# PREDIKAATTILOGIIKKA

1. Predikaattilogiikan kieli
2. Predikaattilogiikan semantiikka
3. Semanttiset taulut predikaattilogiikalle
4. Normaalimuodot
5. Tietämyksen esittämisestä
6. Herbrandin teoreema
7. Unifikaatio
8. Resoluutiosääntö ja -todistukset
9. Ohjelmien oikeellisuustarkastelut

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



# 1 Predikaattilogiikan kieli

- Motivaatio
- Predikaattilogiikan aakkosto
- Kielen määritelmä
- Kvanttoreihin liittyviä määritelmiä
- Lauseiden muodostaminen

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 1.1 Motivaatio

- Lauselogiikka on useisiin tarkoituksiin liian yksinkertainen: olkoon $A =$ "$a$ on viallinen", $B =$ "$b$ on viallinen", $C =$ "$c$ on viallinen".

  Tällöin "kaikki ovat viallisia" $= A \wedge B \wedge C$ ja
  "jokin on viallinen" $= A \vee B \vee C$.

- Erityisesti objektien välisten suhteiden kuvaaminen on hankalaa (tarvitaan paljon lauseita, jotka ovat muodoltaan samankaltaisia).

**Esimerkki.**

"Jos $x$ on isompi kuin $y$ ja $y$ on isompi kuin $z$,
niin $x$ on isompi kuin $z$".

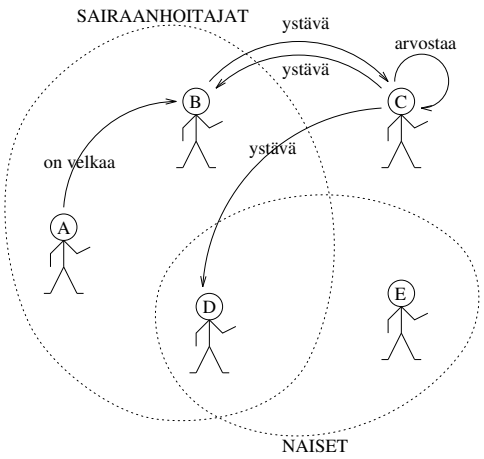
$C_d =$ "$c$ on isompi kuin $d$", $D_e =$ "$d$ on isompi kuin $e$", ...

$(C_d \wedge D_e \rightarrow C_e) \wedge (C_e \wedge E_d \rightarrow C_d) \wedge (D_e \wedge E_c \rightarrow D_c) \wedge \cdots$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Alla on kuvattu joitain henkilöiden välisiä suhteita.




© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## 1.2 Predikaattilogiikan aakkosto

Predikaattilogiikan kielessä $\mathcal{L}$ käytetään seuraavia symboleja:

- Muuttujasymbolit $\mathcal{V} = \{x, y, z, \ldots\}$
- Vakiosymbolit $\mathcal{C} = \{a, b, c, \ldots\}$
- Funktiosymbolit $\mathcal{F} = \{f, g, h, \ldots\}$
- Predikaattisymbolit $\mathcal{P} = \{=, P, Q, R, \ldots\}$
- Lauselogiikan konnektiivit $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- Kvanttorisymbolit $\exists, \forall$
- Sulut () ja pilkku ,

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Määritelmä.**

- Jokaisella funktiosymbolilla $f \in \mathcal{F}$ on *paikkaluku* $n > 0$ (joka määrää $f$:n argumenttien lukumäärän).
- Vastaavasti predikaattisymboleilla $P \in \mathcal{P}$ on paikkaluvut $n \geq 0$.
- Määritellään $\mathcal{F}_n = \{f \in \mathcal{F} \mid f\text{:n paikkaluku on } n\}$, kun $n > 0$, ja $\mathcal{P}_n = \{P \in \mathcal{P} \mid P\text{:n paikkaluku on } n\}$, kun $n \geq 0$.
- Täten $\mathcal{F} = \cup\{\mathcal{F}_n \mid n > 0\}$ ja $\mathcal{P} = \cup\{\mathcal{P}_n \mid n \geq 0\}$.

**Huomioita.**

- Yhtäsuuruuspredikaatti $= \in \mathcal{P}$ on kaksipaikkainen, eli $= \in \mathcal{P}_2$.
- Vakiosymbolien joukko $\mathcal{C}$ voitaisiin vaihtoehtoisesti määritellä 0-paikkaisten funktiosymbolien joukkona $\mathcal{F}_0$.
- Joukon $\mathcal{P}_0$ symbolit vastaavat lauselogiikan atomisia lauseita.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 1.3 Kielen määritelmä

Predikaattilogiikan kielen $\mathcal{L}$ määritelmä on kolmitasoinen: ensin määritellään termit, sitten atomikaavat ja lopulta varsinaiset kaavat.

**Määritelmä.**

1. Jokainen muuttujasymboli $v \in \mathcal{V}$ on *termi*.
2. Jokainen vakiosymboli $c \in \mathcal{C}$ on *termi*.
3. Jos $f \in \mathcal{F}_n$ on $n$-paikkainen funktiosymboli ja $t_1, \ldots, t_n$ ovat termejä, niin myös $f(t_1, \ldots, t_n)$ on *termi*.
4. Muita termejä ei ole.

**Esimerkki.** $x$, $c$, $f(x)$, $f(f(f(f(f(x)))))$, $g(f(x), g(f(x), g(x, c)))$.

**Määritelmä.** Termi, jossa ei esiinny muuttujia, on *muuttujaton termi* (engl. *ground term*).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Määritelmä.**

1. Jos $t_1$ ja $t_2$ ovat termejä, niin $t_1 = t_2$ on *atomikaava*.
2. Jos $P \in \mathcal{P}_0$ on 0-paikkainen predikaattisymboli, niin $P$ on *atomikaava*.
3. Jos $P \in \mathcal{P}_n$ on $n$-paikkainen predikaattisymboli (missä $n > 0$) ja $t_1, \ldots, t_n$ ovat termejä, niin $P(t_1, \ldots, t_n)$ on *atomikaava*.
4. Muita atomikaavoja ei ole.

**Esimerkki.** Atomikaavoja ovat mm.

$$P(x_1), \quad Q, \quad x_1 = x_2, \quad g(x_1, x_2) = f(f(c_1)),$$
$$R(c_1, x_1, y_1) \quad \text{ja} \quad S(x_1, c_1, f(x), h(f(x_1), c_1, x_1), x_2, x_2, x_3),$$

mutta esim. $f(R(x), c)$ ei ole atomikaava (eikä edes termi).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Määritelmä.**

1. Jokainen atomikaava $\phi$ on *kaava*.
2. Jos $\phi$ ja $\psi$ ovat kaavoja ja $x$ on muuttuja, niin myös
   $$(\neg\phi),\ (\phi\vee\psi),\ (\phi\wedge\psi),\ (\phi\rightarrow\psi),\ (\phi\leftrightarrow\psi),\ (\forall x\phi),\ (\exists x\phi)$$
   ovat *kaavoja*.
3. Muita kaavoja ei ole.

**Esimerkki.** Predikaattilogiikan kaavoja: $P(c)$, $(\forall x(P(x)\rightarrow Q(x)))$, $(\forall x(P(x)\vee(\exists yQ(x,y))))$, $(\exists x(\forall y(\forall zP(x,y,z))))$.

Symbolijoukkoihin $\mathcal{V},\mathcal{C},\mathcal{F}$ ja $\mathcal{P}$ perustuva predikaattilogiikan *kieli* $\mathcal{L}$ määritellään edellä annetuilla periaatteilla muodostettavissa olevien kaavojen joukkona.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Sopimukset sulkeiden käytöstä**

- Konnektiivien presedenssiluokat ovat seuraavat:
  1. $\neg$, $\forall v$ ja $\exists v$ (missä $v\in\mathcal{V}$) ovat vahvimmat konnektiivit.
  2. $\vee$ ja $\wedge$ ovat näitä heikompia, mutta vahvempia kuin $\rightarrow$ ja $\leftrightarrow$.
  3. $\rightarrow$ ja $\leftrightarrow$ ovat heikoimmat konnektiivit.
- Lauselogiikan yhteydessä käyttöönotettuja periaatteita sulkeiden vähentämiseksi käytetään myös kaavoja kirjoitettaessa.

**Esimerkki.** Täten kaava

$$(\exists x(\forall y((\exists z(P(x,z)\wedge P(z,y)))\rightarrow((Q(x)\vee Q(y))\vee R(x,y)))))$$

voidaan kirjoittaa selkeämmin

$$\exists x\forall y(\exists z(P(x,z)\wedge P(z,y))\rightarrow Q(x)\vee Q(y)\vee R(x,y)).$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

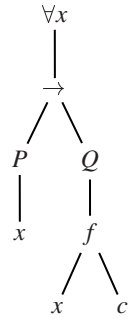


**Kaavojen jäsennyspuut**

Predikaattilogiikan kaavoilla on yksikäsitteinen jäsennyspuu.

Kaavan $\forall x(P(x)\rightarrow Q(f(x,c)))$ jäsennyspuu on annettu oikealla.

Kyseinen kaava on muodoltaan *universaalisti kvantifioitu kaava*.




**Huomio.** Jäsennyspuun juuressa oleva konnektiivi määrää edelleen, mitä muotoa lause on. Esimerkiksi $\exists xP(x)\rightarrow\forall xP(x)$ on muodoltaan *implikaatio*, kun taas $\exists x(P(x)\rightarrow\forall yQ(y))$ on muodoltaan *eksistentiaalisesti kvantifioitu kaava*.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 1.4 Kvanttoreihin liittyviä määritelmiä

Kaavan *alikaavat* määräytyvät seuraavasti:

- Atomisen kaavan $\psi$ ainoa alikaava on $\psi$ itse.
- Kaavan $\exists x\psi$ ($\forall x\psi$) alikaavat ovat $\exists x\psi$ ($\forall x\psi$) ja $\psi$:n alikaavat.
- Lauselogiikan konnektiivit $(\neg,\rightarrow,\leftrightarrow,\vee,\wedge)$ käsitellään vastaavalla tavalla (vrt. alilauseiden määritelmä lauselogiikan tapauksessa).

**Esimerkki.** Kaavan $\phi=\exists xA(x,x)\wedge\exists x(S(x)\wedge N(x))$ alikaavoja ovat

$$\phi,\ \exists xA(x,x),\ \exists x(S(x)\wedge N(x)),\ A(x,x),\ S(x)\wedge N(x),\ S(x)\text{ ja }N(x).$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

### Vapaat ja sidotut muuttujat kaavassa

**Määritelmä.** Olkoot $\exists x\psi$ ja $\forall x\phi$ predikaattilogiikan kaavoja. Alikaava $\psi$ on kvanttorin $\exists x$ *vaikutusalue* kaavassa $\exists x\psi$. Vastaavasti alikaava $\phi$ on kvanttorin $\forall x$ vaikutusalue kaavassa $\forall x\phi$.

**Esimerkki.**

$$\forall x(P(x) \rightarrow \exists y Q(x,y))$$

$$\exists x(Q(x) \leftrightarrow \forall x P(x,z)) \quad \vee \exists x R(x)$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Määritelmä.** Muuttujan $x$ esiintymä kaavassa $\phi$ on *sidottu*, jos se sijaitsee kvanttorin $\forall x$ (tai $\exists x$) vaikutusalueessa. Kvanttorisymbolia seuraava muuttujaesiintymä on sidottu.

Jos muuttujan $x$ esiintymä ei ole sidottu, niin $x$:n esiintymä on *vapaa*.

Muuttuja $x$ *esiintyy vapaana* $\phi$:ssä, jos sillä on vapaa esiintymä $\phi$:ssä.

**Esimerkki.** Tarkastellaan muuttujaesiintymiä seuraavassa lauseessa:

$$\forall x(P(x,y,z) \vee \exists y(Q(y) \rightarrow R(x,z)))$$

(sid., sid., vap., vap., sid., sid., sid., vap.)

**Esimerkki.** Kaavan $\exists x(P(x) \wedge \exists x Q(x))$ muuttujaesiintymät ovat sidotut.

**Määritelmä.** Kaavaa $\phi$ kutsutaan *lauseeksi*, jos siinä ei ole vapaita muuttujaesiintymiä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Muuttujattoman termin sijoittaminen kaavaan

**Määritelmä.** Olkoon $\phi(x)$ kaava, jossa muuttuja $x$ mahdollisesti esiintyy vapaana ja $t$ muuttujaton termi. Kaavalla $\phi(t)$ tarkoitetaan kaavaa $\phi(x)$, jossa jokainen muuttujan $x$ vapaa esiintymä on korvattu termillä $t$.

**Esimerkki.**

1. Olkoon $\phi(y) = \exists x(P(x,y) \vee Q(y,x))$.
   - Sijoittamalla muuttujattomat termit $c$ ja $f(f(d))$ kaavaan $\phi(y)$ saadaan $\phi(c) = \exists x(P(x,c) \vee Q(c,x))$ ja
   $$\phi(f(f(d))) = \exists x(P(x,f(f(d))) \vee Q(f(f(d)),x)).$$
2. Olkoon $\psi(x) = \exists x P(x) \wedge Q(x)$.
   - Sijoittamalla vakio $c$ saadaan $\psi(c) = \exists x P(x) \wedge Q(c)$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 1.5 Lauseiden muodostaminen

1. Tunnistetaan kuvattavaan järjestelmään liittyvät objektit:
   - Otetaan käyttöön vakiosymboli jokaiselle objektille, johon on tarve viitata erikseen, eli *nimetään* tarvittavat objektit.
   - Mikäli objektien välillä on funktionaalisia riippuvuuksia, otetaan käyttöön vastaavat funktiosymbolit.
2. Tutkitaan millaisia relaatioita objektien välillä on ja otetaan käyttöön näitä vastaavat predikaattisymbolit.
3. Kuvataan relaatioiden väliset riippuvuudet kirjoittamalla niille määritelmät predikaattilogiikan lausein.

**Huomio.** Funktiosymbolin voi tarvittaessa korvata predikaattisymbolilla, mutta tällöin tällöin määritelmään tulee liittää funktionaalisuusehto.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.**

Olkoon $t =$ "tuoli", $h =$ "hattu", $s =$ "sateenvarjo" vakioita ja $P(x,y) =$ "$x$ on $y$:n päällä" kaksipaikkainen predikaatti. Tällöin:

$\neg P(s,t) =$ "sateenvarjo ei ole tuolin päällä".

$\exists x P(x,h) =$
"on olemassa $x$, joka on hatun päällä",
eli "hatun päällä on jotakin".

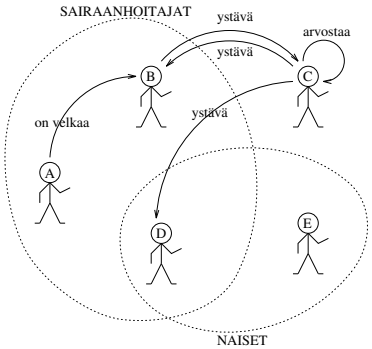
$\exists x \forall y \neg P(y,x) =$
"on olemassa $x$ siten, että mikään $y$ ei ole $x$:n päällä",
eli "jonkin päällä ei ole mitään".

$\forall x(P(x,h) \rightarrow P(x,t)) =$
"kaikille $x$: jos $x$ on hatun päällä, niin $x$ on tuolin päällä",
eli "kaikki hatun päällä olevat ovat tuolin päällä".

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Kuvataan henkilöiden välisiä suhteita predikaattilogiikalla.




$N(e) \rightarrow A(c,c)$

$\exists x \exists y(Y(x,y) \wedge Y(y,x))$

$\exists x \exists y(S(x) \wedge S(y) \wedge V(x,y))$

$\exists x A(x,x) \wedge \exists x(S(x) \wedge N(x))$

$\neg \forall x(S(x) \rightarrow N(x))$

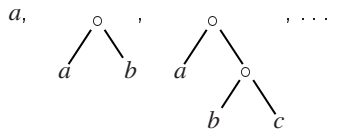
$\forall x(Y(x,c) \rightarrow V(a,x))$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Funktiosymbolit tarjoavat tavan esittää induktiivisia tietorakenteita termien avulla.

1. Luonnolliset luvut: vakiosymboli $0$ ja funktiosymboli $s \in \mathcal{F}_1$.
   - Termit $0, s(0), s(s(0)), \ldots$ vastaavat luonnollisia lukuja $0, 1, 2, \ldots$
2. Listat: vakiosymboli $e$ (tyhjä lista) ja funktiosymboli $c \in \mathcal{F}_2$.
   - Termit $e, c(a,e), c(a,c(b,e)), \ldots$ vastaavat listoja $[\,], [a], [a,b], \ldots$
3. Binääripuut: funktiosymbolit $l \in \mathcal{F}_1$ (lehtisolmut) ja $t \in \mathcal{F}_2$ (sisäsolmut).
   - Termit $l(a), t(l(a),l(b)), t(l(a),t(l(b),l(c))), \ldots$ vastaavat puita




© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Esitetään binääripuut kuten edellä funktiosymbolien $l$ ja $t$ avulla. Kirjoitetaan määritelmä seuraavalle predikaatille:

$$P(x,y) = \text{"binääripuu } x \text{ on binääripuun } y \text{ peilikuva"}.$$

Koska binääripuut muodostavat induktiivisen tietorakenteen, on luontevaa, että predikaatille $P(x,y)$ joudutaan kirjoittamaan induktiivinen/rekursiivinen määritelmä seuraavalla tavalla.

- Perustapaus (pelkästä lehtisolmusta koostuvat binääripuut):
$$\forall x(P(l(x),l(x))).$$
- Induktioaskel (monimutkaisemmat binääripuut):
$$\forall x \forall y \forall z \forall v(P(x,y) \wedge P(z,v) \rightarrow P(t(x,z),t(v,y))).$$

$\Longrightarrow$ Määritelmä kattaa kaikki binääripuut.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

# 2 Predikaattilogiikan semantiikka

- Struktuurit
- Predikaattilogiikan totuusmääritelmä
- Semanttiset peruskäsitteet
- Peruskäsitteiden väliset yhteydet

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 2.1 Struktuurit

Predikaattilogiikassa totuusjakelut korvataan struktuureilla.

**Määritelmä.** *Struktuuri* (rakenne) $\mathcal{S}$ kielelle $\mathcal{L}$ koostuu

- universumista $U$, joka on jokin *ei-tyhjä* joukko, sekä
- vakio-, muuttuja-, funktio- ja predikaattisymbolien tulkinnoista:
  1. Vakiosymbolin $c \in \mathcal{C}$ tulkintana on alkio $c^{\mathcal{S}} \in U$.
  2. Muuttujasymbolin $v \in \mathcal{V}$ tulkintana on alkio $v^{\mathcal{S}} \in U$.
  3. Funktiosymbolin $f \in \mathcal{F}_n$ tulkintana on funktio $f^{\mathcal{S}} \colon U^n \to U$.
  4. Predikaattisymbolin $P \in \mathcal{P}_n$ tulkintana on relaatio $P^{\mathcal{S}} \subseteq U^n$.

Struktuuri voidaan edelleen ymmärtää yhden asiaintilan kuvauksena.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Huomioita.**

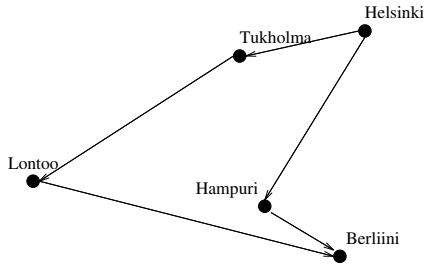
- Joukon $U^n = \overbrace{U \times \cdots \times U}^{n \text{ kpl}}$ alkiot ovat monikkoja (tai jonoja) $\langle a_1, \ldots, a_n \rangle$, missä alkiot $a_1 \in U$, $\ldots$, $a_n \in U$.
- Erikoistapaukset: $U^1 = U$ ja $U^0 = \{\langle\,\rangle\}$, missä $\langle\,\rangle$ on *tyhjä jono*.
- Kvanttoreilla $\exists v$ ja $\forall v$ tullaan jatkossa viittaamaan universumin eri alkioihin. Muuttujan $v$ arvon vaihtaminen struktuurissa $\mathcal{S}$ tapahtuu seuraavalla tavalla.

**Määritelmä.** Struktuurilla $\mathcal{S}[v \mapsto a]$ tarkoitetaan struktuuria $\mathcal{S}'$, joka on muuten sama kuin $\mathcal{S}$, mutta muuttujasymbolin $v \in \mathcal{V}$ tulkintana $v^{\mathcal{S}'}$ onkin annettu alkio $a \in U$ (alkion $v^{\mathcal{S}}$ asemesta).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.**




$$
\begin{array}{rclrcl}
U & = & \{he, tu, ha, be, lo\} & \text{Helsinki}^{\mathcal{S}} & = & he \\
\text{Tukholma}^{\mathcal{S}} & = & tu & \text{Hampuri}^{\mathcal{S}} & = & ha \\
\text{Berliini}^{\mathcal{S}} & = & be & \text{Lontoo}^{\mathcal{S}} & = & lo \\
\text{pääkaupunki}^{\mathcal{S}} & = & \{he, tu, be, lo\} \subseteq U^1 = U & & & \\
\text{lento}^{\mathcal{S}} & = & \{\langle he, tu\rangle, \langle tu, lo\rangle, \langle lo, be\rangle, \langle he, ha\rangle, \langle ha, be\rangle\} \subseteq U^2 & & &
\end{array}
$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## 2.2 Predikaattilogiikan totuusmääritelmä

### Termien tulkinta struktuurissa

**Määritelmä.** Olkoon $\mathcal{S}$ struktuuri kielelle $\mathcal{L}$ ja $U$ struktuurin $\mathcal{S}$ universumi.

- Vakio $c \in \mathcal{C}$ *nimeää* universumin $U$ alkion $c^{\mathcal{S}}$.
- Muuttuja $v \in \mathcal{V}$ *nimeää* universumin $U$ alkion $v^{\mathcal{S}}$.
- Jos termit $t_1, \ldots, t_n$ nimeävät universumin $U$ alkiot $t_1^{\mathcal{S}}, \ldots, t_n^{\mathcal{S}}$ ja $f \in \mathcal{F}_n$, niin termi $f(t_1, \ldots, t_n)$ *nimeää* universumin $U$ alkion $f^{\mathcal{S}}(t_1^{\mathcal{S}}, \ldots, t_n^{\mathcal{S}})$.

Näin voimme viitata kielen $\mathcal{L}$ termeillä universumin $U$ alkioihin (kunhan vakio-, muuttuja-, ja funktiosymbolien tulkinnat on annettu).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Tarkastellaan vakiosymbolia $c \in \mathcal{C}$ ja funktiosymboleja $f \in \mathcal{F}_1$ ja $g \in \mathcal{F}_2$. Olkoon struktuurin $\mathcal{S}$ universumina $U$ luonnollisten lukujen joukko $0, 1, 2, \ldots$. Valitaan em. symbolien tulkinnat seuraavasti:

$$\begin{aligned} c^{\mathcal{S}} &= 0, \\ f^{\mathcal{S}} &= \text{seuraajafunktio, eli } f^{\mathcal{S}}(n) = n+1, \text{ ja} \\ g^{\mathcal{S}} &= \text{summafunktio, eli } g^{\mathcal{S}}(n,m) = n+m. \end{aligned}$$

| | |
|---|---|
| Tällöin $c$ | nimeää alkion $0$, |
| $f(c)$ | nimeää alkion $1$, |
| $f^n(c) = \underbrace{f(f(\ldots f}_{n \text{ kpl}}(c)\ldots))$ | nimeää alkion $n$ ja |
| $g(f(c), f(f(c)))$ | nimeää alkion $3$. |

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Kaavojen totuusarvojen laskeminen struktuurissa

Olkoon $\mathcal{S}$ struktuuri kielelle $\mathcal{L}$ ja $U$ struktuurin $\mathcal{S}$ universumi.

**Määritelmä.** Seuraavassa määritellään, milloin kaava $\phi \in \mathcal{L}$ on *tosi* struktuurissa $\mathcal{S}$ (merk. $\mathcal{S} \models \phi$) ja milloin *epätosi* (merk. $\mathcal{S} \not\models \phi$).

1. $\mathcal{S} \models t_1 = t_2 \iff t_1^{\mathcal{S}}$ ja $t_2^{\mathcal{S}}$ ovat sama universumin $U$ alkio (yllä $t_1$ ja $t_2$ ovat mitä tahansa termejä).
2. $\mathcal{S} \models P(t_1, \ldots, t_n) \iff$ $\langle t_1, \ldots, t_n \rangle^{\mathcal{S}}$ (eli jono $\langle t_1^{\mathcal{S}}, \ldots, t_n^{\mathcal{S}} \rangle$) kuuluu tulkintaan $P^{\mathcal{S}}$ (yllä $n > 0$, $P \in \mathcal{P}_n$ ja $t_1, \ldots, t_n$ ovat mitä tahansa termejä).
3. $\mathcal{S} \models P \iff$ tyhjä jono $\langle\rangle$ kuuluu tulkintaan $P^{\mathcal{S}}$ (missä $P \in \mathcal{P}_0$).
4. $\mathcal{S} \models \neg\alpha \iff \mathcal{S} \not\models \alpha$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



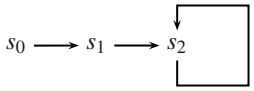
5. $\mathcal{S} \models \alpha \wedge \beta \iff \mathcal{S} \models \alpha$ ja $\mathcal{S} \models \beta$.
6. $\mathcal{S} \models \alpha \vee \beta \iff \mathcal{S} \models \alpha$ tai $\mathcal{S} \models \beta$.
7. $\mathcal{S} \models \alpha \to \beta \iff \mathcal{S} \not\models \alpha$ tai $\mathcal{S} \models \beta$.
8. $\mathcal{S} \models \alpha \leftrightarrow \beta \iff$ joko $\mathcal{S} \models \alpha$ ja $\mathcal{S} \models \beta$, tai $\mathcal{S} \not\models \alpha$ ja $\mathcal{S} \not\models \beta$.
9. $\mathcal{S} \models \exists x\, \alpha(x) \iff$ $\mathcal{S}[x \mapsto a] \models \alpha(x)$ *jollekin* universumin $U$ alkiolle $a \in U$.
10. $\mathcal{S} \models \forall x\, \alpha(x) \iff$ $\mathcal{S}[x \mapsto a] \models \alpha(x)$ *kaikille* universumin $U$ alkioille $a \in U$.

**Väite.** Kaikille kaavoille $\phi \in \mathcal{L}$ pätee joko $\mathcal{S} \models \phi$ tai $\mathcal{S} \not\models \phi$.

**Väite.** Jos kaava $\phi \in \mathcal{L}$ on lisäksi *lause*, sen totuusarvo ei riipu muuttujien $v \in \mathcal{V}$ tulkinnoista struktuurissa $\mathcal{S}$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Tarkastellaan graafin solmujen joukkoa (universumi) ja esitetään kaaret kaksipaikkaisen predikaatin $K$ avulla. Nyt esim. graafia




vastaa struktuuri $\mathcal{S}$, jonka universumi on $U = \{s_0, s_1, s_2\}$ ja $K$:n tulkinta $K^{\mathcal{S}} = \{\langle s_0, s_1\rangle, \langle s_1, s_2\rangle, \langle s_2, s_2\rangle\}$ (muuttujat $x$ ja $y$ tulkittavissa vapaasti).

1. $\langle s_0, s_1\rangle \in K^{\mathcal{S}} \implies \langle x, y\rangle^{\mathcal{S}[x\mapsto s_0, y\mapsto s_1]} \in K^{\mathcal{S}[x\mapsto s_0, y\mapsto s_1]}$
   $\implies \mathcal{S}[x \mapsto s_0, y \mapsto s_1] \models K(x, y)$
   $\implies \mathcal{S}[x \mapsto s_0] \models \exists y K(x, y)$.
2. Vastaavasti $\langle s_1, s_2\rangle \in K^{\mathcal{S}} \implies \mathcal{S}[x \mapsto s_1] \models \exists y K(x, y)$.
3. Vastaavasti $\langle s_2, s_2\rangle \in K^{\mathcal{S}} \implies \mathcal{S}[x \mapsto s_2] \models \exists y K(x, y)$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** (jatkoa)

4. Koska $\mathcal{S}[x \mapsto s_i] \models \exists y K(x, y)$ jokaiselle universumin alkiolle $s_i \in U$, voimme todeta, että $\mathcal{S} \models \forall x \exists y K(x, y)$.
5. Lisäksi esim.
   $\langle s_2, s_2\rangle \in K^{\mathcal{S}} \implies \langle x, x\rangle^{\mathcal{S}[x\mapsto s_2]} \in K^{\mathcal{S}[x\mapsto s_2]}$
   $\implies \mathcal{S}[x \mapsto s_2] \models K(x, x)$
   $\implies \mathcal{S}[x \mapsto s_2] \not\models \neg K(x, x)$
   $\implies \mathcal{S} \not\models \forall x \neg K(x, x)$
   $\implies \mathcal{S} \models \neg\forall x \neg K(x, x)$.

Mieti millaisia graafin ominaisuuksia lauseet $\forall x \exists y K(x, y)$ ja $\neg\forall x \neg K(x, x)$ itse asiassa tarkoittavat!

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Tarkastellaan seuraavia opiskelijatietokannan relaatiotauluja:

Op (opiskelija)

| | |
|---|---|
| 3310D | Taina |
| 4820H | Otso |
| 9055Z | Elsi |
| 4358E | Uljas |
| ... | ... |

Ko (koulutusohjelma)

| | |
|---|---|
| 4358E | Tieto |
| 3310D | Sähkö |
| 9055Z | Kemia |
| 4820H | Rakennus |
| ... | ... |

1. *Kyselyn* $\exists x(\mathsf{Op}(x, \mathsf{Elsi}) \wedge \mathsf{Ko}(x, \mathsf{Tieto}))$ *evaluointi* tarkoittaa kyseisen kaavan totuusarvon laskentaa vastaavassa struktuurissa.
2. Kyselyyn $\neg\forall x(\mathsf{Op}(x, \mathsf{Elsi}) \rightarrow \neg\mathsf{Ko}(x, \mathsf{Tieto}))$ saadaan sama vastaus. Miten tämä on perusteltavissa?

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 2.3 Semanttiset peruskäsitteet

- Semanttisten peruskäsitteiden määritelmät ovat muodoltaan samat.
- Olennaisena erona lauselogiikkaan on, että lauseiden rakenne on monipuolisempi ja että struktuurit korvaavat totuusjakelut.

**Määritelmä.** Struktuuri $\mathcal{S}$ on lauseen $\alpha \in \mathcal{L}$ *malli*, joss lause $\alpha$ on tosi struktuurissa $\mathcal{S}$ eli $\mathcal{S} \models \alpha$.

**Määritelmä.** Struktuuri $\mathcal{S}$ on lausejoukon $\Sigma \subseteq \mathcal{L}$ malli, joss kaikille lausejoukon $\Sigma$ lauseille $\sigma \in \Sigma$ pätee $\mathcal{S} \models \sigma$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Määritelmä.** Lause $\alpha \in \mathcal{L}$ (tai lausejoukko $\Sigma \subseteq \mathcal{L}$) on *toteutuva*, joss ainakin yksi struktuuri $\mathcal{S}$ on sen malli.

**Esimerkki.** $\exists x \forall y P(x,y)$ on toteutuva.
Olkoon struktuurin $\mathcal{S}$ universumi $U = \{1,2\}$ ja $P^{\mathcal{S}} = \{\langle 1,1\rangle, \langle 1,2\rangle\}$.

1. $\langle 1,1\rangle \in P^{\mathcal{S}} \implies \langle x,y\rangle^{\mathcal{S}[x\mapsto 1, y\mapsto 1]} \in P^{\mathcal{S}[x\mapsto 1, y\mapsto 1]}$
   $\implies \mathcal{S}[x\mapsto 1, y\mapsto 1] \models P(x,y)$.
2. $\langle 1,2\rangle \in P^{\mathcal{S}} \implies \langle x,y\rangle^{\mathcal{S}[x\mapsto 1, y\mapsto 2]} \in P^{\mathcal{S}[x\mapsto 1, y\mapsto 2]}$
   $\implies \mathcal{S}[x\mapsto 1, y\mapsto 2] \models P(x,y)$.
3. Siis $\mathcal{S}[x\mapsto 1] \models \forall y P(x,y)$ ja $\mathcal{S} \models \exists x \forall y P(x,y)$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Määritelmä.** Kielen $\mathcal{L}$ lause $\alpha$ on *pätevä* (merkitään $\models \alpha$), joss $\mathcal{S} \models \alpha$ kaikissa $\mathcal{L}$:n struktuureissa $\mathcal{S}$.

**Esimerkki.** Osoitetaan $\models \forall x P(x) \leftrightarrow \neg\exists x \neg P(x)$.

Kaikille kielen $\mathcal{L}$ struktuureille $\mathcal{S}$ ja vastaaville universumeille $U$ pätee:

$\mathcal{S} \models \forall x P(x)$
$\iff$ $\mathcal{S}[x\mapsto a] \models P(x)$ kaikille $a \in U$
$\iff$ ei ole niin, että $\mathcal{S}[x\mapsto a] \not\models P(x)$ jollekin $a \in U$
$\iff$ ei ole niin, että $\mathcal{S}[x\mapsto a] \models \neg P(x)$ jollekin $a \in U$
$\iff$ $\mathcal{S} \not\models \exists x \neg P(x)$
$\iff$ $\mathcal{S} \models \neg\exists x \neg P(x)$.

Siis $\mathcal{S} \models \forall x P(x) \leftrightarrow \neg\exists x \neg P(x)$ struktuurista $\mathcal{S}$ riippumatta.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** $\models \forall x P(x) \vee \neg\forall x P(x)$
Olkoon $\mathcal{S}$ mielivaltainen $\mathcal{L}$:n struktuuri.
Nyt $\mathcal{S} \models \forall x P(x) \iff \mathcal{S} \not\models \neg\forall x P(x)$, joten $\mathcal{S} \models \forall x P(x) \vee \neg\forall x P(x)$.

**Määritelmä.** Kielen $\mathcal{L}$ lause $\alpha$ on lausejoukon $\Sigma \subseteq \mathcal{L}$ *looginen seuraus* (merkitään $\Sigma \models \alpha$), joss $\alpha$ on tosi jokaisessa lausejoukon $\Sigma$ mallissa $\mathcal{S}$.

**Esimerkki.** $\{\forall x P(x)\} \models \exists x P(x)$

Olkoon $\mathcal{S}$ struktuuri siten, että $\mathcal{S} \models \forall x P(x)$. Tällöin
$\implies$ kaikille $a \in U$ pätee $\mathcal{S}[x\mapsto a] \models P(x)$
$\implies$ jollekin $a \in U$ pätee $\mathcal{S}[x\mapsto a] \models P(x)$
(universumi $U$ on ei-tyhjä struktuurin määritelmän perusteella)
$\implies$ $\mathcal{S} \models \exists x P(x)$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.**

$$\Sigma = \left\{ \begin{array}{l} \text{tentti(tiistai)}, \\ \text{tentti(keskiviikko)}, \\ \text{luento(keskiviikko)}, \\ \forall x(\neg\text{tentti}(x) \wedge \neg\text{luento}(x) \rightarrow \text{vapaa}(x)) \end{array} \right\}$$

Onko $\Sigma \models$ vapaa(perjantai)? *Ei*, koska löytyy *vastamalli* $\mathcal{S}$, jonka perusteella $\Sigma \not\models$ vapaa(perjantai) eli $\mathcal{S} \models \Sigma$ ja $\mathcal{S} \not\models$ vapaa(perjantai)!

Olkoon universumi $U = \{t,k,p\}$ ja symbolien tulkinnat seuraavat:

| | | | | | |
|---|---|---|---|---|---|
| tiistai$^{\mathcal{S}}$ | $=$ | $t$, | keskiviikko$^{\mathcal{S}}$ | $=$ | $k$, |
| perjantai$^{\mathcal{S}}$ | $=$ | $p$, | tentti$^{\mathcal{S}}$ | $=$ | $\{t,k\}$, |
| luento$^{\mathcal{S}}$ | $=$ | $\{k,p\}$, | vapaa$^{\mathcal{S}}$ | $=$ | $\{\}$. |

Mutta: $\Sigma \cup \{\neg\text{tentti(perjantai)}, \neg\text{luento(perjantai)}\} \models$ vapaa(perjantai).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## 2.4 Peruskäsitteiden väliset yhteydet

Seuraavat ominaisuudet ovat voimassa myös predikaattilogiikassa:

- $\models \alpha \iff \neg\alpha$ on toteutumaton.
- $\Sigma \models \alpha \iff \Sigma \cup \{\neg\alpha\}$ on toteutumaton.
- $\models \alpha \iff \emptyset \models \alpha$.
- $\{\alpha_1, \ldots, \alpha_n\} \models \alpha \iff \models \alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \alpha$.

Olkoon $\mathrm{Cn}(\Sigma) = \{\phi \in \mathcal{L} \mid \phi$ on lause ja $\Sigma \models \phi\}$ lausejoukolle $\Sigma \subseteq \mathcal{L}$:

- $\Sigma \subseteq \mathrm{Cn}(\Sigma)$ ja $\Sigma \equiv \mathrm{Cn}(\Sigma)$.
- Monotonisuus: $\Sigma_1 \subseteq \Sigma_2 \implies \mathrm{Cn}(\Sigma_1) \subseteq \mathrm{Cn}(\Sigma_2)$.
- $\mathrm{Cn}(\mathrm{Cn}(\Sigma)) = \mathrm{Cn}(\Sigma)$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



# 3 Semanttiset taulut predikaattilogiikalle

- Taulusäännöt kvanttoreiden käsittelyyn
- Semanttiset taulut predikaattilogiikalle
- Ohjeita taulutodistusten laadintaan
- Systemaattinen taulu
- Vastamallin konstruointi

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 3.1 Taulusäännöt kvanttorien käsittelyyn

- Muotoa $T\,\exists x\varphi(x)$ (tai $E\,\forall x\varphi(x)$) oleva solmu tulee hajoittaa *kertaalleen* käyttäen jotain (hajoitushetkellä) *uutta vakiota* $c$.

| $T\,\exists x\varphi(x)$ <br> $\mid$ <br> $T\varphi(c)$ | $E\,\forall x\varphi(x)$ <br> $\mid$ <br> $E\varphi(c)$ |
|---|---|
| $c$ uusi vakio | $c$ uusi vakio |

- Olkoon $P$ polku (juurisolmusta lehtisolmuun), jolla solmu $T\,\exists x\varphi(x)$ ($E\,\forall x\varphi(x)$) esiintyy ja jota on tarkoitus jatkaa ao. taulusäännöllä. Vakio $c$ on *uusi*, mikäli se ei esiinny polulla $P$.

**Huomio.** Uuden vakion käyttöönotto johtuu siitä, ettemme tiedä, millä universumin alkiolla on ko. ominaisuus $\varphi$ (tai ei ole ominaisuutta $\varphi$).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



- Muotoa $T\,\forall x\varphi(x)$ (tai $E\,\exists x\varphi(x)$) oleva solmu tulee (tarvittaessa) hajoittaa kaikille *muuttujattomille termeille* $t$ (vakioita tai vakioista ja funktiosymboleista rakentuvia monimutkaisempia termejä).

| $T\,\forall x\varphi(x)$ <br> $\mid$ <br> $T\varphi(t)$ | $E\,\exists x\varphi(x)$ <br> $\mid$ <br> $E\varphi(t)$ |
|---|---|
| $t$ muuttujaton termi | $t$ muuttujaton termi |

**Huomio.**

- Muuttujattomia termejä voi olla ääretön määrä, joten muotoa $T\,\forall x\varphi(x)$ (tai $E\,\exists x\varphi(x)$) olevaa solmua ei välttämättä saada hajoitetuksi soveltamalla ao. taulusääntöä äärellisen monta kertaa.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Juurisolmusta $T\,\exists x\forall y\exists z P(x,y,z)$ muodostettua semanttista taulua ei saada valmiiksi äärellisellä määrällä hajoituksia.

1. $T\,\exists x\forall y\exists z P(x,y,z)$
2. $T\,\forall y\exists z P(a,y,z)$ 1. $x/a$ ($a$ uusi vakio)
3. $T\,\exists z P(a,a,z)$ 2. $y/a$
4. $TP(a,a,b)$ 3. $z/b$ ($b$ uusi vakio)
5. $T\,\exists z P(a,b,z)$ 2. $y/b$
6. $TP(a,b,c)$ 5. $z/c$ ($c$ uusi vakio)

jne.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 3.2 Semanttiset taulut predikaattilogiikalle

- Semanttisten taulujen määritelmä säilyy ennallaan.
- Ehtoja, millä polun solmu on *hajoitettu*, joudutaan täydentämään:
  Olkoon $\tau$ semanttinen taulu ja $P$ polku juurisolmusta lehtisolmuun $\tau$:ssa. $P$:n solmu $T\,\forall x\varphi(x)$ ($E\,\exists x\varphi(x)$) hajoitettu polulla $P$, jos
  - polulla $P$ esiintyy $T\varphi(t)$ ($E\varphi(t)$) kaikille muuttujattomille termeille $t$, jotka voidaan muodostaa polulla $P$ esiintyvistä vakio- ja funktiosymboleista (vakiosymboleja on oltava ainakin yksi).

  **Huomio.** Mikäli polulla $P$ ei esiinny vakiosymboleita, $T\,\forall x\varphi(x)$ ($E\,\exists x\varphi(x)$) tulee hajoittaa käyttäen jotain uutta vakiosymbolia $c$.
- Muilta osin semanttisen taulun (ja sen polkujen) ristiriitaisuuden ja valmiuden määritelmät säilyvät ennallaan.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Loogisten ongelmien ratkominen

Taulumenetelmää voidaan käyttää erilaisten loogisten ongelmien ratkomiseen aivan kuten lauselogiikankin tapauksessa.

**Määritelmä.** Taulu $\tau$ on lauseen $\phi$ *todistus*, jos taulun $\tau$ juurisolmuna on $E\phi$ ja $\tau$ on ristiriitainen. Jos lauseella $\phi$ on todistus, on $\phi$ *teoreema/todistuva* (merkitään $\vdash\phi$).

**Väite.** $\vdash\phi \iff \models\phi$ (virheettömyys ja täydellisyys).

**Määritelmä.** Lause $\phi$ *on johdettavissa* äärellisestä lausejoukosta $\Sigma=\{\phi_1,\ldots,\phi_n\}$ (merkitään $\Sigma\vdash\phi$), joss juurisolmusta

$$E(\phi_1\wedge\cdots\wedge\phi_n\to\phi)$$

muodostettu valmis semanttinen taulu on ristiriitainen.

**Väite.** $\Sigma\vdash\phi \iff \Sigma\models\phi$ (virheettömyys ja täydellisyys).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Onko $\{P(a),\forall x(P(x)\to Q(x))\}\vdash Q(a)$ ?

1. $T\,\forall x(P(x)\to Q(x))$
2. $T\,P(a)$
3. $E\,Q(a)$
4. $T(P(a)\to Q(a))$ 1. $x/a$
   - 5. $E\,P(a)$ 4. ×
   - 5. $T\,Q(a)$ 4. ×

Taulu on ristiiriitainen. Lause $Q(a)$ on siis johdettavissa lausejoukosta $\{P(a),\forall x(P(x)\to Q(x))\}$ (sekä lausejoukon looginen seuraus).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Onko $\{\exists x(P(x) \wedge Q(x))\} \vdash P(a)$ ?

1. $T\,\exists x(P(x) \wedge Q(x))$
2. $EP(a)$
3. $T(P(b) \wedge Q(b))$ 1. $x/b$
4. $TP(b)$ 3.
5. $TQ(b)$ 3.

Taulu saatiin valmiiksi muttei ristiriitaiseksi. Lause $P(a)$ ei ole johdettavissa lausejoukosta $\{\exists x(P(x) \wedge Q(x))\}$
(eikä myöskään lausejoukon looginen seuraus).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 3.3 Ohjeita taulutodistusten laadintaan

- Lauseen rakenne määrää edelleen, mitä taulusääntöä tulee käyttää (jäsennyspuun juuressa oleva konnektiivi).
- Solmujen hajoittamisjärjestyksellä voi usein vaikuttaa taulun kokoon (haarautumista kannattaa välttää).
- Jälkimmäisissä kvanttorisäännöissä esiintyvän termin $t$ tilalle valitaan *hajoittamishetkellä* (eikä myöhemmin) jokin vakio tai funktio- ja vakiosymboleista rakentuva muuttujaton termi.
- Valitsemalla muuttujattomat termit $t$ sopivasti voidaan usein nopeuttaa taulun valmistumista.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Taulutodistusten erityispiirteitä predikaattilogiikan tapauksessa**

1. Valitaan muuttujattomaksi termiksi $t$ vakio, joka ei esiinny lauseissa.

**Esimerkki.** Osoitetaan $\{\forall xP(x)\} \vdash \exists xP(x)$.

1. $T\,\forall xP(x)$
2. $E\,\exists xP(x)$
3. $TP(c)$ 1. $x/c$
4. $EP(c)$ 2. $x/c$
   ×

**Huomio.** Tämä on perusteltua, koska universumissa $U$ on aina vähintään yksi alkio $a \in U$, joka voidaan nimetä (eli $c^{\mathcal{S}} = a$).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



2. Solmu $T\,\forall x\varphi(x)$ (tai $E\,\exists x\varphi(x)$) joudutaan hajoittamaan useasti.

**Esimerkki.** $\{\forall xP(x)\} \vdash P(a) \wedge P(b)$

1. $T\,\forall xP(x)$
2. $E(P(a) \wedge P(b))$
3. $TP(a)$ 1. $x/a$
4. $TP(b)$ 1. $x/b$
5. $EP(a)$ 2. ×
6. $EP(b)$ 2. ×

1. $T\,\forall xP(x)$
2. $E(P(a) \wedge P(b))$
3. $EP(a)$ 2. — 4. $TP(a)$ 1. $x/a$ ×
3. $EP(b)$ 2. — 4. $TP(b)$ 1. $x/b$ ×

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

3. Muuttujien korvaaminen sopivilla muuttujattomilla termeillä.

**Esimerkki.** $\{\forall x\forall y\forall z(P(x,y)\wedge P(y,z)\rightarrow P(x,z)), P(a,b), P(b,c)\}\vdash P(a,c)$

- Semanttiseen tauluun tulee solmu $T\,\forall x\forall y\forall z(P(x,y)\wedge P(y,z)\rightarrow P(x,z))$, josta voidaan johtaa kvanttorisäännöillä 27 erilaista todeksi merkittyä implikaatiota.
- Ristiriidan johtamisen kannalta olennaisia ovat implikaatioista ne, joissa esiintyy atomisia lauseita $P(a,b)$, $P(b,c)$ ja $P(a,c)$.
- Esimerkin tapauksessa tämä johtaa ensimmäiseksi $x$:n, $y$:n ja $z$:n korvaamiseen vakoilla $a$, $b$, ja $c$ (näin saatava implikaatio riittää).
- Muita implikaatioita ei tarvita, ja niiden johtaminen ja mahdollinen hajoittaminen johtaa semanttisen taulun tarpeettomaan kasvuun.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



Taulutodistus muodostuu kokonaisuudessaan seuraavaksi:

1. $T\,\forall x\forall y\forall z(P(x,y)\wedge P(y,z)\rightarrow P(x,z))$
2. $TP(a,b)$
3. $TP(b,c)$
4. $EP(a,c)$
5. $T\,\forall y\forall z(P(a,y)\wedge P(y,z)\rightarrow P(a,z))^{1.\ x/a}$
6. $T\,\forall z(P(a,b)\wedge P(b,z)\rightarrow P(a,z))^{5.\ y/b}$
7. $T(P(a,b)\wedge P(b,c)\rightarrow P(a,c))^{6.\ z/c}$
8. $E(P(a,b)\wedge P(b,c))^{7.}$ | 8. $TP(a,c)^{7.}$ ×
9. $EP(a,b)^{8.}$ × | 9. $EP(b,c)^{8.}$ ×

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



4. Solmujen keskinäinen hajoitusjärjestys voi olla ratkaisevassa roolissa.

**Esimerkki.** $\{\forall xP(x), \forall x(P(x)\rightarrow Q(x)), Q(a)\}\vdash \forall xQ(x)$

1. $T\,\forall xP(x)$
2. $T\,\forall x(P(x)\rightarrow Q(x))$
3. $T\,Q(a)$
4. $E\,\forall xQ(x)$
5. $E\,Q(c)^{4.\ x/c}$
6. $T\,P(c)^{1.\ x/c}$
7. $T(P(c)\rightarrow Q(c))^{2.\ x/c}$
8. $E\,P(c)^{7.}$ × | 8. $T\,Q(c)^{7.}$ ×

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## Kvanttorisekvenssien käsittely

Jatkossa sallimme seuraavien johdettujen taulusääntöjen käytön kvanttorisekvenssien käsittelyssä:

| | |
|---|---|
| $T\,\forall x_1\cdots\forall x_n\varphi(x_1,\ldots,x_n)$ <br> \| <br> $T\varphi(t_1,\ldots,t_n)$ | $E\,\exists x_1\cdots\exists x_n\varphi(x_1,\ldots,x_n)$ <br> \| <br> $E\varphi(t_1,\ldots,t_n)$ |
| $T\,\exists x_1\cdots\exists x_n\varphi(x_1,\ldots,x_n)$ <br> \| <br> $T\varphi(c_1,\ldots,c_n)$ | $E\,\forall x_1\cdots\forall x_n\varphi(x_1,\ldots,x_n)$ <br> \| <br> $E\varphi(c_1,\ldots,c_n)$ |

Yllä $c_1,\ldots,c_n$ ovat ao. taulusääntöjen edellyttämiä *uusia vakioita* ja vastaavasti $t_1,\ldots,t_n$ ovat valittuja *muuttujattomia termejä*.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## 3.4 Systemaattinen taulu

- Lauselogiikan keskeiset päättelyongelmat ovat *ratkeavia*.

**Esimerkki.** Voidaan konstruoida *deterministinen* Turing-kone $T$, jonka suoritus pysähtyy syötteeksi annetulla lauselogiikan lauseella $\varphi$

1. hyväksyvään tilaan $k$ (kyllä), jos syöte $\varphi$ on pätevä, ja
2. hylkäävään tilaan $e$ (ei), jos syöte $\varphi$ ei ole pätevä.

**Huomioita.**

- Tällainen algoritmi voi perustua esim. totuustaulukkoihin, semanttisiin tauluihin tai resoluutioon.
- Myös looginen ekvivalenssi, looginen seuraavuus ja toteutuvuus ovat lauselogiikan tapauksessa ratkeavia ongelmia.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



- Predikaattilogiikka ei ole ratkeava, vaan *puoliratkeava*.

**Esimerkki.** Lauseen $\varphi$ pätevyyden tarkastamista varten voidaan konstruoida seuraavanlainen deterministinen Turing-kone $T$:

1. Jos syöte $\varphi$ on pätevä, $T$ pysähtyy hyväksyvään tilaan $k$ (kyllä).
2. Jos syöte $\varphi$ ei ole pätevä, $T$ pysähtyy *joskus* hylkäävään tilaan $e$ (ei) ja *joskus $T$ ei pysähdy lainkaan*.

**Huomio.** Tällainen algoritmi voi perustua semanttisiin tauluihin:

- Rakentamalla semanttinen taulu tietyllä tavalla *systemaattisesti*, voidaan taata, että taulu saadaan aina ristiriitaiseksi, kun sen juuressa on $E\varphi$ ja $\varphi$ on pätevä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Systemaattisen taulun periaatteita

- Tuotetaan indeksoimalla riittävä määrä uusia vakioita $c_1, c_2, c_3, \ldots$, kun hajoitetaan muotoa $T\exists x\psi(x)$ tai $E\forall x\psi(x)$ olevia solmuja.
- Tuotetaan tarpeen mukaan muuttujattomia termejä $t_1, t_2, t_3, \ldots$, jotka rakentuvat $E\varphi$:ssä esiintyvistä vakio- ja funktiosymboleista sekä mahdollisesti käyttöönotetuista uusista vakioista $c_1, c_2, c_3, \ldots$.
- Sekvenssin $t_1, t_2, t_3, \ldots$ on oltava *reilu*: jokainen em. symboleista rakentuva muuttujaton termi $t$ esiintyy siinä jonain terminä $t_i$.
- *Hajoitusten reiluus*: taataan, että taulun keskeneräisillä poluilla esiintyvät hajoittamattomat solmut tulevat hajoitusvuoroon (seuraavan kerran) äärellisen monen muun hajoituksen jälkeen.
- Muotoa $T\forall x\psi(x)$ tai $E\exists x\psi(x)$ olevia solmuja hajoitetaan järjestyksessä käyttäen muuttujattomia termejä $t_1, t_2, t_3, \ldots$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Huomio.** Muotoa $T\forall x\phi(x)$ tai $E\exists x\phi(x)$ olevia solmuja sisältäviä polkuja ei välttämättä saada hajoitetuksi äärellisellä askelmäärällä, jolloin valmis taulu muodostuu äärettömäksi (puoliratkeavuuden ilmentymä).

**Esimerkki.** Kirjoitetaan luonnollisten lukujen $>$-relaatiolle määritelmä. Olkoon $G(x,y) =$ "$x > y$"ja $s(x)$ luvun $x$ seuraaja (eli $x+1$).

- Määritelmä: $\forall x G(s(x),x)$ ja $\forall x\forall y(G(x,y) \to G(s(x),y))$.
- Kysely: onko $G(s(s(s(0))),s(0))$ määritelmän looginen seuraus?

Hajoitetaan systemaattisesti semanttisen taulun solmuja

$$T\forall x G(s(x),x) \text{ ja } T\forall x\forall y(G(x,y) \to G(s(x),y))$$

käyttäen muuttujattomia termejä: $0, s(0), s(s(0)), s(s(s(0))), \ldots$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

- Mikäli solmujen hajoitusjärjestys rikkoo edellä esitettyä reiluusperiaatetta, todistuksen löytyminen ei ole taattu.

**Esimerkki.** Järjestys, missä hajoitetaan ainoastaan solmua $T\,\forall x G(s(x),x)$ em. muuttujattomien termien suhteen, ei ole reilu:

1. $T\,\forall x G(s(x),x)$
2. $T\,\forall x\forall y(G(x,y)\rightarrow G(s(x),y))$
3. $E\,G(s(s(s(0))),s(0))$
4. $T\,G(s(0),0)$ 1. $x/0$
5. $T\,G(s(s(0)),s(0))$ 1. $x/s(0)$
6. $T\,G(s(s(s(0))),s(s(0)))$ 1. $x/s(s(0))$
7. $\vdots$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Käytetään reilua hajoitusjärjestystä:

1. $T\,\forall x G(s(x),x)$
2. $T\,\forall x\forall y(G(x,y)\rightarrow G(s(x),y))$
3. $E\,G(s(s(s(0))),s(0))$
4. $T\,G(s(0),0)$ 1. $x/0$
5. $T\,\forall y(G(0,y)\rightarrow G(s(0),y))$ 2. $x/0$
6. $T(G(0,0)\rightarrow G(s(0),0))$ 5. $y/0$
7. $E\,G(0,0)$ | 7. $T\,G(s(0),0)$
8. $T\,G(s(s(0)),s(0))$ 1. $x/s(0)$ | 8. $T\,G(s(s(0)),s(0))$ 1. $x/s(0)$
9. $T\,\forall y(G(s(0),y)\rightarrow G(s(s(0)),y))$ 2. $x/s(0)$
10. $\vdots$ | $\vdots$

Systemaattinen taulu voi tehdä turhaa työtä $\Longrightarrow$ heuristiikkaa tarvitaan!

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Valitsemalla muuttujattomat termit aikaisemmin esitetyillä periaatteilla semanttinen taulu jää huomattavasti pienemmäksi:

1. $T\,\forall x G(s(x),x)$
2. $T\,\forall x\forall y(G(x,y)\rightarrow G(s(x),y))$
3. $E\,G(s(s(s(0))),s(0))$
4. $T\,\forall y(G(s(s(0)),y)\rightarrow G(s(s(s(0))),y))$ 2. $x/s(s(0))$
5. $T\,(G(s(s(0)),s(0))\rightarrow G(s(s(s(0))),s(0)))$ 5. $y/s(0)$
6. $E\,G(s(s(0)),s(0))$ | 6. $T\,G(s(s(s(0))),s(0))$ ×
7. $T\,G(s(s(0)),s(0))$ 1. $x/s(0)$
8. ×

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 3.5 Vastamallien konstruointi

- Vastamallin (struktuuri) $\mathcal{S}$ konstruoinnissa voidaan hyödyntää semanttisen taulun ristiriidattomasta polusta saatavia *atomisia lauseita* koskevia totuusarvovaatimuksia $T\,P(t_1,\ldots,t_n)$, $E\,Q(s_1,\ldots,s_m)$, ... ($t_i$:t ja $s_j$:t ovat muuttujattomia termejä).
- Valitaan riittävän iso universumi $U$, jotta pystytään antamaan tulkinnat totuusarvovaatimuksissa esiintyville vakio- ja funktiosymboleille.
- Tämän jälkeen valitaan predikaattien tulkinnat totuusarvovaatimusten mukaisesti:
  1. Jos $T\,P(t_1,\ldots,t_n)$ on polulla, $\langle t_1^{\mathcal{S}},\ldots,t_n^{\mathcal{S}}\rangle \in P^{\mathcal{S}}$.
  2. Jos $E\,Q(s_1,\ldots,s_m)$ on polulla, $\langle s_1^{\mathcal{S}},\ldots,s_m^{\mathcal{S}}\rangle \notin Q^{\mathcal{S}}$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

- Menettely on käyttökelpoinen erityisesti, jos *valmiin semanttisen taulun* ristiriidaton polku muodostuu *äärelliseksi*:

**Esimerkki.** Vastamalli $\mathcal{S}$ lauseen $\forall x(P(x) \to Q(x))$ pätevyydelle.

$$\begin{array}{l} 1.\ E\,\forall x(P(x) \to Q(x)) \\ \quad | \\ 2.\ E(P(c) \to Q(c))\ {}^{1.\ x/c} \\ \quad | \\ 3.\ T(P(c))\ {}^{2.} \\ \quad | \\ 4.\ E(Q(c))\ {}^{2.} \end{array}$$

1. Totuusarvovaatimukset ristiriidattomasta polusta: $T\,P(c)$ ja $E\,Q(c)$.
2. Riittää, että universumiin $U = \{1\}$ otetaan yksi alkio s.e. $c^{\mathcal{S}} = 1$.
3. Totuusarvovaatimusten nojalla: $1 \in P^{\mathcal{S}}$ ja $1 \notin Q^{\mathcal{S}}$.
4. Nämä vaatimukset toteutuvat valinnoilla $P^{\mathcal{S}} = \{1\}$ ja $Q^{\mathcal{S}} = \emptyset$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** $\{\forall x(P(x) \wedge Q(x) \to R(x))\} \not\models \forall x(Q(x) \to R(x))$.

$$\begin{array}{l} 1.\ T\,\forall x(P(x) \wedge Q(x) \to R(x)) \\ \quad | \\ 2.\ E\,\forall x(Q(x) \to R(x)) \\ \quad | \\ 3.\ E(Q(c) \to R(c))\ {}^{2.\ x/c} \\ \quad | \\ 4.\ T\,Q(c)\ {}^{3.} \\ \quad | \\ 5.\ E\,R(c)\ {}^{3.} \\ \quad | \\ 6.\ T\,(P(c) \wedge Q(c) \to R(c))\ {}^{1.\ x/c} \\ \quad\quad \diagup \quad \diagdown \\ 7.\ E(P(c) \wedge Q(c)) \qquad 7.\ T\,R(c) \\ \quad\quad \diagup \quad \diagdown \qquad\quad \times \\ 8.\ E\,P(c) \qquad 8.\ E\,Q(c) \\ \qquad\qquad \times \end{array}$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



- Tarkastellaan taulun ainoaa ristiriidatonta polkua $P$.
  - Polulla esiintyy yksi vakiosymboli $c$ muttei funktiosymboleja.
  - Voidaan muodostaa ainoastaan yksi muuttujaton termi eli $c$ itse.
  - Täten solmu $T\,\forall x(P(x) \wedge Q(x) \to R(x))$ on hajoitettu polulla $P$, koska polulla $P$ on solmu $T(P(t) \wedge Q(t) \to R(t))$ jokaista muuttujatonta termiä $t \in \{c\}$ kohti.
  - Polku $P$ on siis valmis.
- Näin ollen taulu on kokonaisuutena myös valmis.
- Polulta $P$ saadaan totuusarvovaatimukset $E\,P(c)$, $T\,Q(c)$ ja $E\,R(c)$.
- Muodostetaan vastamalli $\mathcal{S}$ valitsemalla universumiksi $U = \{1\}$ ja symbolien tulkinnoiksi $c^{\mathcal{S}} = 1$, $P^{\mathcal{S}} = R^{\mathcal{S}} = \{\}$ ja $Q^{\mathcal{S}} = \{1\}$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Joskus äärettömästäkin ristiriidattomasta polusta voi onnistua muodostamaan vastamallin, jolla on *äärellinen* universumi $U$.

$$\begin{array}{l} 1.\ E\,\exists x(P(a) \vee P(f(x))) \\ \quad | \\ 2.\ E(P(a) \vee P(f(a)))\ {}^{1.\ x/a} \\ \quad | \\ 3.\ E\,P(a)\ {}^{2.} \\ \quad | \\ 4.\ E\,P(f(a))\ {}^{2.} \\ \quad | \\ 5.\ E(P(a) \vee P(f(f(a))))\ {}^{1.\ x/f(a)} \\ \quad | \\ 6.\ E\,P(a)\ {}^{5.} \\ \quad | \\ 7.\ E\,P(f(f(a)))\ {}^{5.} \\ \quad \vdots \end{array}$$

Edellytyksenä on muuttujattomien termien tulkinta samalle $U$:n alkiolle.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

- Tarkastellaan taulun ainoata ristiriidatonta polkua $P$.
- Polku $P$ ei ole valmis, koska solmua $E\,\exists x(P(a) \vee P(f(x)))$ ei ole hajoitettu esim. muuttujattoman termin $t = f(f(a))$ suhteen.
- Polulta $P$ saadaan ääretön sekvenssi totuusarvovaatimuksia

$$E\,P(a),\ E\,P(f(a)),\ E\,P(f(f(a))),\ \ldots$$

- Näiden säännönmukaisuudesta johtuen valitaan universumiksi $U = \{1\}$ ja symbolien tulkinnoiksi $a^{\mathcal{S}} = 1$, $f^{\mathcal{S}} : 1 \mapsto 1$ ja $P^{\mathcal{S}} = \emptyset$.
- Koska taulu ei ollut valmis, lisäksi on syytä todeta totuusmääritelmästä lähtien, että kysymyksessä on todella vastaesimerkki eli $\mathcal{S} \not\models \exists x(P(a) \vee P(f(x)))$.
- Näin muodostettu struktuuri $\mathcal{S}$ muodostaa vastamallin lauseen $\exists x(P(a) \vee P(f(x)))$ pätevyydelle.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



# 4 Normaalimuodot

- Prenex-normaalimuoto
- Konjunktiivinen normaalimuoto
- Eksistenssikvanttorien eliminointi
- Lauseiden klausuulimuoto predikaattilogiikassa

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 4.1 Prenex-normaalimuoto

Lause $\alpha$ on *prenex*-normaalimuodossa, mikäli se on muotoa

$$Q_1 x_1\, Q_2 x_2 \cdots Q_n x_n \phi,$$

missä jokainen $Q_i$ on jompikumpi kvanttoreista ($\forall$ tai $\exists$) ja alikaava $\phi$ ei sisällä kvanttoreita.

**Esimerkki.** Seuraavat lauseet ovat prenex-normaalimuodossa:
$P(a)$, $\forall x P(x)$, $\forall x \exists y\ P(x,y)$ ja
$\forall x \exists y \forall z \forall w (P(x,y,z) \to (Q(y,z,w) \to R(z,w,x)))$.

**Väite.** Jokainen predikaattilogiikan lause on loogisesti ekvivalentti jonkin prenex-normaalimuodossa olevan lauseen kanssa.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Lauseiden muuttaminen prenex-normaalimuotoon**

Mikä tahansa predikaattilogiikan lause voidaan muuttaa prenex-normaalimuotoon seuraavalla menettelyllä:

1. Poistetaan konnektiivit $\to$ ja $\leftrightarrow$:

$$\phi \to \psi \rightsquigarrow \neg\phi \vee \psi$$
$$\phi \leftrightarrow \psi \rightsquigarrow (\neg\phi \vee \psi) \wedge (\phi \vee \neg\psi)$$

2. Viedään negaatiot lauserakenteen sisään (atomisten kaavojen eteen):

$$\neg\neg\varphi \rightsquigarrow \varphi$$
$$\neg(\varphi \wedge \psi) \rightsquigarrow \neg\varphi \vee \neg\psi \qquad \vec{Q}x\neg\forall y\varphi \rightsquigarrow \vec{Q}x\exists y\neg\varphi$$
$$\neg(\varphi \vee \psi) \rightsquigarrow \neg\varphi \wedge \neg\psi \qquad \vec{Q}x\neg\exists y\varphi \rightsquigarrow \vec{Q}x\forall y\neg\varphi$$

Yllä $\vec{Q}x$ on mikä tahansa kvanttorien sekvenssi.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

3. Tuodaan kvanttorit ulos lauserakenteesta:

$$\vec{Q}x(\forall y\varphi(y) \vee \psi) \rightsquigarrow \vec{Q}x\forall z(\varphi(z) \vee \psi)$$
$$\vec{Q}x(\forall y\varphi(y) \wedge \psi) \rightsquigarrow \vec{Q}x\forall z(\varphi(z) \wedge \psi)$$
$$\vec{Q}x(\psi \vee \forall y\varphi(y)) \rightsquigarrow \vec{Q}x\forall z(\psi \vee \varphi(z))$$
$$\vec{Q}x(\psi \wedge \forall y\varphi(y)) \rightsquigarrow \vec{Q}x\forall z(\psi \wedge \varphi(z))$$

- Yllä $y$ korvataan uudella muuttujalla $z$, mikäli $y$ esiintyy vapaana alikaavassa $\psi$. Muussa tapauksessa $z$ voi aivan hyvin olla $y$.
- Eksistenssikvanttorit $\exists y$ käsitellään samaan tapaan (saadaan 4 vastaavanmuotoista sääntöä lisää).

**Esimerkki.** Muuttujan korvaaminen uudella on olennaista:

$$\forall xP(x) \vee \forall xQ(x) \rightsquigarrow \forall x(P(x) \vee \forall xQ(x)) \rightsquigarrow \forall x\forall y(P(x) \vee Q(y)).$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Muunnetaan seuraava lause prenex-normaalimuotoon:

$$\begin{aligned}
& \forall x(P(x) \to \exists zR(z,x)) \to \exists xQ(x) \\
\rightsquigarrow\ & \neg\forall x(P(x) \to \exists zR(z,x)) \vee \exists xQ(x) \\
\rightsquigarrow\ & \neg\forall x(\neg P(x) \vee \exists zR(z,x)) \vee \exists xQ(x) \\
\rightsquigarrow\ & \exists x\neg(\neg P(x) \vee \exists zR(z,x)) \vee \exists xQ(x) \\
\rightsquigarrow\ & \exists x(\neg\neg P(x) \wedge \neg\exists zR(z,x)) \vee \exists xQ(x) \\
\rightsquigarrow\ & \exists x(P(x) \wedge \forall z\neg R(z,x)) \vee \exists xQ(x) \\
\rightsquigarrow\ & \exists x(\exists x(P(x) \wedge \forall z\neg R(z,x)) \vee Q(x)) \\
\rightsquigarrow\ & \exists x\exists y((P(y) \wedge \forall z\neg R(z,y)) \vee Q(x)) \\
\rightsquigarrow\ & \exists x\exists y(\forall z(P(y) \wedge \neg R(z,y)) \vee Q(x)) \\
\rightsquigarrow\ & \exists x\exists y\forall z((P(y) \wedge \neg R(z,y)) \vee Q(x)).
\end{aligned}$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 4.2 Konjunktiivinen normaalimuoto

**Määritelmä.** *Literaalit* ovat joko

1. atomikaavoja $P(\vec{t})$ (eli *positiivisia literaaleja*) tai
2. atomikaavojen negaatioita $\neg P(\vec{t})$ (eli *negatiivisia literaaleja*).

Yllä $\vec{t}$ tarkoittaa termien $t_1, \ldots, t_n$ sekvenssiä kun $P \in \mathcal{P}_n$. Jos $P \in \mathcal{P}_0$, sekvenssi on tyhjä ja tällöin myös sulut jätetään kirjoittamatta näkyviin.

**Määritelmä.** Lause $\alpha$ on konjunktiivisessa normaalimuodossa, mikäli se on on prenex-normaalimuodossa $Q_1x_1Q_2x_2\cdots Q_nx_n\phi$, missä kvanttoreita sisältämätön osa $\phi = \phi_1 \wedge \cdots \wedge \phi_n$ ja jokainen konjunktion jäsen $\phi_i$ on muodoltaan literaalien disjunktio.

**Esimerkki.** Seuraava lause on konjunktiivisessa normaalimuodossa: $\forall x\exists y\forall z((\neg P(x,y) \vee Q(y,x)) \wedge R(z) \wedge (\neg R(x) \vee P(y,z) \vee Q(x,z)))$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



- Tarvittaessa prenex-normaalimuodon kvanttoreita sisältämätön osa voidaan järjestää konjunktiiviseen normaalimuotoon seuraavasti:

$$\varphi \vee (\phi \wedge \psi) \quad \rightsquigarrow \quad (\varphi \vee \phi) \wedge (\varphi \vee \psi)$$
$$(\phi \wedge \psi) \vee \varphi \quad \rightsquigarrow \quad (\phi \vee \varphi) \wedge (\psi \vee \varphi)$$

- Näin ollen voimme todeta seuraavan tuloksen:

**Väite.** Jokainen predikaattilogiikan lause on loogisesti ekvivalentti jonkin konjunktiivisessa normaalimuodossa olevan lauseen kanssa.

**Esimerkki.** Muunnetaan edellä johdettu prenex-normaalimuoto edelleen konjunktiiviseksi normaalimuodoksi:

$$\begin{aligned}
& \exists x\exists y\forall z((P(y) \wedge \neg R(z,y)) \vee Q(x)) \\
\rightsquigarrow\ & \exists x\exists y\forall z((P(y) \vee Q(x)) \wedge (\neg R(z,y) \vee Q(x))).
\end{aligned}$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## 4.3 Eksistenssikvanttorien eliminointi

**Esimerkki.** Tarkastellaan kahta kokonaislukuja koskevaa väittämää:

1. Summafunktiolla on vasen identiteetti:

   $$\exists x \forall y (x + y = y).$$

   Identiteettialkio voidaan nimetä vakiosymbolilla $0$:

   $$\forall y (0 + y = y).$$

2. Jokaisella kokonaisluvulla on vastaluku:

   $$\forall x \exists y (x + y = 0).$$

   Vastalukufunktio voidaan nimetä funktiosymbolilla $-$:

   $$\forall x (x + -(x) = 0).$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Eksistenssikvanttorien eliminointi yleisessä tapauksessa

Olkoon $\vec{Q}x\phi$ prenex-normaalimuodossa ja $\exists x$ kvanttorisekvenssin $\vec{Q}x$ ensimmäinen eksistenssikvanttori.

1. Jos kvanttori $\exists x$ on sekvenssissä $\vec{Q}x$ vieläpä ensimmäisenä, poistetaan $\exists x$ sekvenssistä ja korvataan näin syntyvät muuttujan $x$ vapaat esintymät jollain uudella vakiolla $c$.
2. Jos kvanttoria $\exists x$ esiintyy sekvenssissä $\vec{Q}x$ universaalikvanttorien $\forall y_1 \cdots \forall y_n$ jälkeen, poistetaan kvanttori $\exists x$ ja korvataan näin syntyvät muuttujan $x$ vapaat esiintymät termillä $f(y_1, \ldots, y_n)$, missä $f$ on uusi funktiosymboli.

**Huomio.** Jälkimmäisessä tapauksessa funktion $f$ avulla kuvataan muuttujan $x$ *mahdollinen* riippuvuus muuttujista $y_1, \ldots, y_n$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



- Eksistenssikvanttorien eliminointia kutsutaan kehittäjänsä mukaan Skolemoinniksi ja vastaavasti ko. prosessissa valittavia uusia vakio- ja funktiosymboleita *Skolem-vakioiksi ja -funktioiksi*.

**Esimerkki.** Suoritetaan Skolemointi seuraaville lauseille:

$$\begin{aligned}
\exists x P(x) &\rightsquigarrow P(c) \\
\exists x \forall y \exists x P(x,y) &\rightsquigarrow \forall y P(f(y),y) \\
\forall x \exists y (P(x,y) \wedge Q(y,x)) &\rightsquigarrow \forall x\ (P(x,f(x)) \wedge Q(f(x),x)) \\
\forall x \exists y \forall z \exists w P(w,z,y,x) &\rightsquigarrow \forall x \forall z P(g(x,z),z,f(x),x) \\
\exists x \exists y \forall z P(x,y,z,x) &\rightsquigarrow \forall z P(c_1,c_2,z,c_1) \\
\forall x \exists y \exists z P(x,y,z) &\rightsquigarrow \forall x P(x,f_1(x),f_2(x))
\end{aligned}$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Skolemoinnin loogiset ominaisuudet

**Väite.** Prenex-normaalimuodossa oleva lause $\varphi$ on toteutuva $\iff$ lauseen $\varphi$ skolemoitu muoto $\varphi'$ on toteutuva.

**Huomio.** Prenex-normaalimuodossa olevan lauseen $\varphi$ skolemoitu muoto $\varphi'$ **ei välttämättä ole** loogisesti ekvivalentti lauseen $\varphi$ kanssa.

**Esimerkki.** Lause $\exists x P(x)$ ja sen skolemoitu muoto $P(c)$.

Nyt $\models P(c) \rightarrow \exists x P(x)$, mutta $\not\models \exists x P(x) \rightarrow P(c)$.

Vastamalli $\mathcal{S}$: universumi $U = \{1, 2\}$, $c^{\mathcal{S}} = 1$ ja $P^{\mathcal{S}} = \{2\}$.
Nyt $\mathcal{S} \models \exists x P(x)$, mutta $\mathcal{S} \not\models P(c)$.

Täten myös $\not\models \exists x\ P(x) \leftrightarrow P(c)$ ja edelleen $\exists x P(x) \not\equiv P(c)$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## 4.4 Lauseiden klausuulimuoto predikaattilogiikassa

Mille tahansa lauseelle voidaan hakea klausuulimuoto seuraavasti:

1. Haetaan prenex-normaalimuoto.
2. Muunnetaan tämä konjunktiiviseen normaalimuotoon.
3. Tarvittaessa poistetaan eksistenssikvanttorit Skolemoimalla.
4. Kirjoitetaan klausuuliesitys (joukko literaalien joukkoja).

**Esimerkki.** Klausuuliesitys lauseelle $\forall x(\neg(P(x) \to \forall y Q(x,y)) \vee R(x))$:

$$\rightsquigarrow \quad \forall x \exists z((P(x) \wedge \neg Q(x,z)) \vee R(x)) \qquad (1)$$

$$\rightsquigarrow \quad \forall x \exists z((P(x) \vee R(x)) \wedge (\neg Q(x,z) \vee R(x))) \qquad (2)$$

$$\rightsquigarrow \quad \forall x((P(x) \vee R(x)) \wedge (\neg Q(x,f(x)) \vee R(x))) \qquad (3)$$

$$\rightsquigarrow \quad \{\{P(x),R(x)\},\{\neg Q(x,f(x)),R(x)\}\}. \qquad (4)$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Huomio.** Jos lause on muodoltaan *konjunktio* $\phi_1 \wedge \cdots \wedge \phi_n$, on mahdollista saattaa konjunktion jäsenet $\phi_1, \ldots, \phi_n$ klausuulimuotoon *erikseen*. Muista myös, että $\neg(\phi_1 \wedge \cdots \wedge \phi_n \to \phi) \equiv \phi_1 \wedge \cdots \wedge \phi_n \wedge \neg\phi$.

**Esimerkki.** $\forall x P(x) \wedge \exists x Q(x)$. Konjunktion jäsenille saadaan klausuuliesitykset $\{\{P(x)\}\}$ ja $\{\{Q(c)\}\}$ ja näinollen koko lauseen klausuuliesitykseksi näiden unioni $\{\{P(x)\},\{Q(c)\}\}$.

**Huomio.** Existenssikvanttorit kannattaa tuoda ulos lauserakenteesta ennen universaalikvanttoreita (mikäli mahdollista).

**Esimerkki.** Siis $\forall x P(x) \vee \exists x Q(x)$ kirjoitetaan muotoon $\exists x \forall y(P(y) \vee Q(x))$ eikä muotoon $\forall x \exists y(P(x) \vee Q(y))$.

Näin saatetaan välttää Skolem-funktioiden käyttöönotto tai ainakin vähennetään Skolem-funktioiden argumenttien lukumäärää
$\Longrightarrow$ klausuulimuodosta tulee rakenteeltaan yksinkertaisempi.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



# 5 Tietämyksen esittämisestä

- Tietämyksen esittäminen predikaattilogiikalla
- Ohjeita predikaattien määrittelemiseen
- Nimien yksikäsitteisyys ja kattavuus
- Negatiiviset ehdot ja johtopäätökset

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 5.1 Tietämyksen esittäminen predikaattilogiikalla

Annettuun järjestelmään liittyvää tietämystä voidaan esittää valitsemalla

- sopiva predikaattilogiikan aakkosto (joukot $\mathcal{P}$, $\mathcal{C}$ ja $\mathcal{F}$) ja
- vastaavaan kieleen $\mathcal{L}$ perustuva lausejoukko $\Sigma \subseteq \mathcal{L}$, jonka lauseet määrittelevät järjestelmän ominaisuudet.

Tarkastellaan määritelmiä vastaavan lausejoukon $\Sigma \subseteq \mathcal{L}$ *loogisten seurausten* joukkoa $\mathrm{Cn}(\Sigma) = \{\phi \in \mathcal{L} \mid \phi \text{ on lause ja } \Sigma \models \phi\}$. Nyt

- $\Sigma$ muodostaa järjestelmää koskevan *eksplisiittisen tietämyksen* ja
- joukon $\mathrm{Cn}(\Sigma) - \Sigma$ lauseet ovat *implisiittistä tietämystä* eli väittämiä, jotka voidaan päätellä eksplisiittisestä tietämyksestä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Kuvataan sähköverkkoa predikaattilogiikan lausein:

$$\Sigma = \{\, \forall x \forall y \forall z(\text{johto}(x,y) \wedge \text{jännite}(x,z) \rightarrow \text{jännite}(y,z)),$$
$$\text{johto}(\text{p1},\text{p2}),\ \text{johto}(\text{p2},\text{p3}),\ \text{jännite}(\text{p1},220)\,\}.$$

Nyt esim. lause $\text{jännite}(\text{p3},220)$ on implisiittistä tietämystä:

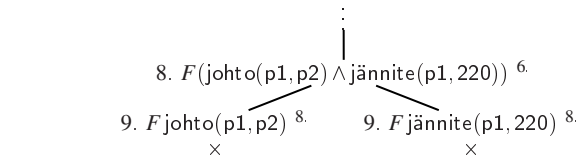
1. $T\ \forall x \forall y \forall z(\text{johto}(x,y) \wedge \text{jännite}(x,z) \rightarrow \text{jännite}(y,z))$
2. $T\ \text{johto}(\text{p1},\text{p2})$
3. $T\ \text{johto}(\text{p2},\text{p3})$
4. $T\ \text{jännite}(\text{p1},220)$
5. $F\ \text{jännite}(\text{p3},220)$
6. $T\ \text{johto}(\text{p1},\text{p2}) \wedge \text{jännite}(\text{p1},220) \rightarrow \text{jännite}(\text{p2},220)$ 1. $x/\text{p1}, y/\text{p2}, z/220$
7. $T\ \text{johto}(\text{p2},\text{p3}) \wedge \text{jännite}(\text{p2},220) \rightarrow \text{jännite}(\text{p3},220)$ 1. $x/\text{p2}, y/\text{p3}, z/220$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

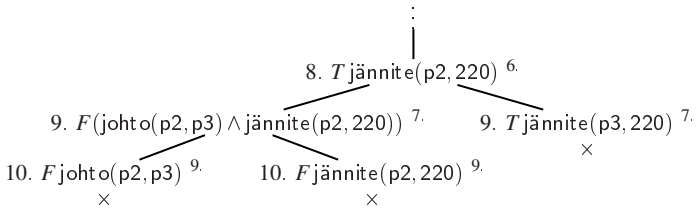


Hajoitettaessa polun 6. solmu taulu jakautuu kahteen haaraan:

- Vasen haara ($\tau_1$):




- Oikea haara ($\tau_2$):




© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Tietämyksen esittämisen problematiikkaa (kertaus)**

- Kaikki struktuurit ovat tyhjän lausejoukon $\emptyset$ malleja, joten $\text{Cn}(\emptyset)$ on itse asiassa pätevien lauseiden joukko.
- Enemmän lauseita $\Rightarrow$ vähemmän malleja $\Rightarrow$ enemmän loogisia seurauksia: siis $\Sigma_1 \subseteq \Sigma_2 \Rightarrow \text{Cn}(\Sigma_1) \subseteq \text{Cn}(\Sigma_2)$ (*monotonisuus*).
- Jos lausejoukko tulee ristiriitaiseksi, sillä ei ole yhtään mallia ja kaikki lauseet ovat tällöin lausejoukon loogisia seurauksia.

$\Longrightarrow$ Tavoitteena rajata predikaattilogiikan lausejoukolla $\Sigma \subseteq \mathcal{L}$ mallien joukko siten, että saadaan halutut loogiset seuraukset.

**Esimerkki.** Jos $\Sigma = \{\forall x(P(x) \rightarrow Q(x))\}$, niin $\Sigma \not\models Q(a)$.

- Vastamalli $\mathcal{S}$: universumi $U = \{1\}$, $a^{\mathcal{S}} = 1$ ja $P^{\mathcal{S}} = Q^{\mathcal{S}} = \emptyset$.
- Esim. lisäämällä $P(a)$ saadaan $\Sigma' = \Sigma \cup \{P(a)\}$, jolle $\Sigma' \models Q(a)$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 5.2 Ohjeita predikaattien määrittelemiseen

- Tavoitteena kirjoittaa predikaatille $P$ määritelmä joidenkin muiden predikaattien avulla.
- Mielivaltainen predikaattilogiikan kaava $\phi$ voidaan saattaa muotoon

  $$Q_1 x_1 Q_2 x_2 \cdots Q_n x_n \psi$$

  missä kukin kvanttori $Q_i$ on joko $\forall$ tai $\exists$, ja kaava $\psi$ on konjunktiivisessa normaalimuodossa eikä sisällä kvanttoreita.
- Yllä $\psi = \psi_1 \wedge \cdots \wedge \psi_m$, missä kukin $\psi_i$ on literaalien disjunktio

  $$\neg Q_1(\vec{t}_1) \vee \cdots \vee \neg Q_k(\vec{t}_k) \vee P_1(\vec{s}_1) \vee \cdots \vee P_l(\vec{s}_l)$$
  $$\equiv \quad Q_1(\vec{t}_1) \wedge \cdots \wedge Q_k(\vec{t}_k) \rightarrow P_1(\vec{s}_1) \vee \cdots \vee P_l(\vec{s}_l).$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

- Predikaattilogiikalla annetut määritelmät ovat korkeintaan yhtä monimutkaisia kuin edellä kuvattu normaalimuoto.
- Jos jokainen kvanttori $Q_i$ on universaalikvanttori $\forall$, $m = 1$ ja $l = 1$, saamme erikoistapauksena muotoa

$$\forall x_1 \forall x_2 \cdots \forall x_n (Q_1(\vec{t}_1) \wedge \cdots \wedge Q_k(\vec{t}_k) \rightarrow P(\vec{t}))$$

  olevia lauseita, missä predikaattien $Q_1, \ldots, Q_k$ ja $P$ argumentteina on vakiosymboleista, muuttujista $x_1, \ldots, x_n$ ja funktiosymboleista rakentuvien termien jonoja $\vec{t}_1, \ldots, \vec{t}_k$ ja $\vec{t}$.
- Määritelmiä voidaan usein kirjoittaa tähän muotoon: mietitään millä ehdoilla $Q_1(\vec{t}_1), \ldots, Q_k(\vec{t}_k)$ voidaan päätellä predikaattia $P$ koskeva väittämä $P(\vec{t})$. Ko. muotoa olevia lauseita saatetaan tarvita useita.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



- Predikaatin $P \in \mathcal{P}_n$ määrittelevää lausejoukkoa $\Sigma_P$ voidaan pitää riittävänä, jos kaikille muuttujattomille termeille $t_1, \ldots, t_n$ pätee: jono $\langle t_1, \ldots, t_n \rangle$ kuuluu predikaatin $P$ tarkoittamaan relaatioon $\Longleftrightarrow$

$$\Sigma_P \models P(t_1, \ldots, t_n).$$

- Tälläinen *positivistinen* määritelmä ei ole välttämättä *täydellinen* eli määritelmä ei takaa, että $\Sigma_P \models \neg P(t_1, \ldots, t_n)$ mikäli jono $\langle t_1, \ldots, t_n \rangle$ ei kuulu predikaatin $P$ tarkoittamaan relaatioon.

**Esimerkki.** Tarkastellaan määritelmää $\Sigma$, jossa on seuraavat lauseet: linkki$(a,b)$, linkki$(b,c)$, linkki$(d,e)$, $\forall x$ yhteys$(x,x)$, $\forall x \forall y$(linkki$(x,y) \rightarrow$ yhteys$(x,y)$), $\forall x \forall y$(yhteys$(x,y) \rightarrow$ yhteys$(y,x)$) ja $\forall x \forall y \forall z$(linkki$(x,y) \wedge$ yhteys$(y,z) \rightarrow$ yhteys$(x,z)$).

Nyt esim. $\Sigma \models$ yhteys$(c,a)$, mutta $\Sigma \not\models \neg$yhteys$(a,e)$!

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Olkoon annettuna predikaatit

1. sairastaa$(x)$ = "henkilö $x$ on sairas" ja
2. tapaa$(x,y)$ = "henkilö $x$ tapaa henkilön $y$".

Tarkoituksena on määritellä näiden avulla predikaatti

tartuntavaarassa$(x)$ = "henkilö $x$ on tartuntavaarassa".

Kysymys: millä ehdoilla jonkin henkilö on tartuntavaarassa?

1. Jos henkilö tapaa jonkun sairaan henkilön.
2. Jos henkilö tapaa jonkun toisen tartuntavaarassa olevan henkilön.

Yritetään kirjoittaa nämä edellä esitetyn mukaisesti muotoon

$$\forall x \forall y \ldots (Q_1(\vec{t}_1) \wedge \ldots \wedge Q_k(\vec{t}_k) \rightarrow \text{tartuntavaarassa}(x)).$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



- Näin saadaan muodostettua lauseet

$$\forall x \forall y(\text{tapaa}(x,y) \wedge \text{sairastaa}(y) \rightarrow \text{tartuntavaarassa}(x)) \text{ ja}$$
$$\forall x \forall y(\text{tapaa}(x,y) \wedge \text{tartuntavaarassa}(y) \rightarrow \text{tartuntavaarassa}(x)).$$

- Kysymyksessä on itse asiassa tartuntavaarassa-predikaatin induktiivinen (rekursiivinen) määritelmä. Lauseista ensimmäinen vastaa perustapausta ja jälkimmäinen induktioaskelta.
- Lisäksi voidaan todeta tapaamiset symmetrisiksi:

$$\forall x \forall y(\text{tapaa}(x,y) \rightarrow \text{tapaa}(y,x)).$$

- Universumin voidaan ajatella koostuvan pelkästään henkilöistä (eli edellä annetut lauseet puhuvat kaikista henkilöistä).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## Määritelmien käyttö konkreettisessa päättelyssä

**Esimerkki.** Lisätään edellä johdettuun tartuntavaarassa-predikaatin määritelmään tietokanta, jossa kuvataan tapaamiset ja sairastamiset:

Näin saadaan lausejoukko

$$\Sigma = \{\ \forall x \forall y(\text{tapaa}(x,y) \wedge \text{sairastaa}(y) \rightarrow \text{tartuntavaarassa}(x)),$$
$$\forall x \forall y(\text{tapaa}(x,y) \wedge \text{tartuntavaarassa}(y) \rightarrow \text{tartuntavaarassa}(x)),$$
$$\forall x \forall y(\text{tapaa}(x,y) \rightarrow \text{tapaa}(y,x)),$$
$$\text{tapaa}(\text{Lyyli}, \text{Hemmo}),\ \text{tapaa}(\text{Lyyli}, \text{Erkki}),\ \text{sairastaa}(\text{Erkki})\ \}.$$

- Kyseisessä asetelmassa saadaan $\Sigma \models \text{tartuntavaarassa}(\text{Lyyli}) \wedge \text{tartuntavaarassa}(\text{Hemmo})$.
- Kokeile tämän osoittamista semanttisella taululla!

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Suoritetaan vastaava päättely OTTER-teoreemantodistimella. Tarvittava syötetiedosto:

```
set(auto).
formula_list(usable).

% Section A: database
tapaa(lyyli,hemmo). tapaa(lyyli,erkki). sairastaa(erkki).

% Section B: definitions
all x y (tapaa(x,y) -> tapaa(y,x)).
all x y (tapaa(x,y) & sairastaa(y) -> tartuntavaarassa(x)).
all x y (tapaa(x,y) & tartuntavaarassa(y) -> tartuntavaarassa(x)).

% Section C: negation of the query
-(tartuntavaarassa(lyyli) & tartuntavaarassa(hemmo)).

end_of_list.
```

- OTTER pystyy osoittamaan lausejoukon helposti ristiriitaiseksi.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## Tyypitetyt kvanttorit

- Usein on mielekästä ajatella universumin koostuvan tyypiltään erilaisista alkioista.
- Tällöin syntyy tarve rajata kvantifiointia koskemaan ainoastaan tiettyä tyyppiä $T$ olevia alkioita seuraavaan tapaan:
$$\forall x \in T : \phi(x) \text{ ja } \exists x \in T : \phi(x).$$
- Tyyppi $T$ voidaan esittää yksipaikkaisen predikaatin avulla:
$$T(x) = \text{“alkio } x \text{ on tyyppiä } T\text{”}.$$
- Tyypitetyt kvanttorit ilmaistaan predikaattilogiikassa seuraavasti:
$$\forall x(T(x) \rightarrow \phi(x)) \text{ ja } \exists x(T(x) \wedge \phi(x)).$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Lisätään edellisen esimerkkiin tyyppipredikaatteja.

- Määritellään predikaatit henkilöiden ja tautien erottelemiseksi: $\text{henkilö}(x) = \text{“}x \text{ on henkilö”}$ ja $\text{tauti}(x) = \text{“}x \text{ on tauti”}$.
- Määritellään predikaatit ilman tyyppi-informaatiota:
  – $\text{tapaa}(x,y) = \text{“}x \text{ tapaa } y\text{:n”}$,
  – $\text{sairastaa}(x,y) = \text{“}x \text{ sairastaa } y\text{:tä”}$ ja
  – $\text{tartuntavaarassa}(x,y) = \text{“}x \text{ on vaarassa sairastua } y\text{:hyn”}$.
- Lauseet saadaan nyt seuraavaan muotoon:

$$\forall x \forall y \forall z(\text{henkilö}(x) \wedge \text{henkilö}(y) \wedge \text{tapaa}(x,y) \wedge$$
$$\text{tauti}(z) \wedge \text{sairastaa}(y,z) \rightarrow \text{tartuntavaarassa}(x,z)) \text{ ja}$$
$$\forall x \forall y \forall z(\text{henkilö}(x) \wedge \text{henkilö}(y) \wedge \text{tapaa}(x,y) \wedge \text{tauti}(z) \wedge$$
$$\text{tartuntavaarassa}(y,z) \rightarrow \text{tartuntavaarassa}(x,z)).$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

- Tyypeillä voi olla erilaisia suhteita:
  - Erillisyys: $\forall x \neg(\text{henkilö}(x) \wedge \text{tauti}(x))$.
  - Kattavuus: $\forall x(\text{henkilö}(x) \vee \text{tauti}(x))$.
  - Alityyppi: $\forall x(\text{rokko}(x) \rightarrow \text{tauti}(x))$.
- Yksi mahdollisuus on tyypittää predikaatit erikseen:

$$\forall x \forall y(\text{tapaa}(x,y) \rightarrow \text{henkilö}(x) \wedge \text{henkilö}(y))$$
$$\forall x \forall y(\text{sairastaa}(x,y) \rightarrow \text{henkilö}(x) \wedge \text{tauti}(y))$$

- Tällöin varsinainen tartuntavaarassa-predikaatin määritelmä voidaan kirjoittaa yksinkertaisemmin ilman tyyppi-informaatiota:

$\forall x \forall y \forall z(\text{tapaa}(x,y) \wedge \text{sairastaa}(y,z) \rightarrow \text{tartuntavaarassa}(x,z))$ ja
$\forall x \forall y \forall z(\text{tapaa}(x,y) \wedge \text{tartuntavaarassa}(y) \rightarrow \text{tartuntavaarassa}(x,z))$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Muodoltaan monimutkaisempia määritelmiä

- Edellä otettiin lähtökohdaksi muotoa

$$\forall x_1 \forall x_2 \cdots \forall x_n(Q_1(\vec{t}_1) \wedge \cdots \wedge Q_k(\vec{t}_k) \rightarrow P(\vec{t}))$$

  olevat määritelmät. Näiden ilmaisuvoima ei ole aina riittävä.
- Joissain tilanteissa tarvitaan eksistentiaalista kvantifiointia:

$$\forall x(\text{solmu}(x) \rightarrow \exists y(\text{väri}(y) \wedge \text{väritetty}(x,y)))$$
$$\equiv \quad \forall x \exists y(\text{solmu}(x) \rightarrow \text{väri}(y) \wedge \text{väritetty}(x,y)).$$

- Implikaation seurauksena voi olla myös atomien disjunktio $P_1(\vec{s}_1) \vee \cdots \vee P_l(\vec{s}_l)$ pelkän atomin $P(\vec{t})$ sijaan:

$$\forall x(\text{bitti}(x) \rightarrow \text{nolla}(x) \vee \text{yksi}(x)).$$

**Huomio.** Edellä oli keskeistä vaihtoehtoisuuden ilmaiseminen.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 5.3 Nimien yksikäsitteisyys ja kattavuus

- Rajoitutaan jatkossa predikaattilogiikan kieliin $\mathcal{L}$, joissa ei ole funktiosymboleita ja ainoastaan äärellinen määrä vakiosymboleita.
- Predikaattilogiikassa struktuurin $\mathcal{S}$ määritelmä ja tapa jolla vakiosymbolit tulkitaan $\mathcal{S}$:ssa mahdollistavat, että
  1. jokin universumin $U$ alkio $a \in U$ on useamman vakion $c_1, \ldots, c_n$ $(n > 1)$ nimeämä: $c_1^{\mathcal{S}} = \ldots = c_n^{\mathcal{S}} = a$.
  2. jokin universumin $U$ alkio $a \in U$ ei ole minkään vakion nimeämä (eli kaikille vakiosymboleille $c$ pätee $c^{\mathcal{S}} \neq a$).
- Tietämyksen esittämisen kannalta tällainen mahdollisuus muodostuu usein jopa turhaksi vapausasteeksi.
- Nimeäminen voidaan pakottaa yksikäsitteiseksi lauseita lisäämällä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Nimien yksikäsitteisyys

- Vastaava käsite englanniksi on *unique names assumption* (UNA).
- Kun kielessä on äärellinen määrä vakiosymboleita $c_1, \ldots, c_n$, riittää lisätä muotoa

$$\neg(c_i = c_j)$$

  olevat lauseet, missä $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, n\}$ ja $i < j$.
- Lauseita tarvitaan neliöllinen määrä (yhteensä $\frac{n^2-n}{2}$ kappaletta).

**Esimerkki.** Olkoon kielessä $\mathcal{L}$ vakiosymbolit Lyyli, Hemmo ja Erkki. Yksikäsitteisten nimien oletus ilmaistaan seuraavasti:

$$\neg(\text{Lyyli} = \text{Hemmo}),\ \neg(\text{Lyyli} = \text{Erkki}) \text{ ja } \neg(\text{Hemmo} = \text{Erkki}).$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Tarkastellaan lausejoukon

$$\Sigma_{\text{UNA}} = \{\neg(\text{Lyyli} = \text{Hemmo}), \neg(\text{Lyyli} = \text{Erkki}), \neg(\text{Hemmo} = \text{Erkki})\}$$

malleja $\mathcal{S}_i$, kun universumina $U_i$ on joukko henkilöitä $h_1, h_2, \ldots$ .

| $U_i$ | Lyyli$^{\mathcal{S}_i}$ | Hemmo$^{\mathcal{S}_i}$ | Erkki$^{\mathcal{S}_i}$ |
|---|---|---|---|
| $\{h_1, h_2, h_3\}$ | $h_1$ | $h_2$ | $h_3$ |
| $\{h_1, h_2, h_3\}$ | $h_1$ | $h_3$ | $h_2$ |
| $\{h_1, h_2, h_3\}$ | $h_2$ | $h_1$ | $h_3$ |
| $\{h_1, h_2, h_3\}$ | $h_2$ | $h_3$ | $h_1$ |
| $\{h_1, h_2, h_3\}$ | $h_3$ | $h_1$ | $h_2$ |
| $\{h_1, h_2, h_3\}$ | $h_3$ | $h_2$ | $h_1$ |
| $\{h_1, h_2, h_3, h_4\}$ | $h_1$ | $h_2$ | $h_3$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |

$\Longrightarrow$ Universumissa oltava vähintään 3 henkilöä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## Nimien kattavuus

- Vastaava käsite englanniksi on *domain closure assumption* (DCA).
- Kun kielessä on äärellinen määrä vakiosymboleita $c_1, \ldots, c_n$, riittää lisätä seuraavaa muotoa oleva lause:

$$\forall x(x = c_1 \vee \cdots \vee x = c_n).$$

- Tarvittavan lauseen pituus riippuu lineaarisesti vakioiden lukumäärästä $n$.

**Esimerkki.** Edellisen esimerkin mukaisessa kielessä tarvitaan lause

$$\forall x(x = \text{Lyyli} \vee x = \text{Hemmo} \vee x = \text{Erkki}).$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Tarkastellaan lausejoukon

$$\Sigma_{\text{DCA}} = \{\forall x(x = \text{Lyyli} \vee x = \text{Hemmo} \vee x = \text{Erkki})\}$$

malleja $\mathcal{S}_i$, kun universumina $U_i$ on joukko henkilöitä $h_1, h_2, \ldots$ .

| $U_i$ | Lyyli$^{\mathcal{S}_i}$ | Hemmo$^{\mathcal{S}_i}$ | Erkki$^{\mathcal{S}_i}$ |
|---|---|---|---|
| $\{h_1\}$ | $h_1$ | $h_1$ | $h_1$ |
| $\{h_1, h_2\}$ | $h_1$ | $h_1$ | $h_2$ |
| $\{h_1, h_2\}$ | $h_1$ | $h_2$ | $h_1$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\{h_1, h_2, h_3\}$ | $h_1$ | $h_2$ | $h_3$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\{h_1, h_2, h_3\}$ | $h_3$ | $h_2$ | $h_1$ |

$\Longrightarrow$ Universumissa voi olla korkeintaan 3 henkilöä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Tarkastellaan vielä edeltävien lausejoukkojen unionin

$$\begin{aligned}\Sigma_{\text{UNA}} \cup \Sigma_{\text{DCA}} = \{\ & \neg(\text{Lyyli} = \text{Hemmo}), \neg(\text{Lyyli} = \text{Erkki}),\\ & \neg(\text{Hemmo} = \text{Erkki}),\\ & \forall x(x = \text{Lyyli} \vee x = \text{Hemmo} \vee x = \text{Erkki})\ \}\end{aligned}$$

malleja $\mathcal{S}_i$, kun universumina $U_i$ on joukko henkilöitä $h_1, h_2, \ldots$ .

| $U_i$ | Lyyli$^{\mathcal{S}_i}$ | Hemmo$^{\mathcal{S}_i}$ | Erkki$^{\mathcal{S}_i}$ |
|---|---|---|---|
| $\{h_1, h_2, h_3\}$ | $h_1$ | $h_2$ | $h_3$ |
| $\{h_1, h_2, h_3\}$ | $h_1$ | $h_3$ | $h_2$ |
| $\{h_1, h_2, h_3\}$ | $h_2$ | $h_1$ | $h_3$ |
| $\{h_1, h_2, h_3\}$ | $h_2$ | $h_3$ | $h_1$ |
| $\{h_1, h_2, h_3\}$ | $h_3$ | $h_1$ | $h_2$ |
| $\{h_1, h_2, h_3\}$ | $h_3$ | $h_2$ | $h_1$ |

$\Longrightarrow$ Universumissa on oltava täsmälleen 3 henkilöä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## 5.4 Negatiiviset ehdot ja johtopäätökset

- Tarkastellaan muotoa

$$\forall x_1 \forall x_2 \cdots \forall x_n (Q_1(\vec{t_1}) \wedge \cdots \wedge Q_k(\vec{t_k}) \rightarrow P(\vec{t}))$$

olevien määritelmien yleistämistä tapaukseen, missä sallitaan atomien $Q_i(\vec{t_i})$ lisäksi myös negatiivisia literaaleja $\neg Q_i(\vec{t_i})$.

- Negatiivinen ehto $\neg Q_i(\vec{t_i})$ voidaan muuntaa positiiviseksi vaihtoehdoksi $Q_i(\vec{t_i})$ seuraukselle $P(\vec{t})$.

**Esimerkki.**

$$\forall x(\neg \text{sairastaa}(x) \wedge \neg \text{tartuntavaarassa}(x) \rightarrow \text{turvassa}(x))$$
$$\equiv \quad \forall x(\text{sairastaa}(x) \vee \text{tartuntavaarassa}(x) \vee \text{turvassa}(x)).$$

- Jotta negatiiviset ehdot tulisivat määritellyiksi, määritelmistä tulisi seurata loogisesti $\neg Q_i(\vec{t_i})$ mikäli $Q_i(\vec{t_i})$ ei ole looginen seuraus.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Määritelmien täydellisyys

**Määritelmä.** Predikaatin $P \in \mathcal{P}_n$ määritelmä $\Sigma_P \subseteq \mathcal{L}$ on *täydellinen*, jos

$$\Sigma_P \models P(t_1, \ldots, t_n) \text{ tai } \Sigma_P \models \neg P(t_1, \ldots, t_n).$$

kaikille kielen $\mathcal{L}$ muuttujattomille termeille $t_1, \ldots, t_n$.

**Huomioita.**

- Jos predikaatin $P \in \mathcal{P}_n$ määritelmä $\Sigma_P$ on ristiriitainen, se on triviaalisti täydellinen: kaikille muuttujattomille termeille $t_1, \ldots, t_n$ pätee tällöin sekä $\Sigma_P \models P(t_1, \ldots, t_n)$ että $\Sigma_P \models \neg P(t_1, \ldots, t_n)$.
- Jos predikaatin $P \in \mathcal{P}_n$ määritelmä $\Sigma_P$ on täydellinen ja $\Sigma_P \not\models P(t_1, \ldots, t_n)$ joillekin muuttujattomille termeille $t_1, \ldots, t_n$, niin $\Sigma_P \models \neg P(t_1, \ldots, t_n)$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Tarkastellaan *muunnelmaa* tartuntavaara-esimerkistä:

$$\Sigma = \{\ \forall x \forall y(\text{tapaa}(x,y) \wedge \text{sairastaa}(y) \rightarrow \text{tartuntavaarassa}(x)),$$
$$\forall x(\neg \text{sairastaa}(x) \wedge \neg \text{tartuntavaarassa}(x) \rightarrow \text{turvassa}(x)),$$
$$\text{tapaa}(\text{Lyyli}, \text{Hemmo}),\ \text{tapaa}(\text{Lyyli}, \text{Erkki}),\ \text{sairastaa}(\text{Erkki})\ \}.$$

- Nyt $\Sigma \models \text{tartuntavaarassa}(\text{Lyyli})$, $\Sigma \not\models \text{tartuntavaarassa}(\text{Hemmo})$ ja $\Sigma \not\models \neg\text{tartuntavaarassa}(\text{Hemmo})$.
- Täten $\Sigma$ ei ole täydellinen määritelmä tartuntavaarassa-predikaatille.
- Jotta näin olisi, määritelmästä tulisi seurata loogisesti $\neg\text{tartuntavaarassa}(\text{Hemmo})$ ja $\neg\text{tartuntavaarassa}(\text{Erkki})$.
- Kyseinen määritelmä $\Sigma$ ei ole myöskään täydellinen muille ao. kielen predikaateille (tapaa, sairastaa, turvassa ja $=$).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Predikaatin määritelmän täydentäminen

**Esimerkki.** Täydennetään edellisen esimerkin predikaattien määritelmät.

1. Yhtäsuuruuspredikaatin osalta riittää todeta nimien yksikäsitteisyys:

$$\neg(\text{Lyyli} = \text{Hemmo}),\ \neg(\text{Lyyli} = \text{Erkki}),\ \neg(\text{Hemmo} = \text{Erkki}) \text{ ja}$$
$$\forall x(x = \text{Lyyli} \vee x = \text{Hemmo} \vee x = \text{Erkki}).$$

2. Predikaateille tapaa ja sairastaa voidaan kirjoittaa tiiviit esitykset yhtäsuuruuspredikaatin avulla:

$$\forall x(\text{sairastaa}(x) \leftrightarrow x = \text{Erkki}) \text{ ja}$$
$$\forall x(\text{tapaa}(x,y) \leftrightarrow (x = \text{Lyyli} \wedge y = \text{Hemmo}) \vee$$
$$(x = \text{Lyyli} \wedge y = \text{Erkki})).$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

3. Predikaattien tartuntavaarassa ja turvassa määritelmät voidaan kirjoittaa vastaavasti ekvivalensseiksi:

$$\forall x(\text{tartuntavaarassa}(x) \leftrightarrow \exists y(\text{tapaa}(x,y) \wedge \text{sairastaa}(y))) \text{ ja}$$
$$\forall x(\text{turvassa}(x) \leftrightarrow \neg\text{sairastaa}(x) \wedge \neg\text{tartuntavaarassa}(x)).$$

- Täydennetyillä määritelmillä on haluamamme loogiset seuraukset:

| sairastaa | tartuntavaarassa | turvassa |
|---|---|---|
| ¬sairastaa(Lyyli) | tartuntavaarassa(Lyyli) | ¬turvassa(Lyyli) |
| ¬sairastaa(Hemmo) | ¬tartuntavaarassa(Hemmo) | turvassa(Hemmo) |
| sairastaa(Erkki) | ¬tartuntavaarassa(Erkki) | ¬turvassa(Erkki) |

- Predikaattien täydentäminen ei valitettavasti tuota haluttua lopputulosta *rekursiivisten* määritelmien tapauksessa, kuten seuraavassa esimerkissä osoitetaan.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Tarkastellaan vastaavaa konstruktiota lausejoukolle

$$\Sigma = \{\ \text{tuntee1}(\text{Lyyli},\text{Lyyli}), \text{tuntee1}(\text{Hemmo},\text{Hemmo}),$$
$$\forall x\forall y(\text{tuntee1}(x,y) \rightarrow \text{tuntee2}(x,y)),$$
$$\forall x\forall y(\text{tuntee2}(y,x) \rightarrow \text{tuntee2}(x,y))\ \}.$$

- Rekursiivisesti määritellyn predikaatin tuntee2 tarkoituksena on täydentää predikaatti tuntee1 *symmetriseksi*.
- Täydennettynä määritelmät saadaan muotoon

$$\Sigma' = \{\ \neg(\text{Lyyli} = \text{Hemmo}),\quad \forall x(x = \text{Lyyli} \vee x = \text{Hemmo}),$$
$$\forall x\forall y(\text{tuntee1}(x,y) \leftrightarrow (x = \text{Lyyli} \wedge y = \text{Lyyli}) \vee$$
$$(x = \text{Hemmo} \wedge y = \text{Hemmo})),$$
$$\forall x\forall y(\text{tuntee2}(x,y) \leftrightarrow \text{tuntee1}(x,y) \vee \text{tuntee2}(y,x))\ \}.$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



- Yllättäen täydennetyistä määritelmistä ei seuraa loogisesti ¬tuntee2(Lyyli, Hemmo) eikä ¬tuntee2(Hemmo, Lyyli).
- Lausejoukolla $\Sigma'$ on seuraava epäintuitiivinen malli $\mathcal{S}$:
  Universumi $U = \{h_1, h_2\}$,
  $\text{Lyyli}^{\mathcal{S}} = h_1$, $\text{Hemmo}^{\mathcal{S}} = h_2$,
  $\text{tuntee1}^{\mathcal{S}} = \{\langle h_1,h_1\rangle, \langle h_2,h_2\rangle\}$ ja
  $\text{tuntee2}^{\mathcal{S}} = \{\langle h_1,h_1\rangle, \langle h_1,h_2\rangle, \langle h_2,h_1\rangle, \langle h_2,h_2\rangle\}$.
- Kyseinen struktuuri $\mathcal{S}$ on vastamalli, koska
  $\mathcal{S} \not\models \neg\text{tuntee2}(\text{Lyyli},\text{Hemmo})$ ja $\mathcal{S} \not\models \neg\text{tuntee2}(\text{Hemmo},\text{Lyyli})$.

**Huomio.** Tentissä eikä myöskään 3. kotitehtävässä ei edellytetä täydellisten määritelmien kirjoittamista predikaateille (ellei tätä sitten erikseen jossain yksinkertaisessa tapauksessa pyydetä).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



# 6 Herbrandin teoreema

- Herbrand-universumit
- Herbrand-struktuurit ja -mallit
- Herbrandin teoreema
- Lause- ja predikaattilogiikan suhteesta

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## 6.1 Herbrand-universumit

**Määritelmä.** Predikaattilogiikan kielen $\mathcal{L}$ *Herbrand-universumi* $H$ on niiden muuttujattomien termien $t$ joukko, jotka ovat muodostettavissa kielen $\mathcal{L}$ vakio- ja funktiosymboleista.

**Esimerkki.** Olkoon kielessä $\mathcal{L}$ ainoastaan yksi vakiosymboli $c$ ja yksi funktiosymboli $f \in \mathcal{F}_2$.

Herbrand-universumiksi saadaan muuttujattomien termien joukko

$$H = \{c, f(c,c), f(f(c,c),c), f(c,f(c,c)), f(f(c,c),f(c,c)), \ldots\}.$$

**Huomio.** Jos kielessä $\mathcal{L}$ ei ole funktiosymboleita ja ainoastaan äärellinen määrä vakioita, Herbrand-universumi $H$ jää tällöin äärelliseksi.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



- Herbrand-universumi voidaan määritellä myös annetusta lausejoukosta $\Sigma$ lähtien.
- Jos lausejoukossa $\Sigma$ ei esiinny yhtään vakiosymbolia, Herbrand-universumiin valitaan ainakin yksi vakiosymboli $c$ (struktuurien määritelmän mukaan universumit ovat aina ei-tyhjiä).

**Määritelmä.** Lausejoukon $\Sigma$ Herbrand-universumi $H$ on niiden muuttujattomien termien $t$ joukko, jotka ovat muodostettavissa lausejoukossa $\Sigma$ esiintyvistä vakio- ja funktiosymboleista.

**Esimerkki.** Lausejoukon $\Sigma = \{\forall x P(x, f(x))\}$ Herbrand-universumi on

$$H = \{c, f(c), f(f(c)), \ldots\} = \{f^n(c) \mid n \geq 0\}.$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 6.2 Herbrand-struktuurit ja -mallit

**Määritelmä.** Kielen $\mathcal{L}$ *Herbrand-struktuuri* on struktuuri $\mathcal{H}$, jonka

1. universumina on kielen $\mathcal{L}$ Herbrand-universumi $H$,
2. jokaisen vakiosymbolin $c \in \mathcal{C}$ tulkintana $c^{\mathcal{H}}$ on $c$ itse,
3. jokaisen funktiosymbolin $f \in \mathcal{F}_n$ tulkintana on funktio $f^{\mathcal{H}}$, joka kuvaa muuttujattomat termit $t_1, \ldots, t_n$ muuttujattomaksi termiksi $f(t_1, \ldots, t_n)$, ja
4. jokaisen predikaattisymbolin $P \in \mathcal{P}_n$ tulkintana on $P^{\mathcal{H}} \subseteq H^n$ (yhtäsuuruuspredikaatille "=" tulkinta $=^{\mathcal{H}}$ on $\{\langle t,t\rangle \mid t \in H\}$).

- Lauseen $\phi \in \mathcal{L}$ totuusarvo Herbrand-struktuurissa $\mathcal{H}$ lasketaan predikaattilogiikan totuusmääritelmän mukaisesti.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Määritelmä.** Kielen $\mathcal{L}$ Herbrand-struktuuri $\mathcal{H}$ on

1. lauseen $\phi \in \mathcal{L}$ *Herbrand-malli* $\iff \mathcal{H} \models \phi$, ja
2. lausejoukon $\Sigma \subseteq \mathcal{L}$ *Herbrand-malli* $\iff \mathcal{H} \models \sigma$ kaikille $\sigma \in \Sigma$.

**Esimerkki.** Tarkastellaan lausejoukkoa

$$\Sigma = \{P(a),\ \forall x(P(x) \to Q(x)),\ \forall x(Q(x) \to Q(f(x)) \wedge R(x, f(x)))\}.$$

- Lausejoukon $\Sigma$ Herbrand-universumi on $H = \{f^n(a) \mid n \geq 0\}$.
- Muodostetaan Herbrand-struktuuri on $\mathcal{H}$, jonka universumina on $H$ siten, että jokainen muuttujaton termi $t \in H$ tulkitaan $t^{\mathcal{H}} = t$, ja
$$P^{\mathcal{H}} = \{a\},\ Q^{\mathcal{H}} = H \text{ ja } R^{\mathcal{H}} = \{\langle f^n(a), f^{n+1}(a)\rangle \mid n \geq 0\}.$$
- Kyseinen struktuuri $\mathcal{H}$ on lausejoukon $\Sigma$ Herbrand-malli.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Määritelmä.** Lausejoukon $\Sigma$ (kielen $\mathcal{L}$) *Herbrand-kanta* $B$ on niiden *atomisten lauseiden* joukko, jotka voidaan muodostaa lausejoukossa $\Sigma$ esiintyvistä (kielen $\mathcal{L}$) predikaattisymboleista ja vastaavan Herbrand-universumin $H$ muuttujattomista termeistä.

**Esimerkki.** Edellisen esimerkin tapauksessa Herbrand-kantana on $B = \{P(f^n(a)), Q(f^n(a)) \mid n \geq 0\} \cup \{R(f^n(a), f^m(a)) \mid n \geq 0, m \geq 0\}$.

- Tämä mahdollistaa Herbrand-struktuurien $\mathcal{H}$ määrittelemisen Herbrand-kannan $B$ osajoukkoina: jokaiselle $P \in \mathcal{P}_n$ pätee
$$P(t_1, \ldots, t_n) \in \mathcal{H} \iff \langle t_1, \ldots, t_n \rangle \in P^{\mathcal{H}}.$$
- Herbrand-struktuurille $\mathcal{H}$ voidaan antaa myös literaaliesitys:
$$lit(\mathcal{H}) = \{\, P(a), Q(a), \neg R(a,a), \neg P(f(a)), Q(f(a)), R(a, f(a)), \neg R(f(a), a), \neg R(f(a), f(a)), \ldots\}.$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 6.3 Herbrandin teoreema

- Rajoitutaan tarkastelemaan klausuulijoukkoja.
- Merkintä $C(x_1, \ldots, x_n)$ tarkoittaa muuttujat $x_1, \ldots, x_n$ sisältävää klausuulia $\{P_1(\vec{t_1}), \ldots, P_k(\vec{t_k}), \neg Q_1(\vec{s_1}), \ldots, \neg Q_l(\vec{s_l})\}$.
- Kausuuli $C(x_1, \ldots, x_n)$ vastaa universaalisti kvantifioitua lausetta $\forall x_1 \cdots \forall x_n \phi_C(x_1, \ldots, x_n)$, missä $\phi_C(x_1, \ldots, x_n)$ on klausuulin $C(x_1, \ldots, x_n)$ esitys literaalien disjunktiona.
- Klausuulijoukolle $S$ voidaan määritellä Herbrand-struktuurit samaan tapaan kuin lausejoukoillekin.
- Klausuulijoukko $S$ voidaan *instantioida* vastaavan Herbrand-universumin $H_S$ suhteen seuraavasti.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Määritelmä.** Klausuulijoukon $S$ *Herbrand-instanssien joukko* $S'$ koostuu muuttujattomista klausuuleista $C(t_1, \ldots, t_n)$, missä $C(x_1, \ldots, x_n) \in S$ ja muuttujattomat termit $t_1 \in H_S, \ldots, t_n \in H_S$.

- Mikäli $S$ ja $H_S$ ovat äärelliset, myös $S'$ on äärellinen.
- Joukko $S'$ voidaan tulkita lauselogiikan klausuulijoukoksi (atomisina lauseina Herbrand-kannan $B$ atomiset lauseet).

**Esimerkki.** Tarkastellaan klausuulijoukkoa $S = \{\{P(a)\}, \{\neg P(x), P(f(x))\}\}$.

1. Herbrand-universumi $H_S = \{a, f(a), f(f(a)), \ldots\}$.
2. Herbrand-instanssien joukko $S' = \{\{P(a)\}, \{\neg P(a), P(f(a))\}, \{\neg P(f(a)), P(f(f(a)))\}, \ldots\}$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Väite.** (Hebrandin teoreema). Olkoon $S$ joukko klausuuleita ja $S'$ sen Herbrand-instanssien joukko. Tällöin

1. $S$ on toteutumaton $\iff$ $S'$ on toteutumaton, ja
2. $S$ on toteutumaton $\iff$ $\exists$ joukon $S'$ äärellinen osajoukko $S''$, joka on toteutumaton.

**Huomioita.** Predikaattilogiikan tapauksessa voidaan täten rajoittua *syntaktisiin malleihin* (Herbrand-malleihin) mielivaltaisten mallien sijaan.

Herbrandin teoreema johtaa myös naiiviin proseduuriin klausuulijoukon $S$ toteutuvuusongelman ratkaisemiseksi:

(i) tuotetaan äärellinen Herbrand-instanssien osajoukko $S''$ ja

(ii) testataan, onko $S''$ on toteutumaton. Jos on, lopetetaan ja todetaan $S$ toteutumattomaksi. Muutoin jatketaan kohdasta (i).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## 6.4 Lauselogiikan ja predikaattilogiikan suhteesta

- Lauselogiikka on osa predikaattilogiikkaa:
  - Kaikki lauselogiikan konnektiivit ovat käytettävissä predikaattilogiikassa.
  - 0-paikkaiset predikaatit vastaavat atomisia lauseita.
- Lauselogiikan päätelmät ja loogiset ongelmat voidaan suorittaa/ratkoa sellaisenaan predikaattilogiikan puitteissa.
- Herbrandin teoreeman nojalla predikaattilogiikan päättely voidaan palauttaa lauselogiikan päättelyksi.
- Lauselogiikan ja predikaattilogiikan *ilmaisuvoimassa* (eli kyvyssä esittää tietämystä) on kuitenkin huomattava ero.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



- Ilmaisuvoimaeron ilmentyminen:
  - Äärellistä predikaattilogiikan lausejoukkoa saattaa vastata ääretön lauselogiikan lausejoukko.
  - Lauselogiikan ratkeavuus vs. predikaattilogiikan puoliratkeavuus.
- Rajoittamalla syntaksia sopivasti saadaan predikaattilogiikallekin ratkeavia (ja ilmaisuvoimaltaan heikompia) osajoukkoja.
  - Esim. jos $S$ on äärellinen ja siinä ei esiinny funktiosymboleja, sen Herbrand-instanssien joukko $S'$ jää äärelliseksi.
  - Tällöin $S'$:n toteutuvuus on selvitettävissä äärellisessä ajassa.

**Esimerkki.** Klausuulijoukon $S = \{\{P(a)\}, \{\neg P(x), P(b)\}\}$ Herbrand-universumi $H = \{a, b\}$ ja Herbrand-instanssien joukko $S' = \{\{P(a)\}, \{\neg P(a), P(b)\}, \{\neg P(b), P(b)\}\}$, joka voidaan nähdä lauselogiikan klausuulijoukkona $S' = \{\{P\}, \{\neg P, Q\}, \{\neg Q, Q\}\}$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



# 7 Unifikaatio

- Substituutiot
- Yleisimmät unifioijat
- Unifikaatioalgoritmi

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 7.1 Substituutiot

**Määritelmä.** *Substituutio* (tai *korvaus*) $\theta$ on äärellinen joukko

$$\{x_1/t_1, x_2/t_2, \ldots, x_n/t_n\},$$

missä $x_i$:t ovat muuttujia ja $t_i$:t korvaavia termejä siten, että

1. korvattavat muuttujat $x_1, \ldots, x_n$ ovat toisistaan eriävät ja
2. mikään korvaava termi $t_i$ ei ole muuttuja $x_i$ itse eli $t_i \neq x_i$.

Lisäksi erotetaan seuraavat erikoistapaukset:

- Jos korvaavat termit $t_i$ ovat muuttujattomia, $\theta$ on *muuttujaton*.
- Jos korvaavat termit $t_i$ ovat muuttujia, $\theta$ on *nimeämissubstituutio*.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Esimerkkeinä todettakoon

- tyhjä substituutio $\varepsilon = \{\}$,
- substituutio $\theta_1 = \{x/y, y/a, z/f(w)\}$,
- muuttujaton substituutio $\theta_2 = \{x/a, y/g(c,c)\}$ ja
- nimeämissubstituutio $\theta_3 = \{x/y, y/z, z/x\}$.

**Määritelmä.** Olkoon $E$ jokin *lauseke* (eli termi, atomikaava, literaali, klausuuli tms.) ja $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ substituutio.

Lauseke $E\theta$ on muutoin rakenteeltaan kuten $E$, paitsi että jokainen muuttujan $x_i$ esiintymä lausekkeessa $E$ on korvattu termillä $t_i$.

Jos lausekkeessa $E\theta$ ei esiinny muuttujia, kutsutaan lauseketta $E\theta$ lausekkeen $E$ *muuttujattomaksi instanssiksi*.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Olkoon lauseke $E = P(x, y, f(z), v, w)$ ja substituutio $\theta = \{x/y, y/x, z/x, v/f(z), w/g(f(y), c)\}$.

Tällöin $E\theta$ on $P(y, x, f(x), f(z), g(f(y), c))$.

**Määritelmä.** Olkoot $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ ja $\lambda = \{y_1/u_1, \ldots, y_m/u_m\}$ kaksi substituutiota.

Substituutioiden $\theta$ ja $\lambda$ *kompositio* $\theta\lambda$ määritellään joukkona

$$\{x_i/t_i\lambda \mid i \in \{1, \ldots, n\} \text{ ja } x_i \neq t_i\lambda\} \cup$$
$$\{y_i/u_i \mid i \in \{1, \ldots, m\} \text{ ja } y_i \notin \{x_1, \ldots, x_n\}\}.$$

**Huomio.** Määritelmän tavoitteena on saada aikaan kokonaisvaikutus $E(\theta\lambda) = (E\theta)\lambda$ mille tahansa lausekkeelle $E$.

**Esimerkki.** Substituutioiden $\theta = \{x/f(y), y/z\}$ ja $\lambda = \{x/a, y/b, z/y\}$. kompositio on $\{x/f(b), z/y\}$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 7.2 Yleisimmät unifioijat

**Määritelmä.** Olkoon $S = \{E_1, \ldots, E_n\}$ joukko lausekkeita. Substituutio $\theta$ on lausekejoukon $S$ *unifioija*, jos $E_1\theta = E_2\theta = \cdots = E_n\theta$.

Lausekejoukko $S$ on *unifioituva*, mikäli sillä on ainakin yksi unifioija.

**Esimerkki.** Tarkastellaan seuraavien joukkojen unifioituvuutta.

| Joukko $S$: | Unifioija $\theta$: |
|---|---|
| $\{P(x, f(a)), P(y, z)\}$ | $\{y/x, z/f(a)\}$ tai $\{x/y, z/f(a)\}$ |
| $\{P(x, f(x)), P(f(a), y)\}$ | $\{x/f(a), y/f(f(a))\}$ |
| $\{P(a), P(f(x))\}$ | ei unifioijaa |
| $\{P(x), P(f(x))\}$ | ei unifioijaa (termit aina äärellisiä) |

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Määritelmä.** Olkoon $\sigma$ lausekejoukon $S = \{E_1, \ldots, E_n\}$ unifioija.

Substituutiota $\sigma$ kutsutaan lausekejoukon $S$ *yleisimmäksi unifioijaksi*, mikäli jokainen $S$:n unifioija $\theta = \sigma\lambda$ jollekin substituutiolle $\lambda$.

- Vastaava käsite englanniksi on *most general unifier* (MGU).

**Esimerkki.** Joukon $S = \{P(x, f(y)), P(a, z)\}$ unifioijia ovat mm. $\theta = \{x/a, z/f(b), y/b\}$ ja $\sigma = \{x/a, z/f(y)\}$.

Näistä $\sigma$ on $L$:n yleisin unifioija, koska esim. $\theta = \sigma\{y/b\}$.

- Yleisimmät unifioijat ovat yksikäsitteisiä seuraavaan tapaan:

**Väite.** Olkoot $\theta$ ja $\sigma$ joukon $S$ yleisimpiä unifioijia. Tällöin on olemassa nimeämissubstituutio $\lambda$ siten että $S\theta\lambda = S\sigma$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## 7.3 Unifikaatioalgoritmi

- Tavoitteena laskea atomikaavojen joukolle $S \neq \emptyset$ yleisin unifioija $\sigma$.

**Määritelmä.** Olkoon $S$ ei-tyhjä joukko johonkin predikaattisymboliin $P$ perustuvia atomikaavoja $\{P(\vec{t_1}), \ldots, P(\vec{t_n})\}$.

1. Joukon $S$ *erokohta* on järjestyksessä ensimmäinen kohta (vasemmalta oikealle siirryttäessä), jossa joukon $S$ atomikaavojen merkkijonoesityksissä on jokin eroavaisuus.
2. Joukon $S$ *erojoukkoon* $D(S)$ kuuluvat atomikaavojen $P(\vec{t_1}), \ldots, P(\vec{t_n})$ erokohdasta alkavat termit $u_1, \ldots, u_n$.

**Esimerkki.** Joukon $S_1 = \{P(x,a), P(x,y)\}$ erojoukko $D(S_1) = \{a, y\}$. Joukon $S_2 = \{Q(g(x,y),y), Q(g(x,f(z)),x), Q(g(x,x),f(a))\}$ erojoukko $D(S_2) = \{y, f(z), x\}$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



*Unifikaatioalgoritmi* ei-tyhjälle atomikaavojen joukolle $S$:

1. Jos joukon $S$ atomikaavojen predikaattisymbolit eivät ole samat, totea, ettei $S$ unifioidu ja lopeta algoritmin suoritus.
2. Aseta $k := 0$, $S_k := S$ ja $\sigma_k := \varepsilon$.
3. Jos $S_k$ on yksialkioinen (ja siten jo unifioitunut) joukko, totea $S$ unifioituvaksi ja lopeta algoritmin suoritus.
4. Laske joukon $S_k$ erojoukko $D(S_k)$.
5. Jos $D(S_k)$:ssa on muuttuja $v_k$ ja termi $t_k$ siten, että $v_k$ ei esiinny $t_k$:ssa, jatka algoritmin suoritusta kohdasta 7.
6. Muutoin totea, ettei $S$ ole unifioituva, ja lopeta algoritmin suoritus.
7. Aseta $\sigma_{k+1} := \{v_k/t_k\}$ ja laske $S_{k+1} := S_k\{v_k/t_k\}$.
8. Aseta $k := k+1$ ja jatka algoritmin suorittamista kohdasta 3.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Väite.** Olkoon $S$ äärellinen ei-tyhjä joukko atomikaavoja.

- Jos $S$ on unifioituva, niin unifikaatioalgoritmin suoritus päättyy askeleen 3 kohdalla ja substituutioiden $\sigma_0, \sigma_1, \ldots, \sigma_k$ kompositio $\sigma = \sigma_0\sigma_1 \ldots \sigma_k$ on joukon $S$ yleisin unifioija
- Jos $S$ ei ole unifioituva, niin unifikaatioalgoritimin laskenta päättyy askeleessa 1 tai askeleessa 6.

**Esimerkki.** Lasketaan unifikaatioalgoritmilla joukon $S = \{P(x, f(x)), P(g(a), z)\}$ yleisin unifioija:

1. Predikaattisymbolit ovat samat, jatketaan.
2. $k = 0, S_0 = \{P(x, f(x)), P(g(a), z)\}, \sigma_0 = \varepsilon$.
3. $S_0$ ei ole yksialkoinen, jatketaan.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



4. $D(S_0) = \{x, g(a)\}$.
5. Valitaan muuttuja $v_0 = x$ ja termi $t_0 = g(a)$.
7. $\sigma_1 = \{x/g(a)\}, S_1 = \{P(g(a), f(g(a))), P(g(a), z)\}$.
8. $k = 1$.
3. $S_1$ ei ole yksialkioinen, jatketaan.
4. $D(S_1) = \{f(g(a)), z\}$.
5. Valitaan muuttuja $v_1 = z$ ja termi $t_1 = f(g(a))$.
7. $\sigma_2 = \{z/f(g(a))\}$, $S_2 = \{P(g(a), f(g(a)))\}$.
8. $k = 2$.
3. $S_2$ on yksialkioinen, joten $S$ on unifioituva.

Yleisin unifioija on $\sigma = \sigma_0\sigma_1\sigma_2 = \varepsilon\{x/g(a)\}\{z/f(g(a))\} = \{x/g(a), z/f(g(a))\}$ ja $S\sigma = \{P(g(a), f(g(a)))\} = S_2$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

# 8 Resoluutiosääntö ja -todistukset

- Resoluutiosääntö predikaattilogiikan tapauksessa
- Resoluutiotodistukset
- Ohjeita resoluutiotodistusten kirjoittamiseen
- Resoluutiostrategioita
- Automaattinen päättely (OTTER)

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 8.1 Resoluutiosääntö predikaattilogiikan tapauksessa

**Määritelmä.** Olkoot

$$C_1 = C_1' \sqcup \{P(\vec{t_1}), \ldots, P(\vec{t_n})\} \text{ ja } C_2 = C_2' \sqcup \{\neg P(\vec{u_1}), \ldots, \neg P(\vec{u_m})\}$$

kaksi klausuulia,

1. joissa *ei esiinny yhteisiä* muuttujia ja
2. joissa esiintyvien atomikaavojen joukko
$$\{P(\vec{t_1}), \ldots, P(\vec{t_n}), P(\vec{u_1}), \ldots, P(\vec{u_m})\}$$
on unifioituva (yleisimpänä unifioijana $\sigma$).

Klausuulien $C_1$ ja $C_2$ *yhdistelmä* on klausuuli $C_1'\sigma \cup C_2'\sigma$.

**Huomio.** Yllä käytetty merkintä $A \sqcup B$ tarkoittaa keskenään alkiovieraiden ($A \cap B = \emptyset$) joukkojen $A$ ja $B$ unionia $A \cup B$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Tarkastellaan seuraavia klausuuleja:

$$C_1 = \{Q(x), \neg R(y), P(x,y), P(f(z), f(z))\} \text{ ja}$$
$$C_2 = \{\neg N(u), \neg R(w), \neg P(f(a), f(a)), \neg P(f(w), f(w))\}.$$

Klausuuleissa ei esiinny yhteisiä muuttujia ja joukon

$$\{P(x,y), P(f(z), f(z)), P(f(a), f(a)), P(f(w), f(w))\}$$

yleisin unifioija on $\sigma = \{x/f(a), y/f(a), z/a, w/a\}$. Klausuulien yhdistelmäksi saadaan $\{Q(f(a)), \neg R(f(a)), \neg N(u), \neg R(a)\}$.

**Esimerkki.** (Faktorointi) Klausuleilla voi olla useita eri yhdistelmiä. Klausulien $\{P(x_1), P(y_1)\}$ ja $\{\neg P(x_2), \neg P(y_2)\}$ yhdistelmiä ovat mm.

- $\{P(x_1), \neg P(x_2)\}$ (joukolle $\{P(y_1), P(y_2)\}$ MGU $\sigma = \{y_2/y_1\}$) ja
- tyhjä klausuuli $\square$ (joukolle $\{P(x_1), P(y_1), P(x_2), P(y_2)\}$ MGU $\sigma = \{y_1/x_1, x_2/x_1, y_2/x_1\}$).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Logiikkaohjelmoinnissa (PROLOG) laskenta-askelet perustuvat *järjestettyjen klausuulien* väliseen resoluutioon.

**Määritelmä.** Olkoon $G = \{\neg B_1(\vec{u_1}), \ldots, \neg B_m(\vec{u_m})\}$ kyselyn negaatiota vastaava järjestetty *maaliklausuuli* ja $C = \{A(\vec{t}), \neg A_1(\vec{t_1}), \ldots, \neg A_n(\vec{t_n})\}$ järjestetty *ohjelmaklausuuli* (ohjelman sääntö) siten, että

1. klausuuleilla $G$ ja $C$ ei ole yhteisiä muuttujia ja
2. atomilla $A(\vec{t})$ ja *valintafunktion* $R$ määräämässä literaalissa $R(G) = \neg B_i(\vec{u_i})$ esiintyvällä atomilla $B_i(\vec{u_i})$ on yleisin unifioija $\theta$.

Klausuulien $G$ ja $C$ yhdistelmäksi saadaan järjestetty maaliklausuuli

$$G' = \{\ \neg B_1(\vec{u_1}), \ldots, \neg B_{i-1}(\vec{u_{i-1}}),$$
$$\neg A_1(\vec{t_1}), \ldots, \neg A_n(\vec{t_n}),$$
$$\neg B_{i+1}(\vec{u_{i+1}}), \ldots, \neg B_m(\vec{u_m})\ \}\theta.$$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

### Joitain PROLOGin erityispiirteitä

- Tyypillisessä PROLOG-toteutuksessa muuttujasymbolit erotetaan muista symboleista ison alkukirjamen perusteella.
- Literaalijoukkojen sijaan järjestetyt ohjelma- ja maaliklausuulit kirjoitetaan *sääntöinä* seuraavaan tapaan:

$$\begin{aligned} \{N(0)\} &\leadsto \texttt{n(0).} \\ \{N(s(x)), \neg N(x)\} &\leadsto \texttt{n(s(X)) :- n(X).} \\ \{\neg N(s(0)), \neg N(s(s(y)))\} &\leadsto \texttt{:- n(s(0)), n(s(s(Y))).} \end{aligned}$$

- Tyypillinen valintafunktio valitsee maaliklausuulin 1. atomin.

**Esimerkki.** Tyypillinen PROLOG-toteutus johtaa seuraavat maaliklausuulit: (1) `:- n(0), n(s(s(Y)))`, (2) `:- n(s(s(Y)))`, (3) `:- n(s(Y))`, (4) `:- n(Y)` ja (5) `:-` (tyhjä klausuuli).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 8.2 Resoluutiotodistukset

- Lähtökohtana on joukko klausuuleita $S$, jonka klausuuleista johdetaan uusia klausuuleita resoluutiosäännöllä.
- *Johtojen* ja *hylkäyksen* määritelmät säilyvät ennallaan, mutta resoluutioaskeleiden tulee täyttää resoluutiosäännön vaatimukset.
- Tarvittaessa klausuulien muuttujat tulee nimetä uudelleen.
- Resoluutio on myös predikaattilogiikan tapauksessa virheetön ja täydellinen menettely klausuulijoukon toteutuvuuden tutkimiseen.

**Väite.** Klausuulijoukolle $S$ löytyy hylkäys (eli klausuulijoukosta $S$ on johto $C_1, \ldots, C_n$ tyhjälle klausuulille $C_n = \Box$) $\iff$ $S$ on toteutumaton.

*Todistus.* Sivuutetaan.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Osoitetaan predikaattilogiikan lausejoukko

$$\Sigma = \{\forall x \exists y (P(x) \wedge P(y)),\ \forall x \forall y (P(x) \rightarrow \neg P(y))\}$$

toteutumattomaksi. Haetaan lauseille ensin klausuuliesitykset:

- $\forall x \exists y (P(x) \wedge P(y)) \leadsto \forall x (P(x) \wedge P(f(x))) \leadsto$ $S_1 = \{\{P(x)\}, \{P(f(x))\}\}$.
- $\forall x \forall y (P(x) \rightarrow \neg P(y))\} \leadsto \forall x \forall y (\neg P(x) \vee \neg P(y))\} \leadsto$ $S_2 = \{\{\neg P(x), \neg P(y)\}\}$.

| Hylkäys: | | |
|---|---|---|
| | 1. $\{P(x)\}$ | $S_1$ |
| | 2. $\{\neg P(z), \neg P(y)\}$ | $S_2\{x/z\}$ |
| | 3. $\Box$ | 1,2,MGU $\{x/y, z/y\}$ |

$\Longrightarrow$ $S_1 \cup S_2$ on toteutumaton $\Longrightarrow$ $\Sigma$ on toteutumaton.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Muiden loogisten ongelmien ratkominen

- Resoluutiolla voidaan selvittää lauseiden pätevyyttä ja loogista ekvivalenssia sekä tutkia lausejoukon loogisia seuraavuuksia.
- Koska Skolemointi ei säilytä loogista ekvivalenssia vaan toteutuvuuden, nämä tulee muuntaa toteutuvuusongelmiksi.

**Väite.** Olkoon $\phi$ ja $\psi$ lauseita ja $\Sigma$ lausejoukko.

1. Pätevyys: $\models \phi \iff \mathrm{KM}(\{\neg\phi\})$:lle löytyy hylkäys.
2. Ekvivalenssi: $\phi \equiv \psi \iff \mathrm{KM}(\{\neg(\phi \leftrightarrow \psi)\})$:lle löytyy hylkäys.
3. Looginen seuraavuus: $\Sigma \models \phi$ $\iff$ klausuulijoukolle $\mathrm{KM}(\Sigma \cup \{\neg\phi\})$ löytyy hylkäys.

Yllä $\mathrm{KM}(\Gamma)$ tarkoitaa lausejoukon $\Gamma$ klausuulimuotoa, mikä saadaan ottamalla yksittäisten lauseiden $\gamma \in \Gamma$ klausuulimuotojen unioni.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Osoitetaan resoluutiolla, että $\models \forall x P(x) \rightarrow \exists x P(x)$.

- Haetaan lauseen negaatiolle klausuulimuoto:

$$\begin{aligned} \neg(\forall x P(x) \rightarrow \exists x P(x)) &\leadsto \forall x P(x) \wedge \neg \exists x P(x) \\ &\leadsto \forall x P(x) \wedge \forall x \neg P(x) \\ &\leadsto \forall x \forall y (P(x) \wedge \neg P(y)). \\ &\leadsto S = \{\{P(x)\}, \{\neg P(y)\}\}. \end{aligned}$$

- Klausuuleista $\{P(x)\}$ ja $\{\neg P(y)\}$ saadaan tyhjä klausuuli $\square$ (MGU $\{x/y\}$) yhdellä resoluutioaskelella.
- Täten klausuulijoukolle $S$ on hylkäys
  $\Longrightarrow$ $S$ on toteutumaton
  $\Longrightarrow$ $\neg(\forall x P(x) \rightarrow \exists x P(x))$ on toteutumaton
  $\Longrightarrow$ $\forall x P(x) \rightarrow \exists x P(x)$ on pätevä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Osoitetaan lause $\exists x(E(x) \wedge K(x))$ lausejoukon

$$\Sigma = \{\forall x(I(x) \rightarrow E(x)), \exists x(I(x) \wedge K(x))\}$$

loogiseksi seuraukseksi. Haetaan tarvittavat klausuulimuodot:

$$\begin{aligned} \forall x(I(x) \rightarrow E(x)) &\leadsto \forall x(\neg I(x) \vee E(x)) \\ &\leadsto S_1 = \{\{\neg I(x), E(x)\}\}. \\ \exists x(I(x) \wedge K(x)) &\leadsto I(c) \wedge K(c) \\ &\leadsto S_2 = \{\{I(c)\}, \{K(c)\}\} \\ \neg\exists x(E(x) \wedge K(x)) &\leadsto \forall x \neg(E(x) \wedge K(x)) \\ &\leadsto \forall x(\neg E(x) \vee \neg K(x)) \\ &\leadsto S_3 = \{\{\neg E(x), \neg K(x)\}\} \end{aligned}$$

Kokonaisuutena saadaan siis klausuulijoukko $S = S_1 \cup S_2 \cup S_3$
$= \{\{\neg I(x), E(x)\}, \{I(c)\}, \{K(c)\}, \{\neg E(x), \neg K(x)\}\}$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



Hylkäys löydetään esimerkiksi seuraavasti:

| | | |
|---|---|---|
| 1. | $\{\neg I(x), E(x)\}$ | $S$ |
| 2. | $\{I(c)\}$ | $S$ |
| 3. | $\{K(c)\}$ | $S$ |
| 4. | $\{\neg E(y), \neg K(y)\}$ | $S\{x/y\}$ |
| 5. | $\{\neg I(y), \neg K(y)\}$ | 1,4,MGU $\{x/y\}$ |
| 6. | $\{\neg K(c)\}$ | 2,5,MGU $\{y/c\}$ |
| 7. | $\square$ | 3,6,MGU $\varepsilon$ |

- Yleisimpien unifioijien kompositio $\{x/y\}\{y/c\}\varepsilon = \{x/c, y/c\}$.
- Täten $S$ on toteutumaton
  $\Longrightarrow$ $\Sigma \cup \{\neg\exists x(E(x) \wedge K(x))\}$ on toteutumaton
  $\Longrightarrow$ $\exists x(E(x) \wedge K(x))$ on joukon $\Sigma$ looginen seuraus.

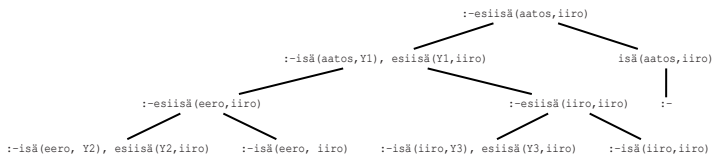
© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Tarkastellaan seuraavaa PROLOG-ohjelmaa:

```
isä(aatos,eero). isä(aatos,iiro). isä(oiva,aatos).

esiisä(X,Z) :- isä(X,Y), esiisä(Y,Z).
esiisä(X,Y) :- isä(X,Y).
```

Maaliklausuuli `:- esiisä(aatos, iiro)` johtaa seuraavaan hakuun:




$\Longrightarrow$ Koska tyhjän klausuulin `:-` johtaminen onnistuu, PROLOG-tulkki vastaa kyselyyn myöntävästi: `esiisä(aatos, iiro)` on johdettavissa.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## 8.3 Ohjeita resoluutiotodistusten kirjoittamiseen

- Muuttujien uudelleennimeäminen on hyvä suorittaa systemaattisesti (esimerkiksi alaindeksien avulla).
- Yksittäistä klausuulijoukon klausuulia saatetaan tarvita useita kertoja resoluutiotodistuksessa (jolloin muuttujien uudelleennimeäminen on välttämätöntä).
- Kirjoita yleisimmät unifioijat (MGU:t) näkyviin.
- Ellet kirjoita todistusta binääripuun muotoon, numeroi klausuulit ja ilmoita, mistä klausuuleista mikin klausuuli on johdettu.
- Laske yleisimpien unifioijien kompositio selvittääksesi kyselyssä esiintyville muuttujille arvot.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Esitetään listat vakion $e$ (tyhjä lista) ja kaksipaikkaisen funktiosymbolin $c$ avulla (näin lista $[1,2]$ saa esityksen $c(1,c(2,e))$).

Määritellään seuraavat listoja koskevat predikaatit

- $K(x,y) =$ "listan $x$ alkioina ovat listan $y$ alkiot käänteisessä järjestyksessä":
  1. $K(e,e)$ ja
  2. $\forall x\forall y\forall z\forall v(K(x,y)\wedge L(y,v,z)\rightarrow K(c(v,x),z))$.
- $L(y,v,z) =$ "lista $z$ on lista $y$, jonka perään on liitetty alkio $v$":
  3. $\forall x L(e,x,c(x,e))$ ja
  4. $\forall y\forall v\forall z\forall x(L(y,v,z)\rightarrow L(c(x,y),v,c(x,z)))$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



- Johdetaan lauseille klausuulimuodot:
  (1) $\rightsquigarrow \{\{K(e,e)\}\}$
  (2) $\rightsquigarrow \forall x\forall y\forall z\forall v(\neg K(x,y)\vee\neg L(y,v,z)\vee K(c(v,x),z))$
  $\rightsquigarrow \{\{\neg K(x,y),\neg L(y,v,z),K(c(v,x),z)\}\}$
  (3) $\rightsquigarrow \{\{L(e,x,c(x,e))\}\}$
  (4) $\rightsquigarrow \forall y\forall v\forall z\forall x(\neg L(y,v,z)\vee L(c(x,y),v,c(x,z))$
  $\rightsquigarrow \{\{\neg L(y,v,z),L(c(x,y),v,c(x,z))\}\}$
  (5) Todistettavan lauseen negaatio $\neg\exists x K(c(1,c(2,e)),x)$
  $\rightsquigarrow \forall x\neg K(c(1,c(2,e)),x) \rightsquigarrow \{\{\neg K(c(1,c(2,e)),x)\}\}$
- Haluamme siis selvittää, millainen on lista $[1,2]$ käännettynä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Resoluutiotodistus:**

1. $\{\neg K(c(1,c(2,e)),x_0)\}$ P5
2. $\{\neg K(x_1,y_1),\neg L(y_1,v_1,z_1),K(c(v_1,x_1),z_1)\}$ P2
3. $\{\neg K(c(2,e),y_1),\neg L(y_1,1,z_1)\}$ 1,2,MGU $\{v_1/1,x_1/c(2,e),x_0/z_1\}$
4. $\{\neg K(x_2,y_2),\neg L(y_2,v_2,z_2),K(c(v_2,x_2),z_2)\}$ P2
5. $\{\neg K(e,y_2),\neg L(y_2,2,y_1),\neg L(y_1,1,z_1)\}$ 3,4,MGU $\{v_2/2,x_2/e,z_2/y_1\}$
6. $\{K(e,e)\}$ P1
7. $\{\neg L(e,2,y_1),\neg L(y_1,1,z_1)\}$ 5,6,MGU $\{y_2/e\}$
8. $\{L(e,x_3,c(x_3,e))\}$ P3

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

9. $\{\neg L(c(2,e),1,z_1)\}$ 7,8,MGU $\{x_3/2, y_1/c(2,e)\}$
10. $\{\neg L(y_4,v_4,z_4), L(c(x_4,y_4),v_4,c(x_4,z_4))\}$ P4
11. $\{\neg L(e,1,z_4)\}$ 9,10,MGU $\{x_4/2, y_4/e, v_4/1, z_1/c(2,z_4)\}$
12. $\{L(e,x_5,c(x_5,e)))\}$ P3
13. $\square$ 11,12,MGU $\{x_5/1, z_4/c(1,e)\}$

- Unifioijien kompositio: $\{\, v_1/1,\ x_1/c(2,e),\ x_0/c(2,c(1,e)),$
  $v_2/2,\ x_2/e,\ z_2/c(2,e),$
  $y_2/e,$
  $x_3/2,\ y_1/c(2,e)$
  $x_4/2,\ y_4/e,\ v_4/1,\ z_1/c(2,c(1,e))$
  $x_5/1,\ z_4/c(1,e)\ \}.$
- Rajaus kyselyyn: $\{x_0/c(2,c(1,e))\}$ (ns. *vastaussubstituutio*).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 8.4 Resoluutiostrategioita

### Kerrossaturaatiostrategia

Suoritetaan resoluutioaskeleita kerroksittain seuraavasti:

1. Joukko $S$ muodostaa kerroksen $S_0$.
2. Kerros $S_i$ $(i>0)$ muodostuu klausuuleista $C$, jotka saadaan resoluutiosäännöllä joistain klausuuleista $C_1 \in S_{i-1}$ ja $C_2 \in S_0 \cup \cdots \cup S_{i-1}$.

**Esimerkki.** Olkoon alimpana kerroksena
$S_0 = S = \{\{\neg I(x), E(x)\}, \{I(c)\}, \{K(c)\}, \{\neg E(y), \neg K(y)\}\}$.

Tällöin $S_1 = \{\{E(c)\}, \{\neg E(c)\}, \{\neg I(x), \neg K(x)\}\}$
$S_2 = \{\square, \{\neg I(c)\}, \{\neg K(c)\}\}$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Tukijoukkostrategia

**Määritelmä.** Klausuulijoukon $S$ osajoukko $T$ on *tukijoukko* (engl. *set of support*), jos $S-T$ on toteutuva.

- *Tukijoukkostrategiassa* ei milloinkaan suoriteta resoluutiota joukon $S-T$ klausuuleille keskenään.
- Tutkittaessa loogista seuraavuutta $\Sigma \models \phi$ lausejoukko $\Sigma$ (olettamukset) on tyypillisesti toteutuva. Tällöin voidaan ajatella:
  1. $T$ muodostuu lauseesta $\neg\phi$ saatavien klausuulien joukosta ja
  2. joukkoon $S-T$ kuuluvat lausejoukosta $\Sigma$ saatavat klausuulit.
- Ristiriita aiheutuu siis konkreettisesti tukijoukon $T$ klausuuleista, mikäli $\Sigma \models \phi$ (eli $\Sigma \cup \{\neg\phi\}$ on toteutumaton).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



Klausuulijoukkoa $S$ ja tukijoukkoa $T \subseteq S$ päivitetään seuraavasti:

1. Mikäli $T = \emptyset$ voidaan lopettaa ja todeta $S-T=S$ toteutuvaksi, muutoin valitaan jokin tukijoukon klausuuli $C \in T$.
2. Muodostetaan kaikki klausuulit $C_1, \ldots, C_n$, jotka saadaan resoluutiosäännöllä klausuulista $C$ ja joukon $S$ klausuuleista. Jos $\square \in \{C_1, \ldots, C_n\}$ lopetetaan ja todetaan $S$ toteutumattomaksi.
3. Muutoin päivitetään $S$ ja $T$ joukoiksi $S' = S \cup \{C_1, \ldots, C_n\}$ ja $T' = (T - \{C\}) \cup \{C_1, \ldots, C_n\}$ ja palataan kohtaan 1.

**Huomioita.** Päivityksen yhteydessä $C$ siirtyy tukijoukon ulkopuolelle

Jos klausuulin $C$ avulla ei voida johtaa uusia klausuuleja eli $n=0$ kohdassa 2, tukijoukko pienenee.

**Väite.** Tukijoukkostrategiaan perustuva resoluutio on täydellistä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Tutkitaan seuraako lause $\phi = \exists y(E(y) \wedge K(y))$ loogisesti lausejoukosta $\Sigma = \{\forall x(I(x) \rightarrow E(x)),\ \exists x(I(x) \wedge K(x))\}$.

Lausejoukosta $\Sigma$ saadaan $S - T_0 = \{\{\neg I(x), E(x)\}, \{I(c)\}, \{K(c)\}\}$ ja lauseesta $\neg\phi$ tukijoukko $T_0 = \{\{\neg E(y), \neg K(y)\}\}$.

1. Valitaan klausuuli $C_1 = \{\neg E(y), \neg K(y)\} \in T_0$:
– Klausuulista $\{K(c)\}$ saadaan $\{\neg E(c)\}$ (MGU $\{y/c\}$).
– Klausuulista $\{\neg I(x), E(x)\}$ saadaan $\{\neg I(x), \neg K(x)\}$ (MGU $\{y/x\}$).

2. Valitaan $C_2 = \{\neg E(c)\} \in T_1 = \{\{\neg E(c)\}, \{\neg I(x), \neg K(x)\}\}$:
– Klausuulista $\{\neg I(x), E(x)\}$ saadaan $\{\neg I(c)\}$ (MGU $\{x/c\}$).

3. Valitaan $C_3 = \{\neg I(c)\} \in T_2 = \{\{\neg I(c)\}, \{\neg I(x), \neg K(x)\}\}$:
– Klausuulista $\{I(c)\}$ saadaan $\square$ (MGU $\varepsilon$).

$\Longrightarrow$ $S$ ja $\Sigma \cup \{\neg\phi\}$ ovat toteutumattomia, joten $\Sigma \models \phi$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 8.5 Automaattinen päättely (OTTER)

OTTER on resoluutiosääntöön perustuva automaattinen teoreemantodistin, jota käytämme viimeisessä kotitehtävässä.

Johdantona OTTERin käyttöön käsittelemme seuraavat asiat:

- Kaavojen syntaksi
- Syötetiedoston rakenne
- Joitain yleisiä käyttöohjeita
- Tulosteiden tulkitseminen
- Kotitehtävän palautus
- Saavutuksia matematiikassa

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Kaavojen syntaksi

- OTTER noudattaa kaavojen osalta seuraavaa syntaksia:

| | | |
|---|---|---|
| $\forall x \forall y \forall z \varphi$ | $\rightsquigarrow$ | `all x y z φ` |
| $\exists x \exists y \exists z \varphi$ | $\rightsquigarrow$ | `exists x y z φ` |
| $\varphi \wedge \psi$ | $\rightsquigarrow$ | `φ & ψ` |
| $\varphi \vee \psi$ | $\rightsquigarrow$ | `φ \| ψ` |
| $\varphi \rightarrow \psi$ | $\rightsquigarrow$ | `φ -> ψ` |
| $\varphi \leftrightarrow \psi$ | $\rightsquigarrow$ | `φ <-> ψ` |
| $t_1 = t_2$ | $\rightsquigarrow$ | $t_1$ `=` $t_2$ |
| $\neg\varphi$ | $\rightsquigarrow$ | `-φ` |
| $(\ldots)$ | $\rightsquigarrow$ | `(...)` |

- Syötetiedostossa lauseiden loppuun tulee kirjoittaa piste "." !

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.**

| | | |
|---|---|---|
| $\forall x \forall y \exists z P(x,y,z)$ | $\rightsquigarrow$ | `all x y (exists z P(x,y,z)).` |
| $\forall x(P(x) \wedge (Q(x) \vee R(x)))$ | $\rightsquigarrow$ | `all x (P(x) & (Q(x) \| R(x))).` |

- Eri symbolit erotetaan toisistaan sijainnin perusteella. OTTER sallii samannimiset vakio-, muuttuja-, funktio- ja predikaattisymbolit!
- OTTER erottelee muuttujat vakioista kvanttorien perusteella (näin mahdolliset "vapaat" muuttujaesiintymät tulkitaan vakioiksi).
- OTTER muuntaa lauseet automaattisesti klausuuleiksi, joita voi käyttää myös tiettyjen syntaksivirheiden tunnistamiseen.

**Esimerkki.** Lausekkeessa `(all x y(P(x,y)))` esiintyvä `y` on sekä predikaatti- että vakiosymboli ja `P` funktiosymboli, mutta `y` tulkitaan muuttujaksi ja `P` predikaatiksi lausekkeessa `(all x y (P(x,y)))` !

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## Syötetiedoston rakenne

- Yksinkertainen esimerkki syötetiedosta on annettu alla.
- `set(auto)`-asetuksella OTTER valitsee itse käytettävät päättelysäännöt (mm. resoluutiosääntö muunnelmineen).

```
set(auto).
formula_list(usable).

% tähän tulee sitten tarvittava lausejoukko

% ja kyselyn negaatio (vain yksi kysely kerrallaan)

end_of_list.
```

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Palautetaan mieliin aikaisempi tartuntavaaraesimerkki.

```
set(auto).
formula_list(usable).
% Section A: database
tapaa(a,b). tapaa(a,c). sairastaa(c).

% Section B: definitions
all x y (tapaa(x,y) -> tapaa(y,x)).
all x y (tapaa(x,y) & sairastaa(y) -> tartuntavaarassa(x)).
all x y (tapaa(x,y) & tartuntavaarassa(y) -> tartuntavaarassa(x)).

% Section C: negation of the query
-(tartuntavaarassa(a) & tartuntavaarassa(b)).

end_of_list.
```

- OTTER pystyy osoittamaan lausejoukon helposti ristiriitaiseksi.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## Joitain yleisiä käyttöohjeita

- OTTER otettaan käyttöön Atk-keskuksen unix-koneissa komennolla

  `use otter`

- Manuaaliin löytyy linkki kotitehtäväpalvelimesta sivulta

  `http://logic.tcs.hut.fi/~ltp/`

- OTTERin voi käynnistää esim. seuraavilla tavoilla:

  ```
  otter < file.in
  otter < file.in > file.out
  otter < file.in | less
  ```

- Hakemistosta `/p/edu/tik-79.144` löytyy myös dokumentaatiota ja esimerkkejä (pääasiassa matematiikasta).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## Tulosteiden tulkitseminen

OTTERin tulostiedostosta löytyvät mm.

- Lauseille automaattisesti haetut klausuulimuodot.
- Klausuulien luokittelu (tukijoukon identifiointi).
- Annetuista klausuuleista johdetut uudet klausuulit, kun OTTER yritää johtaa tyhjän klausuulin.
- Mahdollisesti löydetty todistus, johon eristetään tyhjän klausuulin johtamisessa tarvittavat klausuulit.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Tutustutaan tarkemmin OTTERin tulosteisiin tartuntavaaraesimerkissä.

1. Klausuulimuodosta tulee seuraava:

```
list(usable).
0 [] tapaa(a,b).
0 [] tapaa(a,c).
0 [] sairastaa(c).
0 [] -tapaa(x,y)|tapaa(y,x).
0 [] -tapaa(x,y)| -sairastaa(y)|tartuntavaarassa(x).
0 [] -tapaa(x,y)| -tartuntavaarassa(y)|tartuntavaarassa(x).
0 [] -tartuntavaarassa(a)| -tartuntavaarassa(b).
end_of_list.
```

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



2. set(auto)-asetuksen myötä OTTER päättää itse tukijoukkoon (engl. set of support) tulevat klausuulit:

```
------------> process usable:
** KEPT (pick-wt=6): 1 [] -tapaa(x,y)|tapaa(y,x).
** KEPT (pick-wt=7): 2 [] -tapaa(x,y)| -sairastaa(y)|
                         tartuntavaarassa(x).
** KEPT (pick-wt=7): 3 [] -tapaa(x,y)| -tartuntavaarassa(y)|
                         tartuntavaarassa(x).
** KEPT (pick-wt=4): 4 [] -tartuntavaarassa(a)|
                         -tartuntavaarassa(b).
------------> process sos:
** KEPT (pick-wt=3): 5 [] tapaa(a,b).
** KEPT (pick-wt=3): 6 [] tapaa(a,c).
** KEPT (pick-wt=2): 7 [] sairastaa(c).
```

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



3. Tämän jälkeen yritetään johtaa tyhjä klausuuli, mikä onnistuukin:

```
=========== start of search ===========

given clause #1: (wt=3) 5 [] tapaa(a,b).
given clause #2: (wt=2) 7 [] sairastaa(c).
given clause #3: (wt=3) 6 [] tapaa(a,c).
given clause #4: (wt=2) 9 [hyper,6,2,7] tartuntavaarassa(a).
given clause #5: (wt=3) 8 [hyper,5,1] tapaa(b,a).
given clause #6: (wt=3) 10 [hyper,6,1] tapaa(c,a).
given clause #7: (wt=2) 11 [hyper,8,3,9] tartuntavaarassa(b).

-----> EMPTY CLAUSE at   0.00 sec ----> 13 [hyper,11,4,9] $F.
```

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



4. Tämän jälkeen OTTER eristää johdon tyhjälle klausuulille:

```
Length of proof is 3.  Level of proof is 2.
---------------- PROOF ----------------
1 [] -tapaa(x,y)|tapaa(y,x).
2 [] -tapaa(x,y)| -sairastaa(y)|tartuntavaarassa(x).
3 [] -tapaa(x,y)| -tartuntavaarassa(y)|tartuntavaarassa(x).
4 [] -tartuntavaarassa(a)| -tartuntavaarassa(b).
5 [] tapaa(a,b).
6 [] tapaa(a,c).
7 [] sairastaa(c).
8 [hyper,5,1] tapaa(b,a).
9 [hyper,6,2,7] tartuntavaarassa(a).
11 [hyper,8,3,9] tartuntavaarassa(b).
13 [hyper,11,4,9] $F.
------------ end of proof -------------
```

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Huomioita.**

- Mikäli OTTER ei löydä todistusta, OTTER voi pysähtyä ilmoittaen "Search stopped because sos empty." tai jäädä ikuiseen silmukkaan.
- Muuttujasidontojen eristämistä varten OTTERissa voi määritellä `$ans`-alkuisia predikatteja, joiden argumenteiksi kirjataan mielenkiinnon kohteena olevat muuttujat.

**Esimerkki.** Haetaan jokin tartuntavaarassa oleva henkilö kyselyllä `(exists x (tartuntavaarassa(x) & $ans(x)))`:

```
2 [] -tapaa(x,y)| -sairastaa(y)|tartuntavaarassa(x).
4 [] -tartuntavaarassa(x)| -$ans(x).
6 [] tapaa(a,c).
7 [] sairastaa(c).
9 [hyper,6,2,7] tartuntavaarassa(a).
10 [binary,9.1,4.1] -$ans(a).
```

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Kotitehtävän palautus**

- Ratkaisuiksi laaditaan 3 syötetiedostoa OTTERille.
- Kohtaan 3a palautetaan syötetiedosto, joka sisältää pelkät predikaattien määritelmät (ei tietokantaa eikä kyselyä).
- Kohtiin 3b ja 3c palautetaan syötetiedosto, joka sisältää em. määritelmien lisäksi myös tietokannan ja tähän liittyvän kyselyn.
- Kohdan 3a ratkaisuksi palautettua syötetiedostoa käytetään
  1. kohdassa 3a määritelmien konsistenssin tarkastamiseen,
  2. kohdassa 3d vertailuun mallimääritelmien kanssa ja
  3. kohdassa 3e testikyselyn evaluointiin jonkin tietokannan suhteen.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Saavutuksia matematiikassa**

- OTTERia on käytetty mm. matematiikkaan liittyvien avointen ongelmien ratkomiseen.
- Ensimmäisenä merkittävänä ongelmana pystyttiin osoittamaan kahden Boolen algebroille esitetyn aksiomatisoinnin ekvivalenssi (Huntingtonin ja Robbinsin määritelmät).
- Tuloksen yksityiskohtia on selvitetty tarkemmin sivulla

  `http://www-unix.mcs.anl.gov/~mccune/papers/robbins/`
- Muita OTTERilla (ja OTTERiin liittyvillä ohjelmilla) osoitettuja tuloksia on raportoitu sivulla

  `http://www-unix.mcs.anl.gov/AR/`

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



# 9 Ohjelmien oikeellisuustarkastelut

- Tarkasteltava ohjelmointikieli
- Ehtolausekkeiden ekvivalenssi
- Ohjelmien esi- ja jälkiehdot
- Toistolausekkeiden invariantit
- Täydellinen oikeellisuus

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

### Motivaatio

Miksi tietokoneohjelmille tulisi kirjoittaa formaaleja spesifikaatioita?

- Spesifikaatiota laadittaessa joudutaan suunnittelemaan ennalta varsin tarkkaan mitä ohjelmiston on tarkoitus tehdä.
- Järjestelmän toteutus voidaan *verifioida* eli todeta määrittelynsä mukaiseksi vasta, kun spesifikaatio on tehty.
- Formaalissa spesifioinnissa etuna on määritelmien yksikäsitteisyys.
- *Turvallisuuskriittiset järjestelmät* (esim. lentokoneen ohjausjärjestelmä) vaativat perinpohjaista määrittelyä ja verifiointia.
- Hyvin määritellyn ohjelman uudelleenkäyttö on helpompaa.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 9.1 Tarkasteltava ohjelmointikieli

Tarkastellaan seuraavaa kokonaislukujen käsittelyyn riittävää osajoukkoa tyypillisistä lausekielistä kuten Pascal, C, C++ ja Java.

**Määritelmä.** *Kokonaislukulausekkeet* $E$ määritellään seuraavasti.

1. Mikä tahansa kokonaisluku $\ldots$, `-1`, `0`, `1`, $\ldots$ on kokonaislukulauseke.
2. Kokonaislukumuuttujat `x`, `y`, $\ldots$ ovat kokonaislukulausekkeita.
3. Jos $E_1$ ja $E_2$ ovat kokonaislukulausekkeita, niin myös
   summa $(E_1 + E_2)$, erotus $(E_1 - E_2)$ ja tulo $(E_1 * E_2)$
   ovat kokonaislukulausekkeita.
4. Muita kokonaislukulausekkeita ei ole.

**Esimerkki.** Merkkijono `((x-y)*x)` on kokonaislukulauseke.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Määritelmä.** *Boolen lausekkeet* $B$ määritellään seuraavasti.

1. Boolen vakiot `false` ja `true` ovat Boolen lausekkeita.
2. Jos $E_1$ ja $E_2$ ovat kokonaislukulausekkeita, niin $E_1 > E_2$ on Boolen lauseke.
3. Jos $B_1$ ja $B_2$ ovat Boolen lausekkeita, niin
   negaatio $!B_1$, konjunktio $B_1$ `&&` $B_2$ ja disjunktio $B_1$ `||` $B_2$
   ovat myös Boolen lausekkeita.
4. Muita Boolen lausekkeita ei ole.

**Huomioita.** Lausekkeet $E_1 == E_2$ (yhtäsuuruus), $E_1 != E_2$, $E_1 <= E_2$, $E_1 < E_2$ ja $E_1 >= E_2$ ovat lyhennysmerkintöjä lausekkeille $!(E_1 > E_2)$ `&&` $!(E_2 > E_1)$, $!(E_1 == E_2)$, $!(E_1 > E_2)$, $E_2 > E_1$ ja $!(E_1 < E_2)$.

Implikaatio $B_1 \texttt{->} B_2$ on lyhennysmerkintä lausekkeelle $!B_1$ `||` $B_2$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Määritelmä.** Ko. ohjelmointikielen *komennot* $C$ määritellään seuraavasti.

1. Jos $x$ on kokonaislukumuuttuja ja $E$ on kokonaislukulauseke, niin *sijoituslauseke* `x=`$E$ on komento.
2. Jos $B$ on Boolen lauseke sekä $C_1$ ja $C_2$ ovat komentoja, niin myös

   | | |
   |---|---|
   | $C_1$ `;` $C_2$ | (peräkkäinen suorittaminen) |
   | `if(`$B$`) then {`$C_1$`} else {`$C_2$`}` | (ehdollinen suorittaminen) |
   | `while(`$B$`) {`$C_1$`}` | (toistolauseke) |

   ovat komentoja.
3. Muita komentoja ei ole.

$\Longrightarrow$ *Ohjelmat* ovat määritelmän mukaisia (rakenteisia) komentoja.

**Esimerkki.** Ohjelma `y=1; z=1; while(z!=x) {z=z+1; y=y*z}` laskee muuttujan `y` arvoksi muuttujan arvon `x` kertoman (kun `x>0`).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Määritelmä.** Struktuuri $\mathcal{S}$ on $\mathbb{Z}$-struktuuri $\iff$ (i) struktuurin $\mathcal{S}$ universumina $U$ on kokonaislukujen joukko $\mathbb{Z}$, (ii) symboleilla `+`, `-`, `*` ja `>` on *standarditulkinnat*: yhteen-, vähennys- ja kertolaskufunktiot sekä suurempi kuin -relaatio kokonaislukujen joukossa.

**Huomioita.**

- *Ohjelman tila* voidaan rinnastaa $\mathbb{Z}$-struktuuriin $\mathcal{S}$.
- Kokonaislukulausekkeen $E$ arvo tilassa $\mathcal{S}$ on kokonaisluku $E^{\mathcal{S}}$.
- Boolen lauseke $B$ on tosi tilassa $\mathcal{S} \iff \mathcal{S} \models B$.

**Esimerkki.** Tarkastellaan tilaa $\mathcal{S}$, missä $\mathtt{x}^{\mathcal{S}} = 2$ ja $\mathtt{y}^{\mathcal{S}} = 6$ ja $\mathtt{z}^{\mathcal{S}} = 3$.

Lausekkeiden `(x*x)` ja `(z*z)` arvot ovat $(\mathtt{x*x})^{\mathcal{S}} = 4$ ja $(\mathtt{z*z})^{\mathcal{S}} = 9$.

Niinpä $\mathcal{S} \models (\mathtt{x*x} < \mathtt{y})\ \mathtt{\&\&}\ (\mathtt{y} < \mathtt{z*z})$, mutta $\mathcal{S} \not\models (\mathtt{x*y} < \mathtt{z})$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Komentojen suorittamisen vaikutus tilaan

**Määritelmä.** Määritellään tilansiirtorelaatio $\mathcal{S} \xrightarrow{C} \mathcal{S}'$ eli tila $\mathcal{S}'$, johon päädytään tilasta $\mathcal{S}$, jos ja kun komennon $C$ suoritus päättyy.

1. Jos $C$ on sijoituslauseke `x=`$E$, niin tila $\mathcal{S}'$ on $\mathcal{S}[\mathtt{x} \mapsto E^{\mathcal{S}}]$.
2. Jos $C$ on muodoltaan $C_1$ `;` $C_2$, niin $\mathcal{S}'$ on tila, joka saavutetaan suorittamalla ensin $C_1$ ja sitten $C_2$.
3. Jos $C$ on ehtolauseke `if(`$B$`) then {`$C_1$`} else {`$C_2$`}`, niin $\mathcal{S}'$ on tila, joka saavutetaan suorittamalla $C_1$, jos $\mathcal{S} \models B$, ja $C_2$, jos $\mathcal{S} \not\models B$.
4. Jos $C$ on toistolauseke `while(`$B$`) {`$C_1$`}` ja $\mathcal{S} \not\models B$, niin $\mathcal{S}' = \mathcal{S}$.
5. Jos $C$ on toistolauseke `while(`$B$`) {`$C_1$`}` ja $\mathcal{S} \models B$, niin $\mathcal{S}'$ on tila, joka saavutetaan suorittamalla $C_1$ `;` `while(`$B$`) {`$C_1$`}`.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Huomioita.**

- Ainoastaan sijoituslausekkeiden suorittaminen voi muuttaa tilaa.
- `while`-rakenteen suorittamisen päättyminen ei ole taattua.

**Esimerkki.** Tarkastellaan ohjelman

```
y=1 ; z=1 ; while(z!=x) {z=z+1 ; y=y*z}
```

suoritusta tilasta $\mathcal{S}$, missä $\mathtt{x}^{\mathcal{S}} = 3$. Ohjelman suorituksen aikana alkutilaa päivitetään seuraavasti:

$$\mathtt{y} \mapsto 1,\ \mathtt{z} \mapsto 1,\ \mathtt{z} \mapsto 2,\ \mathtt{y} \mapsto 2,\ \mathtt{z} \mapsto 3 \text{ ja } \mathtt{y} \mapsto 6.$$

$\implies$ muuttujan `y` arvona on muuttujan `x` arvon kertoma.

**Esimerkki.** Ohjelman `x=1 ; while(x>0) {y=y+1}` suoritus ei pääty, koska komennon `y=y+1` toistaminen ei vaikuta ehdon toteutumiseen.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 9.2 Ehtolausekkeiden ekvivalenssi

- Ohjelmointikielissä käytetään paljon ehtolausekkeita kontrolloimaan, millä ehdoilla ja mitä toimintoja suoritetaan.
- Jos ehtolausekkeita muutetaan esim. optimointitarkoituksessa, halutaan varmistua että toiminnot suoritetaan samoilla ehdoilla.
- Ehtolausekkeiden ekvivalenssin osoittamiseen voidaan käyttää sekä lauselogiikan että predikaattilogiikan menetelmiä.
- Jos ehtolausekkeiden evaluoinnilla on *sivuvaikutuksena* muutoksia ohjelman tilaan, pelkkä loogisen ekvivalenssin tarkastaminen ei välttämättä riitä.

**Esimerkki.** Tällainen sivuvaikutus voi olla esim. virhetilanne, joka on aiheutunut ehtojen evaluoinnista väärässä järjestyksessä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Vertaillaan kahta eri ohjelmaa:

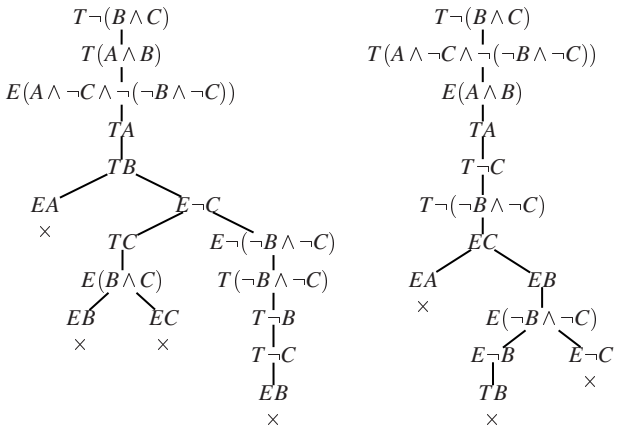
```
if((x>0) && !(y>x)) then {
    if(x!=y) then {z=x} else {z=y}
} else {z=0}

if(x>0) then {
    if(x>y) then {z=x} else {z=y}
} else {z=0}
```

- Valitaan atomiset lauseet $A =$ "x>0", $B =$ "x>y" ja $C =$ "y>x".
- Nyt esim. sijoituslauseke z=x suoritetaan näissä ohjelmissa seuraavilla ehdoilla: $A \wedge \neg C \wedge \neg(\neg B \wedge \neg C)$ ja $A \wedge B$.
- Lauselogiikan nojalla näiden välinen ekvivalenssi on looginen seuraus lauseesta $\neg(B \wedge C)$, joka on aina voimassa $B$:n ja $C$:n välillä.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** (Jatkoa) ekvivalenssi voidaan todeta semanttisella taululla:




Näiden perusteella $\{\neg(B \wedge C)\} \models (A \wedge B) \leftrightarrow (A \wedge \neg C \wedge \neg(\neg B \wedge \neg C))$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Kytkennät predikaattilogiikkaan

- Boolen lauseke $B$ on *$\mathbb{Z}$-pätevä* (merk. $\models_{\mathbb{Z}} B$) $\iff$ $\mathcal{S} \models B$ kaikissa $\mathbb{Z}$-struktuureissa $\mathcal{S}$.
- Näin lausekkeiden muuttujat saavat universaalin tulkinnan.
- Boolen lausekkeet $B_1$ ja $B_2$ ovat *$\mathbb{Z}$-ekvivalentit* (merk. $B_1 \equiv_{\mathbb{Z}} B_2$) $\iff$ lausekkeilla on sama totuusarvo kaikissa $\mathbb{Z}$-struktuureissa.
- Huomaa, että $\models B \implies \models_{\mathbb{Z}} B$ ja $B_1 \equiv B_2 \implies B_1 \equiv_{\mathbb{Z}} B_2$, mutta käänteiset implikaatiot eivät välttämättä ole voimassa.

**Esimerkki.** $\models_{\mathbb{Z}} !((x>y)\,\&\&\,(y>x))$, mutta $\not\models !((x>y)\,\&\&\,(y>x))$, koska löytyy vastamalli $\mathcal{S}$, jolle $U = \{0\}$, $x^{\mathcal{S}} = y^{\mathcal{S}} = 0$ ja $>^{\mathcal{S}} = \{\langle 0,0\rangle\}$.

$\implies$ relaation > suhde funktioihin +, - ja * joudutaan kuvaamaan erikseen (vrt. $\neg(B \wedge C)$ edellä), jos käytetään predikaattilogiikkaa.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 9.3 Ohjelmien esi- ja jälkiehdot

- Tarkasteltavan ohjelmointikielen ohjelmilla on ääretön tila-avaruus, jonka läpikäyminen on käytännössä mahdotonta.
- Yksi mahdollisuus on tarkastella Boolen lausekkeiden $B$ määrittelemiä tilajoukkoja $\{\mathcal{S} \mid \mathcal{S} \models B\}$ ja analysoida, millaisia muutoksia annettu ohjelma näihin aiheuttaa.
- Mille tahansa ohjelmalle P voidaan asettaa *esi- ja jälkiehdot* $B_1$ ja $B_2$ kirjoittamalla ns. Hoaren kolmikko $[B_1]$ P $[B_2]$.
- Karkeasti ottaen ajatuksena on, että esiehdon $B_1$ on tarkoitus taata jälkiehdon $B_2$ voimaantulo ohjelman P suorituksen päättyessä.

**Esimerkki.** Olkoon Succ ohjelma `if(x==0) then {y=1} else {y=x+1}`, jolle voidaan antaa spesifikaatio [true] Succ [y==x+1].

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

## Osittainen ja täydellinen oikeellisuus

Olkoon P ohjelma sekä $B_1$ ja $B_2$ kaksi Boolen lauseketta.

**Määritelmä.** Ohjelma P on *osittain oikeellinen* annettujen esi- ja jälkiehtojen $B_1$ ja $B_2$ suhteen (merk. $\models_p [B_1]\ \texttt{P}\ [B_2]$) $\iff$ $\mathcal{S}' \models B_2$ pätee saavutettavalle tilalle $\mathcal{S}'$ aina kun ohjelman P suoritus aloitetaan tilasta $\mathcal{S}$, missä $\mathcal{S} \models B_1$, ja ohjelman P *suoritus päättyy* tilaan $\mathcal{S}'$.

**Esimerkki.** Osittainen oikeellisuus ei edellytä suorituksen päättymistä: $\models_p [\texttt{true}]\ \texttt{while(x!=y) \{z=x; x=y; y=z\}}\ [\texttt{x==y}]$.

**Määritelmä.** Ohjelma P on *täysin oikeellinen* annettujen esi- ja jälkiehtojen $B_1$ ja $B_2$ suhteen (merk. $\models_t [B_1]\ \texttt{P}\ [B_2]$) $\iff$ $\models_p [B_1]\ \texttt{P}\ [B_2]$ ja ohjelman P suoritus päättyy aina kun $\mathcal{S} \models B_1$ alkutilalle $\mathcal{S}$.

**Huomio.** Vastaavat englannin kieliset termit ovat *partial correctness* ($\models_p$) ja *total correctness* ($\models_t$).

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## Päättelysääntöjä osittaiselle oikeellisuudelle

Alla $B$, $B_0$, $B_1$ ja $B_2$ ovat Boolen lausekkeita, ja $C$, $C_1$ ja $C_2$ komentoja.

Sijoituslauseke: $\dfrac{}{[B\{\texttt{x}/E\}]\ \texttt{x=}E\ [B]}$

Kompositio: $\dfrac{[B_0]\ C_1\ [B_1] \qquad [B_1]\ C_2\ [B_2]}{[B_0]\ C_1\ \texttt{;}\ C_2\ [B_2]}$

Ehtolauseke: $\dfrac{[B_1 \texttt{ \&\& } B]\ C_1\ [B_2] \qquad [B_1 \texttt{ \&\& !}B]\ C_2\ [B_2]}{[B_1]\ \texttt{if(}B\texttt{) then \{}C_1\texttt{\} else \{}C_2\texttt{\}}\ [B_2]}$

Toistolauseke: $\dfrac{[B_1 \texttt{ \&\& } B_2]\ C\ [B_1]}{[B_1]\ \texttt{while(}B_2\texttt{) \{}C\texttt{\}}\ [B_1 \texttt{ \&\& !}B_2]}$

Implikaatio: $\dfrac{\models_{\mathbb{Z}} B_1 \texttt{->} B_2 \qquad [B_2]\ C\ [B_3] \qquad \models_{\mathbb{Z}} B_3 \texttt{->} B_4}{[B_1]\ C\ [B_4]}$

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Olkoon $n$ ja $m$ mitä tahansa kokonaislukuja. Osoitetaan $\models_p [\texttt{(x==}n\texttt{) \&\& (y==}m\texttt{)}]\ \texttt{z=x; x=y; y=z}\ [\texttt{(x==}m\texttt{) \&\& (y==}n\texttt{)}]$.

*Todistus.* Käytetään edellä esiteltyjä päättelysääntöjä:

| | | |
|---|---|---|
| 1. | $[\texttt{(x==}m\texttt{) \&\& (z==}n\texttt{)}]\ \texttt{y=z}\ [\texttt{(x==}m\texttt{) \&\& (y==}n\texttt{)}]$ | Sij. |
| 2. | $[\texttt{(y==}m\texttt{) \&\& (z==}n\texttt{)}]\ \texttt{x=y}\ [\texttt{(x==}m\texttt{) \&\& (z==}n\texttt{)}]$ | Sij. |
| 3. | $[\texttt{(y==}m\texttt{) \&\& (z==}n\texttt{)}]\ \texttt{x=y; y=z}\ [\texttt{(x==}m\texttt{) \&\& (y==}n\texttt{)}]$ | 1,2,Komp. |
| 4. | $[\texttt{(y==}m\texttt{) \&\& (x==}n\texttt{)}]\ \texttt{z=x}\ [\texttt{(y==}m\texttt{) \&\& (z==}n\texttt{)}]$ | Sij. |
| 5. | $[\texttt{(y==}m\texttt{) \&\& (x==}n\texttt{)}]\ \texttt{z=x; x=y; y=z}\ [\texttt{(x==}m\texttt{) \&\& (y==}n\texttt{)}]$ | 3,4,Komp. |
| 6. | $[\texttt{(x==}n\texttt{) \&\& (y==}m\texttt{)}]\ \texttt{z=x; x=y; y=z}\ [\texttt{(x==}m\texttt{) \&\& (y==}n\texttt{)}]$ | Impl. |

Viimeisessä askelessa hyödynnetään Boolen lausekkeiden $\texttt{(x==}n\texttt{) \&\& (y==}m\texttt{)}$ ja $\texttt{(y==}m\texttt{) \&\& (x==}n\texttt{)}$ välistä ekvivalenssia.

**Huomioita.** Todistus voitaisiin kirjoittaa myös puun muotoon. Lisäksi luvut $n$ ja $m$ voitaisiin korvata kokonaislukumuuttujilla n ja m.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## Heikoimmat esiehdot

- Olkoon P ohjelma $C_1\ \texttt{;}\ \ldots\ \texttt{;}\ C_n$, missä $C_1, \ldots, C_n$ ovat järjestyksessä peräkkäin suoritettavat komennot.
- Ominaisuuden $\models_p [B_0]\ \texttt{P}\ [B_n]$ osoittaminen voidaan pilkkoa osaongelmiin: tulisi löytää sopivat ehdot $B_1, \ldots, B_{n-1}$ siten, että $\models_p [B_{i-1}]\ C_i\ [B_i]$ on osoitettavissa kaikille $i \in \{1, \ldots, n\}$.
- Usein tällaiset ehdot voidaan löytää analysoimalla komentosekvenssiä takaperin: haetaan komennolle $C_i$ (missä $i$ saa arvot $n, n-1, \ldots, 1$) *heikoin esiehto* $B_{i-1}$ siten, että $\models_p [B_{i-1}]\ C_i\ [B_i]$.
- Jatkossa tällaisia todistuksia kirjoitetaan sekvensseiksi
$$[B_0]\ C_1\ [B_1]\ C_2\ [B_2] \ldots\ C_n\ [B_n],$$
vaikka käytännössä sekvenssi muodostetaankin usein takaperin.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

Tarkastellaan seuraavaksi, millaisista todistusaskelista tällainen sekvenssi voidaan muodostaa edellä esiteltyjen päättelysääntöjen nojalla.

1. Jos $B$ on jälkiehto sijoituslausekkeelle `x=`$E$, heikoimmaksi esiehtoksi voidaan kirjata $B\{\texttt{x}/E\}$ eli $\models_p [B\{\texttt{x}/E\}]$ `x=`$E$ $[B]$.
   **Esimerkki.** `[x-1>0] y=x-1 [y>0]`
2. Jos $\models_p [B_1]\ C\ [B_2]$ on jo osoitettu ja $B_0$ on esiehdon $B_1$ vahvennus ($\models_{\mathbb{Z}} B_0 \texttt{->} B_1$), niin kirjataan $[B_0]\ [B_1]\ C\ [B_2]$, koska $\models_p [B_0]\ C\ [B_2]$.
   **Esimerkki.** `[x>1] [x-1>0] y=x-1 [y>0]`
3. Jos $\models_p [B_1]\ C_1\ [B_3]$ ja $\models_p [B_2]\ C_2\ [B_3]$ ovat jo (rekursiivisesti) osoitetut jälkiehdolle $B_3$, niin lausekkeen `if(`$B$`) then {`$C_1$`} else {`$C_2$`}` heikoimmaksi esiehdoksi kirjataan $(B \texttt{\&\&} B_1) \texttt{||} (\texttt{!}B \texttt{\&\&} B_2)$.
   **Esimerkki.** `[!(x>y)] [((x>y) && (x==y)) || (!(x>y) && (y==y))] if(x>y) then {x=x} else {x=y} [x==y]`

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Tarkastellaan ohjelman `Succ` muunnelmaa.

```
[true]
[(x==0) || !(x==0)]
[(((x+1)-1==0) && (x==0)) || (!((x+1)-1==0) && (x+1==x+1))]
z=x+1
[((z-1==0) && (x==0)) || (!(z-1==0) && (z==x+1))]
if(z-1==0) then {
    [x==0] [1==x+1]  y=1 [y==x+1]
} else {
    [z==x+1]  y=z [y==x+1]
}
[y==x+1]
```

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 9.4 Toistolausekkeiden invariantit

- Ohjelmointikielten keskeisiä primitiivejä ovat toistolausekkeet, joiden avulla komentoja voidaan toistaa haluttu määrä.

  `z=0;v=0;while(!(z==x)) {z=z+1;v=v+y}`

- Ongelma: kuinka voitaisiin osoittaa toistorakenteita sisältävien algoritmien toimivuus kaikissa tilanteissa?
- Toistorakenteelle halutaan tyypillisesti todistaa *invariantti* eli ominaisuus, joka säilyy voimassa toistorakenteen suorituksen ajan.

**Määritelmä.** Toistolausekkeen `while(`$B$`) {`$C$`}` invariantti $I$ on mikä tahansa Boolen lauseke siten, että $\models_p [B \texttt{\&\&} I]\ C\ [I]$.

**Huomio.** Invariantti $I$ ei välttämättä ole jatkuvasti tosi komennon $C$ suorituksen aikana, mutta ehdottomasti $C$:n suorituksen jälkeen.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



### Sopivan invariantin hakeminen

- Osittaisen oikeellisuuden $\models_p [B_1]$ `while(`$B$`) {`$C$`}` $[B_2]$ todistaminen voi perustua sopivan invariantin $I$ käyttöön:
  1. $\models_{\mathbb{Z}} B_1 \texttt{->} I$,
  2. $\models_{\mathbb{Z}} (I \texttt{\&\&} \texttt{!}B) \texttt{->} B_2$, ja
  3. $\models_p [I]$ `while(`$B$`) {`$C$`}` $[I \texttt{\&\&} \texttt{!}B]$.
- Kuten aiemminkin, oikeellisuustodistusta voi hakea takaperin:

  A1. Valitaan $I$ siten, että $\models_{\mathbb{Z}} (I \texttt{\&\&} \texttt{!}B) \texttt{->} B_2$.

  A2. Haetaan heikoin esiehto $I'$ siten, että $\models_p [I']\ C\ [I]$.

  A3. Osoitetaan $\models_{\mathbb{Z}} I \texttt{\&\&} B \texttt{->} I'$, minkä nojalla $\models_p [I \texttt{\&\&} B]\ C\ [I]$ ja edelleen $\models_p [I]$ `while(`$B$`) {`$C$`}` $[I \texttt{\&\&} \texttt{!}B]$.

  A4. Osoitetaan $\models_{\mathbb{Z}} B_1 \texttt{->} I$.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio

**Esimerkki.** Osoitetaan edellä annetun kertolaskuohjelman `Multi` osittainen oikeellisuus eli $\models_p$ `[true] Multi [v==x*y]`.

`[true] [0==0*y] z=0 [0==z*y] v=0 [v==z*y]` (A4)

```
while(!(x==z)) {
    [(v==z*y) && !(x==z)]   (A3)
    [v+y==(z+1)*y] z=z+1 [v+y==z*y] v=v+y [v==z*y]   (A2)
}
```

`[(v==z*y) && (x==z)] [v==x*y]` (A1)

**Huomioita.** Todistuksessa käytetty invariantti on `v==z*y`.

Edellä esitetyt neljä todistusaskelta (A1)...(A4) on merkitty ylös.

Ohjelman suoritus päättyy, jos ja vain jos `!(x<0)` on tosi.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



## 9.5 Täydellinen oikeellisuus

- Tieto osittaisesta oikeellisuudesta ($\models_p [B_1]\ C\ [B_2]$) on hyödyllinen ainoastaan, mikäli komennon $C$ suoritus todella päättyy.
- Täydellisen oikeellisuuden ($\models_t$) osoittamiseksi joudutaan todistamaan erikseen, että komennossa $C$ esiintyvien toistolausekeiden suoritus päättyy lopulta.
- Tätä varten tarvitaan vahvennettua päättelysääntöä

$$\frac{[B_1 \texttt{ \&\& } B_2 \texttt{ \&\& } (E\texttt{==n})]\ C\ [B_1 \texttt{ \&\& } (E\texttt{<n})]}{[B_1]\ \texttt{while}(B_2)\ \{C\}\ [B_1 \texttt{ \&\& !} B_2]},$$

  missä $E$ on sopiva kokonaislukulauseke, `n` on (uusi) kokonaislukumuuttuja ja $B_1$ on vahvennettu invariantti $B$`&& (0<=`$E$`)`.
- Näin lausekkeen $E$ arvo pienenee jatkuvasti toistettaessa $C$:tä.
  $\Longrightarrow$ Suoritus päättyy vääjäämättä, koska `0<=`$E$ säilyy voimassa.

© 2003 TKK / Tietojenkäsittelyteorian laboratorio



**Esimerkki.** Osoitetaan $\models_t$ `[0<=x] Multi [v==x*y]` seuraavasti:

```
[0<=x] [(0==0*y) && (0<=x-0)] z=0 [(0==z*y) && (0<=x-z)]
v=0 [(v==z*y) && (0<=x-z)]
while(!(x==z)) {
    [(v==z*y) && (0<=x-z) && (x-z==n) && !(x==z)]
    [(v+y==(z+1)*y) && (0<=x-(z+1)) && (x-(z+1)<n)]
    z=z+1 [(v+y==z*y) && (0<=x-z) && (x-z<n)]
    v=v+y [(v==z*y) && (0<=x-z) && (x-z<n)]
}
[(v==z*y) && (0<=x-z) && (x==z)] [v==x*y]
```

**Huomioita.** Lauseke, jonka arvo vähenee aidosti, on $x - z$.

Esiehto `0<=x` on yllä välttämätön, koska $\not\models_t$ `[true] Multi [v==x*y]`!

© 2003 TKK / Tietojenkäsittelyteorian laboratorio