3. kotitehtävän palautus

- Jokainen resoluutioaskel kohtaan 1 (yleisimmät unifioijat, muuttujien nimeäminen). Kohta 1 **ei ole** OTTER-tehtävä.
- Ratkaisuksi laadittu OTTER-syötetiedosto kohtaan 2 ja selitys jokaisesta lauseesta (samankaltaisina toistuvia lauseita ja atomilauseita ei tarvitse erikseen selittää)
- Muutama ajoesimerkki:
  - OTTERin löytämiä todistuksia (pelkkä todistus riittää)
  - kyselyitä joille OTTER ei löytänyt todistusta, ja joille ei pitänytkään todistusta löytyä (jotta näkyy, ettei "tietokanta" ole ristiriitainen).
- Palautettava viimeistään 1.12.1997.

Prolog-ohjelmat predikaattilogiikassa

Prolog	Klausuuliesitys	Vastaava lause
`p(X)`	$\{P(x)\}$	$\forall x(P(x))$
`r(X,Y) :- q(X)`	$\{R(x,y), \neg Q(x)\}$	$\forall x \forall y(Q(x) \rightarrow R(x,y))$
`:-q(X),s(X)`	$\{\neg Q(x), \neg S(x)\}$	$\neg \exists x(Q(x) \wedge S(x))$

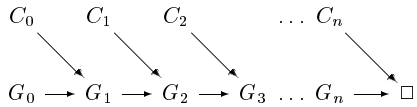
- Ohjelmaklausuulien muuttujat ovat universaalisti kvantifioidut ja kyselyn muuttujat ovat eksistentiaalisesti kvantifioidut.

Listanotaatio:

Prolog	Vastaava termi
`[]`	e
`[X|[Y|Z]]`	$c(x, c(y, z))$
`[X,Y,Z]`	$c(x, c(y, c(z, e)))$

Lineaarinen input-resoluutio

Olkoon P joukko klausuuleja ja G kysely (sisältää vain negatiivisia literaaleja). Allaoleva on refutaatio *lineaarisella input-resoluutiolla* (LI), joss $G_0 = G$ ja kukin C_i on P:n klausuuli, missä muuttujat on tarvittaessa uudelleennimetty.



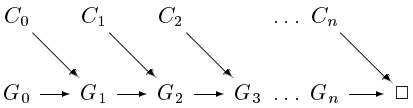
Väite. (Virheettömyys ja täydellisyys) Olkoon P joukko ohjelmaklausuuleja ja G kysely. Nyt $P \cup \{G\}$ on toteutumaton, jos ja vain jos on olemassa $P \cup \{G\}$:n refutaatio lineaarisella input-resoluutiolla siten, että refutaatio alkaa klausuulista $G_0 = G$.

Todistus: (Virheettömyys) Resoluutioaskeleet säilyttävät toteutuvuuden. (Täydellisyys) Lineaarinen resoluutio takaa refutaation olemassaolon (T II.14.4). Klausuuleissa G_i on vain negatiivisia literaaleja, joten jokaisella askeleella klausuuli C_i on välttämättä ohjelmaklausuuli (jossa on siis täsmälleen yksi positiivinen literaali).

Järjestetyt Horn-klausuulit

Olkoot $G = \{\neg A_0, \ldots, \neg A_n\}$ ja $C = \{B, \neg B_0, \ldots, \neg B_m\}$ järjestettyjä klausuuleita, ja θ literaalien A_i ja B yleisin unifioija. Klausuuleista G ja C saadaan resoluutiolla järjestetty klausuuli $G = \{\neg A_0, \ldots, \neg A_{i-1}, \neg B_0, \ldots, \neg B_m, \neg A_{i+1}, \ldots, \neg A_n\}\theta$.

Olkoon P joukko järjestettyjä klausuuleita ja G järjestetty kysely. Allaoleva on refutaatio *LD-resoluutiolla* (Linear Definite), jos $G_0 = G$ ja kukin C_i on P:n klausuuli, missä muuttujat on tarvittaessa uudelleennimetty.



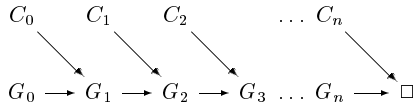
LD-resoluutio valintasäännöllä = SLD-resoluutio

Valintasääntö on funktio R, joka valitsee järjestetystä ei-tyhjästä klausuulista C literaalin $L = R(C)$.

Prologin valintasääntö R_P:

Klausuulille $C = \{\neg A_1, \ldots, \neg A_n\}$, $R_P(C) = \neg A_1$.

Olkoon P joukko järjestettyjä klausuuleita ja G järjestetty kysely. Allaoleva on refutaatio *SLD-resoluutiolla* (Linear Definite), jos $G_0 = G$ ja kukin C_i on P:n klausuuli (muuttujat uudelleennimetty), ja *askeleella i resoluutiosääntöä sovellettiin klausuuleihin G_i ja C_i literaaliin $R(G_i)$.*

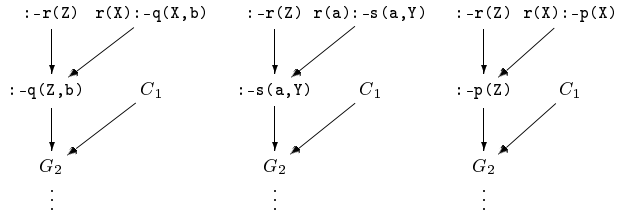


Tämä on edelleen virheetön ja täydellinen resoluutiomenetelmä (suhteessa $P \cup \{G\}$ toteutumattomuuteen).

Prologin hakumekanismi

```
r(X) :- q(X,b).
r(a) :- s(a,Y).
r(X) :- p(X).
...
```

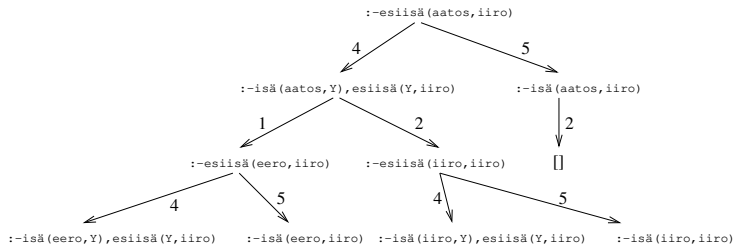
Kysely: `:-r(Z)`



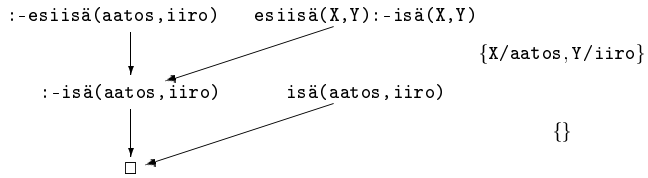
Esimerkki

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

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

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

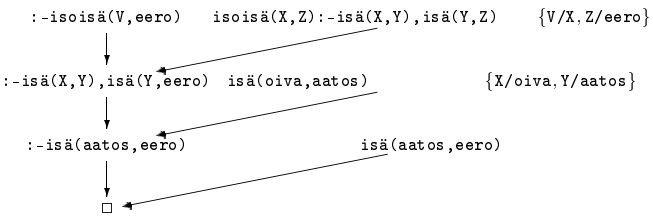

Prologin hakupuu



Tuloksena saatava lineaarinen resoluutiotodistus

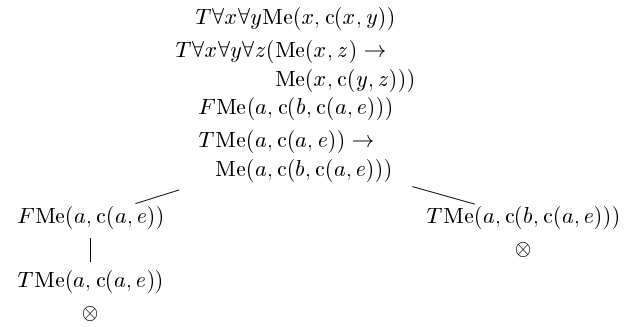


Vastauksena sidokset kyselyn muuttujille



Unifioijien kompositio: {V/oiva, X/oiva, Z/eero, Y/aatos}

Esimerkki

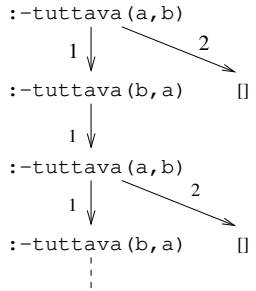



```
Me(X,[X|Y]).    Me(X,[Y|Z]) :- Me(X,Z).    :- Me(a,[b,a])
```


Syvyyshaku: epätäydellisyys

```
1: tuttava(X,Y) :- tuttava(Y,X)
2: tuttava(a,b)
```

Kysely: `:-tuttava(a,b)`



Prologin not-operaattori

- Halutaan käyttää muunkinlaisia sääntöjä kuin $P(t_1, \ldots, t_n)$ ja $P(t_1, \ldots, t_n)$:- $P_1(s_{11}, \ldots, s_{1n_1}), \ldots, P_m(s_{m1}, \ldots, s_{mn_m})$. Esim. ei-operaatiota säännön ehto-osassa.
- Koko predikaattilogiikka liian monimutkainen: ohjelmien suorittaminen ei olisi kyllin tehokasta
- Laajennetaan resoluutioproseduuria:
  - Käytetään predikaattilogiikan negaation "¬" asemesta toisentyyppistä negaatiota "`not`".
  - Tulokset käytännön sovelluksissa useimmiten haluttuja

NOT – negaatio epäonnistumisena

"Jos ei voida päätellä ϕ, päätellään `not` ϕ"

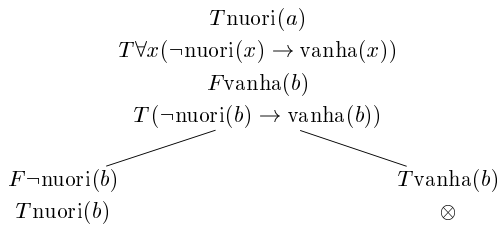
```
koululainen(oskari).
koululainen(emilia).
sairas(oskari).
koulussa(X) :- koululainen(X), not sairas(X).
```

1. Kysely: `:-koulussa(oskari)`
2. Kysely: `:-koulussa(emilia)`

Ero klassiseen negaatioon

```
nuori(a).
vanha(X) :- not nuori(X).
```

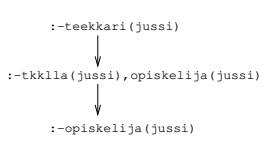
Prolog vastaa kyselyyn vanha(b) myönteisesti. Lause vanha(b) ei ole kuitenkaan ohjelman looginen seuraus, jos klausuuleissa esiintyvä Prologin negaatio tulkitaan predikaattilogiikan negaatioksi:



Määritelmä. *Äärellinen epäonnistunut SLD-puu* P:lle G:lle ja R:lle on äärellinen SLD-resoluution hakupuu, jonka mikään lehti ei ole tyhjä klausuuli $\square$.

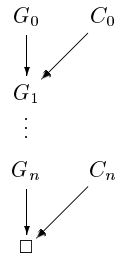
Äärellinen epäonnistunut SLD-puu kyselylle `teekkari(jussi)`:

```
teekkari(X) :- tkklla(X),
               opiskelija(X).
opiskelija(teemu).
tkklla(jussi).
tkklla(teemu).
```



SLDNF-resoluutio

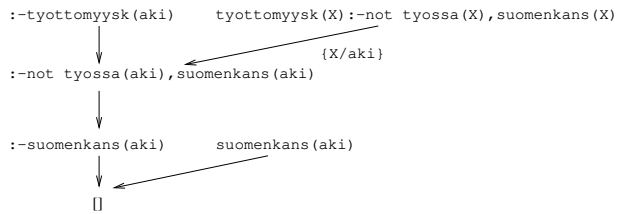
Olkoon P joukko järjestettyjä klausuuleita ja G järjestetty kysely. Vieressä oleva on refutaatio *SLDNF-resoluutiolla* (Selection function, Linear Definite, Negation as Failure), jos $G_0 = G$ ja kukin C_i on P:n klausuuli (mahdollisesti muuttujat uudelleennimettynä), ja



- $R(G_i)$ on not L, L on muuttujaton, ja L:llä P:llä ja R:llä on äärellinen epäonnistunut SLDNF-puu ja G_{i+1} on $G_i - \{\text{not } L\}$, tai
- askeleella i suoritettiin resoluutio klausuuleille G_i ja C_i literaalilla $R(G_i)$.


```
suomenkans(aki).
suomenkans(eve).
tyossa(eve).
tyottomyysk(X) :- not tyossa(X), suomenkans(X).
```

Kyselylle `:-tyottomyysk(aki)` saadaan seuraava SLDNF-refutaatio:



Huomaa, että alikysely `:-tyossa(aki)` epäonnistuu.

Prolog-ohjelman täydentäminen

$$P(t_{11},\ldots,t_{1n}) \text{ :- } l_{11},\ldots,l_{1m_1}$$
$$\vdots$$
$$P(t_{r1},\ldots,t_{rn}) \text{ :- } l_{r1},\ldots,l_{rm_r}$$

P-predikaatin klausuulit täydenetään lauseeksi

$$\begin{aligned}&\forall x_1\cdots\forall x_n\ (P(x_1,\ldots,x_n)\leftrightarrow\\ &\quad(\exists y_{11}\cdots\exists y_{1v_1}(x_1=t_{11}\wedge\cdots\wedge x_n=t_{1n}\wedge l_{11}\wedge\cdots\wedge l_{1m_1})\vee\\ &\quad\cdots\\ &\quad\vee\exists y_{r1}\cdots\exists y_{rv_r}(x_1=t_{r1}\wedge\cdots\wedge x_n=t_{rn}\wedge l_{r1}\wedge\cdots\wedge l_{rm_r})))\end{aligned}$$

Ohjelmaa P vastaava lausejoukko Comp(P) sisältää täydennettyjen klausuulien lisäksi yksikäsitteisten nimien aksioomat.

Ohjelma

```
suomenkansalainen(aki).
suomenkansalainen(eve).
tyossa(eve).
tyottomyyskorvausta(X) :- not tyossa(X),
                          suomenkansalainen(X).
```

on täydennettynä

$$\begin{aligned}&\forall x\ (\text{suomenkansalainen}(x)\leftrightarrow(x=\text{aki}\vee x=\text{eve}))\\ &\forall x\ (\text{tyossa}(x)\leftrightarrow(x=\text{eve}))\\ &\forall x\ (\text{tyottomyyskorvausta}(x)\leftrightarrow\\ &\qquad(\neg\ \text{tyossa}(x)\wedge\ \text{suomenkansalainen}(x)))\\ &\neg(\text{aki}=\text{eve})\end{aligned}$$

Väite. Jos not-operaatiota sisältävällä ohjelmalla P ja kyselyllä G $= \{\neg L_1,\ldots,\neg L_m\}$ on SLDNF-refutaatio, niin

$$\text{Comp(P)} \models \exists x_1\cdots\exists x_n(L_1\wedge\cdots\wedge L_m).$$

Esimerkki. Kaikkia Comp(P):n loogisia seurauksia ei saada todistetuksi SLDNF-resoluutiolla:

`v(X) :- not q(X).`	$\forall x(v(x)\leftrightarrow\neg q(x))$
`q(a) :- not r(a).`	$q(a)\leftrightarrow\neg r(a)$
`r(a) :- p(a).`	$r(a)\leftrightarrow(p(a)\vee\neg p(a))$ eli $r(a)$
`r(a) :- not p(a).`	$\forall x(p(x)\leftrightarrow p(f(x)))$
`p(X) :- p(f(X)).`	

Kyselyyn `:-v(a)` ei saada vastausta, vaikka Comp(P) $\models v(a)$.

Jos negaation yhteydessä käytetään muuttujia sisältäviä literaaleja, SDLNF-resoluution virheettömyys menetetään (floundering):

```
koululainen(oskari).
sairas(liisa).
koulussa(X) :- not sairas(X), koululainen(X).
```

Kysely `:-koulussa(X)` epäonnistuu.

$$\begin{aligned}\text{Comp(P)} = \{\ &\forall x(\text{koululainen}(x)\leftrightarrow x=\text{oskari}),\\ &\forall x(\text{sairas}(x)\leftrightarrow x=\text{liisa}),\\ &\forall x(\text{koulussa}(x)\leftrightarrow(\neg\text{sairas}(x)\wedge\text{koululainen}(x))),\\ &\neg(\text{oskari}=\text{liisa})\}\end{aligned}$$

Mutta: Comp(P) $\models \exists x\text{koulussa}(x)$.