1 SAAVUTETTAVUUSANALYYSI

- Seuraavassa tarkastellaan eräitä menetelmiä jolla saavutettavusgraafin kokoa voidaan pienentää.
- Tämä tarkoittaa, että osa informaatiosta häviää.
- Seuraavassa esitellään kaksi menetelmää:
  1. Symmetriamenetelmä
  2. Itsepäiset joukot
- Molemmat menetelmät voivat nopeuttaa saavutettavuusgraafin generointia, mutta eivät kuitenkaan ratkaise generoinnin ominaislaatuun kuuluvaa kompleksisuutta sinänsä.
- Menetelmät ovat oleellisesti heuristiikkoja, jotka hävittävät osan verkkoon tallettavasta, halutun analysointituloksen kannalta epäolennaisesta, informaatiosta.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

SAAVUTETTAVUUSANALYYSI

- Saavutettavuusgraafin generointialgoritmissa on kaksi laskennallista kompleksisuutta aiheuttavaa kohtaa:
  - rivillä 6 oleva **for**-looppi, ja
  - rivillä 8 oleva $M' \notin G$ testi.

REACHABILITY-GRAPH($N :< S, T; F, K, M_0, W >$)

```
 1  G : graph; Marking : stack;
 2  G ← ∅;
 3  Markings ← M_0;
 4  while Markings ≠ ∅
 5  do M ← pop(Markings);
 6      for t in enabled(M)
 7      do M' ← fire(t, M);
 8          if M' ∉ G
 9              then push(M', Markings);
10          G ← G ∪ {< M, t, M' >};
```

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

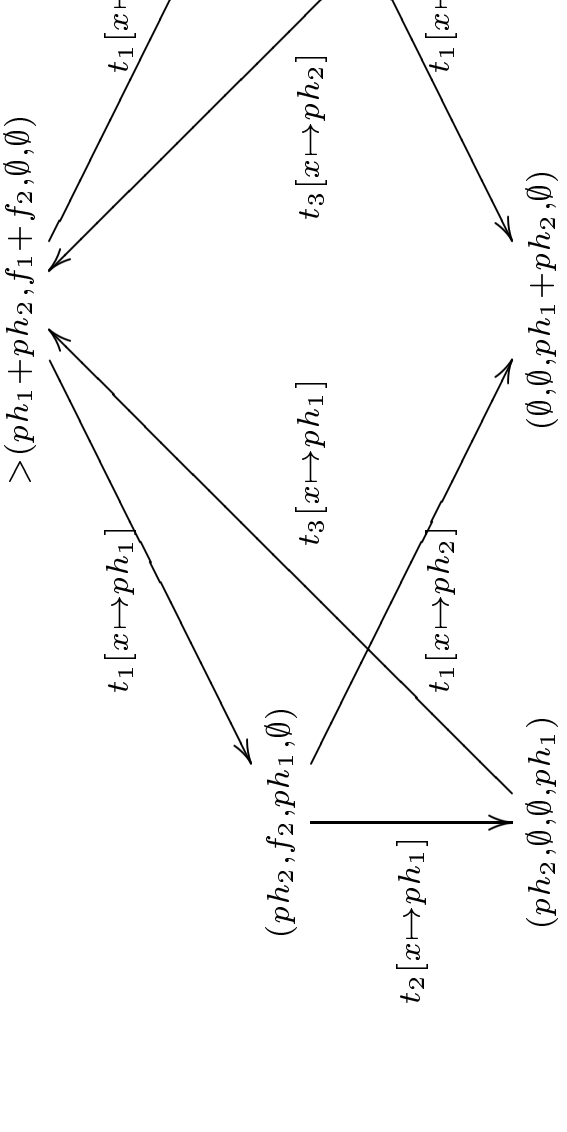
Saavutettavuusanalyysi

- Symmetriamenetelmä pyrkii pienentämään saavutettavuusgraafin kokoa tallettamalla vain merkintöjen ekvivalenssiluokkia. Näin ollen talletettavien merkintöjen joukko pienenee.
- Itsepäisten joukkojen menetelmä pyrkii pienentämään saavutettavuusgraafia laukaisemalla rivillä 6 $t \in enabled(M)$ vain transitiot jotka kuuluvat joukkon $stubborn(M) \subseteq enabled(M)$. Näin ollen saavutettavuusgraafin haarautumista rajoitetaan, mikä vuorostaan pienentää merkintöjen määrää.
- Molemmissa menetelmissä kuitenkin periaatteessa vaihdetaan tilaa ajaksi, koska sekä symmetristen merkintöjen, että itsepäisten transitiojoukkojen laskeminen voi olla kompleksista.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

SYMMETRIAMENETELMÄ

- Tarkastellaan seuraavaa saavutettavuusgraafia:



- Graafi on selvästi symmetrinen alkumerkinnän ja lukkiuman välisen viivan suhteen.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

SYMMETRIAMENETELMÄ

- P/T-verkon *symmetria* $\sigma : S \cup T \to S \cup T$ on *automorfimsi* (P/T-verkon isomorfismi itsensä kanssa), joka kunnoittaa kaaripainoja:

$$W(< a, a' >) = W(< \sigma(a), \sigma(a') >)$$

  ja kapasiteetteja.
- Symmetria σ voidaan laajentaa kuvaukseksi merkinnöille $\sigma : (S \to \mathbb{N}) \to (\mathbb{S} \to \mathbb{N})$ määrittelemällä:

$$(\sigma(M))(s) = M(\sigma^{-1}(s))$$

- Tästä nähdään helposti, että

$$M[t > M' \equiv \sigma(M)[\sigma(t) > \sigma(M')$$

  kaikille $M, M' \in \mathbb{M}, t \in T$.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

Symmetriamenetelmä

- Joukko symmetrioita Σ, jolle pätee:
  1. $id \in \Sigma$, missä $id(M) = M$,
  2. $\sigma \in \Sigma \Rightarrow \sigma^{-1} \in \Sigma$, ja
  3. $\sigma, \sigma' \in \Sigma \Rightarrow \sigma' \circ \sigma \in \Sigma$

  muodostaa *symmetria-ryhmän*
- Symmetria-ryhmä määrittelee ekvivalenssirelaation merkinnöille:

$$M \equiv M' \stackrel{\text{def}}{=} (\exists \sigma \in \Sigma : M = \sigma(M'))$$

- Merkintää M kutsutaan *symmetriseksi* joss

$$\forall \sigma \in \Sigma : M = \sigma(M)$$

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

SYMMETRIAMENETELMÄ

SYMMETRIC-RG($N : P/T - Net, S : SymmetryGroup$)

```
 1  G : graph; Marking : stack;
 2  G ← ∅;
 3  Markings ← M_0;
 4  while Markings ≠ ∅
 5  do M ← pop(Markings);
 6      for t in enabled(M)
 7      do M' ← fire(t, M);
 8          if ∃M'' ∈ G ∃σ ∈ S : M'' = σ(M')
 9              then
10                  G ← G ∪ {< M, t, M'' >};
11              else
12                  G ← G ∪ {< M, t, M' >};
13                  push(M', Markings);
```

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

Symmetriamenetelmä

- Jos alkumerkintä on symmetrinen säilyttää menetelmä saavutettavat merkinnät:

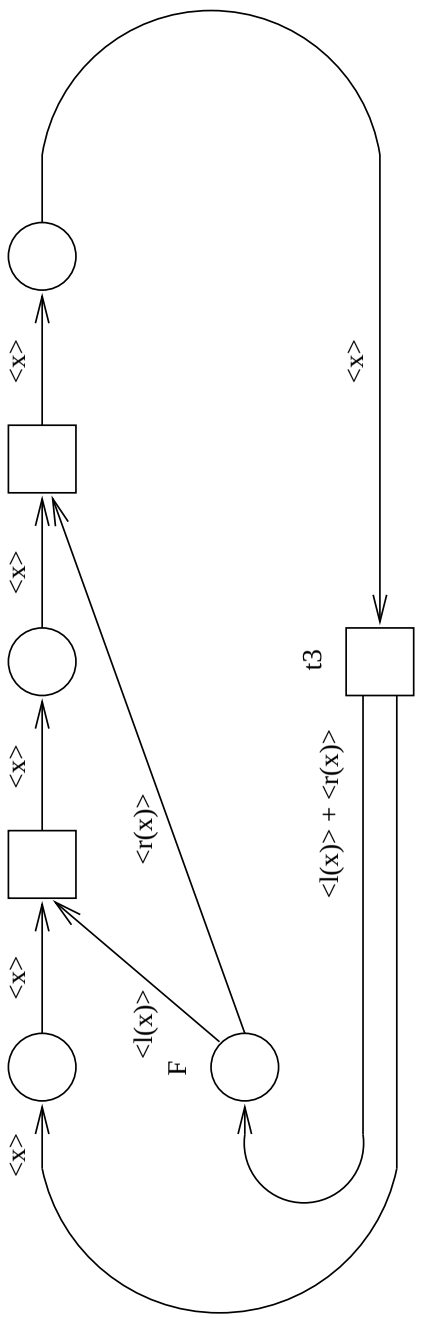
$$M_0[> M \Leftrightarrow \exists M'(M_0[> M' \wedge M' \cong M)$$

- Yksittäisen transition elävyys ei välttämättä säily.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

SYMMETRIAMENETELMÄ

- Tarkastellaan filosofiesimerkkiä kahdella filosofilla:



- Alkumerkintä: $P \mapsto \{\langle ph_1 \rangle + \langle ph_2 \rangle\}, F \mapsto \{\langle f_1 \rangle + \langle f_2 \rangle\}$
- Funktiot: $l(ph_i) = f_i,\ r(ph_i) = f_{(i\ mod\ 2)+1}$

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

- Yksi symmetria on:

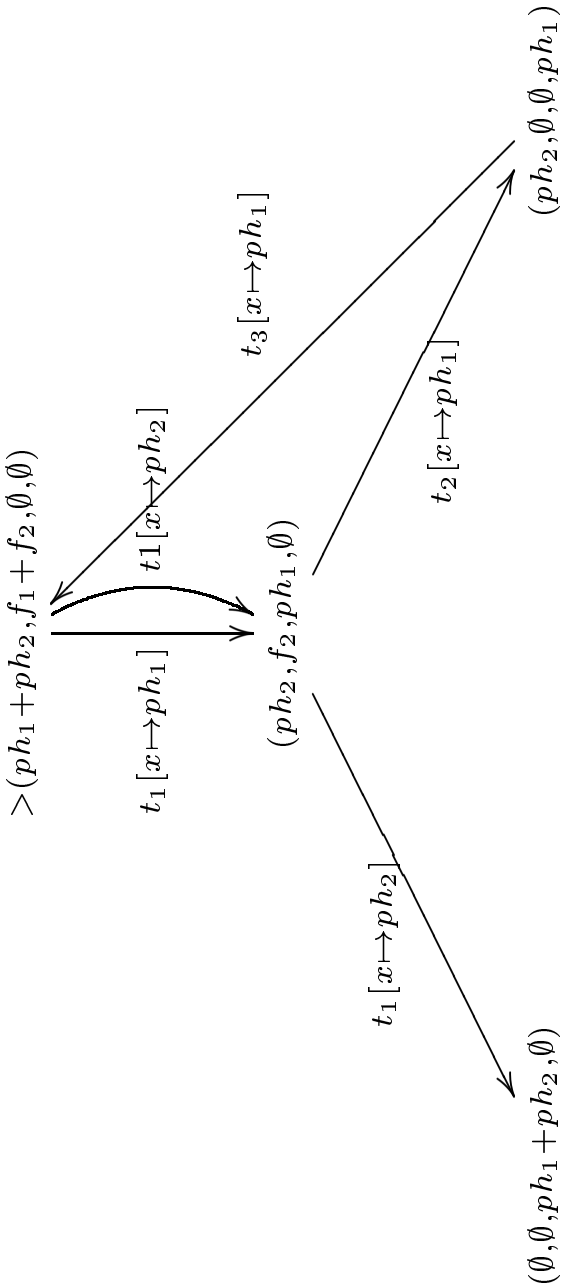
$$\sigma = \begin{pmatrix} P(ph_1) & \mapsto & P(ph_2) \\ P(ph_2) & \mapsto & P(ph_1) \\ F(f_1) & \mapsto & F(f_2) \\ F(f_2) & \mapsto & F(f_1) \\ W(ph_1) & \mapsto & W(ph_2) \\ W(ph_2) & \mapsto & W(ph_1) \\ E(ph_1) & \mapsto & E(ph_2) \\ E(ph_2) & \mapsto & E(ph_1) \end{pmatrix}$$

- Itse asiassa tämä symmetria *generoi* kaikki muut symmetriat: esim $id = \sigma \circ \sigma$

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

SYMMETRIAMENETELMÄ

- Redusoitu saavutettavuusgraafi:



©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

Symmetriamenetelmä

- On selvää, että edellä esitetty algoritmi pienentää saavutettavuusgraafin kokoa.
- Yllättävämpää on, että algoritmi saattaa usein olla nopeampi kuin perinteinen.
- Vaikka jokaisella kierroksella joudutaan käsittelemään symmetrioita, saattaa symmetrioden ansiosta kokonaisia saavutettavuusgraafin alipuita jäädä tutkimatta.
- Menetelmän pullonkaula on symmetrioiden löytäminen:
  - Käyttäjä antaa työkalulle.
  - Työkalu laskee itse: Tämä on kuitenkin huomattavan aikaavievä prosessi. Olellisesti kyseessä on graafin automorfismiryhmän laskenta ja tämä ongelma on NP-täydellinen.
- `PROD`:lla on mahdollista generoida symmetrisesti redusoitu saavutettavuusgraafi. `PROD` laskee symmetriat itse.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

Itsepäiset joukot

- Itsepäisten joukkojen menetelmän ideana on, että rivillä 6 pyritään tarkastelemaan vireillä olevien transitioiden joukon osajoukkoja. Tämä pienentää saavutettavuusgraafin haarautumista.
- Menetelmä perustuu seuraavaan ideaan:

  Pyritään jakamaan transitioiden joukko T kahtia joukkoihin T_s ja $T \backslash T_s$ siten, että mikään joukon T_s transitio ei estä joukon $T \backslash T_s$ transition laukeamista.

  Oletetaan, että seuraava diagrammi kommutoi

$$\begin{array}{ccc} M & \xrightarrow{\sigma} & M' \\ \downarrow{\scriptstyle t} & & \downarrow{\scriptstyle t} \\ M'' & \xrightarrow{\sigma} & M''' \end{array}$$

  missä $t \in T_s$ ja kaikki $t' \in \sigma : t' \in T \backslash T_s$.

  Sellaisen ominaisuuden osalta, joka pätee merkinnässä M''' on "yhdentekevää" löydetäänkö se polkua $t\sigma$ vai σt pitkin.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

Itsepäiset joukot

Definition 1 Olkoon N P/T-verkko, M jokin sen merkintä. Tällöin T_s on *itsepäinen* merkinnässä M, joss:

$$T_s \subseteq T \wedge (\exists t \in T_s \wedge M[t>),$$

$$\forall t \in T_s \quad : \quad M[t> \Rightarrow ({}^\bullet t)^\bullet \subseteq T_s,$$

$$\forall t \in T_s \quad : \quad \neg M[t> \Rightarrow (\exists s \in {}^\bullet t \wedge M(s) \leq W(s,t) \wedge {}^\bullet s \subseteq T_s)$$

- Kaikkien transitioiden joukko T on triviaalisti itsepäinen.
- Merkinnästä M itsepäinen joukko voidaan (tehottomasti) laskea seuraavasti:
  1. Valitaan transitio t joka on vireessä merkinnässä M. Olkoon $T_s = \{t\}$.
  2. Valitaan transitio $t' \in T_s$ (jota ei ole vielä tutkittu):
     (a) $M[t>$: Ehto 2. toteutuu kun asetetaan $T_s = T_s \cup ({}^\bullet t)^\bullet$. Palataan kohtaan 2.
     (b) $\neg M[t>$: Ehdon 3. mukaan etsitään paikka s, jonka takia t ei ole vireessä, ja asetetaan $T_s = T_s \cup {}^\bullet s$. Palataan kohtaan 2.
- Algoritmi päättyy kun joukko T_s ei enää muutu iteraatiokierroksen aikana.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

Itsepäiset joukot

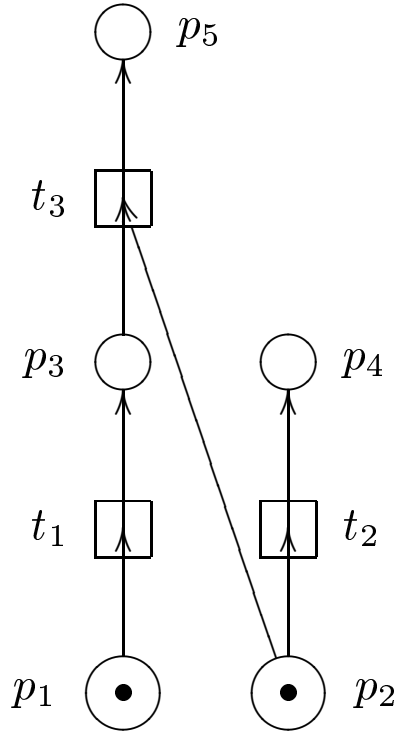
Stubborn-Reachability-Graph($N :< S, T; F, K, M_0, W >$)

```
 1  G : graph; Marking : stack;
 2  G ← ∅;
 3  Markings ← M_0;
 4  while Markings ≠ ∅
 5  do M ← pop(Markings);
 6      for t in stubborn(M)
 7      do M' ← fire(t, M);
 8          if M' ∉ G
 9             then push(M', Markings);
10          G ← G ∪ {< M, t, M' >};
```

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

Itsepäiset joukot

- Tarkastellan seuraavaa esimerkkiä:



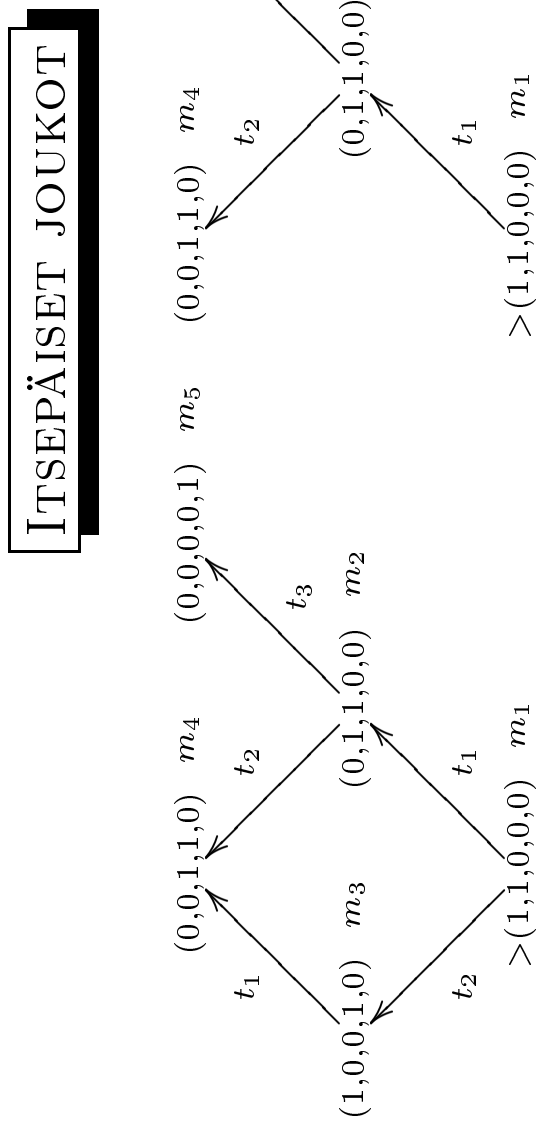
  - Valitaan t_2:

$$M[t_2 > \quad \Rightarrow \quad ({}^\bullet t_2)^\bullet = \{t_2, t_3\} \subseteq T_s$$
$$\neg M[t_3 > \quad \Rightarrow \quad {}^\bullet(p_3) = \{t_1\} \subseteq T_s$$

    saadaan siis, $T_s = \{t_1, t_2, t_3\} = T$.
  - Jos valitaan t_1 saadaan ehdosta 2. $T_s = \{t_1\}$

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

Itsepäiset joukot



©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

Itsepäiset joukot

- Itsepäisen joukon koko on hyvin riippuvainen siitä mistä vireessä olevasta transitiosta laskenta aloitetaan.
- Lisäksi ehdon 3. mukaisen paikan s valinta vaikutta itsepäisen joukon kokoon.
- Pienin mahdollinen itsepäinen joukko saadaan käyttämällä *poistoalgoritmia*. Tällöin aloitetaan kaikkien transitioiden joukosta, poistetaan yksi transitio ja sen jälkeen poistetaan transitoita kunnes joukko on jälleen itsepäinen.
- Edellä esitettyä ns. *inkrementaalista* algoritmia voidaan parantaa käyttämällä hyväksi Tarjanin vahvasti kytkettyjen komponenttien algoritmia.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001

Itsepäiset joukot

- Perusversio säilyttää kaikki järjestelmän saavutettavuusgraafin lukkiumat.
- On olemassa itsepäisten joukkojen menetelmän muunnos, jolle voidaan osoittaa, että menetelmä säilyttää kaikki sellaiset lineaarisen temporaalilogiikan kaavat jotka eivät sisällä `next` ($\bigcirc$) operaattoria.
- `PROD` tukee itsepäisten joukkojen menetelmää eri laskenta-algoritmeilla.
- Kätevin ja tehokkain tapa hyödyntää itsepäisiä joukkoja on käyttää testereitä ja ajaa saavutettavuusgraafigeneraattori optiolla `-C`

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2001