Slide 1

Sääntöpohjainen päättely II

- Horn-klausuulit tarjoavat loogisen pohjan sääntöpohjaiselle päättelylle.
- Typpillisesti tarvitaan kuitenkin Horn-klausuuleja ilmaisuvoimaisempi kieli, jossa negaatio/komplementti:

  logiikkaohjelmointi, deduktiiviset tietokannat, asiantuntijajärjestelmät
- Rekursiivisten sääntöjen ja negaation yhdistäminen ei onnistu suoraviivaisesti.

  ☞ Stabiilien mallien semantiikka
- Toteutustekniikat edenneet merkittävästi viime vuosina.

Slide 2

Negaation lisääminen Horn-klausuuleihin

- Relaation komplementti tarvitaan useimmissa sovellutuksissa.

  $a(X) \leftarrow mba(X) \wedge \text{not } b(X)$
- Ad hoc -ratkaisut johtavat ongelmiin:
  semantiikka riippuu toteutusyksityiskohdista
  (sääntöjen talletusjärjestyksestä, tulkin toteutuksesta yms).

  Esimerkki.
  $b(X) \leftarrow eng(X) \wedge \text{not } a(X)$
  $a(X) \leftarrow mba(X) \wedge \text{not } b(X)$
  $eng(l)$
  $mba(l)$

  ?- $b(l)$

Slide 3

Stabiilien mallien semantiikka

(Stable model semantics [Gelfond and Lifschitz, 1988])

- Käsitellään normaaleja logiikkaohjelmia, joissa säännöt ovat muotoa

$$H \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \text{not } B_{m+1} \wedge \cdots \wedge \text{not } B_n$$

- Stabiilien mallien idea:
  (i) $\text{not } p$ on tosi stabiilissa mallissa joss p ei kuulu malliin,
  (ii) Malli on stabiili, jos se on pienin Herbrand-malli niille säännöille, joiden not-literaalit ovat tosia mallissa.

Esimerkki. Ohjelma:
$b(l) \leftarrow eng(l) \wedge \text{not } a(l)$
$a(l) \leftarrow mba(l) \wedge \text{not } b(l)$
$eng(l)$
$mba(l)$

Stabiili malli:
$\{eng(l), mba(l), b(l)\}$.

Slide 4

Määritelmä 1 *Olkoon P muuttujaton ohjelma ja M joukko atomeja. P^M on ohjelmaklausuulijoukko, joka saadaan ohjelmasta P poistamalla (i) jokainen sääntö, jossa on $\text{not } B$, jolle $B \in M$ ja (ii) kaikki $\text{not } B$ jäljelle jääneistä säännöistä.*

Määritelmä 2 *Olkoon P muuttujaton logiikkaohjelma. Muuttujattomien atomilauseiden joukko M on ohjelman P stabiili malli joss M on ohjelmaklausuulijoukon P^M pienin malli.*

Esimerkki.
Ohjelma P:
$p \leftarrow \text{not } q \wedge r$
$q \leftarrow \text{not } p$
$r \leftarrow \text{not } s$
$s \leftarrow \text{not } p$

$M_1 = \{r, p\}$ on P:n stabiili malli, koska
$P^{M_1} = \{p \leftarrow r, r\}$
ja M_1 on P^{M_1}:n pienin malli.
Esim. $M_2 = \{p, s\}$ ei ole P:n stabiili malli, koska
$P^{M_2} = \{p \leftarrow r\}$ ja tämän pienin malli on $\{\}$.
($\{s, q\}$ on myös P:n stabiili malli.)

Slide 5

- $P^M = \{H \leftarrow B_1 \wedge \cdots \wedge B_m \mid$
  $H \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \mathrm{not}\ B_{m+1} \wedge \cdots \wedge \mathrm{not}\ B_n \in P,$
  $\{B_{m+1}, \ldots, B_n\} \cap M = \emptyset\}$
- Määritellään operaattori

  $$\Gamma_P(M) = \{A \mid A \text{ on atomi, } P^M \models A\} = \mathrm{lfp}(\mathrm{T}_{P^M})$$

- M on ohjelman P stabiili malli joss $M = \Gamma_P(M)$.
- Γ_P on antimonotoninen: jos $M \subseteq M'$, $\Gamma_P(M') \subseteq \Gamma_P(M)$ koska jos $M \subseteq M'$, $P^{M'} \subseteq P^M$.
- Stabiilit mallit ovat *minimaalisia*: jos M on ohjelman P stabiili malli, mikään $M' \subset M$ ei ole ohjelman P stabiili malli.
- Stabiilit mallit ovat *hyvin perustettuja*: jos M on ohjelman P stabiili malli, $A \in M$ joss $P^M \models A$.

Slide 6

Muuttujalliset ohjelmat:

Logiikkaohjelmalle määritellään Herbrand-instanssit kuten Horn-klausuuleille.

Määritelmä 3 *Olkoon P logiikkaohjelma ja P_g sen Herbrand-instanssien joukko.*
Muuttujattomien atomilauseiden joukko M on ohjelman P stabiili malli joss M on ohjelmaklausuulijoukon P_g^M pienin malli ($M = \Gamma_{P_g}(M)$).

Esimerkki. Ohjelma P:	Herbrand-instanssi P_g:
$b(X) \leftarrow eng(X) \wedge \mathrm{not}\ a(X)$	$b(l) \leftarrow eng(l) \wedge \mathrm{not}\ a(l)$
$a(X) \leftarrow mba(X) \wedge \mathrm{not}\ b(X)$	$a(l) \leftarrow mba(l) \wedge \mathrm{not}\ b(l)$
$eng(l)$	$eng(l)$
$mba(l)$	$mba(l)$

Stabiilit mallit: $\{eng(l), mba(l), a(l)\}$ ja $\{eng(l), mba(l), b(l)\}$.

Slide 7

Esimerkkejä

Logiikkaohjelmat ja stabiilit mallit tarjoavat varsin joustavan tavan mallittaa myös dynaamisia tilanteita.

- Sääntö, jolla poikkeuksia:

  $fly(X) \leftarrow b(X) \wedge \mathrm{not}\ exc\_bird(X)$
  $exc\_bird(X) \leftarrow pinguin(X)$
  $exc\_bird(X) \leftarrow no\_wings(X)$
  $exc\_bird(X) \leftarrow bird(X) \wedge footinconcrete(X)$
  $\vdots$

- Sääntö, jolla vaihtoehtoisia johtopäätöksiä:

  $lo(X) \leftarrow \mathrm{not}\ hi(X) \wedge nondet(X) \wedge precond(X)$
  $hi(X) \leftarrow \mathrm{not}\ lo(X) \wedge nondet(X) \wedge precond(X)$

Slide 8

- Tyypillisesti logiikkaohjelmointijärjestelmän tehtävä on vastata annettuun kyselyyn (kyllä/ei -vastaus tai vastaussijoitus).
- Stabiilit mallit tarjoavat mahdollisuuden logiikkaohjelman *rajoiteohjelmointitulkintaan:*

  (i) Ohjelman säännöt nähdään rajoitteina ohjelman malleille.
  (ii) Sääntöjen antamien rajoitteiden ratkaisuna on joukko atomeja (stabiilit mallit).
  (iii) Rajoiteohjelmointijärjestelmän tehtävänä on löytää ratkaisujoukkoja.

- Näin saatu rajoiteohjelmointi paradigma on varsin ilmaisuvoimainen:

  Monen tyypilliset rajoiteohjelmointitehtävät voidaan suoraviivaisesti palauttaa ongelmaksi löytää logiikkaohjelmalle stabiili malli.

Slide 9

Lauselogiikan toteutuvuusongelma:

Olkoon S joukko klausuuleja. Muodostetaan ohjelma P_S seuraavasti:
(i) jokaiselle S:n atomille a, otetaan käyttöön atomit a and $\hat{a}$ ja kaksi sääntöä

$$\hat{a} \leftarrow \text{not } a \text{ ja } a \leftarrow \text{not } \hat{a}$$

(ii) Jokaiselle S:n klausuulille otetaan käyttöön uusi atomi c, ja jokaiselle klausuulin literaalille l yksi sääntö seuraavasti:
(a) jos l on positiivinem atomi a, otetaan $c \leftarrow a$ ja
(b) jos l on $\neg a$, otetaan $c \leftarrow \hat{a}$.

Propositio 4 *Klausuulijoukko S on toteutuva joss ohjelmalla P_S on stabiili malli, joka sisältää atomit $\{c_1, \ldots, c_n\}$ missä $c_1, \ldots, c_n$ ovat S:n klausuuleille käyttöönotetut atomit.*

Slide 10

Esimerkki. Klausuulijoukko S:

$a \vee \neg b$

$\neg b \vee \neg a$

$b \vee a$

Ohjelma P_S:

$a \leftarrow \text{not } \hat{a}$
$\hat{a} \leftarrow \text{not } a$
$b \leftarrow \text{not } \hat{b}$
$\hat{b} \leftarrow \text{not } b$
$c1 \leftarrow a$
$c1 \leftarrow \hat{b}$
$c2 \leftarrow \hat{b}$
$c2 \leftarrow \hat{a}$
$c3 \leftarrow b$
$c3 \leftarrow a$

S on toteutuva joss ohjelmalla P_S on stabiili malli, joka sisältää atomit $\{c1, c2, c3\}$.

Slide 11

3-väritys:

Olkoon graafi G annettuna tietokantana, jossa on kaksi relaatiota:

$vertex(\cdot)$ $\quad$ $arc(\cdot, \cdot)$

Muodostetaan ohjelma P_G^{3c}, joka muodostuu tietokannasta ja seuraavista säännöistä:

$color(V, blue) \leftarrow vertex(V) \wedge \text{not } color(V, green) \wedge \text{not } color(V, red)$
$color(V, green) \leftarrow vertex(V) \wedge \text{not } color(V, blue) \wedge \text{not } color(V, red)$
$color(V, red) \leftarrow vertex(V) \wedge \text{not } color(V, green) \wedge \text{not } color(V, blue)$
$\perp \leftarrow arc(V1, V2) \wedge color(V1, C) \wedge color(V2, C).$

Propositio 5 *Graafilla G on 3-väritys joss ohjelmalla P_G^{3c} on stabiili malli, joka ei sisällä atomia $\perp$.*

Slide 12

Hamiltonin silmukka:
Olkoon graafi G annettuna tietokantana, jossa on kaksi relaatiota:
$vertex(\cdot)$ $\quad$ $arc(\cdot, \cdot)$
Muodostetaan ohjelma $P^H(G)$, joka muodostuu tietokannasta ja seuraavista säännöistä:
$hc(V1, V2) \leftarrow arc(V1, V2) \wedge \text{not } otherroute(V1, V2)$
$otherroute(V1, V2) \leftarrow arc(V1, V2) \wedge hc(V1, V3) \wedge \text{not } (V2 = V3)$
$otherroute(V1, V2) \leftarrow arc(V1, V2) \wedge hc(V3, V2) \wedge \text{not } (V1 = V3)$
$reached(V2) \leftarrow hc(V1, V2) \wedge reached(V1).$
$reached(V2) \leftarrow hc(V1, V2) \wedge initialnode(V1).$
$\perp \leftarrow vertex(V), \text{not } reached(V)$
sekä valitusta alkusolmusta $initialnode(v)$.
Ohjelmalla $P^H(G)$ on stabiili malli, jossa ei ole atomia $\perp$ joss G:llä on Hamiltonin silmukka (polku, joka käy jokaisessa solmussa täsmälleen kerran ja palaa alkusolmuunsa).

Slide 13

Kompakti karakterisointi

- not-literaaleilla keskeinen rooli stabiilien mallien määrittelyssä
- $\mathrm{NA}(P)$: ohjelmassa P not-literaaleissa esiintyvät atomilauseet.
- Ohjelman P stabiileja malleja voidaan karakterisoida joukon $\mathrm{NA}(P)$ osajoukoilla.
- Määritellään joukolle literaaleja F: $\mathrm{Dcl}(P, F)$ on ohjelmaklausuulien

  $$\begin{aligned} P(F) = \; & \{H \leftarrow B_1 \wedge \cdots \wedge B_m \mid \\ & H \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \mathrm{not}\ B_{m+1} \wedge \cdots \wedge \mathrm{not}\ B_n \in P, \\ & \{\mathrm{not}\ B_{m+1}, \ldots, \mathrm{not}\ B_n\} \subseteq F\} \end{aligned}$$

  pienin malli.

  Huom! $\mathrm{Dcl}(P, F) = \Gamma_P(\{A \mid \mathrm{not}\ A \notin F\})$

Slide 14

Määritelmä 6 *Joukkoa F not-literaaleja kutsutaan P-täydeksi joss kaikille $A \in \mathrm{NA}(P)$, $\mathrm{not}\ A \in F$ joss $A \notin \mathrm{Dcl}(P, F)$.*

Teoreema 7 *Olkoon P muuttujaton ohjelma.*
(i) Jos F on P-täysi, $\mathrm{Dcl}(P, F)$ on P:n stabiili malli.
(ii) Jos M on P:n stabiili malli, $F = \{\mathrm{not}\ A \mid A \in \mathrm{NA}(P) - M\}$ on P-täysi ja $M = \mathrm{Dcl}(P, F)$.

☞ P-täysi joukko on stabiilin mallin komplementti not-lit. suhteen.

Esimerkki.	$\mathrm{NA}(P) = \{q, p, s\}$
Ohjelma P:	$F_1 = \{\mathrm{not}\ s, \mathrm{not}\ q\}$ on P-täysi, koska
$p \leftarrow \mathrm{not}\ q \wedge r$	$P(F_1) = \{p \leftarrow r, r\}$ ja $\mathrm{Dcl}(P, F_1) = \{r, p\}$.
$q \leftarrow \mathrm{not}\ p$	Esim. $F_2 = \{\mathrm{not}\ q\}$ ei ole P-täysi, koska
$r \leftarrow \mathrm{not}\ s$	$P(F_2) = \{p \leftarrow r\}$ ja $\mathrm{Dcl}(P, F_2) = \{\}$.
$s \leftarrow \mathrm{not}\ p$	Myös $F_3 = \{\mathrm{not}\ p\}$ on P-täysi, koska
	$P(F_3) = \{q, s\}$ ja $\mathrm{Dcl}(P, F_3) = \{q, s\}$.

Slide 15

Approksimointi

- P:n stabiileja malleja voidaan approksimoida operaattorilla Γ_P.
- Olkoon M "varmat" johtopäätökset.

  (i) $$\begin{aligned} P^M = \; & \{H \leftarrow B_1 \wedge \cdots \wedge B_m \mid \\ & H \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \mathrm{not}\ B_{m+1} \wedge \cdots \wedge \mathrm{not}\ B_n \in P, \\ & \{B_{m+1}, \ldots, B_n\} \cap M = \emptyset\} \end{aligned}$$

  (ii) $\Gamma_P(M)$: "potentiaalisesti" johtopäätökset.

  (iii) $\Gamma_P^2(M) = \Gamma_P(\Gamma_P(M))$: "varmat johtopäätökset".
- Γ_P on antimonotoninen.
- Γ_P^2 on monotoninen.
- Operaattorilla on pienin kiintopiste $\mathrm{lfp}(\Gamma_P^2)$.

Slide 16

- Kaikille ohjelman P stabiileille malleille M pätee:

  (a) $\mathrm{lfp}(\Gamma_P^2) \subseteq M$ ja

  (b) $M \subseteq \Gamma_P(\mathrm{lfp}(\Gamma_P^2))$.
- Well-founded semantics [Van Gelder et al., 1988]: osittaismalli (literaalijoukko)

  Määritelmä 8 *Muuttujattoman ohjelman P WF-malli on literaalijoukko*

  $$\mathrm{lfp}(\Gamma_P^2) \cup \{\mathrm{not}\ A \mid A \notin \Gamma_P(\mathrm{lfp}(\Gamma_P^2))\}$$

- WF-malli on aina olemassa (mutta $\mathrm{lfp}(\Gamma_P^2)$ saavuttamiseksi saatetaan tarvita yli ω iteraatiota jos P on ääretön).

Slide 17

Esimerkki. a) Ohjelmalle $P = \{p \leftarrow \text{not } p\}$

$\Gamma_P(\emptyset) = \{p\}$
$\Gamma^2_P(\emptyset) = \{\}$
WF-malli: $\{\}$.

b) Ohjelmalle

$$P' = \{p_1 \leftarrow \text{not } p_0,\quad p_2 \leftarrow \text{not } p_1,\quad p_3 \leftarrow \text{not } p_2,\quad q_1 \leftarrow p_3 \wedge \text{not } q_2,\quad q_2 \leftarrow p_3 \wedge \text{not } q_1\}$$

$$\Gamma_{P'}(\emptyset) = \{p_1, p_2, p_3, q_1, q_2\}$$
$$\Gamma^2_{P'}(\emptyset) = \{p_1\}$$
$$\Gamma_{P'}(\{p_1\}) = \{p_1, p_3, q_1, q_2\}$$
$$\Gamma^2_{P'}(\{p_1\}) = \{p_1, p_3\}$$
$$\Gamma_{P'}(\{p_1, p_3\}) = \{p_1, p_3, q_1, q_2\}$$
$$\Gamma^2_{P'}(\{p_1, p_3\}) = \{p_1, p_3\}$$

WF-malli: $\{p_1, p_3, \text{not } p_0, \text{not } p_2\}$.

Slide 18

Yksikäsitteisyys

- Logiikkaohjelmalla saattaa olla useampia tai ei yhtään stabiilia mallia.
- Milloin stabiileja malleja täsmälleen yksi?
- Keskeisin tällainen ohjelmaluokka: *kerrostuneet ohjelmat* (stratified programs).
- Määritellään ohjelmalle P riippuvuusgraafi seuraavasti:

  (i) Graafin solmut ovat ohjelmassa esiintyvät atomit.
  (ii) Kaari (A, B) kuuluu graafiin joss ohjelmassa on sääntö, jonka pää on A ja rungossa esiintyy atomi B. Kaarta (A, B) kutsutaan negatiiviseksi, joss B esiintyy säännössä not-literaalissa.

Slide 19

- Graafin vahvasti kytketyt komponentit (solmujoukot, joissa kustakin solmusta pääsee mihin tahansa muuhun solmuun jotakin polkua pitkin) antavat toistensa suhteen rekursiivisesti määritellyt atomit.
- Ohjelma on *kerrostunut*, joss sen riippuvuusgraafin vahvasti kytketyissä komponenteissa ei ole negatiivisia kaaria.
- Kerrostuneella ohjelmalla on yksikäsitteinen stabiili malli (joka on myös ohjelman WF-malli).
- Ohjelman kerrostuneisuus voidaan selvittää lineaarisessa ajassa:

  Graafin vahvasti kytketyt komponentit voidaan laskea lineaarisessa ajassa graafin koon suhteen.

Slide 20

Esimerkki. Ohjelma

$$P = \{p \leftarrow q \wedge \text{not } r,\quad q \leftarrow p \wedge \text{not } s,\quad r \leftarrow \text{not } t,\quad s \leftarrow t,\quad t \leftarrow p\}$$

ei ole kerrostunut:
riippuvuusgraafissa yksi vahvasti kytketty komponentti $\{p, q, r, s, t\}$, jossa negatiivinen kaari.

Jos viimeinen sääntö jätetään pois, ohjelma on kerrostunut:
riippuvuusgraafissa vahvasti kytketyt komponentit: $\{p, q\}, \{r\}, \{s\}, \{t\}$.

Slide 21

Toteutustekniikkaa

[Niemelä and Simons, 1996]
Muuttujattomat ohjelmat

- Käytetään kompaktia karakterisointia (täydet joukot)
  ☞ hakuavaruus: not-literaalit
- Bottom-up haku
- Hakuavaruuden karsinta:
  (i) Oletukset tutkittavasta mallista tehdään yksi kerrallaan.
  (ii) Ennen seuraavan oletuksen teko lasketaan approksimaatio kaikille stabiileille malleille, jotka toteuttavat annetut oletukset ja tutkitaan ollaanko jo konfliktissa.

☞ Davis-Putnam -menetelmän tyylinen tekniikka

Slide 22

- Algoritmi säilyttää approksimaatiota stabiileista malleista literaalijoukkona (osittaismallina) FS:
  (i) Jos $A \in FS$, A kuuluu kaikkiin tutkittaviin stabiileihin malleihin.
  (ii) Jos $\mathrm{not}\ A \in FS$, A ei kuulu tutkittaviin stabiileihin malleihin ($\mathrm{not}\ A$ kuuluu täyteen joukkoon).
- Approksimaatio lasketaan käyttäen stabiilien mallien ala- ja ylärajaa:
  $\mathrm{LB}(P, FS)$ on literaalijoukko, jonka jokainen FS:n toteuttava ohjelman P stabiili malli toteuttaa.
  $\mathrm{UB}(P, FS)$ on atomilauseiden joukko, joka sisältää jokaisen ohjelman P stabiilin mallin, joka toteuttaa FS.
- Approksimaatio **expand**(P, FS) on pienin joukko FS', joka sisältää FS ja on suljettu seuraavien sääntöjen suhteen:
  (i) jos $A \in \mathrm{LB}(P, FS')$, $A \in FS'$ ja
  (ii) jos $A \notin \mathrm{UB}(P, FS')$, $\mathrm{not}\ A \in FS'$.

Slide 23

Approksimaatioiden toteutus

- Tavoite: ala- ja yläraja laskettavissa lineaarisessa ajassa.
- Vähintään yhtä tarkat rajat kuin WF-mallilla.
- Hakutilanne (osittaismalli) hyödynnettävä tehokkaasti.
- Oletuksilla FS vielä käytettävissä olevat säännöt:
  $P_{FS} = \{H \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \mathrm{not}\ B_{m+1} \wedge \cdots \wedge \mathrm{not}\ B_n \in P \mid \{\mathrm{not}\ B_1, \ldots, \mathrm{not}\ B_m, B_{m+1}, \ldots, B_n\} \cap FS = \emptyset\}$
- $\overline{L}$ on literaalin L:n komplementti:
  $\overline{\mathrm{not}\ A} = A$ ja $\overline{A} = \mathrm{not}\ A$.

Slide 24

$\mathrm{LB}(P, FS)$: pienin joukko FS', joka sisältää FS ja on suljettu seuraavien sääntöjen suhteen

1. Jos $H \leftarrow L_1 \wedge \cdots \wedge L_n \in P_{FS'}$ ja $\{L_1, \ldots, L_n\} \subseteq FS'$, $H \in FS'$.
2. Jos atomi H ei ole päänä ohjelmassa $P_{FS'}$, $\mathrm{not}\ H \in FS'$.
3. Jos $H \in FS'$ on päänä vain yhdessä säännössä $H \leftarrow L_1 \wedge \cdots \wedge L_n \in P_{FS'}$, $\{L_1, \ldots, L_n\} \subseteq FS'$.
4. Jos $\mathrm{not}\ H \in FS'$, $H \leftarrow L_1 \wedge \cdots \wedge L_n \in P_{FS'}$ ja $\{L_1, \ldots, L_{i-i}, L_{i+1}, \ldots, L_n\} \subseteq FS'$, $\overline{L_i} \in FS'$.

$\mathrm{UB}(P, FS)$: pienin malli ohjelmaklausuulijoukolle, joka saadaan poistamalla not-literaalit ohjelmasta P_{FS}.

Slide 25

Esimerkki. a) P :

$p \leftarrow \text{not } q$

$r \leftarrow \text{not } p$

$q \leftarrow p \wedge \text{not } r \wedge \text{not } s$

$s \leftarrow \text{not } t \wedge r$

$\text{LB}(P, \{\text{not } q\}) =$
$\{\text{not } q, p, \text{not } r, s, \text{not } t, r\}$

b) P' :

$p \leftarrow q$

$q \leftarrow p$

$p \leftarrow \text{not } r$

$r \leftarrow \text{not } s$

$s \leftarrow \text{not } p \wedge \text{not } q$

$\text{LB}(P', \{r\}) = \{r\}$

$\text{UB}(P', \{r\}) = \{r, s\}$

$\text{LB}(P', \{r, \text{not } p, \text{not } q\}) =$
$\{r, \text{not } p, \text{not } q, s, \text{not } r\}$

$\text{UB}(P', \{r, \text{not } p, \text{not } q, s, \text{not } r\}) = \{s\}$

expand$(P, \{r\}) =$
$\{r, \text{not } p, \text{not } q, s, \text{not } r\}$

Slide 26

- Ala- ja ylärajoille voidaan antaa lineaarisen ajan toteutukset (Dowling-Gallier -tyyppiset tietorakenteet).
- Tarkkuus vähintään sama kuin WF-mallilla:
  Jos oletuksia ei ole, **expand**$(P, \{\})$ antaa ohjelman P WF-mallin.
- Hakutilanne (oletukset FS) hyödynnetään dynaamisesti.
  Esim. ohjelmalle P
  $p \leftarrow \text{not } q$
  $q \leftarrow \text{not } p$
  $p \leftarrow q$
  expand$(P, \{q\})$ palauttaa $\{q, p, \text{not } q, \text{not } p\}$.
  Huom! Ohjelman $P \cup \{q\}$ WF-malli on $\{q, p\}$
  (konflikti ei tule esille).

Slide 27

Algoritmi

function smodels(P, FS, φ)
$FS' :=$ **expand**(P, FS);
if conflict(P, FS') returns true **then** return false
else if $\text{NA}(P)$ is covered by FS' **then** return **test**(P, FS', φ)
else
  $\chi :=$ **choose**(P, FS');
  else if smodels$(P, FS' \cup \{\chi\}, \varphi)$ returns true **then**
    return true
  else return **smodels**$(P, FS' \cup \{\overline{\chi}\}, \varphi)$
  end if
end if

Slide 28

- Funktiota **test** käytetään löytyneen mallin testaukseen (φ on testiparametri).
- Funktio **conflict** tarkistaa, sisältääkö saatu approksimaatio konfliktin ($\{A, \text{not } A\} \subseteq FS$).
- Funktio **choose** toteuttaa hakuheuristiikan.
- **smodels** suorittaa *fokusoitua mallinetsintää*:
  smodels(P, FS, φ) palauttaa true joss on olemassa P:n stabiili malli M, joka toteuttaa FS ja **test**(P, FS', φ) palauttaa true missä $M = \text{Dcl}(P, FS')$.

Slide 29

Esimerkki.

a) Tutkitaan, kuuluuko t P:n kaikkiin stabiileihin malleihin etsimällä vastamalli, jossa t ei ole. Aloitetaan $FS = \{\text{not } t\}$ ja **test** palauttaa aina true.

$p \leftarrow \text{not } q$	not t	
$q \leftarrow \text{not } p$	s	
$r \leftarrow p$	not r	
$r \leftarrow q$	p	not p
$s \leftarrow \text{not } r$	r	q
$t \leftarrow \text{not } s$	$\times$	r
		$\times$

Vastaus: kyllä

b) Etsitään ohjelman P

$p \leftarrow \text{not } q \wedge r$

$q \leftarrow \text{not } p$

$r \leftarrow \text{not } s$

$s \leftarrow \text{not } p$

stabiilit mallit: (aluksi $FS = \{\}$)

p	not p
not s	q
r	s
not q	not r

Löytyi $\text{Dcl}(P, \{p, \text{not } s, r, \text{not } q\})$ ja $\text{Dcl}(P, \{\text{not } p, q, s, \text{not } r\})$

Slide 30

Heuristiikka

- lookahead-tekniikka:
  jokaiselle $A \in \text{NA}(P)$, jota FS ei peitä, lasketaan
  $FS_1 = \textbf{expand}(P, FS \cup \{A\})$ ja $FS_2 = \textbf{expand}(P, FS \cup \{\text{not } A\})$
- Saadaan tarkempi alaraja:
  Jos FS_1 sisältää konfliktin, lisätään not A joukkoon FS ja
  jos FS_2 sisältää konfliktin, lisätään A joukkoon FS.
- FS_1 and FS_2 antavat arvion jäljellä olevasta hakuavaruudesta.
- Heuristiikka: valitaan atomi, jolla on pienin arvioitu jäljellä oleva hakuavaruus.

Slide 31

Muuttujalliset ohjelmat

- Jos ohjelmassa ei ole funktiosymboleja, Herbrand-instansseja äärellinen määrä.
- Kaikkia ei tarvita, kun halutaan säilyttää stabiilit mallit.
- Riittävä osajoukko voidaan suoraviivaisesti laskea, kun ohjelman muuttujat rajattuja (range-restricted): jokainen muuttuja, joka esiintyy säännön päässä tai not-literaalissa, esiintyy myös positiivisessa literaalissa säännön rungossa.
- Säännöt voidaan muuntaa rajattuun muotoon, ottamalla käyttöön muuttujille tarvittavat domain-määrittelyt.
  Esim. $eq(X, X) \quad \mapsto \quad eq(X, X) \leftarrow dom(X)$
- Herbrand-instanssien määrä saattaa kasvaa hyvin suureksi.

Slide 32

Arviointia

- Logiikkaohjelmat ja stabiilit mallit ilmaisuvoimainen rajoiteohjelmointi paradigma.
  (Esim. Boolen rajoitteet/lauselogiikka erikoistapaus).
- Hakuavaruuden rakenne poikkeaa lauselogiikan toteutuvuusongelmasta:
  (i) Vain not-literaalit lisäävät hakuavaruutta.
  (ii) Säännöt suunnattuja: Kerrostuneisuus rajaa hakuavaruutta.
  ☞ Enemmän vapautta tietämyksen esittämiseen
- Toteutustekniikat parantuneet ratkaisevasti.
  Esimerkiksi lineaarinen tilavaativuus.