Sääntöpohjainen päättely II

- Horn-klausuulit tarjoavat loogisen pohjan sääntöpohjaiselle päättelylle.
- Tyypillisesti 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.

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

  ?- $b(l)$

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

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

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

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

- 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 $P^M \models A$.

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:

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

Herbrand-instanssi P_g:

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

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

Esimerkkejä

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

- Sääntö, jolla poikkeuksia:

  $fly(X) \leftarrow b(X) \wedge \text{not } exc_bird(X)$
  $exc_bird(X) \leftarrow pinguin(X)$
  $exc_bird(X) \leftarrow no_wings(X)$
  $exc_bird(X) \leftarrow bird(X) \wedge footinconcrete(X)$
  $\vdots$

- Sääntö, jolla vaihtoehtoisia johtopäätöksiä:

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

- 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 malli).
  (iii) Rajoiteohjelmointijärjestelmän tehtävänä on löytää ratkaisujoukkoja.

- Näin saatu rajoiteohjelmointi paradigma on varsin ilmaisuvoimainen:

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

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.*

Esimerkki. Klausuulijoukko S:

$a \vee \neg b$

$\neg b \vee \neg a$

$b \vee a$

Ohjelma P_S:

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

Klausuulijoukko S on toteutuva **joss**
ohjelmalla P_S on stabiili malli M siten, että $\{c1, c2, c3\} \subseteq M$.

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:

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

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

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:
$hc(V1, V2) \leftarrow arc(V1, V2) \wedge \text{not } otherroute(V1, V2)$
$otherroute(V1, V2) \leftarrow arc(V1, V2) \wedge hc(V1, V3) \wedge \text{not } (V2 = V3)$
$otherroute(V1, V2) \leftarrow arc(V1, V2) \wedge hc(V3, V2) \wedge \text{not } (V1 = V3)$
$reached(V2) \leftarrow hc(V1, V2) \wedge initialnode(V1).$
$reached(V2) \leftarrow hc(V1, V2) \wedge reached(V1) \wedge \text{not } initialnode(V1).$
$\bot \leftarrow vertex(V), \text{not } reached(V)$
sekä valitusta alkusolmusta $initialnode(v)$.
Ohjelmalla $P^H(G)$ on stabiili malli, jossa ei ole atomia $\bot$ joss G:llä on Hamiltonin silmukka (polku, joka käy jokaisessa solmussa täsmälleen kerran ja palaa alkusolmuunsa).

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 joukon

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

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

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

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

Approksimointi

- P:n stabiileja malleja voidaan approksimoida operaattorilla Γ_P.
- Antakoon M "varmat" johtopäätökset.
  (i) $P^M = \{\ H \leftarrow B_1 \wedge \cdots \wedge B_m \mid H \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \mathrm{not}\ B_{m+1} \wedge \cdots \wedge \mathrm{not}\ B_n \in P,\ \{B_{m+1}, \ldots, B_n\} \cap M = \emptyset\ \}$
  (ii) $\Gamma_P(M)$: "potentiaaliset" 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 $\mathrm{lfp}(\Gamma_P^2)$.

- 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*

  $$\mathrm{lfp}(\Gamma_P^2) \cup \{\mathrm{not}\ A \mid A \in HB(P)\ \textit{and}\ A \notin \Gamma_P(\mathrm{lfp}(\Gamma_P^2))\}$$

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

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

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

b) Ohjelmalle

$$
\begin{array}{lll}
P' = & \{p_1 \leftarrow \text{not } p_0, & \Gamma_{P'}(\emptyset) = \{p_1, p_2, p_3, q_1, q_2\} \\
& p_2 \leftarrow \text{not } p_1, & \Gamma_{P'}^2(\emptyset) = \{p_1\} \\
& p_3 \leftarrow \text{not } p_2, & \Gamma_{P'}(\{p_1\}) = \{p_1, p_3, q_1, q_2\} \\
& q_1 \leftarrow p_3 \wedge \text{not } q_2, & \Gamma_{P'}^2(\{p_1\}) = \{p_1, p_3\} \\
& q_2 \leftarrow p_3 \wedge \text{not } q_1\} & \Gamma_{P'}(\{p_1, p_3\}) = \{p_1, p_3, q_1, q_2\} \\
& & \Gamma_{P'}^2(\{p_1, p_3\}) = \{p_1, p_3\}
\end{array}
$$

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

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.

- 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.

Esimerkki. Ohjelma

$$
\begin{array}{ll}
P = & \{p \leftarrow q \wedge \text{not } r, \\
& q \leftarrow p \wedge \text{not } s, \\
& r \leftarrow \text{not } t, \\
& s \leftarrow t, \\
& t \leftarrow p\}
\end{array}
$$

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:
$\{t\}, \{s\}, \{r\}, \{p, q\}$.