Logiikka tietotekniikassa: perusteet

Luennot: tiistai, klo 10-12, sali T1

Luennoitsija: TkT Tomi Janhunen (@hut.fi, 451 3255)

Laskuharjoitukset: alkavat viikolla 37

1. ryhmä: tiistai, klo 15-16, sali T2
2. ryhmä: keskiviikko, klo 9-10, sali T2
3. ryhmä: perjantai, klo 9-10, sali T2

Laskuharjoitusassistentit:

DI Tommi Syrjänen (@hut.fi, 451 5082)

DI Toni Jussila (@hut.fi, 451 5245)

Vastaanotto: syksyllä tiistaisin klo 15.15-16.00 huone B335

Tiedottaminen: ensisijaisesti kurssin kotisivu

`http://www.tcs.hut.fi/Teaching/Tik-79.144/`

sekä ilmoitustaulut ja otaxin uutisryhmä `opinnot.tik.logiikka`.

Kirjallisuus:

- A. Nerode ja R. A. Shore, *Logic for Applications*, Springer-Verlag, 1997. Kolme ensimmäistä lukua (luennoilla esitetyssä laajuudessa).
- Opetusmonisteissa toimitetaan laskuharjoitusten mallivastaukset ja luentokalvot (Paketit 1-3; esitellään myöhemmin)

1. tentti: tiistai 19. joulukuuta, klo 12-15, salit DE T1

Kolme pakollista/henkilökohtaista kotitehtävää!

Näkemyksiä loogiseen päättelyyn

- **Inhimillinen päättely**

  Ihmisen suorittama päättely.

  Ihmisen päättelykyky on rajallinen ja päättelyyn käytettävissä oleva aika usein rajallinen.
- **Formaali päättely**

  Täsmällinen matemaattinen malli loogisesta päättelystä.

  Malli on ideaalinen (ei em. rajoituksia) ja abstrakti.

  Päättely voidaan nähdä symbolien käsittelynä.
- **Automaattinen/mekaaninen päättely**

  Toteutetaan looginen päättely tietokoneohjelmana (päättelykone).

Kurssin sisältö

Paketti 1: Lauselogiikka

- syntaksi ja semantiikka
- todistusteoria
- semanttiset/analyyttiset taulut
- normaalimuodot
- resoluutiosääntö

Paketti 2: Predikaattilogiikka

- lauselogiikan yleistys

Paketti 3: Päättelyn automatisointi

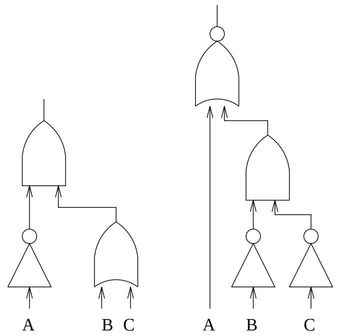
- teoreemantodistimet
- logiikkaohjelmointi

Logiikan sovelluskohteita tietojenkäsittelyssä

Voidaan nähdä karkea kahtiajako:

- **Järjestelmän ominaisuuksien määrittely ja analysointi**
  Ohjelmien oikeellisuustarkastelut
  Ehtolausekkeiden muokkaaminen
  Tarkistetaan, että järjestelmä toteuttaa annetun spesifikaation
- **Päättelykomponentti järjestelmän osana**
  Toteutetaan loogista päättelyä jollain tasolla
  Logiikkaohjelmointi, rajoiteohjelmointi, sääntöpohjainen päättely

Esimerkki: kombinatoriset piirit



Lauselogiikan lauseina: $\neg A \wedge (B \vee C)$ ja $\neg(A \vee (\neg B \wedge \neg C))$

Esimerkki: ohjelmien ehtolausekkeet

Olkoon `myptr` tyyppiä "`char *`" C-kielessä.

```
/* TAPA 1 */

if(myptr != NULL && myptr[0] == '/') doit(myptr);
if(!(myptr == NULL || myptr[0] == '.')) dothat(myptr);

/* TAPA 2 */

if(myptr != NULL) {
  if(myptr[0] == '/') doit(myptr);
  if(myptr[0] != '.') dothat(myptr);
}
```


Esimerkki: ohjelmankehitys

Tiedetään: jos X1 < 0 tai X2 < 0 tai ... tai Xn < 0, niin Z < 0

Vertaa:

```
if (Z < 0)
    do_0()
else {
    if (X1 < 0)
        do_1();
    if (X2 < 0)
        do_2();
    ...
    if (Xn < 0)
        do_n();
}
```

```
if (Z < 0)
    do_0();
```

Esimerkki: ohjelmien oikeellisuuden toteaminen

- **Induktio** tietorakenteiden suhteen

```
duplicate([],[]).
duplicate([E|R1],[E|[E|R2]]) :- duplicate(R1,R2).
```

  Voidaan osoittaa *rakenteisella induktiolla*, että proseduuri `duplicate` käsittelee oikein mielivaltaisen pitkiä listoja.

- **Induktio** silmukan suorituskertojen suhteen

```
n = 0; while(n<10) n = n+1;
```

  Voidaan osoittaa, että n ≤ 10 on silmukan *invariantti* (ominaisuus joka säilyy silmukkaa suoritettaessa).

Esimerkki: relaatiotietokannat

- relaatioiden R_1 ja R_2 unioni

$$\forall \overline{x}(R_1(\overline{x}) \vee R_2(\overline{x}) \to P(\overline{x}))$$

- relaatioiden R_1 ja R_2 leikkaus

$$\forall \overline{x}(R_1(\overline{x}) \wedge R_2(\overline{x}) \to P(\overline{x}))$$

- projektio $XYZ \mapsto XZ$

$$\forall x \forall y \forall z(R(x,y,z) \to P(x,z))$$

- relaatioiden R_1 ja R_2 natural join (alla y:n suhteen)

$$\forall x \forall y \forall z(R_1(x,y) \wedge R_2(y,z) \to R(x,y,z))$$

Esimerkki: deduktiiviset tietokannat

sisarus(a,b)
sisarus(c,e)
vanhempi(a,c)
vanhempi(d,e)

$\forall x \forall y(\text{sisarus}(x,y) \to \text{sisarus}(y,x))$
$\forall x \forall y \forall z(\text{sisarus}(x,y) \wedge \text{sisarus}(y,z) \to \text{sisarus}(x,z))$
$\forall x \forall y(\text{vanhempi}(x,y) \to \neg\text{vanhempi}(y,x))$
$\forall x \forall y \forall z \forall v(\text{vanhempi}(x,y) \wedge \text{vanhempi}(z,v) \wedge \text{sisarus}(x,z) \to$
$\text{serkku}(y,v))$

Esimerkki: logiikkaohjelmointi

```
append([],L,L).
append([A|T],L,[A|S]) :- append(T,L,S).

?- append([1,2,3],[4,5,6],X).

X = [1,2,3,4,5,6]

?- append([1,X,3],Y,[1,2,3,4]).

X = 2, Y=[4]
```

Vrt. yleinen näkemys: PROGRAM = LOGIC + CONTROL