Diskreetit rakenteet, syksy 2011

Itä-Suomen yliopisto, Tietojenkäsittelytieteen laitos

Simo Juvaste

13.12.2011 10:02

Sisältö

1 Johdanto

Johdatus kurssin sisältöön, esimerkkejä

Kurssin tavoitteita :

- Oppia loogista *ajattelua*
  - Voiko loogista ajattelua opettaa???
- Oppia loogista *päättelyä*
- Oppia *täsmällistä* loogista päättelyä

- Oppia pukemaan täsmällinen looginen päättely *muodolliseen muotoon*
  - Tietojärjestelmien määrittely, suunnittelu, toteutus ja testaus
  - Määrittelystä toteutukseen
  - Määrittelystä testaukseen
- Oppia ymmärtämään tietokoneen toimintaa
  - Miten tietokone toimii
  - Miten tietokone päättelee loogisesti
  - Miten tietokone saadaan päättelemään loogisesti
- Oppia tiedon esittämistä digitaalisessa muodossa
- Kerrata laskentoa muita kursseja varten

Historiaa

- Jo muinaiset kreikkalaiset...
- Samat tekniikat ovat syntyneet useasti, pienin variaatioin, eri terminologioin

1. Filosofian (ajattelun) väline
2. Matematiikan väline
3. Tietojenkäsittelyn väline

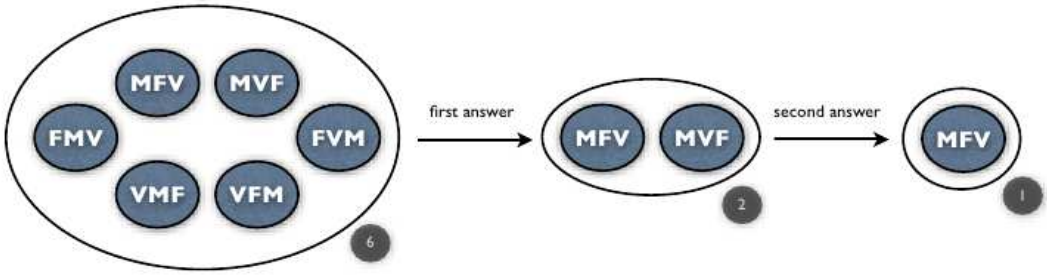
- Kts. Logic in Action, kohta 1.2

1.1 Esimerkkejä loogisesta päättelystä

- Kolme asiakasta istuu pöydässä, yksi on tilannut kalaa, toinen lihaa, kolmas kasviksia.
- Tilausten järjestystä tuntematon tarjoilija tulee tuomaan annoksia.
- Tarjoilija kysyy "Kenelle tulee liha", ja antaa liha-annoksen sille joka sen osoittaa haluavansa.
- Tarjoilija kysyy "Kenelle tulee kala", ja antaa kala-annoksen sille joka sen osoittaa haluavansa.
- Sitten tarjoilija antaa kasvisannoksen oikealle henkilölle ilman lisäkysymyksiä!
- Kuinka hän osasi!?

- Merkitään: Liha = M, Kala = F, Kasvis = V.
- Viimeiselle henkilölle valinta: M tai V tai K, ei F, ei $M \to V$.

Päättelyn eteneminen:



Esimerkkejä loogisesta päättelystä

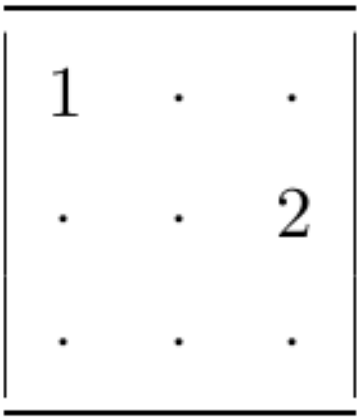
Kolme loogisesti päättelevää toisiaan tuntematonta asiakasta istuu ravintolassa.

- Loogisesti päättelevä tarjoilija kysyy "Ottavatko kaikki kahvia?".
- 1. asiakas vastaa "En tiedä".
- 2. asiakas vastaa "En tiedä".
- 3. asiakas vastaa "Ei, kaikki eivät ota kahvia".

Tarjoilija osaa tuoda kaikille halukkaille kahvit oikein!

- Keille hän toi kahvit ja miten hän sen päätteli?

Esimerkkejä loogisesta päättelystä



3x3 Sudoku

Esimerkkejä loogisesta päättelystä

Faktoja ja seurauksia :

Meneekö näin?

Jos opiskelet ahkerasti, niin pääset töihin
Opiskelet ahkerasti

Niinpä pääset töihin ??

Meneekö näin?

Jos opiskelet ahkerasti, niin pääset töihin
Pääset töihin

Niinpä opiskelit ahkerasti ??

Meneekö näin?

Jos opiskelet ahkerasti, niin pääset töihin
Et opiskele ahkerasti

Niinpä et pääse töihin ??

Meneekö näin?

Jos opiskelet ahkerasti, niin pääset töihin
Et pääse töihin

Niinpä et ole opiskellut ahkerasti ??

[Vast: TEET]

Looginen päättely (inference) , merkitään:

joko $\dfrac{P_1, P_2, \ldots, P_n}{C}$, tai $\dfrac{\begin{array}{c} P_1 \\ P_2 \\ \ldots \\ P_n \end{array}}{C}$

Päättely on siis "merkitty vaakaviivalla". Päättelymme on kelvollinen jos ja vain jos aina kun kaikki *premissit* ("lähtökohdat") $P_1, P_2, \ldots, P_n$ ovat oikein, niin johtopäätös C on oikein. Toisin sanoen:

1. jos kaikki premissit $P_1, P_2, \ldots, P_n$ ovat tosia, niin johtopäätös C on myös tosi.
2. jos johtopäätös C on väärin, vähintään yksi premissi P_i on väärin.

Oikea päättelykuvio :

Jos opiskelet ahkerasti, niin pääset töihin
Opiskelet ahkerasti

Niinpä pääset töihin

Jos A, niin B
A

Niinpä B

Oikea päättelykuvio :

Jos opiskelet ahkerasti, niin pääset töihin
Et pääse töihin

Niinpä et ole opiskellut ahkerasti

Jos A, niin B
ei B

Niinpä ei A

Esimerkki suomeksi

Kokonaisluku x on parillinen tai pariton
Jos x on parillinen, niin $x + x$ on parillinen
Jos x on pariton, niin $x + x$ on parillinen

Niinpä $x + x$ on parillinen.

Miten päättelimme loogisesti?

Kokonaisluku x on A_1 tai A_2
Jos x on A_1, niin C
Jos x on A_2, niin C

Niinpä C.

Huom: Opimme myös myöhemmin todistamaan lauseen "Jos x on pariton, niin $x + x$ on parillinen"(induktiotodistus).

Miten tunnistamme kelvollisen päättelyn? LinA slides, Ch2, page 13 (33)

Joukko-oppia käytetään hyväksi tietokantojen käsittelyssä.

Ajatellaan, että meillä on kolme tietokantaa:

1. opiskelijoiden nimet ja osoitteet-tietokanta
   Taulu 1:

Nimi	Osoite

2. opiskelijoiden nimet, pääaineet
   Taulu 2:

Nimi	Pääaine

3. opiskelijat ja harrastukset
   Taulu 3:

Nimi	harr 1	harr 2	...

Haluamme esim. kaikkien tietojenkäsittelytiedettä lukevien, jalkapalloa harrastavien opiskelijoiden osoitteet selville. Muodostamme tauluista (joukoista) 1 ja 2 yhdisteen $1 \cup 2$, jolloin saamme kaikkien opiskelijoiden pääaineet ja osoitteet samaan tauluun (joukkoon). Sitten otamme yhdistetaulun ja taulun 3 leikkauksen siten, että taulusta 3 valitsemme vain jalkapallon. Näin löydämme tietokannasta jalkapalloa pelaavien opiskelijoiden osoitteet. Otamme vielä siitä taulusta TKT:n lukijat.

Huom! Homman voi tehdä monella eri tavalla. MySQL:ssä voidaan tehdä kysely.

Esim: Asiantuntijajärjestelmissä käytetään *predikaattilogiikkaa*. Tämän käytölle on kehitetty myös omia logiikkaohjelmointikieliä, kuten PROLOG.

Esim: Onko vaikkapa sukupuu jatkuvaa ja diskreettiä tietoa? Sukupuu esitetään dikreettinä rakenteena.

Kurssin tavoite Koettakaa oppia eräitä tekniikoita ja menetelmiä, sekä saamaan pohjaa ymmärrykselle (diskreetistä) tiedon esittämisestä tietokoneella.

2 Propositiologiikka

Logiikassa määrittelemme

1. Atomaarisia ilmaisuja (*propositioita* eli väitelauseita) jotka voivat olla tosia tai epätosia.
2. *Loogisia yhdysmerkkejä* (konnektiiveja) joilla muodostamme uusia propositioita (yleensä: 'ja' ($\wedge$), 'tai' ($\vee$), 'ei' ($\neg$), 'jos, niin' ($\rightarrow$), 'jos ja vain jos' (joss, $\leftrightarrow$)).
3. *Päättelysääntöjä* jotka määrittelevät yksikäsitteisesti millaisen johtopäätöksen voimme annetuista propositoista tehdä sekä sen, onko johtopäätös tosi vai epätosi.

Esimerkki

1. Jos matkapuhelinten kysyntä kasvaa, niin yrityksen on laajennettava
2. Jos yritys laajentaa, niin sen on lisättävä työntekijöitä
3. Jos matkapuhelinten kysyntä ..., niin yrityksen on lisättävä ...

1 ja 2 ovat *premissejä* (lähtökohta, ehto), 3 on *seurauslause* (johtopäätös).

Huom! Tyypillisiä muotoja: JOS A NIIN B: Useissa ohjelmointikielissä IF A THEN B. Propositio on siis A.

Loogisen päättelyn ideana on, että jos premissit ovat tosia ja käytetään oikeita (valideja) päättelysääntöjä, niin johtopäätös on oikea.

Jos johtopäätös ei ole oikea, on selvitettävä onko

1. premissit väärin vai
2. käytetty virheellisiä päättelysääntöjä

Esim:

1. Tässä ohjelmssa on virhe, tai syöte on virheellinen
2. Syöte ei ole virheellinen
3. Ohjelmassa on virhe

Huom! Tässä kohdassa 1 atomaarisia ilmiasuja ovat “ohjelmassa...” ja ”syöte on...”, Niitä yhdistää TAI-konnektiivi.

Käyttämällä formaalia esitystä, jossa atomaariset ilmaisut esitetään isoilla kirjaimilla, saamme e.o. päättelyt muotoon

1. Jos P, niin Q
2. Jos Q, niin R

---

3. Jos P, niin R

Tämä päättely on pätevä mille tahansa ilmaisuille P, Q ja R. Olkoot lauseet esim.

P : Kissa näkee kultakalan
Q : Kissa nappaa kultakalan
R : Kissa syö kultakalan

Tällöin päättely tulee muotoon

1. Jos kissa näkee kultakalan, niin kissa nappaa kultakalan
2. Jos kissa nappaa kultakalan, niin kissa syö kultakalan

---

3. Jos kissa näkee kultakalan, niin kissa syö kultakalan.

Tästä esimerkistä näemme, että päättelysääntö on yleinen, lauseiden sisällöstä riippumaton looginen sääntö.

Huom! Johtopäätöksen (3) oikeellisuus (totuus) riippuu premissien oikeellisuudesta (totuudesta).

Samoin voimme kirjoittaa ”Tässä ohjelmssa on virhe . . . ” -esimerkin muotoon

1. P tai Q
2. Ei Q

---

3. P

Huom! Tämä on esimerkki *päättelysäännöistä*, kun loogista päättelyä toteutetaan tietokoneella, esim. asiantuntijajärjestelmissä.

Seuraava päättelysääntö on *modus ponens* (implikaation eliminointi, ”etujäsenen myöntösääntö”):

1. Jos P, niin Q
2. P

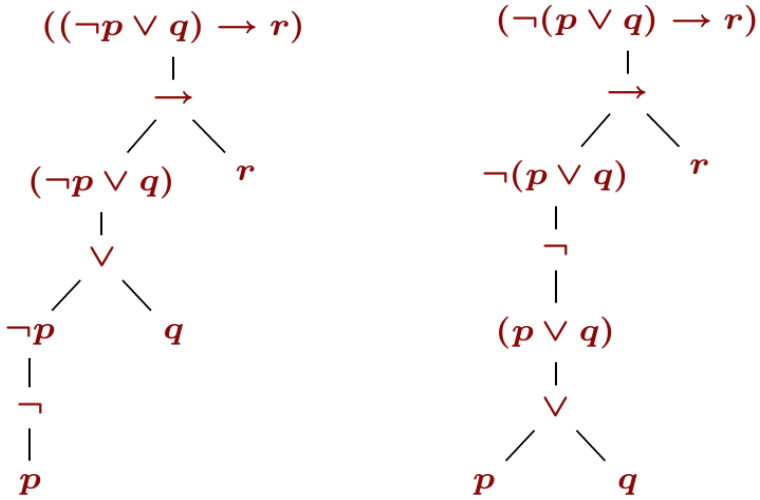
---

3. Q

Esimerkiksi P: Liikennevalo on punainen
Q: Auto pysähtyy

Päättelysääntöihin, niiden johtamiseen ja todistamiseen palaamme myöhemmin.

Loogisen kaavan esittäminen: Monimutkaisempi looginen kaava voidaan esittää lausekkeena suluilla tai puuna joka myös näyttää miten lausekkeen arvo muodostuu. [LiA]



Kaksiarvologiikka

Määritelmä 1: Väitettä, joka on tosi tai epätosi, kutsutaan propositioksi.

Huom! Emme siis voi käsitellä kaikkia mahdollisia väitteitä. Esimerkiksi lause "numero 8 tuoksuu kauhealta" ei ole tosi eikä epätosi, vaan järjetön (ainakin kun 8 on vain abstrakti numero eikä esine)

Myös järkevät väitteet voivat olla totuusarvoltaan epäselviä. Esim. em. esimerkin tietokoneohjelman virhe voi olla myös ohjelman määrittelyssä.

P, Q ja R edellä ovat *muuttujia* ja voivat saada vain arvot *tosi* (T) tai *epätosi* (E). Arvot T ja E ovat vakioita.

2.1 Eri konnektiivien totuustaulut

Määritelmä 2: Propositio, jossa on vain yksi muuttuja tai yksi vakio on *atomilause*. Kaikki muut lauseet ovat *yhdistettyä lauseita*. Kaikissa yhdistetyissä lauseissa on ainakin yksi konnektiivi.

Kun tiedämme premissien mahdolliset totuusarvot (valuaatiot) ja käytämme oikeita päättelysääntöjä, voimme selvittää johtopäätöksen totuusarvon kaikilla premissien totuusarvokombinaatioilla *totuustaulua* käyttäen.

Tarkastellaan kahta esimerkkiä:

1. Pariisi on Ranskassa ja $2 + 2 = 4$
   Pariisi on Ranskassa ja $2 + 2 = 5$
   Pariisi on Englannissa ja $2 + 2 = 4$
   Pariisi on Englannissa ja $2 + 2 = 5$

2. Pariisi on Ranskassa tai $2 + 2 = 4$
   Pariisi on Ranskassa tai $2 + 2 = 5$
   Pariisi on Englannissa tai $2 + 2 = 4$
   Pariisi on Englannissa tai $2 + 2 = 5$

Milloin koko väittämän looginen totuus tuntuu luonnolliselta?

Tutuimmat konnektiivit ovat 'ja' ($\wedge$) ja 'tai' ($\vee$). Näille käytetään myös merkintöjä

$P \wedge Q$ P & Q $P{\cdot}Q$ PQ P AND Q

$P \vee Q$ $P + Q$ P OR Q

Huom! Totuusarvo merkitään T, E, tai T, F tai $1, 0$. Negaatio (kieltolauseen ilmaisu), merkitään $\neg P$ tai NOT P

[LiA slides Ch02 s. 20 (84)]

Konjunktion, disjunktion ja negaation totuustaulut ovat

P	Q	$P \wedge Q$
T	T	T
T	E	E
E	T	E
E	E	E

P	Q	$P \vee Q$
T	T	T
T	E	T
E	T	T
E	E	E

P	$\neg P$
T	E
E	T

Huom! Disjunktion totuustaulussa ilmaisu on tosi myös kun molemmat atomilauseet ovat tosia. Tästä käytetään myös nimitystä "inclusive OR". Jos tarvitaan operaatio joka on epätosi tässä tilanteessa, on kyseessä "exclusive OR, XOR, $\veebar$".

Implikaation ja ekvivalenssin totuustaulut :

P	Q	$P \rightarrow Q$
T	T	T
T	E	E
E	T	T
E	E	T

P	Q	$P \leftrightarrow Q$
T	T	T
T	E	E
E	T	E
E	E	T

Yhdistettyjen lauseiden totuustaulut Merkitään atomilauseista $P, Q, \ldots$ koostuvaa loogista ilmaisua yleisesti $f(P, Q, \ldots)$.

Esim

$$f(P, Q, R) = (P \vee Q) \vee \neg R$$

Loogisen ilmaisun $f(P, Q, R, \ldots)$ totuustaulu saadaan muodostamalla askelittain konnektiiveja vastaavat totuustaulut.

[LiA slides Ch02 s. 24 (118)

Huom! Konnektiivien suoritusjärjestys on $\neg, \wedge, \vee$
siis $\neg P \wedge Q = (\neg P) \wedge Q$

Esim.

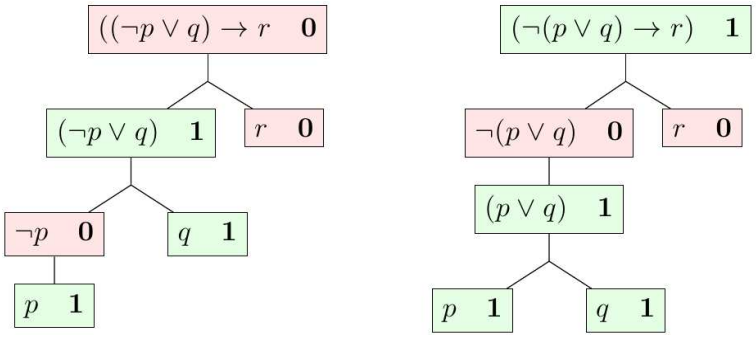
$$f(P, Q) = \neg(P \wedge \neg Q)$$

P	Q	$\neg Q$	$P \wedge \neg Q$	$\neg(P \wedge \neg Q)$
T	T	E	E	T
T	E	T	T	E
E	T	E	E	T
E	E	T	E	T

Toinen merkintätapa:

P	Q	$\neg($	P	$\wedge$	$\neg Q)$
T	T	T	T	E	E
T	E	E	T	T	T
E	T	T	E	E	E
E	T	T	E	E	T
askel		3	1	2	1

Kolmas merkintätapa : [LiA slides Ch02 s. 19 (68)]



Määritelmä 3: Mikäli ilmaisu $f(P, Q, \ldots)$ on tosi kaikilla valuaatioilla (atomilauseiden $P, Q, \ldots$ totuusarvokombinaatioilla), sanotaan ilmaisua *tautologiaksi*! Jos ilmaisu taas ei ole tosi millään valuaatiolla, on ilmaisu *kontradiktio*. Muuten se on (mahdollisesti) *toteutuva* (satisfiable). Tautologian totuustaulun viimeisen sarakkeen kaikki arvot ovat T ja kontradiktion E. Joukko ilmaisuja X on toteutuva jos on olemassa jokin valuaatio jolla kaikki X:n ilmaisut ovat tosia.

Esim. Tautologia: $P \vee \neg P$, kontradiktio: $P \wedge \neg P$.

Määritelmä 4: Ilmaisut $f(P, Q, \ldots)$ ja $g(P, Q, \ldots)$ ovat loogisesti ekvivalentteja jos niiden totuustaulut ovat identtiset. Merkitään

$$f(P, Q, \ldots) \equiv g(P, Q, \ldots)$$

Esim:

$$f(P, Q) = \neg(P \wedge Q)$$

$$g(P, Q) = \neg P \vee \neg Q$$

P	Q	$\neg$	$(P$	$\wedge$	$Q)$
T	T	**E**	T	T	T
T	E	**T**	T	E	E
E	T	**T**	E	E	T
E	E	**T**	E	E	E
askel					

P	Q	$\neg P$	$\vee$	$\neg Q$
T	T	E	**E**	E
T	E	E	**T**	T
E	T	T	**T**	E
E	E	T	**T**	T
askel				

Esim Loogiset ilmaisut (propositiot):

Ei ole niin, että ruusut ovat punaisia ja orvokit sinisiä

ja

Ruusut eivät ole punaisia tai orvokit eivät ole sinisiä

ovat loogisesti ekvivalentteja.

P = "ruusut ovat punaisia", Q = "orvokit ovat sinisiä"

$\neg(P \wedge Q) \quad \equiv \quad \neg P \vee \neg Q$

Ilmaisuja (propositioita) koskevia lakeja

$P \vee P \equiv P$

$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$

$P \wedge P \equiv P$

$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$ (de Morganin laki)

Todistukset?

Kuinka monta eri totuustaulua (konnektiivia) kahdella propositiolla voi yhteensä olla? [LiA slides Ch02 p.39 (226)]

P	Q		$\rightarrow$		$\leftrightarrow$	
T	T		T		T	
T	E		E		E	
E	T		T		E	
E	E		T		T	

$\rightarrow$: implikaatio, seuraus, jos P niin Q.

$\leftrightarrow$: ekvivalenssi, P jos ja vain jos Q.

Nämä viisi konnektiivia: konjunktio, disjunktio, negaatio, implikaatio ja ekvivalenssi muodostavat erään konnektiivien perusjoukon. Muita joskus käytettyjä ovat joko-tai ("XOR", $\veebar$, $\oplus$), Shefferin viiva ("ei molemmat") ja Peircen nuoli ("ei kumpikaan"). Nämä kuitenkin saadaan ilmaistua ylläkuvatuilla operaatioilla (HT).

Konnektiivien keskinäisestä suhteesta

Em. "ylimääräiset" konnektiivit voidaan ilmaista aiemmin esittelemillämme "peruskonnektiiveillä".

Esim XOR: $P \veebar Q \quad \equiv \quad P \leftrightarrow \neg Q \quad \equiv \quad (P \vee Q) \wedge \neg(P \wedge Q)$:

P	Q	$P \veebar Q$			$P \leftrightarrow \neg Q$				$(P \vee Q) \wedge \neg(P \wedge Q)$							
1	1	1		1	1		0	1	1	1	1			1		1
1	0	1		0	1			0	1		0			1		0
0	1	0		1	0			1	0		1			0		1
0	0	0		0	0			0	0		0			0		0

Täydennetään luennolla, tai täydennä itse.

Vastaavasti ”peruskonnektiiveillä” voidaan ilmaista muita ”peruskonnektiivejä:”

Vertaillaan lauseita $P \wedge Q$ ja $\neg(\neg P \vee \neg Q)$:

P	Q	$P \wedge Q$					$\neg(\neg P \vee \neg Q)$			
1	1	1		1			1			1
1	0	1		0			1			0
0	1	0		1			0			1
0	0	0		0			0			0

Vertaillaan lauseita $P \rightarrow Q$ ja $\neg P \vee Q$:

P	Q	$P \rightarrow Q$				$\neg P \vee Q$			
1	1	1		1		1			1
1	0	1		0		1			0
0	1	0		1		0			1
0	0	0		0		0			0

Vertaillaan lauseita $P \leftrightarrow Q$ ja $(P \rightarrow Q) \wedge (Q \rightarrow P)$:

P	Q	$P \leftrightarrow Q$			$(P \rightarrow Q) \wedge (Q \rightarrow P)$						
1	1	1	1	1	1	1	1	1	1	1	1
1	0	1		0	1		0		0		1
0	1	0		1	0		1		1		0
0	0	0		0	0		0		0		0

Onko tässä jotain tuttua?

Keksitkö muita tapoja esittää jokin konnektiivi toisia käyttäen? (HT)

Lauseiden muuttamista ja yhdistelyä tarvitaan mm. tietojärjestelmän määrittelyssä, suunnittelussa ja toteutuksessa. Erityisesti muutettaessa ohjelman logiikkaa (vaikkapa lisättäessä jokin ehto tai vaihtoehto), on oltava erityisen huolellinen, että ohjelma toimii oikein kaikissa tilanteissa.

Kertausta implikaatiosta

Implikaatio-operaatio $(P \rightarrow Q)$ edellyttää ainoastaan, että jos P on tosi, niin Q on tosi. Jos P ei ole tosi, niin Q voi olla tosi tai epätosi.

Luonnollisessa kielessä implikaatio esiintyy eri tavoin, esimerkiksi

Jos P niin Q	$P \rightarrow Q$
P jos Q	$Q \rightarrow P$
P vain jos Q	$P \rightarrow Q$

Viimeinen kohta voi olla vaikea hahmottaa. Otetaan esimerkiksi: $P =$ ”autan sinua”, $Q =$ ”autat minua”. Lause ”Autan sinua vain jos autat minua” voi olla väärin vain jos ”autan sinua” on tosi mutta ”autat minua” on epätosi.

Muistisääntönä voi auttaa, että implikaatio on käänteinen (kokonaislukujen) suurempi tai yhtäsuuri-vertailusta, siis kun $T = 1$ ja $E = 0$, niin $P \rightarrow Q \ \equiv \ P \leq Q$

2.2 Loogisesta päättelystä

Loogista päättelyä voidaan tarkastella totuustaulujen avulla. Tarkastellaan aiemmin esitettyä *modus ponens* -päättelysääntöä:

1.	P $\rightarrow$ Q	P:stä seuraa Q, (Ulkona sataa $\Rightarrow$ Pipo kastuu)
2.	P	P on voimassa, (Ulkona sataa)
3.	Q	Q on siis voimassa, (Siis pipo kastuu)

Lauseita 1 ja 2 kutsutaan *premisseiksi* ja niistä saadaan *johtopäätös* Q. Koska tämä päättely on loogisesti pätevä, käytetään sille seuraavanlaista merkintää, jossa premissit erotellaan pilkuilla ja johtopäätös laitetaan "haarukan" jälkeen.

$P \rightarrow Q,\ P \vDash Q$

Kuten aiemminkin, tätä ei tarvitse sen kummemmin säikähtää, sillä tämä on ainoastaan lyhennetty merkintätapa.

Tämä voidaan esittää myös niin, että Q on implikaatio premissien konjunktiosta, ts.

$(P \rightarrow Q) \wedge P \Rightarrow Q,$

joka on tautologia jos ja vain jos päättely on pätevä.

Osoitetaan esimerkiksi *modus ponens* pätevyys totuustaulun avulla.

P	Q	$((P \rightarrow Q)$	$\wedge$	$P)$	$\Rightarrow$	Q
T	T	T	T	T	T	T
T	E	E	E	T	T	E
E	T	T	E	E	T	T
E	E	T	E	E	T	E
					$\uparrow$	

Yleinen looginen päättelysääntö

Yleisesti looginen päättelysääntö voidaan esittää seuraavasti: Olkoot premissit $A_1, A_2, \ldots, A_n$ ja niistä tehtävä johtopäätös C oikea, silloin merkitään

$A_1, A_2, \ldots, A_n \vDash C$

ja ilmaus

$A_1 \wedge A_2 \wedge \ldots \wedge A_n \Rightarrow C$

on tautologia.

Edellä oleva esitysmuoto saattaa tuntua hankalalta aluksi, mutta ajattele sitä vain luettelona. Tarkoitus on esittää asia mahdollisimman suppeassa muodossa, mutta kuitenkin formaalisti. Yksittäistapauksissa sijoitetaan lausekkeet kirjainten paikalle, jolloin voimme tutkia päättelyn pätevyyttä.

Loogisessa päättelyssä yleensä käytetään joukkoa päättelysääntöjä. Näitä on maailmalla monenlaisia, mutta nyt tarkastelemme tällaista päättelysääntöjen joukkoa (Grassmann & Tremblay, 1996: pp.50):

1.	$A, B \vDash A \wedge B$	(Yhdistämissääntö)
2.	$A \wedge B \vDash B$ $A \wedge B \vDash A$	(Yksinkertaistamissääntö)
3.	$A \vDash A \vee B$ $B \vDash A \vee B$	(Lisäyssääntö, jos A on tosi, niin disjunktiolauseke on tosi riippumatta B:n totuusarvosta)
4.	$A, A \to B \vDash B$	(*modus ponens*)
5.	$\neg B, A \to B \vDash \neg A$	(*modus tollens*)
6.	$A \to B, B \to C \vDash A \to C$	Hypoteettinen syllogismi
7.	$A \vee B, \neg A \vDash B$ $A \vee B, \neg B \vDash A$	Disjunktiivinen syllogismi (*modus tollendo ponens*), jos toinen ei ole totta, niin toisen pitää olla jotta disjunktiolause olisi tosi.
8.	$A \to B, \neg A \to B \vDash B$	(Vaihtoehtojen laki)
9 .	$A \leftrightarrow B \vDash A \to B$ $A \leftrightarrow B \vDash B \to A$	(Implikaatiosääntö)

Kunkin näistä säännöistä voidaan todistaa oikeaksi totuustauluilla (tai muita sääntöjä käyttäen). Esimerkiksi sääntö 5:

A	B	$(\neg B \wedge (A \to B)) \Rightarrow \neg A$								
1	1	0	1		1	1	1		0	1
1	0		0		1		0			1
0	1		1		0		1			0
0	0		0		0		0			0

Lisää todistuksia harjoitustehtävinä.

Kunkin näistä ”merkitystä” tai ”järkevyyttä” voi myös testata asettamalla taas vaikkapa $P =$ ”autan sinua”, $Q =$ ”autat minua”.

Esimerkiksi sääntö 5: Kun ”Jos autan sinua, niin autat minua” ja ”Et auta minua” ovat tosia, niin väistämättä myös ”en auta sinua” on tosi (koska jos olisin auttanut, niin sinäkin olisit auttanut).

Olettamalla premissit todeksi ja käyttämällä näitä päättelysääntöjä voidaan johtaa uusia loogisia totuuksia (johtopäätöksiä, teoreemoja). Oletetaanpa, että meillä on premissit $A_1, A_2, \ldots, A_n$ ja näistä voidaan johtaa em. tyylisellä päättelyketjulla C. Tällöin merkitsemme:

$$A_1, A_2, \ldots, A_n \vdash C$$

Ero aiemmin käytettyyn kaksipiikkiseen “haarukkaan” ($\vDash$) ei ole suuri. $\vDash$-merkkiä käytetään kun kyseessä on suora päättely, mutta jos johtopäätöksen tekemiseen tarvitaan useita päättelysääntöjä, käytetään merkkiä $\vdash$.

Esim. Todista

1. $\neg C \to D$
2. $D \to A \wedge B$
3. $C \to E$
4. $\neg E$

B

Meidän on siis todistettava, että aina kun premissit 1.-4. ovat voimassa, niin myös B on tosi! Tämä voitaisiin toki tehdä totuustaululla, mutta se olisi melko työlästä kun viidellä muuttujalla on 32 valuaatiota!

Voimme kuitenkin käyttää edelläolleita (todistettuja) päättelysääntöjä ja todistaa päättelyn symbolisesti:

Premissit:

1.	$\neg C \to D$	
2.	$D \to A \wedge B$	
3.	$C \to E$	
4.	$\neg E$	

Päättely (apulauseet):

5.	$\neg C$	premisseistä 4,3 käyttäen sääntöä 5
6.	D	apulauseesta 5 ja premissistä 1 käyttäen sääntöä 4
7.	$A \wedge B$	premissistä 2 ja apulauseesta 6 käyttäen sääntöä 4
8.	B	apulauseesta 7 käyttäen sääntöä 2

Esim. Todista, että logiikka ei ole vaikeaa käyttäen seuraavia premissejä

- joko mehiläisenhoito tai kuvanveisto on hauskaa
- jos logiikka on vaikeaa, niin kuvanveisto ei ole hauskaa
- mehiläisenhoito ei ole hauskaa

Formalisoidaan nämä seuraavasti:

A = “logiikka on vaikeaa”
B = “mehiläisenhoito on hauskaa”
C = “kuvanveisto on hauskaa”

1.	$B \vee C$	premissi	
2.	$A \to \neg C$	premissi	
3.	$\neg B$	premissi	
4.	C	1,3	sääntö 7.
5.	$\neg A$	2,4	sääntö 5. (säännössä $\neg C$)

Siis, logiikka ei ole vaikeaa.

Esim: Tarkastellaan ohjelmointikielen lausetta

```
if x > Max then x := Max
```

Haluamme osoittaa, että lauseen suorituksen jälkeen ei ole mahdollista että $x > Max$. Formalisoidaan lause

P = $x > Max$ ennen lauseen suoritusta
Q = $x = Max$ lauseen suorituksen jälkeen
R = $x > Max$ lauseen suorituksen jälkeen

Selvästikin voimme sanoa

1.	$P \rightarrow Q$	premissi
2.	$Q \rightarrow \neg R$	premissi
3.	$\neg P \rightarrow \neg R$	premissi
4.	$P \rightarrow \neg R$	1,2 (sääntö 6.)
5.	$\neg R$	3,4 (sääntö 8.)

Siis, riippumatta siitä mikä lauseen alussa on vertailun $x > Max$ totuusarvo, lauseen suorituksen jälkeen $x > Max$ on epätosi.

Riittävä ja välttämätön ehto

Todessa muotoa "Jos A niin B" olevassa lauseessa A on *riittävä* ehto B:lle (aina kun A tapahtuu/vallitsee, tapahtuu/vallitsee myös B), mutta B on *välttämätön* ehto A:lle (jos B ei tapahdu/vallitse, niin ei myöskään A tapahdu/vallitse) (Halonen, 2004).

Loogisesti päteviä päätelmiä (Halonen, 2004):

(a) *Modus ponens* - etujäsenen myöntösääntö

Jos A niin B, A Siis: B

b) *Modus tollens* - takajäsenen kieltosääntö

Jos A niin B, ei-B Siis: ei-A

Virhepäätelmiä (Halonen, 2004)

(c) Takajäsenen myöntämisen virhe:

Jos A niin B, B Siis: A (**väärin!**)

Esim. Jos ulkona sataa, niin maa kastuu. Maa kastuu. Siis: Ulkona sataa. (Väärin, voi olla myös kastelulaite)

(d) Etujäsenen kieltämisen virhe

Jos A niin B, ei-A Siis: ei-B (**väärin!**)

Esim. Jos hän rakastaa sinua, niin hän menee kanssasi naimisiin. Hän ei rakasta sinua. Siis: Hän ei mene kanssasi naimisiin. (Väärin, hän voi mennä kanssasi naimisiin myös rahasta).

2.3 Ilmaisujen yksinkertaistaminen

Jos ajattelemme, että meillä on käytössä viisi konnektivia ja joukko niihin liittyviä päättelysääntöjä on em. päättelyiden automatisointi (toteutus tietokoneella) monimutkaista. Mikäli kaava voidaan muuntaa niin, että premisseissä käytetään vain disjunktiota ja negaatiota, automatisointi helpottuu huomattavasti.

Lisäksi asiantuntijajärjestelmien yhteydessä päättely pitää esittää jonkinlaisessa vakiomuodossa, että voimme käsitellä niitä mahdollisimman tehokkaasti. Tässä kohdassa tarkastelemme lauseiden sievennystä sekä niiden normaalimuotojen muodostamista.

Sievennyksellä tarkoitetaan tässä yhteydessä esitystavan selkiyttämistä, ei niinkään matemaattista lauseiden supistamista, joka varmaan ensimmäisenä tulee mieleen.

Sieventämisellä on kolme tarkoitusta:

1. Esittää loogiset lauseet mahdollisimman vähillä erilaisilla konnektiiveilla, esimerkiksi digitaalipiirien erilaisten komponenttien määrän minimoimiseksi.
2. Esittää loogiset lauseet mahdollisimman vähillä konnektiiveilla, esimerkiksi digitaalipiirien komponenttien määrän minimoimiseksi.
3. Esittää looginen lause vakiomuodossa automaattista käsittelyä varten.

Tällainen muokkaus on mahdollista, sillä esim.

$$\begin{aligned} A \to B &\equiv \neg A \vee B \\ &\equiv \neg(A \wedge \neg B) \\ A \leftrightarrow B &\equiv (A \wedge B) \vee (\neg A \wedge \neg B) \\ A \leftrightarrow B &\equiv (A \to B) \wedge (B \to A) \\ &\equiv (\neg A \vee B) \wedge (A \vee B) \end{aligned}$$

Kerrataan vielä pääosin aiemmin esitetyt säännöt konjuktion, disjunktion ja negaation käytöstä:

Laki	Nimi (suomennokset epäselviä)
$A \vee \neg A \equiv T$ $A \wedge \neg A \equiv E$	”Kolmannen vaihtoehdon olemattomuus” ”Kontradiktio”
$A \vee E \equiv A$ $A \wedge T \equiv A$	”yksikkölait”
$A \vee T \equiv T$ $A \wedge E \equiv E$	”hallitsevan lait (domination)”
$A \vee A \equiv A$ $A \wedge A \equiv A$	”neutraalilait (idempotent)”
$\neg(\neg A) \equiv A$	kahden negaation laki

$A \vee B \equiv B \vee A$ $A \wedge B \equiv B \wedge A$	vaihdantalait
$(A \vee B) \vee C \equiv A \vee (B \vee C)$ $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$	assosiatiivisuuslait
$(A \vee B) \wedge (A \vee C) \equiv A \vee (B \wedge C)$ $(A \wedge B) \vee (A \wedge C) \equiv A \wedge (B \vee C)$	osittelulait
$\neg(A \wedge B) \equiv \neg A \vee \neg B$ $\neg(A \vee B) \equiv \neg A \wedge \neg B$	de Morganin lait

Näistä laeista voidaan osa määritellä/todistaa toistensa avulla. Samoin voidaan todistaa lisää lakeja, esimerkiksi *absorbiolait*:

$$\begin{aligned} P \vee (P \wedge Q) &\equiv (P \wedge T) \vee (P \wedge Q) && \text{yksikkölaki} \\ &\equiv P \wedge (T \vee Q) && \text{osittelulaki} \\ &\equiv P \wedge T && \text{hallitsevan laki} \\ &\equiv P && \text{yksikkölaki} \end{aligned}$$

Vastaavasti $P \wedge (P \vee Q) \equiv P$ (todistus harjoitustehtävänä).
Vastaavasti $(P \vee Q) \wedge (\neg P \vee Q) \equiv Q$ (todistus harjoitustehtävänä).

Tavoitteena oli myös saada loogiset lauseet helposti (automaattisesti) käsiteltävään muotoon.

Määritelmä Loogisen ilmauksen sanotaan olevan *disjunktiivisessa normaalimuodossa* (DNM), jos ilmaus on disjunktio ($\vee$), jonka kaikki termit ovat atomilauseiden tai niiden negaatioiden konjunktioita ($\wedge$). Vastaavasti ilmaus on *konjunktiiviessa normaalimuodossa* (KNM), jos ilmaus on konjunktio, jonka kaikki termit ovat atomilauseiden tai niiden negaatioiden disjunktioita.

Loogiset ilmaisut voidaan muuntaa jompaan kumpaan em. normaalimuodoista käyttämällä termien loogisia ekvivalentteja (siis em. sääntöjä) – näitä muotoja voi itse asiassa olla monta.

DNM-muotoinen lause kolmella muuttujalla voi näyttää esimerkiksi tältä:

$$(P \wedge \neg Q \wedge \neg R) \vee (\neg P \wedge Q \wedge R)$$

kun taas KNM-muotoinen lause kolmella muuttujalla voi näyttää esimerkiksi tältä:

$$(P \vee \neg Q \vee \neg R) \wedge (\neg P \vee Q \vee R)$$

Termejä voi olla vaikka kuinka monta (tai vain yksi), kunhan ne ovat oikeassa muodossa. Joten lause $P \wedge \neg Q \wedge \neg R$ on sekä KNM- että DNM-muodossa.

Esim: $\neg((P \vee Q) \wedge \neg R)$ on muunnettava konjunktiiviseen normaalimuotoon.

$$\begin{array}{lcll}
\neg((P \wedge \neg Q) \wedge \neg R) & \equiv & \neg(P \vee \neg Q) \vee \neg\neg R & \text{de Morgan} \\
& \equiv & \neg(P \vee \neg Q) \vee R & \text{kaksois-negaatio} \\
& \equiv & (\neg P \wedge \neg\neg Q) \vee R & \text{de Morgan} \\
& \equiv & (\neg P \wedge Q) \vee R & \text{kaksoisnegaatio} \\
& \equiv & (\neg P \vee R) \wedge (Q \vee R) & \text{osittelulaki}
\end{array}$$

Huomaa, että toiseksi viimeinen rivi on disjunktiivista normaalimuotoa!

Esim: Muunna ilmaus $\neg(P \wedge Q) \rightarrow \neg P \vee \neg(P \wedge \neg Q)$ disjunktiiviseen normaalimuotoon.

$$\begin{array}{lcl}
\neg(P \wedge Q) & \rightarrow & \neg P \vee \neg(P \wedge \neg Q) \\
& \equiv & \neg\neg(P \wedge Q) \vee (\neg P \vee \neg(P \wedge \neg Q)) \\
& \equiv & (P \wedge Q) \vee (\neg P \vee \neg(P \wedge \neg Q) \\
& \equiv & (P \wedge Q) \vee (\neg P \vee \neg P \vee \neg\neg Q) \\
& \equiv & (P \wedge Q) \vee (\neg P \vee \neg P \vee Q) \\
& \equiv & (P \wedge Q) \vee \neg P \vee Q
\end{array}$$

Kuinka muunnos tehdään käytännössä:

1. Muunnetaan implikaatiot ja ekvivalenssit disjunktioksi tai konjunktioksi (riippuen kumpi muoto meillä on tavoitteena).
2. Muunnetaan termit joissa on negaatiota koko termille de Morganin laeilla (tai poistamalla kaksoisnegaatio).
3. Erotellaan mahdollisesti termejä osittelulaeilla.
4. Muunnetaan mahdollisesti konjunktioita disjunktioiksi tai päinvastoin osittelulaeilla.

Termejä, jotka ovat atomilauseiden ja niiden negaatioiden disjunktioita, sanotaan *alkeisdisjunktioksi* (digitaalitekniikassa MAXTERM). Vastaavasti atomilauseiden ja niiden negaatioiden konjunktioita sanotaan *alkeiskonjunktioksi* (MINTERM).

Kun konjunktiivinen normaalimuoto on muodostettu, voidaan lauseketta usein sieventää edelleen. Esim. disjunktio, joka sisältää atomilauseen ja sen negaation, voidaan poistaa tautologiana.

Esim. Sievennä konjunktiivinen normaalimuoto

$$(P \vee Q) \wedge P \wedge (Q \vee R) \wedge (P \vee \neg P \vee \neg R) \wedge (\neg Q \vee R)$$

- termi $(P \vee \neg P \vee R)$ voidaan tautologiana poistaa
- termi $(P \vee Q)$ voidaan poistaa, sillä P on voimassa (absorptiolaki)
- $(Q \vee R) \wedge (\neg Q \vee R) \equiv R$

Näin saadaan lopulta

$$P \wedge R$$

Disjunktiivinen normaalimuoto voidaan tuottaa myös lausekkeelle, jolle tunnetaan vain totuustaulu. Tätä voidaan käyttää hyväksi etsittäessä loogista ilmaisua, joka toteuttaa halutun totuustaulun.

1. Poimi totuustaulusta rivi, jolla lause on tosi.
2. Muodosta kultakin riviltä atomilauseiden konjunktio (alkeiskonjunktio) siten, että atomilause tulee atomilauseena, kun sen totuusarvo rivillä on TOSI ja atomilause tulee negaationa, kun totuusarvo on EPÄTOSI.
3. Muodosta niiden alkeiskonjunktioiden disjunktio.

Esim: Tarkastellaan kolmen atomilauseen, P,Q ja R muodostamaa lausetta $f(P, Q, R)$. Sen totuustaulu on:

P	Q	R	$f(P, Q, R)$
T	T	T	T
T	T	E	E
T	E	T	T
T	E	E	E
E	T	T	E
E	T	E	E
E	E	T	T
E	E	E	E

Alkeiskonjunktiot ovat: $P \wedge Q \wedge R$, $P \wedge \neg Q \wedge R$, $\neg P \wedge \neg Q \wedge R$ ja disjunktiivinen normaalimuoto

$$(P \wedge Q \wedge R) \vee (P \wedge \neg Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R)$$

Tämä sievenee edelleen (yhdistämällä kaksi ensimmäistä termiä) muotoon

$$(P \wedge R) \vee (\neg P \wedge \neg Q \wedge R)$$

Jos luovutaan disjunktiivisesta normaalimuodosta, tämän voi parilla välivaiheella (HT) sieventää edelleen (konjunktiiviseen normaali-)muotoon

$$(P \vee \neg Q) \wedge R$$

(vihje: osittelulailla R:t yhteen, osittelulailla P ”sekaan”).

Huom! Kaikki loogiset ilmaisut voidaan muuntaa em. normaalimuotoihin. Ylläkuvatulla tekniikalla voidaan myös vaikkapa johtaa jonkin konnektiivin ilmaisu disjunktiona ja/tai konjunktiona.

Huom! Kaikki konnektiivit voidaan esittää vain yhden konnektiivin avulla. Tällaisia ovat Shafferin viiva, NAND ($|$) ja Peircen nuoli NOR ($\downarrow$)

Esim: $P \wedge Q \equiv (P \mid Q) \mid (P \mid Q)$

2.4 Karnaugh’n kartat

Normaalimuotojen sieventämiseen voidaan käyttää tehokasta taulukointimenetelmää, Karnaugh’n karttaa. Erityisesti kun muodostamme kaavan totuustaulukosta, niin kaavasta voi tulla hyvin pitkä. Karnaugh’n kartan avulla saamme mahdollisimman sievennetyn DNF-muotoisen kaavan totuustaulusta. Katsomme kartan käyttöä esimerkkien avulla kahdelle, kolmelle ja neljälle termille. Sama tekniikka toimii mielivaltaisen kokoisille taulukoille.

Kartta esittää lauseen totuustaulun hieman eri muodossa kuin mitä tähän asti on käytetty. Kartta rakennetaan kaksiulotteiseksi taulukoksi siten, että vierekkäiset solut vastaavat aina kahta sellaista valuaatiota jotka eroavat toisistaan vain yhden atomilauseen arvon kohdalla. Kahdelle atomilauseelle bitit ovat esim. järjestyksessä: 00, 01, 11, 10, kolmelle esim. 000, 001,

011, 010, 110, 111, 101, 100. Kukin taulukon solu siis vastaa aiemmin käyttämämme totuustaulukon rivin ”lopputulosta” (siis koko lausekkeen totuusarvoa). Esimerkiksi kahden termin

Karnaugh’n kartat AND ($\wedge$) ja OR ($\vee$) operaatioille ovat:

		A	¬A
B	B	1	0
	¬B	0	0

$A \wedge B$

		A	¬A
B	B	1	1
	¬B	1	0

$A \vee B$

Vastaavasti kolmen ja neljän termin Karnaugh’n ”karttapohjat” ovat seuraavia taulukoita:

		AB			
		11	10	00	01
C	1				
	0				

		AB			
		11	10	00	01
CD	11				
	10				
	00				
	01				

Karnaugh’n kartan muodostaminen:

- Muodostetaan totuustaulu lausekkeesta (tai muuten).
- Katetaan kartan kaikki ykköset suorakulmaisilla alueilla:
  - kukin kartan ykkönen kuuluu johonkin alueeseen
  - alueella ei saa olla nollia
  - alueet mahdollisimman suuria suorakulmioita kooltaan 2^k, (eli 1, 2, 4, 8, 16, jne)
  - alue voi ”kiertää” taulukon reunan ympäri (oikeasta reunasta vasemmalle, alhaalta ylös)
  - alueet voivat olla osin päällekkäin
  - pyritään mahdollisimman vähiin alueisiin

Karnaugh’n kartasta muodostetaan sievennetty DNF-muotoinen lause seuraavasti:

- Disjunktiolausekkeeseen termi kutakin aluetta kohti.
- Jokaista alueen muodostamaa termiä kohti on alkeiskonjunktio jossa ovat ne muuttujat joiden arvot ovat samat koko alueella.
- Konjunktion termit tulevat sellaisenaan kun totuustaulussa on 1, ja negaation kanssa jos totuustaulussa on 0.

Esim:

a)

		A	¬A
B	B	1	1
	¬B	0	0

$f(A, B) = (A \wedge B) \vee (\neg A \wedge B)$

Karnaugh'n kartasta tämä saadaan yhdistämällä ylärivin vierekkäiset ykköset. Ko. alueessa B on aina 1, joten kaavaan tulee B. Koko sievennys on siis $f(A, B) = B$.

Tämä toki olisi ollut helppo muutenkin. Huomaamme, että lause on tosi, kun B on tosi riippumatta A:n totuusarvosta. Samoin se on epätosi, kun B on epätosi.

b)

		A	$\neg A$
B	B	1	0
	$\neg B$	1	1

$f(A, B) = (A \wedge B) \vee (A \wedge \neg B) \vee (\neg A \neg B)$

Muodostetaan kaksi 2:n kokoista aluetta. Vasenta saraketta vastaa lauseke A ja alareunaa lauseke $\neg B$. Saamme siis sievennetyn muodon $f(A, B) = A \vee \neg B$.

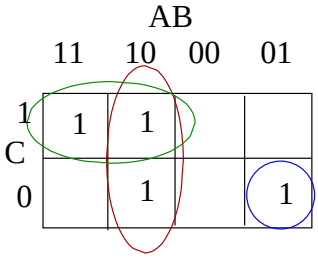
c)

		A	$\neg A$
B	B	1	0
	$\neg B$	0	1

Lauseketta ei voi sieventää, sillä siinä ei ole vierekkäisiä ykkösiä. Lauseke on DNF muodossa $f(A, B) = (A \wedge B) \vee (\neg A \wedge \neg B)$

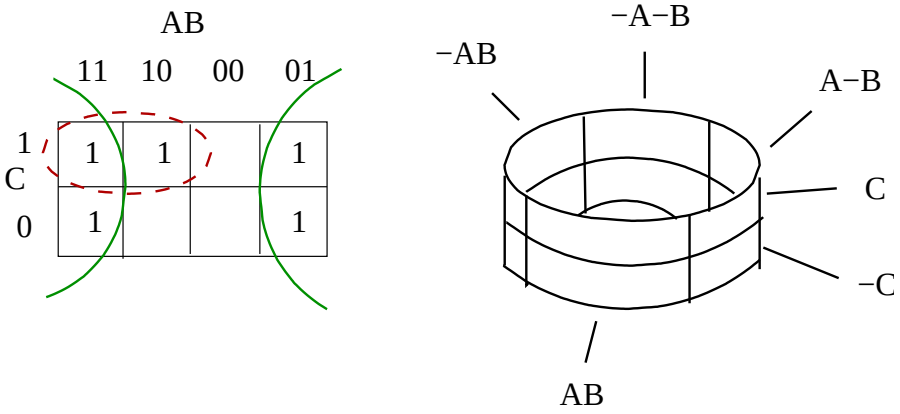
Kolmen termin Karnaugh'n karttoja voidaan sieventää vastaavasti seuraavan esimerkin mukaan.

Esim. a) $f(A, B, C) = (A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg C)$



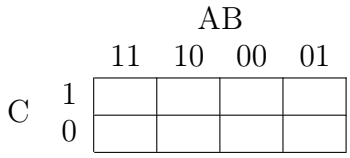
Tästä saadaan sieventämällä $f(A, B, C) = (A \wedge C) \vee (A \wedge \neg B)) \vee (\neg A \wedge B \wedge \neg C)$

b) $f(A, B, C) = (A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \vee (\neg A \wedge B \wedge \neg C)$



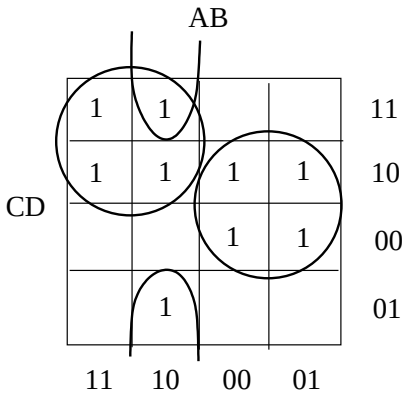
Sievennettäessä löydetään kaksi aluetta, punainen/katkoviiva-alue vasemmassa yläkulmassa jossa $C = 1$ ja $A = 1$ (B vaihtelee), eli $A \wedge C$ ja kun taulukko kierretään päistään renkaaksi sarakkeet 11 ja 01 ovat samaa 4:n kokoista aluetta, saadaan vihreä/yhtenäisen viivan alue jossa $B = 1$ (A ja C vaihtelevat). Sievennetty muoto on siis $f(A, B, C) = (A \wedge C) \vee B$.

c) Sievennetään lause $f(A, B, C) = (A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg C)$



Täydennetään luennolla, tai täydennä itse.

Neljän termin Karnaugh'n karttaa käytetään vastaavalla tavalla. d) $f(A, B, C, D) = (A \wedge B \wedge C \wedge D) \vee (A \wedge \neg B \wedge C \wedge D) \vee (A \wedge B \wedge C \wedge \neg D) \vee (A \wedge \neg B \wedge C \wedge \neg D) \vee (\neg A \neg B \wedge C \wedge \neg D) \vee (\neg A \wedge B \wedge C \wedge \neg D) \vee (\neg A \wedge \neg B \wedge \neg C \wedge \neg D) \vee (\neg A \wedge B \wedge \neg C \wedge \neg D) \vee (A \wedge \neg B \wedge C \wedge D)$



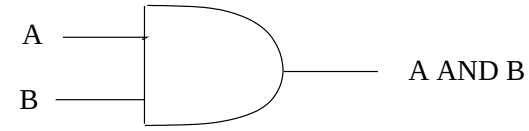
$$f(A, B, C, D) = (A \wedge C) \vee (\neg A \wedge \neg D) \vee (A \wedge \neg B \wedge D)$$

2.5 Konnektiiveja vastaavia logiikkapiirejä

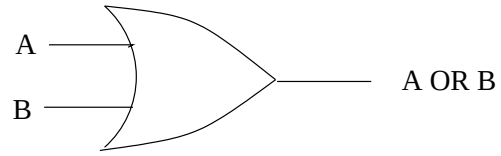
Tarkastellaan seuraavaksi kuinka loogisia operaatioita voidaan toteuttaa elektronisilla logiikkapiireillä. Elektroniikassa käytetyt logiikan peruskomponentit eli digitaalisen logiikan peruspiirit: AND, OR, XOR ja NOT. XOR on eksklusiivinen OR, ts. tai-operaatio, jossa ei hyväksytä sitä vaihtoehtoa, että molemmat propositiot ovat tosia. Voidaan käyttää myös operaatioita NOR (ei kumpikaan) ja NAND (ei molemmat).

Digitaalitekniikassa totuusarvo toteutetaan piiriin tulevan jännitteen avulla. Johtimen jännitearvo +5V vastaa totuusarvoa TOSI ja 0V totuusarvoa EPÄTOSI. Digitaalitekniikassa totuusarvot esitetäänkin arvoina 1 ja 0. Siten peruspiirien totuustaulut ovat seuraavat (Totuustaulujen alla on kutakin loogista operaatiota vastaava digitaalipiirin symboli.)

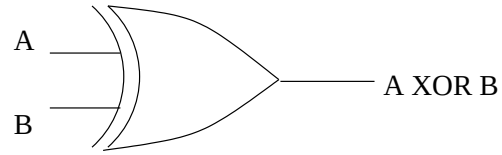
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1



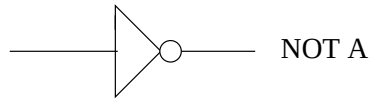
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1



A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

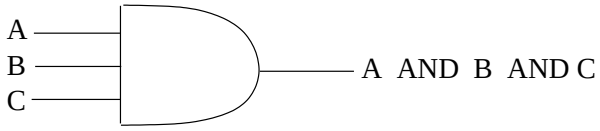


A	NOT A
0	1
1	0

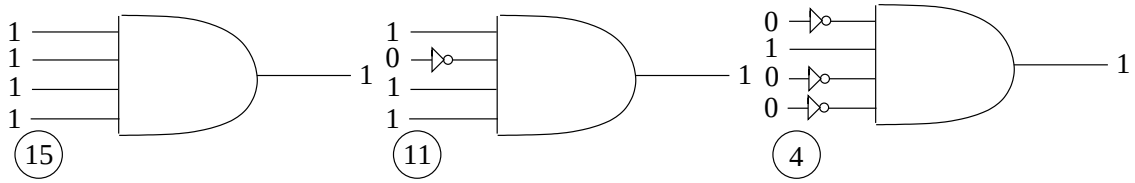


Eräillä piireillä voi olla useampia syötteitä. (AND, OR)

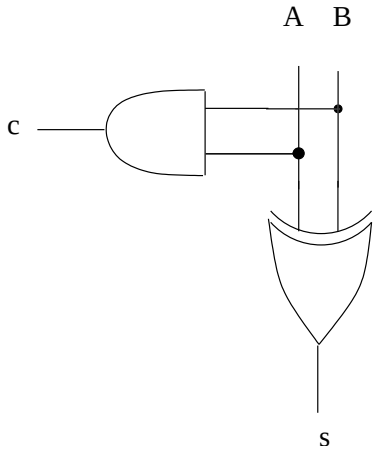
A	B	C	AND(A,B,C)
1	1	1	1
0	X	X	0
X	0	X	0
X	X	0	0



Tällöin AND-piiriä voidaan käyttää tunnistamaan tietty binääriluku.



Tarkastellaan loogista piiriä:



A	B	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Kyseessä on siis kahden bitin yhteenlasku, missä c on “carry”-bitti vrt.

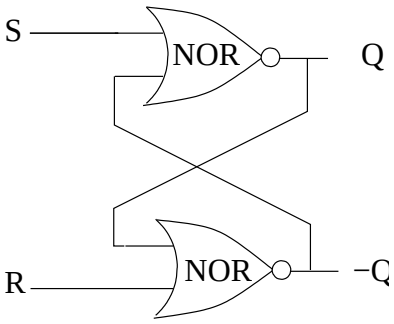
$$\begin{array}{r|r|r} & & 1 \\ 1 & 0 & 1 \\ +\ 0 & +\ 1 & +\ 1 \\ \hline 1 & 1 & 0 \end{array}$$

Siis yhteenlasku on logiikkaa?!

Ns. täydessä yhteenlaskussa (*full adder*) lasketaan yhteen kolme bittiä (kaksi syötettä ja muistinumero (*carry*). Tuloksena kaksi bittiä (tulos ja seuraava muistinumero). Se voidaan tehdä esimerkiksi logiikalla (selitä miksi näin!) :

$S = A \veebar V \veebar C_{in}$ ja $C_{out} = (A \wedge B) \veebar (C_{in} \wedge (A \veebar B))$

Loogisilla piireillä (NAND tai NOR) voidaan toteuttaa myös muistipiiri, ns. Flip-flop.



Q:n ohjaus linjoilla R ja S:

Tila(R,S) = (0,0) : Q säilyttää tilan

Tila(R,S) = (0,1) : Q siirtyy tilaan 0

Tila(R,S) = (1,0) : Q siirtyy tilaan 1

Tila(R,S) = (1,1) : ei sallittu.

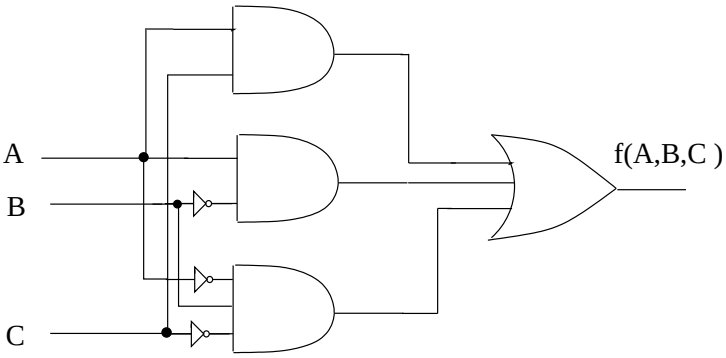
$Q(t)$	S	R	$\neg Q(t+1)$	$Q(t+1)$
0	0	0	1	0
0	0	1	1	0
0	1	0	0	1
0	1	1		
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1		

Karnaugh'n kartat ja digitaalilogiikka

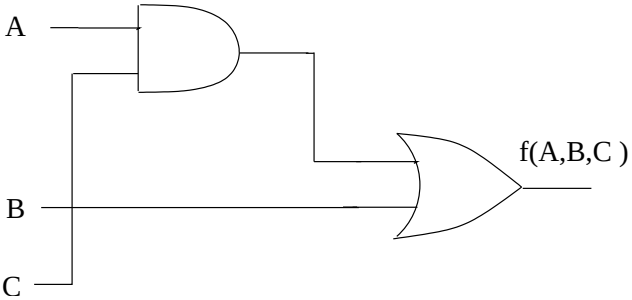
Karnaugh'n karttojen avulla voidaan myös suunnitella digitaalilogiikan piirejä. Kartalla muodostetulla DNF-muotoisella kaavalla voidaan muodostaa halutun tuloksen antava minimaalinen digitaalipiiri.

Esim: Aiempien (s. 23 - 24) kolmen Karnaugh -esimerkin lausekkeet voidaan toteuttaa seuraavilla digitaalipiireillä:

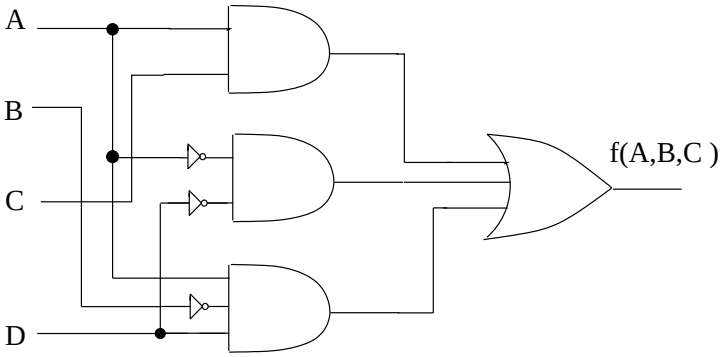
a) (s. 23) $f(A,B,C) = (A \wedge C) \vee (A \wedge \neg B)) \vee (\neg A \wedge B \wedge \neg C)$



b) (s. 23) $f(A,B,C) = (A \wedge C) \vee B$:



d) (s. 24) $f(A, B, C, D) = (A \wedge C) \vee (\neg A \wedge \neg D) \vee (A \wedge \neg B \wedge D)$



2.6 Boolen algebra

Tämä kohta jätetään 2011 luennoilla käsittelemättä, mutta tätä mahdollisesti tarvitaan myöhemmin, tai ainakin tietojenkäsittelijän kuuluu tietää mitä Boolen algebralla tarkoitetaan. Asiassa sinänsä ei ole mitään uutta verrattuna propositiologiikkaan, onpahan vain formalisoitu eri tavalla. Itse asiassa olemme tätä sujuvasti käyttänee koko ajan totuustauluissa.

Tarkastellaan lyhyesti kuinka Boolean algebra liittyy logiikan tarkasteluun.

Boolen algebra (*boolean algebra*) on algebra, jossa käytetään alkioita 0 ja 1 sekä laskutoimituksia $\cdot$ ja $+$. Näille voidaan määritellä algebra, joka toteuttaa seuraavat ehdot.

1. Laskutoimitukselle $+$ on olemassa nolla-alkio, merkitään $0 : X + 0 = X$
2. Laskutoimitukselle $\cdot$ on olemassa ykkösalkio; merkitään $1 : X \cdot 1 = X$
3. operaatio $+$ on kommutatiivinen : $x + y = y + x$
4. operaatio $\cdot$ on kommutatiivinen : $x \cdot y = y \cdot x$
5. $x + (y \cdot z) = (x + y) \cdot (x + z)$
6. $x \cdot (y + z) = x \cdot y + x \cdot z$
7. kaikille alkioille X, on olemassa vasta-alkio $X' : X + X' = 1$
8. kaikille alkioille X, on olemassa käänteisalkio $X^{-1} : X \cdot X^{-1} = 0$
9. alkioita on vähintään kaksi.

Kun tarkastelemme alkioita 0 ja 1, ylläolevat ehdot täyttävät laskutoimitukset $+$ ja $\cdot$ voidaan esittää taulukkona:

$+$	0	1
0	0	1
1	1	1

$\cdot$	0	1
0	0	0
1	0	1

Laskutoimitus $+$ vastaa loogista operaatiota OR ($\vee$) ja laskutoimitus $\cdot$ vastaa operaatiota AND ($\wedge$). Arvot vastaavat totuusarvoja: 0 = E ja 1 = T.

Huom! 0 ja 1 ovat toistensa vasta-alkioita ja käänteisalkioita! Looginen ekvivalenssi voidaan esittää totuustaululla kuten aiemmin.

Esim. Muuta ilmaus

$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$$

Boolen algebran muotoon.

$$P \cdot (Q + R) = (P \cdot Q) + (P \cdot R)$$

Esim. Totea ekvivalenssi totuustauluna.

P	Q	R	P ·	(Q + R)	(P · Q)	+	(P · R)
1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	0
1	0	1	1	1	0	1	1
1	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	1	0	0	1	0	0	0
0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0
			↑	samat		↑	

Esim. Muunna ilmaisut

$$P \vee \neg P \Leftrightarrow T \tag{1}$$

$$P \wedge \neg P \Leftrightarrow E \tag{2}$$

Boolean algebran muotoon.

$$P + \neg P = 1 \tag{1}$$

$$P \cdot (\neg P) = 0 \tag{2}$$

3 Joukko-oppia

Eräs tapa tarkastella ja havainnollistaa logiikkaa on joukko-oppi. Joukko-opissa tarkastelemme erilaisten objektien muodostamia kokoelmia (joukkoja) ja niiden yhdistelmiä ja eroja. Kokoelmaan kuuluvat objektit ovat joukon alkioita.

Esim. Joukkoja ovat :

- luonnollisten lukujen joukko
- punaisten omenien joukko
- totuusarvoltaan TOSI olevien lauseiden joukko
- UEF opiskelijoiden joukko

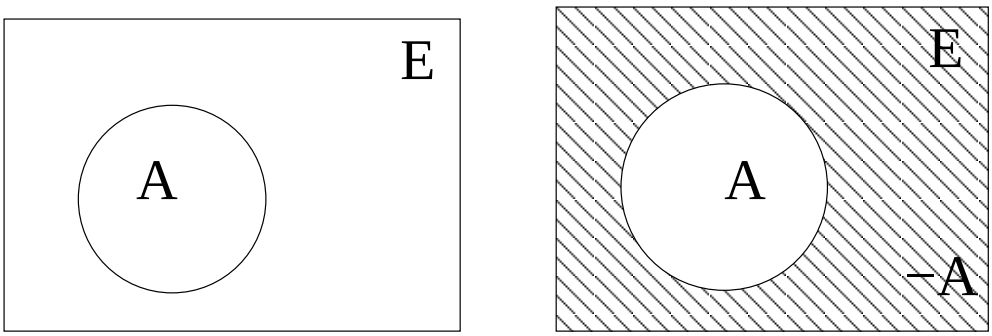
Joukkoja merkitään isoilla kirjamilla ja niiden alkiot luetellaan tai määritellään aaltosulkeissa. Esimerkiksi lukujen 3,4,5 ja 6 muodostamaa joukkoa A merkitään $A = \{3, 4, 5, 6\}$. Alkion kuuluminen joukkoon ilmaistaan $3 \in A$ ja kuulumattomuus $7 \notin A$. Joukon A alkioiden määrää (kardinaliteettia) merkitään $|A|$.

Joukko voi olla myös tyhjä, tätä merkitään $\varnothing$. Joukkoon kuulumattomien alkioiden joukkoa, *joukon A komplementtia*, merkitään $\sim A$ (tai $\overline{A}$). Tällöin oletetaan olevan olemassa jokin perusjoukko, eli universaali joukko E, johon kaikki siinä yhteydessä ajateltavissa olevat alkiot kuuluvat.

Esim: Ajatellaan, että maailmassa on vain punaisia, vihreitä ja kirjavia omenoita. Tällöin tarkasteltaessa "omenoiden joukko-oppia" perusjoukko on

$E = \{\text{kaikki omenat}\}$

Jos joukko $A = \{$ punaiset omenat $\}$, niin $\neg A = \{$ vihreät ja kirjavat omenat $\}$. Tämä voidaan esittää Venn-diagrammilla.



Yleensä perusjoukko on tunnettu ja tällöin sitä ei nimetä erikseen. Joukko voidaan esittää myös formaalisti kuvailemalla alkoiden ominaisuudet, ei luettelemalla alkoita.

Esim: Tarkastellaan omenoiden joukkoa. Silloin punaisten omenoiden joukko voidaan esittää muodossa

$A = \{x \mid x \text{ on punainen } \}$

Esim: Reaalilukujen $\mathbb{R}$ joukossa positiivisten kokonaislukujen joukko on

$$A = \{n \mid n \in \mathbb{Z}, n > 0\}$$

Jos perusjoukko on kokonaislukujen joukko $\mathbb{Z}$, riittää

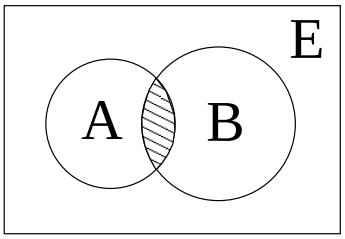
$$A = \{n \mid n > 0\}.$$

Kaksi joukkoa ovat samat silloin ja vain silloin, kun näillä on täsmälleen samat alkiot.

Kahden joukon A ja B leikkaus on joukko, jonka alkiot kuuluvat joukkoon A ja joukkoon B. Leikkausjoukkoa merkitään $A \cap B$.

$$x \in (A \cap B) \equiv (x \in A) \wedge (x \in B)$$

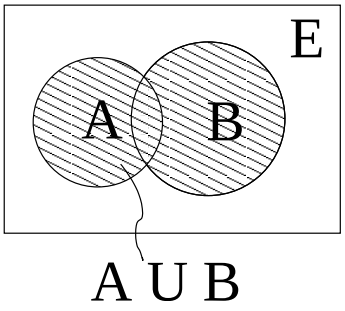
Looginen konjunktio!



Joukkojen A ja B yhdiste eli unioni on niiden alkioiden joukko, jotka kuuluvat joukkoon A tai joukkoon B. Yhdistettä merkitään $A \cup B$.

$$x \in (A \cup B) \equiv (x \in A) \vee (x \in B)$$

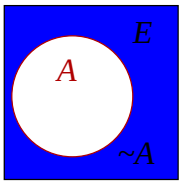
Looginen disjunktio!



Joukon komplementti vastaa negaatiota $\neg A$.

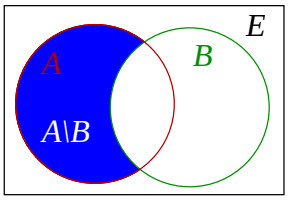
$$x \in {\sim}A \equiv \neg(x \in A)$$

$${\sim}A = \{x \mid x \notin A\}$$

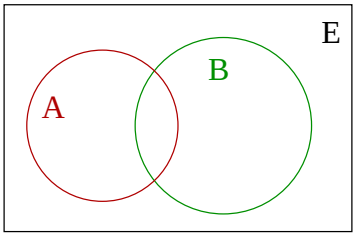


Joukkojen erotus (*difference*), merkitään $A \setminus B$ (tai $A - B$), vastaa käänteistä implikaatiota.

$$x \in (A \setminus B) \equiv (x \in A) \wedge (x \notin B)$$



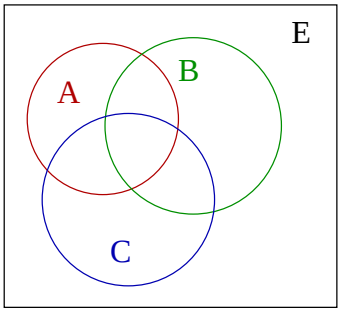
Vastaavasti kaikki muutkin propositiologiikan konnektiivit voidaan tulkita joukko-opiksi ja esittää Venn-kaaviona:

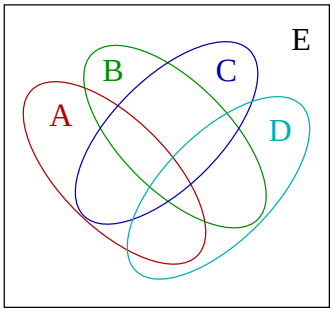


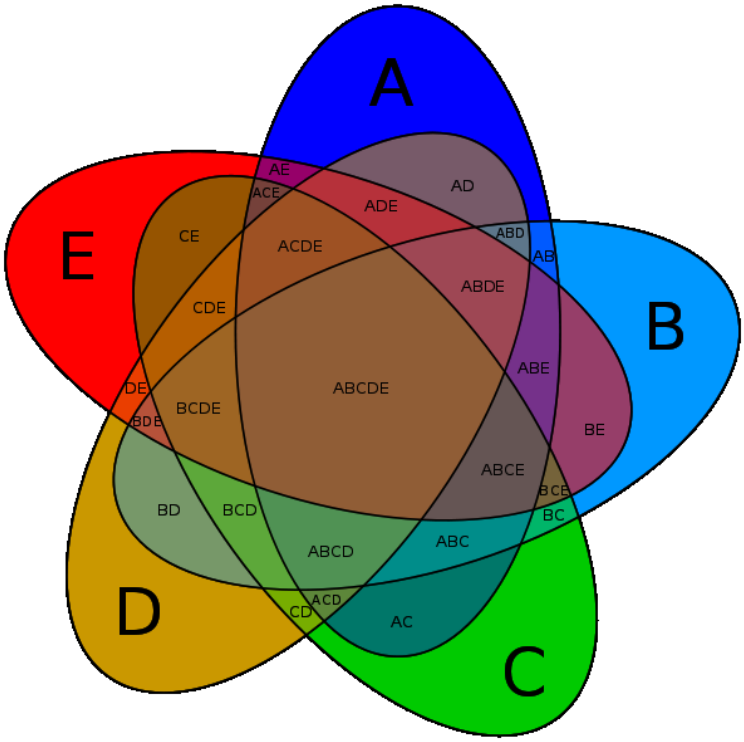
A, B, $\top$, $\bot$, $\veebar$, $\leftrightarrow$, $\rightarrow$, $\not\rightarrow$, NOR, NAND, jne

Piirretään luennolla, tai piirrä itse.

Kuten propositiologiikka, joukko-oppikin toimii mielivaltaisella määrällä joukkoja, tosin piirrettynä kolme (tai neljä) joukkoa on käytännöllinen yläraja.

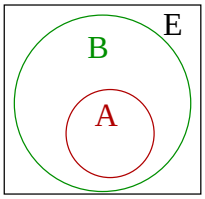






[CC:Grünbaum]

Joukko A on joukon B osajoukko, jos $x \in A \Rightarrow x \in B$. Tätä merkitään $A \subseteq B$. Mikäli lisäksi $A \neq B, C$ on A joukon B aito osajoukko ja sitä merkitään $A \subset B$.



$A = B \Leftrightarrow (A \subseteq B) \wedge (B \subseteq A)$

3.1 Joukko-opin perussäännöt

(1) $A \cup \neg A = E \qquad A \cap \neg A = \varnothing$

(2) $A \cap E = A \qquad A \cup \varnothing = A$

(3) $A \cup E = E \qquad A \cap \varnothing = \varnothing$

(4) $A \cup A = A \qquad A \cap A = A$

(5) $\sim(\sim A) = A$

(6) $A \cup B = B \cup A \qquad A \cap B = B \cap A$

(7) $(A \cup B) \cup C = A \cup (B \cup C) \qquad (A \cap B) \cap C = A \cap (B \cap C)$

(8) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

(9) $\sim (A \cap B) = \sim A \cup \sim B \qquad \sim (A \cup B) = \sim A \cap \sim B$

Nämä säännöt voidaan osoittaa oikeiksi käyttäen loogisten ilmaisujen tunnettuja lakeja. Esim:

$$\begin{aligned} x \in (A \cap B) &\equiv x \in A \wedge x \in B \qquad | \, x \in A = P, x \in B = Q \\ &\equiv P \wedge Q \\ &\equiv Q \wedge P \\ &\equiv x \in B \wedge x \in A \\ &\equiv x \in (B \cap A) \end{aligned}$$

Joukkojen alkioita voidaan joissain tilanteissa tarkastella myös kvanttorien avulla. Merkintä $\forall$ tarkoittaa ”kaikille” ja merkintä $\exists$ tarkoittaa ”on olemassa”, $\nexists$ tarkoittaa ”ei ole olemassa”. Esimerkiksi

$(A \cap B \neq \varnothing) \Leftrightarrow (\exists x$ siten, että $x \in A \wedge x \in B)$.

$(A \subseteq B) \Leftrightarrow (\forall x$ pätee, että $x \in B \rightarrow x \in A)$.

Esimerkkejä verkossa :

http://webspace.ship.edu/deensley/DiscreteMath/flash/

Joukkojen notaatio
Joukko-operaatiot
Vastaesimerkkien luomista joukko-opissa

3.2 Joukkojen karteesinen tulo ja relaatio

Kun A ja B ovat joukkoja, niin järjestetty pari (*pair*, *tuple*) on olio muotoa (a, b), missä $a \in A$ ja $b \in B$. Kaikkien mahdollisten tälläisten parien joukkoa kutsutaan joukkojen A ja B karteesiseksi tuloksi, merkitään $A \times B$. Tulojoukon koko $|A \times B| = |A| \cdot |B|$.

Toisin ilmaisten $A \times B = \{(a, b) \mid (a \in A) \wedge (b \in B)\}$.

Olkoot esimerkiksi $A = \{x, y\}$ ja $B = \{1, 2, 3\}$.

$A \times B = \{(x, 1), (x, 2), (x, 3), (y, 1), (y, 2), (y, 3)\}$.

$B \times A = \{(1, x), (1, y), (2, x), (2, y), (3, x), (3, y)\}$.

$A \times A = \{(x, x), (x, y), (y, x), (y, y)\}$.

Voimme myös piirtää kuvan karteesisesta tulosta $A \times B$:

A		B		
		1	2	3
x		$(x,1)$	$(x,2)$	$(x,3)$
y		$(y,1)$	$(y,2)$	$(y,3)$

Karteesinen tulo joukoille $A_1, A_2, \ldots, A_n$ on joukko $A_1 \times A_2 \times \ldots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in A_i, i = 1, 2, \ldots, n\}$. Tälläisen joukon alkiota sanotaan äärelliseksi listaksi tai n-monikoksi tai n-jonoksi (n-*tuple*). Kun karteesisen tulon molemmat (tai kaikki) osat ovat sama joukko (esim. $A \times A$) puhutaan myös potenssijoukosta (power set), merkitään A^2. Potenssi voi olla suurempikin $A \times A \times A \times A = A^4$. Yleisemmin n-monikko samasta joukosta on A^n.

Esimerkiksi $\mathbb{R} \times \mathbb{R}$, eli $\mathbb{R}^2$ muodostaa (karteesisen) tason.

Esimerkiksi kun $B = \{1, 0\}$, niin B^n on kaikkien n:n mittaisten bittijonojen joukko.

Relaatio Joukkoa n-monikoita sanotaan (n-ariseksi) relaatioksi. Relaatiot (relaatiotietokannoissa) on yksi yleisimmistä tavoista tallettaa tietoa.

Kuvaus Jos kaikille relaation M alkioille (a_1, b_1), (a_2, b_2) pätee $(a_1 \neq a_2)$ tai $(b_1 = b_2)$, kyseessä on kuvaus. Toisin sanoen relaatio on kuvaus jollei siihen voi sisältyä kahta alkiota joiden ensimmäiset alkiot olisivat samat (tai myös toiset jäsenet samat). Tällöin sanotaan, että b on a:n kuva, merkitään $M(a) = b$. Jos kuva on määritelty, se on yksikäsitteinen.

4 Predikaattilogiikka

Propositiologiikalla saatoimme todistaa monenlaisia loogisia johtopäätöksiä, mutta emme läheskään kaikkia. Esimerkiksi seuraava päättely on täysin järkevä, mutta propositiologiikka ei taivu siihen.

1. Kaikilla kissoilla on häntä
2. Tom on kissa.

---

3. Tomilla on häntä.

Predikaattilogiikka lisää keinoihimme tavan kuvata olioiden (esineiden, henkilöiden, eläinten, jne) ominaisuuksia ja/tai suhteita.

Predikaatti $P(x)$ perusjoukossa D on lause joka sisältää muuttujan x siten, että aina kun x korvataan millä tahansa perusjoukon D alkiolla, predikaatin $P(x)$ tulos on johdonmukaisesti tosi tai epätosi.

Luonnollisen kielen kieliopin *predikatiivi*?

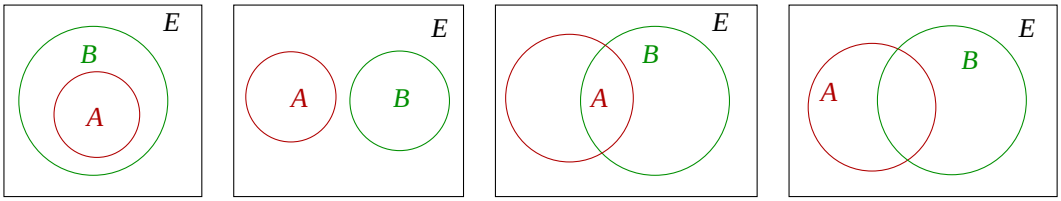
Predikaatilla voi olla usea parametri, esim $P(x, y)$. Tällöin predikaatti P kytkee x:n y:hyn nimenomaan tässä järjestyksessä. Kyseessä on siis relaatio.

Predikaatti $P(x)$ vastaa propositiologiikan atomilausetta. Predikaattilogiikan komponentteja yhdistellään kuten propositiologiikan komponentteja. Lisäksi käytetään *kvanttoreita*: $\forall$ (kaikille, *for all*), $\exists$ (on olemassa, *there exists an*) ja joskus $\nexists$ (ei ole olemassa, *there does not exists an*). "Ei ole olemassa" merkitään kuitenkin käyttämässämme materiaalissa negaatiota käyttäen $\neg\exists \ldots$.

2011 luennot: käytetään materiaalina *Logic in Action* -verkkokirjan lukua 4 "The World according to Predicate Logic" `http://www.logicinaction.org/docs/ch4.pdf` ja sen kalvosarjaa `http://www.logicinaction.org/slides/LiA-chp04-en.pdf`. Linkit löytyvät myös tämän kurssin verkkosivulta. *Logic in Action* käyttää predikaatista merkintää Px eikä $P(x)$.

Lisämateriaalia

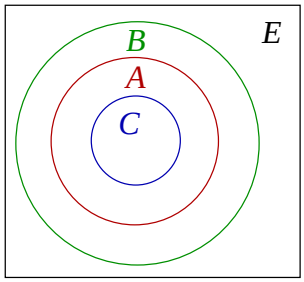
Joukko-opista tuttuja Venn-kaavioita (tai ehkä tarkemmin kaaviot aiemmin esittäneen Leonhard Eulerin mukaan Euler-kaavioita) voidaan käyttää myös predikaattilogiikan havainnollistamiseen:



Kaikki A on B | Mikään A ei ole B | Jokin A on B | Jokin A ei ole B

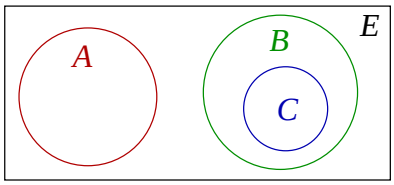
Harjoitus: ilmaise tarkemmin kvanttoreiden avulla.

Samoin kaavioita voi käyttää predikaattilogiikan päättelyn havainnollistamiseen:



Kaikki A ovat B.
Kaikki C ovat A.

Niinpä, kaikki C ovat B.



Mikään A ei ole B.
Kaikki C ovat B.

Niinpä, mikään C ei ole A.

Euler/Venn-kaavioiden puute on siinä, että jos lauseet eivät määrittele kaikkea yksikäsitteisesti, niin kaavioon sitä on vaikea piirtää tekemättä liikaa ”uusia väitteitä”. Erityisesti kvanttori ”jokin” sisältyy kvanttoriin ”kaikki”, mutta tilanteet ovat erilaiset perusmallisessa Euler/Venn-kaaviossa. Tämä oli nähtävissä jo ensimmäisessä kuvassa (s. 35).

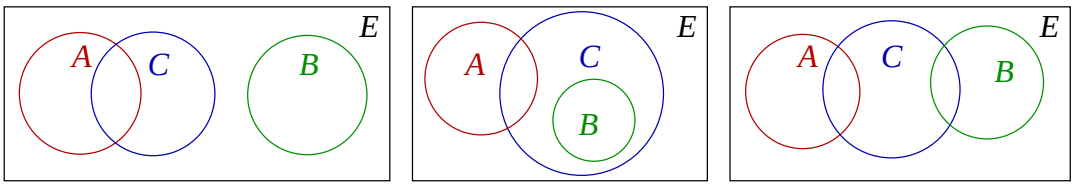
Niinpä näitä kaavioita ei yleensä voi käyttää minkään kaavan todistamiseen, esimerkiksi kahden kaavan ekvivalenttiuden toteamiseen!

Samoin, kaavioilla on vaikea tai mahdoton havainnollistaa relaatiota (kahden tai useamman parametrin predikaattia).

Esimerkiksi seuraavan päättelyn havainnollistamiseen ei yksi kuva oikein riitä vaan kaikki kolme kuvaa sopivat samaan päättelyyn.

Mikään A ei ole B.
Jokin C on A.

Niinpä, jokin C ei ole B.



Joukko-opin termein tämä voidaan selventää seuraavasti: A ja B ovat erilliset ($A \cap B = \varnothing$). A ja C leikkaavat ($A \cap C \neq \varnothing$). Niinpä jokin C:n alkio kuuluu A:han ja siten ei kuulu B:hen. B:n ja C:n keskinäisestä suhteesta ei voida sanoa mitään muuta kuin, että C ei ole B:n alijoukko ($C \not\subseteq B$).

Venn-kaavioista on olemassa täydennettyjä versioita (Peirce, Shin, Venn-II) joilla nämä eri mahdollisuudet voidaan yhdistää samaan kuvaan. Täydennetyt versiot perustuvat erilaisiin merkintöihin osajoukkojen mahdollisesta tyhjyydestä tai useamman kaavion yhdistämiseen. Katso esim. Logic in Action, kohdat 3.4 ja 3.5.

Alla syksyn 2007 DR-kurssin materiaali.

Predikaattilogiikka

Predikaattia sanotaan n-paikkaiseksi predikaatiksi, jos siihen liittyy n argumenttia.

Esim.
$\mathrm{K}(x)$ “x on kissa” on 1-paikkainen
$\mathrm{Ä}(U, H)$ “Ulla on Heikin äiti” on 2-paikkainen

Joskus argumenttien määrä voi olla tilanteen mukaan muuttuva.

Esim:
$\mathrm{SUM2}(x_1, x_2, y)$ “$x_1 + x_2 = y$”
$\mathrm{SUM3}(x_1, x_2, x_3, y)$ “$x_1 + x_2 + x_3 = y$”

Tässä kummallekin summalle on oma predikaatti. Voimme myös ajatella SUMMA-predikaatin sellaiseksi, että siinä kaikki muut luvut ovat yhteenlaskettavia ja viimeinen luku on niiden summa.
$\mathrm{SUMMA}(x_1, \ldots, x_n, y)$

Predikaatti ja siihen liitetty argumenttien lista on *atomilause*. Atomilauseen (predikaatin) argumentit voivat olla muuttujia tai vakioita. Kun argumenttien arvoksi sijoitetaan vakiot, saa atomilause totuusarvon (TOSI tai EPÄTOSI).

Mikäli perusjoukko, josta argumenttien arvot voidaan sijoittaa, on äärellinen, voidaan atomilauseen totuusarvot esittää taulukkona:

Esim: Predikaatti "x on y:n äiti" Ä(x, y)

Ä(x, y)

x\y	Jenni	Heikki	Tuuli	Miina
Ulla	T	T	E	E
Saara	E	E	T	T
Ville	E	E	E	E
Kaisa	E	E	E	E

Esim: Predikaatti "x on suurempi kuin y" SK(x, y)

SK(x, y)

x\y	1	3	10
2	T	E	E
5	T	T	E
100	T	T	T

Esim: Predikaatti "x ja y ovat sisaruksia" SI(x, y)

SI(x, y)

x\y	Heikki	Jenni	Miina
Heikki	?	T	E
Jenni	T	?	E
Miina	E	E	?

Yleisesti, n-paikkaisen predikaatin totuusarvot voidaan esittää n-ulotteisena taulukkona, jos tarkastellaan vakioita äärellisestä joukosta. Jos sijoitettavien vakioiden joukko ei ole äärellinen, on totuusarvo määritettävä kunkin sijoituksen kohdalla erikseen.

Atomilauseella on siis totuusarvo, kun sen argumenteille on sijoitettu vakioarvot. Pelkälle argumenttilistalle muuttujia totuusarvoa ei ole määritelty.

Esim Ä(x, y), SK(x, y), SI(x, y), SUMMA$(x_1, \ldots, x_n, y)$

Sen sijaan päteviä loogisia lausekkeita voidaan määritellä, vaikka atomilausessa on muuttujat.

Esim:

Kissa$(x) \Rightarrow$ OnHäntä(x)

Koira$(x) \wedge$ Ruskea(y)

Arvosana$(x) \Rightarrow (x \geq 0) \wedge (x \leq 3)$

Esimerkkejä

Predikaattiesimerkkejä
Predikaattiesimerkkejä käänteisesti

4.1 Kvanttorit

Muuttujien arvo voidaan valita jostakin arvojen joukosta. Tätä valintaa kuvaamaan otamme käyttään kaksi uutta merkintää, joita kutsutaan kvanttoreiksi.

1. *universaalikvanttori*, $\forall$ (A ylösalaisin) : Väite on tosi kaikille perusjoukon alkioille, ts. "kaikille".
2. *olemassaolokvanttori*, $\exists$, (E väärinpäin) : Väite on tosi joillekin perusjoukon alkioista, ts. "on olemassa".

Katsotaanpa vielä väitettä Kissa$(x) \Rightarrow$ Häntä(x). Jos väite koskee kaikkia kissoja, merkitään

$\forall$ Kissa$(x) \Rightarrow$ Häntä(x).

Jos taas tiedämme, että on olemassa ainakin yksi kissa, jolle väite pätee, mutta emme ole varmoja päteekö väite aina, niin merkitsemme

$\exists$ Kissa$(x) \Rightarrow$ Häntä(x).

Näppärästipä se kävikin. Mutta nyt meillä on taas yksi asia lisää huolehdittavana, nimittäin mikä kulloinkin on perusjoukko. Totuus nimittäin riippuu x:stä: koska x $\in E$, ja jos E on hännällisten kissojen joukko, niin $\forall$ pätee. jos taas E on kaikkien eläinten joukko, niin $\forall$ ei päde.

Universaalikvanttoria ($\forall$) käytettäessä on huomioitava (tiedettävä) mikä on perusjoukko, sillä "$\forall x$" viittaa aina siihen ja totuusarvo riippuu siten perusjoukon valinnasta!

Tämä selviää kun katsotaan kahta melkein samannäköistä lausetta (perusjoukkona kaikki ihmiset):

$\forall x$ Äiti(x, y) $\qquad$ $\forall y$ Äiti(x, y)

Vasemmanpuoleinen tarkoittaa "Kaikki ihmiset ovat jonkun äitejä". Toinen puolestaan "Kaikilla ihmisillä on äiti". Eli sillä, mihin kvanttorin panee, on eroa. Tästä nähdään myös, että kvanttori kohdistuu lausekkeissa tiettyyn muuttujaan. Tällainen muuttuja on *sidottu muuttuja*. Kvanttorien vaikutus kohdistuu vain ennen seuraavaa konnektiivia oleviin atomilauseisiin. Mikäli vaikutus halutaan laajemmaksi, on käytettävä sulkuja.

Esimerkiksi $\forall x$P$(x) \vee$ Q(x) on eri asia kuin $\forall x$(P$(x) \vee$ Q(x)). Tosin tässä tilanteessa kannattaa käyttää jälkimmäisen muuttujan esittämiseen eri kirjainta, vaikkapa y:tä, jolloin siitä näkee helpommin ettei muuttuja olekaan sidottu. Yleensä näin tehdäänkin, mutta on hyvä tietää, että asian voi esittää myös näin.

Katsotaanpa seuraavaa lausetta, ja tutkitaan mitkä muuttujista ovat sidottuja ja mitkä eivät.

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

Melko helposti erottuu, että x ja y ovat sidottuja, ja z on vapaa muutttuja.

Ajatellaanpa seuraavaksi lausetta:

"On olemassa ihminen, joka tuntee kaikki ihmiset"

Jos perusjoukkona on kaikki maailman ihmiset, niin lause ei ole mielekäs, sillä kukapa nyt tuntisi jokaikisen muun ihmisen. Jos taas kyseessä on suppeampi joukko ihmisiä, vaikkapa tietyn työpaikan ihmiset, niin lause voikin olla mielekäs (jos työntekijöitä ole kovin paljon).

Lauseen formalisointiin käytetään nyt predikaattia Tuntee(x, y): "x tuntee y:n".
Ensimmäinen osa on siis "on olemassa ihminen" - $\exists x$. "(x) tuntee kaikki ihmiset on" puolestaan $\forall y$Tuntee(x, y). Joten ne yhdistämällä saamme formaaliksi muodoksi

$\exists x \forall y \text{Tuntee}(x, y)$

Entäpä jos vaihdetaan kvanttorit toisin päin $\forall x \exists y \text{Tuntee}(x, y)$? Sehän tarkoittaa luonnollisella kielellä kutakuinkin

"Jokainen tuntee jonkun" tai
"Kaikki ihmiset tuntevat jonkun ihmisen".

Eli tarkkana on oltava.

Tarkastellaan lausetta "Kukaan ei ole täydellinen." Tämäkin lause sisältää kvanttorin ja ilmaisee ominaisuuden, joka liittyy kaikkiin perusjoukon (ihmiset) alkioihin. Se sisältää myös negaation. Niinpä lause voidaan ajatella kahdella eri tavalla:

$\neg\exists x \text{Täydellinen}(x)$	$\forall \neg \text{Täydellinen}(x)$
"Ei ole olemassa ihmistä, joka on täydellinen"	"Kaikille ihmisille pätee, ihminen ei ole täydellinen"

Mutta lauseiden totuusarvo ovat aivan samat, joten voimme todeta niiden välillä olevan loogisen ekvivalenssin:

$\neg\exists x \text{Täydellinen}(x) \equiv \forall x \neg \text{Täydellinen}(x)$

Mutta tämä pätee vain tälle yhdelle tapaukselle. Yritetäänpä todistaa se yleisesti. Yleisesti kvanttoreita käsitellessä ajattelemme aina jotain perusjoukkoa. Edellisen toteamuksen voimme todistaa yleisemmin näin:

Olkoon perusjoukko $E = \{a_1, a_2, a_3, \ldots, a_n\}$.
Olkoon myös olemassa jokin predikaatti T(x). Ajatellaan, että

$\exists x \text{T}(x) : T(a_1) \vee T(a_2) \vee \ldots \vee T(a_n)$ on TOSI (ainakin yksi arvo on TOSI), sekä
$\forall x \text{T}(x) : T(a_1) \wedge T(a_2) \wedge \ldots \wedge T(a_n)$ on TOSI (kaikki arvot pitää olla TOSI).

Siis toisin sanottuna, tapaus voidaan kirjoittaa

$$\begin{aligned}\neg\exists x\mathrm{T}(x) &\equiv \neg(T(a_1) \vee T(a_2) \vee \ldots \vee T(a_n)) \\ &\equiv \neg T(a_1) \wedge \neg T(a_2) \wedge \ldots \wedge \neg T(a_n) \qquad \text{(de Morgan)} \\ &\equiv \forall x \neg \mathrm{T}(x)\end{aligned}$$

Joten kyllähän se on yleisesti voimassa.

4.2 Looginen päättely predikaattilogiikassa

Predikaattilogiikkaa käytetään tuomaan koneeseen "äly". Esimerkiksi ohjelmointikielet, kuten prolog, pohjautuvat predikaattilogiikkaan. Ihmisten on yleisesti helppo päätellä asioita, mutta koneelle tehtävä on paljon vaikeampi - sen on mentävä sääntöjen mukaan yksi asia kerrallaan.

Seuraavaksi katsomme muutaman esimerkin avulla, kuinka predikaattilogiikassa tehdään loogisia johtopäätöksiä. Päättelyn säännöt ja idea ovat samat kuin propositiologiikassakin. Nyt on vain osattava yhdistää loogiset ilmaisut, jotka koskevat yleisesti muuttujaa ja ilmaisut, joissa on vakiot.

Kvanttoreita käytettäessä totuusarvoja ajatellaan kuten aiemminkin loogisen päättelyn yhteydessä, eli päättelysäännöt ovat yhä päteviä riippumatta lauseiden totuusarvosta. Kun lauseke on premissinä, odotetaan sen olevan tosi. Siten

$\forall x(\text{Kissa}(x) \Rightarrow \text{Häntä}(x))$

tarkoittaa, että implikaatio pätee kaikille x. Siten se pätee erityisesti Pekalle. Näin x:n paikalle voidaan sijoittaa "Pekka" ja lause on tosi.

Esim:

1. $\forall x(\text{Kissa}(x) \Rightarrow \text{Häntä}(x))$ (yleinen sääntö)
2. $\text{Kissa}(Pekka)$ (premissi)
3. $\text{Kissa}(Pekka) \Rightarrow \text{Häntä}(Pekka)$ (1.)
4. $\text{Häntä}(Pekka)$ (2.+3.)

Loogisen lauseen sanotaan olevan *validi*, jos se on tosi kaikille perusjoukon alkioille. Loogisen lauseet $A(x)$ ja $B(x)$ ovat loogisesti ekvivalentit, jos $A(x) \Leftrightarrow B(x)$ on validi kaikille perusjoukon alkioille. Propositiologiikassa, jossa ei ole kvanttoreita, tätä tilannetta vastaa tautologia.

Esim: Tarkastellaan predikaatteja

$L(x)$: x laulaa
$O(x)$: x on onnellinen, sekä lauseketta

a) $\forall x(L(y) \Rightarrow O(x)) \equiv L(y) \Rightarrow \forall x O(x)$

Huomaa, ettei x koske y:tä, eli kun y laulaa, kaikki ovat onnellisia. Jos sidomme myös ensimmäisen muuttujan, saamme

b) $\forall x(L(x) \Rightarrow O(x))$

Siis: Kaikki, jotka laulavat, ovat onnellisia.

Propositiologiikalla todistus olisi hyvin samanlainen: P: "y laulaa"
O: "Kaikki ovat onnellisia"
R: "Kaikki laulavat"
$P \Rightarrow O$ (mutta koskee vain y:tä, *jota ei voi korvata*!

Esim: Todistetaan että Sokrates on kuolevainen.

Kaikki ihmiset ovat kuolevaisia
Sokrates on ihminen
$K(x)$: "x on kuolevainen"
$I(x)$: "x on ihminen"

1. $\forall x(I(x) \Rightarrow K(x))$
2. $I(\text{Sokrates})$
3. $I(\text{Sokrates}) \Rightarrow K(\text{Sokrates})$ (1.)
4. $K(\text{Sokrates})$ (2,3) päättelysääntöjen ja premissien 2,3 avulla

Siis Sokrates on kuolevainen.

Esim: Tarkastellaan esimerkkiä:

Matti on Villen isä.
Ville ei ole Matin tytär.
Jokainen ihminen, jonka isä on matti, on Matin poika tai tytär.

Muodostetaan predikaatit:

$I(x, y)$: "x on y:n isä"
$T(x, y)$: "x on y:n tytär"
$P(x, y)$: "x on y:n poika"

1) $\forall x(I(\text{Matti}, x) \Rightarrow (P(x, \text{Matti}) \wedge T(x, \text{Matti})))$
2) $I(\text{Matti}, \text{Ville})$
3) $\neg T(\text{Ville}, \text{Matti})$
4) $I(\text{Matti}, \text{Ville}) \Rightarrow P(\text{Ville}, \text{Matti}) \wedge T(\text{Ville}, \text{Matti})$ (1)
5) $P(\text{Ville}, \text{Matti}) \wedge T(\text{Ville}, \text{Matti})$ (2,4)
6) $P(\text{Ville}, \text{Matti})$ (3,5)

Siis Ville on Matin poika.

Esim. Prolog-esimerkki:

Prolog-ohjelmointikielessä ongelma kuvataan loogisena päättelynä. Tehtävä esitetään premissien ja tosien lauseiden joukkona, ja kone hoitaa loogisen päättelyn niiden perusteella. Tämä tekee Prologista *logiikkaohjelmointikielen.*

Olkoon annetut premissit:

vanhempi(x,y) ← isä(x,y)
vanhempi(x,y) ← äiti(x,y)
jälkeläinen(y,x) ← vanhempi(x,y)

Logiikkaohjelmoinnissa merkintätapa on vakiintunut tämmöiseksi, eli päättelysäännöt merkitään "väärin päin". Sen lisäksi konjunktiota merkitään pilkulla.

Ongelmaksi muodostuu riittävän tarkkojen päättelysääntöjen määrittäminen. Nyt kone ymmärtäisi jälkeläiseksi vain vanhemman lapset. Mutta laajemmin ajateltuna, lapsenlapsetkin ovat saman ihmisen jälkeläisiä, esimerkiksi "Matti on isoäidin jälkeläinen". Siispä esitämme päättelysäännön toisin:

jälkeläinen ← vanhempi(x,z),jälkeläinen(y,z)
(Esim. jälkeläinen ← vanhempi(x,z),vanhempi(z,w),jälkeläinen(y,w))

Kyseessä on *rekursio*, johon palaamme myöhemmin - eli ei tarvitse huolestua, jos asia ei vielä täysin aukene.

Joka tapauksessa, jos haluamme löytää jälkeläisiä, sijoitamme tiedot (premissit)

isä(Juhani,Mari)
isä(Jukka, Kurt)
äiti(Mari, Jukka)
äiti(Tiina, Kurt)

Ja esitämme kysymyksen jälkeäinen(Jukka, Juhani)?

vanhempi(Juhani,Mari) ← isä(Juhani, Mari)
jälkeläinen(Mari, Juhani) ← vanhempi(Juhani,Mari)
vanhempi(Mari, Jukka) ← äiti(Mari, Jukka)
jälkeläinen(Jukka, Mari) ← vanhempi(Mari, Jukka)
jälkeläinen(Jukka, Juhani) ← vanhempi(Juhani, Mari), Jälkeläinen(Jukka,Mari)

Ja siinähän se onkin!

Esimerkkejä

Kvanttoriesimerkkejä

5 Lukujärjestelmät

Lukujärjestelmä kuvaa sen miten lukuja esitetään luettavassa muodossa. Meille tuttu kymmenjärjestelmä käyttää numeroita 0, 1, 2, 3, 4, 5, 6, 7, 8 ja 9. Kymmenjärjestelmän kantaluku (*base*) on siis 10. Muita tietotekniikassa käytettäviä lukujärjestelmiä ovat binäärijärjestelmä (kantaluku 2), oktaalijärjestelmä (kantaluku 8) ja heksadesimaalijärjestelmä (kantaluku 16).

Koska luvusta voi olla mahdoton nähdä mitä lukujärjestelmää se käyttää, käytetään lukujärjestelmän merkitsemiseen alaindeksiä, esim $6_{10} = 6_{16} = 110_2$.

Jos alaindeksiä ei jossain tilanteessa (esimerkiksi ohjelmakoodissa) voi käyttää, käytetään etuliitettä ”0x” heksadesimaaliluvuille, ”0o” oktaaliluvuille ja ”0b” binääriluvuille.

Mitä vikaa on kymmenjärjestelmässä?

Tietokoneen laskenta, logiikka ja muistit toimivat binäärijärjestelmällä, biteillä. Yhdellä bitillä voidaan esittää kaksi eri lukua (0 ja 1), kahdella bitillä 4 (2^2) (00=4, 01=1, 10=2, 11=3), kolmella bitillä 8 (2^3), neljällä bitillä 16 (2^4), jne.

Kymmenjärjestelmän yhden numeron esittämiseen kolme bittiä ei riitä ja neljännestä menee osa ”hukkaan”. ”Osittaisten bittien” käyttäminen taas on hankalaa (tosin niin itseasiassa tehdään).

Niinpä tietokone käyttää sisäisesti binäärijärjestelmää. Käytettäessä lukujen esittämiseen tulostettuna oktaali (8) tai heksadesimaali (16) -järjestelmiä, kukin 3 tai 4:n bitin ryväs aina vastaa yhtä oktaali/heksadesimaalilukua.

Tietokoneen muistit on nykyään poikkeuksetta jaoteltu kahdeksan bitin tavuihin (jotka on ryhmitelty yleensä neljän tavun (32-bit) sanoihin ja 8 tai 16 -tavun (64/128-bit) kaksois/neloissanoihin. Heksadesimaalilukuja menee sopivasti tasan kaksi yhteen tavuun. Oktaalilukuja sensijaan menee myösvain kaksi ja kaksi bittiä jää käyttämättä. Edes kokonaiseen sanaan ei saa tasalukua oktaalilukuja.

Oktaalilukujen hyöty onkin ihmiselle helpompi hahmotettavuus, ja niitä käytetäänkin yleensä kun ihmisen on tarkoitus esittää jokin bittikuvio. Palataan tähän alempana (s. 46).

Luvun esittäminen

Olkoon lukujärjestelmän kantaluku b. Luku esitetään merkkijonona $a_n a_{n-1} \dots a_1 a_0$ ja lukua vastaava (kymmenjärjestelmän) arvo lasketaan seuraavasti:

$$a_n a_{n-1} \dots a_1 a_0 = \sum_{i=0}^{n} \left(a_i \times b^i \right)$$

Muistetaan, että $x^0 \equiv 1 \; \forall x > 0$.

Esimerkiksi:

$$\begin{aligned} 513_{10} &= 5 \times 10^2 + 1 \times 10^1 + 3 \times 10^0 \\ &= 5 \times 100 + 1 \times 10 + 3 \times 1 \\ &= 500 + 10 + 3 \\ &= 513_{10} \end{aligned}$$

$$
\begin{aligned}
1010_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\
&= 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1 \\
&= 8 + 0 + 2 + 0 \\
&= 10_{10}
\end{aligned}
$$

$$
\begin{aligned}
513_8 &= 5 \times 8^2 + 1 \times 8^1 + 3 \times 8^0 \\
&= 5 \times 64 + 1 \times 8 + 3 \times 1 \\
&= 320 + 8 + 3 \\
&= 331_{10}
\end{aligned}
$$

$$
\begin{aligned}
513_{16} &= 5 \times 16^2 + 1 \times 16^1 + 3 \times 16^0 \\
&= 5 \times 256 + 1 \times 16 + 3 \times 1 \\
&= 1280 + 16 + 3 \\
&= 1299_{10}
\end{aligned}
$$

Numerosymbolit

Binäärijärjestelmässä käytetään vain numeroita 0 ja 1. Oktaalijärjestelmässä käytetään numeroita 0, 1, 2, 3, 4, 5, 6 ja 7.

Heksadesimaalijärjestelmässä tarvitaan enemmän numeroita (symboleja) kuin meille tutut kymmenjärjestelmän $0, \ldots, 9$. Käytössä on lisäksi kuusi kirjainta, eli symbolit ovat 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E ja F. Siis $A_{16} = 10_{10}$, $B_{16} = 11_{10}$, $C_{16} = 12_{10}$, $D_{16} = 13_{10}$, $E_{16} = 14_{10}$ ja $F_{16} = 15_{10}$. Esimerkiksi:

$$
\begin{aligned}
\mathrm{CB3}_{16} &= 12 \times 16^2 + 11 \times 16^1 + 3 \times 16^0 \\
&= 12 \times 256 + 11 \times 16 + 3 \times 1 \\
&= 3072 + 176 + 3 \\
&= 3251_{10}
\end{aligned}
$$

(Kymmenjärjestelmän) luvun muuntaminen haluttuun muotoon

Menetelmä 1 : Vähennetään jäljellä olevasta luvusta mahdollisimman suuria kantaluvun potenssin monikertoja. Kun vähentäminen onnistuu, ko kohdalle tulee se ko. monikerta. Kun vähennetään onnistuneesti $k \times b^i$, niin numero i lopusta alkaen on k, muuten 0. Toistetaan kaikille i alaspäin.

Binääriluvuilla monikerta voi olla vain 1, joten menetelmä on vielä hieman yksinkertaisempi: Vähennetään jäljellä olevasta luvusta mahdollisimman suuria kakkosen potensseja. Kun vähennetään onnistuneesti 2^i, niin numero i lopusta alkaen on 1, muuten 0. Toistetaan kaikille i alaspäin.

Esimerkiksi, muutetaan luku 71_{10} binäärijärjestelmään:

(1) Onnistuuko vähentää $2^7 = 128$, ei.
(2) Onnistuuko vähentää $2^6 = 64$, kyllä, tulee numero **1**,
Jäljelle jää $71 - 64 = 7$.
(3) Onnistuuko vähentää $2^5 = 32$, ei, tulee numero **0**.
(4) Onnistuuko vähentää $2^4 = 16$, ei, tulee numero **0**.
(5) Onnistuuko vähentää $2^3 = 8$, ei, tulee numero **0**.
(6) Onnistuuko vähentää $2^2 = 4$, kyllä, tulee numero **1**,
Jäljelle jää $7 - 4 = 3$.

(7) Onnistuuko vähentää $2^1 = 2$, kyllä, tulee numero **1**,
Jäljelle jää $3 - 2 = 1$.
(8) Onnistuuko vähentää $2^0 = 1$, kyllä, tulee numero **1**,
Jäljelle jää $1 - 1 = 0$.

Luku luetaan alusta loppuun, se siis 1000111_2.

Menetelmä 2 : Jaetaan alkuperäistä lukua kantaluvulla toistuvasti kunnes saadaan nolla ja otetaan jakojäännökset talteen.

Esimerkiksi, muutetaan luku 71_{10} binäärijärjestelmään.

(1) $71/2 = 35$, jakojäännös **1**.
(2) $35/2 = 17$, jakojäännös **1**.
(3) $17/2 = 8$, jakojäännös **1**.
(4) $8/2 = 4$, jakojäännös **0**.
(5) $4/2 = 2$, jakojäännös **0**.
(6) $2/2 = 1$, jakojäännös **0**.
(7) $1/2 = 0$, jakojäännös **1**.

Luku luetaan lopusta alkuun, se siis 1000111_2.

Muistetaan, että luvun alussa olevat nollat eivät ole merkitseviä.

Binääriluvuilla jakojäännös on helppo nähdä (oliko jaettava luku parillinen vai pariton). Suuremmilla kantaluvuilla se pitää erikseen laskea. Käytännössä on laskettava operaatiot

$$a_i = \left\lfloor \frac{\text{luku}_{i-1}}{b} \right\rfloor$$

$$\text{luku}_i = \text{luku}_{i-1} - \left\lfloor \frac{\text{luku}_{i-1}}{b} \right\rfloor \times b$$

Tai, toisin ilmaisten

$$\text{luku}_i = \text{luku}_{i-1} \ \textbf{mod}\ b$$

missä **mod** on jakojäännös (*modulo*).

Esimerkiksi, muutetaan luku 49124_{10} heksadesimaalijärjestelmään:

(1) $49124/16 = 3070$, jakojäännös $4 =$ **4**.
(2) $3070/16 = 191$, jakojäännös $14 =$ **E**.
(3) $191/16 = 11$, jakojäännös $15 =$ **F**.
(4) $11/16 = 0$, jakojäännös $11 =$ **B**.

Siis $49124_{10} = \text{BFE4}_{16}$

Esimerkiksi, muutetaan luku 49124_{10} oktaalijärjestelmään:

(1) $49124/8 = 6140$, jakojäännös **4**.
(2) $6140/8 = 767$, jakojäännös **4**.
(3) $767/8 = 95$, jakojäännös **7**.
(4) $95/8 = 11$, jakojäännös **7**.
(5) $11/8 = 1$, jakojäännös **3**.
(6) $1/8 = 0$, jakojäännös **1**.

Siis $49124_{10} = 137744_8$

Bittikuviot

Oktaaliluvusta on helppo muodostaa suoraan binääriluku kun kukin oktaalinumero vastaa kolmea bittiä:

$137744_8 = 001\ 011\ 111\ 111\ 100\ 100_2$.

Sama onnistuu heksadesimaaliluvustakin, joskin muistettavia numero-bittikuvio -pareja on enemmän kun kukin numero vastaa neljää bittiä:

$BFE4_{16} = 1011\ 1111\ 1110\ 0100_2$.

Vastaavasti muunnos binääriluvusta oktaali- tai heksadesimaaliluvuksi onnistuu ryhmittelemällä bitit lopusta alkaen kolmen tai neljän bitin ryhmiin. Esimerkiksi:

$001\ 110\ 001\ 010\ 100_2 = 16124_8$.

$0001\ 1100\ 0101\ 0100_2 = 1C54_{16}$.

Nämä muutokset sujuvat kivuttomasti ja näppärästi kunhan oppii lukujen 0-7 (tai jopa 0-15) binääriesitykset.

5.1 Logaritmi ja eksponentti

Eksponenttifunktiossa jokin vakio b (kantaluku, *base*) korotetaan muuttuvaan potenssiin n, siis b^n. Jos b on ykköstä suurempi, eksponentti kasvaa nopeasti n:n kasvaessa. Aina kun n kasvaa yhdellä, ekponenttifunktio kasvaa b -kertaiseksi. Esimerkiksi kantaluvulle 2:

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1 024
⋮	⋮

n	2^n
20	1 048 576
30	1 073 741 824
40	1 099 511 627 776
50	1 125 899 906 842 624
60	1 152 921 504 606 846 976
⋮	⋮

Logaritmi

Eksponenttifunktion käänteisfunktio on logaritmi. Kun eksponentin kantaluku on b, sitä vastaa b-kantainen logaritmi. Olkoot $b > 1, n > 1$, niin

$b^n = x \Leftrightarrow \log_b x = n$

$\log_b (b^n) = n$ ja $b^{\log_b n} = n$.

Matematiikassa logaritmin kantalukuna käytetään yleensä ns. Neperin lukua, merkitään e. $e = 2,71828\ldots$. Vastaavaa logaritmia kutsutaan luonnolliseksi logaritmiksi ja merkitään yleensä $\ln \equiv \log_e$.

Teknisissä sovelluksissa logaritmin kantalukuna käytetään yleensä lukua kymmenen (10). Merkitään $\log_{10}$, joskus myös log.

Tietojenkäsittelytieteessä logaritmin kantalukuna käytetään yleensä lukua kaksi (2). Merkitään $\log_2$, yleensä vain log, kun kantaluvusta ei ole epäselvää.

Logaritmin laskusääntöjä

$\log_b x = \frac{\log x}{\log b}$.

Huom: osamäärässä voidaan käyttää mitä tahansa logaritmia!

Niinpä esimerkiksi

$\log_2 x = \frac{\ln x}{\ln 2} = \frac{\log_{10} x}{\log_{10} 2}$.

Koska yo kaavoissa $\ln 2$ ja $\log_{10} 2$ ovat vakioita, niin ne voidaan laskea valmiiksi:

$\ln 2 \approx 0.69315$, $\frac{1}{\ln 2} \approx 1.442695$. Niinpä $\log_2 x \approx 1.4 \times \ln x$.

$\log_{10} 2 \approx 0.30103$, $\frac{1}{\log_{10} 2} \approx 3.3219$. Niinpä $\log_2 x \approx 3.3 \times \log_{10} x$.

Eri logaritmit siis eroavat toisistaan vain vakiokertoimella!

Toinen laskusääntö:

$\log (x \times y) = \log x + \log y$

Ja sen johdannaisia:

$\log_b (b \times x) = \log_b x + 1$

$\log_2 (2 \times x) = \log_2 x + 1$

$\log (x^2) = 2 \times \log x$

$\log(x^k) = k \times \log x$

Merkintöjä:

$\log^2 x = (\log x)^2$.

$\log\log x = \log(\log x)$.

Logaritmin laskusääntöjä tietojenkäsittelijälle

Tarkastellaan aluksi kymmenkantaista logaritmia:

n	10^n	$\log_{10} 10^n$
0	1	0
1	10	1
2	100	2
3	1000	3
4	10000	4
⋮	⋮	⋮
10	10000000000	10
⋮	⋮	⋮

Kymmenkantainen logaritmi siis kertoo luvun nollien määrän! Tarkkaanottaen luvun x numeroiden määrä 10-järjestelmässä esitettynä on $\lfloor \log_{10} x + 1 \rfloor$.

Tarkastellaan seuraavaksi kaksikantaista logaritmia:

n	$(2^n)_{10}$	$(2^n)_2$	$\log_2 2^n$
0	1	1	0
1	2	10	1
2	4	100	2
3	8	1000	3
4	16	10000	4
⋮	⋮	⋮	⋮
10	1024	10000000000	10
20	1048576	10000000000000000000	20
⋮	⋮	⋮	⋮

Kaksikantainen logaritmi siis kertoo luvun binääriesityksen nollien määrän! Tarkkaanottaen luvun x numeroiden määrä 2-järjestelmässä esitettynä on $\lfloor \log_2 x + 1 \rfloor$.

Kaksikantaisen logaritmin laskeminen 10-järjestelmän luvusta

Käytetään tästä lähtien kaksikantaista logaritmia jollei muuta mainita.

Kun muistetaan, että $\log 1000 \approx 10$ ja $\log(x \times y) = \log x + \log y$, voidaan approksimoida helposti minkä tahansa luvun kaksikantainen logaritmi:

$$
\begin{array}{rl}
 & \log(12\,345\,678\,901\,234) \\
\approx & \log(12 \times 1000 \times 1000 \times 1000 \times 1000) \\
= & \log 12 + \log 1000 + \log 1000 + \log 1000 + \log 1000 \\
\approx & \log 12 + 40 \\
\approx & 3.5 + 40 \\
= & 43.5
\end{array}
$$

muista: $\log 8 = 3$, $\log 16 = 4$

Tarkempi vastaus olisi $\approx 43.489071407441488701 84\ldots$, mutta tietojenkäsittelijälle pyöristys (suunnilleen) lähimpään tai ylempään riittää. Riittää vain osata litania 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 ja katsoa mihin väliin haettavan luvun tuhansien potenssista yli jäävä osa sijoittuu. Katso esimerkiksi sivun 46 taulukko.

Vielä yksi tulkinta 2-kantaiselle logaritmille

2-kantainen logaritmi kertoo montako kertaa luvun voi puolittaa ennenkuin päästään nollaan.

Esimerkiksi puolitushaku puolittaa syötteen kahtia jokaisella iteraatiolla. Niinpä iteraatioita tulee enintään $\lfloor \log n \rfloor + 1$ kappaletta jos n on syötteen alkioiden määrä.

6 Funktiot ja rekursio

Materiaali perustuu osin Maija Marttila-Kontion Kuopion kampuksen materiaalin, osin kalvosarjaan kirjasta Discrete Mathematics: Mathematical Reasoning and Proof with Puzzles, Patterns and Games, by Doug Ensley and Winston Crawley. Osin Jussi Parkkisen luentoihin.

Palautetaan mieleen relaation ja kuvauksen määritelmät:

- Karteesinen tulojoukko $A \times B$ muodostaa uuden perusjoukon.
- Relaatio $R \subseteq A \times B$ on joukko järjestettyjä pareja (monikoita) joissa parin ensimmäinen osa kuuluu A:han ja toinen osa B:hen.
  - Joukko A on relaation lähtöjoukko
  - Joukko B on relaation maalijoukko
- Relaation R määrittelyjoukko $M(R)$ on se A:n osajoukko jolle relaatio on määritelty $M(R) = \{x \in A \mid \exists y\ (x, y) \in R\}$.
- Relaation R arvojoukko $A(R)$ on se B:n osajoukko joihin relaatio on määritelty $A(R) = \{y \in B \mid \exists x\ (x, y) \in R\}$.
- Relaation käänteisrelaatio on relaatio $R^{-1} = \{(y, x) \mid \forall (x, y) \in R\}$.

Esimerkki Olkoon M miesjoukko ja N jokin naisjoukko sekä relaatio $R \subseteq M \times N$ määritelty siten, että $xRy \Leftrightarrow x$ on y:n aviomies. Moniavioisuus tai sen kieltäminen ei vaikuta tässä esimerkissä.

Nyt relaation R:

- lähtöjoukko on kaikki M:n miehet,
- maalijoukko on kaikki N:n naiset,
- määrittelyjoukko on kaikki M:n aviossa olevat miehet (aviomiehet),
- arvojoukko on kaikki N:n aviossa olevat naiset (aviovaimot),
- käänteisrelaatio R^{-1} on relaatio $yRx \Leftrightarrow y$ on x:n aviovaimo.

- Relaatio on kuvaus eli funktio, jos jokainen lähtöjoukon alkio $a \in A$ kuvautuu enintään yhdelle maalijoukon B alkiolle. Toisin sanoen $M \subseteq A \times B$ on kuvaus jos $(a, b) \in M, (a, c) \in M \Rightarrow b = c$.
- Usein edellytetään, että jokainen $a \in A$ kuvautuu täsmälleen yhdelle joukon B alkiolle. Toisin sanoen $M \subseteq A \times B$ on funktio jos $\forall x \in A \, \exists y \, (x, y) \in M$.
  Merkitään $f : A \to B$, $y = f(x)$.
  Tällöin määrittelyjoukko on sama kuin lähtöjoukko.
- Jos vaatimusta funktion määrittelystä jokaiselle lähtöjoukon A alkiolle lievennetään (t.s. voi olla olemassa lähtöjoukon alkioita x joille $f(x)$ ei ole määritelty lainkaan), tarvitaan erikseen määrittelyjoukko (*domain*)
  $D(f) = \{x \in A \mid \exists y \in B \; y = f(x)\}$.
  Tällöin määrittelyjoukko on lähtöjoukon osajoukko.

Relaatioita ja funktioita voidaan kuvata ns. nuolikaaviolla. Esimerkkejä. [Ensley 4-1, s. 3-]

Relaatioita $R \subseteq A \times A$ voidaan kuvata myös verkolla, eli graafilla, joihin palataan tällä kurssilla myöhemmin ja TRAII kurssilla.

Reaaliarvoisia funktioita ja relaatioita voidaan kuvata myös tutulla x, y koordinaatistolla.

Injektio, surjektio ja bijektio

- Funktio $f : A \to B$ on injektio jos kullekin maalijoukon alkiolle kuvautuu vain yksi määrittelyjoukon alkio, ts
  $f(x_1) = f(x_2) \Rightarrow x_1 = x_2$, eli $x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$.
- Funktio $f : A \to B$ on surjektio jos kullekin maalijoukon alkiolle kuvautuu vähintään yksi määrittelyjoukon alkio, eli
  $\forall y \in B \; \exists x \in A \; y = f(x)$.
- Funktio $f : A \to B$ on bijektio jos se on injektio ja surjektio.

Yhdistetty funktio

Kun yhden funktion maalijoukko on sama kuin (tai osajoukko) toisen funktion määrittelyjoukko (siis $f : A \to B$ ja $g : B \to C$, voimme yhdistää nämä funktiot siten, funktion f tulosta käytetään suoraan funktion g parametrina, siis $y = g(f(x))$. Yhdistettyä funktiota merkitään $g \circ f : A \to C$. Esim. $y = g \circ f(x)$.

Vastaavasti yhdistetty relaatio.

Esimerkkejä. [Ensley 4-2]

Järjestys- ja ekvivalenssirelaatio

Käytetään merkintätapaa $(x, y) \in R \Leftrightarrow xRy$.

Olkoot R 2-relaatio joukossa A, siis $R \subset A \times A$). R on osittainen järjestys (*partial order*) jos seuraavat ehdot pätevät.

1. $xRx\ \forall x \in A$ — reflektiivisyys
2. xRy ja $yRx\ \Rightarrow\ x = y$ — antisymmetrisyys
3. xRy ja $yRz\ \Rightarrow\ xRz$ — transitiivisyys

Osittaista järjestystä voidaan kuvata suunnatulla syklittömällä verkolla. Verkkoihin palataan kurssin lopussa, mutta tässä pieni esimerkki. Joskus käytetään myös ns. Hassen kaaviota joka on kuten verkko, mutta nuolenpäiden sijaan käytetään pystysuuntaista sijoittelua siten, että jos xRy, niin kaaviossa x on y:n alapuolella ja näiden välillä on viiva.

Lisää variaatiota järjestysrelaatiosta:

- Jos $\neg\exists x\ xRx$, järjestystä sanotaan aidoksi (?) järjestykseksi (*strict order*).
- Jos lisäksi $\forall x \forall y\ xRy \vee yRx$, järjestystä sanotaan täydelliseksi järjestykseksi (*total order*).

Esim. reaalilukujen vertailu $<$ antaa täydellisen järjestyksen.

6.1 Rekursio

Hivenen aivonystyröiden hierontaa: mikä on seuraava numero alla mainituissa numerojonoissa?

a) 1,3,5,7, ...

b) 2,4,8,16,32,...

c) 1,2,6,24,120, ...

Ongelman ratkaisuun liittyy lukujonojen mallin (*pattern*) löytäminen. Mallin voi muodostaa tässä tapauksessa kolmella eri tavalla (mutta ei välttämättä kaikilla):

1. Jokainen luku muodostetaan aritmeettisella operaatiolla edellisiin lukuihin nähden.
2. Jokainen luku voidaan muodostaa tietämällä sen paikka lukujonossa.
3. Kertomalla sanallisesti miten malli muodostuu (esim. *lukujono muodostetaan luettelemalla parittomat kokonaisluvut.*)

Kun puhumme rekusiota, huomio kohdistuu ensimmäiseen tapaan.

Määritelmä *Rekursiivinen kaava*, joka määrittää lukujonon, on kaava missä jokainen luku on kuvattu edellisiin termeihin (tai edelliseen termiin) nähden. Jotta rekursiivista kaavaa voidaan käyttää, on jonon kuvaukseen liityttävä riittävästi tietoa miten ensimmäiset luvut muodostuvat.

Notaatio Lukujonojen luettelussa käytämme pienellä kirjoitettuja kirjaimia (a,b,x,y,jne) ja alaindeksiä ilmaisemaan luvun paikkaa jonossa. Näiden ehtojen mukaisesti e_i merkitsee i:nnettä lukua jonossa e.

Tarkastellaan seuraavaksi lukujonoa 1,3,5,7,.... Näyttää siis, että lukujono koostuu positiivisista parittomista kokonaisluvuista, alkaen ykkösestä. Tarkkasilmäinen voi huomata, että edelliseen lukuun lisätään aina 2 saadaksemme seuraavan luvun. Näistä tiedoista voimme johtaa lukujonon rekursiivinen kaavan:

$$a_1 = 1 \text{ sekä } a_i = a_{i-1} + 2, \forall i > 1.$$

Jotkut funktiot ovat helpompia määritellä rekursiivisti kuin mitenkään muuten. Esimerkiksi funktio $F : \mathbb{N} \to \mathbb{N}$, jonka peräkkäiset arvot muodostavat Fibonaccin luvut, määritellään rekursiivisesti näin:

$$\begin{cases} F(0) = 0; \\ F(1) = 1; \\ F(i+1) = F(i) + F(i-1), \forall i \in \mathbb{N} \end{cases}$$

Laskeaksesi uuden Fibonaccin luvun, laske yhteen kaksi edellistä:

$$\begin{aligned} F(2) &= F(1) + F(0) = 1 + 0 = 1; \\ F(3) &= F(2) + F(1) = 1 + 1 = 2; \\ F(4) &= F(3) + F(2) = 2 + 1 = 3; \\ F(5) &= F(4) + F(3) = 3 + 2 = 5; \end{aligned}$$

ja niin edelleen. Muutama ensimmäinen Fibonaccin luku on helppo laskea, ne ovat 0, 1, 1, 2, 3, 5, 8, 13, ...

Rekursio ohjelmointimielessä

Rekursiota käytetään myös itse ohjelmoinnissa, jolloin sillä tarkoitetaan tilannetta, jossa metodi (tai proseduuri) kutsuu itseään toistuvasti tietyn ehdon ollessa voimassa, joko suoraan tai jonkun toisen metodin välityksellä. Rekursio on yleisen ongelmanratkaisuperiaatteen erikoistapaus (engl. recursion). *Reduktiivisessa* eli osittavassa ongelmanratkaisussa ongelma ratkaistaan jakamalla se osiin, ratkaisemalla osaongelmat ja yhdistelemällä osaongelmien ratkaisut koko ongelman ratkaisuksi. Tätä kutsutaan myös nimellä hajoita-ja-hallitse (*divide-and-conquer*). Reduktiota sanotaan rekursioksi, jos ainakin yksi osaongelma on alkuperäisen ongelman kaltainen.

Monien ongelmien ratkaiseminen rekursiivisesti on yleensä selvästi helpompaa kuin iteratiivisellä (ei-rekursiivisellä) algoritmilla. Ohjelmasta tulee näin selvästi selkeämpi ja huomattavasti lyhyempi. Kuitenkin rekursiiviset ratkaisut ovat aika- ja tilavaatimuksiltaan iteratiivisia ratkaisuja raskaampia **jos** rekursiolla (vahingossa) aiheutetaan monikertaista turhaa työtä. Esimerkkinä tälläisesta väärästä rekursion käytöstä on pseudokoodi Fibonaccin lukujen laskemiselle rekursiivisesti suoraan määritelmän mukaan. Ohessa myös iteratiivinen tehokas ratkaisu.

```
Function FIBONACCI_TEHOTON (n:integer) RETURNS fibonacci : integer;
begin
    if n < 3 then return n - 1;
    else  return (FIBONACCI(n-1) +
                  FIBONACCI(n-2));
    end if
end
```

```
Function FIBONACCI_IT (n:integer) RETURNS fibonacci : integer;
begin
    if n < 3 then return n - 1;
    else
         i := 3;
         first := 0:
         second := 1;
         while i<=n do
            temp := second;
            second := second + first;
            first := temp;
            i := i + 1;
         end while
         return second;
    end if
end
```

Rekursiotyypit

Rekursiosta on olemassa erilaisia variaatioita, epäsuora, suora, lineaarinen ja häntärekursio. Nämä määreet ovat osin toisistaan riippumattomia. Voidaan siis käyttää esim. suoraa lineaarista rekusriosta.

Epäsuora rekursio

- Rekursio tapahtuu jonkin toisen aliohjelman kautta.
- Vaikka mikään ongelman osista ei olisikaan alkuperäisen ongelman kaltainen, voi ositusta jatkettaessa osoittautua, että jokin osaongemista sisältää edelleen alkuperäisen ongelman kaltaisen osaongelman.

Esimerkki. Moduuli M1 ei suoraan kutsu itseään, vaan se voi kutsua moduulia M2, joka kutsuu edelleen moduulia M3, jne.. kunnes lopulta moduuli Mn kutsuu moduulia M1. Epäsuora rekursio ei ole luonnollisestikaan hyväksi algoritmin selkeydelle, eikä sitä sen vuoksi tulisikaan käyttää ilman erityisen hyvää syytä.

Suora rekursio

- Aliohjelma kutsuu itse itseään.

Esimerkki. Kertoman laskeminen rekursiivisesti.

```
Function kertoma (n:integer) RETURNS kertoma : integer;
begin
       if n = 0 then
          return 1
       else
          return (n * kertoma(n-1))
end
```

Suoran rekursion kutsu (kutsu itseensä) on viimeisellä rivillä eli n * kertoma$(n-1)$ Kun kertomaa kutsutaan arvolla n, suoritetaan kutsu uudelleen arvolla $n-1$, jonka jälkeen edetään eteenpäin eli suoritetaan kutsu arvolla $n-2$, jne.. Jatketaan näin kunnes n saa arvon 0, jolloin ehtolauseessa suoritetaankin toinen haara ja saadaan palautusarvo 1. Kutsut ovat muodostaneet jonon, joka alkaa purkautumaan eli kun ohjelmaa kutsuttiin arvolla 0, sai n arvon 1. (kutsu on siis muotoa: kertoma(0)). Tämä jatkuu edelleen seuraavana kutsuna eli kertoma(1), jolloin n:n arvoksi saadaan 1. Tästä edelleen eteenpän kutsulla kertoma(2), vastauksena n:n arvo 2, jne... kunnes päädytään kertomakutsuun, joka oli laskennan kohteena.

Suorituksen kulku esimerkin avulla:

kertoma(4) = 4 * kertoma(3)
kertoma(3) = 3 * kertoma(2)
kertoma(2) = 2 * kertoma(1)
kertoma(1) = 1 * kertoma(0)
kertoma(0) = 1
kertoma(1) = 1 * 1 = 1
kertoma(2) = 2 * 1 = 2
kertoma(3) = 3 * 2 = 6
kertoma(4) = 4 * 6 = 24

Tulokseksi saadaan, että neljän kertoma (4!) on 24.

Lineaarinen rekursio

Lineaarisessa rekursiossa aliohjelma kutsuu itseään vain kerran. Yleensä lineaarinen rekursio on helppo korvata toistolla.

Ei-lineaarinen rekursio

Aliohjelma kutsuu itseään useasti (esim. kahdesti tai $0..k$ kertaa). Rekursio ylläpitää eri ”haaroja” suorituksessa. Tällaisen rekursion korvaaminen suoraan toistolla ei ole helppoa, yleensä tarvitaan apurakenteeksi keskeneräisten töiden pino tai lista. Kts. alla binääripuun läpikäynti.

Häntärekursio

- Lineaarisesti rekursiivisen aliohjelman rekursiivinen kutsu on samalla viimeisin aliohjelmassa suoritettava lause.
- Häntärekursio on oikeastaan vain piilotettu toistorakenne.
- Häntärekursio rasittaa aivan turhaan suoritusaikaista muistitilaa, ja näin sen käyttöä tulisikin välttää!

Esimerkki. Katso edellä olevaa kertoma-esimerkkiä. Siinä rekursiokutsu on viimeinen suoritettava lause ohjelmassa eli kyse on myös häntärekursiosta.

Binääripuun läpikäynti rekursiivisesti

Binääripuu on yksinkertainen, syklitön ja yhtenäinen suunnattu verkko, jonka jokaisen solmun tuloaste on korkeintaan yksi ja lähtöaste korkeintaan kaksi. Jokaiseen binääripuun solmuun johtaa ainoastaan yksi polku. Binääripuun *juurisolmu* on solmu, jonka tuloaste on nolla. Binääripuun solmua sanotaan *lehtisolmuksi*, mikäli sen lähtöaste on nolla. Jokainen binääripuun solmu voidaan ajatella omaksi puukseen.

Binääripuun kaikki solmut voidaan käsitellä kolmella eri tavalla: esi-, sisä-, ja jälkijärjestyksessä. Nämä on helpoin määritellä rekursiivisesti, käyttäen hyväksi tietoa että jokainen solmu on muodostaa oman binääripuunsa.

Esijärjestys

(i) Käsittele juurisolmu.

(ii) Käsittele vasen alipuu esijärjestyksessä.

(iii) Käsittele oikea alipuu esijärjestyksessä.

Sisäjärjestys

(i) Käsittele vasen alipuu sisäjärjestyksessä.

(ii) Käsittele juurisolmu.

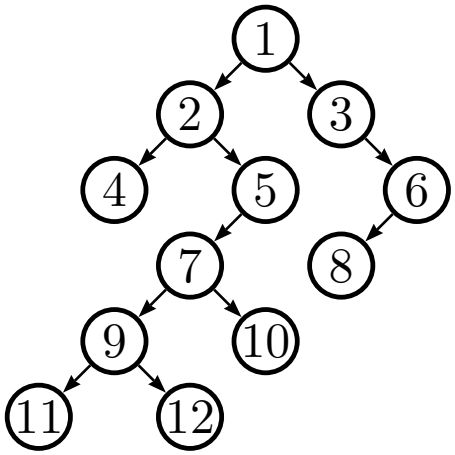
(iii) Käsittele oikea alipuu sisäjärjestyksessä.

Jälkijärjestys

(i) Käsittele vasen alipuu jälkijärjestyksessä.

(ii) Käsittele oikea alipuu jälkijärjestyksessä.

(iii) Käsittele juurisolmu.

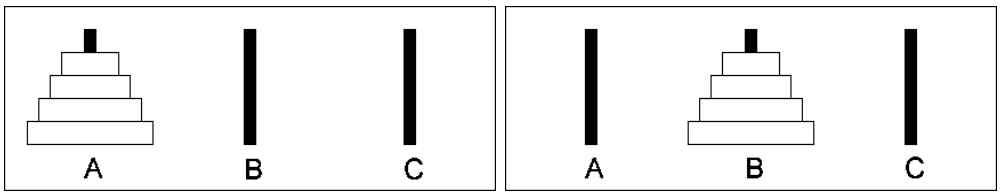


Jos kuvan binääripuu käsiteltäisiin esijärjestyksessä, aloittaisimme seuraavasti. Käsitellään juurisolmu 1. Sitten siirrymme vasemmanpuoleiseen alipuuhun (jonka juuri on 2), ja jätämme solmun 1 oikean alipuun odottamaan kunnes vasen alipuu on käyty kokonaan läpi. Käsittelemme vasemman alipuun juuren (2). Jätämme sen oikean alipuun odottamaan, edeten vasempaan alipuuhun, jonka juuri on 4. Käsittelemme juuren, ja huomaamme ettei vasenta alipuuta ei ole, eikä liiemmin oikeaakaan. Palaamme takaisin solmuun 2, jonka vasen alipuu on nyt käsitelty kokonaan. Siispä jatkamme solmun 2 oikeaan alipuuhun, jonka juuri on 5. Käsittelemme juuren (5). Siirrymme sen vasemmpaan alipuuhun (jonka juuri on 7), ja ...

Solmut käytäisiin siis läpi seuraavassa järjestyksessä:
1,2,4,5,7,9,11,12,10,3,6,8.

Hanoin tornit

Perinteinen esimerkki aidosti rekursiivisesta tehtävästä on Hanoin tornien ongelma. Tämä ongelma on saanut alkunsa Hanoissa sijaitsevaan munkkiluostariin liittyvästä legendasta. Tarinan mukaan luostarin munkit saivat jumalallisessa ilmestyksessä tehtäväkseen siirtää 64 kivistä kiekkoa lähtötolpasta (A, katso kuva alla) maalitolppaan (B). Apunaan munkit saivat käyttää aputolppaa (C). Jumalallisten ohjeiden mukaan suoritusta pitää pystyä vartioimaan ylhäältä käsin, joten isompi kiekko ei saa peittää pienempää missään tilanteessa. Munkkien käsityksen mukaan kivikiekkojen siirtely on hengellisesti kohottavaa toimintaa, ja legendan mukaan tehtävän päätyttyä (kaikki 64 kiekkoa maalitolpassa) koittaa maailmanloppu. Vielä ei kuitenkaan kannata ahdistua tästä, malta mielesi tämän jakson loppuun.



Ongelman riisuttu formaalinen asetelma on seuraava: Kolmessa tolpassa on kiekkoja. Kiekkoja voidaan siirtää yksi kerrallaan missä järjestyksessä tahansa tolpasta toiseen sillä rajoituksella, että isompi kiekko ei saa peittää pienempää. Alussa kiekot ovat lähtötolpassa suuruusjärjestyksessä kuvan mukaisesti (kuvassa ei ole ongelman yksinkertaistamiseksi ja piirtoteknisistä seikoista johtuen 64 kiekkoa).

Tehtävä voi tuntua aluksi vaikealta, jos sitä lähdetään ajattelemaan iteratiivisen ratkaisun pohjalta. Sen sijaan reduktiivinen ajattelu tuo tulosta:

1. Siirrä $n-1$ kiekkoa (alkuperäisessä ongelmassa 63, kuvassa kolme) lähtötolpasta aputolppaan
2. Siirrä pohjimmainen kiekko maalitolppaan
3. Siirrä aputolpassa olevat $n-1$ kiekkoa maalitolppaan

Koska alkuperäinen ongelma oli siirtää kaikki n kiekkoa lähtötolpasta maalitolppaan, huomataan että yo. algoritmin 1. ja 3. vaiheet ovat selvästi rekursiivisia kutsuja ongelman ratkaisevalle moduulille. Kiekkojen määrä pienenee aina yhdellä, ja näin alitehtävät ovat aidosti alkuperäistä yksinkertaisempia. Rekursiokutsuissa tulee myös huomioida tolppien muuttuvat roolit suorituksen aikana; lisäämällä moduulin kutsuun tiedot tolppien rooleista voimme tarkentaa vaiheita 1 ja 3 seuraavasti:

- hanoi(määrä-1, lähtö, apu, maali)
- hanoi(määrä-1, apu, maali, lähtö)

Kun tähän algoritmiin lisätään vielä toiminta triviaalin tapauksen sattuessa, olemme jo ratkaisun ytimessä (tässä yhteydessä esitetty pseudokoodina):

```
hanoi(määrä, lähtö, maali, apu)
{
   JOS (maara = 1)
   {
      Siirrä kiekko lähdöstä maaliin
   }
   MUUTEN
   {
      hanoi(määrä-1, lähto, apu, maali);
      hanoi(1, lähto, maali, apu);
      hanoi(määrä-1, apu, maali, lähto);
   }
}
```

Suoritettavaksi koodiksi muutettuna tämä algoritmi tuottaa kolmella kiekolla seuraavanalaisen tulosteen:

```
Siirrä kiekko tolpasta A tolppaan B
Siirrä kiekko tolpasta A tolppaan C
Siirrä kiekko tolpasta B tolppaan C
```

```
Siirrä kiekko tolpasta A tolppaan B
Siirrä kiekko tolpasta C tolppaan A
Siirrä kiekko tolpasta C tolppaan B
Siirrä kiekko tolpasta A tolppaan B
```

Algoritmin ollessa rekursiivinen sen aikavaativuus (yo. tulosteen yksi rivi vastaa yhtä kiekon siirtoa) kasvaa eksponentiaalisesti. Tarkkaan ottaen algoritmin aikavaativuus on $2^n - 1$ (kts. Punainen kirja sivu s. 85-86), missä n on kiekkojen määrä. Näin voidaan laskea, että 64 kiekolla ongelman ratkaisemiseen menee $2^{64} - 1 = 18446744073709551615$ siirtoa!

Jos oletetaan yhteen siirto-operaatioon kuluvan aikaa yksi sekunti, kestää ratkaisu 64 kiekolla n. 600 miljardia vuotta. Maailmanloppu ei ole siis tulossa ainakaan huomenna.

6.2 Induktiotodistus

Johdatteleva esimerkki induktion käytöstä

Esimerkki: Olkoon tehtävänä laskea n:n ensimmäisen parittoman positiivisen kokonaisluvun summa $1 + 3 + \ldots + (2n - 1)$, käyttämättä aritmeettisen sarjan summan kaavaa. Tutkitaan miten summa käyttäytyy pienillä arvoilla:

$$\begin{aligned} 1 &= 1 \\ 1 + 3 &= 4 \\ 1 + 3 + 5 &= 9 \\ 1 + 3 + 5 + 7 &= 16 \\ 1 + 3 + 5 + 7 + 9 &= 25 \end{aligned}$$

Tästä huomataan, että oikealla puolella on lukujen 1, 2, 3, 4 ja 5 toinen potenssi. Mutta tämä on todistettava jotenkin. Kaikkien arvojen laskeminen ei ole mahdollista, sillä lukuja on äärettömästi. Täytyy siis löytää jokin parempi tapa.

Olkoon s_n lause $1 + 3 + \ldots + (2n - 1) = n^2$. Aiemmin laskimme (todistimme) että ainakin tapaukset $s_1, s_2, \ldots, s_5$ ovat tosia. Arvon s_6 todistaminen käy aiemman lauseen totuuden (s_5) avulla, sillä

$$1 + 3 + 5 + 7 + 9 + 11 = \underbrace{(1 + 3 + 5 + 7 + 9)}_{s_5} + 11 = 25 + 11 = 36$$

Tätä voidaan soveltaa taaksepäin, eli s_5:n totuus tulee osittain s_4:n totuudesta, ja loppujen lopuksi huomaamme että kaikki lähtee s_1:n totuudesta. Kuitenkin meitä kiinnostaa nyt s_6:n totuuden todistaminen. Vastaavalla tavalla kuin s_5:n totuudesta seuraa s_6:n totuus, niin mielivaltaisen s_k:n totuudesta seuraa s_{k+1}:n totuus. Ja koska k on mielivaltainen,niin kaikki arvot tulevat kätevästi todistetuksi samalla kertaa.

Oletetaan, että s_n on tosi arvolla $n = k$, siis $s_k = 1 + 3 + \ldots + (2k - 1) = k^2$. Seuraavaksi johdetaan arvo s_n:n arvolle $n = k + 1$:

$$\begin{aligned} s_{k+1} &= 1 + 3 + \ldots + (2k - 1) + (2(k + 1) - 1) \\ &= \underbrace{1 + 3 + \ldots + (2k - 1)}_{S_k: \quad = k^2} + (2k + 1) \\ &= k^2 + 2k + 1 \\ &= (k + 1)^2 \end{aligned}$$

Nyt s_1 on tosi, s_1:n totuudesta seuraa s_2:n totuus, s_2:sta s_3:n, ja niin edelleen. Yleisesti siis s_k:n totuudesta seuraa s_{k+1}:n totuus. Tästä voimme päätellä matemaattisella induktiolla, että s_n on tosi $\forall n \in \mathbb{Z}_+$, ja väite on todistettu.

Induktioperiaate

Olkoot $\{s_m, s_{m+1}, \ldots \mid m \in \mathbb{Z}\}$ lauseita. Olkoon tehtävänä todistaa, että lause s_n on tosi kaikilla $n \in \{m, m+1, \ldots\}$. Menettelemme seuraavasti:

1) Perusaskel: osoitetaan että s_n on tosi arvolla $n = m$.
2) Induktioaskel: tehdään *induktio-oletus* että s_n on tosi arvolla $n = k$ $(k \geq m)$, ja todistetaan *induktioväite*, että s_n on tosi myös arvolla $n = k + 1$.
   Vaihtoehtoisesti voi osoittaa, että s_{k-1}:n totuudesta aina seuraa s_k:n totuus.
3) Johtopäätös: tämän jälkeen alkuperäinen väitös seuraa induktioperiaatteesta.

Jokaisessa induktiotodistuksessa on nämä kolme vaihetta.

Esimerkki 1 Todista induktiolla, että $n^2 + n$ on parillinen $\forall n \in \mathbb{N}$.

Olkoon $f(n) = n^2 + n$.
Perusaskel: $f(1) = 2$ on parillinen.
Induktio-oletus: $f(k)$ on parillinen, eli $f(k) = 2p \mid p \in \mathbb{N}$.
Induktioväite ja sen todistus:

$$\begin{aligned} f(k+1) &= (k+1)^2 + (k+1) \\ &= \underline{k^2} + 2k + 1 + \underline{k} + 1 \\ &= \underbrace{k^2 + k}_{\text{Tämähän on sama kuin } f(k)} + 2(k+1) \\ &= 2p + 2(k+1) \\ &= 2(p + k + 1), \end{aligned}$$

joten $f(k+1)$ on parillinen.
Johtopäätös: induktioperiaatteen nojalla $f(n)$ on parillinen kaikilla $n \in \mathbb{N}$. $\square$

Esimerkki 2 Todista induktiolla, että $n^3 - n$ on jaollinen kolmella $\forall n \in \mathbb{N}$.

Olkoon $f(n) = n^3 - n$.
$f(1) = 0$ on jaollinen kolmella.
$(f(2) = 2^3 - 2 = 8 - 2 = 6 = 3 \times 2$ on jaollinen kolmella.)
Oletetaan että $f(k)$ on jaollinen kolmella, eli $f(k) = 3p \mid p \in \mathbb{N}$.
Silloin myös $k+1$:n pitää olla jaollinen kolmella:

$$\begin{aligned} f(k+1) &= (k+1)^3 - (k+1) \\ &= k^3 + 3k^2 + 3k + 1 - k - 1 \\ &= \underbrace{k^3 - k}_{f(k)=3p} + 3(k^2 + k) \\ &= 3p + 3(k^2 + k) = 3(p + k^2 + k), \end{aligned}$$

josta huomaamme, että näin on. Induktioperiaatteen nojalla $f(n)$ on jaollinen kolmella $\forall n \in \mathbb{N}$. $\square$

[Lisää esimerkkejä Ensley kohdat 2.2-2.4].

7 Kombinatoriikkaa ja todennäköisyyslaskentaa

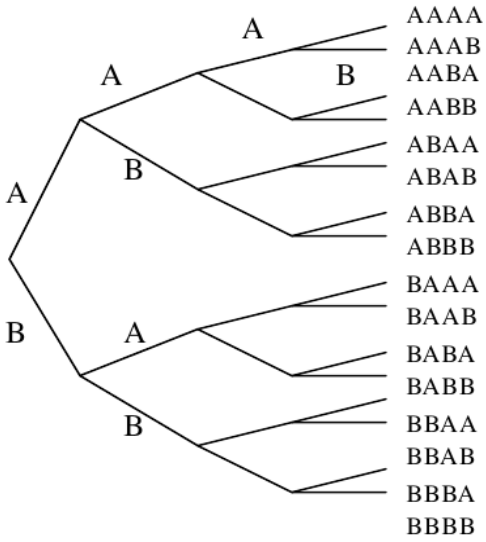
Vuoden 2011 luennoilla käytetään Ensley&Crawley -kirjan kalvomateriaalia. Linkki kurssin www-sivuilla.

Alla hieman materiaalia yhdistetty ym materiaalista ja UEF/Maija Marttila-Kontion DR-materiaalista.

7.1 Kertosääntö

Tarkastellaan aluksi seuraavaa ongelmaa: Kuinka monta nelikirjaimista koodisanaa voidaan tehdä kahdesta kirjaimesta A ja B? Ratkaistaan ongelma tämän puudiagrammin avulla.

Sanoja tulee yhteensä 16 kappaletta.



Kertosääntö Oletetaan, että tehdään t kertaa valintoja (poimintoja), s.e. 1. valinnassa on n_1 eri mahdollisuutta valita, 2. valinnassa on n_2 eri mahdollisuutta valita, ..., t. valinnassa on n_t eri mahdollisuutta valita. Näin eri valintamahdollisuuksien koko määrä on:

$$N = n_1 \times n_2 \times \ldots \times n_t$$

Tämä itseasiassa on tuttu joukkojen karteesisesta tulosta.

Kertosääntö antaa vaihtoehtojen määrän kun sallitaan toisto ja alkioiden järjestyksellä on merkitys.

7.2 Järjestetty otos ja permutaatio

Tutkitaan aluksi seuraavaa esimerkkiä: Kuinka monella eri tavalla joukosta A, B, C, D voidaan valita kaksi kirjainta s.e. (poiminta)järjestyksellä on merkitystä? (ts. AB $\neq$ BA jne).

Tehtävä ei ole esitetty täysin yksiselitteisesti, sillä ratkaisu riippuu nyt siitä, onko toisto sallittua (sallitaanko esim. AA tai DD).

1) Toisto sallitaan:

AA	BA	CA	DA	(16 kpl)
AB	BB	CB	DB	
AC	BC	CC	DC	
AD	BD	CD	DD	

2) Toistoa ei sallita:

AB	BA	CA	DA	(12 kpl)
AC	BC	CB	DB	
AD	BD	CD	DC	

Järjestetty k-otos jossa toisto on sallittu joukosta A on k-jono $(x_1, x_2, \ldots, x_k)$ siten, että:

1. $x_j \in A$ $j = 1..k$
2. Järjestyksellä on merkitystä, ts. $ab \neq ba$.
3. sama alkio voi esiintyä $0..k$ kertaa.

Kertosäännön nojalla järjestettyjen k-otosten (toisto sallittu) kokonaismäärä joukosta $A = a_1, a_2, \ldots, a_n$ on: $N = n \times n \times \ldots \times n = n^k$.

Permutaatio

Kielletään nyt toisto, eli kun yksi alkio on valittu, sitä ei voida enää valita uudelleen. Järjestyksellä on edelleen merkitystä.

1. Kun n alkioisesta joukosta valitaan yksi alkio otoksen ensimmäiseksi, sen voi tehdä n tavalla.
2. Kun $n-1$ alkioisesta jälkellä olevasta joukosta valitaan yksi alkio otoksen toiseksi, sen voi tehdä $n-1$ tavalla.
   $\vdots$
k. Kun $n-k+1$ alkioisesta jälkellä olevasta joukosta valitaan yksi alkio otoksen k:nneksi, sen voi tehdä $n-k+1$ tavalla.

Yhteensä valintoja voitiin tehdä $n(n-1)(n-2)\ldots(n-k+1) = \dfrac{n!}{(n-k)!}$ kappaletta. Merkitään $P(n,k) = \dfrac{n!}{(n-k)!}$, eli n-alkioisen joukon k-mittaisten permutaatioiden määrä.

7.3 Ei-järjestetyt otokset, binomikerroin

Kun permutaatioista tulkitaan eri järjestykset samanlaisiksi, niiden määrä vähenee. k-mittainen jono voidaan laittaa $k!$ järjestykseen. Niinpä erilaisia k-alkioisia osajoukkoja n-alkioisesta joukosta on

$$\frac{P(n,k)}{k!} = \frac{n!}{k!(n-k)!}\text{kappaletta.}$$

Tätä lukua sanotaan binomikertoimeksi ja merkitään
$C(n,k) = \binom{n}{k} = \dfrac{n!}{k!(n-k)!}$. Luetaan "$n$ yli k:n".
Se siis kertoo monellako tavalla k alkiota voidaan valita n alkiosta.

$C(n,k)$ kertoo myös sellaisten n mittaisten binäärilukujen määrän joissa on tasan k ykköstä! Siis monellako tavalla voidaan valita k positiota n:stä mahdollisesta.

Ei-järjestetty otos, toisto sallittu

Edellisessä permutaatiossa valintojen määrä väheni jokaisella kierroksella. Kun toisto sallitaan, valintoja on enemmän. Näitä on $\binom{n+k-1}{k}$.

7.4 Todennäköisyyslaskentaa

Klassisen todennäköisyyslaskennan määritelmiä:

- Sanotaan, että alkeistapahtumat ovat symmetrisiä.
- Tapahtuman $A \subseteq S$, missä $|A| = k$ todennäköisyys on $\Pr(A) = \frac{k}{n}$
- $\Pr(\varnothing) = 0$ ja $\Pr(S) = 1$.

Esimerkkejä:

- Lapsen sukupuoli: $S = \{\text{Tyttö, Poika}\}$.
- Kolikonheitto: $S = \{\text{Kruuna, Klaava}\}$.
- Nopanheiton tulos: $S = \{1, 2, 3, 4, 5, 6\}$.
- Kahden nopanheiton tulokset: $S = \{1, 2, 3, 4, 5, 6\}^2$.
- Hajautusfunktion arvot välillä $\{0 \dots M - 1\}$.

Joukko-oppi ja tapahtumat

A ei tapahdu (komplementti)	$A^c = \{s \in S \mid s \notin A\}$
A tai B tapahtuu	$A \cup B = \{s \in S \mid s \in A \vee s \in B\}$
A ja B tapahtuvat	$A \cap B = \{s \in S \mid s \in A \wedge s \in B\}$
A tapahtuu, mutta B ei	$A \setminus B = \{s \in S \mid s \in A \wedge s \notin B\}$

Venn-kaavioita voidaan käyttää myös todennäköisyyslaskennassa.

Tapahtuman komplementin todennäköisyys : $\Pr(A^c) = 1 - \Pr(A)$.

Toisensa poissulkevat tapahtumat

- Jos $A \cap B = \varnothing$ ne ovat toisensa poissulkevia.
- Jos A ja B ovat toisensa poissulkevia, niin:
  - $\Pr(A \cap B) = 0$
  - $\Pr(A \cup B) = \Pr(A) + \Pr(B)$
  - $\Pr(A \setminus B) = \Pr(A)$

Lisää laskusääntöjä :

- $\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B)$ (yleinen yhteenlaskusääntö)
- $\Pr(A \setminus B) = \Pr(A \cap B^c) = \Pr(A) - \Pr(A \cap B)$
- Jos $B \subseteq A$, niin $\Pr(A) \geq \Pr(B)$ ja
  $\Pr(A \setminus B) = \Pr(A) - \Pr(B)$.

Ehdollinen todennäköisyys :

- Tapahtuman A todennäköisyys sillä ehdolla, että B on tapahtunut:
  $\Pr(A \mid B) = \dfrac{\Pr(A \cap B)}{\Pr(B)}$. $A \perp B \Leftrightarrow \Pr(A \mid B) = \Pr(A)$.
- $\Pr(A \mid A) = 1$

Usean tapahtuman todennäköisyys

Usea riippumaton tapahtuma

- Oletetaan tapahtumat A ja B joiden tapahtuminen ei riipu toisistaan, merkitään $A \perp B$.
- Nyt $\Pr(A \cap B) = \Pr(A) \times \Pr(B)$.
- Vastaavasti k riippumatonta tapahtumaa $A_1, A_2, \ldots, A_k$:
- Nyt $\Pr(A_1 \cap A_2 \cap \ldots \cap A_k) = \Pr(A_1) \times \Pr(A_2) \times \ldots \times \Pr(A_k)$.

Usea tapahtuma yleisessä tilanteessa

- A ja B tapahtuvat:
  $\Pr(A \cap B) = \Pr(A) \times \Pr(B \mid A)$
- vastaavasti k tapahtumalle.

Esimerkkejä

[Ensley]

Syntymäpäiväparadoksi ja sen sovelluksia

Olkoon perusjoukko S lukuja $1 \ldots n$, millä todennäköisyydellä satunnaisesti valituissa k luvussa on vähintään kaksi samaa lukua?

Millä todennäköisyydellä tulee törmäys?

$\Pr(1) = 0$	ei voi tulla törmäystä
$\Pr(2) = \frac{1}{n}$	törmäyksen todennäköisyys toiselle luvulle
$\Pr(3) = \frac{1}{n} + \frac{2}{n}$	(Väärin!) törmäyksen todenn. toiselle tai kolmennelle???
	Toisen alkion mahdollinen törmäys on huomioitava!
$\Pr(k) = \sum_{i=1}^{k}$??	

Kun tehtävänä on laskea todennäköisyys jolla "vähintään kaksi samaa lukua", on helpompi laskea sen komplementti ja vähentää tulos yhdestä.

Lasketaan siis millä todennäköisyydellä ei tule törmäystä ja vähennetään se yhdestä.

Millä todennäköisyydellä tulee törmäys, uusi yritys:

$\Pr(1)$	$=$	$1 - \left(\frac{n}{n}\right) = 0$	mikään alkio ei törmää
$\Pr(2)$	$=$	$1 - \left(\frac{n-1}{n}\right)$	$n-1$ törmäämätöntä mahdoll. 2. alkiolle
$\Pr(3)$	$=$	$1 - \left(\frac{n-1}{n} \cdot \frac{n-2}{n}\right)$	$n-2$ törmäämätöntä mahdoll. 3. alkiolle
$\vdots$			
$\Pr(k)$	$=$	$1 - \left(\frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \ldots \cdot \frac{n-k}{n}\right)$	
	$=$	$1 - \frac{n!}{(n-k)! \cdot n^k}$	

Esimerkkejä törmäyksen todennäköisyydestä:

n	k	Pr
365	5	0.014
365	10	0.12
365	20	0.41
365	23	0.51
365	30	0.71
365	35	0.81
365	40	0.89
365	50	0.97
365	60	0.994
365	80	0.99991
365	100	0.9999997
100000	100	0.048
100000	1000	0.993
1000000	100	0.0049
1000000	1000	0.39
$\vdots$		

Mitä tästä opimme?

- Satunnaisella datalla törmäyksiä tulee.
- Tuuriin ei voi luottaa.
- Hajautuksen törmäyskäsittely
- Tiedoston, tietokannan, kommunikaatioväylän, jne lukitukset

8 Verkot eli graafit

Tarkoituksenahan on etsiä tiedon esitysmuotoja, jotka voidaan muuntaa biteiksi, joita tietokone voi käsitellä. Erilaisiin tarkoituksiin tarvitaan erilaisia tiedon esitysmuotoja. Monet reaalimaailman asiat eivät luonteeltaan ole peräkkäisiä (listarakenteeseen sopivia) tai hierarkkisia (puurakenteeseen sopivia), vaan olioiden suhteet voivat olla monimuotoisempia. Niinpä esittelemme verkot eli graafit joilla voidaan mallintaa monimutkaisia suhteita. Matematiikassa graafiteoria on oma tutkimusalansa ja sen menetelmät ja tulokset hyödyttävät tietojenkäsittelijän reaalimailman ongelmien ratkaisemista.

Graafiteoriaa ja erityisesti graafialgoritmeja sekä niiden toteuttamista käsitellään enemmän Tietorakenteet ja algoritmit II -kurssilla.

Esimerkkejä tiedosta jota graafeilla mallinnetaan:

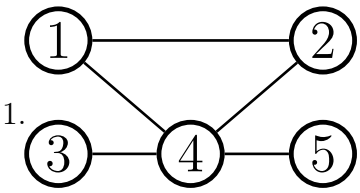
- (Tieto)liikenteen mallinnus.
  - Tietoliikenneverkko, tieverkko, virtapiirit, prosessikaaviot.
  - Reititys, liikennesuunnittelu, virtasuunnittelu.
- Relaatiokaaviot, luokkakaaviot, olio-suhdekaaviot, käyttötapauskaaviot, organisaatiokaaviot, sukupuut, jne.
- Hypertekstit, viittaukset.
- Kemialliset, fysikaaliset, tekniset, biologiset, sosiologiset, taloudelliset verkostot ja niiden suhteet ja vuorovaikutukset.

Määritelmä: *Graafi* $G = (V, E)$ koostuu joukosta V *solmuja* (*vertex, node*) ja niitä yhdistävistä *kaarista* (E) (*edge*). Jos kaaren määrittelevät solmuparit ovat järjestettyjä, on kyseessä suunnattu graafi, muuten graafi on suuntaamaton. Joten V on itse asiassa luettelo solmuista ja E on luettelo kaarista. Suuntaamattoman graafin kaaret merkitään $\{a, b\}$ ja suunnatun (a, b). Huomaa, että $(a, b) \neq (b, a)$, mutta $\{a, b\} = \{b, a\}$.

Huom! Käytetään myös merkintää $G = (V, E, f)$, missä f on solmujen välisen yhteyden määrittelevä kuvaus. Suunnatussa graafissa $E \subseteq V \times V$.

Esim:

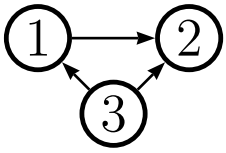
1.



$$G = (\overbrace{\{1, 2, \underbrace{3}_{\text{Solmu}}, 4, 5\}}^{\text{Joukko V}},$$

$$\underbrace{\{\{1,2\},\{1,4\},\{2,4\},\{3,4\},\overbrace{\{4,5\}}^{\text{pari, kaari}}\}}_{\text{Joukko E}})$$

2.



$$G = (\{1, 2, 3\}, \{(1, 2), (3, 1), (3, 2)\})$$

Kaaret

Jos pari $e \in E$ on järjestetty, niin merkitsemme (u, v), ei-järjestettyä paria merkitsemme $\{u, v\}$. Tässä $u, v \in V$.

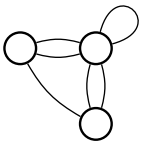
Kaaren yhdistämiä solmuja sanotaan *lähtösolmuksi* ja *maalisolmuksi* (tai *päätesolmuksi*).

Jos kaari alkaa ja loppuu samasta solmusta, muodostaa se silmukan. Kahden solmun välillä voi tarvittaessa olla useampia kaaria. Graafi, jossa on rinnakkaisia kaaria, on *multigraafi*. Vastaavasti *yksinkertaisessa graafissa* on vain yksi kaari solmujen välillä.

Sovelluksesta riippuen silmukan ja rinnakkaiset kaaret voidaan sallia tai kieltää.

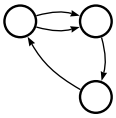
Esim:

a)



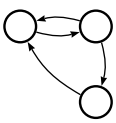
kaksi paria rinnakkaisia kaaria ja silmukka

b)



yksi pari rinnakkaisia kaaria

c)



ei rinnakkaisia kaaria

Painotetut kaaret

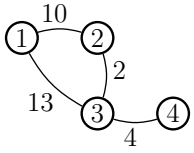
Mikäli liitämme kuhunkin graafin kaareen painon (luvun), on kyseessä *painotettu graafi*. Kaareen voi liittyä muutakin ongelmaan liittyvää tietoa. Tällöin puhutaan semanttisesta verkosta.

Tarvittaessa myös solmut voidaan varustaa painolla ja/tai varustaa muulla sovelluksen kannalta hyödyllisellä tiedolla.

Yleensä myös algoritmin on ylläpidettävä väliaikaista tietoa solmuista ja/tai kaarista. Näppärintä on jos tiedot voi tallentaa suoraan solmuna/kaaren objektiin.

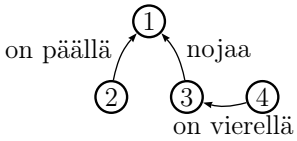
Esim:

1)

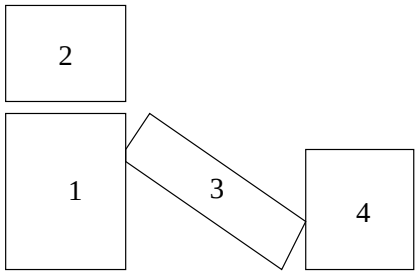


painotettu graafi,
esimerkiksi kaukoliikenteen
linja-autovuorojen ajat

2)



semanttinen verkko, kuvaa
miten laatikot sijoittuvat
toisiinsa nähden



Solmujen asteet

Solmu, joka ei liity mihinkään solmuun, on *isoloitu* eli *eristetty* solmu. Suunnatun graafin kustakin solmusta lähtevien kaarien määrä on *lähtöaste* (*outdegree*) ja tulevien kaarien määrä *tuloaste* (*indegree*). Näiden summa on *solmun aste* eli (*total degree*). Suuntaamattomassa graafissa solmun aste on siihen liittyvien kaarien lukumäärä. Eristetyn solmun aste on 0. Solmun, johon liittyy silmukka, mutta ei muita kaaria, aste on 2.

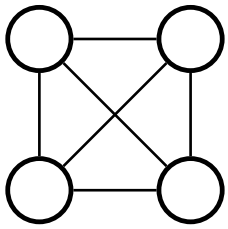
Graafin solmujen asteiden summa on kaksi kertaa kaarien lukumäärä.

Aligraafit

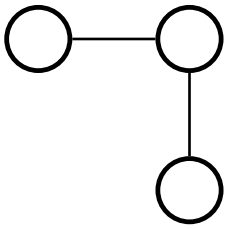
Koska graafit esitetään solmujoukkojen avulla, on luonnollista käyttää joukko-opin ilmaisuja ja sääntöjä tiettyjen asioiden määrittelemiseen.

Olkoon $V(H)$ graafin H solmujen osajoukko ja $V(G)$ graafin G solmujen joukko siten että $V(H) \subseteq V(G)$. Jos kaikki graafin H kaaret ovat myös G:n kaaria, eli $E(H) \subseteq E(G)$, on H graafin G:n *aligraafi* ja sitä merkitään yksinkertaisemmin $H \subseteq G$.

Esim:



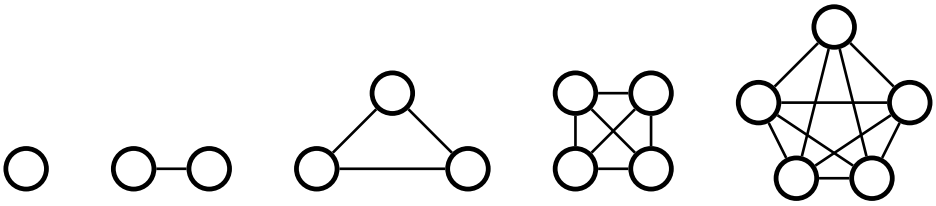
G



H

$H \subseteq G$

Graafi on täydellinen (*complete*), jos kukin solmu on liitetty kaarella kaikkiin muihin solmuihin. Merkitään K_n.



K_1 K_2 K_3 K_4 K_5

Graafi on kaksijakoinen (*bipartite*), jos solmut voidaan jakaa kahdeksi sellaiseksi osajoukoksi, että osajoukon sisällä solmuja ei yhdistä kaari.

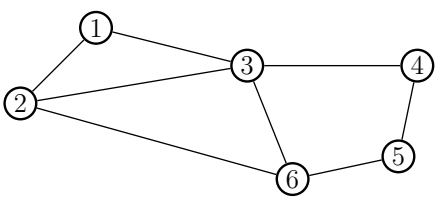
8.1 Polut

Graafeihin liittyvä ongelmanratkaisu on esimerkiksi lyhimmän polun etsimistä.

Olkoon $G = (V, E)$ suunnattu graafi. Kaarien jonoa kutsutaan *poluksi*, jos kunkin kaaren päätesolmu on seuraavan kaaren lähtösolmu.

Polku on siis jono $\langle (v_1, v_2), (v_2, v_3), (v_3, v_4), \ldots, (v_{n-1}, v_n) \rangle$, ja sitä merkitään lyhyemmin $\langle v_1, v_2, v_3, \ldots, v_n \rangle$. Polkuun kuuluvien kaarien määrä on *polun pituus*. Suunnatun graafin polkua, jonka kaikki kaaret ovat erillisiä, kutsutaan *yksinkertaiseksi poluksi* (simple path). Polkua, jonka solmut ovat erillisiä, kutsutaan *alkeispoluksi* (elementary path). Kaikki alkeispolut ovat myös yksinkertaisia. Polku, joka alkaa ja päättyy samaan solmuun, on *sykli*. Suunnattu graafi voi olla sellainen, että siinä ei ole yhtään sykliä.

Esim:



polku:$\langle 3, 2, 3, 6 \rangle$, pituus 3.
yksinkertainen polku:$\langle 3, 2, 1, 3, 6 \rangle$, pituus 4.

alkeispolku:$\langle 3, 1, 2, 6\rangle$, pituus 3
sykli:$\langle 3, 4, 5, 6, 3\rangle$, pituus 4

Solmujen välinen etäisyys

Solmujen u ja v välisen lyhimmän polun pituutta sanotaan solmujen u ja v *väliseksi etäisyydeksi* ja merkitään $d(u, v)$. Huomaa, että etäisyys solmusta itseensä on $d(u, u) = 0$. Kaikille verkoille pätee, että jos on polku u:sta v:hen ja v:stä w:hen, niin on polku u:sta w:hen. Mutta tässäkin pitää olla tarkkana, suunnatussa graafissa ei välttämättä olekaan polkua w:stä u:hun.

Etäisyydelle pätee

$$d(u, v) \geq 0$$
$$d(u, u) = 0$$
$$d(u, v) + d(u, w) \geq d(u, w)$$

Jos solmusta u ei ole polkua solmuun v, merkitään $d(u, v) = \infty$.

Huom! Vaikka olisikin polku $\langle u, v\rangle$ ja polku $\langle v, u\rangle$, ei välttämättä päde $d(u, v) = d(v, u)$.

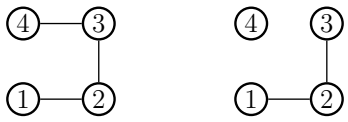
Suunnatussa graafissa kunkin alkeispolun pituus on korkeintaan $n - 1$, kun graafissa on n solmua. Alkeissyklin pituus ei ole suurempi kuin n.

Jos graafin kaarilla on painot, polun pituuten käsitellään yleensä polun kaarten painojen summaa.

8.2 Yhtenäisyys

Suuntaamaton graafi G on *yhtenäinen* (connected) jos kaikkien graafin solmujen välillä on olemassa polku.

Yhtenäinen: Ei-yhtenäinen:



Aiempaan tulokseen

$$\sum_{v \in V} \text{aste}(v) = 2|E|$$

liittyen saamme suuntaamattomassa graafissa ja multigraafissa paritonta astetta olevien solmujen lukumääräksi parillisen luvun.

Suunnatun graafin yhtenäisyys

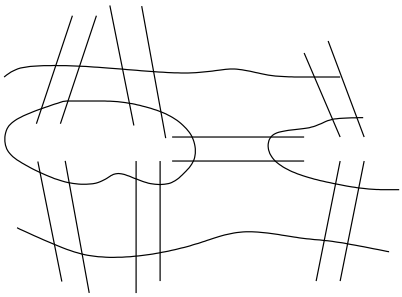
Suunnatun graafin sanotaan olevan vahvasti yhtenäinen jos jokaisesta solmusta on polku jokaiseen muuhun solmuun.

Suunnatun graafin sanotaan olevan yhdensuuntaisesti yhtenäinen jos jostakin solmusta on polku jokaiseen muuhun solmuun.

Suunnatun graafin sanotaan olevan heikosti yhtenäinen jos se olisi yhtenäinen kun jokainen kaari olisi kaksisuuntainen.

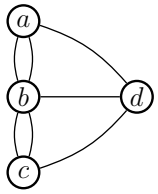
8.3 Piirit

"Königsbergin sillat" on tunnettu graafiteorian ongelma.



Voidaanko sunnuntaikävelyllä kulkea kaikki sillat täsmälleen kerran ja palata lähtöpisteeseen? Lähtöpiste voi olla missä vain. Ongelma voidaan esittää graafin läpikäyntiongelmana:

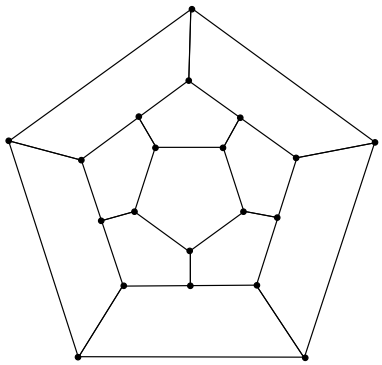
Onko olemassa sellaista polkua, jossa mistä tahansa solmusta lähtien viereinen multigraafi käydään läpi niin, että kukin kaari käydään läpi vain kerran? Graafin solmujen asteet ovat aste(a) = aste(c) = aste(d) = 3 ja aste(b) = 5.



Määritelmä Olkoon $G = (V, E)$ suuntaamaton graafi tai multigraafi, jossa ei ole eristettyjä solmuja. G:stä sanotaan löytyvän *Eulerin piiri*, jos G:ssä on piiri, joka käy läpi graafin kaaret täsmälleen kerran. Jos G:ssä on avoin polku solmusta a solmuun b, joka kulkee kunkin kaaren täsmälleen kerran, polkua sanotaan *Eulerin poluksi*.

Hamiltonin polku ja piiri

Vuonna 1859 irlantilainen matemaatikko Hamilton kehitti pelin dublinilaisille lelutehtaille. Tehtävänä oli kulkea olla oleva graafi niin, että kussakin solmussa käydään vain kerran ja polku muodostaa piirin (siis päättyy samaan solmuun kuin mistä alkaa).

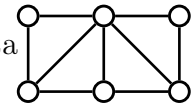


Määritelmä Olkoon $G = (V, E)$ graafi tai multigraafi, jolle pätee $|V| \geq 3$. Graafin piiriä sanotaan *Hamiltonin piiriksi*, jos on olemassa piiri siten, että se käy läpi kaikki solmut täsmälleen kerran. Hamiltonin polku on polku, joka käy läpi kaikki graafin solmut täsmälleen kerran.

Huom! Hamiltonin piiristä saadaan Hamiltonin polku, kun poistetaan mikä tahansa kaari.

Eulerin ja Hamiltonin polkujen löytäminen

Kuinka voimme sitten päätellä, onko jossakin graafissa Eulerin tai Hamiltonin polkua tai piiriä?

Esim: Graafista 

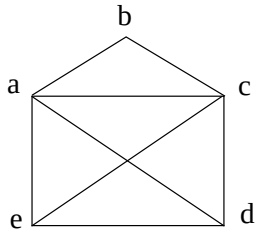
löytyy Hamiltonin piiri, mutta ei Eulerin piiriä.

Olkoon $G = (V, E)$ suuntaamaton graafi tai multigraafi, jossa ei ole eristettyjä solmuja. Tällöin G:stä löytyy Eulerin piiri jos ja vain jos G on yhtenäinen ja kaikkien G:n solmujen aste on parillinen.

Nyt huomaamme myös, että Königsbergin sillat-ongelmaan vastaus on kielteinen, kysyttyä reittiä ei löydy.

Olkoon G suuntaamaton graafi tai multigraafi, jossa ei ole eristettyjä solmuja. Tällöin graafista löytyy Eulerin polku jos ja vain jos graafissa on täsmälleen kaksi solmua, joiden aste on pariton (ellei graafissa ole piiriä, joka lasketaan poluksi).

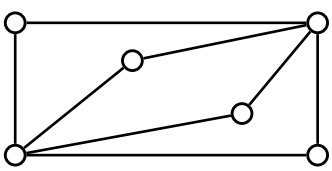
Esim. Piirrä kirjekuori kynää nostamatta ja kukin viiva vain yhteen kertaan.



aste(a) = aste(c) = 4
aste(b) = 2
aste(e) = aste(d) = 3

Königsbergin siltoja ei siis voi kiertää Eulerin polkuakaan pitkin.
Edellä kävimme graafia läpi kiinnittäen huomion kaariin. Tietyssä solmussa voidaan käydä useammin kuin kerran. Tämä sopi Königsbergiin, sillä tarkasteltiin siltoja, jotka luontevasti vastaavat kaaria. Jos ajattelemme joukkoa kaupunkeja, joissa meidän tulisi käydä, siirrymme graafin solmujen tarkasteluun.

Esim: Graafista



löytyy Eulerin piiri, mutta ei Hamiltonin piiriä.

Huom! Eulerin polku ja piiri vaatii, että kaaret käydään läpi vain kerran. Solmussa voidaan käydä useammin. Vastaavasti voidaan graafia tarkastella myös solmujen kannalta; Hamiltonin polku ja piiri vaatii, että solmussa käydään vain kerran, kaari voidaan kulkea useamminkin – tosin Hamiltonin piirille ja polulle ei ole samanlaisia olemessaololauseita kuin Eulerin piirille ja polulle.

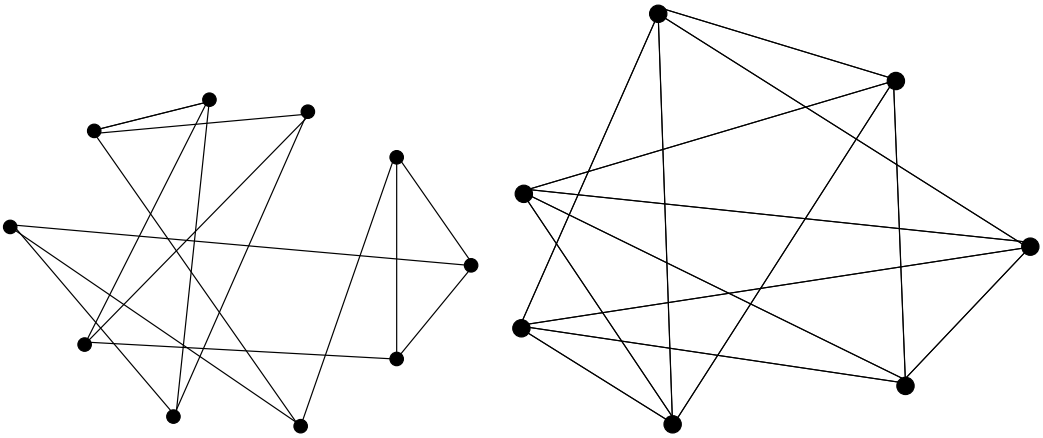
Määritelmä Graafia, jossa ei ole syklejä, sanotaan *puuksi*. Jos puussa kullakin solmulla on korkeintaan kaksi kaarta, on se binääripuu.

8.4 Solmuista ja kaarista

Verkossa $G = (V, G)$ solmun $x \in V$ *naapurisolmuiksi* kutsutaan niitä solmuja, jotka ovat suoraan yhteydessä solmuun x. *Solmun asteluku* on siihen liittyvien kaarten lukumäärä. Edelleen jos solmun asteluku on 0, niin solmu on *eristetty*.

Jos edelleen verkon kaikki solmut ovat eristettyjä, niin verkko on *degeneroitu* eli *nollaverkko*. Jos solmun poistaminen jakaa verkon kahteen erilliseen osaan, niin solmu on *leikkaus-* eli *artikulaatiosolmu*. Lisäksi jos verkon kaikkien solmujen asteluvut ovat samat, niin verkko on *(n-)säännöllinen verkko*, missä n on solmujen asteluku.

Esim: 3- ja 4-säännölliset verkot:



3-säännöllinen graafi 4-säännöllinen graafi

Jos kaaren poistaminen jakaa graafin kahteen erilliseen osaan, niin kaarta kutsutaan *sillaksi*. Jos kaari alkaa ja loppuun samaan solmuun, niin kaarta kutsutaan *silmukaksi* eli *lenkiksi*. Jos taas kahdella (tai useammalla) kaarella on sama alku- ja loppusolmu, niin kaaria sanotaan *rinnakkaisiksi*. Sellaista graafia, jossa ei ole silmukoita tai rinnakkaisia kaaria sanotaan *yksinkertaiseksi graafiksi*.

8.5 Verkon matriisiesitys

Ihmisten on helppo ymmärtää graafin rakenne katsomalla siitä piirrettyä kuvaa. Mutta koneelle asia on hankalampi, ja graafi pitää esittää jossakin muodossa, joka voidaan esittää binäärisesti. Yksi tällainen esitysmuoto on matriisi, jossa matriisin sarake- ja rivitunnukset vastaavat solmujen tunnuksia.

Vierusmatriisi

Olkoon $G = (V, E)$ yksinkertainen suunnattu graafi, jonka solmujen joukko $V = \{v_1, v_2, \ldots, v_n\}$ on järjestetty $v_1$stä v_n:ään. Tällöin $n \times n$-matriisia A, jonka i:nnen rivin j:nnen sarakkeen alkion arvo on 1, mikäli V:n v_i:nnestä solmusta on kaari v_j:een solmuun ja muussa tapauksessa 0, sanotaan graafin G *vierusmatriisiksi* (adjacency matrix).

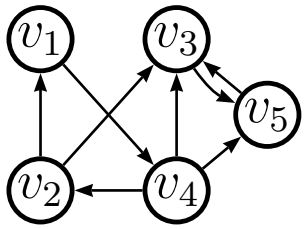
Matriisin A i. rivi määräytyy v_i:stä lähtevien kaarten perusteella. Rivillä olevien ykkösten määrä on yhtäsuuri kuin solmun v_i:n lähtöaste. Vastaavasti, j:nnen sarakkeen ykkösten määrä on yhtäsuuri kuin solmun v_j tuloaste.

Suuntaamattoman graafin $G = (V, E)$ vierusmatriisin esitys riippuu joukon V alkoiden järjestyksestä. Eri järjestykset tuottavat erilaiset vierusmatriisit. On kuitenkin huomattava, että mikä tahansa vierusmatriisi saadaan muunnettua toiseksi vaihtamalla matriisissa rivien ja vastaavien sarakkeiden paikkaa keskenään.

Jos graafi ei ole yksinkertainen, niin sen matriisiesityksessä numerot kertovat rinnakkaisten kaarten määrän. Jos kaaren alku- ja loppusolmut erotetaan toisistaan, eli joukko $E(G)$ ei ole symmetrinen relaatio, niin graafi on sunnattu.

Esim: $G = (V, E)$ on suunnattu matriisi, jossa V on järjestetty joukko $\{v_1, v_2, v_3, v_4, v_5\}$. Sen vierusmatriisi on A.

G:



	1	2	3	4	5
1	0	0	0	1	0
2	1	0	1	0	0
3	0	0	0	0	1
4	0	1	1	0	1
5	0	0	1	0	0

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Jotkin yksinkertaisen suunnatun verkon ominaisuuksista on heti nähtävissä sen vierusmatriisista. Mikäli suunnattu verkko on *refleksiivinen*, niin sen vierusmatriisin lävistäjän alkiot ovat kaikki ykkösiä. Symmetriselle suunnatulle verkolle ja suuntaamattomalle verkolle vierusmatriisi on myös symmetrinen, siis $a_{ij} = a_{ji}\ \forall\ i, j$. Suunnatun graafin matriisiesitys ei ole yleensä symmetrinen.

Vierusmatriisi pidemmille poluille

Edellä tarkastelimme vierusmatriisia vain yhden kaaren pituisille poluille. Nyt tiedämme, että ykkönen vierusmatriisin i:nnen rivin ja j:nnessä sarakkeessa tarkoittaa kaarta (v_i, v_j), eli polkua v_i:stä v_j:hin, jonka pituus on 1. Tarkastellaanpa seuraavaksi vierusmatriisin potensseja. Merkitään matriisin A toisen potenssin A^2 alkioita a_{ij}^2. Tällöin alkioiden arvot määräytyvät seuraavan kaavan mukaan.

$$a_{ij}^2 = \sum_{k=1}^{n} a_{ik} a_{kj}$$

Jollekin määrätylle k:lle $a_{ik}a_{kj} = 1$, jos (ja vain jos) molemmilla alkioilla a_{ik} sekä a_{kj} on arvo 1. Toisin sanoen, jos ja vain jos (v_i, v_k) ja (v_k, v_j) ovat kyseisen verkon kaaria. Jos (v_i, v_k)

ja (v_k, v_j), niin on olemassa polku v_i:stä v_j:in jonka pituus on 2. Siispä A^2:n alkion a^2_{ij} arvo on yhtäsuuri kuin pituudeltaan 2 olevien polkujen lukumäärä, jotka johtavat v_i:stä v_j:in. Tarkastellaan seuraavien A:sta johdettujen matriisien arvoja:

$$A^2 = \begin{pmatrix} 0 & \mathbf{1} & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \quad A^3 = \begin{pmatrix} 1 & 0 & 2 & 0 & 1 \\ 0 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

$$A^4 = \begin{pmatrix} 0 & 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 0 & 2 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & \mathbf{3} & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \quad A^5 = \begin{pmatrix} 0 & 1 & 3 & 0 & 2 \\ 0 & 0 & 2 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 3 & 0 & 3 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Graafissa G on pituudeltaan 2 oleva polku v_1:stä v_2:een, joten A^2:n ensimmäisen rivin toisessa sarakkeessa on arvo 1. Vastaavasti graafissa on kolme kappaletta pituudeltaan neljä olevia polkuja solmusta v_4 solmuun v_3; $\langle 4, 2, 3, 5, 3\rangle$, $\langle 4, 2, 1, 4, 3\rangle$, ja $\langle 4, 5, 3, 5, 3\rangle$, joten $a^4_{43} = 3$. Tämä voidaan yleistää seuraavasti:

Olkoon A suunnatun verkon G vierusmatriisi. Matriisin $A^n (n > 0)$ i:nnen rivin j:nnen alkion arvo on sama kuin n-pituisten polkujen määrä i:nnestä solmusta j:een solmuun.

Polkumatriisi

Vierusmatriiseja voidaan käyttää apuna kun halutaan selvittää onko yhdestä solmusta yhteys toiseen. Jos graafissa on n solmua, riittää että laskemme vierusmatriisit enintään n:n pituisille poluille, sillä jokaisesta polusta voimme muodostaa alkeispolun poistamalla syklit (jolloin pituudeksi tulee enintään $n - 1$) ja jos polku on sykli, niin tarkastelu rajoitetaan ainoastaan alkeisykleihin (pituus enintään n). Kaikki polut ja syklit lasketaan matriisiin

$$B_n = A + A^2 + A^3 + \ldots + A^n, \text{ jossa}$$

i:nnen rivin j:s sarakkeen arvo kertoo enintään n-mittaisten polkujen lukumäärän solmusta v_i solmuun v_j. Jos tämän alkion arvo on erisuuri kuin 0, on selvää että solmusta v_i pääsee solmuun v_j.

Olkoon $G = (V, E)$ yksinkertainen suunnattu verkko, jonka solmujen lukumäärä on n ja solmujoukko on järjestetty. Olkoon lisäksi P $n \times n$-matriisi, jonka alkiot määräytyvät seuraavasti

$$p_{ij} = \begin{cases} 1, & \text{mikäli on olemassa polku } v_i\text{:stä } v_j\text{:in} \\ 0, & \text{muutoin.} \end{cases}$$

Tällöin matriisia P nimitetään verkon G *polkumatriisiksi* (path matrix, reachability matrix).

P saadaan laskettua vierusmatriisien summamatriisista B_n asettamalla

$$p_{ij} = \begin{cases} 1, & \text{jos alkion arvo } b_{ij} > 0 \\ 0, & \text{muutoin} \end{cases}$$

Vaihtoehtoinen tapa muodostaa polkumatriisi on käyttää laskutoimituksissa suoraan Boolen

algebraa, jolloin sallitut arvot ovat 0 ja 1. Huomaa, että polkumatriisin lävistäjän alkio $p_{ii} = 1$ jos ja vain jos solmusta v_i on kaari itseensä.

$$A^2 : a_{ij}^2 = \bigvee_{k=1}^{n} a_{ik} \wedge a_{kj}$$
$$A^3 : a_{ij}^3 = \bigvee_{k=1}^{n} a_{ik}^2 \wedge a_{kj}$$
$$\dots$$
$$A^n : a_{ij}^n = \bigvee_{k=1}^{n} a_{ik}^{n-1} \wedge a_{kj}$$
$$P = A \vee A^2 \vee A^3 \vee \ldots \vee A^n$$

9 Sanastoa

9.1 Logiikan ja diskreettien rakenteiden sanastoa

English	"suomennos"	suomennos
atomic	atomaarinen	jakamaton
axiom	aksiooma	selkeästi tosi ilmaisu, peruslause, lähtökohta
connective	konnektiivi	looginen yhdysmerkki (operaatio)
contradiction	kontradiktio	lauseke joka on aina epätosi
graph	graafi	verkko
inference	päättely	päättely premisseistä johtopäätökseen
permutation	permutaatio	järjestykset
predicate	predikaatti	määre jollekin
premise	premissi	lähtökohta
propositio	propositio	väitelause, totuusarvoinen lause
quantifier	kvanttori	”määräilmaisu” (suomennos hakusessa)
satisfialble	toteutuva	lauseke joka on tosi jollakin valuaatiolla
tautology	tautologia	lauseke joka on aina tosi
valuation	valuaatio	muuttujien arvojen yhdistelmä, totuusjakauma

9.2 Logiikan ja diskreettien rakenteiden merkintöjä

Merkki	merkitys
$\wedge$	AND, JA, konjunktio, molemmat

Merkki	merkitys
$\vee$	OR, TAI, disjunktio, edes toinen
$\veebar$	XOR, JOKOTAI, exclusive-or, täsmälleen toinen, myös $\oplus$
$\neg$	EI, negaatio
$\rightarrow$	'jos, niin', implikaatio, riittävä ehto
$\leftrightarrow$	'jos ja vain jos', ekvivalenssikonnektiivi
$\Leftrightarrow$	'jos ja vain jos'
$\equiv$	ekvivalentti, sama kuin
$\models$	mallintaa, valuaatiolla lauseke on tosi; tästä seuraa
$\vdash$	päättelystä seuraa
$\cap$	joukkojen leikkaus (intersection)
$\cup$	joukkojen yhdiste (union)
$\setminus$	joukkojen erotus (difference)
$\sim$	(joukon) negaatio
$\subseteq$	osajoukko
$\subset$	aito osajoukko
$\varnothing$	tyhjä joukko
$\forall$	kaikille
$\exists$	on olemassa
$\nexists$	ei ole olemassa, myös $\neg\exists$
φ	phi, kreikk. aakkonen, käytetään symbolina logiikan kaavasta
ψ	psi, kreikk. aakkonen, käytetään symbolina logiikan kaavasta
χ	chi, kreikk. aakkonen, käytetään symbolina logiikan kaavasta
$n!$	kertoma, $1 \times 2 \times 3 \times \ldots \times n$
$P(n,k)$	k-mittaisten permutaatioiden määrä n alkioista, $n(n-1)(n-2)\ldots(n-k+1) = n!/(n-k)!$.
$\binom{n}{k}$	k-alkioisten kombinaatioiden määrä, $P(n,k)/k!$, myös $C(n,k)$
$\square$	Mikä oli todistettava.
$\mathbb{N}$	Luonnollisten lukujen joukko $\{0,1,2,3,\ldots\}$
$\mathbb{Z}_+$	Positiivisten kokonaislukujen joukko $\{1,2,3,\ldots\}$
$\mathbb{Z}$	Kokonaislukujen joukko $\{\ldots,-3,-2,-1,0,1,2,3,\ldots\}$
$\mathbb{R}$	Reaalilukujen joukko
$\bot$	Riippumaton (todennäköisyyslaskennassa)

10 Lähteitä

Grassmann, K.W., Tremblay, J. (1996). *Logic And Discrete Mathematics A Computer Science Perspective.* Prentice Hall, New Jersey.

Halonen, I. (2004). *Johdatus tieteenfilosofiaan - Tieteellinen päättely.* WWW-sivusto, http://www.helsinki.fi/hum/fil/tietfil/Luento05.htm. (9.6.2004)

Logic in Action Verkkokirja ja kalvosarja, http://www.logicinaction.org/

Ensley, D. *Discrete Math Resources*, Flash Applications, http://webspace.ship.edu/deensley/DiscreteMath/flash/. Myös kirja: Ensley, D., Crawley, W.: *Discrete Mathematics: Mathematical Reasoning and Proof with Puzzles, Patterns and Games.*