoneet (tai niiden käyttäytymiset)

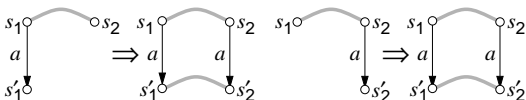
$$TK_1 = (S_1, \Sigma, \Delta_1, \hat{S}_1, \Pi, val_1)$$

$$TK_2 = (S_2, \Sigma, \Delta_2, \hat{S}_2, \Pi, val_2)$$

- aakkostot ovat samat
- emme roikuta mukana alaindeksiä "κ"

Vahva bisimulaatio

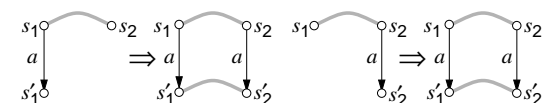
- vahvan bisimilaarisuuden perusidea on, että verrattavien tilakoneiden täytyy voida simuloida toisiaan tietyssä mielessä



- ko. simulointia ei voi määrittellä suoraan
 - syntyisi kehämääritelmä: $s_1 \sim s_2$ jos ... ja $s'_1 \sim s'_2$
- kehä vältetään kokeilemalla kaikki relaatiot $\subseteq S_1 \times S_2$ täyttääkö jokin niistä simulaatiolta vaaditut ehdot
 - jos yksikin täyttää, niin vahva bisimilaarisuus pätee
- ehdot täyttävä relaatio on vahva bisimulaatio
- se on monta-monelle relaatio, joka määrää, mitkä tilat vastaavat toisiaan

Määritelmä 2.6 Olkoot TK_1 ja TK_2 tilakoneita kuten yllä. Relaatio " \sim " $\subseteq S_1 \times S_2$ on vahva bisimulaatio, jos ja vain jos se täyttää seuraavat ehdot kaikille $s_1 \in S_1, s_2 \in S_2, a \in \Sigma \cup \{\tau\}$ ja $\pi \in \Pi$:

- jos $s_1 \sim s_2$, niin $(\pi, s_1) \in val_1 \Leftrightarrow (\pi, s_2) \in val_2$
 - jos $(s_1, a, s'_1) \in \Delta_1$ ja $s_1 \sim s_2$, niin on olemassa $s'_2 \in S_2$ siten, että $(s_2, a, s'_2) \in \Delta_2$ ja $s'_1 \sim s'_2$
 - jos $(s_2, a, s'_2) \in \Delta_2$ ja $s_1 \sim s_2$, niin on olemassa $s'_1 \in S_1$ siten, että $(s_1, a, s'_1) \in \Delta_1$ ja $s'_1 \sim s'_2$. □
 - toisin sanoen
 - vastintilojen tulee antaa tilapropositioille samat arvot
 - jos tilasta alkaen voi tehdä perusaskelen, niin vastintilasta alkaen täytyy voida tehdä perusaskel samalla tapahtuman nimellä siten, että perusaskelten jälkeiset tilat ovat toistensa vastintiloja
 - jälkimmäinen ehto vaaditaan molempiin suuntiin
- ⇒ jos vahvan bisimulaation " \sim " osapuolet vaihdetaan, on lopputulos vahva bisimulaatio
- $$"\sim^{-1}" := \{ (s_1, s_2) \mid (s_2, s_1) \in "\sim" \}$$
- ts. $s_1 \sim^{-1} s_2 \Leftrightarrow s_2 \sim s_1$



- jos peruskaskel $(s_2, a, s'_2) \in \Delta_2$ on vahvan bisimulaation määritelmän mukaisessa suhteessa peruskaskeeseen $(s_1, a, s'_1) \in \Delta_1$, niin sanomme, että se *simuloi* jälkimmäistä
 - sama tietysti "1" ja "2" vaihdettuina

Huomautuksia vahvasta bisimulaatiosta

- vahvan bisimulaation määritelmä ei (suoraan) kerro, mitkä tilat pitää asettaa toistensa vastintiloiksi, vaan se on ikään kuin testi, joka kertoo, onko annettu vastintiloiksi asetettu (eli relaatio "~") kelvollinen
- vahvan bisimulaation määritelmä ei vaadi, että jokaisella tilalla on vastintila



- **emme** ole vielä vaatineet, että alkutilojen tulisi simuloida toisiaan
 - tämän tapainen vaatimus tulee myöhemmin
- ⇒ tyhjä relaatio $\emptyset \subseteq S_1 \times S_2$ on vahva bisimulaatio
- yhdellä tilalla voi olla monta vastintilaa
- peruskaskeleiden täytyy olla simuloitavissa alkupäänsä tilan jokaisesta vastintilasta käsin



- myös kaikki vastakkaiseen suuntaan vaaditut simulaatiot ovat mukana, esimerkiksi
 - tilanteessa $s_{13} \sim s_{23}$ ja $s_{23} \xrightarrow{b} s_{25}$ simulointi on $s_{11} \sim s_{25}$ ja $s_{13} \xrightarrow{b} s_{11}$
 - tilanteessa $s_{15} \sim s_{23}$ ja $s_{23} \xrightarrow{b} s_{25}$ simulointi on $s_{11} \sim s_{25}$ ja $s_{15} \xrightarrow{b} s_{11}$

Esimerkistä voidaan tehdä mielenkiintoisia havaintoja

- silmukka
 - $s_{21} \xrightarrow{a} s_{22} \xrightarrow{b} s_{23} \xrightarrow{b} s_{25} \xrightarrow{a} s_{21}$
 - $s_{27} \xrightarrow{b} s_{26} \xrightarrow{b} s_{21}$
 simuloi kaksi silmukan
 - $s_{11} \xrightarrow{a} s_{12} \xrightarrow{b} s_{13} \xrightarrow{b} s_{11}$
 kierrosta

⇒ on mahdollista, että seuraavien tilakoneiden välillä voi olla vahva bisimulaatio:



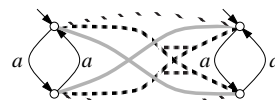
- tilasta ja vastintiloista lähtevien tilasiirtymien määrien ei tarvitse olla samat
 - esim. s_{13} ja s_{26}
 - tilasiirtymien tapahtumanimien joukkojen täytyy kuitenkin olla samat

Tilasiirtymäjonojen simulointia koskeva lause

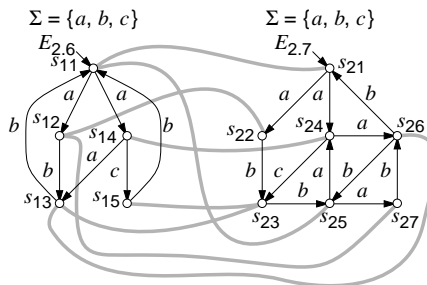
Lause 2.7 Olkoot TK_1 ja TK_2 tilakoneita kuten edellä, olkoon "~" niiden vahva bisimulaatio, ja olkoon $n \geq 0$.

- Jos $s_{10} \sim s_{20}$ ja $s_{10} \xrightarrow{a_1} s_{11} \xrightarrow{a_2} s_{12} \dots \xrightarrow{a_n} s_{1n}$, niin on olemassa s_{21}, \dots, s_{2n} siten, että $s_{11} \sim s_{21}, \dots, s_{1n} \sim s_{2n}$ ja $s_{20} \xrightarrow{a_1} s_{21} \xrightarrow{a_2} s_{22} \dots \xrightarrow{a_n} s_{2n}$.

- samojen tilakoneiden välillä voi olla monta vahvaa bisimulaatiota

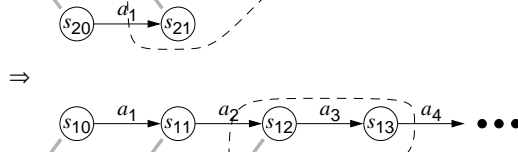
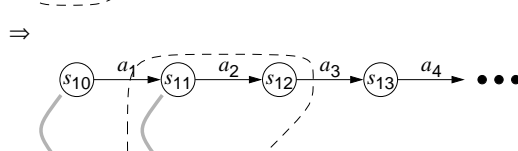
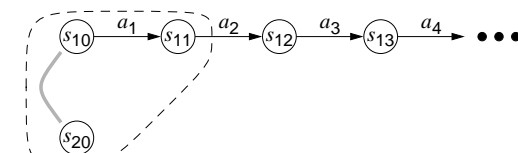


Edellisiä isompi esimerkki vahvasta bisimulaatiosta



- esimerkki: koska $s_{14} \sim s_{24}$, peruskaskelella $s_{14} \xrightarrow{c} s_{15}$ täytyy olla simuloiva peruskaskel
 - ja on: $s_{24} \xrightarrow{c} s_{23}$ ja $s_{15} \sim s_{23}$
- vastaavasti tilanteessa $s_{11} \sim s_{21}$ ja $s_{11} \xrightarrow{a} s_{12}$ simulointi on $s_{12} \sim s_{22}$ ja $s_{21} \xrightarrow{a} s_{22}$
- lisää esimerkkejä
 - tilanteessa $s_{11} \sim s_{21}$ ja $s_{11} \xrightarrow{a} s_{14}$ simulointi on $s_{14} \sim s_{24}$ ja $s_{21} \xrightarrow{a} s_{24}$
 - tilanteessa $s_{11} \sim s_{25}$ ja $s_{11} \xrightarrow{a} s_{12}$ simulointi on $s_{12} \sim s_{27}$ ja $s_{25} \xrightarrow{a} s_{27}$

- Jos $s_{10} \sim s_{20}$ ja $s_{20} \xrightarrow{a_1} s_{21} \xrightarrow{a_2} s_{22} \dots \xrightarrow{a_n} s_{2n}$, niin on olemassa s_{11}, \dots, s_{1n} siten, että $s_{11} \sim s_{21}, \dots, s_{1n} \sim s_{2n}$ ja $s_{10} \xrightarrow{a_1} s_{11} \xrightarrow{a_2} s_{12} \dots \xrightarrow{a_n} s_{1n}$.
- Vastava pätee myös päättymättömille tilasiirtymien jonoille $s_{i0} \xrightarrow{a_1} s_{i1} \xrightarrow{a_2} s_{i2} \xrightarrow{a_3} s_{i3} \dots$ □
- todistuksen perusidea on havainnollistettavissa kuvina:



- jne.

Todistus induktiolla $n:n$ suhteen

- todistamme vain 1. väitteen
 - jälkimmäinen on symmetrinen "1" ja "2" vaihtaen
 - viimeinen menee samalla tavalla
 - pohja: jos $n = 0$, niin lause ei väitä mitään (on triviaalisti voimassa) \cdot .
 - induktioaskel: jos $s_{10} \xrightarrow{a_1} s_{11} \xrightarrow{a_2} \dots \xrightarrow{a_{n+1}} s_{1(n+1)}$, niin induktio-oletuksen nojalla on olemassa tilat s_{21}, \dots, s_{2n} siten, että $s_{11} \sim s_{21}, \dots, s_{1n} \sim s_{2n}$ ja $s_{20} \xrightarrow{a_1} s_{21} \xrightarrow{a_2} \dots \xrightarrow{a_n} s_{2n}$
 - koska $s_{1n} \sim s_{2n}$ ja $s_{1n} \xrightarrow{a_{n+1}} s_{1(n+1)}$, niin " \sim ":n määritelmän nojalla on olemassa $s_{2(n+1)}$ siten, että $s_{1(n+1)} \sim s_{2(n+1)}$ ja $s_{2n} \xrightarrow{a_{n+1}} s_{2(n+1)}$
- \Rightarrow on olemassa $s_{21}, \dots, s_{2(n+1)}$ siten, että $s_{11} \sim s_{21}, \dots, s_{1(n+1)} \sim s_{2(n+1)}$ ja $s_{20} \xrightarrow{a_1} s_{21} \xrightarrow{a_2} \dots \xrightarrow{a_{n+1}} s_{2(n+1)}$. \square

Nyt voidaan määritellä tilakoneiden välinen vahva bisimilaarisuus

Määritelmä 2.8 Olkoot TK_1 ja TK_2 tilakoneita kuten edellä (erityisesti $\Sigma_1 = \Sigma_2$ ja $\Pi_1 = \Pi_2$). Ne ovat *vahvasti bisimilaariset*, merkitään $TK_1 \equiv_{sb} TK_2$, jos ja vain jos on olemassa vahva bimsimulaatio " \sim " $\subseteq S_1 \times S_2$ siten, että:

- $\forall \hat{s}_1 \in \hat{S}_1: \exists \hat{s}_2 \in \hat{S}_2: \hat{s}_1 \sim \hat{s}_2$
- $\forall \hat{s}_2 \in \hat{S}_2: \exists \hat{s}_1 \in \hat{S}_1: \hat{s}_1 \sim \hat{s}_2$. \square
- eli: täytyy olla olemassa ainakin yksi TK_1 :n ja TK_2 :n tilojen joukkojen välinen vahva bisimulaatio siten, että se täyttää *alkutilaehdon*: TK_1 :n jokaiselle alkutilalle on vastintila TK_2 :ssa, ja toisinpäin

- $E_{2.1}$ ja $E_{2.4}$ eivät ole vahvasti bisimilaariset
 - vasta oletus: " \sim " on ...
 - kuten äsken päädyttiin vaatimukseen $1 \sim 8$
 - \Rightarrow ristiriita, koska 1 ei voi simuloida $8 \xrightarrow{\tau} 7$

Vahvan bisimilaarisuuden refleksiivisyys, symmetrisyys ja transitiivisyys

- seuraava ominaisuus tunnetaan nimellä *refleksiivisyys*:

Pikkulause 2.9 Jokainen tilakone on vahvasti bisimilaarinen itsensä kanssa. \square

- todistus: relaatio " \sim ": $s_1 \sim s_2 \Leftrightarrow s_1 = s_2$ on alkutiloja koskevan ehdon täyttävä vahva bisimulaatio \square
- ts. jokainen tilakone voi simuloida itseään, koska kukin tila voi simuloida itseään
- seuraava ominaisuus tunnetaan nimellä *symmetrisyys*:

Pikkulause 2.10 Jos tilakone TK_1 on vahvasti bisimilaarinen tilakoneen TK_2 kanssa, niin TK_2 on vahvasti bisimilaarinen TK_1 :n kanssa. \square

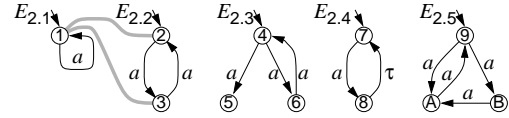
- todistus:
 - jos " \sim " $\subseteq S_1 \times S_2$ on vahva bisimulaatio, niin myös " \sim^{-1} " := $\{(s_1, s_2) \mid (s_2, s_1) \in \sim\} \subseteq S_2 \times S_1$ on vahva bisimulaatio, kuten edellä todettiin
 - jos lisäksi " \sim " täyttää alkutiloja koskevan ehdon, niin myös " \sim^{-1} " täyttää sen (ks. alemmaa)
- " \sim^{-1} " saadaan " \sim ":sta lukemalla "harmaat kaaret" "takaperin": $s_1 \sim^{-1} s_2 \Leftrightarrow s_2 \sim s_1$
- alkutiloja koskeva ehto on
 - $\forall \hat{s}_1 \in \hat{S}_1: \exists \hat{s}_2 \in \hat{S}_2: \hat{s}_1 \sim \hat{s}_2$
 - $\wedge \forall \hat{s}_2 \in \hat{S}_2: \exists \hat{s}_1 \in \hat{S}_1: \hat{s}_1 \sim \hat{s}_2$

- myös aakkostojen ja tilapropositioiden joukkojen täytyy olla samat

Huomautuksia

- jos samojen tilakoneiden välillä on monta vahvaa bisimulaatiota, riittää, että niistä yksikin toteuttaa alkutilaehdon
- kaikissa edeltävissä esimerkeissä, joissa on näytetty alkutiloja, ainakin yksi vahva bisimulaatio asetti alkutilat toistensa vastintiloiksi \Rightarrow ko. tilakoneet ovat vahvasti bisimilaariset

Esimerkkejä



- $E_{2.1}$ ja $E_{2.2}$ ovat vahvasti bisimilaariset
 - kuva näyttää erään vahvan bisimulaation
 - alkutiloja koskeva ehto toteutuu:
 - $\forall \hat{s}_1 \in \{s_1\}: \exists \hat{s}_2 \in \{s_2\}: \hat{s}_1 \sim \hat{s}_2$
 - $\wedge \forall \hat{s}_2 \in \{s_2\}: \exists \hat{s}_1 \in \{s_1\}: \hat{s}_1 \sim \hat{s}_2$
- $E_{2.1}$ ja $E_{2.3}$ eivät ole vahvasti bisimilaariset
 - vasta oletus: " \sim " on niiden välinen vahva bisimulaatio joka täyttää alkutilojen ehdon \Rightarrow täytyy olla $1 \sim 4$
 - $\Rightarrow E_{2.1}$:n täytyy voida simuloida 1:stä tilasiirtymää $4 \xrightarrow{a} 5$
 - ainoa a -siirtymä 1:stä on $1 \xrightarrow{a} 1$
 - \Rightarrow täytyy olla $1 \sim 5$
 - \Rightarrow ristiriita, koska 5 ei voi simuloida siirtymää $1 \xrightarrow{a} 1$

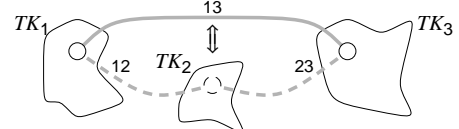
- $\Rightarrow \forall \hat{s}_1 \in \hat{S}_1: \exists \hat{s}_2 \in \hat{S}_2: \hat{s}_2 \sim^{-1} \hat{s}_1$
- $\wedge \forall \hat{s}_2 \in \hat{S}_2: \exists \hat{s}_1 \in \hat{S}_1: \hat{s}_2 \sim^{-1} \hat{s}_1$
- $\Rightarrow \forall \hat{s}_2 \in \hat{S}_2: \exists \hat{s}_1 \in \hat{S}_1: \hat{s}_2 \sim^{-1} \hat{s}_1$
- $\wedge \forall \hat{s}_1 \in \hat{S}_1: \exists \hat{s}_2 \in \hat{S}_2: \hat{s}_2 \sim^{-1} \hat{s}_1$ \square

- vahvan bisimilaarisuuden symmetrisyys seuraa siis siitä, että alkutiloja koskeva ehto on symmetrinen, ja vahvan bisimulaation käänteisrelaatiokin on vahva bisimulaatio, mikä taas seuraa siitä, että vahvan bisimulaation määritelmässä simuloitavuus vaaditaan molempiin suuntiin

- seuraava ominaisuus tunnetaan nimellä *transitiivisyys*:

Pikkulause 2.11 Jos tilakone TK_1 on vahvasti bisimilaarinen tilakoneen TK_2 kanssa ja TK_2 on vahvasti bisimilaarinen tilakoneen TK_3 kanssa, niin TK_1 on vahvasti bisimilaarinen TK_3 :n kanssa. \square

- todistus: osoitamme vaiheittain, että jos " \sim_{12} " $\subseteq S_1 \times S_2$ on alkutilaehdon täyttävä vahva bisimulaatio TK_1 :stä TK_2 :een ja " \sim_{23} " $\subseteq S_2 \times S_3$ on alkutilaehdon täyttävä vahva bisimulaatio TK_2 :stä TK_3 :een, niin " \sim_{13} " := $\{(s_1, s_3) \mid \exists s_2: (s_1, s_2) \in \sim_{12} \wedge (s_2, s_3) \in \sim_{23}\}$ on alkutilaehdon täyttävä vahva bisimulaatio TK_1 :stä TK_3 :een
- ts. $s_1 \sim_{13} s_3 \Leftrightarrow \exists s_2: s_1 \sim_{12} s_2 \wedge s_2 \sim_{23} s_3$



- “ \sim_{13} ” $\subseteq S_1 \times S_3$ suoraan määritelmänsä vuoksi
 - $(s_1, s_2) \in \sim_{12} \Rightarrow s_1 \in S_1$, vastaavasti $s_3 \in S_3$
- “ \sim_{13} ” toteuttaa alkutilaehdon: olkoon $\hat{s}_1 \in \hat{S}_1$
 - koska “ \sim_{12} ” toteuttaa alkutilaehdon, niin $\hat{s}_1 \sim \hat{s}_2$ pätee ainakin yhdelle $\hat{s}_2 \in \hat{S}_2$
 - kyseiselle \hat{s}_2 ja ainakin yhdelle $\hat{s}_3 \in \hat{S}_3$ pätee $\hat{s}_2 \sim \hat{s}_3$, koska “ \sim_{23} ” toteuttaa alkutilaehdon $\Rightarrow \hat{s}_2$ kelpaa “ \sim_{13} ”:n määritelmän s_2 :ksi $\Rightarrow \hat{s}_1 \sim \hat{s}_3$
 - sama päättely toiseen suuntaan
- olkoot $s_1 \in S_1$ ja $s_3 \in S_3$ siten, että $s_1 \sim_{13} s_3$, ja olkoon $\pi \in \Pi$
 - “ \sim_{13} ” lupaa, että $\exists s_2: s_1 \sim_{12} s_2 \sim_{23} s_3$
 - koska “ \sim_{12} ” ja “ \sim_{23} ” ovat vahvoja bisimul., pätee $(\pi, s_1) \in val_1 \Leftrightarrow (\pi, s_2) \in val_2 \Leftrightarrow (\pi, s_3) \in val_3$ siis $(\pi, s_1) \in val_1 \Leftrightarrow (\pi, s_3) \in val_3$
- olkoot $s_1 \in S_1$ ja $s_3 \in S_3$ siten, että $s_1 \sim_{13} s_3$, ja olkoon $(s_1, a, s'_1) \in \Delta_1$
 - nytkin $\exists s_2: s_1 \sim_{12} s_2 \sim_{23} s_3$
 - koska “ \sim_{12} ” on vbs., niin s_2 voi simuloida (s_1, a, s'_1) : $\exists s'_2: (s_2, a, s'_2) \in \Delta_2 \wedge s'_1 \sim_{12} s'_2$
 - koska “ \sim_{23} ” on vbs. niin s_3 voi simuloida (s_2, a, s'_2) : $\exists s'_3: (s_3, a, s'_3) \in \Delta_3 \wedge s'_2 \sim_{23} s'_3$
 - koska $s'_1 \sim_{12} s'_2 \wedge s'_2 \sim_{23} s'_3$, niin $s'_1 \sim_{13} s'_3$ $\Rightarrow \exists s'_3 \in S_3: (s_2, a, s'_2) \in \Delta_2 \wedge s'_1 \sim s'_3$
- viimeinen vahvan bisimulaation ehto osoitetaan kuten edellinen \Rightarrow “ \sim_{13} ” on vahva bisimulaatio \square
- sivun 45 $E_{2.1}$ ja $E_{2.2}$ todettiin vahvasti bisimilaarisiksi ja laskuharjoituksessa todetaan $E_{2.2}$ ja $E_{2.5}$ $\Rightarrow E_{2.1}$ ja $E_{2.5}$ ovat vahvasti bisimilaariset

Ekvivalenssi

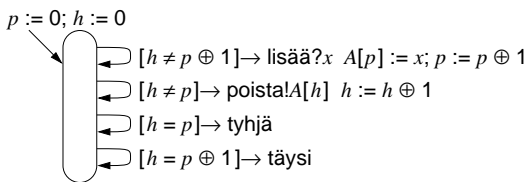
- on helppo hyväksyä, että kun formalisoidaan “samoin käyttäytymisen” tai yleensä minkälaista “samanlaisuuden” käsitettä, siltä on vaadittava refleksiivisyys, transitiivisuus ja symmetrisyys
- joissakin tilanteissa joudutaan lisäksi vaatimaan niin sanottu *kongruenssiominaisuus*
 - tarkoittaa eräänlaista vaihtokelpoisuutta isomman kokonaisuuden osana
- kokemus on osoittanut, että muuta ei tarvitse vaatia \Rightarrow jos ei haluta määritellä, minkälaisen isompien kokonaisuuksien osana esimerkiksi tilakone saa olla, on mielekästä vaatia refleksiivisyys, transitiivisuus ja symmetrisyys, mutta ei enempää \Rightarrow matematiikassa on annettu oma nimitys binäärirelaatiolle, joka on refleksiivinen, transitiivinen ja symmetrinen: *ekvivalenssi*

Pikkulause 2.12 Vahva bisimilaarisuus on ekvivalenssi. \square

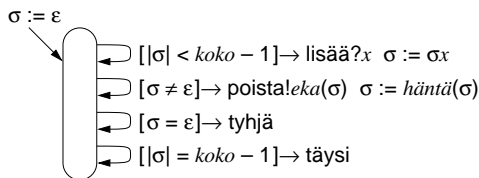
- muita ekvivalensseja:
 - lukujen “=”
 - logiikan “ \Leftrightarrow ”
 - ihmisten “samat vanhemmat”
- esimerkkejä binäärirelaatioista, jotka eivät aivan ole ekvivalensseja:
 - lukujen “ \leq ” ei ole symmetrinen
 - lukujen “ \approx ” ei ole transitiivinen
 - ihmisten “veli tai sisar” ei ole refleksiivinen
 - vektorien “saman suuntainen” ei ole transitiivinen, jos nollavektori tulkitaan samansuuntaiseksi jokaisen vektorin kanssa

Esimerkki: rengaspuskurin vertaaminen jonoon

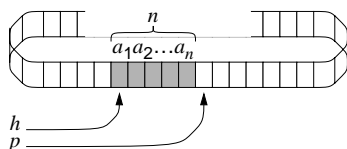
- $koko > 0$
- rengaspuskuri $(x \oplus y)$ tarkoittaa $(x + y) \bmod koko$



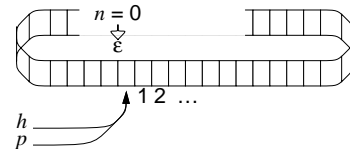
- jono tilakoneena
 - jos $\sigma = a_1 a_2 \dots a_n$, missä $n > 0$, niin $eka(\sigma) = a_1$ ja $häntä(\sigma) = a_2 \dots a_n$



- vahva bisimulaatio: $\langle p, h, A \rangle \sim a_1 a_2 \dots a_n$, jos ja vain jos
 1. $0 \leq p < koko$ ja $0 \leq h < koko$ ja $0 \leq n < koko$
 2. $h \oplus |a_1 a_2 \dots a_n| = p$ eli $h \oplus n = p$ ja
 3. $\forall i; 0 \leq i < n: a_{i+1} = A[h \oplus i]$

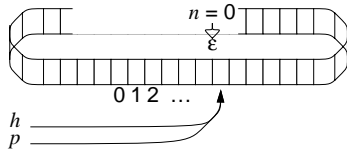


- alkutiloja koskeva ehto (määritelmä 2.8 sivu 44)
 - $p = h = 0, A = ?$ ja $a_1 a_2 \dots a_n = \varepsilon$ eli $n = 0$
 - 1. $0 \leq 0 < koko$ ja $0 \leq 0 < koko$ ja $0 \leq 0 < koko$
 - 2. $0 \oplus 0 = 0$
 - 3. $\forall i; 0 \leq i < 0: \dots \cdot /$ $\Rightarrow \langle 0, 0, A \rangle \sim \varepsilon$, olipa A :n sisältö mikä tahansa \Rightarrow jonon ainoa alkutila simuloi rengaspuskurin jokaista alkutilaa ja päinvastoin $\cdot /$

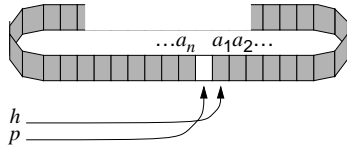


- sitten määritelmän 2.6 (sivu 39) kohdat kullekin tilasiirtymälle
 - olkoon jatkossa koko ajan $s_{rp} = \langle p, h, A \rangle \sim a_1 a_2 \dots a_n = s_{jn}$
- tilapropositioita ei ole \Rightarrow niitä koskeva ehto on triviaalisti voimassa
- tilasiirtymien simuloitumisen todistamiseksi osoitamme ensin, että kunkin tilasiirtymän lähtöehto saa saman totuusarvon tilassa ja sen vastintilassa:
 - $h = p \Leftrightarrow n = 0 \Leftrightarrow a_1 a_2 \dots a_n = \varepsilon$ (koska $h \oplus n = p$ ja h, p ja n on rajattu välille $0 \leq x < koko$)
 - $h = p \oplus 1 \Leftrightarrow n = koko - 1 \Leftrightarrow |a_1 a_2 \dots a_n| = koko - 1$ \Rightarrow jos $s_{rp} \text{—tyhjä} \rightarrow s'_{rp}$, niin $\exists s'_{jn}: s_{jn} \text{—tyhjä} \rightarrow s'_{jn}$
 - “tyhjä” ei muuta muuttujien arvoja kummassakaan tilakoneessa $\Rightarrow s'_{rp} = s_{rp} \sim s_{jn} = s'_{jn} \Rightarrow s'_{rp} \sim s'_{jn}$
 - $\Rightarrow s_{jn} \text{—tyhjä} \rightarrow s'_{jn}$ simuloi $s_{rp} \text{—tyhjä} \rightarrow s'_{rp}$

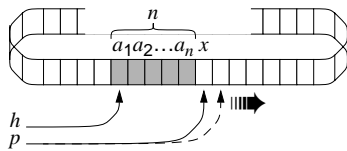
- samoin jos s_{jn} —tyhjä→ s'_{jn} , niin ... s_{rp} —tyhjä→ s'_{rp} simuloi
- ⇒ jono voi simuloida rengaspuskurin "tyhjä"-siirtymiä ja toisinpäin



- samanlainen päättely toimii "täysi"-tilasiirtymille

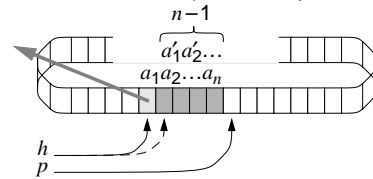


- jos $\langle p, h, A \rangle$ —lisää(x)→ $\langle p', h', A' \rangle$, niin
 - $p' = p \oplus 1$
 - $h' = h$
 - $A'[p] = x$, ja $A'[i] = A[i]$ kun $i \neq p$



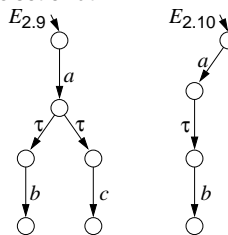
- on jo osoitettu, että $\langle p, h, A \rangle$ —lisää(x)→ takaa, että $a_1 a_2 \dots a_n$ —lisää(x)→ s'_{jn} jollekin s'_{jn}
 - tilasiirtymistä luemme, että $s'_{jn} = a_1 a_2 \dots a_n x$

- nyt riittää osoittaa $\langle p', h', A' \rangle \sim a_1 a_2 \dots a_n x$
 1. $0 \leq p' = p \oplus 1 < koko$ -/.
 2. $0 \leq h' = h < koko$ -/.
 3. vielä puuttuu ehdon
 - $\forall i; 0 \leq i < n : a_{i+1} = A[h \oplus i]$ vastine tilasiirtymien jälkeisille tiloille, eli $(\forall i; 0 \leq i < n : a_{i+1} = A'[h' \oplus i]) \wedge x = A'[h' \oplus n]$
- kun $0 \leq i < n$, niin $h' \oplus i \neq p$, joten $A'[h' \oplus i] = A[h' \oplus i] = A[h \oplus i] = a_{i+1}$
- $h' \oplus n = h \oplus n = p$, joten $A'[h' \oplus n] = A'[p] = x$
- ⇒ jono voi simuloida rengaspuskurin "lisää"-siirtymiä
- samaan tapaan voidaan todistaa, että rengaspuskuri voi simuloida jonon "lisää"-siirtymiä, ja että kumpikin voi simuloida toisensa "poista"-siirtymiä

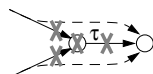


Käyttäytymisen samuudesta, kun näkymättömiä tapahtumia ei haluta ottaa huomioon samoin kuin näkyviä

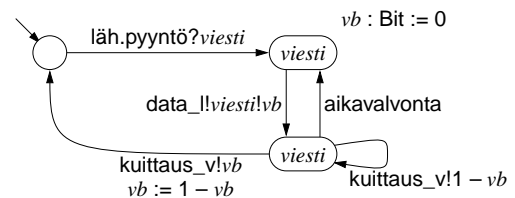
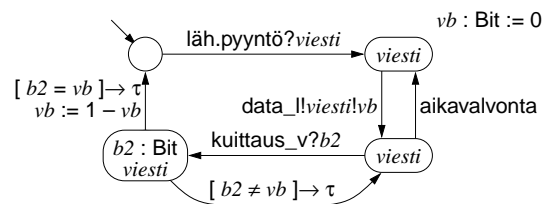
- siinä tilanteessa käyttäytymisen samuudelle löytyy useita olennaisesti erilaisia mielekkäitä määritelmiä
 - esim. jotkut haluavat tulkita seuraavat erilaisiksi, toiset eivät



- joissakin tapauksissa käyttäytymisen samanlaisuus on vaikea määritellä siten, että osien samanlaisuus takaa kokonaisuuden samanlaisuuden
 - kongruenssiominaisuus
- ⇒ aihealueen teoria on laaja ja monimutkainenkin
 - aiheena luvusta 4 alkaen
- jokseenkin kaikille mielekkäille käyttäytymisen samanlaisuuskaasiteille pätee: jos tilasta lähtee vain yksi siirtymä ja se on τ -siirtymä joka vie johonkin muuhun tilaan, niin käyttäytyminen säilyy samana, jos ko. siirtymä ohitetaan ja poistetaan



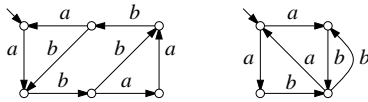
- ⇒ esimerkiksi molemmat lähettimen versiot käyttäytyvät tässä mielessä samoin
 - ehdot $b2 = vb$ ja $b2 \neq vb$ sulkevat toisensa pois
 - ⇒ vasemman alanurkan tilaa vastaavista kokonaistiloista lähtee vain yksi τ -siirtymä kustakin



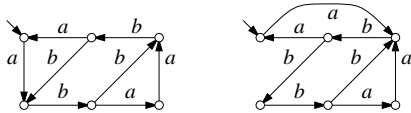
- tätä voi käyttää nyrkkisääntönä, mutta jos ei tunne taustalla olevaa teoriaa, niin ei tiedä, mitä oikeastaan tekee

Harjoitustehtäviä

1. (a) Osoita, että oikeiset tilakoneet ovat vahvasti bisimilaariset.



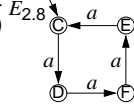
- (b) Ovatko seuraavat tilakoneet vahvasti bisimilaariset?



2. (a) Olkoot $E'_{2,6}$ ja $E'_{2,7}$ ne tilakoneet, jotka saadaan kääntämällä sivun 41 tilakoneiden $E_{2,6}$ ja $E_{2,7}$ kaikki tilasiirtymät takaperin. Ovatko $E_{2,6}$ ja $E_{2,7}$ vahvasti bisimilaariset?

- (b) Jos TK_1 ja TK_2 ovat vahvasti bisimilaariset ja niissä ei ole saavuttamattomia tiloja ja s_1 ja s_2 ovat toistensa vastintiloja, voiko s_1 :een tulevissa tilasiirtymissä esiintyä tapahtumien nimiä, joita s_2 :een tulevissa tilasiirtymissä ei esiinny?

3. (a) Muodosta kaikki mahdolliset vahvat bisimulaatiot tilakoneiden $E_{2,2}$ (sivu 45) ja $E_{2,8}$ välille. Mitkä niistä toteuttavat alkutilaehdon?

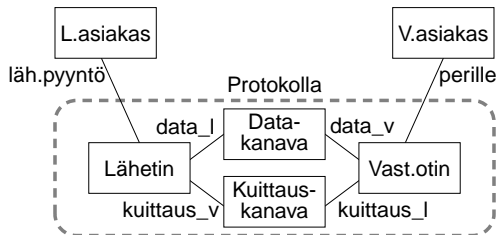


- (a) Muodosta kaikki mahdolliset vahvat bisimulaatiot tilakoneiden $E_{2,2}$ ja $E_{2,5}$ välille.

4. Ovatko tilakoneet $E_{2,9}$ ja $E_{2,10}$ (sivu 54) vahvasti bisimilaariset? Perustelee.

3 TOISIINSA KYTKETYT TILAKONEET

Tässä luvussa kerrotaan, miten tilakoneita voi yhdistää eri operaatioilla



- rinnankytkentä "||"
- kätöntä "hide"
- relationaalinen- eli moniuidelleennimeäminen
- tilapropositioiden uudelleennimeäminen
- (muuta)

Tilakoneiden kytkentä määritellään siten, että tuloksena on tilakone

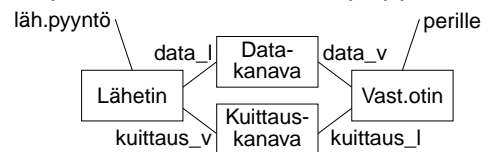
Jokainen operaattori määritellään aluksi muuttujattomille tilakoneille

- muuttujattomien tilakoneiden operaattori toimii määritelmänä sille, miten operaation tulos käyttäytyy
- jokaiseen muuttujallisten tilakoneiden kytkentään liittyy lause: tuloksena syntyvä tilakone käyttäytyy määritelmän mukaisesti
- tilapropositioiden ja muuttujien aineisto uutta \Rightarrow sähläysvaara ...

5. Osoita, että TK ja $saav_osa(TK)$ ovat vahvasti bisimilaariset.
6. (a) Osoita, että sivun 50 rengaspuiskuri voi simuloida jonon "lisää"-siirtymiä.
- (b) Osoita, että jono voi simuloida rengaspuiskurin "poista"-siirtymiä ja toisinpäin.
7. Piirrä tilanne, jossa tilasta lähtee vain yksi siirtymä ja se on τ -siirtymä joka ei vie johonkin muuhun tilaan. Muuttuuko käyttäytyminen ja jos niin miten, jos ko. siirtymä ohitetaan ja poistetaan?

3.1 Rinnankytkentä, ei muuttujia

Tehtävänä on määrittellä muuttujaton tilakone, joka käyttäytyy kuten muuttujattomien tilakoneiden rinnankytkennän tulisi mielestämme käyttäytyä



Olkoot tilakoneet $TK_1 = (S_1, \Sigma_1, \Delta_1, \hat{S}_1, \Pi_1, val_1)$, $TK_2 = (S_2, \Sigma_2, \Delta_2, \hat{S}_2, \Pi_2, val_2)$, ..., $TK_n = (S_n, \Sigma_n, \Delta_n, \hat{S}_n, \Pi_n, val_n)$

- kuvassa esimerkiksi $TK_1 =$ Lähetin, $TK_2 =$ Vast.otin, $TK_3 =$ Datakanava, $TK_4 =$ Kuittauskanava
- kertaus määritelmästä 2.2 (sivu 24)
 - $\tau \notin \Sigma_i$
 - $\Delta_i \subseteq S_i \times (\Sigma_i \cup \{\tau\}) \times S_i$
 - $\hat{S}_i \subseteq S_i$
 - $val_i \subseteq \Pi_i \times S_i$

Määritelmässä käytetään apukäsitettä *synkroninen tulo*

- rinnankytkennällä kootulla järjestelmällä on tyypillisesti valtava määrä saavuttamattomia tiloja
 - laskuharjoitukset
- ne eivät vaikuta käyttäytymiseen \Rightarrow ne halutaan jättää pois määritelmästä
- vertaa s. 32 $saav_osa(TK) = (S', \Sigma, \Delta', \hat{S}, \Pi, val')$ missä
 - $S' = saav_tilat(TK)$
 - $\Delta' = \{(s, a, s') \in \Delta \mid s \in S'\}$
 - $val' = val \cap (\Pi \times S')$

- saavuttamattomia tiloja on hankala karsia pois heti
⇒ ensin määritellään synkroninen tulo, jossa niitä ei ole vielä karsittu
- synkroninen tulo on tärkeämpi teorian kuin käytännön kannalta
 - yksinkertaistaa matematiikkaa
 - tietokoneohjelma saadaan laskemaan saavutettava osa suoraan

Rinnankytkentä voi suorittaa sen nimisiä näkyviä tapahtumia, mitä sen osatkin voivat

⇒ rinnankytkennän aakkosto on osatilakoneiden aakkostojen unioni

- $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \dots \cup \Sigma_n$
- toteuttaa $\tau \notin \Sigma$, koska $\forall i: \tau \notin \Sigma_i$

Rinnankytkennän tila sisältää tiedon jokaisen osatilakoneen paikallisesta tilasta, eikä muuta

⇒ rinnankytkennän tila määritellään vektorina, jonka komponentteina ovat osatilakoneiden tilat

- $\langle s_1, s_2, \dots, s_n \rangle \in S_1 \times S_2 \times \dots \times S_n$

⇒ synkroniselle tulolle määritellään

$$S = S_1 \times S_2 \times \dots \times S_n,$$

rinnankytkennälle tästä karsitaan saavuttamattomat

- rinnankytkennän alkutilat ovat kaikki vektorit, jotka voidaan muodostaa osatilakoneiden alkutiloista
 - $\hat{S} = \hat{S}_1 \times \hat{S}_2 \times \dots \times \hat{S}_n$
 - takaa $\hat{S} \subseteq S$, koska $\forall i: \hat{S}_i \subseteq S_i$

- tausta-ajatus: nimi on yhteyden (esim. johdon) nimi

Tilasiirtymien määritelmä takaa $\Delta \subseteq S \times (\Sigma \cup \{\tau\}) \times S$

Tilapropositioiden osalta olisi monta mahdollisuutta

- tällä kurssilla tilapropositioita voi käyttää ainoastaan tiedon välittämiseen tilakoneesta ulkomaailmaan
 - kun ei vielä aivan tiedetä mikä olisi paras ratkaisu, valittiin matemaattisesti yksinkertaisin hyödyllinen

⇒ tilakoneet eivät voi tutkia toistensa tilapropositioiden arvoja

- (muita mahdollisuuksia olisi ollut esimerkiksi liittää tilasiirtymän alku- (ja loppu)ehtoon mahdollisuus testata toisten osatilakoneiden tilapropositioita)
- jotta eri osatilakoneiden tilapropositiot eivät menisi sekaisin, vaaditaan, että niillä on eri nimet
 - $\Pi_i \cap \Pi_j = \emptyset$ aina kun $1 \leq i < j \leq n$
- (muita mahdollisuuksia olisi ollut
 - tuloksen tilapropositioiden arvo kootaan osatilakoneiden tilapropositioiden arvoista loogisella lausekkeella
 - tilasiirtymä on mahdollinen vain, jos kaikki osatilakoneet ovat samaa mieltä yhteisen tilapropositioidensa loppuarvosta)

⇒ rinnankytkennän tilapropositioiden joukko on kaikki osatilakoneiden tilapropositiot

- $\Pi = \Pi_1 \cup \Pi_2 \cup \dots \cup \Pi_n$

- tilapropositioiden arvo on se, minkä sen omistava osatilakone sille arvoksi antaa paikallisessa tilassaan
 - $val = \{ (\pi, \langle s_1, s_2, \dots, s_n \rangle) \in \Pi \times S \mid \exists i: 1 \leq i \leq n: (\pi, s_i) \in val_i \}$
 - selvästi $val \subseteq \Pi \times S$

Tilasiirtymien $\langle s_1, s_2, \dots, s_n \rangle \xrightarrow{a} \langle s'_1, s'_2, \dots, s'_n \rangle$ määritelmä jakautuu kahteen tapaukseen

1. näkymätön tilasiirtymä:

- **jokin** (yksi!) tilakone suorittaa näkymättömän tilasiirtymän
- muut tilakoneet pysyvät paikoillaan

- $a = \tau$, ja on olemassa $1 \leq i \leq n$ siten, että $(s_i, \tau, s'_i) \in \Delta_i$, ja $s'_j = s_j$ kun $1 \leq j \leq n$ ja $j \neq i$

- näkymätön tilasiirtymä on näkymätön myös naapuritilakoneille

2. näkyvä tilasiirtymä:

- **jokainen** tilakone, jonka aakkostoon kyseinen kokonaisnimi kuuluu suorittaa sen
- muut tilakoneet pysyvät paikoillaan

- $a \in \Sigma$, ja jokaiselle $1 \leq i \leq n$ $(s_i, a, s'_i) \in \Delta_i$, kun $a \in \Sigma_i$, ja $s'_i = s_i$, kun $a \notin \Sigma_i$

- näkyvä tilasiirtymä on tilakoneiden keskinäistä ja/tai tilakoneiden ja järjestelmän ympäristön välistä vuorovaikutusta

- kaikkien osapuolten tulee olla samaa mieltä tapahtumassa välitettävistä arvoista
⇒ samaa mieltä kokonaisnimestä

- jos kokonaisnimi on koottu tapahtuman nimestä ja tapahtuman parametreista, tapahtumaan osallistuu jokainen osatilakone, joka tuntee ko. tapahtuman nimen
 - ts. sisältää nimen aakkostossaan
 - jos parametrien määrästä, tyypeistä ja/tai arvoista ei päästä sopuun, on tapahtuma mahdoton

Määritelmä 3.1 Jos $TK_1 = (S_1, \Sigma_1, \Delta_1, \hat{S}_1, \Pi_1, val_1)$, $TK_2 = (S_2, \Sigma_2, \Delta_2, \hat{S}_2, \Pi_2, val_2)$, ..., $TK_n = (S_n, \Sigma_n, \Delta_n, \hat{S}_n, \Pi_n, val_n)$ ovat tilakoneita siten, että

$$\Pi_i \cap \Pi_j = \emptyset \text{ aina kun } 1 \leq i < j \leq n,$$

niin niiden synkroninen tulo on

$$TK_1 \times TK_2 \times \dots \times TK_n = (S, \Sigma, \Delta, \hat{S}, \Pi, val),$$

missä

- $S = S_1 \times S_2 \times \dots \times S_n$
- $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \dots \cup \Sigma_n$
- $(s, a, s') \in \Delta$, missä $s = \langle s_1, s_2, \dots, s_n \rangle \in S$ ja $s' = \langle s'_1, s'_2, \dots, s'_n \rangle \in S$, jos ja vain jos joko $a = \tau$, ja on olemassa $1 \leq i \leq n$ siten, että $(s_i, \tau, s'_i) \in \Delta_i$, ja $s'_j = s_j$ kun $1 \leq j \leq n$ ja $j \neq i$
tai $a \in \Sigma$, ja jokaiselle $1 \leq i \leq n$ $(s_i, a, s'_i) \in \Delta_i$, kun $a \in \Sigma_i$, ja $s'_i = s_i$, kun $a \notin \Sigma_i$
- $\hat{S} = \hat{S}_1 \times \hat{S}_2 \times \dots \times \hat{S}_n$
- $\Pi = \Pi_1 \cup \Pi_2 \cup \dots \cup \Pi_n$
- $val = \{ (\pi, \langle s_1, s_2, \dots, s_n \rangle) \in \Pi \times S \mid \exists i: 1 \leq i \leq n: (\pi, s_i) \in val_i \}$

Rinnankytkentä on

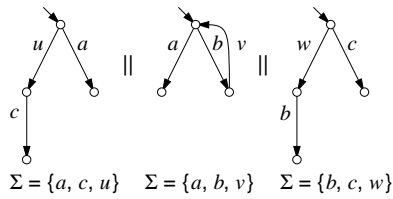
$$TK_1 \parallel TK_2 \parallel \dots \parallel TK_n =$$

$$saav_osa(TK_1 \times TK_2 \times \dots \times TK_n).$$

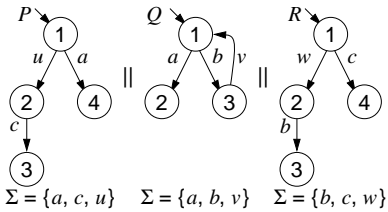
□

Esimerkki rinnankytkennästä

- piirrettävä seuraava rinnankytketty tilakone:



- kirjanpidon ja puhumisen helpottamiseksi
 - annamme osatilakoneille nimet
 - annamme osatilakoneiden tiloille numerokoodit

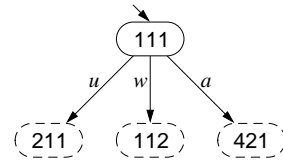


- rinnankytkennän alkutilat ovat osatilakoneiden alkutilojen yhdistelmät, siis (1, 1, 1)
 - esimerkissä jokaisella osatilakoneella ja myös tuloksella on vain yksi alkutila
 - lyhyden vuoksi kirjoitamme sen "111"

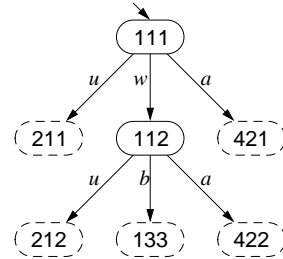


- katkoviivalla ilmaisemme, että tilan jatkoa ei ole vielä tutkittu

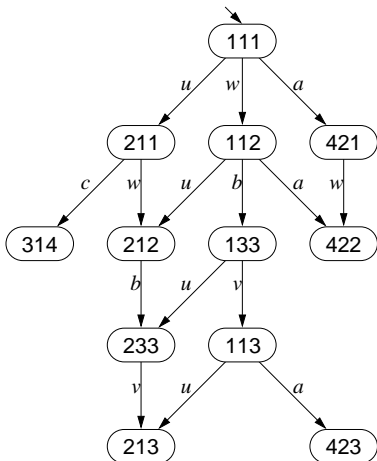
- alkutilasta voi tehdä tilasiirtymiä seuraavasti:
 - a:lla tilaan 421
 - b:llä ei minnekään, koska R estää sen
 - c:llä ei minnekään, koska P estää sen
 - u:lla tilaan 211
 - v:llä ei minnekään, koska Q estää sen
 - w:llä tilaan 112



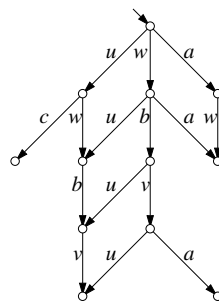
- tilan 112 käsittely tuottaa



- jatkamalla samaan tapaan kunnes kaikki tilat on käsitelty saamme



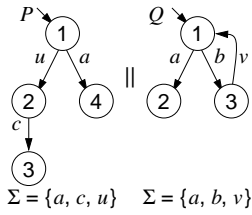
- poistakaamme lopulta tilojen numerokoodit:



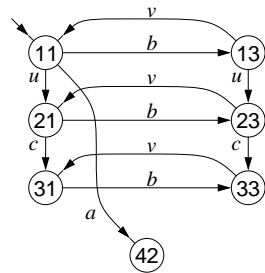
Käytännön vinkkejä rinnankytkentöjen piirtämiseksi

- valitse tilasiirtymien piirtämiseksi joitakin pääsuuntia
- yritä piirtää saman osatilakoneiden joukon suorittamat tilasiirtymät aina samaan (tai vastakkaiseen) suuntaan
- joitakin käytäntöjä äskeisessä esimerkissä:
 - P:n paikallinen tapahtuma u aina alavasemmalle
 - P:n ja Q:n yhteinen tapahtuma a aina alaoikealle
- rinnankytkentä tuottaa usein monta säännöllistä suunnikasta
 - esim. yllä suunnikkaat $uwww$, $waaw$, $ubbu$
 - ⇒ jos jokin suunnikas on vajaa, kannattaa tarkistaa, puuttuuko siitä jotakin
 - esim. bv :n puuttuminen oikeanpuolimmaiselta sarakkeelta ei ole virhe

- myös tilojen numerot noudattavat usein säännöllistä hahmoa
 - esim. edellä vasemmanpuoleisin numero on kunkin sarakkeen sisällä sama
- jos on vain kaksi osatilakonetta, on usein kätevää käyttää tilojen numeroita x - ja y -koordinaatteina
 - esim.



tuottaa



- yli kahden tilakoneen rinnankytkentä voidaan laskea useana kahden tilakoneen rinnankytkentänä
 - esim. $P \parallel Q \parallel R \parallel S = (P \parallel Q) \parallel (R \parallel S)$
 - onko ilmeistä, että näin saadaan oikea tulos?

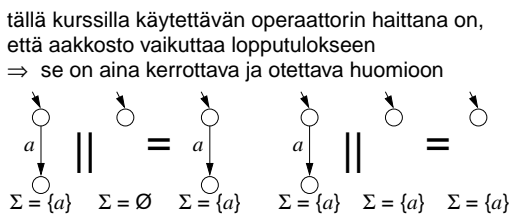
Aito rinnakkaisuus

- edellä oleva määritelmä sulkee pois mahdollisuuden, että kaksi eri osatilakonetta tekee erinimiset tilasiirtymät täysin yhtäaikaa
 - tämä mahdollisuus voitaisiin ottaa mukaan matematiikkaan nimeämällä kokonaistilakoneen kaaret tapahtumien epätähjällä joukolla
 - tällainen toisi vain vähän (jos ollenkaan) lisäarvoa teorian käytölle, ja monimutkaistaisi matematiikkaa ja ohjelmistojä
 - aitoa rinnakkaisuutta on tutkittu ja yritetty soveltaa kauan!
- ⇒ emme tee niin
- pidämme kuitenkin mahdollisuuden mielessä, jos sille löytyisi joskus tarvetta

Vaihtoehtoisia rinnankytkennän määritelmiä

- prosessialgebrallisessa kirjallisuudessa on esitetty muitakin tapoja päättää, mitkä prosessit (so. tilakoneet) osallistuvat tilasiirtymään
- Lotos-kielessä on kolme operaattoria:
 - $P \parallel [a_1, a_2, \dots, a_k] Q$: a_1, a_2, \dots, a_k sekä ns. onnistunut lopetus "δ" suoritetaan yhdessä, muut yksin
 - $P \parallel \parallel Q$: vain δ suoritetaan yhdessä
 - $P \parallel Q$: kaikki paitsi τ suoritetaan yhdessä

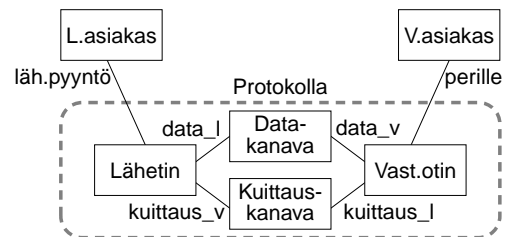
- CCS-kielessä näkyvät tapahtumat on jaettu tavallisiin ja hatutettuihin, ja $P \mid Q$ toimii seuraavasti:
 - P voi tehdä minkä tahansa tapahtumansa yksin
 - Q voi tehdä minkä tahansa tapahtumansa yksin
 - P ja Q voivat tehdä tapahtuman yhdessä siten, että toinen tekee sen tavallisena ja toinen hatutettuna; ulos näytettävä nimi on τ



- kurssin "||":n ilmaisuvoima on sama kuin Lotos- ja suurempi kuin CCS-operaattoreilla
 - CCS ei salli > 2 osapuolta samaan tilasiirtymään
 - Lotoksen "[[a₁, a₂, ..., a_k]]" voi matkia kurssin "||"
 - luvussa 3.4 nähdään, miten kurssin "||" voi matkia monenlaisia rinnankytkentöjä
- kurssin "||" on matemaattisesti yksinkertaisempi kuin Lotoksen "[[a₁, a₂, ..., a_k]]"
- liitännäinen
- "[[a₁, a₂, ..., a_k]]":n matkiminen kurssin "||":lla voi vaatia eksponentiaalisen määrän operaattoreita (Valmari & Kervinen 2002)
- pätee kaikille matkimiskeinoille, epäsuorillekin!
- tällä kurssilla matemaattinen yksinkertaisuus asetetaan kätevyyden edelle
 - tavoitteena ilmiöiden ymmärtäminen eikä suunnittelu- tms. menetelmä

Yhteisen muistin vuorovaikutusmekanismien matkiminen

- tilakoneiden rinnankytkennällä on helppo matkia yhteiseen muistiin perustuvia vuorovaikutusmekanismeja tekemällä yhteisestä muistista itsenäinen tilakone
- esimerkki: datakanava



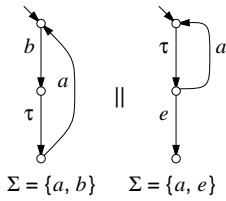
- laskuharjoituksissa on pari esimerkkiä lisää
- ⇒ havainto

Kurssin rinnankytkennällä voi matkia jokseenkin kaikkia käytössä olevia vuorovaikutusmekanismeja.

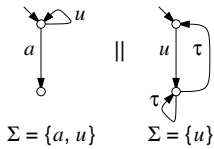
Harjoitustehtäviä

1. Piirrä seuraavat tilakoneiden rinnankytkennät.

(a)



(b)



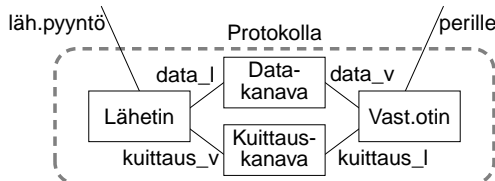
2. (a) Kuinka monta tilaa on synkronisella tulolla $TK_1 \times TK_2 \times \dots \times TK_n$, ilmaistuna $TK_1:n, TK_2:n, \dots, TK_n:n$ tilojen määrän funktiona?
- (b) Olettakaamme, että tiedetään $TK_1:n, TK_2:n, \dots, TK_n:n$ saavutettavien osien tilojen ja alkutilojen määrät $|S_1|, |S_2|, \dots, |S_n|, |\hat{S}_1|, |\hat{S}_2|, \dots, |\hat{S}_n|$ mutta ei muuta. Anna alaraja ja yläraja $TK_1 \parallel TK_2 \parallel \dots \parallel TK_n$:n tilojen määrälle. Osoita esimerkein, että rajasi ovat tarkat.
3. Osoita esimerkin avulla, että Lotoksen "[a_1, a_2, \dots, a_k]" ei ole liittäminen.

3.2 Kätkentä, ei muuttujia

Kätkentä muuttaa näkyviä tapahtumia näkymättömiksi

- **hide** a, b **in** TK käyttäytyy muuten aivan kuten TK , mutta aina kun TK tekisi a - tai b -tilasiirtymän (mahdollisesti parametrien kanssa), **hide** a, b **in** TK tekee vastaavan tilasiirtymän τ -nimisenä ja ilman parametreja
- **hide** on peräisin ISON standardoimasta kielestä Lotos, ja on eri nimisenä mutta toiminnaltaan samanlaisena CSP-kielessä
 - mutta ei CCS-kielessä

Tyypillinen käyttötilanne: rinnankytkennän jälkeen tarpeettomaksi käyneiden tapahtumien abstrahointi pois



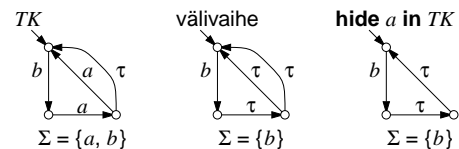
hide a_1, a_2, \dots, a_n **in** TK voidaan määritellä epäformaalisti seuraavasti:

- vaaditaan, että $a_i \neq \tau$, kun $1 \leq i \leq n$
- kätkevien tapahtumien nimet a_1, a_2, \dots, a_n poistetaan aakkostosta
- kätkevien tapahtumien nimet (tapahtuman parametreineen) korvataan " τ ":lla tilasiirtymissä

4. (a) Piirrä tilakone "Lippu", jonka aakkosto on { "nosta", "laske", "ylhäällä" ja "alhaalla" } ja joka toimii kuten aakkosto vihjaa.
- (b) Mallinna käyttöjärjestelmien kurssilta tuttu opastin (semaphor) tilakoneena.
5. (a) Kirjoita rinnankytkennälle vaihtoehtoinen määritelmä siten, että osatilakoneilla voi olla yhteisiä tilapropositiioita, mutta tilasiirtymä ei ole vireessä, jos osatilakoneet ovat eri mieltä yhteisen tilapropositiion seuraavasta arvosta.
- (b) Kirjoita rinnankytkennälle vaihtoehtoinen määritelmä siten, että osatilakoneen tilasiirtymä voi testata muiden osatilakoneiden tilapropositioiden arvoja.

- jos (kokonaistilakoneessa) syntyy useampia kuin yksi τ -tilasiirtymä samaan suuntaan samojen tilojen välillä, ylimääräiset poistetaan
 - (kun muuttujat ovat mukana, tämä ei päde: on otettava huomioon myös R -relaatiot)
- huomautus! sallimme joukon $\{ a_1, a_2, \dots, a_n \}$ olevan ääretön
 - tärkeää, koska tapahtumien parametrit tuottavat samalle tapahtuman nimelle äärettömästi eri kokonaisnimiä

Esimerkki: **hide** a **in** TK



Kätkevien tapahtumien nimien poistaminen aakkostosta on tarpeen, koska aakkosto määrää, mihin synkronointeihin tilakone osallistuu

- ilman kätkentää **hide** a **in** TK olisi kyvytön suorittamaan a -tapahtumia ja estäisi muitakin suorittamasta niitä
- (vertaa kätkeviä tapahtumien nimiä ohjelmointikielten paikallisiin muuttujiin)

Määritelmä 3.2 Jos $TK = (S, \Sigma, \Delta, \hat{S}, \Pi, val)$ on tilakone ja $a_i \neq \tau$ kun $1 \leq i \leq n$, niin **hide** a_1, a_2, \dots, a_n **in** $TK = (S', \Sigma', \Delta', \hat{S}', \Pi', val')$, missä

- $S' = S$
- $\Sigma' = \Sigma - \{a_1, a_2, \dots, a_n\}$
- $\Delta' = \{(s, \text{hide}(a), s') \mid (s, a, s') \in \Delta\}$, missä $\text{hide}(a) = \tau$, kun $a \in \{a_1, a_2, \dots, a_n\}$, ja $\text{hide}(a) = a$, muutoin
- $\hat{S}' = \hat{S}$
- $\Pi' = \Pi$
- $val' = val$ □

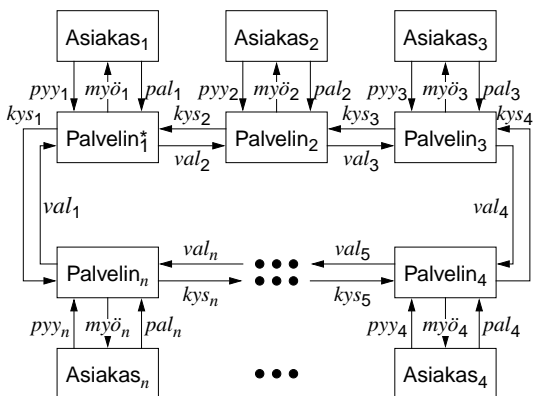
Huomautuksia

- määritelmä hävittää ylimääräiset τ -tilasiirtymät automaattisesti, koska joukko ei voi sisältää samaa alkioita useammin kuin kerran
- saattaisi tuntua luontevalta vaatia $a_i \in \Sigma$ vaatimuksen $a_i \neq \tau$ sijaan, mutta tällainen muutos ei vaikuta kätkenän toimintaan (mutta rajaisi sen laillisten käyttökohteiden joukkoa)
- myös olisi voitu sallia $a_i = \tau$
 - τ :n tarkoitus olla näkymätön, niin että siihen ei pääse mitenkään tarttumaan
 - \Rightarrow tuntuu epäjohtonumkaisuelta sallia operaattoreiden kohdistuvan siihen (vaikka ilman vaikutustakin), joten vaadimme $a_i \neq \tau$
 - tämä asia ei ole olennainen

3.3 Moniuudelleennimeäminen, ei muuttujia

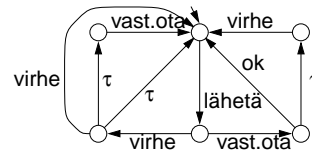
Moniuudelleennimeäminen muuttaa tilasiirtymien nimiä, ja tarvittaessa kopioi tilasiirtymän usealle eri nimelle

- muuntaa näkyviä tapahtumia näkyviksi tapahtumiksi
- esimerkiksi $TK[alb, a/c, bla, d/c]$ muuntaa kaikki b -tilasiirtymät a -tilasiirtymiksi ja päinvastoin, ja muuntaa c -tilasiirtymät sekä a - että d -tilasiirtymiksi
- sallii saman tilakoneen käytön useana kopiona samassa järjestelmässä
- esimerkki: valtuudenkierrätysjärjestelmä

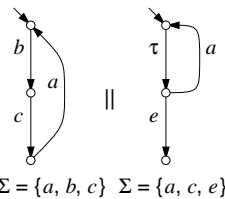


Harjoitustehtäviä

1. Piirrä **hide** *ok, virhe, täystuho* **in** TK , missä TK on kuvan tilakone. Tarvitseeko TK :n aakkostosta olettaa mitään?



2. Piirrä **hide** a **in** TK , missä TK on kuvasta saatava tilakone.



3. Määrittele operaattori, joka poistaa yhden tai useamman tilaproposition tilakoneesta.

4. Päteekö aina

$$\text{hide } a \text{ in } (P \parallel Q) = (\text{hide } a \text{ in } P) \parallel (\text{hide } a \text{ in } Q)?$$

Jollei, anna vastaesimerkki. Jos kyllä, todista, että se pätee aina tai selitä, miksi se pätee aina.

- "moni"-mahdollisuudesta on hyötyä
 - tähtikytkeissä järjestelmissä
 - myöhemmin esitettävän yleistetyn rinnankytkennän matematiikan rakentamisessa
- nimenmuutoslista on usein kömpelö kaavoissa \Rightarrow käytämme sen edustajana symbolia Φ , siis $TK\Phi$
- operaattori on saanut vaikutteita sekä Lotoksesta että CSP:stä
- tunnetaan myös nimellä "relaationaalinen uudelleennimeäminen"

$TK[b_1/a_1, \dots, b_n/a_n]$ määritellään samaan tyyliin kuin kätkenä

- vaadimme, että $a_i \neq \tau$ ja $b_i \neq \tau$ kun $1 \leq i \leq n$
- siis
 - τ :ta ei voi nimetä uudelleen
 - uudelleennimeäminen ei voi tuottaa τ :ta
- moniuudelleennimeäminen \Rightarrow ei vaadita " $a_i \neq a_j$ kun $i \neq j$ "
- jokainen a_i , $1 \leq i \leq n$, poistetaan aakkostosta
- aakkostoa täydennetään niillä b_i , joita vastaava a_i kuuluu alkuperäiseen aakkostoon
 - tämä sääntö estää mielivaltaisten symbolien lisäämisen aakkostoon
 - ei olennainen
- tilasiirtymät korvataan uudelleen nimetyillä tilasiirtymillä, ja mahdollisesti syntyvät ylimääräiset kopiot poistetaan
- niihin aakkosiin ei kajota, jotka eivät esiinny a_i :nä
- sallitaan ääretön $b_1/a_1, \dots, b_n/a_n$

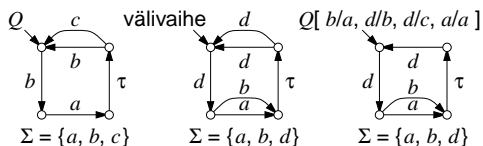
Määritelmä 3.3 Jos $TK = (S, \Sigma, \Delta, \hat{S}, \Pi, val)$ on tilakone ja $a_i \neq \tau \neq b_i$ kun $1 \leq i \leq n$, niin

$TK[b_1/a_1, b_2/a_2, \dots, b_n/a_n] = (S', \Sigma', \Delta', \hat{S}', \Pi', val')$, missä

- $S' = S$
- $\Sigma' = (\Sigma - \{a_1, a_2, \dots, a_n\}) \cup \{b_i \mid a_i \in \Sigma\}$
- $\Delta' = \{(s, a', s') \mid \exists a: (s, a, s') \in \Delta \wedge \text{rename}(a, a')\}$, missä $\text{rename}(a, a')$ on voimassa jos ja vain jos $\exists i; 1 \leq i \leq n: a = a_i \wedge a' = b_i$, tai $a' = a \notin \{a_1, a_2, \dots, a_n\}$
- $\hat{S}' = \hat{S}$
- $\Pi' = \Pi$
- $val' = val$

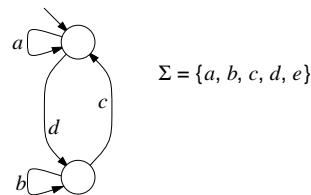
□

Esimerkki: $Q[b/a, d/b, d/c, a/a]$



Harjoitustehtäviä

1. Piirrä $TK[on_0/a, on_1/b, nollaa/a, nollaa/c]$, missä TK on kuvan tilakone.



2. Valitse k, e_1, \dots, e_k , ja f_1, \dots, f_k siten, että $TK[f_1/e_1, \dots, f_k/e_k] = TK[b_1/a_1, \dots, b_n/a_n][d_1/c_1, \dots, d_m/c_m]$
Jos menee kovin hankalaksi, niin laajenna ensin “[$d_1/c_1, \dots, d_m/c_m$]” siten, että kaikki TK :n aakkoset ovat mukana.
3. (a) Määrittele operaattori, joka ottaa käyttöön uuden tilapropositioiden ja antaa sille arvon olemassa olevien tilapropositioiden arvojen loogisena lausekkeena. Voit merkitä ko. loogista lauseketta $\varphi(\pi_1, \dots, \pi_n)$.
(b) Osoita, että tilapropositioiden moniuidelleennimeäminen voidaan toteuttaa tähän mennessä määritellyillä operaattoreilla.

3.4 TVT-työkalun rinnankytkentä

TVT-työkalussa (Tampere Verification Tool) on rinnankytkentäoperaattori, joka

- yhdistää perinteisen rinnankytkennän, kätken, moniuidelleennimeäminen ja tilapropositiot
- tukee joitakin edistyneitä verifointimenetelmiä
 - kätevä datan mallinnus
 - kätevä saman tilakoneen käyttö useana osatilakoneena
 - ns. itsepäisten joukkojen menetelmän edellyttämä tieto kätkenästä
- ei ole liian vaikea toteuttaa
- on hankala antaa kätevä matemaattinen merkki
- Konsta Karsisto 1997, 2003; Hansen & Virtanen & Valmari 2003

Tässä esitetään sen peruseriaate

- ei tarkkaa syntaksia yms.
- (tämän yksityiskohtia ei ole tarkistettu TVT:n toteutusta vastaan)

Operaattori muodostuu

- tuloaakkostosta Σ
- (mahdollisesti tyhjistä) joukosta *synkronointisääntöjä*
- (mahdollisesti tyhjistä) joukosta *tilapropositiosääntöjä*

Olko

- $n \geq 1$
- osatilakoneet $(S_1, \Sigma_1, \Delta_1, \hat{S}_1, \Pi_1, val_1), \dots, (S_n, \Sigma_n, \Delta_n, \hat{S}_n, \Pi_n, val_n)$
 - TVT:ssä tilakoneella on aina tasan yksi alkutila

Tulostilakone $(S, \Sigma, \Delta, \hat{S}, \Pi, val)$ on seuraavan tilakoneen saavutettava osa:

- $S = S_1 \times \dots \times S_n$ (kuten tavallisessa “[τ]”)
- Σ annetaan erikseen
- Δ määräytyy synkronointisäännöistä jatkossa kuvattavalla tavalla
- $\hat{S} = \langle \hat{S}_1, \dots, \hat{S}_n \rangle$
- Π ja val määräytyvät tilapropositiosäännöistä kuten jatkossa (ei aivan pian) kerrotaan

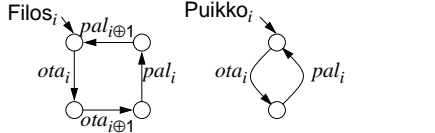
Synkronointisääntö

- olkoon “ $_$ ” symboli siten, että “ $_$ ” $\notin \Sigma_1 \cup \dots \cup \Sigma_n$
 - tarkoittaa “osatilakone ei osallistu tilasiirtymään”
- *synkronointisääntö* on $(n+1)$ -vektori $\langle a_1, \dots, a_n; a \rangle$, missä
 - $a \in \Sigma \cup \{\tau\}$
 - $a_i \in \Sigma_i \cup \{_$ “}, kun $1 \leq i \leq n$
 - ainakin yksi a_i ei ole “ $_$ ”
- säännön tuottama tuloksen tilasiirtymä
 - on a -tilasiirtymä
 - muodostuu jokaisen sellaisen osatilakoneen yhtäaikaista a_i -tilasiirtymästä, jolle $a_i \neq _$ ”
- sen lisäksi osatilakoneiden τ -tilasiirtymät aiheuttavat lopputulokseen τ -tilasiirtymiä kuten tavallisessa rinnankytkennässä

- niinpä $\langle s_1, \dots, s_n \rangle \rightarrow \langle s'_1, \dots, s'_n \rangle$ jos ja vain jos **joko** on olemassa synkronointisääntö $\langle a_1, \dots, a_n; a \rangle$ siten, että $b = a$, ja aina kun $i \in \{1, 2, \dots, n\}$:
 - jos $a_i \neq \text{"-"}$, niin $s_i \rightarrow s'_i$
 - jos $a_i = \text{"-"}$, niin $s'_i = s_i$
- tai** $b = \tau$, ja on olemassa $i \in \{1, 2, \dots, n\}$ siten, että
 - $s_i \rightarrow s'_i$ ja
 - $s'_j = s_j$ aina kun $1 \leq j \leq n \wedge j \neq i$

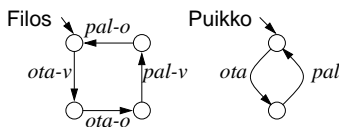
Käyttöesimerkki: ruokailevat filosofit

- perinteinen malli: $\text{Puiikko}_1 \parallel \text{Filos}_1 \parallel \dots \parallel \text{Puiikko}_n \parallel \text{Filos}_n$, missä osatilakoneet ovat:



– “ $\oplus 1$ ” lasketaan “mod $n + 1$ ”; siis $n \oplus 1 = 1$

- riisa: Filos ja Puiikko täytyy olla useana kopiona eri tapahtumanimillä
- malli käyttäen TVT:n rinnankytkentää: tilakoneet Puiikko, Filos, Puiikko, Filos, ..., Puiikko, Filos



TVT:n rinnankytkentäoperaattorin simulointi klassisilla operaattoreilla

- TVT:n rinnankytkentäoperaattoria ilman tilapropositio-ominaisuuksia voi simuloida operaattoreilla “ \parallel ”, “**hide**” ja “[\cdot / \cdot , ..., \cdot / \cdot]”
- \Rightarrow se ei lisää periaatteellista ilmaisuvoimaa
 - tekee kuitenkin mallittamisesta kätevää!
- olkoon **stop_A** se yksitilainen muuttujan tilakone, jolla ei ole tilasiirtymiä eikä tilapropositioita, ja jonka aakkosto on A
 - $P \parallel \text{stop}_A$ käyttäytyy kuten P , josta on poistettu kaikki tilasiirtymät joiden nimi on A :ssa
- olkoot synkronointisäännöt $\{ \langle a_{11}, \dots, a_{1n}; a_1 \rangle, \dots, \langle a_{k1}, \dots, a_{kn}; a_k \rangle \}$
- olkoot
 - r_1, \dots, r_k uusia nimiä, siis ei joukossa $\Sigma_1 \cup \dots \cup \Sigma_n$ (ts. jokaiselle synkronointisäännölle oma tunnus)
 - $H = \{ r_j \mid 1 \leq j \leq k \wedge a_j = \tau \}$ (ts. niiden synkronointisääntöjen tunnukset, jotka tuottavat τ :n)
 - $B_i = \Sigma_i - \{ a_{1i}, \dots, a_{ki} \}$, kun $1 \leq i \leq n$ (ts. ne TK_i :n näkyvien tapahtumien nimet, joita ei käytetä synkronointisäännöissä)
 - $B = B_1 \cup \dots \cup B_n$
 - $C = \Sigma - \{ a_1, \dots, a_k \}$ (ts. ne annetun aakkoston tapahtumanimet, joita mikään synkronointisääntö ei tuota)



- synkronointisäännöt

- $\langle \text{ota}, \text{ota-v}, -, -, -, \dots -, -; \text{ota}_1 \rangle$
- $\langle -, \text{ota-o}, \text{ota}, -, -, \dots -, -; \text{ota}_2 \rangle$
- $\langle \text{pal}, \text{pal-v}, -, -, -, \dots -, -; \text{pal}_1 \rangle$
- $\langle -, \text{pal-o}, \text{pal}, -, -, \dots -, -; \text{pal}_2 \rangle$
- $\langle -, -, \text{ota}, \text{ota-v}, -, \dots -, -; \text{ota}_2 \rangle$
- $\langle -, -, -, \text{ota-o}, \text{ota}, \dots -, -; \text{ota}_3 \rangle$
- $\langle -, -, \text{pal}, \text{pal-v}, -, \dots -, -; \text{pal}_2 \rangle$
- $\langle -, -, -, \text{pal-o}, \text{pal}, \dots -, -; \text{pal}_3 \rangle$
- ...
- $\langle -, -, -, -, -, \dots \text{ota}, \text{ota-v}; \text{ota}_n \rangle$
- $\langle \text{ota}, -, -, -, -, \dots -, \text{ota-o}; \text{ota}_1 \rangle$
- $\langle -, -, -, -, -, \dots \text{pal}, \text{pal-v}; \text{pal}_n \rangle$
- $\langle \text{pal}, -, -, -, -, \dots -, \text{pal-o}; \text{pal}_1 \rangle$

Monia perinteisiä operaattoreita voi simuloida TVT:n rinnankytkennällä

- perinteinen rinnankytkentä “ \parallel ”: valitaan $\Sigma = \Sigma_1 \cup \dots \cup \Sigma_n$ ja käytetään synkronointisääntöjä $\{ \langle a_1, \dots, a_n; a \rangle \mid a \in \Sigma \}$, missä
 - $a_i = a$, jos $a \in \Sigma_i$
 - $a_i = \text{"-"}$, jos $a \notin \Sigma_i$
- hide** a_1, \dots, a_n in TK_1 : valitaan $\Sigma = \Sigma_1 - \{ a_1, a_2, \dots, a_n \}$ ja käytetään synkronointisääntöjä $\{ \langle a; \tau \rangle \mid a \in \Sigma_1 \cap \{ a_1, \dots, a_n \} \} \cup \{ \langle a; a \rangle \mid a \in \Sigma_1 - \{ a_1, \dots, a_n \} \}$
- $TK_1[b_1/a_1, b_2/a_2, \dots, b_n/a_n]$: valitaan $\Sigma = (\Sigma_1 - \{ a_1, a_2, \dots, a_n \}) \cup \{ b_i \mid a_i \in \Sigma_1 \}$ ja käytetään synkronointisääntöjä $\{ \langle a_i; b_i \rangle \mid a_i \in \Sigma_1 \wedge 1 \leq i \leq n \} \cup \{ \langle a; a \rangle \mid a \in \Sigma_1 - \{ a_1, \dots, a_n \} \}$

- simulaatio on muotoa $(\text{hide } H \cup B \text{ in } (TK_1[G_1] \parallel \dots \parallel TK_n[G_n] \parallel \text{stop}_B)) [G]$ $\parallel \text{stop}_C$ missä

- G_i on $r_1/a_{1i}, \dots, r_k/a_{ki}$, josta on poistettu ne r_j/a_{ji} , joille $a_{ji} = \text{"-"}$
- G on $a_1/r_1, \dots, a_k/r_k$, josta on poistettu ne a_j/r_j , joille $a_j = \tau$

Kuinka simulaatio toimii?

- G_i nimeää ne TK_i :n tilasiirtymät uudelleen, jotka voivat osallistua synkronointisääntöihin
 - uusi nimi on säännölle varattu r_j
 - jos tilasiirtymä voi osallistua useaan sääntöön, niin sille annetaan monta uutta nimeä, yksi kullekin säännölle
- stop_B** estää TK_i :tä suorittamasta niitä tilasiirtymiään, jotka eivät osallistu mihinkään synkronointisääntöön
 - B olisi voitu yhtä hyvin määritellä $B = \Sigma_1 \cup \dots \cup \Sigma_n$
- sisemmät “ \parallel ” antavat tarkalleen niiden tilasiirtymien tapahtua, jotka syntyvät synkronointisäännöistä, paitsi että tilasiirtymän nimenä on yhä säännön nimi
- hide** $H \cup B$ antaa lopputuloksen τ -tilasiirtymille oikean nimen “ τ ”
- hide** $H \cup B$ myös poistaa aakkostosta ne alkuperäisten tapahtumien nimet, joita ei voi tapahtua koska ne eivät esiinny synkronointisäännöissä
- $[G]$ antaa lopullisille näkyville tilasiirtymille oikeat nimet

- **stop_C** lisää aakkostoon ne näkyvien tapahtumien nimet, jotka kuuluvat Σ:aan mutta puuttuvat tuloksesta
 - ts. ne, jotka eivät synny synkronointisäännöistä

Tilapropositiosäännöt

- rinnankytkennän lopputuloksen tilapropositiot ja niiden arvot määräytyvät tilapropositiosäännöistä
- tilapropositio tulkitaan tilan totuusarvoiseksi funktioksi
 - $val(\pi, s) \Leftrightarrow (\pi, s) \in val$
- *tilapropositiosääntö* on pari (π, λ) , jonka osat ovat
 - Π:n alkio π
 - looginen lauseke λ, jonka perusosat ovat joukosta $\{i.\pi \mid 1 \leq i \leq n \wedge \pi \in \Pi_i\}$
- “i.π” on vain kikka sen sanomiseksi, että
 - lauseke käyttää osatilakoneiden tilapropositioita
 - eri osatilakoneiden tilapropositioita kohdellaan erillisinä, vaikka niillä sattuisi olemaan sama nimi
- jokaisen Π:n alkion täytyy esiintyä täsmälleen kerran tilapropositiosäännön ensimmäisenä osana
 - ⇒ Π voidaan päätellä säännöistä
 - ⇒ sitä ei tarvitse ilmoittaa erikseen
- päteekö $(\pi, \langle s_1, s_2, \dots, s_n \rangle) \in val$ saadaan tietysti laskemalla kaavasta λ, missä $i.\pi \Leftrightarrow (\pi, s_i) \in val_i$
- (TVT:ssä on lisäksi muutama erikoinen tilapropositioden laji)

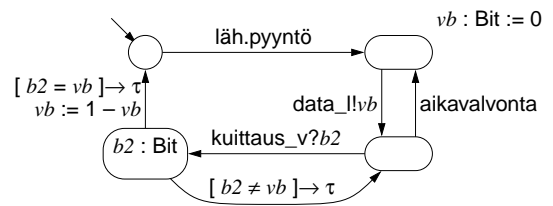
Synkronointisääntöjen luettelo voi olla pitkä

- kenties epäkäytännöllistä ihmisille
- ei ongelma kääntäjille, jotka tuottavat synkronointisäännöistä järjestelmäkuvauksista jollakin ihmisläheisemmällä kielellä

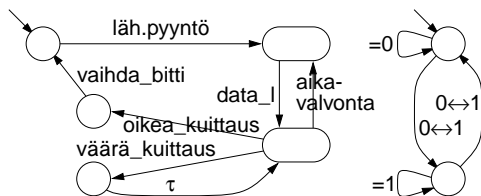
- kohta näytetään, miten synkronointisäännöillä voi mallittaa muuttujia kätevästi
- näimme jo toisen sovelluksen: järjestelmät, joissa on sama tilakone moneen kertaan
- jokainen sääntö vastaa synkronointitapaa
 - jokainen synkronointitapa täytyy tutkia tavallistakin rinnankytkentää laskettaessa
 - ⇒ ei lisätyötä
 - itse asiassa päinvastoin: rinnankytkijän ei tarvitse päätellä synkronointitapoja, koska ne on lueteltu valmiiksi
- ⇒ monimutkaisuus on saatu hyvin jaettua rinnankytkijän ja kääntäjän välille

Kokonaistilakoneen laskeminen TVT:n rinnankytkennällä

- tämä sovellus oli yksi pääsyy TVT:n rinnankytkentäoperaattorin kehittämiseen
- esitämme rakenteen ensin olettaen, että muuttujat ovat käytössä jokaisessa tilassa
- esitämme rakenteen vain esimerkin avulla



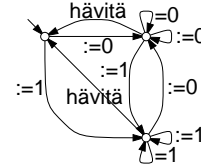
- kokonaistilakone
 - Diagram showing a sequence of states and transitions. Labels include 'läh.pyyntö', 'data_l(0)', 'kuittaus_v(1)', 'aikavalvonta', 'kuittaus_v(0)', and 'τ'.
- sama käyttäytyminen saadaan oheisilla muuttujattomilla tilakoneilla ja synkronointisäännöillä



```

< läh.pyyntö,    -;   läh.pyyntö >
< data_l,       =0;  data_l(0) >
< data_l,       =1;  data_l(1) >
< aikavalvonta, -;   aikavalvonta >
< oikea_kuittaus, =0; kuittaus_v(0) >
< oikea_kuittaus, =1; kuittaus_v(1) >
< väärä_kuittaus, =0; kuittaus_v(1) >
< väärä_kuittaus, =1; kuittaus_v(0) >
< vaihda_bitti, 0 ↔ 1; τ >
    
```

- esimerkissä
 - koska b2 esitettiin osana tilaa, oikea ja väärä kuittaus tarvitsevat lähettimessä eri nimet
 - muutoin synkronointisäännöt ovat suoraviivaiset
- vain joissakin tiloissa olemassa oleva muuttuja voidaan toteuttaa lisäämällä sille yksi arvo “olematon”, operaatio “hävitä” sekä sijoituslauseisiin liittyvät kaaret “olematon”-tilasta muihin tiloihin

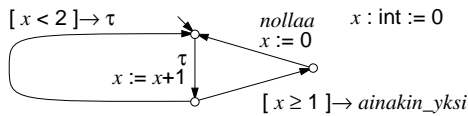


- ⇒ muuttujia voi varsin pitkälle matkia TVT:n rinnankytkennällä
- ⇒ siis myös tavallisella rinnankytkennällä, moniuudelleennimeämällä ja kätkenällä
- ⇒ havainto

Muuttujan käyttö on ymmärrettävissä rinnankytkentänä.

Harjoitustehtäviä

1. Esitä oheinen tilakone kontrollia kuvaavan muuttujattoman tilakoneen ja muuttujaa x kuvaavan muuttujattoman tilakoneen TVT-rinnankytkentänä.



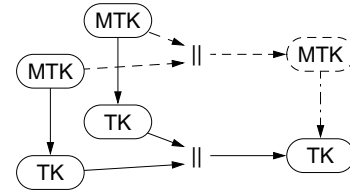
2. Spesifiointikielessä Lotos ja prosessialgebrassa CSP on rinnankytkentäoperaattori $|A|$, missä A on joukko näkyvien tapahtumien nimiä. $P|A|Q$ voi suorittaa tapahtuman kolmella tavalla: (1) P suorittaa yksinään tapahtuman jonka nimi ei kuulu A :han, (2) Q suorittaa yksinään tapahtuman jonka nimi ei kuulu A :han ja (3) P ja Q suorittavat yhtäaikaan tapahtuman, jonka nimi kuuluu A :han.

- (a) Näytä, miten kahden muuttujattoman tilapropositiottoman tilakoneen tavallinen rinnankytkentä voidaan esittää " $|A|$ ":n avulla.
- (b) Anna " $|A|$ ":lle samantapainen määritelmä kuin " $||$ ":lle on annettu luvussa 3.1 (sivu 63).
3. (a) Näytä, miten " $|A|$ " voidaan matkia TVT:n rinnankytkennällä.
- (b) Anna esimerkki tilanteesta, missä (a)-kohdan konstruktiosu tuottama sääntöjen määrä kasvaa eksponentiaalisesti lausekkeen koon funktiona. (Lausekkeessa voi olla monta rinnankytkentäoperaattoria.)

3.5 Muuttujallisten tilakoneiden operaatiot

On jo määritelty

- muuttujallisten tilakoneiden käyttäytyminen
 - kokonaistilakone
 - muuttujattomien tilakoneiden rinnankytkentä, kätöntä ja moniuidelleennimeäminen
- ⇒ muuttujallisten tilakoneiden rinnankytkennän käyttäytyminen on määritelty kuvan ehjien viivojen mukaisesti



Nyt haluamme määritellä rinnankytkennän jne. muuttujallisille tilakoneille siten, että lopputuloksen käyttäytyminen on käyttäytymisten rinnankytkentä

- katkoviivojen mukainen operaatio on määriteltävä siten, että kun siitä jatketaan pistekatkoviivalla, päästään oikeaan lopputulokseen
 - samaan lopputulokseen kuin ehjäviivoitettua reittiä
- pistekatkoviivan mukainen operaatio on jo määritelty
 - muuttujallisen tilakoneen käyttäytyminen

Kertaus sivulta 21: muuttujallisen tilakoneen osat

- S tilat
- $Types$ tyypit
 - kaikille tilakoneille yhteinen
 - $\forall T \in Types: \perp \notin T \neq \emptyset$
- V muuttujien määrä
- VC muuttujille mahdolliset arvoyhdistelmät; muotoa $(T_1 \cup \{\perp\}) \times \dots \times (T_V \cup \{\perp\})$ joillekin $T_1 \in Types, \dots, T_V \in Types$
- Σ näkyvien tilasiirtymien aakkosto
 - $\tau \notin \Sigma$
- Δ tilasiirtymät: jokainen Δ :n alkio on muotoa (s, a, R, s') , missä
 - $s \in S$ ja $s' \in S$
 - $a \in \Sigma \cup \{\tau\}$
 - $\exists k: R \subseteq VC \times (T_1 \times \dots \times T_k) \times VC$, missä $T_1, \dots, T_k \in Types$
 - jos $a = \tau$, niin $k = 0$
- $\hat{S} \subseteq S$ alkutilat
- alkuarvot: $\forall \hat{s} \in \hat{S}: Init(\hat{s}) \subseteq VC \wedge Init(\hat{s}) \neq \emptyset$
- Π tilapropositiot
- tilapropositioiden arvot: $val \subseteq \Pi \times S \times VC$

Oletukset

- $TK_i = (S_i, Types, V_i, VC_i, \Sigma_i, \Delta_i, \hat{S}_i, \Pi_i, Init_i, val_i)$ ($1 \leq i \leq n$) ovat tilakoneita
 - $Types$ on yhteinen myös lopputulokselle
- $\Pi_i \cap \Pi_j = \emptyset$ aina kun $1 \leq i < j \leq n$

- tapahtumien nimille järkevyysehto: jos $a \neq b$, niin
 - $(\{a\} \times U^*) \cap (\{b\} \times U^*) = \emptyset$
 - ts. eri tapahtumat eivät voi muuttua samannimisiksi jos niille annetaan sopivat parametrit

Muuttujallinen synkroninen tulo

- "helpot" osat
 - $S = S_1 \times S_2 \times \dots \times S_n$
 - $\hat{S} = \hat{S}_1 \times \hat{S}_2 \times \dots \times \hat{S}_n$
 - $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \dots \cup \Sigma_n$
 - $\Pi = \Pi_1 \cup \Pi_2 \cup \dots \cup \Pi_n$
- yhdistelmätilan muuttujat ovat tilojen muuttujat yhteensä
- ⇒ osatilakoneiden muuttujia koskeva tieto kootaan yhteen
 - $V = V_1 + V_2 + \dots + V_n$
 - $VC = VC_1 \times VC_2 \times \dots \times VC_n$
- kuvilla
 - kokonaisuuden muuttujat ovat osien muuttujat yhteensä
 - jos osilla on samannimisisiä muuttujia, erotetaan ne toisistaan vaikka alaindekseillä
- $Init(\langle \hat{s}_1, \hat{s}_2, \dots, \hat{s}_n \rangle) = Init(\hat{s}_1) \times Init(\hat{s}_2) \times \dots \times Init(\hat{s}_n)$
- $Init(\hat{s}_i)$:t voi tulkita myös muuttujia koskeviksi loogisiksi väittämissä, jolloin
 - $Init(\langle \hat{s}_1, \hat{s}_2, \dots, \hat{s}_n \rangle) \Leftrightarrow Init(\hat{s}_1) \wedge Init(\hat{s}_2) \wedge \dots \wedge Init(\hat{s}_n)$
 - kätevää kuvissa
- jatkoa varten etsimme osatilakoneen i muuttujien numerot lopputuloksessa
 - $e(i) = V_1 + V_2 + \dots + V_{i-1} + 1$ (e = "eka")
 - $v(i) = V_1 + V_2 + \dots + V_{i-1} + V_i$ (v = "vika")

- (paljonko on $v(n)?$)
 $\Rightarrow val = \{ (\pi, \langle s_1, s_2, \dots, s_n \rangle, \langle v_1, v_2, \dots, v_n \rangle) \in \Pi \times S \times VC \mid \exists i; 1 \leq i \leq n: (\pi, s_i, \langle v_{e(i)}, \dots, v_{v(i)} \rangle) \in val_i \}$
- jos val_i :t tulkitaan muuttujia koskeviksi loogisiksi väittämiksi, niin kuvissa $val(\pi, \langle s_1, s_2, \dots, s_n \rangle)$ on se $val_i(\pi, s_i)$, jolle $\pi \in \Pi_i$
- näkyvät tilasiirtymät ovat neliköt $(\langle s_1, \dots, s_n \rangle, a, R, \langle s'_1, \dots, s'_n \rangle)$, missä
 - kaikille $1 \leq i \leq n$ pätee $s_i \in S_i$ ja $s'_i \in S_i$
 - $a \in \Sigma$
 - $\exists k: \exists T_1, \dots, T_k \in Types: R \subseteq VC \times (T_1 \times \dots \times T_k) \times VC$
 - niille i , joille $1 \leq i \leq n \wedge a \notin \Sigma_i$, pätee $s'_i = s_i$
 - niille i , joille $1 \leq i \leq n \wedge a \in \Sigma_i$, on olemassa R_i siten, että $(s_i, a, R_i, s'_i) \in \Delta_i$ ja $R = \{ (\langle v_1, \dots, v_n \rangle, \langle p_1, \dots, p_k \rangle, \langle v'_1, \dots, v'_n \rangle) \mid \forall i; 1 \leq i \leq n \wedge a \in \Sigma_i: (\langle v_{e(i)}, \dots, v_{v(i)} \rangle, \langle p_1, \dots, p_k \rangle, \langle v'_{e(i)}, \dots, v'_{v(i)} \rangle) \in R_i \wedge \forall i; 1 \leq i \leq n \wedge a \notin \Sigma_i: \forall j; e(i) \leq j \leq v(i): v_j = v'_j \}$
- siis tilasiirtymä kootaan osapuolten tilasiirtymistä siten, että
 - niillä on oltava sama nimi
 - niiden relaatioissa on oltava sama määrä dataparametreja
 - jos R ja R_i :t tulkitaan väittämiksi, niin $R = (\bigwedge_{i \in \text{osapuoleet}} R_i) \wedge (\bigwedge_{i \notin \text{osapuoleet}} I_i)$ missä I_i sanoo, että ko. tilakoneen muuttujien arvot eivät muutu
- R :ssä jokainen R_i käyttää omia v_j - ja v'_j -muuttujia, mutta p_j :t (ja siis myös k) ovat kaikille R_i yhteisiä

- tilasiirtymään osallistumattomista sanotaan, että niiden tila ja muuttujien arvot säilyvät
 - ts. niiden kokonaistila säilyy
- vastaavasti kustakin osallistujasta sanotaan, että sen osuus tilasiirtymästä on sen kokonaistilasiirtymä
- jos nähdään, että $R = \emptyset$ eli $R \Leftrightarrow \text{False}$, niin tilasiirtymän saa jättää pois
 - esim. erimielisyys p_1 :n arvosta voi taata $R = \emptyset$
- τ -tilasiirtymät määritellään vastaavasti siten, että
 - yhdelle i on R_i siten, että $(s_i, \tau, R_i, s'_i) \in \Delta_i$
 - muille i pätee $s'_i = s_i$
 - R on (väittämäksi tulkittuna) $R_i \wedge$ "muiden osatilakoneiden muuttujien arvot eivät muutu"

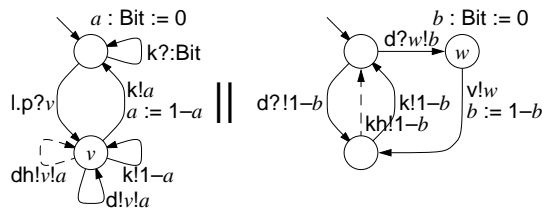
Muuttujallinen rinnankytkentä

- emme määrittele tarkasti, vaan vaadimme ainoastaan, että se on muuttujallisen synkronisen tulon alkutiloista saavutettavan osan jokin yläliikiarvo
 - saavutettavan osan tarkka laskenta voi olla liian vaikeaa
 - R voi olla $\Leftrightarrow \text{False}$ vaikei R :n kaavaesityksestä sitä helposti näe
- \Rightarrow tila voi näyttää saavutettavalta vaikka ei ole
 \Rightarrow on pakko sallia yläliikiarvon käyttö

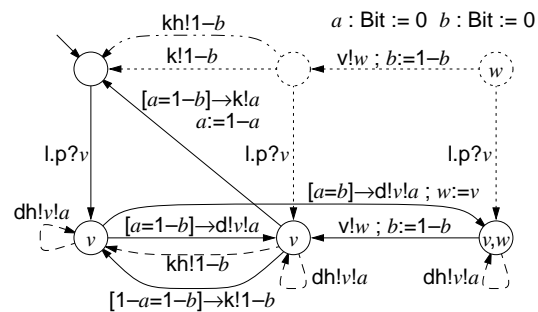
Esimerkki

- äärimmilleen yksinkertaistettu vuorottelevan bitin protokolla
 - yksinkertaistettu, ettei tulisi liikaa piirrettävää
- kanava on korvattu suoralla synkronoinnilla ja "hukkaa"-tilasiirtymillä dh, kh

- aikavalvonta ja uudelleenlähetys on korvattu silmukalla
- "hukkaa"-siirtymät ovat katkoviivoilla jatkon vuoksi
- lähetin ja vastaanotin

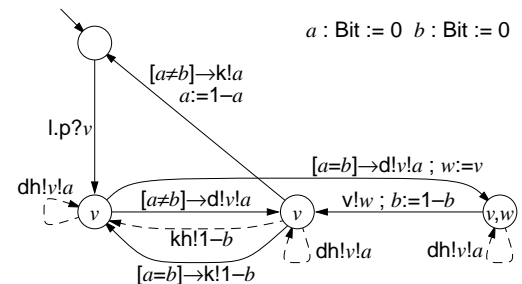


- synkroninen tulo ja rinnankytkentä



- yksi ehjäviivoitettu tilasiirtymä on tarpeeton — mikä?

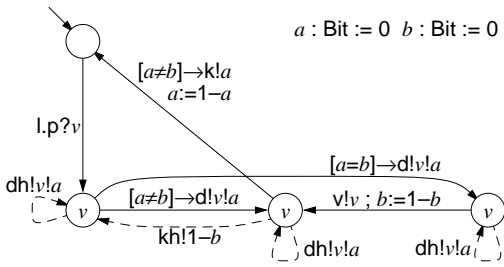
- sievennetään rinnankytkentä ottaen huomioon $a = 1-b \Leftrightarrow a \neq b$



- havainto
 - "v, w"-tilaan tullaan muualta vain jos $a = b$
 - sen paikallissilmukka ei muuta a :ta eikä b :tä

\Rightarrow "v, w"-tilassa pätee aina $a = b$
 \Rightarrow keskimmäiseen tilaan tultaessa pätee aina $a \neq b$
- senkään paikallissilmukka ei muuta a :ta eikä b :tä
 \Rightarrow siellä pätee aina $a \neq b$
 \Rightarrow "[$a=b$]->k!1-b"-kaari on tarpeeton
- harjoitustehtävä: jos katkoviivoitetut tilasiirtymät poistetaan, mikä ehjäviivoitettu käy tarpeettomaksi?

- myös muuttuja w on tarpeeton
 - silloin kun siinä on arvo, se on sama kuin v :ssä



Muuttujallisen synkronisen tulo oikeellisuus

- kokonaistilakone $(S_K, \Sigma_K, \Delta_K, \hat{S}_K, \Pi, val_K)$
 - synkronisen tulo käyttäytymisen tilat ovat muotoa $\langle s_1, \dots, s_n \rangle \langle v_1, \dots, v_V \rangle$
 - käyttäytymisen synkronisen tulo tilat ovat muotoa $\langle s_1 \langle v_{e(1)}, \dots, v_{v(1)} \rangle, \dots, s_n \langle v_{e(n)}, \dots, v_{v(n)} \rangle \rangle$
- \Rightarrow samat osat eri järjestyksessä
- osoittaaksemme käyttäytymiset isomorfisiksi samaistamme ym. kokonaistilat $\Rightarrow S_K$ täsmää
 - kokonaisuakkostot Σ_K täsmäävät, koska $(\Sigma_1 \cup \dots \cup \Sigma_n) \times U^* = (\Sigma_1 \times U^*) \cup \dots \cup (\Sigma_n \times U^*)$
 - synkronisen tulo alkukokonaistilat: ne $\langle \hat{s}_1, \hat{s}_2, \dots, \hat{s}_n \rangle \langle v_1, \dots, v_V \rangle$, joille
 - $\hat{s}_i \in \hat{S}_i$ kun $1 \leq i \leq n$, ja
 - $Init(\langle \hat{s}_1, \hat{s}_2, \dots, \hat{s}_n \rangle) \langle v_1, \dots, v_V \rangle$ pätee, ts. $Init(\hat{s}_i) \langle v_{e(i)}, \dots, v_{v(i)} \rangle$ pätee aina kun $1 \leq i \leq n$

- alkukokonaistilojen synkroninen tulo: ne $\langle \hat{s}_1 \langle v_{e(1)}, \dots, v_{v(1)} \rangle, \hat{s}_2 \langle v_{e(2)}, \dots, v_{v(2)} \rangle, \dots, \hat{s}_n \langle v_{e(n)}, \dots, v_{v(n)} \rangle \rangle$ joille $\hat{s}_i \in \hat{S}_i$ ja $Init(\hat{s}_i) \langle v_{e(i)}, \dots, v_{v(i)} \rangle$ pätee kun $1 \leq i \leq n$
- $\Rightarrow \hat{S}_K$ täsmää
- kokonaistilapropositioiksi kumpikin laskutapa antaa $\Pi_1 \cup \dots \cup \Pi_n$
 - synkronisen tulo kokonaistilakoneen val on $\{ (\pi, \langle s_1, s_2, \dots, s_n \rangle, \langle v_1, v_2, \dots, v_V \rangle) \in \Pi \times S \times VC \mid \exists i; 1 \leq i \leq n: (\pi, s_i, \langle v_{e(i)}, \dots, v_{v(i)} \rangle) \in val_i \}$
 - osien kokonaistilakoneiden val_i -ien synkroninen tulo on $\{ (\pi, \langle s_1 \langle v_{e(1)}, \dots, v_{v(1)} \rangle, \dots, s_n \langle v_{e(n)}, \dots, v_{v(n)} \rangle \rangle) \in \Pi \times (S \times VC) \mid \exists i; 1 \leq i \leq n: (\pi, s_i \langle v_{e(i)}, \dots, v_{v(i)} \rangle) \in val_i \}$
- \Rightarrow täsmää, kun otetaan tilojen vastaavuus huomioon
- synkronisella tulolla on näkyvä kokonaistilasiirtymä $\langle s_1, \dots, s_n \rangle \langle v_1, \dots, v_V \rangle -a \langle p_1, \dots, p_k \rangle \rightarrow \langle s'_1, \dots, s'_n \rangle \langle v'_1, \dots, v'_V \rangle \Leftrightarrow$ synkronisella tulolla on $(\langle s_1, \dots, s_n \rangle, a, R, \langle s'_1, \dots, s'_n \rangle)$ -kaari jolle pätee $R(\langle v_1, \dots, v_V \rangle, \langle p_1, \dots, p_k \rangle, \langle v'_1, \dots, v'_V \rangle)$
 - \Leftrightarrow jokaisella osallistuvalla osatilakoneella on (s_i, a, R_i, s'_i) -kaari ja sen omille muuttujille pätee $R_i(\langle v_{e(i)}, \dots, v_{v(i)} \rangle, \langle p_1, \dots, p_k \rangle, \langle v'_{e(i)}, \dots, v'_{v(i)} \rangle)$ ja muiden osatilakon. tila ja muuttujat eivät muutu

- \Leftrightarrow jokaisella osallistuvan osatilakoneen käyt.sellä on $s_i \langle v_{e(i)}, \dots, v_{v(i)} \rangle -a \langle p_1, \dots, p_k \rangle \rightarrow s'_i \langle v'_{e(i)}, \dots, v'_{v(i)} \rangle$ (yhteinen a ja p_j :t, paikalliset s_i, s'_i, v_j :t ja v'_j :t) ja muiden osatilakon. tila ja muuttujat eivät muutu
- \Leftrightarrow käyttäytymisen synkronisella tulolla on $\langle s_1 \langle v_{e(1)}, \dots, v_{v(1)} \rangle, \dots, s_n \langle v_{e(n)}, \dots, v_{v(n)} \rangle \rangle -a \langle p_1, \dots, p_k \rangle \rightarrow \langle s'_1 \langle v'_{e(1)}, \dots, v'_{v(1)} \rangle, \dots, s'_n \langle v'_{e(n)}, \dots, v'_{v(n)} \rangle \rangle$
- näkymättömät tilasiirtymät samaan tapaan $\Rightarrow \Delta_K$ täsmää

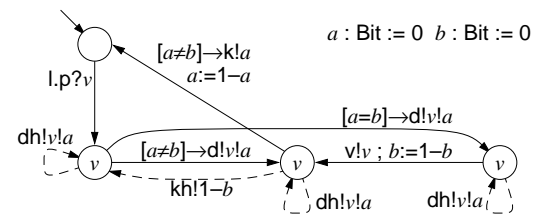
Muuttujallinen kätöntä

- paljon rinnankytkentää helpompi
 - vain Σ ja Δ muuttuvat
- nytkin oletamme tapahtumanimille yksikäsitteisyys ehdon: jos $a \neq b$, niin $(\{a\} \times U^*) \cap (\{b\} \times U^*) = \emptyset$
 - ts. eri tapahtumat eivät voi muuttua saman nimisiksi jos niille annetaan sopivat parametrit
- kätettävät $\{a_1, \dots, a_n\}$ poistetaan Σ :sta
 - $\{a_1, \dots, a_n\} \times U^*$ poistuu Σ_K :sta
- kätettäviä a vastaavat tilasiirtymät (s, a, R, s') muutetaan muotoon (s, τ, R', s') , missä $R'(\langle v_1, \dots, v_V \rangle, \langle v'_1, \dots, v'_V \rangle) \Leftrightarrow \exists p_1, \dots, p_k: R(\langle v_1, \dots, v_V \rangle, \langle p_1, \dots, p_k \rangle, \langle v'_1, \dots, v'_V \rangle)$
- oikeellisuus: kokonaistilakone(**hide** a_1, \dots, a_n **in** TK) = **hide** $\{a_1, \dots, a_n\} \times U^*$ **in** kokonaistilakone(TK)

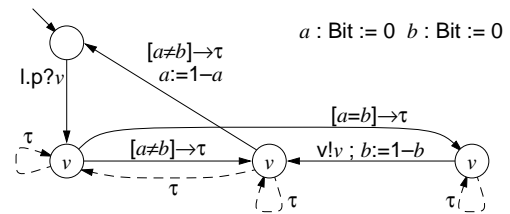
Moniuidelleennimeäminen analogisesti

Esimerkki jatkuu

- äskeinen rinnankytkentä



- kätetään d, k, dh ja kh



- hävitetään b jakamalla tilat " $a = b$ " ja " $a \neq b$ "-versioihin

