TIEP1020 Diskreetit rakenteet

Antti Valmari

Jyväskylän yliopisto
Informaatioteknologian tiedekunta

	Symboleita	1
1	Johdanto	6
2	Syntaksi	24
3	Logiikka yhtälöiden yms. ratkaisemisessa	50
4	Propositiologiikka	86
5	Predikaattilogiikka	153
6	Modulaarinen aritmetiikka	170
7	Päättelyesimerkkejä	179
	lisää tulee ...	

Symboleita

Logiikan symboleita

$\neg$	87	negaatio eli "ei"
$\wedge$	87	konjunktio eli "ja"
$\vee$	87	disjunktio eli "tai"
$\rightarrow$	87	totuusarvojen implikaatio eli "jos ... niin"
$\leftrightarrow$	87	totuusarvojen ekvivalenssi eli "jos ja vain jos"
F	87	totuusarvo "epätosi" (false)
T	87	totuusarvo "tosi" (true)
U	136	totuusarvo "määrittelemätön" (undefined)
$\Rightarrow$	144	päättelyimplikaatio eli "jos ... niin"
$\Leftrightarrow$	144	päättelyekvivalenssi eli "jos ja vain jos"
$\equiv$	144	(tällä kurssilla) sama totuusarvo
$\forall$	154	universaalikvanttori eli "jokaisella"
$\exists$	154	eksistenssikvanttori eli "on olemassa"
$\lfloor \cdots \rceil$	139	(tällä kurssilla) "on määritelty"

&&	5	"ja"-operaattori, jonka oikea puoli lasketaan vain jos vasen tuottaa **T**
||	5	"tai"-operaattori, jonka oikea puoli lasketaan vain jos vasen tuottaa **F**

Muita symboleita

$::=$	38	BNF:n "määritellään"-symboli
ε	38	BNF:n "ei mitään"-symboli (kreikkalainen kirjain pieni epsilon)
div	77	kokonaislukuosamäärä, esim. $123 \operatorname{div} 5 = 24$
mod	77	jakojäännös, esim. $123 \bmod 5 = 3$
syt	174	suurin yhteinen tekijä
$=_M$	171	(tällä kurssilla) yhtäsuuruus modulo M eli $n \bmod M = m \bmod M$
$\div_M$	173	(tällä kurssilla) kertolaskun käänteistoiminto aritmetiikassa modulo M
$\mathbb{N}$		$\{0, 1, 2, \ldots\}$ luonnolliset luvut
$\mathbb{Z}$		$\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ kokonaisluvut
$\mathbb{Z}^+$		$\{1, 2, 3, \ldots\}$ positiiviset kokonaisluvut
$\mathbb{Z}^-$		$\{\ldots, -3, -2, -1\}$ negatiiviset kokonaisluvut

$\mathbb{Q}$	$\{\frac{n}{m} \mid n \in \mathbb{Z} \wedge m \in \mathbb{Z}^+\}$	rationaaliluvut
$\mathbb{Q}^+$	$\{x \in \mathbb{Q} \mid x > 0\}$	positiiviset rationaaliluvut
$\mathbb{Q}^-$	$\{x \in \mathbb{Q} \mid x < 0\}$	negatiiviset rationaaliluvut
$\mathbb{R}$	reaaliluvut (kaikki lukusuoran luvut)	
$\mathbb{R}^+$	$\{x \in \mathbb{R} \mid x > 0\}$	positiiviset reaaliluvut
$\mathbb{R}^-$	$\{x \in \mathbb{R} \mid x < 0\}$	negatiiviset reaaliluvut

Pieniä kreikk. kirjaimia: α (alfa), β (beta), γ (gamma), δ (delta), ε (epsilon), λ (lambda), μ (mu), ξ (xi), π (pi), ρ (rho), σ (sigma), τ (tau), φ (phi), χ (chi), ψ (psi), ω (omega)

Kurssilla käytettävä pseudokoodi

a := lauseke	Muuttuja a saa arvokseen lausekkeen arvon.
if ehto **then** lauseita	Jos ehto on voimassa, niin suoritetaan lauseita. Muussa tapauksessa mennään jatkoon tekemättä mitään.
if ehto **then** lauseita1 **else** lauseita2	Jos ehto on voimassa, niin suoritetaan lauseita1. Muussa tapauksessa suoritetaan lauseita2.
for i := ala **to** ylä **do** lauseita	Suoritetaan lauseita ensin i:n arvolla ala, sitten ala $+ 1$, sitten ala $+ 2$ ja niin edelleen, niin kauan kuin $i \leq$ ylä. Jos aluksi ala $>$ ylä, niin mennään jatkoon tekemättä mitään.
while ehto **do** lauseita	Jos ehto ei ole voimassa, niin mennään jatkoon. Muussa tapauksessa suoritetaan lauseita ja palataan kokeilemaan onko ehto voimassa. Jos ehto ei tule koskaan voimaan, niin lauseita suoritetaan uudelleen ja uudelleen loputtomasti.
lause1 ; lause2	Puolipiste helpottaa lukemista erottamalla samalla rivillä olevat lauseet toisistaan. Se ei vaikuta toimintaan.

ehto1 && ehto2	Lasketaan ehto1. Jos se tuottaa **T** niin lasketaan ehto2 ja palautetaan sen tulos. Muussa tapauksessa palautetaan ehto1:n tulos. Kaatuu jos ja vain jos joko ehto1 kaatuu tai ehto1 tuottaa **T** ja ehto2 kaatuu. (katso myös sivu 143)
ehto1 || ehto2	Lasketaan ehto1. Jos se tuottaa **F** niin lasketaan ehto2 ja palautetaan sen tulos. Muussa tapauksessa palautetaan ehto1:n tulos. Kaatuu jos ja vain jos joko ehto1 kaatuu tai ehto1 tuottaa **F** ja ehto2 kaatuu. (katso myös sivu 143)

if-, **for**- ja **while**-lauseiden vaikutusalue osoitetaan sisentämällä.

1 Johdanto

1.1 Matematiikka ohjelmistoammateissa

Tällä kurssilla käsitellään *ohjelmistotyössä* ja *tietojenkäsittelytieteessä* tarvittavaa matematiikkaa, erityisesti logiikkaa

- vrt. sisällysluettelo
- suurin osa asioista on mukana tärkeytensä vuoksi
- itsessään vähemmän tärkeitä asioita on mukana mm. kertomassa sovelluksista tai siksi, että ne sopivat hyvin päättelytaidon kehittämiseen

Ohjelmistoprojekteilla on vahva taipumus epäonnistua

- Software Hall of Shame on tiivis luettelo monista kalliista epäonnistumisista
- kirjassa Bits and Bugs – A Scientific and Historical Review of Software Failures in Computational Science [HuN19] on runsaasti hyvin dokumentoituja tapauksia
- ”Tuhlaamme miljardeja dollareita vuosittain täysin estettävissä oleviin virheisiin” [Cha05]

Ohjelmoijien tuottavuudessa on moninkertaisia eroja

- monen mielestä tämä on tosielämässä ilmeistä
- tieteellinen kirjallisuus tukee väitettä ainakin jossain määrin
- väite on pyritty osoittamaan urbaaniksi legendaksi, mutta ei vakuuttavasti
- kirjallisuusviitteitä, keskustelua ym.: [McC11]
- väitteestä ei seuraa, että pitää palkata vain huippuja
- väitteestä kyllä seuraa, että ohjelmoinnin opetusta ja opiskelua kannattaa yrittää muuttaa paremmaksi

Päätavoite: hyvien tietojärjestelmien suunnittelu ja toteutus

- saivartelevasti toimivat tietokoneet täytyy saada toteuttamaan käyttäjien tarpeet
- mitä se tarkoittaa?

Tapahtui 13.2.2017 Helsingissä Malmin Prisman kassajonossa [Lai17]

- pyörätuolissa istuva mies halusi ostaa saappaat, joiden hinta oli 74,50 €
- hänellä oli Kelan maksusitoumus 70 € saakka, ja hän halusi maksaa loput 4,50 € itse
- Kassa kieltäytyi:

  ”Maksusitoumuksella maksettaessa tuote ei saa maksaa yli sitoumuksen summan verran.”

- paikalle kutsuttiin vuorovastaava ja asia selitettiin hänelle
  - ei onnistu
- paikalle tuli vartija, mies itki
- jonossa seuraavana ollut nainen maksoi saappaat omista rahoistaan
- Malmin Prisman johtaja jälkikäteen:

  > ”Kassoilla ja infossa on ollut kirjalliset ohjeet, miten tällaisissa tilanteissa toimitaan. Ensimmäisenä niissä lukee, että maksusitoumuksesta pitää tarkastaa rahasumma, eikä summaa saa ylittää. Virhe on tullut tässä.”

- tarkoitus ei ollut kieltää maksamasta ylimenevää summaa erikseen
  - sen kirjoitettu ohje kuitenkin teki
  - sen mukaan kassahenkilö ja vuoropäällikkö toimivat

  > ”Se on ihan selkeästi meidän oma virhe ja ohjeistuksen virhe.”

- kauppa hyvitti summan naiselle ja jäi selvittämään, kuka mies oli
- kauppa joutui silti someraivon kohteeksi [Kar17]
- miten ohje olisi pitänyt muotoilla?

Tapaukseen ei liittynyt tietokoneita, joten miten se koskee ohjelmointia?

- hyvin monen ohjelmistosotkun syy on samanlainen
- on erittäin vaikeaa ajoissa ennakoida, mitä ohjeista, spesifikaatiosta tms. seuraa

- kukaan ei näe tällaisia virheitä etukäteen, paitsi ehkä joku nörtti
- jos nörtti varoittaa virheestä, häntä pidetään saivartelijana
- ohjelmistoammattilaisten pitää siis
  - ymmärtää käyttäjien näkökulma ja
  - osata ennakoida vaatimusten ja suunnitelmien loogiset seuraukset

Täsmällisen määrittelemisen taito on hyvin tärkeä!

- opitaan harjoittelemalla, aloittamalla sopivan helpoista tehtävistä
- esim. esiintyykö ana kerran vai kahdesti sanassa ananas?
- esim. autojonon viimeisen mustan auton jälkeen tulee vain punaisia autoja
  - sin mus val mus pun pun toteuttaa ja sin mus val mus pun sin rikkoo
  - entä jos viimeinen auto on musta?
  - entä jos jonossa ei ole mustia autoja?

Kyky katsoa asioita loppukäyttäjän näkökulmasta on hyvin tärkeä!

- käyttäjän ei voi odottaa osaavan analysoida ja ilmaista selvästi mitä tarvitsee

$\Rightarrow$ ohjelmistoammattilaisen tulee toimia välittäjänä loppukäyttäjien inhimillisen maailman ja tietokoneiden saivartelevan maailman välillä

Päättelytaito on hyvin tärkeä!

- sekin opitaan harjoittelemalla, aloittamalla sopivan helpoista tehtävistä

Sama tieto voidaan esittää tietokoneessa monin eri tavoin

- jotkin ovat ihmiselle luontevia
- jotkin mahdollistavat tehokkaan käsittelyn
- esim. viikkolukujärjestys + poikkeukset vs. oppimistapahtumien luettelo

$\Rightarrow$ on hyvin tärkeää pohtia vaihtoehtoisia esitystapoja!

Hyvien spesifikaatioiden laatimisen opettamisesta

- suoria tehokkaita keinoja ei tunneta
  - opettajien määrä ja kyvyt eivät riitä löytämään virheet sanallisista harjoitustöistä
- ohjelmoinnin ja ongelmanratkaisun opettaminen auttaa välillisesti
  - kone saadaan etsimään virheet
  - kun ajattelutapa on opittu jossakin, sitä voi soveltaa muuallakin
- parhaiten tässä toimii melko vaativa ohjelmointi

Mitä matematiikkaa?

- yleisesti ajatellaan ohjelmistoalalla tarvittavan matemaattista ajattelua
- silti moni ohjelmistoammattilainen julistaa että ei ole tarvinnut matematiikkaa
- tieteellinen kirjallisuus yms. on seuraavien lainausten suuntaista
  (lisää löytyy ainakin [KB+05, PuA09, Sur07])

- [Let00]: "*Because of the low importance and high forgetability of continuous mathematics and basic science, universities and colleges should either place less emphasis on these topics or they should teach them in a way that makes them more relevant to software engineering students.*"
- [ACM13]: "*We recognize that general facility with mathematics is an important requirement for all CS students. Still, . . . For example, an understanding of linear algebra plays a critical role in some areas of computing such as graphics and the analysis of graph algorithms. However, linear algebra would not necessarily be a requirement for all areas of computing (indeed, many high quality CS programs do not have an explicit linear algebra requirement). . . . we note that undergraduate CS students need enough mathematical maturity to have the basis on which to then build CS-specific mathematics . . . , which, importantly, does not explicitly require any significant college-level coursework in calculus, differential equations, or linear algebra.*"

$\Rightarrow$ ohjelmoijien matematiikkaa on olemassa, mutta perinteinen paketti ei ole sitä

- tärkeintä on osata ajatella käyttäjien näkökulmaa matemaattis-loogisesti
  - käyttäjien tarpeiden tunnistaminen ja muotoileminen riittävän täsmällisiksi
  - käyttäjien ilmoittamissa tarpeissa olevien ristiriitaisuuksien ja epätarkoituksenmukaisuuksien havaitseminen
  - vaatimusten ja suunnitelmien seurausten ennakointi
  - ongelmanratkaisutaito

Pohjana kurssin suunnittelulle olivat

- ACM Computing Curriculum: Computer Science [ACM13]
  - perusteellisesti ja pitkäjänteisesti mietitty kansainvälinen näkemys, mitä CS-tutkintojen tulee vähintään sisältää
  - maailmalla laajassa käytössä
- paikka opetusohjelmassa
  - esitiedot: lyhyt matematiikka
- opettajan laaja kokemus ohjelmoinnista ja teoreettisesta tietojenkäsittelytieteestä
- kollegojen neuvot
- MathCheckin tarjoamat mahdollisuudet opiskelun tukena
- kahden viime vuoden kokemukset

MathCheck

- ohjelma, joka tarkastaa ratkaisuja vaihe vaiheelta, ei pelkästään lopputulosta
  $\Rightarrow$ mahdollistaa mm. ratkaisun korjaamisen omatoimisesti
- sisältää tämän kurssin kannalta hyödyllisiä tehtävälajeja, joita opetusohjelmissa ei tavallisesti ole
  - syntaksitehtävät
  - logiikkatehtävät
- keskeneräinen
  $\Rightarrow$ kaikki ei toimi parhaalla mahdollisella tavalla

1.2 Matemaattisten asioiden oppimisesta ja ongelmanratkaisutaidosta

”Kestäviä, käyttökelpoisia luokkahuoneopetuksen teorioita ei vielä ole” [HiG07, s. 373]

On kuitenkin olemassa hyväksi havaittuja periaatteita

- [AnW09] on harkittu ytimekäs kokoelma

Matematiikan opiskelun tavoitteita [KSF01 luku 4, AnW09 s. 6]

- **käsitteellinen ymmärtäminen** (conceptual understanding)
- **menetelmällinen sujuvuus** eli rutiinilaskutaito (procedural fluency)
- **strateginen kyvykkyys** (strategic competence)
- **tilanteen mukainen päättelytaito** (adaptive reasoning)
- **usko omiin kykyihin oppia ja hyödyntää matematiikkaa** (productive disposition)

  ”Matematiikan oppimisen pitäisi tarkoittaa tien löytämistä uudentyyppisten tehtävien läpi ja vastuun ottamista tuloksista” [Hassler Whitney, ehkä 1985]
- nämä viisi kietoutuvat toisiinsa
  - niitä ei opiskella peräkkäin, vaan kaikkia yhtäaikaa

Tavoitellaan ongelmanratkaisutaitoa

- ongelmia eri alueilta: matematiikka, ohjelmointi, sivuaineen valinta, ilmastonmuutokseen sopeutuminen, . . .

- oikea vastaus ei ole maali, vaan välitavoite
- vastauksen löytämisprosessi on tärkeä
- kyky tarkastaa vastaus itse on tärkeä
  - jos speksaan näin, sujuuko tenttitulosten syöttäminen sisään
- kyky perustella vastaus muille on tärkeä
- tiedonhakutaito auttaa vain, jos pohja on kunnossa
  - jos et usko, niin lue vaikka Wikipedian sivu Knuth-Morrison-Pratt–algoritmista

Tavoitellaan pitkäkestoista osaamista

- tavoitellaan taitoja, joista on hyötyä työelämässä

$\Rightarrow$ niiden tulee säilyä sinne asti

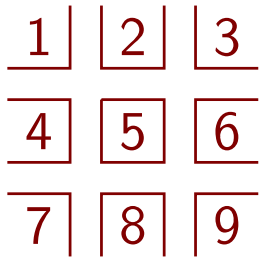
- kurssilla voidaan testata vain säilyykö tenttiin asti, mutta sekin on parempi kuin että säilyy vain seuraavan päivän demotilaisuuteen asti

Käsitteellinen ymmärtäminen muuttaa vaikeasti muistettavia luetteloita helpoiksi muistaa

- esim. onko helppo muistaa tämä salakirjoitus? (videolta C. Brabrand ja J. Andersen: Teaching Teaching & Understanding Understanding, 2006)

- salakirjoitus on muodostettu numeroruudukosta kuvien osoittamalla tavalla

1	2	3
4	5	6
7	8	9



Ilman käsitteellistä ymmärtämistä asioiden määrä tuntuu valtavalta

- esimerkiksi erilaisia prosenttilaskutilanteita on paljon
  - ne kaikki on helposti johdettavissa muutamasta periaatteesta
- esimerkiksi [ACM13] nimeää (jos laskin oikein) yhteensä melkein 500 pakollista ja "näistä ainakin 90 % otettava" melko isoa aihetta
  - esim. "funktiot ja parametrinvälitys"
  - esim. "propositiologiikan päättelysäännöt"
- aivan liikaa muistettavaksi, mutta realistinen opittavaksi ymmärtämisen kautta

Intuitio vastaan formaali päättely

- intuitio tarkoittaa vaistoa, luovuutta, ratkaisu vain putkahtaa mieleen
- matematiikkaa pidetään tiukan täsmällisenä tieteenä
  - "minusta tuntuu" ei kelpaa perusteluksi
- periaate: ratkaisut löydetään intuitiolla ja tarkastetaan tarkalla päättelyllä

- ratkaisuvaiheessa kaikki oma työ mikä vie eteenpäin on sallittua
  - villit arvaukset
  - kokeileminen yksittäisillä lukuarvoilla
  - kuvien piirto
  - ...
- tulokset on esitettävä siten, että muut voivat tarkastaa ne
  - usein tämä edellyttää suurta uudelleenkirjoittamista
  - usein tätä tehdessä ratkaisu paljastuu puutteelliseksi

  $\Rightarrow$ sitä joutuu täydentämään tai korjaamaan
- opettaja saattaa pedagogisista syistä haluta nähdä myös ratkaisuprosessia
  - arvaaminen on sallittua, jos arvattu tulos todistetaan (esim. yhtälön juuren voi todistaa sijoittamalla yhtälöön)
  - ei kuitenkaan ole uskottavaa, että juuri löytyi arvaamalla, jos se on $\dfrac{13-5\sqrt{7}}{6}$
- itsestäänselvyyksiä ei tarvitse perustella
  - opettaja voi olla eri mieltä siitä, onko jokin itsestään selvää
  - aikaa myöten kehittyy kyky tunnistaa, mikä on itsestään selvää muiden mielestä

  $\Rightarrow$ yritä rohkeasti ja hyväksy se, että ratkaisua voidaan pyytää parantamaan

Kaksi avainasiaa käsitteellisen ymmärtämisen saavuttamiseen [HiG07 s. 383–392]

- teachers and students attend explicitly to concepts
- students struggle with important mathematics

Rutiinilaskutaito

- jos rutiinivaiheista ei selviä sujuvasti, niin työstä ei tule mitään
  - ratkaisua ei voi suunnitella, jos ei tiedä, millaisia osia voi käyttää
  - ratkaisua ei voi toteuttaa, jos ei osaa toteuttaa sen osia
  - isojen asioiden miettimistä häiritsee, jos on mietittävä myös pikkuasioita
- vertaa vieraankielisen tekstin lukemiseen, jos osaa vain vähän sanoja
  - jokaisen sanan voi periaatteessa katsoa jostain
  - käytännössä se hidastaa ja häiritsee lukemista liikaa

$\Rightarrow$ monet perusasiat pitää hioa niin että vastaus löytyy sujuvasti

  - yleensä ei riitä, että tuo demoon kotona löydetyn ratkaisun, vaan tehtävä pitää pystyä ratkaisemaan demossa uudelleen
- käsitteellinen ymmärtäminen tukee rutiinilaskutaidon saavuttamista [HiG07 s. 390]
  - esim. auttaa keksimään oikoteitä ja päättelemään, ovatko ne päteviä
  - esim. kun laskee saman useasti ymmärryksen kautta, lopulta oppii sen ulkoa
- numerotaito auttaa varsinkin käytännöllisessä ongelmanratkaisussa
  - kyky arvioida suuruusluokka päässä

Vaikeiden asioiden omaksuminen vaatii sekä työtunteja että kalenteriaikaa

- yön yli nukkuminen todellakin usein auttaa
- pitää työskennellä ahkerasti kurssin alusta saakka, ei vain juuri ennen tenttiä
- tentti ei saa olla liian pian luentojen jälkeen

Jälkityöskentely on erityisen tehokas tapa oppia

- ratkaisun kanssa työskenteleminen sen jälkeen kun se on löydetty
  - muokkaaminen helposti tarkastettavaan muotoon
  - sovellusten miettiminen
  - yleistysten miettiminen

Valitettavasti työhön pitää jonkin verran pakottaa

- jollei ole määräaikoja pitkin kurssia, niin työskennellään vain tentin alla [Gib10]
- "Opiskelijapolvet joutuvat kärsimään edeltäjiensä synneistä." [Terhi Kilamo 2019]
  - moni ikävä sääntö on syntynyt siten, että yritettiin ilman, mutta se ei toiminut
- "Pakko on rajallinen luonnonvara. Sitä pitää käyttää harkiten." [A. Valmari $\approx$ 2017]

Kurssin aikana pitää antaa palautetta mutta ei pisteitä, kurssin jälkeen toisinpäin [Gib10]

- palautetyöskentely on erityisen tehokas tapa oppia
- palaute tentin jälkeen tutkitusti ei kiinnosta
- palaute viikkotehtävänkään jälkeen ei kiinnosta, jos opiskelija suhtautuu tehtävään onnistuneena tai menetettynä mahdollisuutena saada piste

$\Rightarrow$ opiskelija täytyy saada suhtautumaan palautteeseen mahdollisuutena saavuttaa jotain hyötyä *tulevaisuudessa*

- tentissä
- työelämässä

Usko omiin kykyihin

- ”kaikki opiskelijat, iästä riippumatta, voivat kehittää myönteisen matemaattisen identiteetin ja tulla tehokkaiksi matemaattisiksi oppijoiksi” [AnW09, s. 6]
- tehtävät eivät saa olla liian helppoja eivätkä liian vaikeita
  - tiedän, että kaikki eivät oppineet koulumatematiikasta läheskään kaikkea

  $\Rightarrow$ yritän järjestää paljon tehtäviä koulumatematiikastakin

Sopivan tasoiset neuvot

- opiskelijaa ei saa jättää pulaan
- liian tarkat neuvot poistavat opiskelijalta sen työn, josta oppii eniten

$\Rightarrow$ kysy rohkeasti, mutta vastaus ei välttämättä ole suora vastaus vaan vinkki

Älä keskity yksityiskohtien ulkoaopetteluun, vaan niiden taustalla olevien periaatteiden ymmärtämiseen!

Tehtäviä tehdään siksi, että opittaisiin *jatkossa tärkeitä* taitoja tai asioita!

- tehtävän aihe voi olla ihan turha, mutta oppimistavoite ei ole
- jos bluffaa alkuvaiheen tehtävissä, niin loppuvaiheen tehtävistä tulee ylivoimaisia
- ”tentin (tai harjoituksen) jälkeen voin unohtaa” on huono asenne

1.3 Kurssin suorittaminen

Kurssin suoritustapa

- harjoituksilla hankitaan lupa tulla tenttiin
  - kerättävä vähintään minimipistemäärä riittävän pitkän ajan kuluessa
  - ilman tätä tentissä olisi lukuisia valmistautumattomia
  - minimi tuskin antaa riittävän osaamisen $\Rightarrow$ tee enemmän 
- pisteiden saanti perustuu suurelta osin opiskelijan omaan ilmoitukseen
  - opiskelija tietää oliko vastaus oikein MathCheckin palautteesta
  - opettajille palautetaan vain pieni osa vastauksista
  - pisteiden jakamisessa ei niuhoteta (mutta pisterajasta pidetään kiinni)
- arvosana määräytyy yksinomaan tentin perusteella
- olen kokeillut vuosien saatossa monenlaisia harjoituspistejärjestelmiä
  - kannustusvaikutus on vaikea saada juuri oikeaksi
  - päähuomio ei saa ohjautua siihen, onko vai eikö jokin vastaus pisteen arvoinen
  - on mahdotonta selvittää, missä määrin opiskelija teki kotitehtävät itse
  - ei ole hyvä kannustaa unohtamaan asia heti kun siitä on pisteet saatu!
- MathCheck mahdollistaa omatoimisen harjoittelun
  $\Rightarrow$ ehdotus: harjoittele, kunnes koet osaavasi riittävästi!
  - toki opettajat ymmärtävät, että kaikki eivät tähtää arvosanaan 5 (juuri siksi aineistossa on helppoja ja vaikeita osuuksia)

- opettajat pyrkivät siihen, että työmäärä olisi oikea
  - 5 op on virallisesti 133 tuntia mediaaniopiskelijan työtä kaikki työ huomioiden
  - koska tämä periodi on pitkä, se tarkoittaa n. 13 tuntia / viikko
  - samaan lukuun päädytään siitä, että 60 op vuodessa vastaa 15 op periodissa

  $\Rightarrow$ 5 op vastaa kolmasosaa työajasta, ja kolmasosa viikkotyöajasta on noin 13 tuntia

Yksi tehtävä tentissä tulee olemaan esseetehtävä (painoarvo 10% . . . 15%)

- aiheen pitää liittyä kurssin teemaan
- essee ei saa olla suora kopio kurssin aineistosta
- saat valita aiheesi ihan itse (ym. rajauksella)
- voit valmistella esseetä etukäteen, mutta se pitää tuoda tenttiin päässä eikä paperilla
- pisteitä voi saada proosatekstilläkin, mutta täydet pisteet vaativat matematiikkaa
- sopiva pituus on puolesta sivusta yhteen sivuun

Taustatiedot

- ei oleteta, että muita matematiikan kursseja on jo suoritettu
- oletetaan, että opiskelija tarvittaessa ottaa eri asenteen kuin lukiossa
  - tavoitteena on asioiden *ymmärtäminen*, ei mekaaninen tekeminen
- ohjelmointiosaamisesta on paljon hyötyä, mutta välttämätöntä se ei ole

Kurssi pyrkii hyödyntämään omaa sisältöään sekä etu- että jälkikäteen

- esim. kielioppeja hyödynnetään logiikan kielten esittelemisessä
- monenlaisia päättelyitä on pitkin matkaa
- jotta kotitehtävät eivät olisi yhtä ja samaa, niissä käsitellään yhtäaikaa paria kolmea teemaa
- yhteenliittyvät asiat on kuitenkin koottu luentoruuduissa yhteen

$\Rightarrow$ luennot eivät etene täysin sivunumerojärjestyksessä

Esityksessä käytettävä värikoodaus

- erityisen tärkeitä asioita on osoitettu keltaisella taustalla
  - opiskele ainakin ne
  - usein niiden ymmärtäminen edellyttää myös ympäröiviä asioita
- himmeänkeltainen tausta osoittaa keskitärkeitä asioita
  - pieni paino tentissä
  - arvosanoihin 3, 4 ja 5 tähtääville
  - arvosanaan 5 tähtäävän kannattaa opiskella myös korostamattomat asiat
- *johtopäätöksiä yms. tärkeää* on korostettu violeteilla vinoilla kirjaimilla
- **määriteltävä sana** tai **käsite** on vahvennettuna sinisellä
  - ei välttämättä tärkeä, katso taustan tai ympäröivän tekstin väri

- esimerkeissä käytetään usein ruskeaa
  - esimerkkejä ei kysytä tentissä, vaikka olisivat mustallakin
  - esimerkit ovat vain asioiden ymmärtämisen tueksi
- jokin asia on osoitettu **hyväksi** tai **huonoksi** vihreällä tai punaisella
- harmaalla kirjoitettua ei varmasti kysytä tentissä
  - harmaata käytetään, kun asia muuten näyttäisi tentissä kysyttävältä

2 Syntaksi

Tässä luvussa käsitellään tekstuaalisiin ilmauksiin liittyviä rakenteita

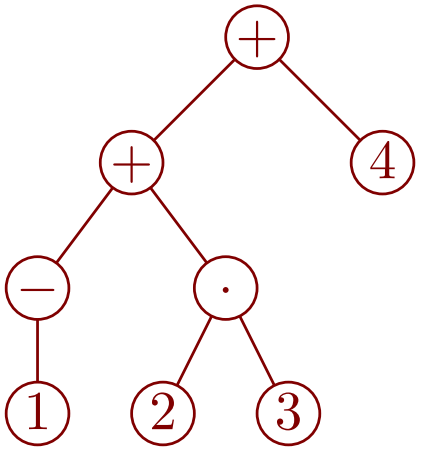
- miten ne esittävät hierarkkisia rakenteita
- miten ne itse voidaan määritellä hierakkisina rakenteina

2.1 Lausekkeista

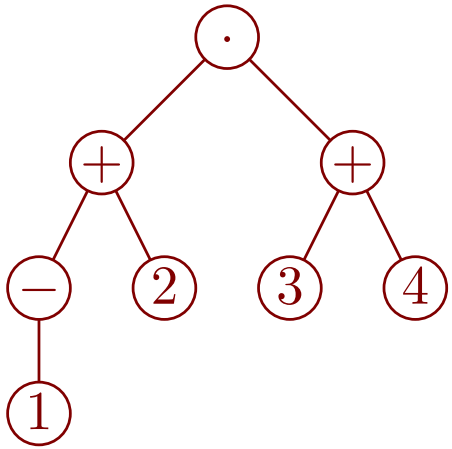
Lausekepuu

- tietokonekielissä lauseke on tekstuaalinen keino ilmaista puumainen rakenne

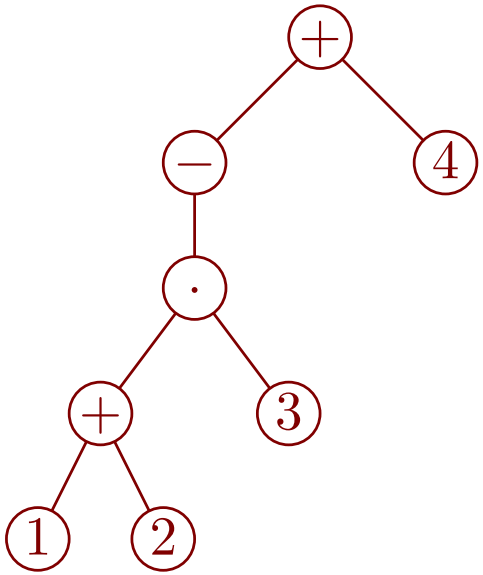
$-1 + 2 \cdot 3 + 4$



$(-1 + 2) \cdot (3 + 4)$



$-((1 + 2) \cdot 3) + 4$



- tällainen puu on **lausekepuu**

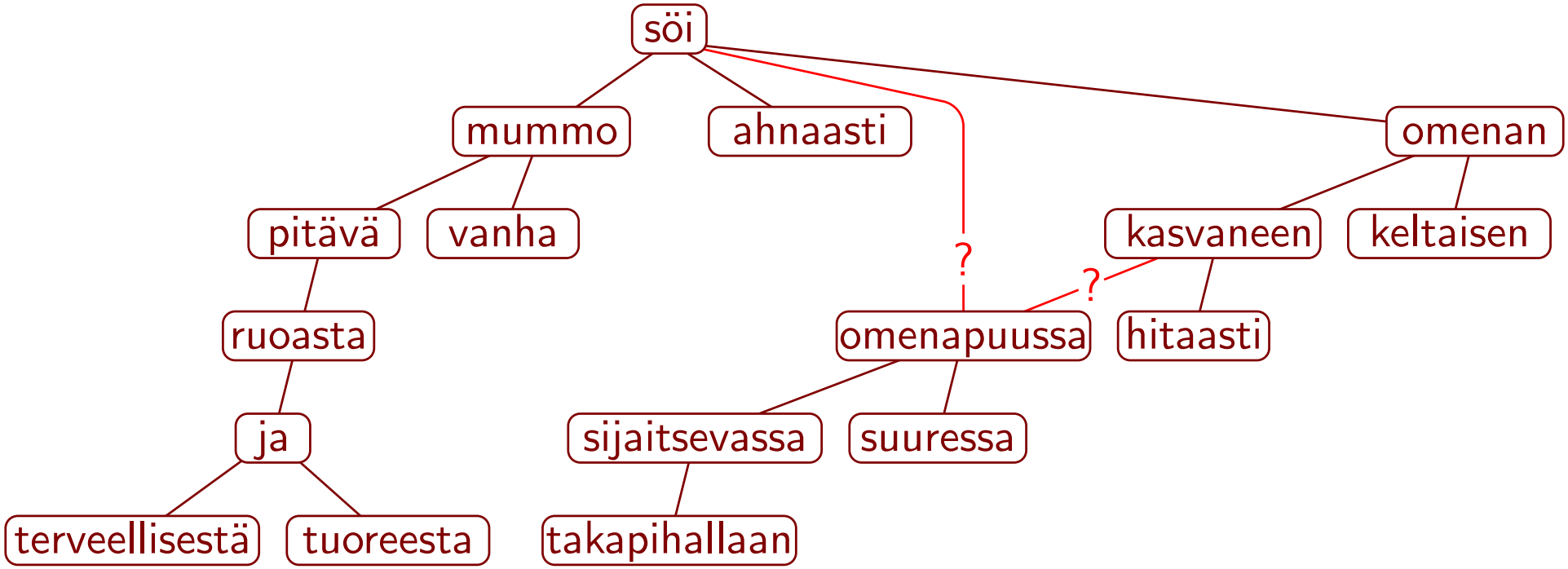
Piirrettyinä puut vievät paljon tilaa ja piirtäminen on työlästä

$\Rightarrow$ käytetään tiiviimpiä esitystapoja
  - matemaattiset merkinnät
  - ohjelmointikielten lausekkeet

- puumainen rakenne voidaan aina esittää siten, että lehdet esitetään sellaisinaan ja muut solmut muodossa (sisältö alipuu ... alipuu)
  - esim. $(/\ (\pm\ (-\ b)\ (\sqrt{}\ (-\ (\hat{}\ b\ 2)\ (\cdot\ (\cdot\ 4\ a)\ c))))\ (\cdot\ 2\ a))$
  - käytössä esim. LISP-kielessä
- matematiikassa on kuitenkin yleistä
  - sijoittaa solmun sisältö keskelle, jos solmulla on kaksi alipuuta
    $(((-\ b)\ \pm\ (\sqrt{}\ ((b\ \hat{}\ 2)\ -\ ((4\ \cdot\ a)\ \cdot\ c))))\ /\ (2\ \cdot\ a))$
  - jättää kertolaskut merkitsemättä
    $(((-\ b)\ \pm\ (\sqrt{}\ ((b\ \hat{}\ 2)\ -\ ((4\ a)\ c))))\ /\ (2\ a))$
  - osoittaa joidenkin alipuiden rajat muilla keinoilla kuin suluilla
    $$\frac{(-\ b)\ \pm\ \sqrt{(b)\ ^2\ -\ ((4\ a)\ c)}}{2\ a}$$
  - käyttää välejä jossain määrin puurakenteen mukaisesti $\dfrac{(-b) \pm \sqrt{b^2 - ((4a)c)}}{2a}$
  - sopia tulkinnasta, jos sulkuja on jätetty pois $\dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
- monissa ohjelmointikielissä tehdään samoin, paitsi että tekstin rivit rikkovia keinoja ei ole käytettävissä

Luonnollisessakin kielessä on hierarkkisia rakenteita

terveellisestä ja tuoreesta ruoasta pitävä vanha mummo
söi ahnaasti takapihallaan sijaitsevassa suuressa omenapuussa
hitaasti kasvaneen keltaisen omenan



- joskus ne eivät jäsenny tarkoitetusti

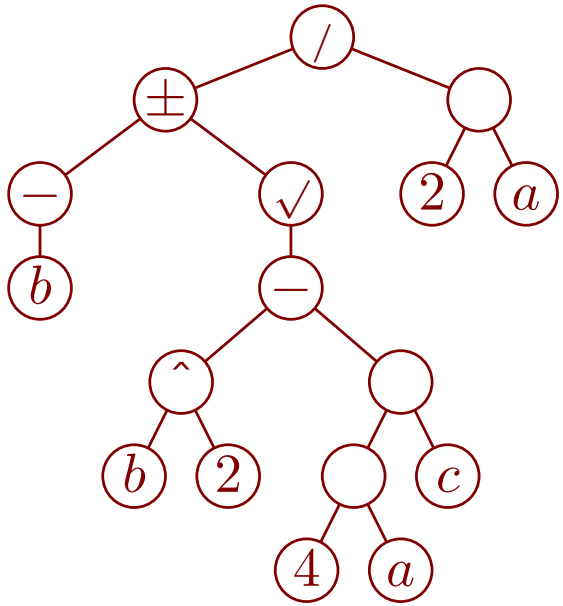
  Molemmat miehet vietiin putkaan. Putkassa
  vuonna 1954 syntynyt mies jatkoi riehumista.

- "Peppi Pitkätossu söi omenapuussa ruisleivän" ei olisi kummallista
  $\Rightarrow$ on periaatteessa mahdollista, että edellä "omenapuussa"
  ei liitykään omenaan vaan syömispaikkaan

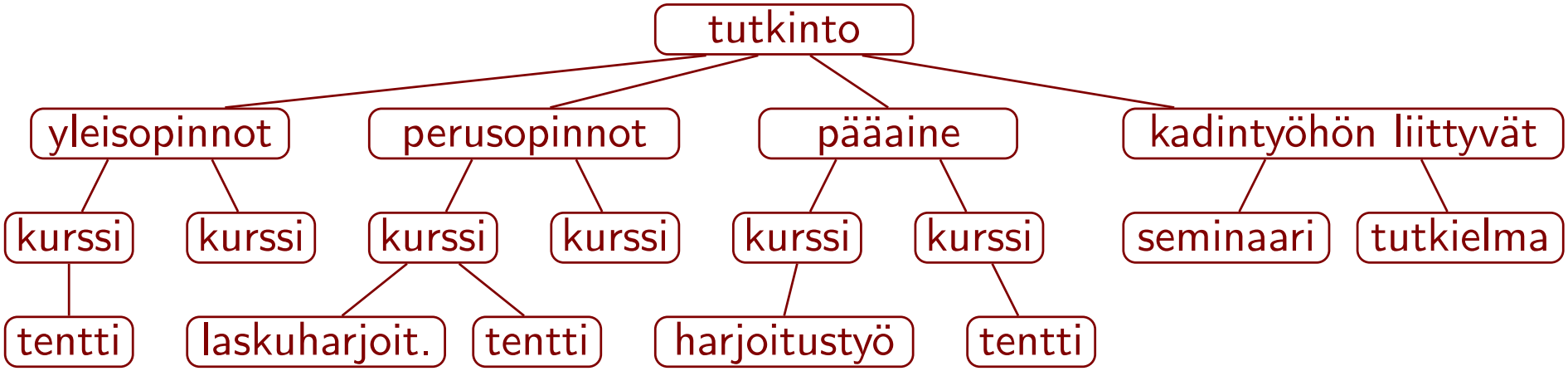
Lauseke (expression)

- kuten todettiin, lauseke esittää puumaisen rakenteen

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



- ilmaisee esim.
  - miten arvo lasketaan
  - miten kokonaisuus muodostuu osistaan
- esimerkki kokonaisuuden muodostumisesta osistaan: yksinkertaistettu tutkinnon rakenne



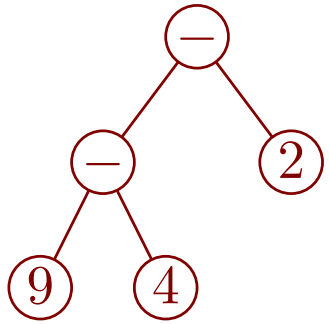
- lausekepuu ei sisällä sulkuja
  - sulut eivät ole osa varsinaista asiasisältöä, vaan keino ilmaista rakennetta
  - lausekepuussa sitä keinoa ei tarvita

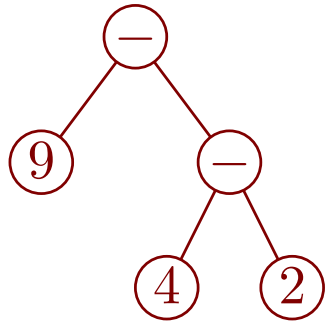
$\Rightarrow$ tentissä älä tee suluista solmua lausekepuuhun

Lausekepuu on eri asia kuin lausekkeen merkitys

- näissä esimerkeissä lausekkeen merkitys on lausekkeen tuottama tulos
  - yleisemmin matemaattisen lausekkeen merkitykseksi ajatellaan lausekkeen esittämä funktio
- $(9-4)-2=5-2=3$ mutta $9-(4-2)=9-2=7\neq 3$

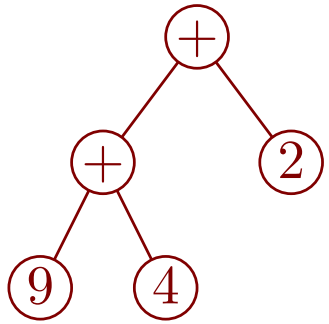
$\Rightarrow$ lausekkeet $(9-4)-2$ ja $9-(4-2)$ tuottavat eri arvon, ja siksi niillä on välttämättä eri rakenne

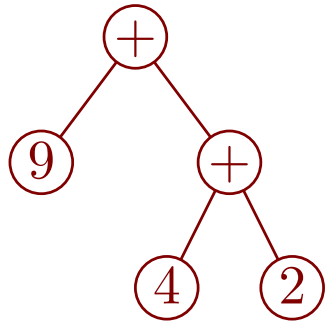




- lausekkeet $(9+4)+2$ ja $9+(4+2)$ tuottavat saman arvon, mutta niillä on silti eri rakenne

$\Rightarrow$ niitä vastaavat lausekepuut ovat erit, vaikka ne tuottavat saman arvon





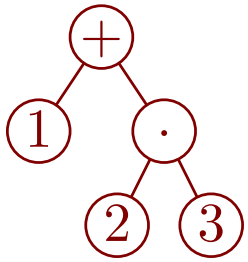
- myös lausekkeet 5 ja $5+0$ tuottavat saman arvon, vaikka niillä on eri rakenne (ja eri lausekepuut)
- lausekepuu ottaa kantaa vain lausekkeen rakenteeseen, ei merkitykseen
  - $+$ ja $-$ ovat sille eri symboleita
  - se ei välitä mitä ne tarkoittavat

  $\Rightarrow$ se ei "tiedä" että $(9+4)+2=9+(4+2)$, eikä piittaa siitä
- samoilla lausekepuilla on välttämättä sama merkitys, mutta eri lausekepuilla voi olla sama tai eri merkitys

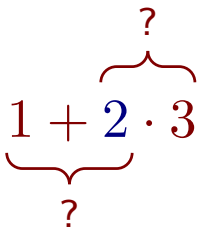
Lauseke ilman sulkuja tuottaa saman lausekepuun kuin jokin sen sulutettu versio

- $9 - 4 + 2$ tuottaa saman lausekepuun kuin $(9 - 4) + 2$
- $9 + 4 + 2$ tuottaa saman lausekepuun kuin $(9 + 4) + 2$
- $9 + 4 + 2$ tuottaa eri lausekepuun kuin $9 + (4 + 2)$, vaikka niillä on sama merkitys
- mitä versiota suluton lauseke vastaa ilmaistaan usein sitovuustasojen ja sitovuuden suunnan avulla
  - seuraavaksi tarkastelemme niitä

Sitovuustasot (precedence)



- matematiikassa on käytäntö, että kertolasku sitoo voimakkaammin kuin yhteenlasku
  - lausekkeilla $1 + 2 \cdot 3$ ja $1 + (2 \cdot 3)$ on sama lausekepuu, joka on eri kuin lausekkeen $(1 + 2) \cdot 3$ lausekepuu
- kielessä voi olla lukuisia operaattoreita ryhmiteltyinä sitovuustasoille
  - sitoo voimakkaammin = korkeampi sitovuustaso
  - jos kaksi operaattoria kilpailee välissään olevasta kohteesta, se liitetään voimakkaammin sitovaan:



- esim. C++ [Stroustrup 1997]: 68 operaattoria, 18 sitovuustasoa

. [] ++ -- ...
++ -- ! - + ...

* / %
+ -
<< >>
< <= > >=

Sitovuuden suunta

- sitovuustasot eivät ratkaise tilannetta, jos kohteen molemmin puolin on sama operaattori
  - onko $8 - 5 - 2$ sama kuin $8 - (5 - 2) = 8 - 3 = 5$
    vai sama kuin $(8 - 5) - 2 = 3 - 2 = 1$?

$$\underbrace{8 - 5}_{?} \overbrace{- 2}^{?}$$

$\Rightarrow$ operaattorille määritellään sitovuuden suunta

- jos operaattori sitoo vasemmalle (on **left associative**), niin
  - kohde liittyy vasemmalla puolellaan olevaan operaattoriin
  - laskenta etenee vasemmalta oikealle
- jos operaattori sitoo oikealle (on **right associative**), niin päinvastoin
- yleensä operaattorit sitovat vasemmalle
- esimerkkejä oikealle sitovista
  - potenssilasku matematiikassa: 2^{1^3} on sama kuin $2^{(1^3)} = 2^1 = 2$
    eikä sama kuin $(2^1)^3 = 2^3 = 8$
  - sijoitusoperaattorit C++:ssa: `i = n = 0;`
- liitännäisyys (associativity) on eri asia
  - operaattori $\diamond$ on liitännäinen, jos aina $(x \diamond y) \diamond z = x \diamond (y \diamond z)$
  - liitännäisyys liittyy operaattorin merkitykseen eikä lausekepuun muodostamissääntöihin
  - jos operaattori on liitännäinen, niin sen sitovuuden suunnalla ei ole väliä

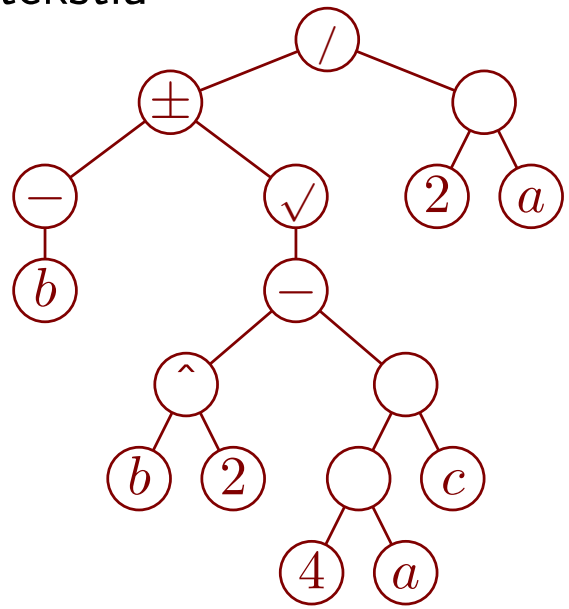
Matematiikan sitovuussäännöt eivät ole aina hyvin määritellyt tai johdonmukaiset

- esim. kaksinkertaisen kulman sini: $\sin 2x = 2 \sin x \cos x$
  - jos $\sin$ sitoo voimakkaammin, saadaan $(\sin 2)x = 2(\sin x) \cos x$
  - jos kertolasku sitoo voimakkaammin, saadaan $\sin(2x) = 2 \sin(x \cos x)$
  - tarkoitetaan $\sin(2x) = 2(\sin x) \cos x$
- esim. onko $\log(x+1)x$ sama kuin $\log\left((x+1)x\right)$ vai sama kuin $(\log(x+1))x$?
  - $\sin 2x$:n tulkinnan mukaan olisi johdonmukaista, että sama kuin $\log((x+1)x)$
  - kuitenkin jos heti funktion nimen perässä on (, niin tavallisesti argumentin katsotaan loppuvan vastaavan):n kohdalla
  - tällä tulkinnalla $\log(x+1)x$ on sama kuin $(\log(x+1))x$

Matematiikassa lauseke ei välttämättä ole yhdellä rivillä pysyvää tekstiä

esim. $\dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

- jakolasku on ilmaistu vaakaviivalla muodossa $\dfrac{\text{osoittaja}}{\text{nimittäjä}}$
  - MathCheckissä: `(osoittaja)/(nimittäjä)`
- neliöjuuren argumentin päättyminen on ilmaistu ylhäällä olevan vaakaviivan pituudella
  - vrt. $\pm\sqrt{b^2 - 4ac} - b$
  - MathCheck: `sqrt(argumentti)`



- potenssiin korotus on ilmaistu nostamalla eksponentti yläindeksiin
  - vrt. $b^2 - 4ac$ ja $b^{2-4a}c$
  - MathCheck: kantaluku^eksponentti
- $\pm$ on lyhennemerkintä, joka tarkoittaa, että on annettu kaksi lauseketta, $\frac{-b+\sqrt{b^2-4ac}}{2a}$ ja $\frac{-b-\sqrt{b^2-4ac}}{2a}$
  - se on monikäsitteinen silloin, kun $\pm$ toistuu
  - esim. tarkoittaako $\pm a \pm b$ kahta lauseketta $a+b$ ja $-a-b$ vai neljää lauseketta $a+b$, $a-b$, $-a+b$ ja $-a-b$?

  $\Rightarrow$ MathCheckissä ei ole merkintää $\pm$ vaan pitää sanoa esim. $x = \frac{-b+\sqrt{b^2-4ac}}{2a} \vee x = \frac{-b-\sqrt{b^2-4ac}}{2a}$
- matemaattisetkin lausekkeet kannattaa ymmärtää lausekepuiden esitystapoina

MathCheck jättää *rakenteen kannalta* turhat sulut pois

- `(9-4) + 2` tuottaa $9-4+2$
- `9 - (4+2)` tuottaa $9-(4+2)$
- `9 + (4+2)` tuottaa $9+(4+2)$
- kovat sulut `#(` ja `#)` jäävät näkyviin, vaikka olisivat turhat
  - `#(9-4#) + 2` tuottaa $(9-4)+2$

MathCheckin syntaksi vastaa hyvin pitkälle tavallista matemaattista käytäntöä

- matematiikan jakolasku, potenssi ja neliöjuuri eivät pysy yhdellä rivillä
  $\Rightarrow$ niille jouduttiin määrittelemään toinen esitystapa
  - `/`, `^` ja `sqrt` ovat yleisiä ohjelmointikielissä $\Rightarrow$ operaattoreiksi valittiin ne
  - niille jouduttiin valitsemaan sitovuustaso tai pakollinen sulkujen käyttö
  - pakollinen sulkujen käyttö on kankeaa $\Rightarrow$ valittiin sitovuustaso
  - ohjelmointikielissä /:lla on yleensä sama sitovuustaso kuin $*$:lla
  - se osoittautui harhaanjohtavaksi, joten /:n sitovuustaso asetettiin korkeammaksi
  - käyttäjän ei tarvitse muistaa sitovuustasoja, jos hän kirjoittaa tarpeeksi sulkuja
- `/` ei sido oikealle eikä vasemmalle harhaanjohtavuusvaaran vuoksi
  - pitäisikö `x/2/3`:n tarkoittaa $\frac{\frac{x}{2}}{3}$ eli $\frac{x}{6}$ vai $\frac{x}{\frac{2}{3}}$ eli $\frac{3x}{2}$?
- matematiikassa peräkkäisyys tarkoittaa usein näkymätöntä kertolaskua, mutta ei aina
  - jos $x = 3$, niin $2x = 2 \cdot 3 = 6$, mutta 23 ei ole $2 \cdot 3$ vaan $2 \cdot 10 + 3$
  - jos $x = \frac{1}{2}$, niin $2x = 2 \cdot \frac{1}{2} = 1$, mutta $2\frac{1}{2}$ ei ole $2 \cdot \frac{1}{2}$ vaan $2 + \frac{1}{2} = \frac{5}{2} \approx 2{,}5$
  - jos $x = -1$, niin $2x = 2 \cdot (-1) = -2$, mutta $2{-}1$ ei ole $2 \cdot (-1)$ vaan 1

$\Rightarrow$ jotta ei syntyisi vääriä tulkintoja, MathCheck ei salli näkymättömässä kertolaskussa numeerisia vakioita muualla kuin alussa

- samasta syystä MathCheck toisinaan tulostaa ylimääräiseltä vaikuttavia sulkuja
- näkyvän kertolaskun ympärille tulostuu hieman väliä
  - $\sin 2 \cdot x$ näyttää tarkoittavan $(\sin 2)x$
  $\Rightarrow$ se laitettiin tarkoittamaan sitä antamalla $*$:lle alempi sitovuustaso kuin $\sin$:lle

Unaari ja binääri

- **unaarioperaattorilla** on yksi argumentti
  - esim. etumerkki--, logiikan $\neg$, etuliite-++, jälkiliite-++
- **binäärioperaattorilla** on kaksi argumenttia
  - esim. vähennyslasku--, kertolasku $\cdot$
- **ternäärioperaattorilla** on kolme argumenttia
  - esim. C++ `?:` `x < 0 ? -x : x`

Etumerkki-− on eri operaattori kuin vähennyslasku-−, vaikka kirjoitusasu on sama

2.2 Backus-Naur form

BNF eli **Backus-Naur form** on laajalti käytetty tapa määritellä syntaksi

- BNF:stä on monenlaisia muunnelmia
- esittelemme yhden, josta idea käy ilmi
- matemaatikot tuntevat saman asian nimellä **context-free grammar** eli **CFG**

Esimerkki: yksinkertaistettu ohjelmointikielen lauseke

$$
\begin{aligned}
\mathit{Lauseke} &::= \mathit{Termi} \mid \mathit{Lauseke}\ \text{"+"}\ \mathit{Termi} \mid \mathit{Lauseke}\ \text{"-"}\ \mathit{Termi} \\
\mathit{Termi} &::= \mathit{Tekijä} \mid \mathit{Termi}\ \text{"*"}\ \mathit{Tekijä} \mid \mathit{Termi}\ \text{"/"}\ \mathit{Tekijä} \\
\mathit{Tekijä} &::= \mathit{Atomi} \mid \text{"+"}\ \mathit{Atomi} \mid \text{"-"}\ \mathit{Atomi} \\
\mathit{Atomi} &::= \mathit{luku} \mid \mathit{muuttuja} \mid \text{"("}\ \mathit{Lauseke}\ \text{")"}
\end{aligned}
$$

- esim. `37`
  - `37` on *luku*, joten `37` on *Atomi*
  - `37` on *Atomi*, joten `37` on *Tekijä*
  - `37` on *Tekijä*, joten `37` on *Termi*
  - `37` on *Termi*, joten `37` on *Lauseke*
- esim. `37+37*-x`
  - `x` on *muuttuja*, joten `x` on *Atomi*
  - `x` on *Atomi*, joten `-x` on *Tekijä*
  - `37` on *Termi* ja `-x` on *Tekijä*, joten `37*-x` on *Termi*
  - `37` on *Lauseke* ja `37*-x` on *Termi*, joten `37+37*-x` on *Lauseke*

- esim. `(37+37*-x)/3`
  - `37+37*-x` on *Lauseke*, joten `(37+37*-x)` on *Atomi*
  - `(37+37*-x)` on *Atomi*, joten se on myös *Tekijä* ja *Termi*
  - `3` on *luku*, ja siksi myös *Atomi* ja *Tekijä*
  - `(37+37*-x)` on *Termi* ja `3` on *Tekijä*, joten `(37+37*-x)/3` on *Termi*
  - `(37+37*-x)/3` on *Termi*, joten se on myös *Lauseke*
- olkoon *luku* epätyhjä jono numeroita ja *muuttuja* epätyhjä jono kirjaimia
  ⇒ *Atomi* alkaa joko numerolla, kirjaimella tai `(`:lla
  - *Tekijä* alkaa `+`:lla, `-`:lla tai sillä millä *Atomi* alkaa
  ⇒ *Tekijä* alkaa numerolla, kirjaimella, `(`:lla, `+`:lla tai `-`:lla
  - *Termi* alkaa joko sillä millä *Tekijä* alkaa tai sillä millä *Termi* alkaa
  - "*Termi* alkaa sillä millä *Termi* alkaa" ei tuo uusia mahdollisuuksia
  ⇒ *Termi* alkaa numerolla, kirjaimella, `(`:lla, `+`:lla tai `-`:lla

Tekstialkio (token) on jakamaton lopullisen tekstin osa

- useissa kielissä se tarkoittaa, että tekstialkion sisällä ei voi olla välejä, mutta tekstialkioiden välissä voi olla
- esim. C-kääntäjä sallii `x=x+1020;` ja `x = x + 1020 ;` mutta ei `x = x + 1 020 ;`
  ⇒ vaikka esim. `1020` koostuu neljästä merkistä, se on yksi tekstialkio
- esimerkissä tekstialkioita ovat `+`, `-`, `*`, `/`, `(` ja `)` sekä *luku* ja *muuttuja*
- esimerkki-BNF jättää kertomatta, miten *luku* ja *muuttuja* koostuvat merkeistä

BNF-määritelmä määrittelee yhden tai useita **kieliä** (**language**)

- siksi sitä voidaan kutsua myös **kieliopiksi** (**grammar**)
- esimerkki määrittelee kielet $\mathit{Lauseke}$, Termi, $\mathit{Tekijä}$ ja Atomi
- ellei toisin sanota, koko määritelmän kieli on ensimmäisenä määritelty kieli
  - esimerkissä $\mathit{Lauseke}$
- kukin kielen nimi esiintyy ::=:n vasemmalla puolella enintään kerran

Kieli on joukko merkkijonoja

- tässä asiayhteydessä **merkkijono** (**string**) on nolla tai useampia tekstialkioita peräkkäin
  - tässä tapauksessa todellakin tekstialkioita, eikä välttämättä merkkejä
- jos tekstialkio voi koostua useasta merkistä, niin tilanteessa on kaksi kieltä:
  - miten ilmaus muodostuu tekstialkioista, esim. $\mathit{muuttuja}$ `=` $\mathit{muuttuja}$ `+` luku `;`
  - miten tekstialkio muodostuu merkeistä, esim. $\mathit{muuttuja}$ on `x` ja luku on `1020`
- merkkijonoja on tapana merkitä pienillä kreikkalaisilla kirjaimilla σ (sigma), ρ (rho), α (alfa), β (beta) jne.
- ε (pieni epsilon) tarkoittaa tyhjää merkkijonoa
  - se merkkijono, jossa ei ole yhtään merkkiä
- koska tyhjän lisääminen merkkijonon eteen tai perään ei muuta merkkijonoa, jokaiselle merkkijonolle σ pätee $\varepsilon\sigma = \sigma\varepsilon = \sigma$

BNF:ää käytettäessä kieli määritellään säännöllä muotoa

$$\mathit{Nimi} \;::=\; \mathit{vaihtoehto} \mid \mathit{vaihtoehto} \mid \cdots \mid \mathit{vaihtoehto}$$

- kukin *vaihtoehto* on joko ε tai jono kielten nimiä ja/tai tekstialkioita
- *vaihtoehto*a käytettäessä kunkin siinä esiintyvän kielen nimen paikalle laitetaan mikä tahansa ko. kieleen kuuluva merkkijono
  - esim. $\mathit{Termi} ::= \mathit{Termi} \mid \mathit{Termi}\ \text{"*"}\ \mathit{Tekijä} \mid \mathit{Termi}\ \text{"/"}\ \mathit{Tekijä}$
  - `(37+37*-x)` on *Termi* ja `3` on *Tekijä*, joten `(37+37*-x)/3` on *Termi*

  $\Rightarrow$ kieleen kuuluvia merkkijonoja voidaan muodostaa lyhyistä alkaen
- ε tarkoittaa, että ei laiteta mitään

$\Rightarrow$ ε ei esitä itseään

- siis ε ei esitä kreikkalaista kirjainta pieni epsilon, vaan tyhjää merkkijonoa
- "ε" esittää kreikkalaisen kirjaimen pieni epsilon

$\Rightarrow$ *Tekijä* voidaan yhtäpitävästi määritellä myös näin:

$$\begin{aligned} \mathit{Tekijä} &::= \mathit{Etumerkki}\ \mathit{Atomi} \\ \mathit{Etumerkki} &::= \varepsilon \mid \text{"+"} \mid \text{"-"} \end{aligned}$$

Otamme käyttöön merkinnän $\mathcal{L}(K)$ tarkoittamaan kieltä, jonka $K ::= \ldots$ tuottaa

- $\sigma \in \mathcal{L}(K)$ tarkoittaa, että määritelmä $K ::= \ldots$ tuottaa merkkijonon σ
- esim. `37` $\in \mathcal{L}(\mathit{Lauseke})$, `37+37*-x` $\in \mathcal{L}(\mathit{Lauseke})$ ja `(37+37*-x)/3` $\in \mathcal{L}(\mathit{Lauseke})$

Pitkät kielten nimet ja "lainausmerkit" ovat toisinaan kömpelöitä

$\Rightarrow$ teemme silloin kuten matemaatikot:
- isot kirjaimet ovat kielten nimiä
- muut näkyvät merkit ovat tekstialkioita
- lainausmerkkejä ei käytetä ainakaan lainausmerkkien tehtävässä

- tässä on tärkeää muistaa, että ε ei esitä itseään!
  - kreikkalaista kirjainta pieni epsilon ei voi tässä esittää mitenkään
  - edes tavallisia isoja kirjaimia ei voi tässä esittää mitenkään

Lisää esimerkkejä

- jos $A ::= \varepsilon \mid aA$, niin $\mathcal{L}(A) = \{\varepsilon, a, aa, aaa, \ldots\}$
  - koska $A ::= \varepsilon \mid aA$, saamme $\varepsilon \in \mathcal{L}(A)$
  - koska $A ::= \varepsilon \mid aA$ ja $\varepsilon \in \mathcal{L}(A)$, saamme $a\varepsilon \in \mathcal{L}(A)$ eli $a \in \mathcal{L}(A)$
  - koska $A ::= \varepsilon \mid aA$ ja $a \in \mathcal{L}(A)$, saamme $aa \in \mathcal{L}(A)$
  - koska $A ::= \varepsilon \mid aA$ ja $aa \in \mathcal{L}(A)$, saamme $aaa \in \mathcal{L}(A)$
  - koska $A ::= \varepsilon \mid aA$ ja $aaa \in \mathcal{L}(A)$, saamme $aaaa \in \mathcal{L}(A)$
  - ...
- jos $A ::= \varepsilon \mid Aa$, niin $\mathcal{L}(A) = \{\varepsilon, a, aa, aaa, \ldots\}$
  - koska $A ::= \varepsilon \mid Aa$, saamme $\varepsilon \in \mathcal{L}(A)$
  - koska $A ::= \varepsilon \mid Aa$ ja $\varepsilon \in \mathcal{L}(A)$, saamme $\varepsilon a \in \mathcal{L}(A)$ eli $a \in \mathcal{L}(A)$
  - koska $A ::= \varepsilon \mid Aa$ ja $a \in \mathcal{L}(A)$, saamme $aa \in \mathcal{L}(A)$
  - koska $A ::= \varepsilon \mid Aa$ ja $aa \in \mathcal{L}(A)$, saamme $aaa \in \mathcal{L}(A)$
  - ...

- jos $A ::= \varepsilon \mid aAb$, niin $\mathcal{L}(A) = \{\varepsilon, ab, aabb, aaabbb, \ldots\} = \{a^n b^n \mid n \in \mathbb{N}\}$
  - koska $A ::= \varepsilon \mid aAb$, saamme $\varepsilon \in \mathcal{L}(A)$
  - koska $A ::= \varepsilon \mid aAb$ ja $\varepsilon \in \mathcal{L}(A)$, saamme $a\varepsilon b \in \mathcal{L}(A)$ eli $ab \in \mathcal{L}(A)$
  - koska $A ::= \varepsilon \mid aAb$ ja $ab \in \mathcal{L}(A)$, saamme $aabb \in \mathcal{L}(A)$
  - koska $A ::= \varepsilon \mid aAb$ ja $aabb \in \mathcal{L}(A)$, saamme $aaabbb \in \mathcal{L}(A)$
  - koska $A ::= \varepsilon \mid aAb$ ja $aaabbb \in \mathcal{L}(A)$, saamme $aaaabbbb \in \mathcal{L}(A)$
  - ...
- jos $A ::=$ a $\mid [B]$ ja $B ::= AA$, niin $\mathcal{L}(A)$:n ja $\mathcal{L}(B)$:n alkioita voidaan tuottaa vuorotellen seuraavasti:

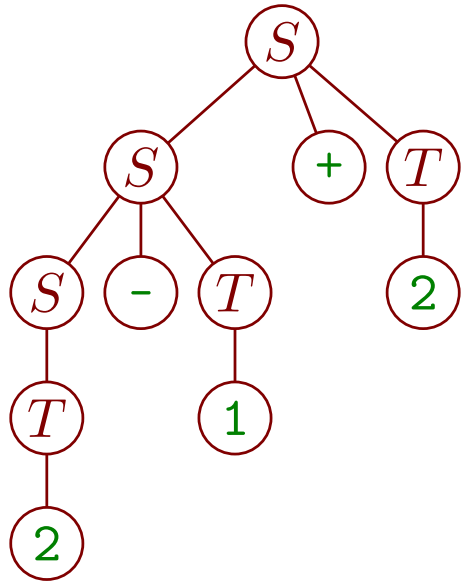
$\mathcal{L}(A)$	a	[aa]	[a[aa]]	[[aa]a]	[[aa][aa]]	...
$\mathcal{L}(B)$	aa	a[aa]	[aa]a	[aa][aa]	a[a[aa]]	...

- jos $A ::= aA$, niin $\mathcal{L}(A) = \emptyset$ eli tyhjä joukko
  - kieleen ei siis kuulu yhtään merkkijonoa
  - yhtään merkkijonoa ei voida tuottaa, koska aA tuottaa jotain vain jos A:ssa on valmiiksi jotain
  - merkkijonojen tuottaminen ei siis pääse alkuun, eikä siksi tuota mitään
- samasta syystä myös mm. seuraavat tuottavat tyhjän kielen:
  - $A ::= B$ ja $B ::= AA$
  - $A ::= A$
  - $A ::= aBc \mid cBa$ ja muita sääntöjä ei ole (joten B:lle ei ole sääntöä)

- jos $S ::= T \mid S\texttt{+}T \mid S\texttt{-}T$ ja $T ::= \texttt{1} \mid \texttt{2} \mid \texttt{(}S\texttt{)}$, niin
  - $\mathcal{L}(S) = \{\texttt{1}, \texttt{2}, \texttt{1+1}, \texttt{1+2}, \texttt{2+1}, \texttt{2+2}, \texttt{1-1}, \ldots, \texttt{(1)}, \texttt{(2)}, \ldots, \texttt{2-1+2}, \texttt{(2-(1+2))}, \ldots\}$
  - $\mathcal{L}(T) = \{\texttt{1}, \texttt{2}, \texttt{(1)}, \texttt{(2)}, \texttt{(1+1)}, \texttt{(1+2)}, \ldots, \texttt{(2-1+2)}, \texttt{(2-(1+2))}, \ldots\}$

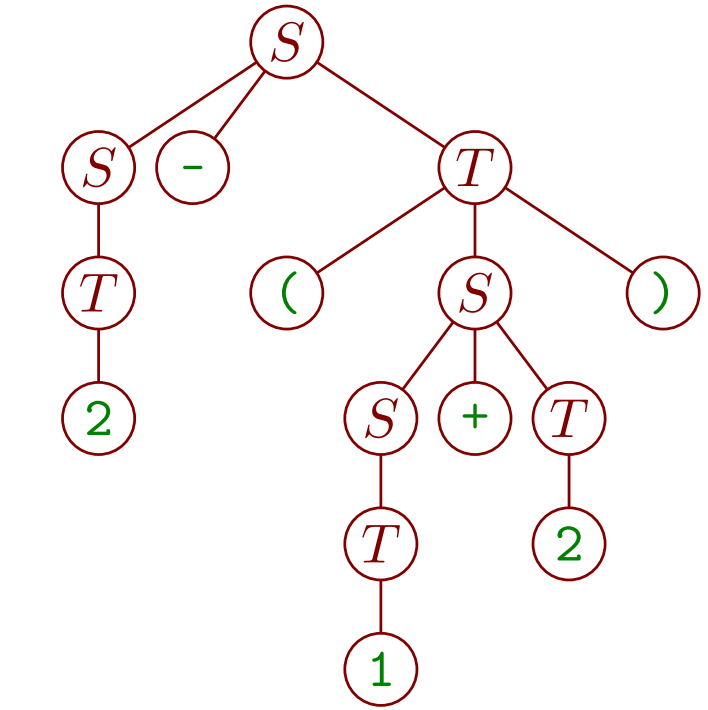
Kielen BNF-kielioppia voi käyttää

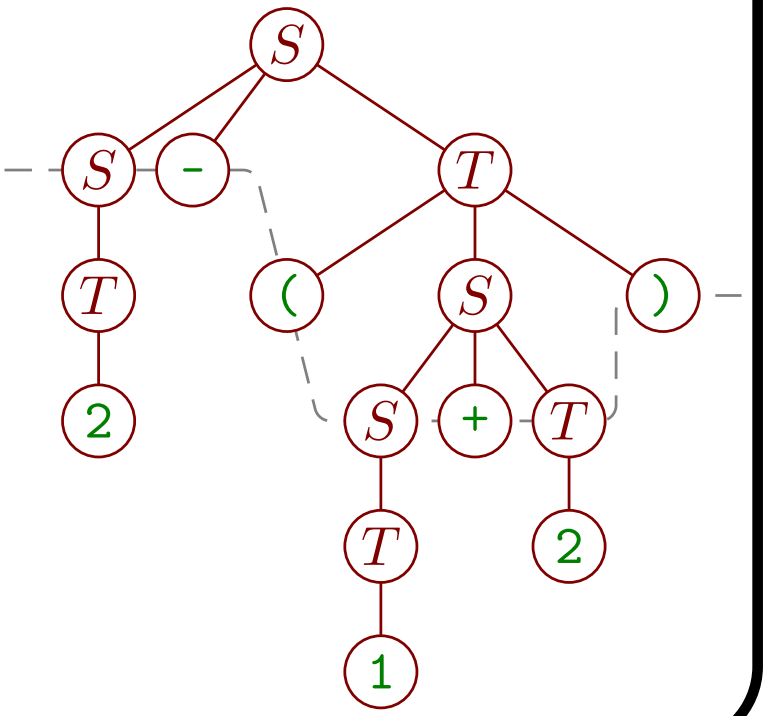
- alhaalta ylös
  - $\texttt{2} \in \mathcal{L}(T) \;\Rightarrow\; \texttt{2} \in \mathcal{L}(S)$
  - $\texttt{2} \in \mathcal{L}(S)$ ja $\texttt{1} \in \mathcal{L}(T) \;\Rightarrow\; \texttt{2-1} \in \mathcal{L}(S)$
  - $\texttt{2-1} \in \mathcal{L}(S)$ ja $\texttt{2} \in \mathcal{L}(T) \;\Rightarrow\; \texttt{2-1+2} \in \mathcal{L}(S)$
- ylhäältä alas, vasemmanpuoleisin ensin
  - $S \rightsquigarrow S\texttt{+}T \rightsquigarrow S\texttt{-}T\texttt{+}T \rightsquigarrow T\texttt{-}T\texttt{+}T \rightsquigarrow$ $\texttt{2-}T\texttt{+}T \rightsquigarrow \texttt{2-1+}T \rightsquigarrow \texttt{2-1+2}$
- ylhäältä alas, oikeanpuoleisin ensin
  - $S \rightsquigarrow S\texttt{+}T \rightsquigarrow S\texttt{+2} \rightsquigarrow S\texttt{-}T\texttt{+2} \rightsquigarrow S\texttt{-1+2} \rightsquigarrow T\texttt{-1+2} \rightsquigarrow \texttt{2-1+2}$
- sääntöjen käytön voi esittää kuvana, jolloin suuntaan ei oteta kantaa
- tällainen kuva on **jäsennyspuu** (**parse tree**)
- merkkijono kuuluu kieliopin tuottamaan kieleen jos ja vain jos sillä on jäsennyspuu



Toinen esimerkki

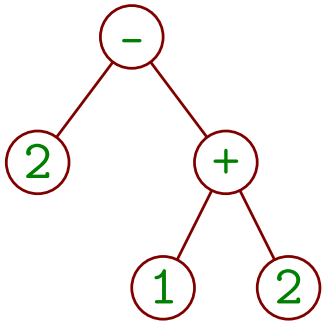
- alhaalta ylös
  - $2 \in \mathcal{L}(T) \Rightarrow 2 \in \mathcal{L}(S)$
  - $1 \in \mathcal{L}(T) \Rightarrow 1 \in \mathcal{L}(S)$
  - $2 \in \mathcal{L}(T)$
  - $1 \in \mathcal{L}(S)$ ja $2 \in \mathcal{L}(T) \Rightarrow 1+2 \in \mathcal{L}(S) \Rightarrow (1+2) \in \mathcal{L}(T)$
  - $2 \in \mathcal{L}(S)$ ja $(1+2) \in \mathcal{L}(T) \Rightarrow 2-(1+2) \in \mathcal{L}(S)$
- ylhäältä alas, vasemmanpuoleisin ensin
  - $S \rightsquigarrow S-T \rightsquigarrow T-T \rightsquigarrow 2-T \rightsquigarrow 2-(S) \rightsquigarrow 2-(S+T) \rightsquigarrow 2-(T+T) \rightsquigarrow 2-(1+T) \rightsquigarrow 2-(1+2)$
- ylhäältä alas, oikeanpuoleisin ensin
  - $S \rightsquigarrow S-T \rightsquigarrow S-(S) \rightsquigarrow S-(S+T) \rightsquigarrow S-(S+2) \rightsquigarrow S-(T+2) \rightsquigarrow S-(1+2) \rightsquigarrow T-(1+2) \rightsquigarrow 2-(1+2)$
- kuvassa on välivaihe $S-(S+T)$





Jäsennyspuu on eri asia kuin lausekepuu

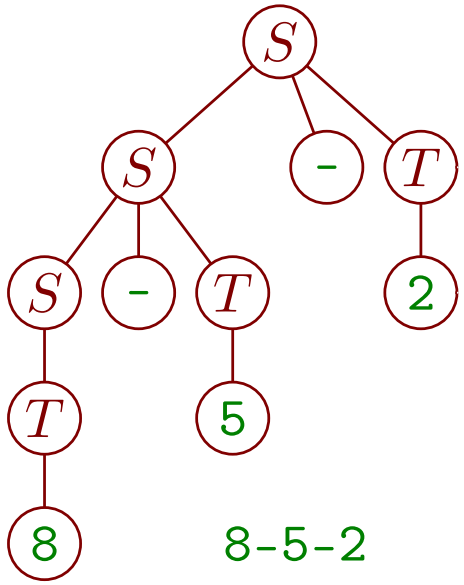
- lausekepuu esittää asian loogisen rakenteen
- jäsennyspuu esittää miten merkkijono saadaan kieliopista



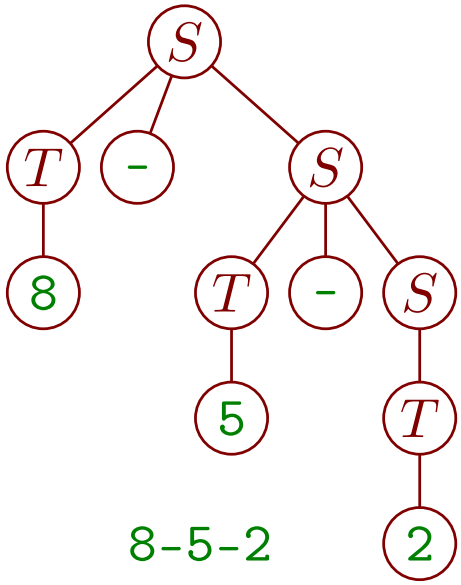
- jäsennyspuu voi sisältää apukielten nimiä, jäsennystä ohjaavia välimerkkejä yms.
  - esim. sulut, sijoituslauseen lopussa oleva `;`, sana `else`
- jäsennyspuulla ei välttämättä ole yhteyttä asian loogiseen rakenteeseen, koska voidaan käyttää "vääriä" jäsennyspuita tuottavaa kielioppia
- jotta voisi olla lausekepuu, tarvitaan lauseke!
  - $A ::= \varepsilon \mid aAb$ tuottaa mm. $aabb$, mikä olisi sen lausekepuu?

Eri kieliopit voivat tuottaa saman kielen eri jäsennyspuilla

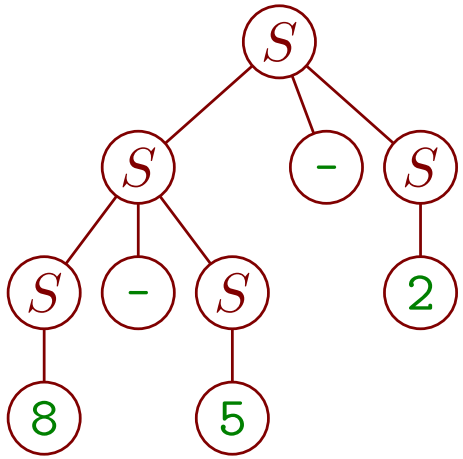
- $S ::= T \mid S\texttt{+}T \mid S\texttt{-}T$ ja $T ::= \texttt{0} \mid \cdots \mid \texttt{9} \mid \texttt{(}S\texttt{)}$
- $S ::= T \mid T\texttt{+}S \mid T\texttt{-}S$ ja $T ::= \texttt{0} \mid \cdots \mid \texttt{9} \mid \texttt{(}S\texttt{)}$
- $S ::= S\texttt{+}S \mid S\texttt{-}S \mid \texttt{0} \mid \cdots \mid \texttt{9} \mid \texttt{(}S\texttt{)}$



8-5-2



8-5-2



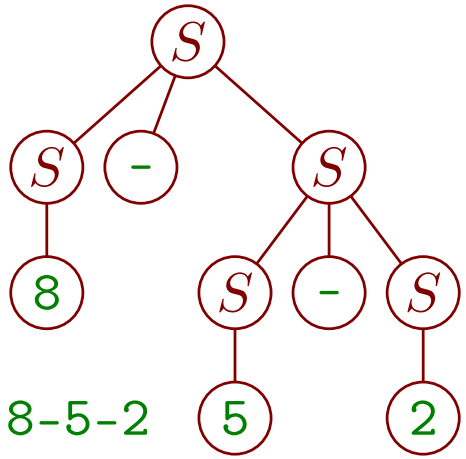
8-5-2

Mikäli mahdollista, kannattaa valita kielioppi, jonka tuottamat jäsennyspuut vastaavat kohteena olevia rakenteita

- matematiikassa $8-5-2=(8-5)-2=3-2=1$
  eikä $8-5-2=8-(5-2)=8-3=5$

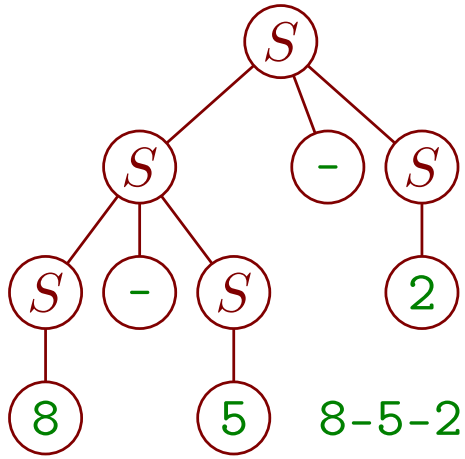
$\Rightarrow$ esim. $S ::= T \mid S\texttt{+}T \mid S\texttt{-}T$
eikä $S ::= T \mid T\texttt{+}S \mid T\texttt{-}S$

$\Rightarrow$ kielioppi saadaan tukemaan rakenteen ymmärtämistä



- esim.
  - potenssilasku sitoo voimakkaammin kuin kertolasku
  - kertolasku sitoo vasemmalle ja potenssilasku sitoo oikealle
  - esim. $3ax^{2^n} = (3a)(x^{(2^n)})$

$$K ::= P \mid KP$$
$$P ::= A \mid A^P$$



Jos ja vain jos jollekin merkkijonolle on kaksi eri jäsennyspuuta, niin kielioppi on **monikäsitteinen** (**ambiguous**)

- esim. $S ::= S\texttt{+}S \mid S\texttt{-}S \mid \texttt{0} \mid \cdots \mid \texttt{9} \mid \texttt{(}S\texttt{)}$
- tällaisiakin kielioppeja näkee, erityisesti kun on haluttu tiivis esitys
  - esim. $L ::= luku \mid muuttuja \mid L+L \mid L-L \mid LL \mid L/L \mid L^L$
- on olemassa BNF:llä määriteltävissä olevia kieliä, joiden kaikki BNF-määritelmät ovat monikäsitteisiä

Kielioppi *saattaa* olla laadittu niin, että jäsennyspuut vastaavat asian rakennetta, mutta varmaa se ei ole

- matematiikassa kielen käsite ei ole mitään muuta kuin merkkijonojen jako kahtia: mukana ja ulkona
  - muitakin asioita voidaan tutkia matemaattisesti, mutta niistä käytetään eri nimiä
  - esim. ilmauksien rakenteita voi tutkia kielioppien avulla
  - esim. ilmauksien merkityksiä voi tutkia logiikan malliteorian avulla
- kieliopin avulla *voidaan* esittää varsin paljon kielen syntaktisesta rakenteesta, *jos halutaan*
- on luvallista ja toisinaan järkevää käyttää kielioppeja ilmaisemaan kieliä, joiden merkkijonoilla ei ole tärkeää syntaktista rakennetta
  - millä perusteella $A ::= \varepsilon \mid Aa$ on oikeampi tai väärempi määritelmä kielelle $\{\varepsilon, a, aa, aaa, \ldots\}$ kuin $A ::= \varepsilon \mid aA$?
- toisinaan on helpompaa rakentaa jäsenninohjelma "väärälle" kieliopille ja korjata tulos myöhemmissä käsittelyvaiheissa
  - on helpompaa rakentaa jäsentäjä säännölle $S ::= T \mid T\text{+}S \mid T\text{-}S$ kuin $S ::= T \mid S\text{+}T \mid S\text{-}T$

  $\Rightarrow$ mm. MathCheckin jäsentäjä perustuu "väärään" sääntöön $S ::= T \mid T\text{+}S \mid T\text{-}S$

Ohjelmointikielten nelitasomalli ja kaksitasomalli

- leksikaalitaso
- varsinainen syntaksi

} syntaksi

- staattinen semantiikka
- dynaaminen semantiikka

} semantiikka

Alakoululaisten esimerkki syntaksin ja semantiikan erosta: mihin kesä loppuu?

- semantiikka: puiden lehtien kellastumiseen
- syntaksi: ä-kirjaimeen

Leksikaalitaso

- merkkien ryhmittyminen tekstialkioiksi
  - varatut sanat: esim. `while`, `int`
  - ohjelmoijan määrittelemät nimet: esim. `x`, `oma_tyyppi`
  - literaalivakiot: esim. `3.14159`, `"tämä on merkkijono"`
  - merkeillä esitetyt: esim. `=`, `<=`, `++` ja `;`
- tyypillisesti tekstialkioiden sisällä ei saa ja välissä saa olla ylimääräistä tyhjää
  - tyhjä tila = välilyönnit, rivinsiirrot, sarkaimet eli tabulaattorit ja kommentit
  - esim. `while ( i > 0 )` ja `while(i>0)` kelpaavat, `wh ile ( i > 0 )` ei kelpaa
  - kuitenkin merkkijonoissa voi olla tyhjää tilaa, mutta siellä sillä on merkitys

- jos tekstialkio voi olla toisen alkuosa, niin tyypillisesti valitaan pisin mahdollinen
  - esim. `i- -1` kelpaa C-kääntäjälle, tarkoittaa vähennyslaskua
  - `i--1` ei kelpaa, `--` on operaattori jolle `i--` on mahdollinen mutta `i--1` ei ole
- tekstialkioiden muodostuminen merkeistä määritellään usein jollain BNF:n versiolla
- luonnollisissakin kielissä on (kenties sumeita) leksikaalisia sääntöjä
  - seuraavat on tehty arpomalla kirjaimia kolmen edellisen kirjaimen perusteella jakaumalla, joka on laskettu suomen tai englannin sanojen luettelosta

    raimplativer tomon cochred tuffrancork anougglashesteng ing proad

    saleittantua kuus kea ta-ampuvuotoutiikka punsijäämällipoida poskisti
  - ei ole vaikea huomata, kumpi on suomesta ja kumpi englannista

Varsinainen syntaksi

- tekstialkioiden järjestystä koskevat muotoseikat
  - esim. `-(1+2)*3` on ja `*1+)2(3-` ei ole kelvollinen *Lauseke*
  - esim. jokaisella alkusululla `(` täytyy olla vastaava loppusulku `)`
  - esim. `int while < i 0` ei kelpaa C-kääntäjälle
- määritellään usein jollain BNF:n versiolla
- tärkeä taso, koska
  - voidaan jäsentää tehokkaasti
  - voidaan ilmaista monimutkaisia hierarkkisia ihmisille luontevia rakenteita
- järjestyssääntöjä suomenkielessäkin on, huomata kuten voi

Staattinen semantiikka

- muut käännösaikana tarkastettavat asiat kuin syntaksi
  - esim. muuttujat on määriteltävä ennen käyttöä $\rightsquigarrow$ symbolitaulu
  - esim. tyyppitarkastukset

Dynaaminen semantiikka

- ajoaikana tai ei edes silloin tarkastettavat asiat
  - esim. nollalla jako, `int *p = 0; p* = 3;`
  - esim. `A[i++] = i;`
- ohjelman merkitys
  - mitä se laskee
  - miten se sen laskee
- luonnollisten kielten semantiikaksi kutsutaan ilmauksien merkitystä
  - esim. täsä-o syn daxi päeMÄNTYä mutt, siilti shemandiickan t a j u u
  - esim. syntaksi oikein, semantiikka ei:
    kuusi on havupuu, joten seitsemän on havupuu plus yksi

Joskus on tärkeää tiedostaa ero ilmauksen ja sen merkityksen välillä

- käyttämisen ja mainitsemisen ero (use–mention distinction)
- tietotekniikassa tarvitaan toisinaan kikkoja kun halutaan mainita eikä käyttää
  - HTML:ssä mm. `<` ja `&` eivät tarkoita itseään ⇒
    usein joudutaan kirjoittamaan `<` ja `&`
  - C:n merkkijonoissa `"` ja `\` eivät tarkoita itseään ⇒
    joudutaan kirjoittamaan `\"` ja `\\`
  - LaTeXissa `&` ei tarkoita itseään ⇒ joudutaan kirjoittamaan `\&`
  - Lispissä `(+ 2 3)` tuottaa `5`, mutta `'(+ 2 3)` tuottaa `(+ 2 3)`
- edellä ”mihin kesä loppuu” voidaan myös nähdä esimerkkinä tästä

3 Logiikka yhtälöiden yms. ratkaisemisessa

Ensimmäisen asteen yhtälö $3x - 12 = 0$ voidaan ratkaista seuraavasti:

- siirretään vakiotermi toiselle puolelle etumerkki vaihtaen: $3x = 12$
- jätetään muuttujan kerroin pois vasemmalta puolelta, ja jaetaan oikea puoli sillä: $x = \frac{12}{3} = 4$

Tavoitteenamme on

- ymmärtää, miksi tämä on oikein
- oppia keksimään lisää oikeita ratkaisumenetelmiä
- oppia käyttämään logiikan merkintöjä päättelyiden ilmaisemiseen

Otamme käyttöön seuraavat merkinnät ilmaisemaan **päättelyaskelia**:

vasen	$\Rightarrow$	oikea	jos vasen puoli on totta, niin myös oikea puoli on totta
vasen	$\Leftarrow$	oikea	jos oikea puoli on totta, niin myös vasen puoli on totta
vasen	$\Leftrightarrow$	oikea	jos jompikumpi puoli on totta, niin myös toinen puoli on totta

- vasen $\Leftarrow$ oikea tarkoittaa samaa kuin oikea $\Rightarrow$ vasen
- vasen $\Leftrightarrow$ oikea tarkoittaa samaa kuin vasen $\Leftarrow$ oikea ja oikea $\Rightarrow$ vasen yhdessä

Esimerkkejä, joille vasen $\Rightarrow$ oikea pätee:

- Jussin äiti on Tuula $\Rightarrow$ Tuulalla on lapsi
- $x > 8 \Rightarrow x > 5$

Niille vasen $\Leftrightarrow$ oikea ei päde:

- voi olla vaikka niin, että Jussin äiti on Tiina ja Tiinan äiti on Tuula
- kun $x = 6$, niin $x > 5$ on totta mutta $x > 8$ ei ole totta

Esimerkkejä, joille vasen $\Leftrightarrow$ oikea pätee:

- Jussin äiti tai isä on Tuula $\Leftrightarrow$ Jussi on Tuulan lapsi
- $x > 8 \Leftrightarrow x - 3 > 5$

Vasen $\Leftrightarrow$ oikea pätee jos ja vain jos sekä vasen $\Rightarrow$ oikea että vasen $\Leftarrow$ oikea pätee

- usein esiintyvä virhe: päätellään vasen $\Leftarrow$ oikea, kun tiedetään vain vasen $\Rightarrow$ oikea
- siitä, että Tuulalla on lapsi, ei voi päätellä, että Jussi on se lapsi
  - se voi olla totta, mutta annettu tieto ei riitä takaamaan, että se on totta
- $x > 5$ ei yksinään takaa, että $x > 8$

Päättelyaskelia saa ketjuttaa

- esim. $3x - 12 = 0 \Leftrightarrow 3x = 12 \Leftrightarrow x = 4$
- ketju pätee, jos ja vain jos sen jokainen askel pätee

Miksi $3x - 12 = 0 \Leftrightarrow 3x = 12 \Leftrightarrow x = 4$ on pätevä?

- tutkimme kummankin suunnan erikseen

Miksi $3x - 12 = 0 \Rightarrow 3x = 12 \Rightarrow x = 4$ on pätevä?

- kun yhtäsuurille tehdään sama laillinen temppu, ovat lopputuloksetkin yhtäsuuret
  - jos kaksi samanlaista valkoista autoa maalataan punaisiksi, saadaan kaksi samanlaista punaista autoa
  - jos Jussi on nyt yhtä vanha kuin Veera, niin myös kahden vuoden päästä Jussi on yhtä vanha kuin Veera

$\Rightarrow$ jos $3x - 12 = 0$, niin $(3x - 12) + 12 = 0 + 12$

- jos jokin on yhtäsuuri toisen kanssa ja se toinen kolmannen kanssa, niin ensimmäinen on yhtäsuuri kolmannen kanssa
  - jos Jussin auto on samanvärinen kuin Veeran auto ja Veeran auto on samanvärinen kuin Aaron auto, niin Jussin auto on samanvärinen kuin Aaron auto
  - jos Jussi on yhtä vanha kuin Veera ja Veera on yhtä vanha kuin Aaro, niin Jussi on yhtä vanha kuin Aaro
- nollan määritelmästä seuraa $0 + 12 = 12$
- olemme saaneet $(3x - 12) + 12 = 0 + 12$ ja $0 + 12 = 12$, joten voimme päätellä $(3x - 12) + 12 = 12$
- vähennyslaskun määritelmästä seuraa $(3x - 12) + 12 = 3x$

- jos jokin on yhtäsuuri toisen kanssa, niin se toinen on yhtäsuuri sen jonkin kanssa
  - jos Jussin auto on samanvärinen kuin Veeran auto, niin Veeran auto on samanvärinen kuin Jussin auto
  - jos Jussi on yhtä vanha kuin Veera, niin Veera on yhtä vanha kuin Jussi
- olemme saaneet $(3x - 12) + 12 = 3x$, joten voimme päätellä $3x = (3x - 12) + 12$
- olemme saaneet $3x = (3x - 12) + 12$ ja $(3x - 12) + 12 = 12$, joten voimme päätellä $3x = 12$
- siksi $3x - 12 = 0 \Rightarrow 3x = 12$ on pätevä
- kun yhtäsuurille tehdään sama laillinen temppu, ovat lopputuloksetkin yhtäsuuret

$\Rightarrow$ jos $3x = 12$, niin $\frac{3x}{3} = \frac{12}{3}$

- koska $\frac{3x}{3} = x$ ja $\frac{12}{3} = 4$, saadaan $x = \frac{3x}{3} = \frac{12}{3} = 4$, joten $x = 4$
- siksi $3x = 12 \Rightarrow x = 4$ on pätevä
- koko ketju on nyt tarkastettu $\Rightarrow$-suuntaan

Mitä lainalaisuuksia käytettiin?

- yhtäsuuruuden ominaisuuksia: kaikille reaaliluvuille a, b ja c pätee
  - jos $a = b$ ja $b = c$, niin $a = c$ — transitiivisuus
  - jos $a = b$ niin $b = a$ — symmetrisyys
  - jos $a = b$ niin $f(a) = f(b)$ jne. — kongruenssiominaisuus
  - lisäksi on yksi, jota ei käytetty: jokainen on yhtäsuuri itsensä kanssa $a = a$ — refleksiivisyys

- lukujen ominaisuuksia: seuraavat pätevät kaikille reaaliluvuille a ja b
  - nollan määritelmästä seuraa: $0 + a = a$
  - vähennyslaskun määritelmästä seuraa: $(a - b) + b = a$
  - jakolaskua koskevia tietoja: $\frac{12}{3} = 4$ ja jos $a \neq 0$, niin $\frac{ab}{a} = b$
  - lukujen ominaisuuksien luettelo on pitkä, palataan siihen myöhemmin

Miksi edellä puhuttiin "laillisista" tempuista?

- esim. $x = 5 \Rightarrow \frac{x}{0} = \frac{5}{0}$ ei ole pätevä, koska nollalla ei saa jakaa
- tuo on helppo välttää, mutta nollalla jako voi syntyä myös vaikeammin havaittavalla tavalla
- esim. $x^2 = x \Rightarrow \frac{x^2}{x} = \frac{x}{x} \Rightarrow x = 1$ on väärin:
  kun $x = 0$, niin $x^2 = x$ on totta mutta $x = 1$ ei ole totta

Miksi $3x - 12 = 0 \Leftarrow 3x = 12 \Leftarrow x = 4$ on pätevä?

- kun yhtälön $3x = 12$ molemmilta puolilta vähennetään 12, saadaan $3x - 12 = 12 - 12$
- $12 - 12 = 0$
- siksi $3x = 12 \Rightarrow 3x - 12 = 0$
- kun yhtälön $x = 4$ molemmat puolet kerrotaan 3:lla, saadaan $3x = 12$
- siksi $x = 4 \Rightarrow 3x = 12$

Toimiiko $\Leftrightarrow$ aina, kun $=$:n molemmille puolille tehdään sama laillinen temppu?

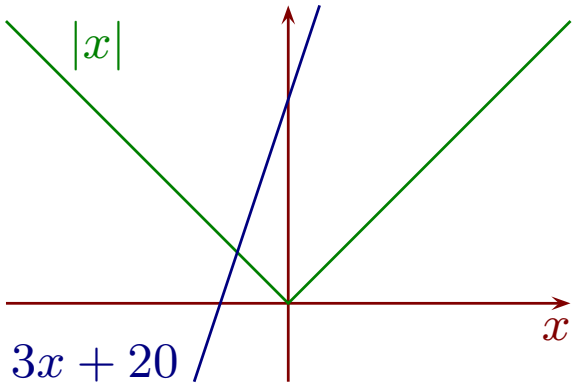
- ei todellakaan!
- $x = 7 \;\Rightarrow\; 0x = 0$ on pätevä, mutta $x = 7 \;\Leftarrow\; 0x = 0$ ei ole
  - vastaesimerkki: kun $x = 1$, niin oikea puoli on totta mutta vasen ei ole
  - tälle on loputtomasti vastaesimerkkejä
- $x = 2 \;\Rightarrow\; x^2 = 4$ on pätevä, mutta $x = 2 \;\Leftarrow\; x^2 = 4$ ei ole
  - vastaesimerkki: kun $x = -2$, niin oikea puoli on totta mutta vasen ei ole
  - tälle ei ole muita vastaesimerkkejä, mutta yksikin riittää

Esimerkki päättelyn jakamisesta haaroihin

- miten matkustaa Matematiikan päiville Ouluun, jotka olivat 2.–3.1.2020?
- jos bussilla, niin
  - tulee pitkä istuminen epämukavissa oloissa
  - paluumatkalle pitää lähteä niin ajoissa, että paikallisbussilla varmasti ehtii pitkän matkan bussiin
  - menomatka on hankala, koska uudenvuodenpäivänä ei ajeta kaikkia vuoroja
- jos junalla, niin ...
- jos omalla autolla, niin
  - tulee pitkä ajomatka, josta suuri osa pimeässä talvisäässä
  - perillä voi tulla pysäköintiongelmia
- havainto: eri haaroissa aivan eri asiat voivat määrätä, mikä on pätevää päättelyä

Esimerkki päättelyn jakamisesta haaroihin: ratkaise $|x| = 3x + 20$

- itseisarvon määritelmä: $|a| = \begin{cases} -a & \text{kun } a < 0 \\ a & \text{kun } a \geq 0 \end{cases}$



$\Rightarrow$ joko $x < 0$ ja $-x = 3x + 20$, tai $x \geq 0$ ja $x = 3x + 20$

- tapaus $x < 0$: $-x = 3x + 20 \Leftrightarrow -x - 3x = 20 \Leftrightarrow$
  $-4x = 20 \Leftrightarrow 4x = -20 \Leftrightarrow x = -5$
- tapaus $x \geq 0$: $x = 3x + 20 \Leftrightarrow 0 = 3x - x + 20 \Leftrightarrow$
  $0 = 2x + 20 \Leftrightarrow 2x = -20 \Leftrightarrow x = -10 \Leftrightarrow$ **F**
- **F** tarkoittaa "false" eli epätosi eli että mikään luku ei kelpaa x:ksi
- $x = -10 \Leftrightarrow$ **F** ei normaalisti ole pätevä, koska sen vasen puoli on totta mutta oikea puoli ei ole kun $x = -10$
- nyt $x = -10$ ei kuitenkaan kelpaa vastaesimerkiksi lähtöoletuksen $x \geq 0$ vuoksi

Päättelyaskeleen pätevyys voi siis riippua asiayhteydestä

- $x = -10 \Leftrightarrow$ **F** ei normaalisti ole pätevä, mutta oli äskeisessä esimerkissä
- $x > 5 \Rightarrow x \geq 6$ on kokonaisluvuilla pätevä, mutta reaaliluvuilla esim. $x = 5\frac{1}{2}$ on sille vastaesimerkki
  - reaaliluvut = "kaikki tutut luvut", muutkin kuin kokonaisluvut
- jos jako miesten ja naisten etunimiin oletetaan, niin seuraava on pätevä:
  Jussin äiti on Tuula $\Leftrightarrow$ Jussi on Tuulan poika

Asiayhteyttä on kahta lajia

1. **kohdealue** (**domain of discourse**): se mistä maailmasta ylipäänsä puhutaan
   - tuo mukanaan (usein paljon) merkintöjä ja lainalaisuuksia

- esimerkkejä
  - reaaliluvut: 0, 1, $\sqrt{2}$, $a(b+c) = ab + ac$, ...
  - sukulaisuussuhteet: äiti, äiti on nainen, jos x on y:n äiti niin y on x:n lapsi, ...
  - jonkin ohjelmointikielen ilmaukset
  - opintohallinto: kurssit, tentit, aikataulut, ilmoittautumiset jne.
- kohdealueiden rajojen rikkominen johtaa usein sekopäisiin ilmauksiin
  - esim. Jussin äiti + 3 = aurinkoinen sää

2. tilapäiset lähtöoletukset
   - esim. tapauksen ehto $x \geq 0$
   - tilapäisten merkintöjen ominaisuudet, esim. annetaan nimi luvullle, jonka olemassaolo on osoitettu mutta arvoa ei tunneta
   - esim. ristiriitatodistus: oletetaan sen vastakohta mikä halutaan todistaa, ja osoitetaan että se johtaisi mahdottomuuteen

Vastaesimerkki on yksittäistapaus, jossa väite ei ole tosi tai päättelyaskel ei ole pätevä

- sen täytyy noudattaa kohdealueen lainalaisuuksia ja tilapäisiä lähtöoletuksia
  - merkitsemme tätä hieman alempana (*)
- vastaesimerkki vastaa ohjelmoinnissa syötettä, jolla ohjelma toimii väärin
- yksikin vastaesimerkki riittää osoittamaan väitteen tai päättelyn virheelliseksi
  - matematiikassa poikkeus ei vahvista sääntöä, vaan kumoaa sen

Väite on totta, jos ja vain jos sille ei ole olemassa yhtään vastaesimerkkiä

Päättelyaskel on pätevä, jos ja vain jos sille ei ole olemassa yhtään vastaesimerkkiä

päättelyaskel	vastaesimerkki
vasen $\Rightarrow$ oikea	sellainen tilanne, että (*) ja vasen puoli on totta mutta oikea ei ole
vasen $\Leftarrow$ oikea	sellainen tilanne, että (*) ja oikea puoli on totta mutta vasen ei ole
vasen $\Leftrightarrow$ oikea	sellainen tilanne, että (*) ja toinen puoli on totta mutta toinen ei ole

Siis

- vasen $\Rightarrow$ oikea on pätevä, jos ja vain jos kaikissa sellaisissa tilanteissa, jotka toteuttavat kohdealueen lainalaisuudet ja tilapäiset lähtöoletukset, ja joissa vasen puoli on totta, myös oikea puoli on totta
- vasen $\Leftrightarrow$ oikea on pätevä, jos ja vain jos sekä vasen $\Rightarrow$ oikea että oikea $\Rightarrow$ vasen on pätevä

Voidaan myös ajatella, että jos vasen $\Rightarrow$ oikea, niin vasen sisältää ainakin saman informaation tilanteesta kuin oikea, mutta voi sisältää enemmän

Pätevää päättelyä ei siis määritelty tässä siten, että jokin hyväksytty sääntö tuottaa oikean puolen vasemmasta

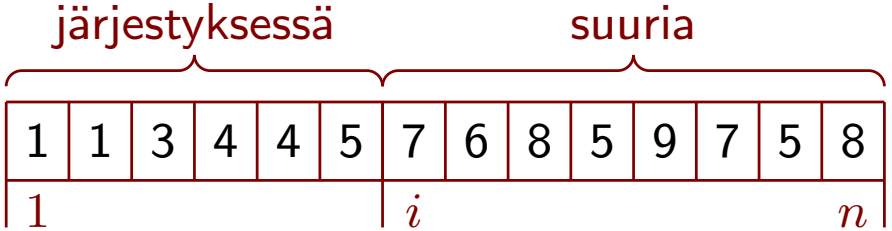
- niinkin olisi voitu tehdä
- käytännön päättelyissä tehdään usein pitkiä loikkia
  - esim. $3x - 12 = 0 \Leftrightarrow 3x = 12$ esitetään yleensä korkeintaan yhtenä askeleena, vaikka se koostuu monesta pikkuaskeleesta, kuten näimme
- on vaikea vetää raja sille, kuinka pitkä loikka hyväksytään päteväksi askeleeksi
- jos miten pitkä loikka tahansa hyväksytään, saadaan sama tulos kuin kriteerillä "jos ja vain jos ei ole yhtään vastaesimerkkiä"
  - Kurt Gödel todisti tämän 23-vuotiaana väitöskirjassaan 1929
  - pätee ns. ensimmäisen kertaluvun logiikalle, mutta juuri sitä käytämme
  - tämä on hyvin syvällinen ja vaikeasti todistettava tulos
- "ei vastaesimerkkejä" -määritelmä on helpompi
  - ei tarvitse (vielä!) antaa pitkää luetteloa pätevistä päättelysäännöistä
- usein käytännössä päättelyn voi nähdä päteväksi vain päättelysääntöjen perusteella
- käytännön matematiikassa "pätevä päättely" on sumea käsite
  - periaatteessa olisi mahdollista määritellä täysin täsmällisesti
  - käytännössä hukuttaisiin tolkuttomaan määrään toisarvoisia yksityiskohtia

Esimerkki: selection-sort

- järjestää taulukon $A[1 \ldots n]$ kasvavaan suuruusjärjestykseen
- etsii taulukon ensimmäisen pienimmän ja vaihtaa sen ensimmäiseksi; etsii loppuosan ensimmäisen pienimmän ja vaihtaa sen loppuosan ensimmäiseksi jne.
  - ”ensimmäisen pienimmän” eikä ”pienimmän”, koska yhtäsuuria voi olla monta

1. osuus $A[1 \ldots i-1]$ on aina rivin 1 alussa kasvavassa suuruusjärjestyksessä
2. jokainen osuuden $A[i \ldots n]$ alkio on vähintään yhtä suuri kuin jokainen osuuden $A[1 \ldots i-1]$ alkio

```
1  for i := 1 to n − 1 do
2      m := i
3      for j := i + 1 to n do
4          if A[j] < A[m] then m := j
5      x := A[i]; A[i] := A[m]; A[m] := x
```



- alussa $i = 1$, joten järjestyksessä olevien osuus on tyhjä
  - onko se silloin kasvavassa suuruusjärjestyksessä, eli onko 1. silloin totta?
- ihmiskunta voi itse päättää sanojen merkitykset, kunhan on niille johdonmukainen
- ”tyhjä osuus on järjestyksessä” on kätevämpi kuin ”. . . ei ole järjestyksessä”, koska alkutilanteesta ei tarvitse tehdä erikoistapausta 1.:ssä
- se on myös loogisempi (vaikka sitä voi olla vaikea nähdä tässä vaiheessa)

- se täsmää periaatteeseen ”totta, jos ja vain jos ei vastaesimerkkejä”
  - vastaesimerkki järjestykselle olisi sellaiset $1 \leq k < \ell < i$, että $A[k] > A[\ell]$
  - kun $i = 0$, sellaisia ei voi olla
  - mikä on pienin osuuden koko, joka voi olla epäjärjestyksessä?
  - mikä on pienin sitä vastaava i?

$\Rightarrow$ on valittu tulkinta ”tyhjä osuus on järjestyksessä”

- tyhjän osuuden alkiot ovat myös enintään yhtäsuuria kuin osuuden $A[i \ldots n]$ alkiot
  - ei ole olemassa liian suurta alkiota koska alkioita ei ole lainkaan

Mikä tahansa olemattomia otuksia koskeva ”jokainen”/”jokaisella”-väite on totta

- väite on totta, jos ja vain jos sille ei ole olemassa yhtään vastaesimerkkiä
- jokainen yksisarvinen pitää kärpässienistä
- jokainen yksisarvinen vihaa kärpässieniä
- jokaisella yksisarvisella on kolme sarvea

Jos vasen puoli ei voi toteutua, niin vasen $\Rightarrow$ oikea on aina pätevä

- silloin vastaesimerkkejä ei voi olla
- tämäkään ei ole niin hölmöä kuin miltä ehkä vaikuttaa
  - tilanne, jossa lähtöoletuksia on mahdoton täyttää, voi syntyä luonnollisesti samaan tapaan kuin tyhjä taulukon osa
  - ns. ristiriitatodistuksessa sellaisesta aloitetaan tarkoituksella, juuri siksi, että saamalla mahdoton seuraus osoitettaisiin lähtökohta mahdottomaksi

- siis esim. $x = x + 1 \Rightarrow 1 = 2$ on pätevä päättely
  - itse asiassa sen saa helposti tutuilla päättelysäännöillä!

    $x = x + 1$ vähennetään kummaltakin puolelta x

    $\Rightarrow 0 = 1$ lisätään kummallekin puolellle 1

    $\Rightarrow 1 = 2$

$\Rightarrow$ päättely voi olla pätevä, vaikka lopputulos olisi mahdoton!

- myös $x = x + 1 \Rightarrow x = 7$ on pätevä päättely
  - $x = x + 1 \Rightarrow 0 = 1 \Rightarrow 0(x - 7) = 1(x - 7) \Rightarrow 0 = x - 7 \Rightarrow x = 7$
- onko $x = x + 1 \Leftarrow x = 7$ pätevä päättely?
  - missä kohdassa ja miksi seuraava on väärin?

    $x = x + 1 \Leftarrow 0 = 1 \Leftarrow 0(x - 7) = 1(x - 7) \Leftarrow 0 = x - 7 \Leftarrow x = 7$
- myös $x^2 = -1 \Rightarrow x = 7$ on reaaliluvuilla pätevä päättely
  - reaaliluvuilla $x^2 \geq 0$, joten $-1 \geq 0$
  - tietenkin $-1 \leq 0$
  - jos $a \geq b$ ja $a \leq b$, niin $a = b$

  $\Rightarrow -1 = 0$

  $\Rightarrow -1(7 - x) = 0(7 - x) \Rightarrow x - 7 = 0 \Rightarrow x = 7$
- olisi liian vaikeaa rakentaa päättelysäännöt sellaisiksi, että mahdottomuudesta ei seuraisi mitä tahansa
- toisaalta tämä ilmiö ei tuota olemassaolevia asioita koskevia vääriä johtopäätöksiä

$\Rightarrow$ sen kanssa voi ja pitää elää!

Jos lähtöoletuksia on mahdoton täyttää, niin kaikki päättely on aina pätevää

- ei voi olla esim. tapausta, jossa lähtöoletukset ja vasen puoli ovat totta mutta oikea puoli ei ole totta, koska ei ole tapausta, jossa lähtöoletukset ovat totta
- tämänkin on paras olla näin, vaikka se voi tuntua hullulta
- tämä ei ole muuten tärkeä asia, mutta siitä on hyvä olla tietoinen siltä varalta, että siihen sattuu törmäämään

Tästä asiasta oli näin paljon puhetta siksi, että

- siihen törmää käytännössä (ja voi törmätä tentissä)
- se voi tuntua järjenvastaiselta, mikä on saanut osan opiskelijoista päättelemään väärin

Logiikan "ja" $\wedge$ ja "tai" $\vee$ ovat käteviä ilmaisemaan välivaiheita ja lopputuloksia

- esim. $x^2 + 4x - 12 = 0 \Leftrightarrow x^2 + 4x + 4 - 4 - 12 = 0 \Leftrightarrow (x+2)^2 - 16 = 0$
  $\Leftrightarrow (x+2)^2 = 16 \Leftrightarrow x + 2 = -4 \vee x + 2 = 4 \Leftrightarrow x = -6 \vee x = 2$
- esim. $|x| = 3x + 20$
  $\Leftrightarrow x < 0 \wedge -x = 3x + 20 \vee x \geq 0 \wedge x = 3x + 20$
  $\Leftrightarrow x < 0 \wedge x = -5 \vee x \geq 0 \wedge x = -10$
  $\Leftrightarrow x = -5$
- esim. $2x + 3y = 6 \wedge y - x = 7 \Leftrightarrow 2x + 3y = 6 \wedge y = x + 7 \Leftrightarrow$
  $2x + 3(x+7) = 6 \wedge y = x + 7 \Leftrightarrow 5x + 21 = 6 \wedge y = x + 7 \Leftrightarrow x = -3 \wedge y = 4$

Seuraavaksi

- perustelemme, että jokainen edellä olleista kolmesta ratkaisusta on pätevä
  - keskitymme yksityiskohtiin, joita hallitsevia periaatteita ei ole vielä esitelty
- käsittelemme siinä esiin nousevia uusia asioita

Yhteen- ja kertolaskulle pätee **osittelulaki**: kaikille reaaliluvuille a, b ja c pätee

$$a(b+c) = ab + ac$$

- sovellus niin että a:n paikalla on $(x+2)$, b:n paikalla on x ja c:n paikalla on 2:

$$(x+2)(x+2) = (x+2)x + (x+2) \cdot 2$$

- koska kertolaskun lopputulos ei riipu järjestyksestä eli kertolasku on **vaihdannainen**, pätee $(a+b)c = c(a+b)$ ja $ca + cb = ac + bc$
- osittelulain vuoksi $c(a+b) = ca + cb$

$\Rightarrow$ saamme $(a+b)c = ac + bc$, eli osittelulaki pätee myös toiselta puolelta

Miksi $x^2 + 4x - 12 = 0 \Leftrightarrow \ldots \Leftrightarrow x = -6 \vee x = 2$ on pätevä?

- $x^2 + 4x - 12 = 0 \Leftrightarrow x^2 + 4x + 4 - 4 - 12 = 0$
  - vasemmalle puolelle lisättiin 0 muodossa $4 - 4$
  - yhteenlaskun lopputulos on riippumaton laskun ryhmittelystä (tämä on niin tuttua että ei tarvitse eritellä nyt, ja toisaalta liian iso asia eriteltäväksi nyt)

  $\Rightarrow x^2 + 4x - 12 = x^2 + 4x + 4 - 4 - 12$

- $x^2 + 4x + 4 - 4 - 12 = 0 \Leftrightarrow (x+2)^2 - 16 = 0$
  - osittelulain vuoksi $(x+2)(x+2) = (x+2)x + (x+2)\cdot 2$,
    $(x+2)x = xx + 2x$ ja $(x+2)\cdot 2 = x\cdot 2 + 2\cdot 2$
    $\Rightarrow (x+2)^2 = (x+2)(x+2) = (x+2)x + (x+2)\cdot 2 =$
    $xx + 2x + x\cdot 2 + 2\cdot 2 = x^2 + 2x + 2x + 4 = x^2 + 4x + 4$
  - lisäksi käytettiin tietoa $-4 - 12 = -16$
- $(x+2)^2 = 16 \Leftrightarrow x + 2 = -4 \vee x + 2 = 4$
  - osittelulain vuoksi $(a+b)(a-b) = a^2 + ba - ab - b^2$
  - kertolaskun vaihdannaisuuden ym. vuoksi $ba - ab = ab - ab = 0$
    $\Rightarrow (a+b)(a-b) = a^2 - b^2$
  - reaaliluvuilla kertolaskun tulos on 0 jos ja vain jos ainakin yksi kerrottava on 0
    $\Rightarrow a^2 = b^2 \Leftrightarrow a^2 - b^2 = 0 \Leftrightarrow (a+b)(a-b) = 0 \Leftrightarrow a = -b \vee a = b$
  - tavoite saadaan sijoittamalla a:n paikalle $x+2$ ja b:n paikalle 4

Hyödyllistä jälkityöskentelyä

- yhtälön **juuri** (**root**) on luku, joka toteuttaa yhtälön
- sen, että -6 ja 2 ovat edellä käsitellyn yhtälön juuria, voi tarkastaa sijoittamalla
- mistä voi tietää, että ne ovat *kaikki* juuret?
- sekä sen että ne ovat juuria, että sen että ne ovat kaikki, voi tarkastaa seuraavasti:
  - huomaa periaatteen "kertolaskun tulos on 0 jos ja vain jos ..." käyttö

$$x = -6 \vee x = 2 \Leftrightarrow x + 6 = 0 \vee x - 2 = 0 \Leftrightarrow (x+6)(x-2) = 0$$
$$\Leftrightarrow x^2 + 6x + x\cdot(-2) + 6\cdot(-2) = 0 \Leftrightarrow x^2 + 4x - 12 = 0$$

Mistä hihasta $+4-4$ vedettiin?

- $x^2+4x+4-4-12=0$
- matematiikan sääntöjen käyttämiseen liittyy kaksi asiaa
  - käytetäänkö sääntöä oikein
  - viekö käyttö kohti tavoitetta
- esim. jalkapallossa
  - ei saa potkia vastapuolen pelaajaa nilkkaan
  - saa potkaista pallon omaan maaliin, mutta se ei kannata
- moniin perustehtäviin tiedetään menetelmä, joka on sääntöjen mukainen ja vie varmasti tavoitteeseen
  - ei tarvitse miettiä, sen kun vain laskee
  - esim. toisen asteen yhtälön ratkaisukaava: $x^2+4x-12=0 \Leftrightarrow$
    $x=\frac{-4\pm\sqrt{4^2-4\cdot1\cdot(-12)}}{2\cdot1}=\frac{-4\pm\sqrt{64}}{2}=\frac{-4\pm8}{2} \Leftrightarrow x=-6 \vee x=2$
- kaikkiin tilanteisiin ei kuitenkaan ole ratkaisukaavaa

$\Rightarrow$ vaikeammissa tehtävissä tarvitaan luovuutta ja kokemusta
  - aivan kuin shakkipelin siirtoa miettiessä

- vaikka kaava olisikin, soveltavissa tehtävissä pitää itse tietää, mitä kaavaa voi käyttää
- vertaa keittiöön:
  - teeveden keittämiseen vedenkeitin on helpompi kuin liesi ja kattila
  - kananmunien, lihakeiton yms. keittäminen ei onnistu vedenkeittimellä

- matematiikassa tulokset pitää esittää muodossa,
  jossa niiden *tarkastaminen* ei vaadi luovuutta
  - *keksiminen* voi vaatia luovuutta
  - päättely saa kulkea yllättäviä reittejä, kunhan sääntöjä käytetään oikein
  - yksi osa matematiikan viehätystä on nähdä yllättäviä päättelyitä
- edellä $+4-4$ oli vaihe menetelmässä nimeltä "neliöksi täydentäminen"
  - se on kuin hella ja kattila, ja ratkaisukaava on kuin vedenkeitin
  - sen avulla oli helppo perustella päättely oikeaksi
  - lukija voi tarkastaa esitetyn päättelyn vaikka ei osaisikaan neliöksi täydentämistä
  - ratkaisukaavaa käytettäessä olisi pitänyt perustella ratkaisukaava oikeaksi
- kokenut ruonlaittaja tuntee basilikan ja tietää mihin ruokaan se sopii
  - syöjät voivat nauttia hyvästä mausta vaikka eivät itse hoksaisi lisätä basilikaa
- kokenut matematiikan käyttäjä tuntee neliöksi täydentämisen
  ja osaa käyttää sitä silloin kun se hoitaa homman
- kuten sivulla 16 todettiin
  - ratkaisua etsiessä ei tarvitse noudattaa matematiikan sääntöjä, saa arvata jne.
  - löytynyt ratkaisu pitää kirjoittaa muotoon, joka noudattaa matematiikan sääntöjä

Miksi $|x| = 3x + 20 \;\Leftrightarrow\; x = -5$ on pätevä?

- $|x| = 3x + 20 \;\Leftrightarrow\; x < 0 \wedge -x = 3x + 20 \;\vee\; x \geq 0 \wedge x = 3x + 20$
  - seuraa suoraan itseisarvon määritelmästä

- $x < 0 \wedge -x = 3x+20 \ \vee \ x \geq 0 \wedge x = 3x+20 \ \Leftrightarrow \ x < 0 \wedge x = -5 \ \vee \ x \geq 0 \wedge x = -10$
  - ratkaistiin yhtälöt $-x = 3x + 20$ ja $x = 3x + 20$
- $x < 0 \wedge x = -5 \ \Leftrightarrow \ x = -5$, koska $-5 < 0$ on totta
  - x:n arvo -5 tekee molemmista puolista totta ja muut x:n arvot epätotta
- $x \geq 0 \wedge x = -10 \ \Leftrightarrow$ **F** , koska $-10 \geq 0$ ei ole totta
  - x:n arvo -10 tekee $x \geq 0$:sta epätotta ja muut x:n arvot $x = -10$:stä epätotta

  $\Rightarrow$ vasen puoli on epätosi kaikilla x:n arvoilla, joten vasen $\Leftrightarrow$ **F**
- $x = -5 \vee$ **F** $\ \Leftrightarrow \ x = -5$
  - aina pätee väite $\vee$ **F** $\Leftrightarrow$ väite

$\Rightarrow x < 0 \wedge x = -5 \ \vee \ x \geq 0 \wedge x = -10 \ \Leftrightarrow \ x = -5 \vee$ **F** $\ \Leftrightarrow \ x = -5$

$\vee$:n, $\wedge$:n ja **F**:n lakeja käsitellään perusteellisemmin luvussa 4

Miksi $5x + 21 = 6 \wedge y = x + 7 \ \Leftrightarrow \ x = -3 \wedge y = 4$ on pätevä?

- x:n arvolla -3 sekä $5x + 21 = 6$ että $x = -3$ tuottavat **T** eli tosi (true), joten
  - vasen puoli sievenee $5x + 21 = 6 \wedge y = x + 7 \ \Leftrightarrow$ **T** $\wedge y = -3 + 7 \ \Leftrightarrow \ y = 4$
  - oikea puoli sievenee $x = -3 \wedge y = 4 \ \Leftrightarrow$ **T** $\wedge y = 4 \ \Leftrightarrow \ y = 4$

  $\Rightarrow$ vasen $\Leftrightarrow$ oikea
- x:n muilla arvoilla sekä $5x + 21 = 6$ että $x = -3$ tuottavat **F**

  $\Rightarrow$ molemmat puolet tuottavat **F**

  $\Rightarrow$ vasen $\Leftrightarrow$ oikea

Miksi $2x + 3y = 6 \wedge y = x + 7 \Leftrightarrow 2x + 3(x + 7) = 6 \wedge y = x + 7$ on pätevä?

- tärkeä kysymys, sillä $2x + 3y = 6 \wedge y = x + 7 \Leftrightarrow 2x + 3(x + 7) = 6$ ei ole pätevä!
  - jos $x = -3$ ja $y = 0$, niin vasen puoli ei ole mutta oikea on tosi
  - $2x + 3y = 6 \wedge y = x + 7 \Rightarrow 2x + 3(x + 7) = 6$ on pätevä
- jos $y \neq x + 7$, niin ruskean väitteen kumpikin puoli tuottaa **F**
- jos $y = x + 7$, niin ruskean väitteen
  - vasen puoli tuottaa $2x + 3y = 6 \wedge$ **T** eli $2x + 3y = 6$
  - oikea puoli tuottaa $2x + 3(x + 7) = 6 \wedge$ **T** eli $2x + 3(x + 7) = 6$
  - koska $y = x + 7$, pätee $2x + 3y = 2x + 3(x + 7)$,
    joten $2x + 3y = 6 \Leftrightarrow 2x + 3(x + 7) = 6$

  $\Rightarrow 2x + 3y = 6 \wedge y = x + 7 \Leftrightarrow 2x + 3y = 6 \wedge$ **T** $\Leftrightarrow 2x + 3y = 6 \Leftrightarrow$
  $2x + 3(x + 7) = 6 \Leftrightarrow 2x + 3(x + 7) = 6 \wedge$ **T** $\Leftrightarrow 2x + 3(x + 7) = 6 \wedge y = x + 7$

$\Rightarrow$ ruskean väitteen molemmat puolet tuottavat aina saman totuusarvon

Yhteenlaskun, nollan, vastaluvun ja vähennyslaskun lait

- kaikki reaalilukujen yhteen- ja vähennyslaskun ominaisuudet voidaan johtaa näistä
- lait pätevät jokaisella reaaliluvulla a, b ja c
- muistathan, että etumerkki-$-$ on eri operaattori kuin vähennyslasku-$-$ (sivu 34)
  - toisella on argumentti vain perässä: $-b$
  - toisella sekä edellä että perässä: $a - b$

$$
\begin{aligned}
a + b &= b + a && \textbf{vaihdannaisuus (commutativity)} \\
(a + b) + c &= a + (b + c) && \textbf{liitännäisyys (associativity)} \\
a + 0 &= a && \textbf{nolla (zero)} \\
a + (-a) &= 0 && \textbf{vastaluku (additive inverse)} \\
a - b &= a + (-b) && \textbf{vähennyslasku (subtraction)}
\end{aligned}
$$

- nollaa kutsutaan myös yhteenlaskun neutraalialkioksi

Esimerkki: kullakin luvulla on vain yksi vastaluku

- oletetaan, että x on mikä tahansa sellainen luku, että $a + x = 0$
- todistamme, että $x = -a$:

$$
\begin{aligned}
& x && \text{nollan perusominaisuus} \\
= \; & x + 0 && \text{vaihdannaisuus} \\
= \; & 0 + x && \text{vastaluku} \\
= \; & (a + (-a)) + x && \text{vaihdannaisuus} \\
= \; & ((-a) + a) + x && \text{liitännäisyys} \\
= \; & (-a) + (a + x) && x\text{:ltä oletettu ominaisuus} \\
= \; & (-a) + 0 && \text{nollan perusominaisuus} \\
= \; & -a
\end{aligned}
$$

Vaihdannaisuudesta ja liitännäisyydestä seuraa, että
yhteenlaskun järjestyksellä ja ryhmittelyllä ei ole väliä

- tosin tuttuihinkin asioihin voidaan tarvita pitkiä päättelyitä!
- esimerkki: todistamme $a+b+c=c+b+a$
  - muistathan, että $+$ sitoo vasemmalle (sivu 30)

  $\Rightarrow$ $a+b+c$ tarkoittaa samaa lausekepuuta kuin $(a+b)+c$ ja eri kuin $a+(b+c)$

$$
\begin{aligned}
& a+b+c && + \text{ sitoo vasemmalle} \\
= \; & (a+b)+c && \text{vaihdannaisuus} \\
= \; & (b+a)+c && \text{vaihdannaisuus} \\
= \; & c+(b+a) && \text{liitännäisyys} \\
= \; & (c+b)+a && + \text{ sitoo vasemmalle} \\
= \; & c+b+a
\end{aligned}
$$

- esimerkki: todistamme edellisen avulla $a+b+c+d=d+c+b+a$

$$
\begin{aligned}
& a+b+c+d && + \text{ sitoo vasemmalle} \\
= \; & ((a+b)+c)+d && \text{edellä todistettu} \\
= \; & ((c+b)+a)+d && \text{vaihdannaisuus} \\
= \; & d+((c+b)+a) && \text{liitännäisyys} \\
= \; & (d+(c+b))+a && \text{liitännäisyys} \\
= \; & ((d+c)+b)+a && + \text{ sitoo vasemmalle} \\
= \; & d+c+b+a
\end{aligned}
$$

- käytännössä emme pura päättelyitä noin pieniksi askeliksi, vaan hyödynnämme yleisenä periaatteena, että
  - järjestyksellä ja ryhmittelyllä ei ole väliä
  - vähennyslasku on sama kuin yhteenlasku vähentäjän vastaluvulla

Kertolaskun, ykkösen, käänteisluvun ja jakolaskun lait

- kaikki reaalilukujen kerto- ja jakolaskun ominaisuudet voidaan johtaa näistä
- lait pätevät jokaisella reaaliluvulla a, b ja c

	ab	$=$	ba	**vaihdannaisuus (commutativity)**
	$(ab)c$	$=$	$a(bc)$	**liitännäisyys (associativity)**
	1	$\neq$	0	
	$a \cdot 1$	$=$	a	**ykkönen (one)**
jos $a \neq 0$, niin	$a \cdot \frac{1}{a}$	$=$	1	**käänteisluku (multiplicative inverse)**
jos $b \neq 0$, niin	$\frac{a}{b}$	$=$	$a \cdot \frac{1}{b}$	**jakolasku (division)**
	$a(b+c)$	$=$	$ab+ac$	**osittelulaki (distributivity)**

- ykköstä kutsutaan myös kertolaskun neutraalialkioksi

Vaihdannaisuus ja liitännäisyys toimivat samoin kuin yhteenlaskussa

$\Rightarrow$ kertolaskun järjestyksellä ja ryhmittelyllä ei ole väliä

- jakolasku eroaa vähennyslaskusta sikäli, että nollalla ei voi jakaa

Esimerkki: todistamme, että jokaisella a pätee $a \cdot 0 = 0$ ja $0a = 0$

- $a = a \cdot 1 = a(1 + 0) = a \cdot 1 + a \cdot 0 = a + a \cdot 0$ (ykkönen, nolla, osittelu, ykkönen)

$\Rightarrow 0 = a - a = (a + a \cdot 0) - a = (a \cdot 0 + a) - a = a \cdot 0 + (a - a) = a \cdot 0 + 0 = a \cdot 0$
(vastaluku, äskeinen tulos, vaihdannaisuus, liitännäisyys, vastaluku, nolla)

$\Rightarrow 0a = 0$ (vaihdannaisuus)

Esimerkki: todistamme että jos $ab = 0$, niin $a = 0 \vee b = 0$

- jos $b = 0$, niin $a = 0 \vee b = 0$
- muussa tapauksessa $b \neq 0$
  $\Rightarrow \frac{1}{b}$ on olemassa ja $b \cdot \frac{1}{b} = 1$
  $\Rightarrow a = a \cdot 1 = a(b \cdot \frac{1}{b}) = (ab) \cdot \frac{1}{b} = 0 \cdot \frac{1}{b} = 0$
  $\Rightarrow a = 0 \vee b = 0$

Edellä käytettiin erisuuruuden symbolia $\neq$

- $a \neq b$ on tosi jos ja vain jos $a = b$ on epätosi
  - saman voi ilmaista logiikan symbolilla $\neg$ eli ”ei”: $a \neq b \Leftrightarrow \neg(a = b)$
- toisinaan tarpeen yhtälöitä ratkaistaessa
  - pätee $x^2 - 4 = 5(x - 2) \Leftrightarrow x^2 - 5x + 6 = 0 \Leftrightarrow x = 2 \vee x = 3$
  - ei päde $\frac{x^2-4}{x-2} = 5 \Leftrightarrow x^2 - 4 = 5(x - 2) \Leftrightarrow x = 2 \vee x = 3$
  - pätee
    $\frac{x^2-4}{x-2} = 5 \Leftrightarrow x \neq 2 \wedge x^2 - 4 = 5(x - 2) \Leftrightarrow x \neq 2 \wedge (x = 2 \vee x = 3) \Leftrightarrow x = 3$

Määrittelemätön ei ole totta

- jos $x = 0$, niin mikään seuraavista ei ole tosi:
  $\frac{1}{x} = 0$, $\frac{1}{x} \neq 0$, $\frac{1}{x} < 0$, $\frac{1}{x} > 0$, $\frac{1}{x} \leq 0$ ja $\frac{1}{x} \geq 0$
- ne eivät myöskään ole epätosia, vaan niiden totuusarvo on "**undefined**" eli **U**

$\Rightarrow$ on kaksi eri tapaa, millä väite voi olla olematta tosi:
  - se on epätosi
  - se on määrittelemätön

- olennaisin ero:
  - $\neg$ **F** on **T** eli "ei epätosi" on tosi
  - $\neg$ **U** on **U** eli "ei määrittelemätön" on määrittelemätön
- sana "epätosi" oli huono valinta, koska se kuulostaa samalta kuin "ei tosi", mitä se siis ei tarkalleen ottaen tarkoita
- muuttujan arvo on aina määritelty, mutta lausekkeen ei välttämättä
  - esim. $x = x$ on aina totta, mutta $\frac{1}{x} = \frac{1}{x}$ ei ole totta kun $x = 0$
- yhtälön juurten ilmoittaminen tyyliin $\frac{x^2-4}{x-2} = 5 \Leftrightarrow x = 3$ sisältää kannanoton **U** $\Leftrightarrow$ **F**
  - kun $x = 2$, yhtälön vasen puoli tuottaa **U** ja oikea **F**
- siis vasen $\Leftrightarrow$ oikea sanoo vain, että vasen ja oikea ovat yhtäaikaa **T**
  - silloin kun ne eivät ole **T**, ne voivat olla erit: toinen **U** ja toinen **F**
- toisinaan tarvitsee olla tarkempi ja sanoa, että niillä on sama totuusarvo
- sen voi sanoa MathCheckissä vasen $\equiv$ oikea

Suuruusjärjestyksen lait

- kaikki suuruusjärjestyksen ominaisuudet voidaan johtaa näistä
- lait pätevät jokaisella reaaliluvulla a, b ja c

  täsmälleen yksi seuraavista pätee: $a < b$, $a = b$ tai $b < a$

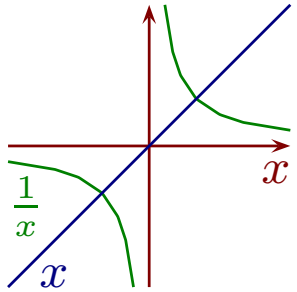
$$
\begin{aligned}
a < b \wedge b < c &\Rightarrow a < c \\
a < b &\Rightarrow a + c < b + c \\
0 < a \wedge 0 < b &\Rightarrow 0 < ab \\
a > b &\Leftrightarrow b < a \\
a \leq b &\Leftrightarrow a < b \vee a = b \\
a \geq b &\Leftrightarrow b \leq a
\end{aligned}
$$

Näistä voi johtaa neljä epäyhtälöiden ratkaisemisessa hyödyllistä lakia:

- $a < b \wedge c > 0 \Rightarrow ac < bc$
  - $a < b \wedge c > 0 \Leftrightarrow 0 < b - a \wedge 0 < c \Rightarrow 0 < (b - a)c = bc - ac \Leftrightarrow ac < bc$
- $a \leq b \wedge c > 0 \Rightarrow ac \leq bc$
  - edellisen tuloksen lisäksi havaitaan, että jos $a = b$, niin $ac = bc$
- $a < b \wedge c < 0 \Rightarrow ac > bc$
  - $a < b \wedge c < 0 \Leftrightarrow 0 < b - a \wedge 0 < -c \Rightarrow 0 < (b-a)(-c) = -bc + ac \Leftrightarrow ac > bc$
- $a \leq b \wedge c < 0 \Rightarrow ac \geq bc$

Siis kun epäyhtälön puolet kerrotaan

- positiivisella luvulla, niin vertailun suunta säilytetään
- negatiivisella luvulla, niin vertailun suunta käännetään



Esimerkki: $\frac{1}{x} \leq x$

- tapaus $x < 0$: $x < 0 \wedge \frac{1}{x} \leq x \;\Leftrightarrow\; x < 0 \wedge 1 \geq x^2 \;\Leftrightarrow\; -1 \leq x < 0$
- tapaus $x > 0$: $x > 0 \wedge \frac{1}{x} \leq x \;\Leftrightarrow\; x > 0 \wedge 1 \leq x^2 \;\Leftrightarrow\; x \geq 1$
- tapaus $x = 0$: ei tuota juuria, koska aiheuttaa nollalla jaon

$\Rightarrow$ juuret ovat $-1 \leq x < 0 \vee 1 \leq x$

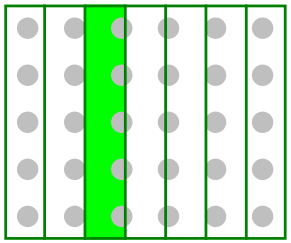
Voi johtaa myös esim. $\neg(a < b) \;\Leftrightarrow\; a \geq b$ ja $a \neq b \;\Leftrightarrow\; a < b \vee a > b$

Lueteltujen reaalilukujen lakien lisäksi on vielä ns. "täydellisyysaksiooma"

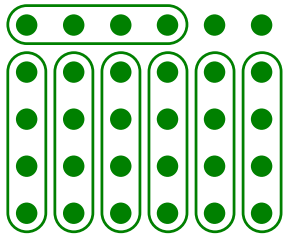
- on olemassa monta järjestelmää, joissa kaikki muut em. lait pätevät
  - esim. rationaaliluvut eli kokonaislukujen osamäärät (jakaja ei saa olla 0)
- täydellisyysaksiooma varmistaa
  - kaikki "lukusuoran luvut" ovat reaalilukuja, kuten rationaaliluvut, $\sqrt{2}$ ja π
  - muita reaalilukuja ei ole, ei esim. äärettömiä reaalilukuja
- täydellisyysaksiooma ei ole ihan helppo selittää

Koulussa ja ohjelmoinnissa on kahdenlaisia jakolaskuja

- reaalilukujen jakolasku: jaettava, jakaja ja tulos ovat reaalilukuja
  - koska kokonaisluvut ovat reaalilukuja, jaettava ja jakaja voivat olla kokonaislukuja
  - olennaista on, että tulos ei välttämättä ole kokonaisluku
  - esim. $\frac{30}{7} = 4\frac{2}{7} \approx 4{,}2857$



- kokonaislukujen jakolasku: kaikki osapuolet ovat kokonaislukuja
  - tulos muodostuu **osamäärästä** (**quotient**) ja **jakojäännöksestä** (**remainder**)
  - esim. 30 jaettuna 7:llä tuottaa osamääräksi 4 ja jakojäännökseksi 2
  - merkitsemme osamäärää $n \operatorname{div} m$ ja jakojäännöstä $n \bmod m$
  - esim. $30 \operatorname{div} 7 = 4$ ja $30 \bmod 7 = 2$



- $0 \leq$ jakojäännös $<$ |jakaja|
  - jos jakaja on 0, niin tämä kaava ei voi toteutua
  - ei sen tarvitsekaan, koska nollalla ei voi jakaa
- kokonaislukujaon osamäärä on reaalilukujaon osamäärä pyöristettynä alaspäin lähimpään kokonaislukuun

Lattiafunktio

- matematiikassa $\lfloor x \rfloor$ on suurin kokonaisluku, joka on enintään x

$\Rightarrow \lfloor x \rfloor \leq x < \lfloor x \rfloor + 1$

- lattiafunktion avulla ilmaistuna: $n \operatorname{div} m = \left\lfloor \frac{n}{m} \right\rfloor$

Kokonaislukujaolle pätee **jakoyhtälö**

$$n = m(n \operatorname{div} m) + (n \bmod m)$$

- $n \operatorname{div} m$ on osamäärä — n on jaettava
- $n \bmod m$ on jakojäännös — m on jakaja
- $0 \leq n \bmod m < |m|$

Jakojäännökselle pätee

$$\ldots = (n - m) \bmod m = n \bmod m = (n + m) \bmod m = (n + 2m) \bmod m = \ldots$$

Ohjelmointikielissä on yleistä, että

- jos `n` ja `m` ovat kokonaislukutyyppiä, niin `n/m` tarkoittaa $\approx$ kokonaislukujen jakoa
- esim. C++
  - `std::cout << "4/3  = " << 4/3  << '\n';` tulostaa `4/3  = 1`
  - `std::cout << "4/3. = " << 4/3. << '\n';` tulostaa `4/3. = 1.33333`

$\Rightarrow$ toisin kuin matematiikassa, kaikki kolmoset eivät ole samanlaisia!

- `n%m` tarkoittaa jakojäännöstä, ainakin jos `n` ja `m` ovat kokonaislukutyyppiä ja > 0
- jakoyhtälö pätee muuten, mutta $0 \leq$ `n%m` $< |$`m`$|$ ei aina päde, jos `n` < 0 tai `m` < 0
  - Javassa `-9%4` tuottaa -1
  - ennen 2011 C++ jätti avoimeksi, tuottaako `-9%4` 3 vai -1
- tästä asiasta on saatu aikaan melkoinen sotku, katso Wikipedia Modulo operation

Luvun lopuksi iso esimerkki: **Puolitushaku** (**binary search**) — taustaa

- nopea keino löytää alkio järjestetystä taulukosta
- haarukoi väliä, josta vastausta etsitään
  - kokeilee välin keskikohtaa
  - jos siinä on liian suuri, jatkaa alkupuolikkaalla
  - jos siinä on liian pieni, jatkaa loppupuolikkaalla
- vaikuttaa yksinkertaiselta!
- Jon Bentley kertoo kirjassaan Programming Pearls [Ben86], kuinka antoi tämän koodattavaksi yli sadalle ammattilaisohjelmoijille
  - pari tuntia aikaa
  - saivat valita itse ohjelmointikielen
  - melkein kaikki ilmoittivat onnistuneensa
  - Bentley testautti vastaukset: enintään 10 % oli oikein!
- Donald Knuth, Sorting and Searching [Knu73]:
  - puolitushaku julkaistiin ensimmäisen kerran 1946
  - virheetön puolitushaku julkaistiin ensimmäisen kerran 1962
- Richard Pattis, Textbook Errors in Binary Searching [Pat88]:
  - 20:stä oppikirjasta 15:ssa puolitushaku oli väärin
  - yhteensä 22 virhettä, joista 11 liittyi toimintaperiaatteen toteutukseen ja 11 väärään parametrinvälitysmekanismiin

$\Rightarrow$ on paikallaan katsoa, miten puolitushaku saadaan oikein logiikan avulla!

- ylivuotobugi
  - Joshua Bloch, blogi [Blo06]: Extra, Extra - Read All About It: Nearly All Binary Searches and Mergesorts are Broken
  - ainakaan Bentley ja Pattis eivät ottaneet sitä mukaan bugimääriin
  - aito virhe, joka esiintyy vain valtavilla taulukoilla

  $\Rightarrow$ käytännön merkitys pieni verrattuna edellä huomioon otettuihin virheisiin
  - sekin olisi vältettävissä todistustekniikoilla, mutta se tarina on liian pitkä tähän

Vertailun vuoksi lineaarinen haku

$a := 1$
while $a \leq n$ && $A[a] < x$ **do**
    $a := a + 1$

Puolitushaun spesifikaatio

- etsitään arvoa x taulukosta $A[1 \ldots n]$ ja vastaus pitää palauttaa muuttujassa a
- oletetaan $A[1] \leq A[2] \leq \ldots \leq A[n]$
- jos x esiintyy taulukossa, niin lopussa täytyy olla $1 \leq a \leq n \wedge A[a] = x$
- onko tämä riittävä spesifikaatio?
- mitä pitää palauttaa, jos x esiintyy taulukossa useasti?

- usein sallitaan palauttaa minkä tahansa esiintymän paikka
  - pieni etu: voidaan lopettaa heti kun tulee osuma
  - haitta: jos tarvitsee käsitellä kaikki osumat, joudutaan selaamaan löydetystä molempiin suuntiin

$\Rightarrow$ ensimmäisen esiintymän paikka on käyttäjälle kätevämpi
  - sen lineaarinen hakukin palauttaa

- mitä pitää palauttaa, jos x ei esiinny taulukossa lainkaan?
  - 0 kelpaa kertomaan, että ei löytynyt
  - lineaarinen haku palauttaa sen paikan, jossa "olisi pitänyt olla"
  - kätevää, jos etsitty halutaan lisätä taulukkoon
  - miten voi testata jälkikäteen, saatiinko osuma vai "olisi pitänyt olla tässä"?

$\Rightarrow$ valitsemme, että täytyy palauttaa se paikka, jossa "olisi pitänyt olla"

- algoritmin pitää siis palauttaa sama kuin minkä lineaarinen hakukin palauttaisi
  - jos x on suurempi kuin mikään taulukossa, niin lopussa $a = n + 1$
  - muutoin lopussa $1 \leq a \leq n$ ja a on pienin sellainen kokonaisluku, että $A[a] \geq x$

  $\Rightarrow$ kaikkiaan a on pienin sellainen kokonaisluku, että $a = n + 1 \vee A[a] \geq x$
  - miten sanotaan "pienin"?

Puolitushaun pseudokoodi sekä a:n ja y:n käyttäytyminen

	$\{0 \le n\}$	
1	$a := 1;\ y := n + 1$	$\{1 \le a \le y \le n + 1\}$
2	**while** $a < y$ **do**	$\{1 \le a < y \le n + 1\}$
3	$\quad v := (a + y) \operatorname{div} 2$	$\{1 \le a \le v < y \le n + 1\}$
4	$\quad$ **if** $A[v] < x$ **then** $a := v + 1$	$\{1 \le a \le y \le n + 1\}$
5	$\quad$ **else** $y := v$	$\{1 \le a \le y \le n + 1\}$
6	$\quad \{1 \le a \le y \le n + 1\}$	
7	$\{1 \le a = y \le n + 1\}$	

- jokaisessa taulukossa on ainakin nolla alkiota, joten aina $n \ge 0$
  $\Rightarrow$ rivin 1 lopussa $1 \le n + 1$
- kokonaistodistus tarvitsee väitteen, joka on voimassa *aina* rivin 2 alussa ja vie maaliin
  - rivien 2, . . . , 6 **while**-silmukan **invariantti**
  - kirjoittaja voi joutua tekemään paljon työtä sellaisen keksimiseksi
  - lukijalle riittää tarkastaa, että päättelyt ovat päteviä
  - tässä todistuksessa se väite on $1 \le a \le y \le n + 1$
- rivin 1 jälkeen $a = 1 \le n + 1 = y$
  $\Rightarrow 1 \le a \le y \le n + 1$
  - $1 = a \le y = n + 1$ ei toimisi kokonaistodistuksessa, koska se ei ole totta, kun riville 2 tullaan riviltä 6
- jos mennään **while**-silmukan vartaloon, niin myös silmukan ehto $a < y$ on totta

- rivillä 3 $a < y \;\Rightarrow\; a < \frac{a+y}{2} < y$
  - koska div pyöristää alaspäin, $v < y$
  - koska a on kokonaisluku ja div pyöristää alas lähimpään kokonaislukuun, $a \leq v$
- rivin 4 alussa $1 \leq v < y \leq n + 1$ ja luvut kokonaisia, joten $1 < v + 1 \leq y \leq n + 1$
  - jos mennään **then**-haaraan, niin sen lopussa $1 < a \leq y \leq n + 1$
- rivin 4 alussa $1 \leq a \leq v < n + 1$
  - jos mennään **else**-haaraan, niin sen lopussa $1 \leq a \leq y < n + 1$

$\Rightarrow$ molemmissa tapauksissa rivin 6 alussa $1 \leq a \leq y \leq n + 1$

$\Rightarrow$ riippumatta siitä mistä riville 2 tullaan, sen alussa $1 \leq a \leq y \leq n + 1$
  - sinne voidaan tulla riviltä 1 tai riviltä 6
  - olemme osoittaneet väitteen molemmissa tapauksissa

$\Rightarrow$ rivin 7 alussa eli algoritmin lopussa $1 \leq a = y \leq n + 1$
  - $1 \leq a \leq y \leq n + 1$, koska tultiin riviltä 2 muuttamatta mitään
  - koska **while**-silmukasta tultiin ulos, $a < y$ ei ole voimassa

Puolitushaun pysähtyminen

- a ei koskaan pienene eikä y kasva
- joka kierroksella a kasvaa tai y pienenee
- a ja y ovat kokonaislukuja

$\Rightarrow$ lopulta ehto $a < y$ lakkaa olemasta voimassa

$\Rightarrow$ algoritmi lopettaa

Tilanne algoritmin lopussa suhteessa A:han

- x:n ja A:n sisältö eivät muutu
  - mikään sijoituslause ei kohdistu niihin
- jos **then**-haaraa ei koskaan suoriteta, niin lopussa $a = 1$
- muutoin lopussa $A[a-1] < x$
  - kun a:ta viimeksi muutettiin, niin todettiin, että $A[v] < x$ ja sijoitettiin $a := v+1$
  - koska sijoituksessa A, v ja x eivät muuttuneet, sen jälkeen $a = v+1$, $a-1 = v$ ja $A[a-1] = A[v] < x$
- jos **else**-haaraa ei koskaan suoriteta, niin lopussa $y = n+1$
- muutoin lopussa $\neg(A[y] < x)$ eli $A[y] \geq x$
  - kun y:tä viimeksi muutettiin, niin todettiin, että $\neg(A[v] < x)$ ja sijoitettiin $y := v$

$\Rightarrow$ lopussa $1 \leq a = y \leq n+1 \wedge (a = 1 \vee A[a-1] < x) \wedge (y = n+1 \vee A[y] \geq x)$

- koska $a = y$, voidaan y:n paikalle kirjoittaa a
  $\Rightarrow 1 \leq a \leq n+1 \wedge (a = 1 \vee A[a-1] < x) \wedge (a = n+1 \vee A[a] \geq x)$
  - koska $y = a$ jätetään tässä sanomatta, tietoa pudotettiin, joten $\Leftrightarrow$ ei päde
- $1 \leq a \leq n+1$ sanoo, että vastaus on laillisella alueella
- $a = 1 \vee A[a-1] < x$ sanoo, että vastaus ei ole liian suuri
  - ts. mikään pienempi arvo ei voi olla oikea vastaus
  - 1 on pienin mahdollinen oikea vastaus, joten se ei ole liian suuri
  - muutoin $A[a-1] < x$ sanoo, että edellinen arvo on liian pieni

- $a = n + 1 \vee A[a] \geq x$ sanoo, että vastaus ei ole liian pieni
  - $n + 1$ on suurin mahdollinen oikea vastaus, joten se ei ole liian pieni
  - muutoin $A[a] \geq x$ sanoo, että tämä tai pienempi arvo on oikea

$\Rightarrow$ vastaus ei ole liian pieni eikä liian suuri, joten se on oikein!

Missä käytettiin tietoa $A[1] \leq A[2] \leq \ldots \leq A[n]$?

- kaavan todistuksessa ei käytetty

$\Rightarrow$ lopussa $1 \leq a \leq n + 1 \wedge (a = 1 \vee A[a - 1] < x) \wedge (a = n + 1 \vee A[a] \geq x)$
vaikka A ei olisi järjestyksessä!

- käytettiin kun pääteltiin, että kaavan mukainen kohta on haluttu kohta
- jos A on järjestyksessä, niin on täsmälleen yksi a, joka toteuttaa tämän kaavan
  - algoritmi löytää sen
  - jos x on A:ssa, niin ensimmäinen niistä on löydetyssä kohdassa
- jos A on epäjärjestyksessä, niin voi olla monta eri a, jotka toteuttavat tämän kaavan
  - algoritmi löytää niistä jonkin
  - vaikka x olisi A:ssa, niin se ei välttämättä ole löydetyssä kohdassa
  - esim. $A = [2,1,3]$, $x = 2$, $a = 3$
  - esim. $A = [3,2,1]$, $x = 2$, $a = 1$

$\Rightarrow$ kaava toteutuu riippumatta siitä onko A järjestyksessä,
mutta kaava takaa löytymisen vain jos A on järjestyksessä

Seuraavassa luvussa käsittelemme operaattorien $\wedge$, $\vee$ ja $\neg$ ominaisuuksia systemaattisesti

4 Propositiologiikka

Miksi tärkeä?

- propositiologiikka on kieli ja teoria
  - totuusarvojen funktioiden esittämiseen
  - totuusarvoista päättelemiseen
- yksinään se on aika tylsä ja keinotekoisen tuntuinen, mutta ...
- ... se on ilmaisuvoimaisempien logiikkojen perusta
- ... tietokoneiden peruspiirit ovat ymmärrettävissä sen avulla
- ... moni muukin aihealue on käännettävissä propositiologiikan kaavoiksi

$\Rightarrow$ sillä on ollut keskeinen merkitys sen tutkimisessa, mitä tehtäviä voidaan ja mitä ei voida ratkaista nopeasti tietokoneilla

- ja toki ohjelmoinnissa on hyödyksi osata loogista ajattelua ⌣̈

Kaikki tämän luvun asiat eivät ole itsessään tärkeitä, mutta ...

- ... on tärkeää touhuta propositiologiikan kanssa riittävästi, jotta siihen tulee tuntuma
- ... luvun päättelyt ovat hyödyllisiä esimerkkejä päättelemisestä

Logiikassa kaavoja merkitään usein pienillä kreikkalaisilla kirjaimilla φ (phi), ψ (psi) ja χ (chi)

4.1 Kaksiarvoinen eli "tavallinen" propositiologiikka

Propositiologiikan kaava voidaan rakentaa seuraavista osista:

- totuusarvovakiot **F** eli *epätosi* (*false*) ja **T** eli *tosi* (*true*)
- totuusarvoiset muuttujat eli *propositiot* P, Q, P', P_1, P_i ...,
- *negaatio* $\neg$ eli "ei"
  - jos φ on kaava, niin $\neg\varphi$ tarkoittaa päinvastaista kuin φ
  - esim. reaaliluvuilla $\neg(x < y)$ tarkoittaa samaa kuin $x \geq y$
    ($x < y$ ja $x \geq y$ ovat predikaattilogiikan mutta eivät propositiologiikan kaavoja)
- *konjunktio* $\wedge$ eli "ja"
  - $\varphi \wedge \psi$ tarkoittaa, että sekä φ pätee että ψ pätee
  - esim. reaaliluvuilla $x \geq 0 \wedge x \neq 0$ tarkoittaa samaa kuin $x > 0$
- *disjunktio* $\vee$ eli "tai"
  - $\varphi \vee \psi$ tarkoittaa, että φ pätee tai ψ pätee tai molemmat pätevät
  - esim. reaaliluvuilla $x \geq 0 \vee x < 1$ tarkoittaa samaa kuin **T**
- *totuusarvojen implikaatio* (tai vain *implikaatio*) $\rightarrow$ eli "jos ... niin"
  - $\varphi \rightarrow \psi$ tarkoittaa, että φ ei päde tai ψ pätee
  - esim. reaaliluvuilla $x > 0 \rightarrow x \geq 0$ tarkoittaa samaa kuin **T**
- *totuusarvojen ekvivalenssi* (tai vain *ekvivalenssi*) $\leftrightarrow$ eli "jos ja vain jos"
  - $\varphi \leftrightarrow \psi$ tarkoittaa, että molemmat pätevat tai kumpikaan ei päde
  - esim. reaaliluvuilla $x > 0 \leftrightarrow x > 1$ tarkoittaa samaa kuin $x \leq 0 \vee x > 1$

- sulut (ja)
  - käytetään ohjaamaan laskujärjestystä kuten koulumatematiikassakin

Sitovuussäännöt

- kuten tavallista, sitovuussäännöillä vähennetään sulkujen tarvetta
  - esim. $(\neg P) \vee (Q \wedge R) \Leftrightarrow \neg P \vee Q \wedge R$
- propositiologiikan sitovuussäännöt ovat vain osittain vakiintuneet

$\Rightarrow$ alla olevat pätevät tällä kurssilla kokonaan ja muualla osittain

- voimakkaimmin sitoo $\neg$, sitten $\wedge$, sitten $\vee$, sitten $\rightarrow$ ja heikoimmin sitoo $\leftrightarrow$
- $\rightarrow$ sitoo oikealle ja muut binäärioperaattorit vasemmalle
  - muut ovat liitännäisiä, joten niiden suunnalla on vain vähän merkitystä

Propositiologiikan kielioppi (tällä kurssilla)

Kaava	::=	Ekvivalenssikaava
Ekvivalenssikaava	::=	Implikaatiokaava | Ekvivalenssikaava $\leftrightarrow$ Implikaatiokaava
Implikaatiokaava	::=	Disjunktiokaava | Disjunktiokaava $\rightarrow$ Implikaatiokaava
Disjunktiokaava	::=	Konjunktiokaava | Disjunktiokaava $\vee$ Konjunktiokaava
Konjunktiokaava	::=	Negaatiokaava | Konjunktiokaava $\wedge$ Negaatiokaava
Negaatiokaava	::=	Atomikaava | $\neg$ Negaatiokaava
Atomikaava	::=	**F** | **T** | Muuttuja | (Kaava)

- huomaa implikaation poikkeava sitovuuden suunta!

Operaattoreiden tulokset

$\neg$	
F	T
T	F

$\wedge$	F	T
F	F	F
T	F	T

$\vee$	F	T
F	F	T
T	T	T

$\rightarrow$	F	T
F	T	T
T	F	T

$\leftrightarrow$	F	T
F	T	F
T	F	T

- $P \diamond Q$ luetaan riviltä P sarakkeesta Q
- siis **F** $\rightarrow$ **T** $\Leftrightarrow$ **T** ja **T** $\rightarrow$ **F** $\Leftrightarrow$ **F**
- muut binäärioperaattorit kuin $\rightarrow$ ovat vaihdannaiset

$\rightarrow$	F	T
F	T	(T)
T	(F)	T

Funktio on miten jokin asia riippuu nollasta, yhdestä tai useammasta asiasta

- henkilön syntymävuosi on henkilön funktio

$$\text{synt: Henkilöt} \rightarrow \text{Vuodet}$$

henkilö	Juho	Urho	Mauno	Martti	Tarja	Sauli
syntymävuosi	1870	1900	1923	1937	1943	1948

  - esim. $\text{synt(Tarja)} = 1943$
- elävän henkilön ikä vuoden lopussa on henkilön ja ko. vuoden funktio

$$\text{ikä: Elävät} \times \text{Vuodet} \rightarrow \text{Kokonaisluvut}$$

  - esim. $\text{ikä(Sauli, 2020)} = 72$

- funktioon "sisään meneviä" tietoja sanotaan funktion argumenteiksi tai parametreiksi
  - esim. funktion ikä argumentti on henkilö
  - esim. funktion synt argumentit ovat elävä henkilö ja vuosi
- funktion *ariteetti* (arity) on funktion argumenttien määrä
  - kullekin funktiolle kiinteä, mutta vaihtelee funktioiden välillä
  - esim. funktion synt ariteetti on 1 ja funktion ikä ariteetti on 2
  - ariteetin 0 funktiot ovat vakioita
- funktion käsitteelle olennaisia ovat nämä ja vain nämä:
  - ariteetti, esim. 2
  - argumenttien tyypit, esim. Henkilöt ja Vuodet
  - funktion tuloksen eli funktion arvon tyyppi, esim. Kokonaisluvut
  - mikä funktion arvo on kullakin argumenttien arvoyhdistelmällä, esim. elävän hlön ikä on tarkasteluvuosi miinus henkilön syntymävuosi eli $\text{ikä}(h, v) = v - \text{synt}(h)$
- kutakin argumenttien arvoyhdistelmää vastaa *täsmälleen yksi* funktion arvo
  - kenelläkään ei ole nollaa tai kahta syntymävuotta, vaan yksi
  - kenelläkään ei ole nollaa tai kahta ikää vuoden 2020 lopusssa, vaan yksi
- *mikä tahansa* nämä ehdot täyttävä on funktio
  $\Rightarrow$ funktion ei tarvitse olla esim. määritelty jollain säännöllä

Propositiologiikan kaava esittää *totuusfunktiota*

- totuusfunktio φ on funktio n totuusarvolta totuusarvolle eli $\varphi : \{\mathbf{F}, \mathbf{T}\}^n \to \{\mathbf{F}, \mathbf{T}\}$

Ariteetin 0 totuusfunktioita eli totuusarvovakioita on kaikkiaan 2 erilaista: **F** ja **T**

Ariteetin 1 totuusfunktioita $\varphi(P)$ on kaikkiaan 4 erilaista

P	
F	F
T	F

P	
F	F
T	T

P	
F	T
T	F

P	
F	T
T	T

Ariteetin 2 totuusfunktioita $\varphi(P, Q)$ on kaikkiaan 16 erilaista

P	Q	
F	F	F
F	T	F
T	F	F
T	T	F

P	Q	
F	F	F
F	T	F
T	F	F
T	T	T

P	Q	
F	F	F
F	T	F
T	F	T
T	T	F

P	Q	
F	F	F
F	T	F
T	F	T
T	T	T

P	Q	
F	F	F
F	T	T
T	F	F
T	T	F

P	Q	
F	F	F
F	T	T
T	F	F
T	T	T

P	Q	
F	F	F
F	T	T
T	F	T
T	T	F

P	Q	
F	F	F
F	T	T
T	F	T
T	T	T

P	Q	
F	F	T
F	T	F
T	F	F
T	T	F

P	Q	
F	F	T
F	T	F
T	F	F
T	T	T

P	Q	
F	F	T
F	T	F
T	F	T
T	T	F

P	Q	
F	F	T
F	T	F
T	F	T
T	T	T

P	Q	
F	F	T
F	T	T
T	F	F
T	T	F

P	Q	
F	F	T
F	T	T
T	F	F
T	T	T

P	Q	
F	F	T
F	T	T
T	F	T
T	T	F

P	Q	
F	F	T
F	T	T
T	F	T
T	T	T

Tämä totuusfunktion esitystapa on *totuustaulu*

- rivi jokaista muuttujien arvoyhdistelmää kohden
- sarake lopputulosta ja jokaista syötemuuttujaa kohden

Totuusfunktioita voi esittää myös propositiologiikan kaavoina

- esim. $P \vee \neg Q$ esittää saman totuusfunktion kuin tämä totuustaulu:
  - myös $Q \to P$ esittää saman totuusfunktion
  - myös $\neg(\neg P \wedge Q)$ esittää saman totuusfunktion

$\Rightarrow$ monta eri kaavaa voi esittää samaa totuusfunktiota

P	Q	
F	F	T
F	T	F
T	F	T
T	T	T

Totuusfunktioiden määrä kasvaa nopeasti ariteetin kasvaessa

- ariteetin 0 totuusfunktioita on kaksi: **F** ja **T**
- ariteetin 1 totuusfunktioita on neljä: **F**, P, $\neg P$ ja **T**
- ariteetin 2 totuusfunktioita on 16
- ariteetin 3 totuusfunktioita on 256
- ariteetin 4 totuusfunktioita on 65 536
- ariteetin 5 totuusfunktioita on 4 294 967 296
- ariteetin 6 totuusfunktioita on 18 446 744 073 709 551 616
- ariteetin 7 totuusfunktioita on 340 282 366 920 938 463 463 374 607 431 768 211 456
- ariteetin n totuusfunktioita on 2^{2^n}
  - muuttujien arvoyhdistelmiä on 2^n
  - kullekin niistä funktio tuottaa yhden arvon 2 vaihtoehdosta: **F** tai **T**
- huom! x^{y^z} lasketaan $x^{(y^z)}$ eikä $(x^y)^z$
  $\Rightarrow$ ariteetin 3 totuusfunktioita on $2^{(2^3)} = 2^8 = 256$ eikä $(2^2)^3 = 4^3 = 64$

Jokainen totuusfunktio voidaan ilmaista muuttujilla ja operaattoreilla **F**, $\neg$, $\wedge$ ja $\vee$

- ts. jokaisella $n \in \mathbb{N}$, jokainen $\varphi : \{\mathbf{F}, \mathbf{T}\}^n \to \{\mathbf{F}, \mathbf{T}\}$ voidaan ilmaista niillä
- jos funktio tuottaa **F** kaikilla syötteillä, sen ilmaisee kaava **F**
  - esim. ariteetti 2: $\varphi(P, Q) \Leftrightarrow \mathbf{F}$
  - ei haittaa, että kaava ei käytä kaikkia muuttujasymboleita P ja Q

P	Q	
F	F	F
F	T	F
T	F	F
T	T	F

- muussa tapauksessa funktion totuustaulussa on ainakin yksi rivi, joka loppuu **T**
- jokaista sellaista riviä kohti muodostetaan muuttujilla, $\neg$:lla ja/tai $\wedge$:lla kaava, joka toteutuu sillä ja vain sillä rivillä
  - esimerkin 1. rivi: $\neg P \wedge \neg Q$
  - esimerkin 2. rivi loppuu **F**, joten ei tehdä kaavaa
  - esimerkin 3. rivi: $P \wedge \neg Q$
  - esimerkin 4. rivi: $P \wedge Q$
  - erikoistapaus: ariteetin 0 vakiofunktio **T** voidaan muodostaa $\neg$ **F**

P	Q	
F	**F**	**T**
F	**T**	**F**
T	**F**	**T**
T	**T**	**T**

- ne yhdistetään $\vee$:llä
  - esimerkistä tulee $\neg P \wedge \neg Q \ \vee \ P \wedge \neg Q \ \vee \ P \wedge Q$

Toinen esimerkki

	P	Q	R	
	F	**F**	**F**	**F**
	F	**F**	**T**	**F**
$\neg P \wedge Q \wedge \neg R$	**F**	**T**	**F**	**T**
	F	**T**	**T**	**F**
$P \wedge \neg Q \wedge \neg R$	**T**	**F**	**F**	**T**
	T	**F**	**T**	**F**
$P \wedge Q \wedge \neg R$	**T**	**T**	**F**	**T**
$P \wedge Q \wedge R$	**T**	**T**	**T**	**T**

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

Tällä tavalla tulee usein pitkiä kaavoja!

- se on osittain tämän muodostustavan vika
- $\neg P \wedge \neg Q \ \vee \ P \wedge \neg Q \ \vee \ P \wedge Q$ voidaan ilmaista lyhyemmin $P \vee \neg Q$
- $\neg P \wedge Q \wedge \neg R \ \vee \ P \wedge \neg Q \wedge \neg R \ \vee \ P \wedge Q \wedge \neg R \ \vee \ P \wedge Q \wedge R$
  voidaan ilmaista lyhyemmin $\neg P \wedge Q \wedge \neg R \ \vee \ P \wedge (Q \vee \neg R)$
  - tarkastamme totuustaululla, että ne esittävät saman totuusfunktion

P	Q	R	$\neg P \wedge Q \wedge \neg R$	$Q \vee \neg R$	$P \wedge (Q \vee \neg R)$	koko kaava
F	F	F	F	T	F	F
F	F	T	F	F	F	F
F	T	F	T	T	F	T
F	T	T	F	T	F	F
T	F	F	F	T	T	T
T	F	T	F	F	F	F
T	T	F	F	T	T	T
T	T	T	F	T	T	T

  - oikein meni, mutta löytyisikö helpompi laskentakeino kuin totuustaulut?
- pitkien kaavojen syntyminen on vain osittain tämän muodostustavan vika
  - ariteetin n totuusfunktioita on paljon, paitsi jos n on hyvin pieni
  - lyhyitä kaavoja on vähän

  $\Rightarrow$ useimmille totuusfunktioille ei riitä lyhyttä kaavaa

Propositiologiikan lakeja

- kaikki alla mainitut pätevät, kun käytössä on vain totuusarvot **F** ja **T**
- luvussa 4.3 käyttöön otetaan **U** edustamaan määrittelemättömiä tapauksia
  - esim. $x + 0 = x \Leftrightarrow \mathbf{T}$, mutta $\frac{1}{x} + 0 = \frac{1}{x}$ ei ole **T** vaan **U** kun $x = 0$
- kun **U** on käytössä, harmaat lait lakkaavat pätemästä
- kun **U** on käytössä, vaaleansiniset lait jäävät voimaan annetussa muodossa, mutta rikkovat periaatetta "jos $P \Leftrightarrow Q$, niin $\varphi(P) \Leftrightarrow \varphi(Q)$"
  - esim. vaikka $\frac{1}{x} = 0 \wedge \frac{1}{x} \neq 0 \Leftrightarrow \mathbf{F}$, ei päde $\neg(\frac{1}{x} = 0 \wedge \frac{1}{x} \neq 0) \Leftrightarrow \neg\,\mathbf{F}$, koska kun $x = 0$, niin sen vasen puoli tuottaa **U** ja oikea **T**

  $\Rightarrow$ niiden käyttö on altis virheille
- kun **U** on käytössä, siniset lait jäävät voimaan ja niitä voi huoletta käyttää

Totuusarvovakioiden lakeja

- nämä lait ovat ilmeiset ja kannattaa osata sujuvasti
  - esim. $P \wedge \mathbf{F} \Leftrightarrow \mathbf{F}$ ilmaisee, että jokainen väite muotoa "alkuosa ja loppuosa", jossa loppuosa on epätosi, on epätosi
  - esim. aurinko paistaa ja Jyväskylä on Suomen pienin kaupunki on epätosi silloinkin kun aurinko paistaa

$$\mathbf{F} \wedge P \Leftrightarrow P \wedge \mathbf{F} \Leftrightarrow \mathbf{F} \qquad \mathbf{T} \wedge P \Leftrightarrow P \wedge \mathbf{T} \Leftrightarrow P$$
$$\mathbf{F} \vee P \Leftrightarrow P \vee \mathbf{F} \Leftrightarrow P \qquad \mathbf{T} \vee P \Leftrightarrow P \vee \mathbf{T} \Leftrightarrow \mathbf{T}$$

- nämä lait ovat ilmeiset ja välttämätön osata
  - esim. ei ole totta, että aurinko ei paista tarkoittaa samaa kuin aurinko paistaa

    $\neg\mathbf{F} \Leftrightarrow \mathbf{T} \qquad \neg\mathbf{T} \Leftrightarrow \mathbf{F}$
- näiden osaaminen ei ole välttämätöntä, koska, kuten kohta näytetään, $\rightarrow$ ja $\leftrightarrow$ voidaan eliminoida
  - $\leftrightarrow$ on hankalahko eliminoida ja ylemmät lait ovat toisinaan käteviä

  $\Rightarrow$ ehkä ne kannattaa painaa mieleen
  - esim. $P \leftrightarrow \mathbf{F}$:

    syön mämmiä jos ja vain jos lehmät lentää tarkoittaa samaa kuin en syö mämmiä

    $\mathbf{F} \leftrightarrow P \Leftrightarrow P \leftrightarrow \mathbf{F} \Leftrightarrow \neg P \qquad \mathbf{T} \leftrightarrow P \Leftrightarrow P \leftrightarrow \mathbf{T} \Leftrightarrow P$

    $\mathbf{F} \rightarrow P \Leftrightarrow \mathbf{T} \qquad P \rightarrow \mathbf{F} \Leftrightarrow \neg P \qquad \mathbf{T} \rightarrow P \Leftrightarrow P \qquad P \rightarrow \mathbf{T} \Leftrightarrow \mathbf{T}$

Toiston eliminointi

- nämä lait ovat ilmeiset ja välttämätön osata
  - esim. suklaa on hyvää tai suklaa on hyvää tarkoittaa samaa kuin suklaa on hyvää

    $\neg\neg P \Leftrightarrow P \qquad P \wedge P \Leftrightarrow P \qquad P \vee P \Leftrightarrow P$
- näiden osaaminen ei ole tarpeen, koska $\rightarrow$ ja $\leftrightarrow$ voidaan eliminoida
  - esim. jos on lämmintä niin on lämmintä on totta
  - esim. jos $\frac{2}{0} = 3$ niin $\frac{2}{0} = 3$ ei ole totta vaan määrittelemätön

    $P \rightarrow P \Leftrightarrow \mathbf{T} \qquad P \leftrightarrow P \Leftrightarrow \mathbf{T}$

Vaihdannaisuus ja liitännäisyys

- on ilmeistä ja kannattaa muistaa, että $\wedge$, $\vee$ ja $\leftrightarrow$ ovat vaihdannaisia
  - esim. aurinko paistaa ja tuulee tarkoittaa samaa kuin tuulee ja aurinko paistaa

$$P \wedge Q \;\Leftrightarrow\; Q \wedge P \qquad P \vee Q \;\Leftrightarrow\; Q \vee P \qquad P \leftrightarrow Q \;\Leftrightarrow\; Q \leftrightarrow P$$

- ne ovat myös liitännäisiä, mikä kannattaa muistaa ainakin $\wedge$:n ja $\vee$:n osalta

$$(P \wedge Q) \wedge R \;\Leftrightarrow\; P \wedge (Q \wedge R) \;\Leftrightarrow\; P \wedge Q \wedge R$$
$$(P \vee Q) \vee R \;\Leftrightarrow\; P \vee (Q \vee R) \;\Leftrightarrow\; P \vee Q \vee R$$
$$(P \leftrightarrow Q) \leftrightarrow R \;\Leftrightarrow\; P \leftrightarrow (Q \leftrightarrow R) \;\Leftrightarrow\; P \leftrightarrow Q \leftrightarrow R$$

- ne ovat ilmaistavissa muodossa, jossa vaihdannaisuus ja liitännäisyys ovat ilmeiset
  - $P_1 \wedge P_2 \wedge \cdots \wedge P_n \Leftrightarrow \mathbf{T}$ jos ja vain jos jokaisella $1 \leq i \leq n$ pätee $P_i \Leftrightarrow \mathbf{T}$
  - $P_1 \vee P_2 \vee \cdots \vee P_n \Leftrightarrow \mathbf{T}$ jos ja vain jos ainakin yhdellä $1 \leq i \leq n$ pätee $P_i \Leftrightarrow \mathbf{T}$
  - $P_1 \leftrightarrow P_2 \leftrightarrow \cdots \leftrightarrow P_n \Leftrightarrow \mathbf{T}$ jos ja vain jos mikään P_i ei tuota **U** ja parillisella määrällä $1 \leq i \leq n$ pätee $P_i \Leftrightarrow \mathbf{F}$

$\rightarrow$	F	T
F	T	(T)
T	(F)	T

- implikaatiolle ei päde vaihdannaisuus, vaan

$$P \rightarrow Q \;\Leftrightarrow\; \neg Q \rightarrow \neg P$$

- implikaatiolle ei päde liitännäisyys
  - jos $P \Leftrightarrow Q \Leftrightarrow R \Leftrightarrow \mathbf{F}$, niin $(P \rightarrow Q) \rightarrow R \Leftrightarrow \mathbf{T} \rightarrow \mathbf{F} \Leftrightarrow \mathbf{F}$ mutta $P \rightarrow (Q \rightarrow R) \Leftrightarrow \mathbf{F} \rightarrow \mathbf{T} \Leftrightarrow \mathbf{T}$
  - tällainen pätee: $P \rightarrow (Q \rightarrow R) \;\Leftrightarrow\; P \wedge Q \rightarrow R$

Osittelulait

- hyvin tärkeitä, kannattaa varmistaa että osaa
- helppo muistaa samankaltaisuudesta aritmetiikan osittelulakiin $a(b+c) = ab + ac$
- toisin kuin aritmetiikassa, toimivat molemmin päin
  - aritmetiikassa $a(b+c) = ab + ac$, mutta pääsääntöisesti $a + bc \neq (a+b)(a+c)$
- esim. ruskea osa ilmauksesta en lähde ulos, jos on kylmä ja sataa tai tuulee
  - se tulkitaan yleensä näin: on kylmä ja (sataa tai tuulee)
  - se tarkoittaa samaa kuin (on kylmä ja sataa) tai (on kylmä ja tuulee)

$$P \wedge (Q \vee R) \;\Leftrightarrow\; P \wedge Q \vee P \wedge R \qquad P \vee Q \wedge R \;\Leftrightarrow\; (P \vee Q) \wedge (P \vee R)$$

de Morganin lait

- hyvin tärkeitä, kannattaa varmistaa että osaa
- ei ole niin, että sataa tai tuulee tarkoittaa samaa kuin ei sada ja ei tuule

$$\neg(P \wedge Q) \;\Leftrightarrow\; \neg P \vee \neg Q \qquad \neg(P \vee Q) \;\Leftrightarrow\; \neg P \wedge \neg Q$$

- de Morganin lakeja ja kaksoisnegaation poistoa käytetään usein "painamaan negaatiot alas"
  - esim. $\neg(\neg P \wedge Q \wedge \neg R \;\vee\; P \wedge \neg(\neg Q \wedge R))$
    $\Leftrightarrow\; \neg(\neg P \wedge Q \wedge \neg R) \;\wedge\; \neg(P \wedge \neg(\neg Q \wedge R))$
    $\Leftrightarrow\; (\neg\neg P \vee \neg Q \vee \neg\neg R) \;\wedge\; (\neg P \vee \neg\neg(\neg Q \wedge R))$
    $\Leftrightarrow\; (P \vee \neg Q \vee R) \;\wedge\; (\neg P \vee \neg Q \wedge R)$

- reaalilukujen vertailujen tapauksessa näin päästään negaatioista kokonaan eroon
  - esim. $\neg(n < 2 \wedge m > 5 \vee \neg(m < n \vee n = m))$
    $\Leftrightarrow \ (\neg(n < 2) \vee \neg(m > 5)) \wedge \neg\neg(m < n \vee n = m)$
    $\Leftrightarrow \ (n \geq 2 \vee m \leq 5) \wedge (m < n \vee n = m)$

Ekvivalenssin ja implikaation eliminointi

- näillä saa aina $\rightarrow$ ja $\leftrightarrow$ poistettua kaavoista

$\Rightarrow$ jos osaa käyttää näitä, niin ei tarvitse osata muita lakeja $\rightarrow$:lle ja $\leftrightarrow$:lle

$$P \leftrightarrow Q \ \Leftrightarrow \ (P \rightarrow Q) \wedge (Q \rightarrow P) \qquad P \rightarrow Q \ \Leftrightarrow \ \neg P \vee Q$$

Muita lakeja

- kaksiarvologiikassa jokainen väittämä on tosi tai epätosi, eikä molempia:

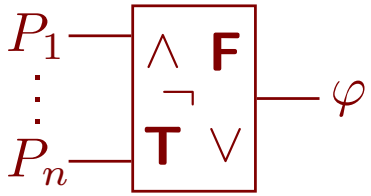
  $P \wedge \neg P \ \Leftrightarrow$ **F** $\qquad P \vee \neg P \ \Leftrightarrow$ **T**

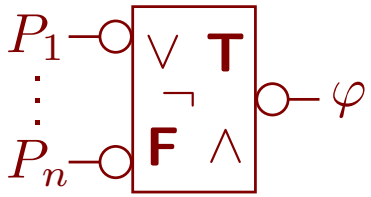
  - kolmiarvologiikassa (luku 4.3) väittämä voi olla myös määrittelemätön
- absorptiolakeja: $P \wedge (P \vee Q) \ \Leftrightarrow \ P \qquad P \vee P \wedge Q \ \Leftrightarrow \ P$
- nämä muuttuvat ohjelmoinnin kannalta mielenkiintoisiksi luvussa 4.3:

  $P \wedge (\neg P \vee Q) \ \Leftrightarrow \ P \wedge Q \qquad P \vee \neg P \wedge Q \ \Leftrightarrow \ P \vee Q$

Usein kaavan tai lain voi tarkastaa näppärästi seuraavasti

- valitse jokin muuttuja
- sijoita sen arvoksi **F**, sievennä ja tarkasta tulos
- sijoita sen arvoksi **T**, sievennä ja tarkasta tulos
- jos tuloksen tarkastaminen on vaikeaa, valitse toinenkin muuttuja
- esim. $P \vee Q \wedge R \Leftrightarrow (P \vee Q) \wedge (P \vee R)$
  - sijoitetaan $P \Leftrightarrow \mathbf{F}$
  - vasen puoli tuottaa $\mathbf{F} \vee Q \wedge R \Leftrightarrow Q \wedge R$
  - oikea puoli tuottaa $(\mathbf{F} \vee Q) \wedge (\mathbf{F} \vee R) \Leftrightarrow Q \wedge R$
  - tuli samat!
  - sijoitetaan $P \Leftrightarrow \mathbf{T}$
  - vasen puoli tuottaa $\mathbf{T} \vee Q \wedge R \Leftrightarrow \mathbf{T}$
  - oikea puoli tuottaa $(\mathbf{T} \vee Q) \wedge (\mathbf{T} \vee R) \Leftrightarrow \mathbf{T} \wedge \mathbf{T} \Leftrightarrow \mathbf{T}$
  - tuli samat!





F:n ja **T**:n symmetria

- kuten edellä nähtiin, $\rightarrow$ ja $\leftrightarrow$ voidaan eliminoida kaavoista
- de Morganin laeista seuraa, että jos kaava φ ei sisällä $\rightarrow$ eikä $\leftrightarrow$, niin $\neg\varphi$ saadaan vaihtamalla $\mathbf{F} \rightleftharpoons \mathbf{T}$, $\wedge \rightleftharpoons \vee$ ja $P \rightleftharpoons \neg P$ jokaiselle muuttujalle P

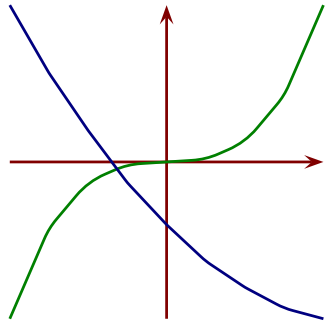
$\Rightarrow$ **F**:n ja **T**:n välillä vallitsee varsin vahva symmetria

- symmetria ei kuitenkaan yleensä yllä logiikkaa käyttäviin teorioihin
- esimerkki
  - on olemassa lukijalle helppo, jokaiselle ratkeavalle sudokulle toimiva keino todistaa, että sudokulla on ratkaisu: näytetään ratkaisu
  - todennäköisesti ei ole olemassa lukijalle helppoa, jokaiselle ratkeamattomalle sudokulle toimivaa keinoa todistaa, että sudokulla ei ole ratkaisua
- symmetria rikkoutuu luvussa 4.3 toisesta syystä
  - määrittelemätön voidaan samaistaa epätoteen mutta ei toteen

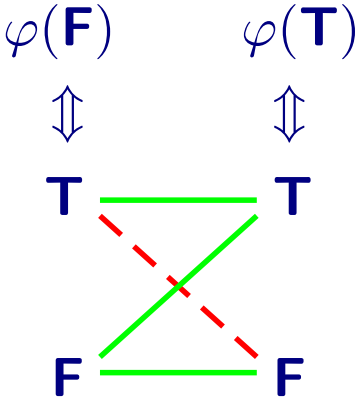
Kasvavuus kaksiarvologiikassa

- aritmetiikassa, jos f on funktio ja jokaisella x ja y, joilla $x < y$, pätee
  - $f(x) \leq f(y)$, niin f on kasvava — kuvaaja nousee tai kulkee vaakasuoraan
  - $f(x) < f(y)$, niin f on aidosti kasvava — kuvaaja nousee
  - $f(x) \geq f(y)$, niin f on vähenevä — kuvaaja laskee tai kulkee vaakasuoraan
  - $f(x) > f(y)$, niin f on aidosti vähenevä — kuvaaja laskee
- yhden muuttujan totuusfunktiolle vastaava on:

	F	P	$\neg P$	**T**
kasvava	×	×		×
aidosti kasvava		×		
vähenevä	×		×	×
aidosti vähenevä			×	



- kun sanomme, että kaava on kasvava / vähenevä muuttujan suhteen, niin tarkoitamme, että kaavan esittämä totuusfunktio on kasvava / vähenevä muuttujaa vastaavan argumentin suhteen
  - $\neg Q$ on kasvava P:n ja vähenevä sekä P:n että Q:n suhteen
  - jos $\varphi(P, Q) \Leftrightarrow \neg Q$, niin $\varphi(P, Q)$ on kasvava ensimmäisen ja vähenevä molempien argumenttiensa suhteen



- $\varphi(P_1, \ldots, P_n)$ on kasvava P_i:n suhteen, missä $1 \leq i \leq n$, jos ja vain jos kaikilla muuttujien $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n$ arvoilla
  - joko $\varphi(P_1, \ldots, P_{i-1}, \mathbf{F}, P_{i+1}, \ldots, P_n) \Leftrightarrow \mathbf{F}$
  - tai $\varphi(P_1, \ldots, P_{i-1}, \mathbf{T}, P_{i+1}, \ldots, P_n) \Leftrightarrow \mathbf{T}$
- toisin sanoen, kun P_i vaihtaa arvoa **F**:sta **T**:ksi ja muut P_j eivät muutu, niin $\varphi(P_1, \ldots, P_n)$ joko ei muutu tai vaihtaa arvoa **F**:sta **T**:ksi
- saman voi ilmaista myös näin:
  $\varphi(P_1, \ldots, P_{i-1}, \mathbf{F}, P_{i+1}, \ldots, P_n) \Rightarrow \varphi(P_1, \ldots, P_{i-1}, \mathbf{T}, P_{i+1}, \ldots, P_n)$
- vähenevyydessä tulos muuttuu tietysti toisinpäin tai ei ollenkaan:
  kun P_i vaihtaa arvoa **F**:sta **T**:ksi ja muut P_j eivät muutu,
  niin $\varphi(P_1, \ldots, P_n)$ joko ei muutu tai vaihtaa arvoa **T**:sta **F**:ksi
- esim. $P \wedge Q$ ja $P \vee Q$ ovat kasvavia sekä P:n että Q:n suhteen
  - kun niiden oheisissa tauluissa mennään oikealle tai alas, arvo ei muutu tai muuttuu **F**:sta **T**:ksi

$\wedge$	**F**	**T**
F	**F**	**F**
T	**F**	**T**

$\vee$	**F**	**T**
F	**F**	**T**
T	**T**	**T**

- esim. $P \to Q$ on vähenevä P:n suhteen ja kasvava Q:n suhteen
- esim. $P \leftrightarrow Q$ ei ole kasvava eikä vähenevä P:n eikä Q:n suhteen
  - yhdellä sarakkeella alas muutos **T**:sta **F**:ksi, toisella toisinpäin
  - yhdellä rivillä oikealle muutos **T**:sta **F**:ksi, toisella toisinpäin

$\to$	**F**	**T**
F	**T**	**T**
T	**F**	**T**

$\leftrightarrow$	**F**	**T**
F	**T**	**F**
T	**F**	**T**

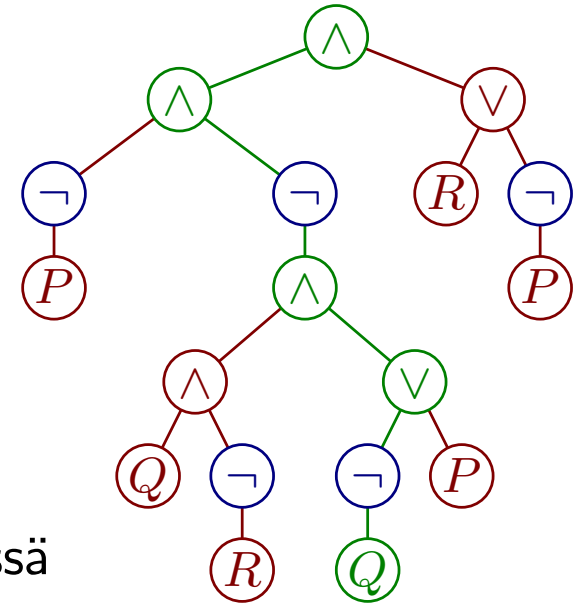
- ariteetin 2 totuusfunktio ei voi olla aidosti kasvava molempien argumenttien suhteen
  - $\varphi(\mathbf{T}, \mathbf{T})$:n pitäisi olla isompi kuin $\varphi(\mathbf{T}, \mathbf{F})$ ja sen isompi kuin $\varphi(\mathbf{F}, \mathbf{F})$, mutta on vain kaksi totuusarvoa
  - miksi ei voi olla aidosti kasvava toisen ja aidosti vähenevä toisen argumentin suhteen?

$\Rightarrow$ aito kasvavuus ja aito vähenevyys eivät ole kovin hyödyllisiä käsitteitä totuusfunktioille

Kasvavuus, vähenevyys ja $\neg$

- jos φ ja ψ ovat kasvavia P:n suhteen, niin myös $\varphi \wedge \psi$ ja $\varphi \vee \psi$ ovat kasvavia P:n suhteen
- jos φ ja ψ ovat väheneviä P:n suhteen, niin myös $\varphi \wedge \psi$ ja $\varphi \vee \psi$ ovat väheneviä P:n suhteen
- jos φ on kasvava P:n suhteen, niin $\neg\varphi$ on vähenevä P:n suhteen
- jos φ on vähenevä P:n suhteen, niin $\neg\varphi$ on kasvava P:n suhteen
- tietenkin P on kasvava itsensä suhteen

- jos P ei esiinny φ:ssä, niin φ on kasvava ja vähenevä P:n suhteen
  - φ ei muutu P:n muuttuessa
- oletamme, että kaavassa ei esiinny muita operaattoreita kuin $\neg$, $\wedge$ ja $\vee$
- sanomme, että muuttujan esiintymä on n:n negaation vaikutuspiirissä, jos ja vain jos lausekepuussa reitillä juuresta esiintymään on n $\neg$:ta
  - esimerkissä Q:lla on 2 esiintymää, joista toinen yhden ja toinen kahden negaation vaikutuspiirissä



$\Rightarrow$ jos muuttujan jokainen esiintymä on
  - parillisen määrän negaatioita vaikutuspiirissä, niin funktio on kasvava muuttujan suhteen
  - parittoman määrän negaatioita vaikutuspiirissä, niin funktio on vähenevä muuttujan suhteen

- esim.
  - P:n kaikki esiintymät ovat yhden negaation vaikutuspiirissä
  $\Rightarrow$ funktio on vähenevä P:n suhteen
  - R:n kaikki esiintymät ovat nollan tai kahden negaation vaikutuspiirissä
  $\Rightarrow$ funktio on kasvava R:n suhteen
  - Q:lla on esiintymä yhden ja toinen esiintymä kahden negaation vaikutuspiirissä
  $\Rightarrow$ ei saatu tietoa funktion kasvavuudesta / vähenevyydestä Q:n suhteen
- funktio voi olla kasvava tai vähenevä P:n suhteen, vaikka P esiintyisi sekä parittoman että parillisen määrän negaatioita vaikutuspiirissä
  - esim. $P \wedge \neg P$

Kuinka monta operaattoria tarvitaan esittämään kaikki totuusfunktiot?

- de Morganin laista seuraa $P \vee Q \Leftrightarrow \neg(\neg P \wedge \neg Q)$
- lisäksi tiedetään $\mathbf{T} \Leftrightarrow \neg\mathbf{F}$

$\Rightarrow$ jokainen totuusfunktio voidaan esittää muuttujilla ja operaattoreilla $\mathbf{F}$, $\neg$ ja $\wedge$

- jos muuttujia on ainakin yksi, niin $\mathbf{F}$ on tarpeeton, koska $\mathbf{F} \Leftrightarrow P \wedge \neg P$
- operaattorien $\neg$ ja $\wedge$ sijaan riittää $\mathrm{NAND}(P, Q) \Leftrightarrow \neg(P \wedge Q)$
  - $\neg P \Leftrightarrow \mathrm{NAND}(P, P)$
  - $P \wedge Q \Leftrightarrow \mathrm{NAND}(\ \mathrm{NAND}(P, Q), \mathrm{NAND}(P, Q)\ )$

Implikaatio on valittu sitomaan oikealle, koska

- silloin $P_1 \to \cdots \to P_{n-1} \to P_n \ \Leftrightarrow\ P_1 \wedge \cdots \wedge P_{n-1} \to P_n$
  - vrt. Hornin klausuuli sivu 108
- päinvastainen valinta ei johda yhtä hyödylliseen kaavaan

4.2 CNF, DNF, automatisoitu päättely ja P $\stackrel{?}{=}$ NP -ongelma

Tässä alaluvussa

- esitetään kaksi säännöllistä muotoa, joihin jokaisen kaavan voi muuntaa
  - muodoista on helppo nähdä tietyt asiat
  - hyödyllisiä automaattisessa päättelyssä
- pohditaan hieman propositiologiikan tehtävien laskennallista vaativuutta
- vilkaistaan propositiologiikan automatisoitua päättelyä

Sanastoa

- kaava φ on **tautologia** (**tautology**), jos ja vain jos $\varphi \Leftrightarrow$ **T**
  - esim. $P \vee \neg P \wedge Q \vee \neg(P \vee Q)$
  - φ:llä on vastaesimerkki, jos ja vain jos φ ei ole tautologia
- kaava φ on **ristiriita** (**contradiction**), jos ja vain jos $\varphi \Leftrightarrow$ **F**
  - esim. $\neg(P \vee \neg P \wedge Q \vee \neg(P \vee Q))$
  - esim. $\neg P \wedge (P \vee \neg Q) \wedge (P \vee Q)$
- kaava φ on **toteutettavissa** (**satisfiable**), jos ja vain jos se ei ole ristiriita
  - esim. $\neg P \vee Q$: toteutuu esim. kun $P \Leftrightarrow Q \Leftrightarrow$ **T**
  - esim. $P \vee \neg P$: toteutuu esim. kun $P \Leftrightarrow$ **F**
- **literaali** (**literal**) on propositiologiikan muuttuja tai sen negaatio
  - esim. P, $\neg Q$, P_1, $\neg P_i$

		koko kaava		
		$\neg(P \vee Q)$		
		$\neg P \wedge Q$		
P	Q			
F	F	F	T	T
F	T	T	F	T
T	F	F	F	T
T	T	F	F	T

- **klausuuli** (**clause**) on kaava, joka koostuu vain literaaleista ja $\vee$:sta
  - esim. Q, $P \vee \neg Q \vee \neg P$, $\neg P_1 \vee P_2 \vee P_4 \vee \neg P_5 \vee P_8$ tai $P \vee P$
- **ja-klausuuli** on kaava, joka koostuu vain literaaleista ja $\wedge$:sta
  - esim. Q, $P \wedge \neg Q \wedge \neg P$, $\neg P_1 \wedge P_2 \wedge P_4 \wedge \neg P_5 \wedge P_8$ tai $P \wedge P$
  - tälle piti keksiä oma nimi, koska kirjallisuudesta löytyneet aiheuttivat sekaannusta
- **konjunktiivinen normaalimuoto** eli **CNF** (**conjunctive normal form**) on joko **F**, **T**, tai klausuuleista $\wedge$:lla muodostettu kaava
  - esim. **T**, $(P \vee \neg Q) \wedge \neg P \wedge (\neg P \vee R \vee \neg S)$, $\neg Q \wedge R$, $\neg Q \vee R$ tai Q
  - esim. seuraavat eivät ole CNF: $\neg(P \wedge Q)$, $\neg\neg P$, $P \vee Q \wedge R$ ja $P \to Q$
- **disjunktiivinen normaalimuoto** eli **DNF** (**disjunctive normal form**) on joko **F**, **T**, tai ja-klausuuleista $\vee$:lla muodostettu kaava
  - esim. **T**, $P \wedge \neg Q \vee \neg P \vee \neg P \wedge R \wedge \neg S$, $\neg Q \vee R$, $\neg Q \wedge R$ tai Q
  - esim. seuraavat eivät ole DNF: $\neg(P \vee Q)$, $\neg\neg P$, $P \wedge (Q \vee R)$ ja $P \to Q$
- Hornin klausuuli on klausuuli, jossa enintään yksi muuttuja on ilman $\neg$
  - esim. $\neg P \vee \neg Q \vee R \vee \neg S$
  - loogisesti yhtäpitävä kaava: $P \wedge Q \wedge S \to R$
  - logiikkaohjelmointikieli Prolog perustuu Hornin klausuuleihin

Kielioppina

$$
\begin{array}{rcl}
\text{Literaali} & ::= & \text{Muuttuja} \mid \neg\ \text{Muuttuja} \\
\text{Klausuuli} & ::= & \text{Literaali} \mid \text{Klausuuli} \vee \text{Literaali} \\
\text{Ja-klausuuli} & ::= & \text{Literaali} \mid \text{Ja-klausuuli} \wedge \text{Literaali} \\
\text{CNF} & ::= & \mathbf{F} \mid \mathbf{T} \mid \text{Klausuuli} \mid \text{CNF} \wedge \text{Klausuuli} \\
\text{DNF} & ::= & \mathbf{F} \mid \mathbf{T} \mid \text{Ja-klausuuli} \mid \text{DNF} \vee \text{Ja-klausuuli}
\end{array}
$$

Kaavan muunto CNF:ksi, joka esittää samaa totuusfunktiota

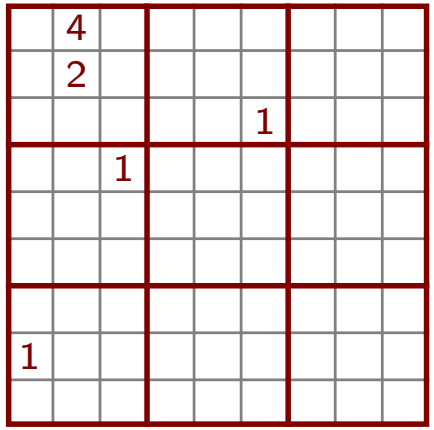
1. sovelletaan totuusarvovakioiden lakeja niin monta kertaa kuin mahdollista
   - jokainen soveltaminen lyhentää kaavaa $\Rightarrow$ prosessi ei voi jatkua loputtomiin
   - kun se päättyy, kaava joko ei sisällä vakioita tai on **F** tai **T**
   - jos kaava on **F** tai **T**, niin lopetetaan
2. sovelletaan ekvivalenssin eliminointilakia niin monta kertaa kuin mahdollista
   - soveltaminen ei tuota vakioita
   - jokainen vähentää $\leftrightarrow$:ien määrää $\Rightarrow$ prosessi ei voi jatkua loputtomiin
   - kun se päättyy, kaavassa ei ole vakioita eikä $\leftrightarrow$
   - soveltaminen voi pidentää kaavaa paljonkin, koska osakaavoista tehdään kopioita
3. sovelletaan implikaation eliminointilakia niin monta kertaa kuin mahdollista
   - soveltaminen ei tuota vakioita eikä $\leftrightarrow$
   - ... $\Rightarrow$ prosessi ei voi jatkua loputtomiin
   - kun se päättyy, kaavassa ei ole vakioita, $\rightarrow$ eikä $\leftrightarrow$
   - voi pidentää kaavaa, mutta vain vähän

4. sovelletaan de Morganin lakeja ja $\neg\neg$:n eliminointilakia
   niin monta kertaa kuin mahdollista
   - soveltaminen ei tuota ...
   - ... $\Rightarrow$ prosessi ei voi jatkua loputtomiin
   - kun se päättyy, kaavassa ei ole vakioita, $\rightarrow$ eikä $\leftrightarrow$,
     ja kaikki negaatiot kohdistuvat muuttujiin
   - de Morganin lain soveltaminen voi pidentää kaavaa, mutta vain vähän
   - $\neg\neg$:n poisto lyhentää kaavaa
5. sovelletaan osittelulakia $P \vee Q \wedge R \Leftrightarrow (P \vee Q) \wedge (P \vee R)$
   niin monta kertaa kuin mahdollista
   - soveltaminen ei tuota ... ... $\Rightarrow$ prosessi ei voi jatkua loputtomiin
   - kun se päättyy, kaava on CNF
   - soveltaminen voi pidentää kaavaa paljonkin, koska osakaavoista tehdään kopioita

- usein lopputulos optimoidaan
  - poistamalla klausuulit, joissa sama muuttuja esiintyy sellaisenaan ja negatoituna
    (jos kaikki klausuulit poistuvat, kaava $\Leftrightarrow$ **T**)
  - poistamalla saman literaalin moninkertaiset esiintymät samassa klausuulissa
  - poistamalla saman klausuulin moninkertaiset esiintymät kaavassa
- esimerkki: $P \leftrightarrow Q \;\Leftrightarrow\; (P \rightarrow Q) \wedge (Q \rightarrow P) \;\Leftrightarrow\; (\neg P \vee Q) \wedge (\neg Q \vee P)$
- esimerkki: $P \leftrightarrow Q \vee R \;\Leftrightarrow\; (P \rightarrow Q \vee R) \wedge (Q \vee R \rightarrow P)$
  $\Leftrightarrow\; (\neg P \vee Q \vee R) \wedge (\neg(Q \vee R) \vee P) \;\Leftrightarrow\; (\neg P \vee Q \vee R) \wedge (\neg Q \wedge \neg R \vee P)$
  $\Leftrightarrow\; (\neg P \vee Q \vee R) \wedge (\neg Q \vee P) \wedge (\neg R \vee P)$

Kaavan muunto DNF:ksi, joka esittää samaa totuusfunktiota

- muuten sama kuin muunto CNF:ksi, mutta vaiheessa 5 käytetään $P \wedge (Q \vee R) \Leftrightarrow P \wedge Q \vee P \wedge R$
- optimointikin on samankaltainen, mutta kaikkien ja-klausuulien poistuminen tuottaa **F**
- esimerkki: $P \leftrightarrow Q \Leftrightarrow \ldots \Leftrightarrow (\neg P \vee Q) \wedge (\neg Q \vee P)$
  $\Leftrightarrow \neg P \wedge \neg Q \vee \neg P \wedge P \vee Q \wedge \neg Q \vee Q \wedge P$
  $\Leftrightarrow \neg P \wedge \neg Q \vee P \wedge Q$

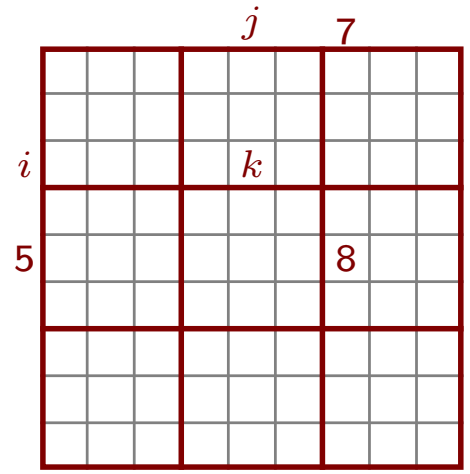


Usein propositiologiikkaa halutaan käyttää käsin tai automaattisesti

- todistamaan, että jokin kaava φ on aina totta eli tautologia
  - sama tavoite toisin sanottuna: että $\neg\varphi$ on ristiriita
  - sama tavoite toisin sanottuna: että φ:llä ei ole vastaesimerkkejä
  - esim. tietokoneohjelma toimii aina oikein, matemaattinen väite pätee
- löytämään muuttujille arvoyhdistelmä, jolla φ toteutuu
  - siis löytämään arvoyhdistelmä, jolla φ tuottaa **T**
  - sama tavoite toisin sanottuna: löytämään arvoyhdistelmä, jolla $\neg\varphi$ tuottaa **F**
  - sama tavoite toisin sanottuna: löytämään $\neg\varphi$:lle vastaesimerkki
  - esim. syöte jolla ohjelma toimii väärin, optimointitehtävän tai sudokun ratkaisu
- jälkimmäinen tehtävä voi onnistua jos ja vain jos $\neg\varphi$ ei ole tautologia
  - esim. sudoku voi olla ratkeamaton

Esimerkki: sudoku CNF:nä

- motivaatio
  - on olemassa ohjelmia sen selvittämiseen, onko CNF toteutettavissa (SAT solver)
  - käytämme sudokua esimerkkinä ongelman muuntamisesta niille sopivaan muotoon
- normaalikokoisessa sudokussa on 9 riviä ja 9 saraketta jaettuna 3×3 laatikoihin
  - 81 ruutua
  - indeksoimme rivit 1, . . . , 9 ja sarakkeet samoin
  - yleisessä tapauksessa n^2 ruutua



- jokaista ruutua kohti varataan 9 muuttujaa
  - $P_{i,j,k} \Leftrightarrow \mathbf{T}$ jos ja vain jos rivin i sarakkeessa j on arvo k
  - kuvassa $P_{5,7,8} \Leftrightarrow \mathbf{T}$
  - 729 muuttujaa
  - yleisessä tapauksessa n^3 muuttujaa
- samassa ruudussa saa olla vain yksi arvo
  - esim. ruudussa (3,5) ei ole sekä 4 että 7: $\neg(P_{3,5,4} \wedge P_{3,5,7})$
  - de Morganin kaavalla tästä tulee $\neg P_{3,5,4} \vee \neg P_{3,5,7}$

  $\Rightarrow$ kaikki tuplamerkinnät voidaan kieltää 36 klausuulilla ruutua kohden:

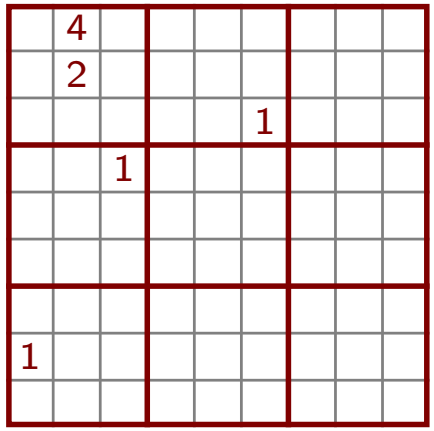
  $$(\neg P_{i,j,1} \vee \neg P_{i,j,2}) \wedge (\neg P_{i,j,1} \vee \neg P_{i,j,3}) \wedge \cdots \wedge (\neg P_{i,j,8} \vee \neg P_{i,j,9})$$

  - yleisessä tapauksessa $\frac{1}{2}n^3(n-1)$ klausuulia ja yhteensä $n^3(n-1)$ literaalia
  - tämä osataan ilmaista tiiviimmin $n^2(4n-7)$ klausuulilla ja $n^2(9n-16)$ literaalilla, jos sallitaan $n^2(n-2)$ muuttujaa lisää

  $\Rightarrow$ $\Theta(n^4)$ paranee $\Theta(n^3)$:ksi

- rivillä 8 on ainakin yksi kuutonen:
  $P_{8,1,6} \vee P_{8,2,6} \vee P_{8,3,6} \vee P_{8,4,6} \vee P_{8,5,6} \vee P_{8,6,6} \vee P_{8,7,6} \vee P_{8,8,6} \vee P_{8,9,6}$
  - sama tarvitaan jokaiselle arvolle
  - sama tarvitaan jokaiselle riville
  - sama tarvitaan sarakkeille ja laatikoille

  $\Rightarrow$ yleisessä tapauksessa $3n^2$ klausuulia ja yhteensä $3n^3$ literaalia



- ruutujen valmiit arvot on helppo ilmaista
  $0, \ldots, n^2$ klausuulilla ja literaalilla
  - esim. $P_{1,2,4} \wedge P_{2,2,2} \wedge P_{3,6,1} \wedge P_{4,3,1} \wedge P_{8,1,1}$

$\Rightarrow$ kaikkiaan $\frac{1}{2}n^4 - \frac{1}{2}n^3 + 3n^2 + n^2$ klausuulia ja $n^4 + 2n^3 + n^2$ literaalia

- normaalikokoisesta sudokusta tulee 3 159 + 81 klausuulia ja 8 019 + 81 literaalia
  - tämä olisi kohtuullinen määrä, jos olisi helppo selvittää, onko CNF toteutettavissa

"Ruudussa (i, j) on enintään yksi arvo" tiiviimmin

- otamme käyttöön apumuuttujat $Y_{i,j,2}, \ldots, Y_{i,j,n-1}$ ja sovimme, että $Y_{i,j,1}$ on $P_{i,j,1}$
- edellä näimme $P \leftrightarrow Q \vee R \;\Leftrightarrow\; (\neg P \vee Q \vee R) \wedge (\neg Q \vee P) \wedge (\neg R \vee P)$

$\Rightarrow$ klausuuleilla $(\neg Y_{i,j,k} \vee Y_{i,j,k-1} \vee P_{i,j,k}) \wedge (\neg Y_{i,j,k-1} \vee Y_{i,j,k}) \wedge (\neg P_{i,j,k} \vee Y_{i,j,k})$
taataan $Y_{i,j,k} \;\Leftrightarrow\; Y_{i,j,k-1} \vee P_{i,j,k} \;\Leftrightarrow\; P_{i,j,1} \vee P_{i,j,2} \vee \cdots \vee P_{i,j,k}$

$\Rightarrow$ tupla-arvot kieltää $(\neg Y_{i,j,1} \vee \neg P_{i,j,2}) \wedge (\neg Y_{i,j,2} \vee \neg P_{i,j,3}) \wedge \cdots \wedge (\neg Y_{i,j,n-1} \vee \neg P_{i,j,n})$

- $n-2$ muuttujaa, $4n-7$ klausuulia ja $9n-16$ literaalia vs. 0, $\frac{1}{2}n(n-1)$ ja $n(n-1)$

$\Rightarrow$ kaikkiaan $2n^3 - 2n^2$ muuttujaa, $4n^3 - 4n^2 + n^2$ klausuulia ja $12n^3 - 16n^2 + n^2$ liter.

Normaalimuotojen ominaisuuksia

- totuustaulusta saa helposti kirjoitettua DNF:n, joka esittää samaa totuusfunktiota
  - jokaisesta **T**-rivistä ja-klausuuli
- optimoidusta CNF:stä näkee helposti, onko kaava tautologia
  - kaava **T** on tietenkin tautologia
  - muunlainen optimoitu CNF ei voi olla tautologia
    (tässä on tärkeää, että klausuulit joissa on sekä P että $\neg P$ on poistettu)
- jälkimmäisessä tapauksessa $\neg\varphi$:n toteuttava arvoyhdistelmä löytyy
  - valitsemalla mikä tahansa klausuuli
  - sijoittamalla sen muotoa P olevien literaalien muuttujiin **F**, sen muotoa $\neg P$ olevien literaalien muuttujiin **T**, ja muihin muuttujiin mitä huvittaa
  - esim. $(P \vee \neg R) \wedge (\neg P \vee \neg Q \vee S)$

P	R	Q	S
F	**T**	F	F

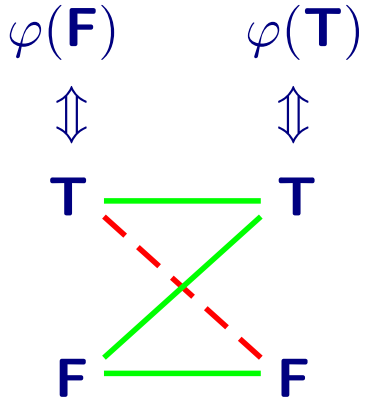
- optimoidusta DNF:stä näkee helposti, onko kaava ristiriita
  - kaava **F** on tietenkin ristiriita
  - muunlainen optimoitu DNF ei voi olla ristiriita
    (tässä on tärkeää, että ja-klausuulit joissa on sekä P että $\neg P$ on poistettu)
- jälkimmäisessä tapauksessa φ:n toteuttava arvoyhdistelmä löytyy ...
  - esim. $P \wedge \neg R \vee \neg P \wedge \neg Q \wedge S$

P	R	Q	S
T	**F**	F	F

Saadaanko siis mikä tahansa propositiologiikaksi käännetty tehtävä ratkaistua näppärästi muuntamalla kaava CNF:ksi tai DNF:ksi?

- ei, koska kaava voi paisua muunnoksessa liian suureksi

Voiko kaavan kahdentumisen $\leftrightarrow$:n eliminoinnissa estää?



- tarvittaessa kertaa kasvavuus / vähenevyys sivulta 102 alkaen
- $\varphi(P)$ on kasvava P:n suhteen, jos ja vain jos jokaisella sen muiden muuttujien kuin P arvoyhdistelmällä pätee $\neg\varphi(\mathbf{F}) \vee \varphi(\mathbf{T})$
- $\varphi(P)$ on vähenevä P:n suhteen, jos ja vain jos jokaisella sen muiden muuttujien kuin P arvoyhdistelmällä pätee $\varphi(\mathbf{F}) \vee \neg\varphi(\mathbf{T})$
- jos φ ja ψ ovat kasvavia, niin
  - $\varphi \wedge \psi$ ja $\varphi \vee \psi$ ovat kasvavia
  - $\neg\varphi$ on vähenevä
  - miten $\rightarrow$ käyttäytyy?
- jos φ ja ψ ovat väheneviä, niin
  - $\varphi \wedge \psi$ ja $\varphi \vee \psi$ ovat väheneviä
  - $\neg\varphi$ on kasvava
  - miten $\rightarrow$ käyttäytyy?

$\Rightarrow$ jos φ:ssä ei esiinny $\leftrightarrow$, ja P esiintyy φ:ssä enintään kerran, niin φ on kasvava tai vähenevä P:n suhteen

- $P \leftrightarrow Q$ ei ole kasvava eikä vähenevä P:n suhteen
  - kun $Q \Leftrightarrow \mathbf{F}$, niin $P \leftrightarrow Q$ ei ole kasvava P:n suhteen
  - kun $Q \Leftrightarrow \mathbf{T}$, niin $P \leftrightarrow Q$ ei ole vähenevä P:n suhteen

$\Rightarrow$ ei voida rakentaa $P \leftrightarrow Q$ siten, että P esiintyy vain kerran

- voidaan siten, että P esiintyy kahdesti: $P \leftrightarrow Q \;\Leftrightarrow\; (P \rightarrow Q) \wedge (Q \rightarrow P)$

Ehkä jokin parempi muunnos tuottaa aina lyhyestä syötteestä lyhyen lopputuloksen?

- tarkastellaan DNF:ää $P_1 \wedge Q_1 \vee \cdots \vee P_n \wedge Q_n$
  - käytämme esimerkkinä tapausta $n = 3$ eli DNF:ää $P_1 \wedge Q_1 \vee P_2 \wedge Q_2 \vee P_3 \wedge Q_3$
- se tuottaa **T** jos ja vain jos jollain i, jolle $1 \leq i \leq n$, pätee $P_i \Leftrightarrow \mathbf{T}$ ja $Q_i \Leftrightarrow \mathbf{T}$
  - esimerkiksi jos $i = 2$ eli $P_2 \Leftrightarrow \mathbf{T}$ ja $Q_2 \Leftrightarrow \mathbf{T}$, niin $P_1 \wedge Q_1 \vee P_2 \wedge Q_2 \vee P_3 \wedge Q_3 \Leftrightarrow P_1 \wedge Q_1 \vee \mathbf{T} \wedge \mathbf{T} \vee P_3 \wedge Q_3 \Leftrightarrow \cdots \vee \mathbf{T} \vee \cdots \Leftrightarrow \mathbf{T}$
- osoitamme, että vastaava optimoitu CNF kasvaa eksponentiaalisesti n:n funktiona

Aputulos: jokaisessa klausuulissa jokaisella $1 \leq i \leq n$ pitää olla literaali P_i tai Q_i

- todistusta varten olkoon $1 \leq i \leq n$
  - esimerkiksi $i = 2$
- kun $P_i \Leftrightarrow Q_i \Leftrightarrow \mathbf{T}$, DNF tuottaa **T**
  - esimerkiksi kun $P_2 \Leftrightarrow Q_2 \Leftrightarrow \mathbf{T}$, niin $P_1 \wedge Q_1 \vee P_2 \wedge Q_2 \vee P_3 \wedge Q_3 \Leftrightarrow \mathbf{T}$
- jos CNF:n klausuulissa ei esiinny literaali P_i eikä Q_i, niin se ja samalla koko CNF tuottaa **F**, kun $P_i \Leftrightarrow Q_i \Leftrightarrow \mathbf{T}$ ja muiden muuttujien arvot on valittu sopivasti
  - $\neg P_i$ ja $\neg Q_i$ saavat esiintyä, sillä ne tuottavat silloin **F**
  - millään $1 \leq j \leq n$ eivät sekä P_j että $\neg P_j$ voi esiintyä, koska CNF on optimoitu
  - esim. $P_1 \vee \neg Q_1 \vee \neg Q_2 \vee \neg P_3 \vee Q_3 \Leftrightarrow \mathbf{F}$, kun $P_1 \Leftrightarrow Q_3 \Leftrightarrow \mathbf{F}$ ja $P_3 \Leftrightarrow Q_1 \Leftrightarrow \mathbf{T}$

$\Rightarrow$ jollei P_i eikä Q_i ole klausuulissa, niin samalla syötteellä DNF tuottaa **T** ja CNF tuottaa **F**
$\Rightarrow$ DNF ja CNF eivät esitä samaa totuusfunktiota

Aputulos: jokaista symbolien $X_1, \ldots, X_n$, missä kukin X_i on joko P_i tai Q_i, yhdistelmää varten pitää olla klausuuli muotoa $X_1 \vee \cdots \vee X_n$ tai $X_1 \vee \cdots \vee X_n \vee$ jotain

- esimerkiksi jos $n = 3$, niin pitää olla nämä 8 klausuulia:

  $P_1 \vee P_2 \vee P_3 \vee \cdots$ $P_1 \vee Q_2 \vee P_3 \vee \cdots$ $Q_1 \vee P_2 \vee P_3 \vee \cdots$ $Q_1 \vee Q_2 \vee P_3 \vee \cdots$
  $P_1 \vee P_2 \vee Q_3 \vee \cdots$ $P_1 \vee Q_2 \vee Q_3 \vee \cdots$ $Q_1 \vee P_2 \vee Q_3 \vee \cdots$ $Q_1 \vee Q_2 \vee Q_3 \vee \cdots$

- oletamme ristiriidan johtamiseksi, että jokin $X_1 \vee \cdots \vee X_n \vee \cdots$ puuttuu
  - esimerkiksi $P_1 \vee Q_2 \vee P_3 \vee \cdots$ puuttuu: X_1 on P_1, X_2 on Q_2 ja X_3 on P_3
- jos X_i on P_i niin olkoon Y_i on Q_i ja jos X_i on Q_i niin olkoon Y_i on P_i
  - siis X_i ja Y_i ovat joko P_i ja Q_i tai Q_i ja P_i
  - esimerkissä Y_1 on Q_1 koska X_1 on P_1, Y_2 on P_2 koska X_2 on Q_2, ja Y_3 on Q_3 koska X_3 on P_3
- osoitamme, että jokaisessa klausuulissa on Y_1 tai Y_2 tai ... tai Y_n
  - edellä nähtiin, että jokaisessa klausuulissa jokaisella $1 \leq i \leq n$ on P_i tai Q_i

  $\Rightarrow$ jokaisessa klausuulissa on X_i tai Y_i, koska X_i ja Y_i ovat joko P_i ja Q_i tai Q_i ja P_i
  - oletettiin, että klausuuleja muotoa $X_1 \vee \cdots \vee X_n \vee \cdots$ ei ole

  $\Rightarrow$ jokaisessa klausuulissa on ainakin yksi Y_i
  - esimerkissä jokaisessa klausuulissa on Q_1, P_2 tai Q_3
- valitsemalla $X_1 \Leftrightarrow \cdots \Leftrightarrow X_n \Leftrightarrow$ **F** ja $Y_1 \Leftrightarrow \cdots \Leftrightarrow Y_n \Leftrightarrow$ **T** ristiriita näkyy
  - jokaisella $1 \leq i \leq n$ pätee $X_i \Leftrightarrow$ **F**, joten $P_i \wedge Q_i \Leftrightarrow X_i \wedge Y_i \Leftrightarrow$ **F**, joten DNF $\Leftrightarrow$ **F**
  - koska jokaisessa klausuulissa on ainakin yksi Y_i, CNF $\Leftrightarrow$ **T**

  $\Rightarrow$ CNF ja DNF eivät esitä samaa totuusfunktiota

- esimerkissä $P_1 \Leftrightarrow Q_2 \Leftrightarrow P_3 \Leftrightarrow \mathbf{F}$ ja $Q_1 \Leftrightarrow P_2 \Leftrightarrow Q_3 \Leftrightarrow \mathbf{T}$
  - DNF $\Leftrightarrow$ $P_1 \wedge Q_1 \vee P_2 \wedge Q_2 \vee P_3 \wedge Q_3 \Leftrightarrow \mathbf{F} \wedge \mathbf{T} \vee \mathbf{T} \wedge \mathbf{F} \vee \mathbf{F} \wedge \mathbf{T} \Leftrightarrow \mathbf{F} \vee \mathbf{F} \vee \mathbf{F} \Leftrightarrow \mathbf{F}$
  - CNF $\Leftrightarrow$ $(\cdots \vee Q_3 \vee \cdots) \wedge (\cdots \vee P_2 \vee \cdots) \wedge (\cdots \vee Q_3 \vee \cdots) \wedge \cdots \wedge (\cdots \vee Q_1 \vee \cdots) \Leftrightarrow (\cdots \vee \mathbf{T} \vee \cdots) \wedge (\cdots \vee \mathbf{T} \vee \cdots) \wedge (\cdots \vee \mathbf{T} \vee \cdots) \wedge \cdots \wedge (\cdots \vee \mathbf{T} \vee \cdots) \Leftrightarrow \mathbf{T} \wedge \mathbf{T} \wedge \mathbf{T} \wedge \cdots \wedge \mathbf{T} \Leftrightarrow \mathbf{T}$

Olemme osoittaneet, että jokainen DNF:ää $P_1 \wedge Q_1 \vee \cdots \vee P_n \wedge Q_n$ vastaava optimoitu CNF sisältää ainakin 2^n klausuulia, joista jokainen sisältää ainakin n muuttujaa

- jokaiselle valinnalle "X_i on P_i tai Q_i" ainakin yhden muotoa $X_1 \vee \cdots \vee X_n \vee \ldots$

$\Rightarrow$ DNF:ssä $P_1 \wedge Q_1 \vee \cdots \vee P_n \wedge Q_n$ on $4n - 1$ symbolia, ja sen kanssa saman totuusfunktion esittäminen CNF:llä vaatii ainakin $n2^{n+1} - 1$ symbolia

- on olemassa ääretön perhe DNF:iä, joilla on vain eksponentiaalisesti pitempiä CNF:iä
- eksponentiaalinen kasvu voi tehdä työmäärästä valtavan jo melko pienellä syötteellä

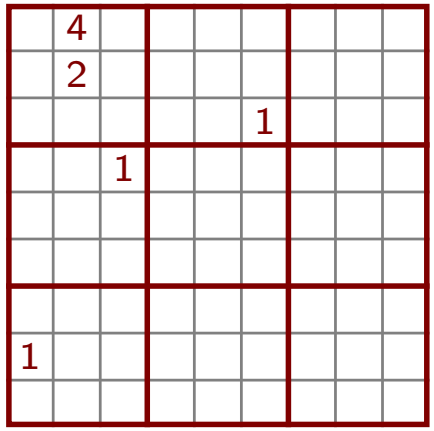
Mitä tämä merkitsee päättelyn automatisoinnille ja, kun ihminen päättelee, vaikeudelle?

- tämä *ei todista*, että päättely vaatii joskus paljon tilaa
  - onko kaava tautologia voidaan selvittää myös kokeilemalla kaikilla muuttujien arvoyhdistelmillä
  - ei vie paljon tilaa tietokoneessa (eikä liitutaululla, jota pyyhitään usein)
  - vie paljon aikaa

- tämä *ei todista*, että päättely vaatii joskus paljon aikaa
  - kenties on olemassa jokin parempi päättelykeino?
- tämä todistaa vain, että muuntaminen CNF:ksi (tai DNF:ksi) on joskus hyvin tilaa vievää (ja siksi hyvin tehotonta)
- vuosikymmenten tutkimus ei ole ratkaissut, onko olemassa jotain muuta, kaikilla syötteillä tehokasta tapaa selvittää, onko propositiologiikan kaava tautologia
  - tämä (täsmällisemmin muotoiltuna) on kuuluisa $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ -ongelma
- on kuitenkin paljon tuloksia, jotka viittaavat siihen, että vastaus on "ei"

Päätöstehtävät

- **päätöstehtävä** on kyllä/ei-kysymys
  - on sovittu jokin luonteva tapa esittää syöte tietokoneelle merkkijonona
  - esim. 9×9 sudoku: 81 merkkiä joukosta {1, 2, 3, 4, 5, 6, 7, 8, 9, -} tyyliin -4--------2-----------1-----1--------------------------------1----------
  - $n \times n$ sudoku on vaikeampi, mietimme sitä sivulla 122
  - tietokoneen pitää vastata "kyllä" tai "ei"
- esim. ratkaise syötteenä annettu sudoku ei ole päätöstehtävä
  - sen ratkaisu ei ole "kyllä" tai "ei" vaan luettelo siitä, mikä numero laitetaan mihinkin ruutuun
- esim. onko annetulla sudokulla ratkaisua on päätöstehtävä
  - kuvassa on esimerkki, jolle vastaus on "ei"



- esim. **CNF-SAT**: saadaanko annetun CNF:n totuusarvoksi **T** valitsemalla propositioille totuusarvot sopivasti
- tietojenkäsittelyteoriassa keskitytään päätöstehtäviin, koska
  - ne ovat yksinkertaisin tehtävälaji: teoria on kehno jollei hallitse edes niitä
  - sivulla 127 esiteltävä luokka **NP** on mielekäs vain niille
  - muita tehtäviä voidaan ratkaista niiden avulla kohtuullisella lisätyöllä
- esimerkki: sudokun ratkaiseminen sudokupäätöstehtävän avulla
  - aluksi kokeile, onko sudokulla lainkaan ratkaisua; jollei, niin lopeta
  - laita ensimmäiseen vapaaseen ruutun 1 ja kokeile, onko ratkaisu yhä olemassa
  - jollei, niin laita em. ruutun 2 eikä 1 ja kokeile, onko ratkaisua
  - jatka näin, kunnes em. ruudussa on luku, jolla on ratkaisu
  - selvitä samalla tavalla seuraavaan vapaaseen ruutuun sopiva luku, ja niin edelleen

Päätöstehtäväluokka **P**

- merkitsemme $|\sigma|$:lla syötteen pituutta laskettuna yksittäisinä merkkeinä
  - esim. 9×9 sudokulla $|\sigma| = 81$
- **polynomiajassa toimiva** tarkoittaa, että on olemassa sellainen polynomi $P(n)$, että algoritmin ajan kulutus on enintään $P(|\sigma|)$
  - $|\sigma|$:n pituisia syötteitä voi olla monta erilaista (ja yleensä onkin)
  - samanpituisilla eri syötteillä voi kulua eri määrä aikaa
  - suurimmankin ajan kulutuksen pitää olla enintään $P(|\sigma|)$
  - siis on olemassa sellainen $k \in \mathbb{N}$, että ajan kulutus on $O(|\sigma|^k)$

- **P** on niiden päätöstehtävien joukko, joille on olemassa polynomiajassa toimiva ratkaisualgoritmi
- esim. 9×9-sudokulle sellainen on, jopa vakioajassa toimiva:

  if syöte = "`-----------------------------------------------------------------------------`" **then T**
  else if syöte = "`-----------------------------------------------------------------------------1`" **then T**
  else if syöte = "`-----------------------------------------------------------------------------2`" **then T**
  . . .
  else if syöte = "`-4--------2-----------1-----1-----------------------------1----------------`" **then F**
  . . .
  else if syöte = "`-4--------2-----------1-----1-----------------------------2----------------`" **then T**
  . . .
  else if syöte = "`999999999999999999999999999999999999999999999999999999999999999999999999999999999`" **then F**

  – **if**-:ssä on `10000000000000000000000000000000000000000000000000000000000000000000000000000000000` haaraa
  – jokaista varten pitää ratkaista vastaava sudoku, jotta tietäisi onko lopussa **F** vai **T**

  ⇒ tämä algoritmi on käytännössä täydellisen mahdoton
  – se on silti olemassa ja olemassaololla todistaa, että 9×9 sudoku ratkeaa vakioajassa!
- 9×9 sudokun ratkeamisen vakioaikaisuus on käytännössä hyödytön ja järjetön tulos

  ⇒ haluamme siitä eroon
- se on silti tieteellisesti täysin pätevä ja aikuisen oikeesti tosi!

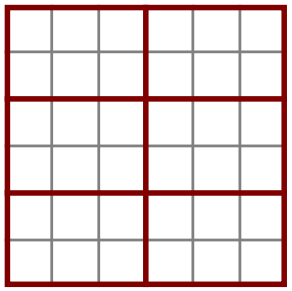
  ⇒ emme pääse siitä eroon noin vain
  – tuli tyhmä vastaus, koska kysymyksemme oli tyhmä

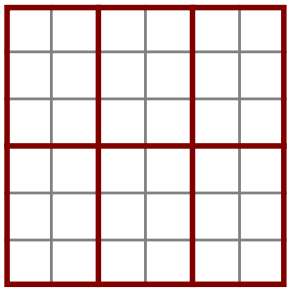
- tulokset muuttuvat mielekkäiksi, kun syötteitä voi olla äärettömän monta erilaista, mutta algoritmien ja syötteiden pitää silti olla äärellisiä
  - edellisen kaltainen **if**-lause ei kelpaa, koska siihen tulisi äärettömän monta haaraa

  $\Rightarrow$ algoritmi ei voi perustua valmiiden vastausten luetteloimiin, vaan sen pitää löytää vastaus "itse"

- esim. sudokua koskevat tulokset muuttuvat mielekkäiksi sallimalla syötteenä miten iso sudoku tahansa
  - ruudun sisältö ei enää voi olla merkki, vaan esim. 10-järjestelmässä kirjoitettu luku
  - kunpaa kuvaa 6×6 tarkoittaa?

  $\Rightarrow$ laatikoiden mitat täytyy ilmoittaa
  - esitystapa voi olla esim. $n^2 + 2$ välilyönneillä toisistaan erotettua kymmenjärjestelmän lukua, joista kaksi ensimmäistä kertoo pikkulaatikon koon
  - tällöin $|\sigma| \approx (n^2 + 2)(\log n + 2)$





- mielivaltaisen kokoinen sudokupäätöstehtävä ei voi ratketa vakioajassa
  - vakioaika ei riitä miten ison sudokun tahansa tapauksessa edes syötteen lukemiseen (puhumattakaan siitä, että ajan pitää riittää laskemiseenkin)
  - sudokun ratkeavuus voi riippua siitä osasta syötettä, jota ei ehditty lukea (on olemassa miten isoja tahansa ratkeavia sudokuja; mikä tahansa niistä voidaan muuntaa ratkeamattomaksi vaihtamalla kahteen viimeiseen ruutuun 1 ja 1)

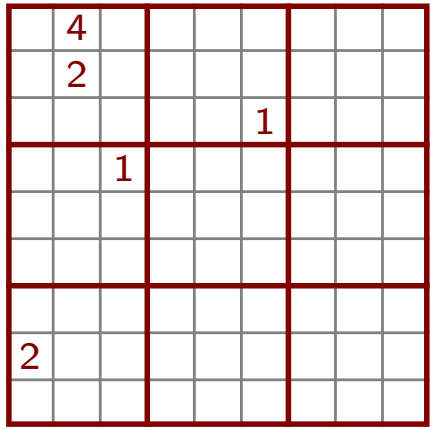
- ei tiedetä, ratkeaako se polynomiajassa
  - on hyvin vahvoja todisteita sille, että ei ratkea

Parturitehtävä

- syöte ja kysymys
  - parturilla on joukko vapaita aikoja
  - jokaisesta asiakkaasta tiedetään, mitkä ajat sopivat hänelle
  - yhden tukan leikkaamiseen menee yksi aika
  - onko olemassa aikataulu, jossa jokaisen asiakkaan tukka saadaan leikattua?
- parturitehtävä ei ole ihan helppo, mutta on silti todistetusti **P**:ssä
- jos asiakkaita on enemmän kuin aikoja, niin ajat eivät riitä, joten aikataulua ei voi olla
- vastaus voi olla "ei" vaikka aikoja olisi yhtä paljon kuin asiakkaita
  - esim. on kaksi asiakasta, joista kummallekin sopii vain aika numero 3
- "näille asiakkaille yhteensä sopivilla ajoilla" tarkoitamme niitä aikoja, jotka sopivat ainakin yhdelle mainituista asiakkaista
- esimerkki
  - asiakkaalle 2 sopivat ajat 4 ja 7
  - asiakkaalle 5 sopivat ajat 4, 9 ja 11
  - asiakkaalle 7 sopivat ajat 2, 8 ja 9

  $\Rightarrow$ asiakkaille 2, 5 ja 7 yhteensä sopivat ajat 2, 4, 7, 8, 9 ja 11
- jos asiakkaista löytyy osajoukko, jolle yhteensä sopivia aikoja on vähemmän kuin osajoukossa asiakkaita, niin aikataulua ei voi olla
  - esim. 6 asiakasta, joille yhteensä sopii vain 5 aikaa
- voidaan todistaa, että jos aikataulua ei voi olla, niin tällainen osajoukko on olemassa

Tarkastusjärjestelmän idea ihmisten avulla selitettynä



- kuvitellaan
  - valtavalla laskentakapasiteetilla varustettu todistaja
  - normaalilla laskentakapasiteetilla varustettu tarkastaja
- näin todistaja voi vakuuttaa tarkastajan, että jollekin sudokulle on ratkaisu:
  - hän ratkaisee sen itse
  - hän näyttää ratkaisun tarkastajalle
  - tarkastaja tarkastaa, että se noudattaa sudokun sääntöjä

1	4	3	2	5	6	7	8	9
5	2	6	7	8	9	1	3	4
7	8	9	3	4	1	2	5	6
3	5	1	4	6	2	8	9	7
4	6	2	8	9	7	3	1	5
8	9	7	1	3	5	4	6	2
6	1	4	5	2	3	9	7	8
2	3	5	9	7	8	6	4	1
9	7	8	6	1	4	5	2	3

- esim. alemman kuvan avulla voit kohtuullisella työllä varmistua, että ylemmän kuvan sudokulla on ratkaisu
- tämä tapa todistaa ratkaisun olemassaolo on täysin yleispätevä ja aina tarkastajalle kohtuullisen vähätöinen
  - toimii jokaikiselle sudokulle, jolle on ratkaisu
  - tarkastajan työmäärä on (käytännön laskutavalla) suoraan verrannollinen sudokun kokoon, mikä ei ole paljon
  - teoreetikot saavat tulokseksi isomman työmäärän, koska he ottavat huomioon, että isojen lukujen käsittely vie kauemmin kuin pienten
  - ero ei kuitenkaan vaikuta siihen, onko aika polynomiaalinen vai ei
  - polynomiaikaa käytetään juuri siksi, että se ei ole herkkä tällaisille eroille
- sama idea toimii myös parturitehtävälle, CNF-SAT:lle ja lukuisille muille tehtäville

- todistaja voi vakuuttaa tarkastajan myös siitä, että parturitehtävällä ei ole ratkaisua
  - sanoo tarkastajalle esim. että asiakkaille 2, 6, 8 ja 9 sopii yhteensä vain 3 aikaa
  - tarkastajan on helppo katsoa, mitkä ajat heille sopivat, ja laskea niiden määrä
  - jos joukolle yhteensä sopivia aikoja on vähemmän kuin joukossa asiakkaita, niin on ilmeistä, että aikataulua ei voi tehdä
  - tämä on aina tarkastajalle kohtuullisen vähätöistä
  - voidaan todistaa, että jokaisella ei-tapauksella on tällainen joukko
- olennaista on, että tarkastaminen on aina *tarkastajalle* vähätöistä
  - tarkastettavan aikataulun, asiakasjoukon tms. löytäminen saa olla vaikeaa
- olennaista on, että *todistajaan ei tarvitse luottaa*
  - jos tarkastus osoittaa, että ruudut on täytetty sudokun sääntöjen mukaisesti, niin ratkaisun olemassaolo on varmaa, riippumatta siitä mistä vastaus tuli
  - jos jokaiselle asiakkaalle on annettu hänelle sopiva aika eikä mitään aikaa ole tuplavarattu, niin se on pätevä aikataulu riippumatta siitä, kuka sitä ehdotti
  - vaikka viisi asiakasta olisi valittu arpomalla, niin jos sattuu olemaan niin, että heille sopii yhteensä vain neljä aikaa, niin pätevää aikataulua varmasti ei ole
- todisteessa saa olla seassa huuhaata, kunhan siitä löytyy vähällä työllä pätevä osuus
- tarkastusjärjestelmä voi olla tarkoitettu ”kyllä”-vastauksille tai ”ei”-vastauksille
- olennaista on, että menetelmä on *yleispätevä omalle vastauslajilleen*
  - ”kyllä”-menetelmän ei tarvitse toimia lainkaan, jos oikea vastaus on ”ei”
  - jos vastaus kysymykseen ”onko pätevää aikataulua olemassa” on ”kyllä”, niin pätevä aikataulu on varmasti olemassa

- pitääkö tarkastajan luottaa tähän:

  jos vastaus kysymykseen "onko pätevää parturiaikataulua olemassa" on "ei",
  niin k asiakkaan joukko, jolle yhteensä sopii alle k aikaa, on varmasti olemassa

  – ei tarvitse, tämä ei ole tarkastajan vaan todistajan murhe!
  – jos se ei olisi totta, niin todistaja voisi epäonnistua "ei"-vastauksen todistamisessa
  – silti jokainen tarkastuksen läpäisevä joukko todistaisi sitovasti, että "ei" on oikein

Tarkastusalgoritmit (**verification algorithm**) tietokoneissa

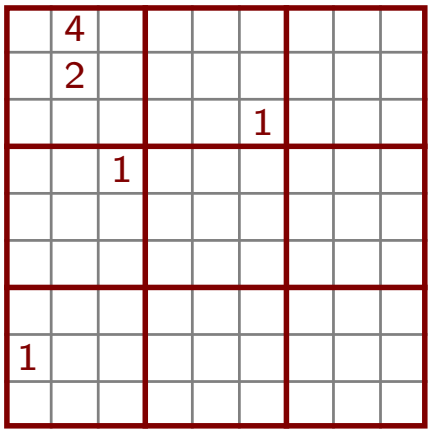
- päätöstehtävän tarkastusalgoritmi lukee kaksi tietoa:
  – päätöstehtävän syöte, esim. sudoku
  – merkkijono, jota kutsutaan **todisteeksi** (**certificate**),
  esim. ehdotus sudokun ratkaisuksi
- tarkastusalgoritmi vastaa "kyllä" tai "ei kelpaa"
- algoritmi suunnitellaan niin, että seuraavat toteutuvat:
  – jos algoritmi vastaa "kyllä", niin päätöstehtävän vastaus on varmasti "kyllä"
  – jos päätöstehtävän vastaus on "kyllä", niin on
  olemassa todiste, jolle algoritmi vastaa "kyllä"
- todisteeksi saa laittaa kelvottoman merkkijonon!
  – esim. sudokun virheellinen ratkaisu
  – ideana on, että tarkastusalgoritmi vastaa virheellisille todisteille "ei kelpaa"
  – todiste on pätevä annetulle syötteelle, jos ja vain jos algoritmi vastaa "kyllä"

Mitä tapahtuu, jos todisteena kokeillaan satunnaisesti valittuja merkkijonoja?

- jos niiden joukossa on pätevä todiste
  - saadaan ainakin kerran vastaus ”kyllä”

  $\Rightarrow$ tiedetään varmasti, että oikea vastaus on ”kyllä”
- jos niiden joukossa ei ole pätevää todistetta
  - saadaan joka kerta vastaus ”ei kelpaa”

  $\Rightarrow$ ei tiedetä, onko oikea vastaus ”ei” vai kävikö niin, että pätevä todiste on olemassa mutta se ei sattunut kokeiltuihin merkkijonoihin



Päätöstehtäväluokka **NP**

- sekä sudokussa että parturitehtävässä jokaiselle ”kyllä”-vastaukselle on olemassa tarkastajalle helppo pätevä todiste
  - sudokun virheetön ratkaisu
  - aikataulu, jossa ei ole tuplavarauksia ja jokaisen tukka saadaan leikattua
- parturitehtävässä jokaiselle ”ei”-vastaukselle on tarkastajalle helppo pätevä todiste
  - joukko asiakkaita, jolle yhteensä sopivia aikoja on liian vähän
- joistakin, *mutta ei kaikista*, sudokuista on helppo nähdä, että ratkaisua ei ole
- keksitkö *yleispätevän*, tarkastajalle kohtuullisen vähätöisen keinon todistaa, että sudokulla ei ole ratkaisua?
  - kukaan muukaan ei ole keksinyt

- **NP** on niiden päätöstehtävien joukko, joille on olemassa polynomiajassa toimiva ”kyllä”-vastausten tarkastusalgoritmi
- määritelmä ei ota kantaa siihen, kuinka helppoa on tarkastaa ”ei”-vastaukset
  $\Rightarrow$ se on epäsymmetrinen ”kyllä”:n ja ”ei”:n suhteen
- ihmisten tarkastusjärjestelmissä
  - jos tarkastaja osaa ratkaista tehtävän helposti itse, hän ei tarvitse todistetta
  - hän voi silti muodon vuoksi ottaa todisteen vastaan, mutta jättää sen tutkimatta
  - todisteessa saa olla seassa huuhaata, kunhan mukana on riittävä pätevä osuus
  - tässä tapauksessa tyhjä riittää päteväksi osuudeksi
- ratkaisualgoritmista saadaan tarkastusalgoritmi lisäämällä toinen parametri, jota algoritmi ei käytä, ja korvaamalla vastaus ”ei” vastauksella ”ei kelpaa”
  - määritelmä toimii tällöin samankaltaisesti kuin ihmisten tarkastusjärjestelmissä

  $\Rightarrow$ jos on olemassa polynomiajassa toimiva ratkaisualgoritmi, niin on olemassa polynomiajassa toimiva tarkastusalgoritmi

  $\Rightarrow$ $\mathbf{P} \subseteq \mathbf{NP}$ eli jokainen luokkaan **P** kuuluva tehtävä kuuluu myös luokkaan **NP**
- ei tiedetä, onko $\mathbf{NP} \subseteq \mathbf{P}$ eli onko $\mathbf{P} = \mathbf{NP}$
  - tämä on kuuluisa $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ -ongelma

Esimerkkejä tehtävistä joukosta **NP**, joiden ei uskota kuuluvan **P**:hen

- sudokun ratkaisu on todiste tehtävälle "onko tälle sudokulle olemassa ratkaisu?"
- tarpeeksi lyhyen reitin kuvaus on todiste tehtävälle "onko olemassa kaikkien kaupunkien kautta kulkeva reitti, jonka pituus on enintään tämä luku?"
- toteuttava muuttujien arvoyhdistelmä on todiste tehtävälle CNF-SAT

Esimerkkejä tehtävistä, jotka varmasti eivät kuulu **NP**:hen

- ilmoita mahdollisimman lyhyt kaikkien kaupunkien kautta kulkeva reitti
  - ei ole päätöstehtävä, eli vastaus ei ole "kyllä" tai "ei"
- tuottaako kaksi BNF-määritelmää saman kielen
  - tälle ei ole ratkaisualgoritmia, vaikka sallittaisiin miten hitaat tahansa
  - pätee jo tehtävälle "tuottaako BNF-määritelmä kaikki merkkijonot aakkostostaan"
- onko luonnollisia lukuja koskeva, korkeintaan muuttujilla ja symboleilla $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$, $\exists$, $=$, $($, $)$, $+$, 0 ja 1 ilmaistu väittämä tosi
  - kertolasku puuttuu!
  - jokainen tarkastusalgoritmi vaatii ainakin kahdesti eksponentiaalisesti aikaa
  - jos kertolasku lisätään, ratkaisualgoritmia ei ole, ei edes hyvin hidasta

Esimerkki tehtävästä, joka hyvin todennäköisesti ei kuulu **NP**:hen

- ”onko mahdollisimman lyhyen kaikkien kaupunkien kautta kulkevan reitin pituus tämä luku?”
- todisteeksi annettu reitti todistaa vain, että sen pituinen reitti on olemassa
- miten todistetaan, että lyhyempiä reittejä ei ole olemassa?

NP-kovat päätöstehtävät

- edellä näimme, miten sudokun voi muuttaa CNF:ksi polynomiajassa
- jos T on päätöstehtävä, niin tarkoittakoon $T(\sigma)$ sen oikeaa vastausta syötteellä σ
- päätöstehtävän T' **polynomiaikainen reduktio** T:ksi tarkoittaa polynomiaikaista algoritmia R merkkijonoilta merkkijonoille, jolle jokaiselle σ pätee $T'(\sigma) = T(R(\sigma))$
  - edellä esitettiin sellainen kun T on sudokupäätöstehtävä ja T' on CFN-SAT
  - oikea vastaus sudokupäätöstehtävälle syötteellä σ on sama kuin CNF-SAT:n vastaus sille CNF:lle, jonka reduktioalgoritmi tekee σ:sta
- jos CNF-SAT:n voi ratkaista polynomiajassa, niin sudokupäätöstehtävän voi ratkaista polynomiajassa
  - lasketaan $R(\sigma)$ eli muunnetaan sudoku CNF:ksi
  - lasketaan $T(R(\sigma))$ eli ratkaistaan näin saatu CNF-SAT-päätöstehtävä
- tehtäviä voi hämmästyttävän laajalti käyttää tällä tavalla ratkaisemaan toisia tehtäviä
- päätöstehtävä T on **NP-kova** (**NP-hard**), jos ja vain jos jokaiselle luokan **NP** päätöstehtävälle T' on olemassa polynomiaikainen reduktio T:ksi

NP-täydelliset päätöstehtävät

- päätöstehtävä on **NP-täydellinen**, jos ja vain jos se on **NP**:ssä ja **NP**-kova
- ei tiedetä, ratkeaako mikään **NP**-täydellinen päätöstehtävä polynomiajassa
- tiedetään, että jos yksikin niistä ratkeaa polynomiajassa, niin jokainen niistä ja jokainen **NP**:n tehtävä ratkeaa polynomiajassa
  - jos T on **NP**-täydellinen, niin T on **NP**-kova

  $\Rightarrow$ jokaiselle luokan **NP** tehtävälle T' on polynomiaikainen reduktio T:ksi

  $\Rightarrow$ T' ratkeaa polynomiajassa muuntamalla syöte T:n syötteeksi ja ratkaisemalla T

  $\Rightarrow$ jokainen luokan **NP** tehtävä ratkeaa polynomiajassa
  - jokainen **NP**-täydellinen tehtävä kuuluu **NP**:hen

  $\Rightarrow$ jokainen **NP**-täydellinen tehtävä ratkeaa polynomiajassa
- on olemassa **NP**-kovia tehtäviä, jotka eivät kuulu **NP**:hen
  - sellainen ei voi ratketa polynomiajassa
- jos yksikin **NP**-täydellinen kuuluu joukkoon **P**, niin **NP** $\subseteq$ **P** (ja niin ollen **P** $=$ **NP**)

Esimerkkejä **NP**-täydellisistä ja muista päätöstehtävistä

- onko tälle sudokulle ratkaisua?
- onko tässä kartassa reittiä, joka käy jokaisessa kaupungissa, ja jonka pituus on enintään tämä luku?
- CNF-SAT: onko tämä CNF toteutettavissa?

- 3-CNF-SAT: onko tämä CNF, jossa jokaisessa klausuulissa on tasan 3 literaalia, toteutettavissa?
  - 2-CNF-SAT $\in$ **P**
- SUBSET-SUM: on annettu luonnolliset luvut t ja $k_1, \ldots, k_n$; voidaanko jälkimmäisistä valita sellainen osa, että niiden summa on t?
- **NP**-täydellisiä tehtäviä tunnetaan sadoittain
- "onko tämä CNF ristiriita" ei todennäköisesti ole edes **NP**:ssä
  - se on muutoin sama tehtävä kuin CNF-SAT, mutta vastaukset "kyllä" ja "ei" ovat vaihtaneet paikkaa
  - toteuttava arvoyhdistelmä on todiste CNF-SAT:ille
  - usein on vaikea keksiä todiste tilanteelle "ei ole toteutettavissa"

$\Rightarrow$ näyttää siltä, että **NP** ei ole symmetrinen "kyllä":n ja "ei":n suhteen (toisin kuin **P**)

Esimerkkejä tehtävistä, joita usein väärin tai perusteettomasti väitetään **NP**:täydellisiksi

- tavallinen 9×9-sudoku
  - se ei ole päätöstehtävä $\Rightarrow$ se ei ole **NP**:ssä
- "onko tällä 9×9-sudokulla ratkaisua?"
  - sillä ei ole äärettömän monta erilaista syötettä $\Rightarrow$ se ratkeaa vakioajassa
- "etsi mahdollisimman lyhyt kaikkien kaupunkien kautta kulkeva reitti"
  - se ei ole päätöstehtävä $\Rightarrow$ se ei ole **NP**:ssä

- ”onko mahdollisimman lyhyen reitin pituus tämä luku?”
  - kuten edellä todettiin, tämä ei todennäköisesti ole **NP**:ssä

Ensimmäiset **NP**-täydellisyystodistukset koskivat tehtäviä SAT ja 3-CNF-SAT

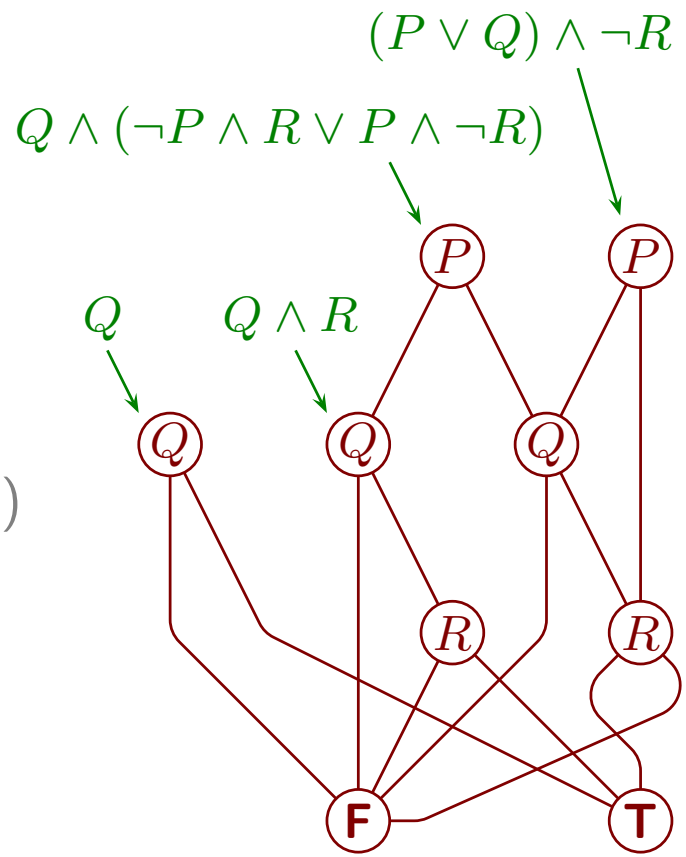
- Stephen Cook 1971
  - todistus perustuu tietokoneen (Turingin koneen) laskennan esittämiseen propositiologiikan kaavana
- samoihin aikoihin Leonid Levin keksi **NP**-täydellisyyden Neuvostoliitossa
  - julkaisu venäjäksi 1973, mutta ajatus oli esillä muutaman vuoden aikaisemmin
  - toisenlainen ongelma

Muita syitä uskoa, että $\mathbf{P} \neq \mathbf{NP}$

- moni päätöstehtävä on todistettu vaativammaksi kuin mikään **NP**:ssä
- tiedetään, että on olemassa päätöstehtäviä, jotka vaativat eksponentiaalisen ajan mutta ei enempää
- tunnetaan päätöstehtäviä, jotka ovat **NP**:ssä mutta näyttävät olevan olematta **P**:ssä tai **NP**-täydellisiä

Reduced ordered binary decision diagram eli ROBDD (tai vain BDD)

- tietorakenne totuusfunktioiden esittämiseen
- laajassa käytössä erityisesti mikropiiriteollisuudessa
- suunnattu silmukaton graafi ("diagram")
  - kuvassa suunta on alas (ei nuolenkärkiä)
- vakiofunktioita **F** ja **T** varten on solmut ("decision")
- jokaisessa muussa solmussa on
  - yksi muuttuja
  - kaksi kaarta eteenpäin: **F**-kaari ja **T**-kaari ("binary")
- muuttujille on valittu järjestys ("ordered")
  - solmun jälkeläisten muuttujat ovat järjestyksessä myöhemmin kuin solmun oma muuttuja
  - muuttujia saa jäädä välistä

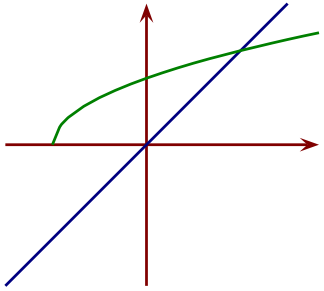


- totuusfunktiota esittää osoitin BDD:n solmuun
  $\Rightarrow$ sama BDD voi esittää useita totuusfunktioita
  - totuusfunktion arvo saadaan kulkemalla muuttujien arvojen määräämä polku
- samanlaiset aligraafit on yhdistetty ("reduced")
  - tehokasta ja melko helppoa hajautustaululla ja rekursiivisella algorimilla
  - onko $f(P_1, \ldots, P_n) \Leftrightarrow g(P_1, \ldots, P_n)$ selviää testillä `if( f_os == g_os )`
- jokaiselle perusoperaatiolle on tehokas algoritmi, mutta BDD:n koko voi kasvaa nopeasti

4.3 Totuusarvo "määrittelemätön"

Haluamme käyttää $\Leftrightarrow$:a ja $\Rightarrow$:a yhtälöiden ratkaisemisessa

- esimerkki: $\sqrt{x+2} = x$ — puolten pitää olla samanmerkkiset
  $\Leftrightarrow\ x \geq 0 \wedge \sqrt{x+2} = x$ — saa neliöidä puolittain, koska puolet ovat samanmerkkiset
  $\Leftrightarrow\ x \geq 0 \wedge x + 2 = x^2$ — vaihdetaan puolet
  $\Leftrightarrow\ x \geq 0 \wedge x^2 = x + 2$ — vähennetään $x+2$ molemmilta puolilta
  $\Leftrightarrow\ x \geq 0 \wedge x^2 - x - 2 = 0$ — ratkaistaan toisen asteen yhtälö
  $\Leftrightarrow\ x \geq 0 \wedge (x = 2 \vee x = -1)$ — käytetään osittelulakia
  $\Leftrightarrow\ x \geq 0 \wedge x = 2 \vee x \geq 0 \wedge x = -1$ — hylätään ehdon rikkova juuri
  $\Leftrightarrow\ x = 2$
- onko siis $\sqrt{x+2} = x \ \Leftrightarrow\ x = 2$?
  - kun $x > 2$, tulee **F** $\Leftrightarrow$ **F**
  - kun $x = 2$, tulee **T** $\Leftrightarrow$ **T**
  - kun $-2 \leq x < 2$, tulee **F** $\Leftrightarrow$ **F**
  - kun $x < -2$, on $\sqrt{x+2}$ määrittelemätön, joten mitä $\sqrt{x+2} = x$ tarkoittaa?



Ongelma ei ratkea päättämällä, että jokainen määrittelemätön vertailu tuottaa **F**

- muutoin, koska $\neg(a \leq b) \Leftrightarrow a > b$, voitaisiin päätellä molemmat alla olevista:
  - **T** $\Leftrightarrow$ $\neg$**F** $\Leftrightarrow$ $\neg(\frac{1}{0} > 0)$, joten $\neg(\frac{1}{0} > 0)$
  - **T** $\Leftrightarrow$ $\neg$**F** $\Leftrightarrow$ $\neg(\frac{1}{0} \leq 0) \Leftrightarrow \frac{1}{0} > 0$, joten $\frac{1}{0} > 0$

$\Rightarrow \frac{1}{0}$ sekä on että ei ole suurempi kuin 0!

- ongelmana on, että $\neg$ voi muuttaa määrittelemättömän vertailun tuottaman **F**:n **T**:ksi
- $\neg$(määrittelemätön) pitäisi olla määrittelemätön

$\Leftrightarrow$	**F**	**U**	**T**
F	$\times$	$\times$	$-$
U	$\times$	$\times$	$-$
T	$-$	$-$	$\times$

$\equiv$	**F**	**U**	**T**
F	$\times$	$-$	$-$
U	$-$	$\times$	$-$
T	$-$	$-$	$\times$

Ratkaisu

- jos aritmeettinen lauseke sisältää määrittelemättömän osan, niin lauseke kokonaisuudessaan katsotaan määrittelemättömäksi
- jos vertailun jompikumpi tai molemmat osapuolet ovat määrittelemättömiä, niin vertailu tuottaa totuusarvon **U** eli **määrittelemätön** (**undefined**)
- jotta $\Leftrightarrow$ toimisi yhtälöiden ratkaisemisessa, sovimme, että **U** $\Leftrightarrow$ **F** ja **F** $\Leftrightarrow$ **U**
  - nyt $\sqrt{x+2} = x \;\Leftrightarrow\; x = 2$ myös kun $x < -2$
- päättelyoperaattorit eivät tuota totuusarvoa, vaan päättelyaskel on tai ei ole pätevä
  - $\sqrt{x+2} = x \;\Leftrightarrow\; x = 2$ on pätevä: voi tuottaa vain **T** $\Leftrightarrow$ **T**, **F** $\Leftrightarrow$ **F** ja **U** $\Leftrightarrow$ **F**
  - $\frac{1}{x} \geq 0 \;\Leftrightarrow\; x \geq 0$ ei ole pätevä, koska kun $x = 0$, se tuottaa **U** $\Leftrightarrow$ **T**
- päättelyoperaattoreihin ei saa kohdistaa loogisia operaattoreita (eikä edes sulkuja)
  $\Rightarrow \neg$(**U** $\Leftrightarrow$ **F**) on syntaksivirhe
  $\Rightarrow$ vaikka **U** $\Leftrightarrow$ **F**, em. ongelmaa $\neg$:n kanssa ei synny
- tulemme jatkossa tarvitsemaan myös päättelyoperaattoria, joka ei samaista **F** ja **U**
  - sama totuusarvo
  - merkitsemme $\equiv$
  - jos $\varphi \equiv \psi$ niin $\varphi \Leftrightarrow \psi$, mutta ei välttämättä päinvastoin

Propositiologiikan operaattorien toiminta

$\neg$	
F	**T**
U	**U**
T	**F**

$\wedge$	**F**	**U**	**T**
F	**F**	**F**	**F**
U	**F**	**U**	**U**
T	**F**	**U**	**T**

$\vee$	**F**	**U**	**T**
F	**F**	**U**	**T**
U	**U**	**U**	**T**
T	**T**	**T**	**T**

$\rightarrow$	**F**	**U**	**T**
F	**T**	**T**	**T**
U	**U**	**U**	**T**
T	**F**	**U**	**T**

$\leftrightarrow$	**F**	**U**	**T**
F	**T**	**U**	**F**
U	**U**	**U**	**U**
T	**F**	**U**	**T**

- $\neg \mathbf{U} \equiv \mathbf{U} \wedge \mathbf{U} \equiv \mathbf{U} \vee \mathbf{U} \equiv \mathbf{U} \rightarrow \mathbf{U} \equiv \mathbf{U} \leftrightarrow \mathbf{U} \equiv \mathbf{U}$
- $\mathbf{U} \wedge \mathbf{F} \equiv \mathbf{F} \wedge \mathbf{U} \equiv \mathbf{F}$ ja $\mathbf{U} \wedge \mathbf{T} \equiv \mathbf{T} \wedge \mathbf{U} \equiv \mathbf{U}$
- $\mathbf{U} \vee \mathbf{F} \equiv \mathbf{F} \vee \mathbf{U} \equiv \mathbf{U}$ ja $\mathbf{U} \vee \mathbf{T} \equiv \mathbf{T} \vee \mathbf{U} \equiv \mathbf{T}$
- $\wedge$:n ja $\vee$:n nämä ja luvun 4.1 lait on helppo muistaa ajattelemalla, että
  - **F** on pienempi kuin **U** ja **U** on pienempi kuin **T**
  - $P \wedge Q$ tuottaa pienimmän
  - $P \vee Q$ tuottaa suurimman

$\Rightarrow$ kasvavuus ja vähenevyys toimivat kuten ennen **U**:n mukaantuloa

- edelleen $P \rightarrow Q$ tarkoittaa samaa kuin $\neg P \vee Q$
  - $\mathbf{F} \rightarrow \mathbf{U} \equiv \mathbf{U} \rightarrow \mathbf{T} \equiv \mathbf{T}$
  - $\mathbf{T} \rightarrow \mathbf{U} \equiv \mathbf{U} \rightarrow \mathbf{F} \equiv \mathbf{U}$
- edelleen $P \leftrightarrow Q$ tarkoittaa samaa kuin $(P \rightarrow Q) \wedge (Q \rightarrow P)$
  - $\mathbf{U} \leftrightarrow \mathbf{F} \equiv \mathbf{F} \leftrightarrow \mathbf{U} \equiv \mathbf{U} \leftrightarrow \mathbf{T} \equiv \mathbf{T} \leftrightarrow \mathbf{U} \equiv \mathbf{U}$

Havainto: taulujen jokaisella rivillä ja jokaisessa sarakkeessa joko keskellä on **U** tai koko rivi / sarake on samaa

$\vee$	**F**	**U**	**T**
F	**F**	**U**	**T**
U	**U**	**U**	**T**
T	**T**	**T**	**T**

- $\varphi(\mathbf{U}) \equiv \mathbf{U}$ tai $\varphi(P)$ ei riipu P:stä
- jokaisella Q:n arvolla $\varphi(\mathbf{U}, Q) \equiv \mathbf{U}$ tai $\varphi(P, Q)$ ei riipu P:stä
  - esim. $\varphi(P, Q) \equiv P \vee Q$: $\mathbf{U} \vee \mathbf{F} \equiv \mathbf{U}$, $\mathbf{U} \vee \mathbf{U} \equiv \mathbf{U}$ ja $P \vee \mathbf{T}$ ei riipu P:stä
- jokaisella P:n arvolla $\varphi(P, \mathbf{U}) \equiv \mathbf{U}$ tai $\varphi(P, Q)$ ei riipu Q:stä
  - esim. $\varphi(P, Q) \equiv P \wedge Q$: $\mathbf{F} \wedge Q$ ei riipu Q:stä, $\mathbf{U} \wedge \mathbf{U} \equiv \mathbf{U}$ ja $\mathbf{T} \wedge \mathbf{U} \equiv \mathbf{U}$
- sama voidaan todistaa jokaiselle kaavalle, jossa on vain $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$ ja/tai $\exists$

$\Rightarrow$ ei saada aikaan kaavaa, joka tuottaisi edellä kerrotun ongelman, joka esti antamasta määrittelemättömille vertailuille totuusarvoksi **F**
  - esim. sellaista kaavaa $\varphi(P)$, että $\varphi(\mathbf{F}) \equiv \varphi(\mathbf{U}) \equiv \mathbf{F}$ ja $\varphi(\mathbf{T}) \equiv \mathbf{T}$

P	$\varphi(P)$
F	**F**
U	**F**
T	**T**

Joitakin tuttuja lakeja menetetään, mutta ei kovin paljoa

$(1) \equiv P \vee \neg P \equiv P \rightarrow P \equiv P \leftrightarrow P$ $(2) \equiv P \wedge (\neg P \vee Q)$ $(3) \equiv P \vee \neg P \wedge Q$

P	$P \wedge \neg P$
F	**F**
U	**U**
T	**F**

P	(1)
F	**T**
U	**U**
T	**T**

(2)	**F**	**U**	**T**
F	**F**	**F**	**F**
U	**U**	**U**	**U**
T	**F**	**U**	**T**

(3)	**F**	**U**	**T**
F	**F**	**U**	**T**
U	**U**	**U**	**U**
T	**T**	**T**	**T**

- sijaan tulevien lakien ilmaisemiseksi tarvitaan seuraavaksi esiteltävä käsite

"On määritelty" -symboli

P	$\varphi(P)$
F	**T**
U	**F**
T	**T**

- edellä näimme, että käyttämillämme logiikan operaattoreilla ei voida kirjoittaa kaavaa $\varphi(P)$, joka tuottaa **F** jos P tuottaa **U** ja **T** muutoin
  - se ei ole P:n suhteen vakio eikä sille $\varphi(\mathbf{U})$ tuota **U**
- silti jokaiselle lausekkeelle ja kaavalle erikseen voidaan kirjoittaa sellainen kaava
  - se ei ole funktio, jolle tutkittavan lausekkeen tai kaavan arvo annetaan parametrina, vaan se täytyy kirjoittaa aina tapauskohtaisesti
  - merkitsemme sitä $\lfloor\varphi\rceil$
  - $\lfloor\varphi\rceil$ tuottaa aina **F** tai **T**, ei koskaan **U**
- muodostustapa aritmeettisille lausekkeille
  - jos x on lukuarvoinen muuttuja, niin $\lfloor x\rceil \equiv \mathbf{T}$
  - $\lfloor f(x)+f(y)\rceil \equiv \lfloor f(x)\rceil \wedge \lfloor f(y)\rceil$, ja samoin vähennys- ja kertolaskulle
  - $\left\lfloor \frac{f(x)}{f(y)} \right\rceil \equiv \lfloor f(x)\rceil \wedge \lfloor f(y)\rceil \wedge f(y) \neq 0$
  - $\left\lfloor \sqrt{f(x)} \right\rceil \equiv \lfloor f(x)\rceil \wedge f(x) \geq 0$
  - ...
  - nämä on helppo keksiä tarvittaessa, kun muistaa periaatteen: jokaisen osan sekä niitä yhdistävän operaation täytyy olla määritelty

- muodostustapa loogisille lausekkeille
  - $\lfloor \mathbf{T} \rceil \equiv \lfloor \mathbf{F} \rceil \equiv \mathbf{T}$ ja $\lfloor \mathbf{U} \rceil \equiv \mathbf{F}$
  - $\lfloor \neg\varphi \rceil \equiv \lfloor \varphi \rceil$
  - $\lfloor \varphi \wedge \psi \rceil \equiv \lfloor \varphi \rceil \wedge \neg\varphi \vee \lfloor \psi \rceil \wedge \neg\psi \vee \lfloor \varphi \rceil \wedge \lfloor \psi \rceil$
  - $\lfloor \varphi \vee \psi \rceil \equiv \lfloor \varphi \rceil \wedge \varphi \vee \lfloor \psi \rceil \wedge \psi \vee \lfloor \varphi \rceil \wedge \lfloor \psi \rceil$
  - $\lfloor \varphi \rightarrow \psi \rceil$ on helppo johtaa edellisistä implikaation eliminoinnilla
  - $\lfloor \varphi \leftrightarrow \psi \rceil \equiv \lfloor \varphi \rceil \wedge \lfloor \psi \rceil$
- nämäkin on helppo keksiä tarvittaessa
- usein voi käyttää helposti muistettavaa alalikiarvoa $\lfloor \varphi \rceil \wedge \lfloor \psi \rceil$

$\varphi \equiv \psi$ jos ja vain jos $\varphi \Leftrightarrow \psi$ ja $\lfloor \varphi \rceil \Leftrightarrow \lfloor \psi \rceil$

"φ on määrittelemätön" saadaan sanomalla $\varphi \equiv \mathbf{U}$, mutta ei saada sanomalla $\varphi \Leftrightarrow \mathbf{U}$

- $\varphi \Leftrightarrow \mathbf{U}$ pätee myös kun φ tuottaa **F**
- ei myöskään saada "φ tuottaa **F**" sanomalla $\varphi \Leftrightarrow \mathbf{F}$, mutta saadaan sanomalla $\varphi \equiv \mathbf{F}$
- sen sijaan jokainen seuraavista sanoo "φ tuottaa **T**"
  - $\varphi \Leftrightarrow \mathbf{T}$
  - $\varphi \equiv \mathbf{T}$
  - φ
- $\neg\varphi$ sanoo "φ tuottaa **F**"
- "φ on määrittelemätön" saadaan myös sanomalla $\neg\lfloor \varphi \rceil$

U:n vaikutus propositiologiikan lakeihin

- luvun 4.1 tummansiniset lait pätevät sekä $\Leftrightarrow$:a käytettäessä että siten, että $\Leftrightarrow$:n paikalla on $\equiv$
- muita lakeja, jotka pätevät sekä $\equiv$:a käytettäessä että siten, että $\equiv$:n paikalla on $\Leftrightarrow$

$$P \vee \neg P \vee \neg \lfloor P \rceil \equiv \mathbf{T} \qquad P \wedge \neg P \wedge \lfloor P \rceil \equiv \mathbf{F} \qquad P \to P \equiv P \leftrightarrow P \equiv P \vee \neg P$$

$$P \vee \neg (P \wedge \lfloor P \rceil) \wedge Q \equiv P \vee Q \qquad P \wedge ((\neg P \wedge \lfloor P \rceil) \vee Q) \equiv P \wedge Q$$

- vaaleansiniset lait pätevät $\Leftrightarrow$:a käytettäessä mutta eivät siten, että $\Leftrightarrow$:n paikalla on $\equiv$

$$P \vee \neg P \Leftrightarrow \lfloor P \rceil \qquad P \wedge \neg P \Leftrightarrow \mathbf{F} \qquad P \wedge (\neg P \vee Q) \Leftrightarrow P \wedge Q$$

Kaikki vertailut, joissa jokin osa on määrittelemätön, tuottavat **U**

- esim. $\frac{1}{0} > 0$ ei päde, mutta myöskään $\frac{1}{0} \leq 0$ ei päde
- $\frac{1}{0} > 0 \equiv \mathbf{U}$, joten $\neg(\frac{1}{0} > 0) \equiv \neg\mathbf{U} \equiv \mathbf{U}$ ja $\frac{1}{0} \leq 0 \equiv \mathbf{U}$, joten $\neg(\frac{1}{0} \leq 0) \equiv \mathbf{U}$

$\Rightarrow \neg(\frac{1}{0} > 0) \equiv \frac{1}{0} \leq 0$ ja $\neg(\frac{1}{0} \leq 0) \equiv \frac{1}{0} > 0$

- siksi monet tutut lait ovat voimassa määrittelemättömillekin lausekkeille, kuten
  - $f(x) \neq g(x) \equiv \neg(f(x) = g(x))$
  - $f(x) \leq g(x) \equiv \neg(f(x) > g(x))$
  - $f(x) \leq g(x) \equiv f(x) < g(x) \vee f(x) = g(x)$
  - $f(x) < g(x) \equiv f(x) \leq g(x) \wedge f(x) \neq g(x)$

Jos $f(x)$ tai $g(x)$ on määrittelemätön, niin $f(x) = g(x)$ ei päde

$\Rightarrow$ sivun 53 kongruenssiominaisuus ei anna lupaa korvata $\varphi(f(x))$:ä $\varphi(g(x))$:llä

- jos $\neg\lfloor f(x)\rceil$, niin edes $f(x) = f(x)$ ei päde
  - muutoin $x = -3$ olisi yhtälön $\sqrt{x-3} = \sqrt{2x}$ juuri, koska sillä tulee $\sqrt{-6} = \sqrt{-6}$
- toisaalta jos sekä $f(x)$ että $g(x)$ ovat määrittelemättömiä, niin
  - sekä $f(x) > 0$ että $g(x) > 0$ tuottaa **U**
  - samoin kaikille muillekin vertailuille, joissa $f(x)$ tai $g(x)$ on osapuolena

  $\Rightarrow \varphi(f(x)) \equiv \varphi(g(x))$

$\Rightarrow$ jos jokaisella x:n arvolla joko $f(x) = g(x)$ tai sekä $f(x)$ että $g(x)$ ovat määrittelemättömiä, niin $\varphi(f(x)) \equiv \varphi(g(x))$

Lisää havaintoja

- tietenkin $\lfloor f(x)\rceil \Rightarrow f(x) = f(x)$
- $f(x) \neq f(x)$ ei päde koskaan, vaan tuottaa aina **F** tai **U**
- $f(x) - f(x) = 0$, $0 \cdot f(x) = 0$ ja $\frac{f(x)}{f(x)} = 1$ eivät päde kun $f(x)$ on määrittelemätön
  - silloin "$=$" tuottaa **U**
- esim. päättely $0 = 0 \Rightarrow 0 = 0 \cdot \frac{1}{0} \Rightarrow 0 = 1$ on väärin,
  koska 0:n korvaaminen $0 \cdot \frac{1}{0}$:lla on väärin,
  koska ne eivät ole yhtäsuuret eivätkä molemmat ole määrittelemättömiä
- sivulla 151 kerrotaan, miten esim. $f(x) - f(x) = 0$ käsitellään kun $\neg\lfloor f(x)\rceil$

Ohjelmointikielten ”ja” ja ”tai”

- ohjelmointikielissä ei yleensä ole totuusarvoa **U**, mutta ohjelma voi kaatua
  - voimme valita **U**:n tarkoittamaan kaatumista
- C:n loogisen operaattorin oikea puoli lasketaan vain jos tarpeen
  ⇒ ne toimivat näin:

&&	F	U	T
F	F	F	F
U	U	U	U
T	F	U	T

||	F	U	T
F	F	U	T
U	U	U	U
T	T	T	T

  - P && $Q \equiv P \wedge (\neg P \vee Q)$ ja $P \,||\, Q \equiv P \vee \neg P \wedge Q$
  - kätevä esim. tilanteessa `for( i = 0; i < n && A[i] != x; ++i );`
- Pascalin loogisen operaattorin molemmat puolet lasketaan aina
  ⇒ ne toimivat näin:

and	F	U	T
F	F	U	F
U	U	U	U
T	F	U	T

or	F	U	T
F	F	U	T
U	U	U	U
T	T	U	T

⇒ ohjelmointikielten loogiset operaattorit eivät ole samat kuin logiikan

4.4 Päättelyoperaattorit

Olemme edellä käyttäneet $\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$ ja $\equiv$ ilman määritelmää

- niitä käytetään ilmaisemaan *päättelyitä*
- **päättelyimplikaatio** (tai vain **implikaatio**) $\varphi \Rightarrow \psi$ tarkoittaa, että kaikissa huomioon otettavissa tilanteissa joissa φ pätee, myös ψ pätee
  - propositiologiikassa tilanne = muuttujien arvoyhdistelmä
  - predikaattilogiikassa tilanteen käsite on monimutkaisempi, sivu 153
  - tilanne, jossa kaava pätee, on kaavan **malli** (**model**)
    (sanaa "malli" käytetään siis jokseenkin päinvastoin kuin esim. tekniikassa)
- $\psi \Leftarrow \varphi$ tarkoittaa samaa kuin edellinen
- **päättelyekvivalenssi** (tai vain **ekvivalenssi**) $\varphi \Leftrightarrow \psi$ tarkoittaa, että kaikissa huomioon otettavissa tilanteissa joissa φ pätee, myös ψ pätee ja toisinpäin
- **totuusarvojen samuus** $\varphi \equiv \psi$ tarkoittaa, että kaikissa huomioon otettavissa tilanteissa φ ja ψ tuottavat saman totuusarvon
  - jos kaikki on määriteltyä, niin $\varphi \equiv \psi$ tarkoittaa samaa kuin $\varphi \Leftrightarrow \psi$
- usein vain osa käsitteellisesti mahdollisista tilanteista otetaan huomioon
  - esim. kun päättely on jaettu tapauksiin $x < 0$ ja $x \geq 0$, ja käsitellään $x < 0$
  - esim. kun ristiriidan johtamiseksi oletetaan $x < 0$
  - esim. jos P tarkoittaa $x > 1$ ja Q tarkoittaa $x > 0$, niin $P \wedge \neg Q$ ei oteta huomioon
- päättelyoperaattorit eivät palauta totuusarvoa, vaan päättelyaskel on tai ei ole pätevä

Päättelyoperaattoreiden pätevyystaulut

$\Rightarrow$	**F**	**U**	**T**
F	$\times$	$\times$	$\times$
U	$\times$	$\times$	$\times$
T	$-$	$-$	$\times$

$\Leftarrow$	**F**	**U**	**T**
F	$\times$	$\times$	$-$
U	$\times$	$\times$	$-$
T	$\times$	$\times$	$\times$

$\Leftrightarrow$	**F**	**U**	**T**
F	$\times$	$\times$	$-$
U	$\times$	$\times$	$-$
T	$-$	$-$	$\times$

$\equiv$	**F**	**U**	**T**
F	$\times$	$-$	$-$
U	$-$	$\times$	$-$
T	$-$	$-$	$\times$

Päättelyoperaattoreiden kielioppi

$$\begin{aligned}\text{Päättely} \;::=\; &\text{Kaava} \mid \text{Päättely} \Rightarrow \text{Kaava} \mid \text{Päättely} \Leftarrow \text{Kaava} \mid \\ &\text{Päättely} \Leftrightarrow \text{Kaava} \mid \text{Päättely} \equiv \text{Kaava}\end{aligned}$$

Päättelyoperaattorit käyttäytyvät syntaktisesti kuten vertailut

- esim. $\varphi \Leftrightarrow \psi \Rightarrow \chi$ tarkoittaa, että $\varphi \Leftrightarrow \psi$ on pätevä askel ja $\psi \Rightarrow \chi$ on pätevä askel
  - vrt. $0 \leq x < 1$ tarkoittaa että $0 \leq x$ ja $x < 1$
- esim. $(x^2 > 0 \to x^2 < 0) \to x = 0$ tuottaa aina **T** ...
  - jos $x = 0$, se tuottaa $(\mathbf{F} \to \mathbf{F}) \to \mathbf{T}$ joka tuottaa $\mathbf{T} \to \mathbf{T}$ joka tuottaa **T**
  - jos $x \neq 0$, se tuottaa $(\mathbf{T} \to \mathbf{F}) \to \mathbf{F}$ joka tuottaa $\mathbf{F} \to \mathbf{F}$ joka tuottaa **T**
- ... mutta $(x^2 > 0 \;\Rightarrow\; x^2 < 0) \;\Rightarrow\; x = 0$ on syntaksivirhe ...
  - kielioppi ei salli sulkuja päättelyn ympärille
  - vrt. $(0 \leq x) < 1$
- ... ja $x^2 > 0 \;\Rightarrow\; x^2 < 0 \;\Rightarrow\; x = 0$ on virheellinen päättely
  - $x^2 > 0 \;\Rightarrow\; x^2 < 0$ ei ole pätevä: kun $x = 1$ niin $x^2 > 0$ mutta ei $x^2 < 0$

- esim. C++ sallii sekä `1<1<1` että `(1<1)<1`, ja sen mielestä molemmat ovat `true`!
  - `1<1` tuottaa `false` eli `0`, joten `1<1<1` tuottaa saman kuin `0<1` eli `1` eli `true`

  $\Rightarrow$ matematiikan $<$ ja ohjelmoinnin `<` ketjuttuvat olennaisesti eri tavalla

Vertailu $\rightarrow$ ja $\leftrightarrow$ vastaan $\Rightarrow$ ja $\Leftrightarrow$

- tuloksen tyyppi on eri
  - $\rightarrow$ ja $\leftrightarrow$: totuusarvo **F**, **U** tai **T**
  - $\Rightarrow$ ja $\Leftrightarrow$: päättelyaskel on tai ei ole pätevä (ei ole kolmatta vaihtoehtoa)
- syntaktiset säännöt ovat erit
  - $(\varphi \rightarrow \psi) \rightarrow \chi$ tuottaa saman lausekepuun kuin $\varphi \rightarrow \psi \rightarrow \chi$
  - $(\varphi \Rightarrow \psi) \Rightarrow \chi$ on syntaksivirhe
- suhtautuminen asiayhteyteen on erilainen
  - $x^2 = 4 \rightarrow x = -2$ ei ole tautologia, koska se tuottaa **F** kun $x = 2$
  - $x^2 = 4 \Rightarrow x = -2$ on pätevä päättelyaskel kun käsitellään tapausta $x < 0$
- käyttäytyminen määrittelemättömissä tilanteissa on erilainen

$\rightarrow$	**F**	**U**	**T**
F	**T**	**T**	**T**
U	**U**	**U**	**T**
T	**F**	**U**	**T**

$\Rightarrow$	**F**	**U**	**T**
F	×	×	×
U	×	×	×
T	–	–	×

$\leftrightarrow$	**F**	**U**	**T**
F	**T**	**U**	**F**
U	**U**	**U**	**U**
T	**F**	**U**	**T**

$\Leftrightarrow$	**F**	**U**	**T**
F	×	×	–
U	×	×	–
T	–	–	×

$\equiv$	**F**	**U**	**T**
F	×	–	–
U	–	×	–
T	–	–	×

- monessa kirjassa esitellään vain $\Rightarrow$ ja $\Leftrightarrow$ tai vain $\rightarrow$ ja $\leftrightarrow$, ja saatetaan antaa esitellyille edellisten ja jälkimmäisten ominaisuuksia ristiin

Päättelyoperaattoreiden ominaisuuksia

- $\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$ ja $\equiv$ ovat **refleksiivisiä**
  - toisin sanoen, $\varphi \Rightarrow \varphi$, $\varphi \Leftrightarrow \varphi$ jne. pätevät aina (ne pätevät, vaikka φ ei pätisi)
- $\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$ ja $\equiv$ ovat **transitiivisia**
  - aina kun $\varphi \Rightarrow \psi$ ja $\psi \Rightarrow \chi$ pätevät myös $\varphi \Rightarrow \chi$ pätee
  - aina kun $\varphi \Leftrightarrow \psi$ ja $\psi \Leftrightarrow \chi$ pätevät myös $\varphi \Leftrightarrow \chi$ pätee
  - aina kun $\varphi \equiv \psi$ ja $\psi \equiv \chi$ pätevät myös $\varphi \equiv \chi$ pätee
- $\Leftrightarrow$ ja $\equiv$ ovat **symmetrisiä**, mutta $\Rightarrow$ ja $\Leftarrow$ eivät ole
  - aina kun $\varphi \Leftrightarrow \psi$ pätee myös $\psi \Leftrightarrow \varphi$ pätee
  - **F** $\Rightarrow$ **T** pätee mutta **T** $\Rightarrow$ **F** ei päde
- $\varphi \Leftrightarrow \psi$ pätee jos ja vain jos sekä $\varphi \Rightarrow \psi$ että $\psi \Rightarrow \varphi$ pätevät
- $\varphi \Rightarrow \psi$ pätee jos ja vain jos $\neg\psi \vee \neg\lfloor\psi\rceil \Rightarrow \neg\varphi \vee \neg\lfloor\varphi\rceil$ pätee
- jos kaikki on määriteltyä, niin $\varphi \Rightarrow \psi$ pätee jos ja vain jos $\neg\psi \Rightarrow \neg\varphi$ pätee
  - esim. vertaa sataa $\Rightarrow$ kastuu ja $\neg$ kastuu $\Rightarrow \neg$ sataa
- jos $\varphi \equiv \psi$, niin kaavan osan φ saa korvata ψ:llä
  - siis jos $\varphi \equiv \psi$, niin $\xi(\varphi) \equiv \xi(\psi)$
  - esim. $x^2 > 0 \equiv x \neq 0$, joten $x^2 \leq 0 \equiv \neg(x^2 > 0) \equiv \neg(x \neq 0) \equiv x = 0$
- sama ei toimi $\Leftrightarrow$:lle koska **U** $\Leftrightarrow$ **F**
  - esim. $x > 0 \Leftrightarrow \frac{1}{x} > 0$ pätee, mutta $x \leq 0 \Leftrightarrow \neg(x > 0) \Leftrightarrow \neg(\frac{1}{x} > 0) \Leftrightarrow \frac{1}{x} \leq 0$ ei

Mahdottomasta saa päätellä mitä tahansa!

- ehkä yllättäen, $x^2 < 0 \Rightarrow x = 0$ on pätevä päättelyaskel!
- "kaikissa huomioon otettavissa tilanteissa joissa φ pätee, myös ψ pätee"
- $x^2 < 0$ ei päde missään tilanteessa, joten $(x = 0)$:n ei tarvitse päteä missään tilanteessa
- tarkastelkaamme melkein kaikkien pätevyksi hyväksymää päättelyä $x > 1 \Rightarrow x > 0$
  - jos $x \leq 0$, se tuottaa **F** $\Rightarrow$ **F**
  - jos $0 < x \leq 1$, se tuottaa **F** $\Rightarrow$ **T**
  - jos $1 < x$, se tuottaa **T** $\Rightarrow$ **T**
  - se ei voi tuottaa **T** $\Rightarrow$ **F**
- tarkastelkaamme melkein kaikkien vääränä pitämää päättelyä $x > 0 \Rightarrow x > 1$
  - jos $x \leq 0$, se tuottaa **F** $\Rightarrow$ **F**
  - jos $0 < x \leq 1$, se tuottaa **T** $\Rightarrow$ **F**
  - jos $1 < x$, se tuottaa **T** $\Rightarrow$ **T**
  - (se ei voi tuottaa **F** $\Rightarrow$ **T**)

$\Rightarrow$ on järkevää hyväksyä päättelyaskel jos ja vain jos se ei missään huomioon otettavassa tilanteessa tuota **T** $\Rightarrow$ **F**

- sitäpaitsi tätä ilmiötä olisi vaikea estää vaikka yritettäisiin, koska useimpien hyväksymillä säännöillä todella voi johtaa väärästä lähtökohdasta monenlaista
  - oletetaan $1 = 2$
  - kertomalla molemmat puolet mielivaltaisella luvulla a saadaan $a = 2a$
  - vähentämällä molemmilta puolilta a saadaan $0 = a$
  - sama onnistuu toisellekin mielivaltaiselle luvulle b, joten $0 = b$
  - $a = 0 = b$, joten $a = b$

  $\Rightarrow$ kaikki luvut ovat yhtäsuuria!

Sijoittaminen kaavan sisään

- kaavan sisässä olevan luvun tuottavan lausekkeen minkä tahansa esiintymän saa korvata yhtäsuuren luvun tuottavalla lausekkeella (huom. luku 5)
  - siis jos $x = y$ niin $\varphi(x) \equiv \varphi(y)$
  - esim. jos $x = 2$ niin $y > x(x-1) \equiv y > 2(x-1) \equiv y > 2(2-1) \equiv y > 2$
  - esim. jos $2x + 1 = y$ niin
    $5x = 2y + 1 \equiv 5x = 2(2x+1) + 1 \equiv 5x = 4x + 3 \equiv x = 3$
  - esim. jos $x^2 = 5x$ niin $x^2 > 0 \equiv 5x > 0$
- alikaavan minkä tahansa esiintymän saa korvata saman totuusarvon tuottavalla alikaavalla
  - siis jos $\varphi \equiv \psi$ niin $\xi(\varphi) \equiv \xi(\psi)$
  - esim. koska $x < 2 \vee x > 2 \equiv x \neq 2$, pätee
    $(x < 2 \vee x > 2) \wedge y > x \equiv x \neq 2 \wedge y > x$

- jos $\varphi \Rightarrow \psi$ ja $\xi(P)$ on kaava, jossa korvattava P:n esiintymä on parillisen määrän negaatioita vaikutuspiirissä eikä ole $\leftrightarrow$:n vaikutuspiirissä, niin $\xi(\varphi) \Rightarrow \xi(\psi)$
  - negaatioksi lasketaan myös $\rightarrow$ vasemman argumenttinsa suhteen
  - esim. koska $x > 1 \Rightarrow x > 0$, pätee $x \leq 2 \wedge x > 1 \Rightarrow x \leq 2 \wedge x > 0$
  - tälle on tärkeää, että kaavassa on vain $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$ ja/tai $\exists$
- perustelu
  - $\xi(P)$ on kasvava P:n suhteen
  - jos $\xi(P)$ ei riipu P:stä, niin triviaalisti $\xi(\varphi) \equiv \xi(\psi)$
  - muussa tapauksessa $\xi(\mathbf{U}) \equiv \mathbf{U}$, joten jos $\xi(\varphi) \equiv \mathbf{T}$, niin kasvavuuden vuoksi $\varphi \equiv \mathbf{T}$, joten $\psi \equiv \mathbf{T}$ ja $\xi(\psi) \equiv \xi(\mathbf{T}) \equiv \xi(\varphi) \equiv \mathbf{T}$
- sama pätee $\Leftrightarrow$:lle
- jos $\varphi \Rightarrow \psi$ ja $\xi(P)$ on kaava, jossa korvattava P:n esiintymä on parittoman määrän negaatioita vaikutuspiirissä eikä ole $\leftrightarrow$:n vaikutuspiirissä, ja φ ja ψ ovat määrittelemättömät täsmälleen samoilla muuttujien arvoyhdistelmillä, niin $\xi(\psi) \Rightarrow \xi(\varphi)$
  - negaatioksi lasketaan myös $\rightarrow$ vasemman argumenttinsa suhteen
  - esim. koska $x > 1 \Rightarrow x > 0$, pätee $x > 0 \rightarrow f(x) > 0 \Rightarrow x > 1 \rightarrow f(x) > 0$
  - $\xi(P)$ on vähenevä P:n suhteen, mikä antaa väitteen, koska $\varphi \Rightarrow \psi$ voi toteutua vain seuraavasti: $\mathbf{F} \Rightarrow \mathbf{F}$, $\mathbf{F} \Rightarrow \mathbf{T}$, $\mathbf{U} \Rightarrow \mathbf{U}$ ja $\mathbf{T} \Rightarrow \mathbf{T}$

Määrittelemättömyyden poisto

- ratkaisemme esimerkin vuoksi yhtälön väärin: $\frac{2x-6}{|x|-3} = x \;\Leftrightarrow\; 2x-6 = x(|x|-3)$

  $\Leftrightarrow\; 5x-6 = x|x| \;\Leftrightarrow\; x<0 \wedge x^2+5x-6=0 \vee x \geq 0 \wedge x^2-5x+6=0$

  $\Leftrightarrow\; x<0 \wedge (x=-6 \vee x=1) \vee x \geq 0 \wedge (x=2 \vee x=3) \;\Leftrightarrow\; x=-6 \vee x=2 \vee x=3$

  - sijoittamalla näkee, että -6 ja 2 ovat yhtälön juuria, mutta 3 ei ole
  - kun $x=3$, niin $\frac{2x-6}{|x|-3} = x \;\equiv\; \frac{0}{0} = 3 \;\equiv$ **U**,
    mutta $2x-6 = x(|x|-3) \;\equiv\; 0=0 \;\equiv$ **T**
- koska **U** $\Leftrightarrow$ **F**, tämä on oikein: $\frac{2x-6}{|x|-3} = x \;\Leftrightarrow\; |x|-3 \neq 0 \wedge \frac{2x-6}{|x|-3} = x$

  $\Leftrightarrow\; |x| \neq 3 \wedge 2x-6 = x(|x|-3) \;\Leftrightarrow\; \ldots \;\Leftrightarrow\; x=-6 \vee x=2$
- mitkä seuraavista ovat oikein?
  - $x=3 \vee \frac{2x-6}{|x|-3} = x \;\Leftrightarrow\; |x| \neq 3 \wedge (x=3 \vee 2x-6 = x(|x|-3))$
  - $x=3 \vee \frac{2x-6}{|x|-3} = x \;\Leftrightarrow\; x=3 \vee (|x| \neq 3 \wedge 2x-6 = x(|x|-3))$

  $\Rightarrow$ määrittelemättömyyden poistava ehto pitää lisätä paikallisesti eikä koko kaavaan
  - sen voi lisätä koko kaavaan säännöllä $\lfloor \varphi \vee \psi \rceil \;\equiv\; \lfloor \varphi \rceil \wedge \varphi \vee \lfloor \psi \rceil \wedge \psi \vee \lfloor \varphi \rceil \wedge \lfloor \psi \rceil$,
    mutta se on kömpelömpää

- väärä esimerkki: $\frac{2x-6}{|x|-3} \neq x \;\Leftrightarrow\; \neg(\frac{2x-6}{|x|-3} = x) \;\Leftrightarrow\; \neg(|x| \neq 3 \wedge 2x-6 = x(|x|-3))$
  $\Leftrightarrow\; \neg(|x| \neq 3 \wedge (x = -6 \vee x = 2 \vee x = 3)) \;\Leftrightarrow\; x \neq -6 \wedge x \neq -2$
  - taas 3 putkahti virheellisesti juureksi

  $\Rightarrow$ määrittelemättömyyden poistava ehto pitää lisätä eri tavalla $\neg$:n alaisuudessa
- jos $\xi(P)$ on kaava, jossa korvattava P:n esiintymä on parillisen määrän negaatioita vaikutuspiirissä eikä ole $\leftrightarrow$:n vaikutuspiirissä, niin $\xi(\varphi) \;\Leftrightarrow\; \xi(\lfloor\varphi\rceil \wedge \varphi)$
  - tälle on tärkeää, että kaavassa on vain $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$ ja/tai $\exists$
- jos $\xi(P)$ on kaava, jossa korvattava P:n esiintymä on parittoman määrän negaatioita vaikutuspiirissä eikä ole $\leftrightarrow$:n vaikutuspiirissä, niin $\xi(\varphi) \;\Leftrightarrow\; \xi(\neg\lfloor\varphi\rceil \vee \varphi)$
  - tälle on tärkeää, että kaavassa on vain $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$ ja/tai $\exists$
- näillä säännöillä määrittelemättömyys voidaan poistaa sitä mukaa kuin se tulee ratkaisuprosessissa ajankohtaiseksi

5 Predikaattilogiikka

5.1 Syntaksi, semantiikka ja lakeja

Predikaattilogiikan kaava on muuten samanlainen kuin propositiologiikan, mutta

- propositioiden tilalla on monimutkaisempi käsite *predikaatti*
- kaavoissa voi käyttää *kvanttoreita* $\forall$ ja $\exists$

Predikaatti

- predikaattilogiikan kaava väittää jotain jostakin kohdemaailmasta
  - esim. reaaliluvut: $\forall x : x \geq 0 \rightarrow \exists y : y^2 = x$
  - esim. kokonaisluvut: $\forall n : \forall m : m \neq 0 \rightarrow \exists q : \exists r : n = qm + r \wedge 0 \leq r < |m|$
  - esim. merkkijonot: $\forall \alpha : \forall \beta : \text{takaperin}(\alpha\beta) = \text{takaperin}(\beta)\text{takaperin}(\alpha)$
  - esim. taulukot: $\forall i : 1 \leq i < n \rightarrow A[i] \leq A[i+1]$
- **termi** on kohdemaailman arvon tuottava lauseke
  - esim. kokonaisluvut: n, 0, qm, $|m|$ ja $(k+1)^2$
  - esim. merkkijonot: $\text{takaperin}(\beta)\text{takaperin}(\alpha)$
- k-paikkainen **predikaatti** ottaa k termiä ja tuottaa totuusarvon
  - esim. $=$, $>$ ja $\geq$ ovat 2-paikkaisia predikaatteja
  - propositiot ovat 0-paikkaisia predikaatteja
  - esim. kokonaisluvuilla $\text{parillinen}(n)$

- emme määrittele termeille ja predikaateille kielioppia, koska jokaisella kohdemaailmalla on omansa
- luettavuuden vuoksi käytämme sulkuja negatoiduissa vertailuissa, kuten $\neg(x > 0)$

Moni propositiologiikkaluvun esimerkki oli todellisuudessa predikaattilogiikan kaava

- esim. $x \geq 0 \wedge x \neq 0 \Leftrightarrow x > 0$ on havainnollisempi kuin $P \wedge Q \Leftrightarrow R$

Kvanttorit

- usein riippuu x:n arvosta, päteekö $\varphi(x)$
  - esim. $x^2 + x = 6$
  - ei kuitenkaan aina, esim. $x^2 \geq 0$, $x < 0 \wedge x > 0$ tai $1 = 2$
- $\forall x : \varphi(x)$ pätee jos ja vain jos $\varphi(x)$ pätee jokaisella x:n arvolla
  - universaalikvanttori
  - esim. $\forall x : x^2 \geq 0$ ja $\forall a : \forall b : a + b = b + a$ pätevät reaaliluvuilla
  - esim. $\forall x : x^2 > 0$ ei päde reaaliluvuilla
- $\exists x : \varphi(x)$ pätee jos ja vain jos $\varphi(x)$ pätee ainakin yhdellä x:n arvolla
  - eksistenssikvanttori
  - esim. $\exists x : x^2 > 0$ pätee reaaliluvuilla
  - esim. $\exists x : x^2 = -1$ ei päde reaaliluvuilla
- kirjallisuudessa kvanttorien syntaksi vaihtelee (myös välimerkit)

Kielioppi tällä kurssilla

$$
\begin{array}{rcl}
\text{Kaava} & ::= & \text{Kvanttorikaava} \\
\text{Kvanttorikaava} & ::= & \text{Ekvivalenssikaava} \\
 & & \mid \forall\ \text{Muuttujaosuus} : \text{Kvanttorikaava} \\
 & & \mid \exists\ \text{Muuttujaosuus} : \text{Kvanttorikaava} \\
\text{Muuttujaosuus} & ::= & \text{Muuttuja} \\
 & & \mid \text{Muuttuja} \in \text{Joukko} \\
 & & \mid \text{Muuttuja}\ ;\ \text{Rajain}
\end{array}
$$

- tapaukset Muuttuja $\in$ Joukko ja Muuttuja ; Rajain käsitellään s. 164

Jos kohdemaailmassa on vain äärellinen määrä eri arvoja $v_1, \ldots, v_n$, niin

- $\forall x : \varphi(x)$ tarkoittaa samaa kuin $\varphi(v_1) \wedge \cdots \wedge \varphi(v_n)$
- $\exists x : \varphi(x)$ tarkoittaa samaa kuin $\varphi(v_1) \vee \cdots \vee \varphi(v_n)$
- tämän avulla voi palauttaa mieleen monta kohta esitettävää lakia

Milloin kvanttorikaava tuottaa **U**?

- $\forall x : \varphi(x) \equiv$ **U** jos ja vain jos
  - ainakin yhdellä x:n arvolla $\varphi(x) \equiv$ **U**
  - muilla x:n arvoilla $\varphi(x) \equiv$ **T**
- $\forall x : x \cdot \frac{1}{x} = 1 \equiv$ **U**
  - kun $x = 0$, on $x \cdot \frac{1}{x} = 1$ määrittelemätön, muulloin se tuottaa **T**

- $\exists x : \varphi(x) \equiv$ **U** jos ja vain jos
  - ainakin yhdellä x:n arvolla $\varphi(x) \equiv$ **U**
  - muilla x:n arvoilla $\varphi(x) \equiv$ **F**
- $\exists x : \frac{1}{(x-3)^2} \leq 0 \equiv$ **U**
  - kun $x = 3$, on $\frac{1}{(x-3)^2} \leq 0$ määrittelemätön, muulloin se tuottaa **F**

$\forall x : \varphi(x)$ ja $\exists x : \varphi(x)$ ovat kasvavia $\varphi(x)$:n suhteen

- esim. jos $\varphi(x) \Rightarrow \psi(x)$ pätee jokaisella x, niin $\forall x : \varphi(x) \Rightarrow \forall x : \psi(x)$

$\Rightarrow \forall$ ja $\exists$ ei katsota negaatioiksi, kun tutkitaan, onko jokin parillisen vai parittoman negaatioiden määrän vaikutuspiirissä

Muuttujan vapaa ja sidottu esiintymä

- jokainen seuraavista väittää eri asiaa:
  - $\forall x : x > 5$ ei päde, koska $x > 5$ ei toteudu esim. kun $x = 2$
  - $x > 5$ pätee tai ei, riippuen x:n arvosta
  - $\exists x : x > 5$ pätee, koska $x > 5$ toteutuu esim. kun $x = 8$
- muuttujan x esiintymä on **sidottu**, jos ja vain jos se on jonkin $\forall x :$ tai $\exists x :$ vaikutuspiirissä
  - myös $\forall x$ ja $\exists x$ itse otetaan huomioon
- muut muuttujan x esiintymät ovat **vapaita**
- esim. $x = 0 \wedge (\exists x : x^2 = 2x) \vee x = 1$ sisältää kaksi vapaata ja kolme sidottua x:n esiintymää

- ohjelmoijalle on luontevaa ajatella, että vapaana esiintyvä muuttuja on eri muuttuja kuin sidottuna esiintyvä, vaikka sillä on sama nimi
  - myös eri kvantifioinneilla luodut muuttujat kannattaa ajatella eri muuttujiksi
  - esim. $(\exists x : (\forall x : x^2 \neq 2) \vee x^2 = 2) \wedge x \neq 1 \wedge (\forall x : x^2 > 0 \vee x = 0) \vee x = 1$ sisältää neljä eri muuttujaa, joiden nimi on x
  - mitä "Kauppakatu" tarkoittaa riippuu siitä, ollaanko Jyväskylässä vai Kuopiossa

Avoin kaava sisältää ja **suljettu** kaava ei sisällä ainakin yhden vapaan esiintymän

- esim.
  - $\forall x : x > 0$ on suljettu
  - $x > 0$ on avoin
  - $\exists x : x > 0$ on suljettu
  - $(\exists x : (\forall x : x^2 \neq 2) \vee x^2 = 2) \wedge (\forall x : x^2 \geq 0) \vee x = 1$ on avoin
- suljetulla kaavalla on yksikäsitteinen totuusarvo **F**, **U** tai **T**
  - tämä sillä oletuksella, että termien ja predikaattien merkitys on kiinnitetty
  - esim. reaaliluvuilla symbolien 0, -123, $+$ ja $\leq$ merkitys on kiinnitetty
- logiikkaa käytetään usein myös siten, että näin ei oleteta
  - esim. mikä on se kokonaislukujen predikaatti $n \mid m$, jolle pätee $n \mid m \Leftrightarrow \exists k : m = kn$?
  - esim. mikä on se positiivisten kokonaislukujen funktio $f(n, m)$, jolle pätee $f(n, m) \mid n \wedge f(n, m) \mid m \wedge \neg(\exists k : k > f(n, m) \wedge k \mid n \wedge k \mid m)$?

- avoimen kaavan totuusarvo voi riippua vapaana esiintyvien muuttujien arvoista
  - esim. $\sqrt{x} > 0$ tuottaa **U** kun $x < 0$, **F** kun $x = 0$ ja **T** kun $x > 0$
  - ei kuitenkaan välttämättä, esim. $x^2 > -1$
- varmista, että ymmärrät kaavojen $\varphi(x)$, $\forall x : \varphi(x)$ ja $\exists x : \varphi(x)$ eron

Termin sijoittaminen muuttujan vapaiden esiintymien paikalle

- jos x on muuttuja, t on termi ja $\varphi(x)$ on kaava, niin $\varphi(t)$ tarkoittaa kaavaa, joka saadaan korvaamalla x:n jokainen vapaa esiintymä $\varphi(x)$:ssä t:llä
- korvauksen tulee tapahtua lausekepuun, ei tekstin tasolla
  $\Rightarrow$ voi olla tarpeen lisätä sulkuja
  - esim. x:n korvaaminen $n+1$:llä $x^2 \geq 0$:ssa: ei $n + 1^2 \geq 0$ vaan $(n+1)^2 \geq 0$
- termissä ei saa olla muuttujia, jotka ovat sidottuja yhdessäkin korvauskohdassa
  - esim. jokaisella $x \in \mathbb{R}$ pätee $\exists y : y > x$
  - x:n korvaaminen siinä $y + 1$:llä tuottaisi $\exists y : y > y + 1$, joka ei päde
  - tämä ongelma ratkeaa kohta "sidotun muuttujan vaihdolla"
- esimerkkejä

$\varphi(x)$	t	$\varphi(t)$
$x(y+z) = xy + xz$	$a - 3$	$(a-3)(y+z) = (a-3)y + (a-3)z$
$x \geq 0 \to \exists y : y^2 = x$	$z^2 + 1$	$z^2 + 1 \geq 0 \to \exists y : y^2 = z^2 + 1$
$x \geq 0 \to \exists y : y^2 = x$	$y^2 + 1$	$y^2 + 1 \geq 0 \to \exists y : y^2 = y^2 + 1$

Predikaattilogiikan lakeja

- de Morganin lait

$$\neg\forall x : \varphi(x) \;\Leftrightarrow\; \exists x : \neg\varphi(x)$$
$$\neg\exists x : \varphi(x) \;\Leftrightarrow\; \forall x : \neg\varphi(x)$$

- samanlaisten kvanttorien vaihto

$$\forall x : \forall y : \varphi(x,y) \;\Leftrightarrow\; \forall y : \forall x : \varphi(x,y) \qquad \exists x : \exists y : \varphi(x,y) \;\Leftrightarrow\; \exists y : \exists x : \varphi(x,y)$$

- erilaisten kvanttorien vaihto toimii vain toiseen suuntaan

$$\exists x : \forall y : \varphi(x,y) \;\Rightarrow\; \forall y : \exists x : \varphi(x,y)$$

  - toisinpäin ei päde, esim. $\forall y : \exists x : x \neq y$ mutta ei $\exists x : \forall y : x \neq y$

- kvantifioinnin jakaminen kahdeksi, huomaa jälkimmäisten suunnat!

$$\forall x : \varphi(x) \wedge \psi(x) \;\Leftrightarrow\; (\forall x : \varphi(x)) \wedge \forall x : \psi(x)$$
$$\exists x : \varphi(x) \vee \psi(x) \;\Leftrightarrow\; (\exists x : \varphi(x)) \vee \exists x : \psi(x)$$
$$\forall x : \varphi(x) \vee \psi(x) \;\Leftarrow\; (\forall x : \varphi(x)) \vee \forall x : \psi(x)$$
$$\exists x : \varphi(x) \wedge \psi(x) \;\Rightarrow\; (\exists x : \varphi(x)) \wedge \exists x : \psi(x)$$

- jos x ei esiinny vapaana φ:ssä, niin

$$\varphi \wedge \forall x : \psi(x) \;\Leftrightarrow\; \forall x : \varphi \wedge \psi(x) \qquad \varphi \wedge \exists x : \psi(x) \;\Leftrightarrow\; \exists x : \varphi \wedge \psi(x)$$
$$\varphi \vee \forall x : \psi(x) \;\Leftrightarrow\; \forall x : \varphi \vee \psi(x) \qquad \varphi \vee \exists x : \psi(x) \;\Leftrightarrow\; \exists x : \varphi \vee \psi(x)$$

  - ilman ehtoa voitaisiin erehtyä esim. $x < 1 \wedge \forall x : x^2 \geq 0 \overset{?}{\Leftrightarrow} \forall x : x < 1 \wedge x^2 \geq 0$
  - $(x < 1)$:n x tarkoittaa eri muuttujia virheellisen $\Leftrightarrow$:n eri puolilla

$\Rightarrow$ moneen lakiin liittyy ehtoja, jotka estävät eri muuttujia muuttumasta samaksi
  - kirjallisuudessa ehdot vaikuttavat sekavilta
  - valitettavasti ne vaikuttavat sekavilta myös näissä luennoissa ☹
  - niiden kanssa pärjää, kun ajattelee, että eri muuttujilla voi olla sama nimi, ja pitää huolta, että nimen esiintymä viittaa aina oikeaan muuttujaan
  - usein auttaa *sidotun muuttujan vaihto*, mutta sekin on toki tehtävä oikein!

- sidotun muuttujan vaihto: jos y ei esiinny $\varphi(x)$:ssä, niin

$$\forall x : \varphi(x) \;\Leftrightarrow\; \forall y : \varphi(y) \qquad \exists x : \varphi(x) \;\Leftrightarrow\; \exists y : \varphi(y)$$

  - esim. $\exists x : x > y$ mutta ei $\exists y : y > y$
  - esim. $\forall x : \exists y : x = y + 1$ mutta ei $\forall y : \exists y : y = y + 1$
  - yleistys: y saa esiintyä sidottuna $\varphi(x)$:ssä, kunhan x ei esiinny vapaana siellä missä y on sidottu

- jos $t_1 = t_2$ ja mikään t_1:ssä tai t_2:ssa esiintyvä muuttuja ei ole sidottu x:n kohdalla $\varphi(x)$:ssä, niin $\varphi(t_1) \Leftrightarrow \varphi(t_2)$
  - vaikka olisi päätelty $x = 1$, niin ei saa päätellä $\exists x : x^2 = 9 \Leftrightarrow \exists x : 1^2 = 9$

Avoimet kaavat päättelyssä

- $\varphi(x) \Rightarrow \psi(x)$ on pätevä jos ja vain jos niissä huomioon otettavissa tilanteissa, joissa $\varphi(x)$ pätee, myös $\psi(x)$ pätee
  - usein vain osa käsitteellisesti mahdollisista tilanteista otetaan huomioon, ks. s. 144
- esim. $x > 1 \Rightarrow x > 0$
- esim. käsiteltäessä tapausta $c \geq 2$: $x > 3 \Rightarrow cx > 6$

Kvanttorien luonti- ja eliminointilait

- universaalikvanttorin eliminointi: jos mikään termissä t esiintyvä muuttuja ei ole sidottu x:n kohdalla $\varphi(x)$:ssä, niin

$$\lfloor \varphi(t) \rceil \wedge \forall x : \varphi(x) \;\Rightarrow\; \varphi(t)$$

  - jos lisäksi t on määritelty kaikilla sisältämiensä muuttujien arvoyhdistelmillä, niin

$$\forall x : \varphi(x) \;\Rightarrow\; \varphi(t)$$

- eksistenssikvanttorin luonti: jos mikään termissä t esiintyvä muuttuja ei ole sidottu x:n kohdalla $\varphi(x)$:ssä, niin

$$\varphi(t) \;\Rightarrow\; \exists x : \varphi(x)$$

- universaalikvanttorin luonti: jos x:stä ei ole oletettu mitään, niin

$$\varphi(x) \;\Rightarrow\; \forall x : \varphi(x)$$

  - x ei saa esiintyä vapaana sen päättelyn lähtökohdissa, jolla $\varphi(x)$ johdettiin

- eksistenssikvanttorin eliminointi: jos c ei esiinny aiemmin, $\varphi(x)$:ssä eikä ψ:ssä, niin

$$\text{jos } \varphi(c) \;\Rightarrow\; \psi \text{ niin } \exists x : \varphi(x) \;\Rightarrow\; \psi$$

  - usein ensin päätellään $\exists x : \varphi(x)$ ja sitten johdetaan $\varphi(c)$:stä ψ
  - c edustaa arvoa, jonka $\exists x : \varphi(x)$ lupaa olevan olemassa
  - jos $\varphi(x)$:n muut vapaat muuttujat kuin x ovat $x_1, \ldots, x_n$, niin c voi riippua niistä; tätä voi korostaa merkinnällä $c(x_1, \ldots, x_n)$ tai $c_{x_1,\ldots,x_n}$

Esimerkki pätevästä päättelystä (olettaen, että c ja z eivät esiinny $\varphi(x, y)$:ssä):

- $\exists x : \forall y : \varphi(x, y)$ lähtökohta
  $\forall y : \varphi(c, y)$ $\exists$:n eliminointi alkaa pätevästi
  $\Rightarrow\ \varphi(c, z)$ pätevä $\forall$:n eliminointi, eikä z:sta oleteta mitään
  $\Rightarrow\ \exists x : \varphi(x, z)$ pätevä $\exists$:n luonti, $t = c$
  $\Rightarrow\ \forall z : \exists x : \varphi(x, z)$ pätevä $\forall$:n luonti, koska z ei ole vapaa $\forall y : \varphi(c, y)$:ssä
  $\Rightarrow\ \forall y : \exists x : \varphi(x, y)$ pätevä sidotun muuttujan vaihto
  $\Rightarrow\ \forall y : \exists x : \varphi(x, y)$ $\exists$:n eliminointi päättyy pätevästi, koska c ei esiinny tuloksessa
- todistimme, että $\exists x : \forall y : \varphi(x, y) \ \Rightarrow\ \forall y : \exists x : \varphi(x, y)$

Esimerkki virheellisestä päättelystä (olettaen, että c ja z eivät esiinny $\varphi(x, y)$:ssä):

- $\forall x : \exists y : y > x$ tosi lähtökohta
  $\Rightarrow\ \forall z : \exists y : y > z$ pätevä sidotun muuttujan vaihto
  $\Rightarrow\ \exists y : y > x$ pätevä $\forall$:n eliminointi
  $c > x$ $\exists$:n eliminointi alkaa pätevästi, luotu c voi riippua x:sta
  $\Rightarrow\ \forall x : c > x$ virheellinen $\forall$:n luonti, koska oletettiin, että $x < c$
  $\Rightarrow\ \exists y : \forall x : y > x$ pätevä $\exists$:n luonti, mutta virheellisestä välituloksesta
- loppu uusiksi merkinnällä $c(x)$: alla $c(x)$-kaavat pätevät, jos esim. $c(x) = x + 1$
  $\Rightarrow\ \exists y : y > x$ tähän asti kuten edellä
  $c(x) > x$ $\exists$:n eliminointi alkaa pätevästi, c:n riippuvuus x:stä näkyy
  $\Rightarrow\ \forall x : c(x) > x$ pätevä $\forall$:n luonti, koska c:n riippuvuus x:stä näkyy
  $\Rightarrow\ \exists y : \forall x : y > x$ virheellinen $\exists$:n luonti sijoituksella $t = c(x)$, x on sidottu

Kuinka nämä lait voi muistaa?

- en tiedä, minä en muista niitä kaikkia!
- aika moni on ilmeinen, kun miettii symboleiden merkitystä
- esim. $\neg\forall x : \varphi(x) \;\Rightarrow\; \exists x : \neg\varphi(x)$
  - jos $\varphi(x)$ ei päde jokaisella x, niin on olemassa jokin x jolla $\varphi(x)$ ei päde
- esim. $\exists x : \forall y : \varphi(x, y) \;\Rightarrow\; \forall y : \exists x : \varphi(x, y)$
  - se x:n arvo, joka kelpaa vasemmalla, kelpaa oikeallakin
  - laki ei toimi toisinpäin, koska vaikka jokaisella y olisi sopiva x, *sama* x ei välttämättä kelpaa jokaiselle y
  - esim. $\forall y : \exists x : x = y$ pätee mutta $\exists x : \forall y : x = y$ ei päde

Tavoitteena *ei* ole oppia päättelemään yksityiskohtaisesti kuten sivun 162 esimerkissä

- jätämme sellaisen päättelemisen koneille!
- tavoitteena on oppia päättelemään ihmisille luontevalla tavalla
  - päättelyaskeleet saavat olla pitempiä kuin yksi lain soveltaminen
  - lähtökohtia, välituloksia ja lopputuloksia saa ilmaista (myös) suomeksi
  - päättelemisen oppimiseksi kurssilla on paljon päättelyesimerkkejä
- asioita suomeksi ilmaistessa tulee käyttää hyvin täsmällisiä ilmaisuja
  - esim. tyypillinen virhe on jättää epäselväksi, edustaako uusi symboli yhtä (kuten $\exists x$) vai jokaista (kuten $\forall x$) arvoa
  - aina ei ole helppoa sanoa suomeksi, esim. $P \to Q \to P \wedge Q$

- tarvitseeko sellaista kertoa, minkä voi olettaa olevan lukijalle selvää kertomattakin?
  - yleensä ei
  - opettaja voi kuitenkin käskeä kertomaan sellaista varmistuakseen, että se on kertojalle itselleen selvää!
  - on helppoa luulla, että jokin on lukijalle selvää vaikkei todellisuudessa olekaan (pelkään pahoin, että näissä luentoruuduissa on lukuisia esimerkkejä)
- päättelyn pitää perustua määritelmiin, luvattuihin lähtökohtiin sekä yleisesti tunnettuihin tosiasioihin ja päättelysääntöihin
  - esim. ristiriitatodistuksessa otetaan lähtökohdaksi tavoitteen negaatio
- matemaattisen todistuksen käsite on sosiaalinen
  - (nykynäkemyksen mukaan) todistuksen pitää aina olla ainakin periaatteessa palautettavissa pikkutarkoiksi, koneellisesti tarkastettaviksi askeliksi
  - sellaisesta palauttamisesta tulisi usein valtava määrä tylsää rutiinia
  - usein on itsestään selvää, että se onnistuisi, jos yritettäisiin

$\Rightarrow$ ei ole järkevää käyttää aikaa moiseen

Rajoitettu kvanttori, tapaus 1

$$\text{Muuttujaosuus} \ ::= \ \text{Muuttuja} \mid \text{Muuttuja} \in \text{Joukko} \mid \text{Muuttuja} \ ; \ \text{Rajain}$$

- $\forall x \in A : \varphi(x)$ tarkoittaa "jokaiselle joukkoon A kuuluvalle x:lle pätee $\varphi(x)$"
  - $\forall x \in A : \varphi(x) \ \equiv \ \forall x : x \in A \to \varphi(x)$
- $\exists x \in A : \varphi(x)$ tarkoittaa "jollekin joukkoon A kuuluvalle x:lle pätee $\varphi(x)$"
  - $\exists x \in A : \varphi(x) \ \equiv \ \exists x : x \in A \wedge \varphi(x)$

- näissä A on yleensä jokin yksinkertaisesti ilmaistavissa oleva joukko, esim. $\mathbb{R}$ tai Σ^*
  - siis A:ta käytetään usein kuten tyyppimääreitä ohjelmoinnissa, vrt. `int n`
  - emme määrittele tarkkaa syntaksia
- Σ^* on kaikkien Σ:n alkioista muodostettavissa olevien äärellisten jonojen joukko
  - esim. jos $\Sigma = \{a, b\}$, niin $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

Rajoitettu kvanttori, tapaus 2

- Rajain on tarkoitettu rajoittamaan taulukon indeksit sallitulle alueelle
  - esim. $\forall i; 1 \leq i < n : A[i] \leq A[i+1]$
  - harvinainen tämän kurssin ulkopuolella
  - senkään syntaksia ei ole tarkoin määritelty
- $\forall x; \xi(x) : \varphi(x) \;\Leftrightarrow\; \forall x : \xi(x) \to \varphi(x)$
- $\exists x; \xi(x) : \varphi(x) \;\Leftrightarrow\; \exists x : \xi(x) \wedge \varphi(x)$
- $\forall x; \xi(x) : \varphi(x)$ voi aiheuttaa yllätyksen, kun $\xi(x)$ tuottaa **F** jokaisella x
  - jokainen tyhjää väliä koskeva sellainen väite tuottaa **T**
  - $\forall x : \varphi(x)$ ei voi koskea tyhjää joukkoa (jollei esim. jako tapauksiin ole tuottanut sellaisen)
- esim. $\forall x : \varphi(x) \Rightarrow \exists x : \varphi(x)$, muttei välttämättä $\forall x; \xi(x) : \varphi(x) \Rightarrow \exists x; \xi(x) : \varphi(x)$
  - $\forall x; \xi(x) : \varphi(x) \Leftrightarrow \forall x : \xi(x) \to \varphi(x) \Rightarrow \exists x : \xi(x) \to \varphi(x) \Leftrightarrow \exists x : \neg\xi(x) \vee \varphi(x)$

- esim. $A[i]$ on pienempi kuin mikään muu taulukon $A[1...n]$ alkio, missä i on laillinen indeksi
  - $1 \leq i \leq n \wedge \forall j; 1 \leq j \leq n \wedge i \neq j : A[i] < A[j]$ sanoo sen oikein
  - kun $n = 0$, se tuottaa **F**, kuten pitääkin
  - j ei esiinny vapaana osakaavassa $1 \leq i \leq n$
  - $\varphi \wedge \forall x : \psi(x) \Leftrightarrow \forall x : \varphi \wedge \psi(x)$, jos x ei esiinny vapaana φ:ssä
  - $\forall j; 1 \leq j \leq n \wedge i \neq j : 1 \leq i \leq n \wedge A[i] < A[j] \equiv \mathbf{T}$, kun $n = 0$
- vastaavasti $\exists x; \xi(x) : \varphi(x)$ voi tuottaa yllätyksen, kun $\xi(x)$ tuottaa **F** jokaisella x
  - jos x ei esiinny vapaana φ:ssä, niin $\varphi \vee \exists x : \psi(x) \Leftrightarrow \exists x : \varphi \vee \psi(x)$, mutta ei välttämättä $\varphi \vee \exists x; \xi(x) : \psi(x) \Leftrightarrow \exists x; \xi(x) : \varphi \vee \psi(x)$
- tapauksissa $\forall x; \xi(x) : \varphi(x)$ ja $\exists x; \xi(x) : \varphi(x)$ päteviä lakeja voi johtaa em. määritelmien avulla
  - esim. $\exists x; \xi(x) : \varphi(x) \vee \psi(x)$
    $\Leftrightarrow \exists x : \xi(x) \wedge (\varphi(x) \vee \psi(x))$
    $\Leftrightarrow \exists x : \xi(x) \wedge \varphi(x) \vee \xi(x) \wedge \psi(x)$
    $\Leftrightarrow (\exists x : \xi(x) \wedge \varphi(x)) \vee \exists x : \xi(x) \wedge \psi(x)$
    $\Leftrightarrow (\exists x; \xi(x) : \varphi(x)) \vee \exists x; \xi(x) : \psi(x)$
  - siis $\exists x; \xi(x) : \varphi(x) \vee \psi(x) \Leftrightarrow (\exists x; \xi(x) : \varphi(x)) \vee \exists x; \xi(x) : \psi(x)$

Esimerkki: kaksi tapaa sanoa, että taulukko $A[1 \ldots n]$ on kasvavassa järjestyksessä

- $\forall i; 1 \leq i < n : A[i] \leq A[i+1]$ ja $\forall i : \forall j; 1 \leq i < j \leq n : A[i] \leq A[j]$
- perustelemme, että jälkimmäisestä seuraa edellinen

  $\forall i : \forall j; 1 \leq i < j \leq n : A[i] \leq A[j]$ valitaan $j = i + 1$

  $\Rightarrow \forall i; 1 \leq i < i+1 \leq n : A[i] \leq A[i+1]$ $i < i + 1 \leq n \Leftrightarrow i < n$

  $\Rightarrow \forall i; 1 \leq i < n : A[i] \leq A[i+1]$
- perustelemme, että edellisestä seuraa jälkimmäinen
  - oletetaan $\forall i; 1 \leq i < n : A[i] \leq A[i+1]$
  - valitaan mielivaltaiset sellaiset i' ja j', että $1 \leq i' < j' \leq n$

  $\Rightarrow j' = i' + k$ jollekin $k \in \mathbb{Z}^+$
  - valitaan sellainen mielivaltainen h, että $i' \leq h < j'$

  $\Rightarrow 1 \leq h < n$
  - valitsemalla h lähtökohdan i:ksi saadaan $A[h] \leq A[h+1]$

  $\Rightarrow A[i'] \leq A[i'+1] \leq A[i'+2] \leq \cdots \leq A[j']$

  $\Rightarrow A[i'] \leq A[j']$
  - siis $\forall i' : \forall j'; 1 \leq i' < j' \leq n : A[i'] \leq A[j']$

$\Rightarrow$ tavat ovat loogisesti yhtäpitävät

Lisää taulukosta $A[1 \ldots n]$ puhuvia esimerkkejä

- jokainen taulukossa esiintyvä arvo esiintyy ainakin kahdesti
  $\forall i; 1 \leq i \leq n : \exists j; 1 \leq j \leq n \wedge j \neq i : A[i] = A[j]$
- monimutkainen tapa sanoa **T**
  $\forall i; 1 \leq i \leq n : \exists j; 1 \leq j \leq n : A[i] = A[j]$
- monimutkainen tapa sanoa, että taulukko on tyhjä
  $\forall i; 1 \leq i \leq n : \exists j; 1 \leq j \leq n : A[i] < A[j]$

Lisää taulukosta $A[1 \ldots n]$ puhuvia esimerkkejä

- taulukon pienin alkio esiintyy (ainakin) kohdassa i, joka on laillisella alueella
  $1 \leq i \leq n \wedge \forall j; 1 \leq j \leq n : A[i] \leq A[j]$
  - i on parametri sanallisessa ilmauksessa

  $\Rightarrow$ kaavan totuusarvon tulee todennäköisesti riippua i:stä

  $\Rightarrow$ i:n tulee todennäköisesti esiintyä vapaana
  - i rajataan lailliselle alueelle, jotta kaava ei koskaan tuottaisi **U**
- taulukon pienin alkio esiintyy (vain) kohdassa i, joka on laillisella alueella
  $1 \leq i \leq n \wedge \forall j; 1 \leq j \leq n : i = j \vee A[i] < A[j]$
- monimutkainen tapa sanoa **F**
  $1 \leq i \leq n \wedge \forall j; 1 \leq j \leq n : A[i] < A[j]$

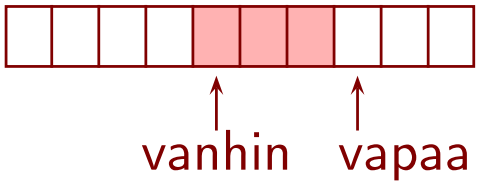
Verrataanpa vielä nämä:

- taulukon pienin alkio esiintyy (ainakin) kohdassa i, joka on laillisella alueella
  $$1 \leq i \leq n \wedge \forall j; 1 \leq j \leq n : A[i] \leq A[j]$$
- monimutkainen tapa sanoa **F**
  $\forall i : 1 \leq i \leq n \wedge \forall j; 1 \leq j \leq n : A[i] \leq A[j]$
- monimutkainen tapa sanoa, että taulukon kaikki alkiot ovat yhtäsuuret
  $\forall i; 1 \leq i \leq n : \forall j; 1 \leq j \leq n : A[i] \leq A[j]$
- monimutkainen tapa sanoa, että taulukko ei ole tyhjä
  $\exists i : 1 \leq i \leq n \wedge \forall j; 1 \leq j \leq n : A[i] \leq A[j]$
- sama hieman toisin muotoiltuna
  $\exists i; 1 \leq i \leq n : \forall j; 1 \leq j \leq n : A[i] \leq A[j]$

6 Modulaarinen aritmetiikka

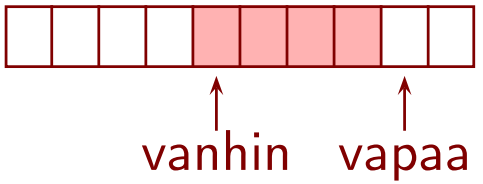
Miksi tärkeä?

- tietokoneet eivät laske kokonaisluvuilla, vaan esim. modulo $4\,294\,967\,296 = 2^{32}$
- rengasmaiset rakenteet ovat yleisiä, esim. rengaspuskuri
- tärkeä salausmenetelmä RSA perustuu siihen

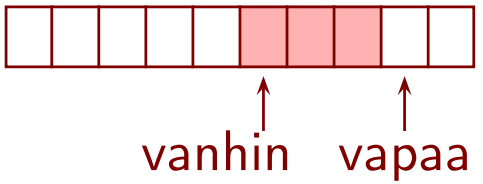


Aritmetiikassa modulo M

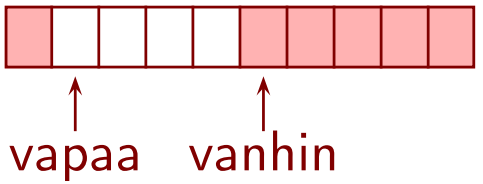
- $M \in \mathbb{Z}^+$ eli M on positiivinen kokonaisuluku
- jokaisen yhteen-, vähennys- ja kertolaskun lopuksi tuloksesta otetaan jakojäännös jaettuna M:llä



$\Rightarrow$ tulos on aina kokonaisluku väliltä $0, 1, \ldots, M-1$



- kertolaskun käänteisfunktio vaatii uudenlaisen jakolaskun (kolmas jakolasku tällä kurssilla!)
- $(3 \operatorname{div} 7) \bmod 10 = 0$, mutta ...
- ... $3 \div_{10} 7 = 9$, koska
  - $(9 \cdot 7) \bmod 10 = 63 \bmod 10 = 3$
  - ei ole muita $0 \leq k < 10$, joille $(k \cdot 7) \bmod 10 = 3$



$\Rightarrow$ käsittelemme jakolaskun ja potenssilaskun myöhemmin

Esim. aritmetiikassa modulo 5

- $3 + 4 =_5 2$, koska $7 \bmod 5 = 2$
- $3 - 4 =_5 4$, koska $-1 \bmod 5 = 4$
- $3 \cdot 4 =_5 2$, koska $12 \bmod 5 = 2$

Toinen tapa ajatella: $\ldots, n-2M, n-M, n, n+M, n+2M, \ldots$ ajatellaan yhtäsuuriksi

- esim. jos $M = 5$, niin $\ldots =_M -12 =_M -7 =_M -2 =_M 3 =_M 8 =_M 13 =_M \ldots$
- siis $n =_M m \Leftrightarrow \exists k \in \mathbb{Z} : n = m + kM \Leftrightarrow n \bmod M = m \bmod M$
  - on olemassa sellainen kokonaisluku k, että $n = m + kM$
  - esim. $-12 =_5 18$, koska $-12 = 18 + (-6) \cdot 5$
- lukujen $m + kM$ edustajana käytetään sitä, joka on välillä $0, \ldots, M-1$

Sama toisesta näkökulmasta

- kokonaisluvun n edustajana käytetään sitä lukua muotoa $n + kM$ ($k \in \mathbb{Z}$), jolle $0 \leq n + kM < M$
  - se on olemassa ja yksikäsitteinen
- esim. kun $M = 5$, lukujen 3, 18 ja -27 edustaja on 3, koska esim. $-27 + 6 \cdot 5 = 3$
- se on $n \bmod M = n - (n \operatorname{div} M)M$
  - kertaa jakoyhtälö sivu 78
- se löytyy toistuvasti lisäämällä tai vähentämällä M kunnes tulos on välillä $0, \ldots, M-1$

Vielä yksi tapa ajatella yhtäsuuruutta modulo M

- $n =_M m$ jos ja vain jos $n - m$ on jaollinen M:llä
- todistus: $n =_M m \;\Leftrightarrow\; \exists k \in \mathbb{Z} : n = m + kM \;\Leftrightarrow\; \exists k \in \mathbb{Z} : n - m = kM$

Yhteen-, vähennys- ja kertolaskun tapauksessa välivaiheissa ei tarvitse laskea $\bmod M$

- $$\begin{aligned}
&(\ (n \bmod M) + (m \bmod M)\ ) \bmod M && x \bmod M = x - (x \operatorname{div} M)M \\
&= (\ n - (n \operatorname{div} M)M + m - (m \operatorname{div} M)M\ ) \bmod M && \text{vaihdetaan värejä} \\
&= (\ n - (n \operatorname{div} M)M + m - (m \operatorname{div} M)M\ ) \bmod M && \text{ryhmitellään uudelleen} \\
&= (\ n + m - ((n \operatorname{div} M) + (m \operatorname{div} M))M\ ) \bmod M && M\text{:n monikerran poisto} \\
&= (\ n + m\ ) \bmod M
\end{aligned}$$
- $$\begin{aligned}
&(\ (n \bmod M)(m \bmod M)\ ) \bmod M \\
&= (\ (n - (n \operatorname{div} M)M)(m - (m \operatorname{div} M)M)\ ) \bmod M \\
&= (\ nm - n(m \operatorname{div} M)M - (n \operatorname{div} M)Mm + (n \operatorname{div} M)M(m \operatorname{div} M)M)\ ) \bmod M \\
&= (\ nm\ ) \bmod M
\end{aligned}$$

Joudumme jatkossa kirjoittamaan useasti $\bmod M$, $\bmod 10$ tms.

- jotta kaavoista olisi helpompi nähdä olennainen, kirjoitamme sen usein harmaalla
- esim. $(n \bmod M)(m \bmod M) = (nm) \bmod M$

Jakolasku aritmetiikassa $\bmod M$ ei aina onnistu, vaikka jakaja $\neq_M 0$

- merkitsemme (tällä kurssilla) $n \div_M m$, koska $\frac{n}{m}$ tarkoittaa muuta
- $x = n \div_M m$ pitää olla sellainen luku väliltä $0, \ldots, M-1$, että $(mx) \bmod M = n \bmod M$
- jos $M = 5$, niin jokainen jakolasku onnistuu kun jakaja $\neq_M 0$

$\cdot$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

$\div_5$	0	1	2	3	4
0	–	0	0	0	0
1	–	1	3	2	4
2	–	2	1	4	3
3	–	3	4	1	2
4	–	4	2	3	1

- jos $M = 10$ ja $m = 2$, niin $mx = 2x$ ja $(mx) \bmod M = (2x) \bmod 10$
- ne ovat parillisia

$\Rightarrow (2x) \bmod 10$ ei voi olla 3

$\Rightarrow 3 \div_{10} 2$ ei ole olemassa

- sekä $(2 \cdot 2) \bmod 10 = 4$ että $(7 \cdot 2) \bmod 10 = 4$, joten kumpi $4 \div_{10} 2$ on, 2 vai 7?
- päätös: puhumme jakolaskusta vain kun osamäärä on olemassa ja yksikäsitteinen

Yhtäsuuruutta $=_M$ käytettäessä yhteen-, vähennys- ja kertolaskuissa, mutta ei jakolaskuissa, voi laskea aivan kuten kokonais- ja reaaliluvuillakin

Tekijä

- olkoot $n \in \mathbb{Z}$ ja $m \in \mathbb{Z}$
- n on m:n **tekijä**, jos ja vain jos on olemassa sellainen kokonaisluku k, että $m = nk$

$\Leftrightarrow$ $n = m = 0$ tai jakolasku m jaettuna n:llä menee tasan

$\Leftrightarrow$ $n = m = 0$ tai $m \bmod n = 0$

Suurin yhteinen tekijä

- olkoot $n \in \mathbb{N}$ ja $m \in \mathbb{N}$ eli n ja m ovat luonnollisia lukuja, ja ainakin toinen $\neq 0$
- 1 on molempien tekijä
- jos $n > 0$ niin mikään kokonaisluku, joka on suurempi kuin n, ei voi olla n:n tekijä
  $\Rightarrow$ ei voi olla molempien tekijä

$\Rightarrow$ on olemassa n:n ja m:n **suurin yhteinen tekijä**
  - suurin kokonaisluku, joka on sekä n:n että m:n tekijä

- merkitsemme sitä $\mathrm{syt}(n, m)$
- yleistys negatiivisille: $\mathrm{syt}(-n, m) = \mathrm{syt}(n, -m) = \mathrm{syt}(-n, -m) = \mathrm{syt}(n, m)$

Mitkä luvut voidaan muodostaa n:stä ja m:stä yhteen- ja vähennyslaskuilla?

- esimerkki: jos $n = 50$ ja $m = 80$, niin voidaan muodostaa mm.
  - $80 + 80 + 50 + 50 + 50 = 310$
  - $50 - 80 = -30$
  - $50 - 30 = 20 = 2 \cdot 50 - 80$

- niitä ovat mm. 0, n, m, $-m$, $2n + 7m$, $4n - 11m$ ja $-4n + 11m$
  - 0 voidaan muodostaa $n - n$
- voidaan muodostaa ainakin kaikki luvut muotoa $xn + ym$, missä $x \in \mathbb{Z}$ ja $y \in \mathbb{Z}$
- ei muita, koska $(x_1 n + y_1 m) \pm (x_2 n + y_2 m) = (x_1 \pm x_2)n + (y_1 \pm y_2)m$

Jos $n \in \mathbb{Z}^+$, niin $\mathrm{syt}(n, 0) = \mathrm{syt}(0, n) = n$

- $n = n \cdot 1$ ja $0 = n \cdot 0$, eikä mikään isompi ole n:n tekijä

Lause: Jos $n \in \mathbb{Z}^+$ ja $m \in \mathbb{Z}^+$, niin $\mathrm{syt}(n, m)$ on pienin positiivinen luku muotoa $xn + ym$, missä $x \in \mathbb{Z}$ ja $y \in \mathbb{Z}$

- esim. $\mathrm{syt}(50, 80) = 10$
  - voidaan muodostaa vain 10:llä jaollisia
  - $10 = -3 \cdot 50 + 2 \cdot 80$

Miksi on varmaa, että "pienin positiivinen ..." on olemassa?

- ei itsestään selvää
  - kokonaislukujoukossa $\{-6, -5, -4\}$ ei ole pienintä positiiviista
  - reaalilukujoukossa $\{\ldots, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}\}$ ei ole pienintä positiiviista
- kaikki muotoa $xn + ym$, missä $n \in \mathbb{Z}^+$, $m \in \mathbb{Z}^+$, $x \in \mathbb{Z}$ ja $y \in \mathbb{Z}$, olevat luvut ovat kokonaislukuja
- niissä on positiivisia, esim. $1n + 0m = n$
- jokaisessa epätyhjässä alhaalta rajoitetussa $\mathbb{Z}$:n osajoukossa on pienin alkio

Syt-lauseen todistus

- olkoon $n \in \mathbb{Z}^+$ ja $m \in \mathbb{Z}^+$
- tarkastellaan kaikkia lukuja muotoa $xn + ym$, missä $x \in \mathbb{Z}$ ja $y \in \mathbb{Z}$
  - siis n ja m säilyttävät arvonsa, ja x ja y käyvät läpi kaikki kokonaisluvut
- merkitsemme pienintä positiivista $s = x'n + y'm$
- jos k on sekä n:n että m:n tekijä, niin k on jokaisen luvun $xn + ym$ tekijä
  - todistus: $n = kn'$ ja $m = km' \Rightarrow xn + ym = xkn' + ykm' = k(xn' + ym')$

$\Rightarrow$ jokainen n:n ja m:n yhteinen tekijä on myös s:n tekijä

$\Rightarrow$ (1) $\mathrm{syt}(n, m)$ on s:n tekijä

- myös $n \bmod s$ on tarkasteltavaa muotoa, koska $n \bmod s = n - (n \operatorname{div} s)s$
  $= n - (n \operatorname{div} s)(x'n + y'm) = (1 - (n \operatorname{div} s)x')n - ((n \operatorname{div} s)y')m$
- jakoyhtälössä (sivu 78) luvataan, että $n \bmod s < s$

$\Rightarrow$ $n \bmod s = 0$, koska muutoin syntyisi ristiriita s:n valinnan kanssa
  - jos pätisi $n \bmod s > 0$, niin $n \bmod s$ olisi pienempi positiivinen tarkasteltavaa muotoa oleva luku kuin s

$\Rightarrow$ (2) s on n:n tekijä

- samasta syystä (3) s on myös m:n tekijä

$\Rightarrow$ (1), (2) ja (3) $\Rightarrow$ $\mathrm{syt}(n, m) = s$

Jakolasku $n \div_M m$ onnistuu jos ja vain jos $m \neq_M 0$ ja $\mathrm{syt}(m, M) = 1$

- olkoon $s = \mathrm{syt}(m, M)$
  $\Rightarrow$ reaalilukujen jakolaskut $\frac{m}{s}$ ja $\frac{M}{s}$ menevät tasan
- jos $s \neq 1$ ja x toteuttaa ehdon $mx =_M n$ eli $(mx) \bmod M = n \bmod M$, niin
  - $s > 1$
  - $x \neq_M x + \frac{M}{s}$, koska $0 < \frac{M}{s} < M$
  - $x + \frac{M}{s}$ toteuttaa ym. ehdon, koska $m(x + \frac{M}{s}) = mx + \frac{m}{s}M =_M mx =_M n$

  $\Rightarrow$ osamäärä on monikäsitteinen

$\Rightarrow$ jos $s \neq 1$, niin osamäärä joko ei ole olemassa tai on monikäsitteinen

- jos $s = 1$, niin
  - on olemassa sellaiset $z \in \mathbb{Z}$ ja $y \in \mathbb{Z}$, että $zm + yM = 1$

  $\Rightarrow nzm + nyM = n$

  $\Rightarrow mnz =_M n$

  $\Rightarrow nz$ toteuttaa ehdon $mx =_M n$
  - kun k käy läpi luvut $0, \ldots, M-1$, niin $k \bmod M$ saa M eri arvoa

  $\Rightarrow$ myös $(mkz) \bmod M$ ja $(kz) \bmod M$ saavat M eri arvoa

  $\Rightarrow$ kaikki luvut $0, \ldots, M-1$ tulevat käytetyiksi

  $\Rightarrow$ ei voi olla siten, että $0 \leq x_1 < x_2 < M$ ja $mx_1 =_M mx_2$

$\Rightarrow$ jos $s = 1$, niin osamäärä on olemassa ja yksikäsitteinen
  - $n \div_M m = (n(1 \div_M m)) \bmod M$, missä $1 \div_M m$ on edellä löydetty z
  - ym. lukua z merkitään tavallisesti m^{-1}

Saako jakolaskussa jättää $\bmod M$ laskematta välivaiheissa?

- nimittäjän osalta kysymys ei nouse esiin, koska $n \div_M m$ ei lasketa tavallisen jakolaskun avulla vaan
  - etsimällä se luku m^{-1}, jolle $0 \leq m^{-1} < M$ ja $(mm^{-1}) \bmod M = 1$, ja
  - laskemalla $(nm^{-1}) \bmod M$
- osoittajassa saa: $(n + M)m^{-1} = nm^{-1} + Mm^{-1} =_M nm^{-1}$

Potenssilaskussa on sekaannuksen vaara

- esim. jos $M = 5$, niin $2^1 \bmod M = 2$ mutta $2^{1+M} \bmod M = 2^6 \bmod M = 64 \bmod M = 4 \neq 2$
- esim. jos $M = 2^8 = 256$, niin $10^3 \bmod M = 232$, mutta $10^m \bmod M = 0$ jos $m \geq 8$

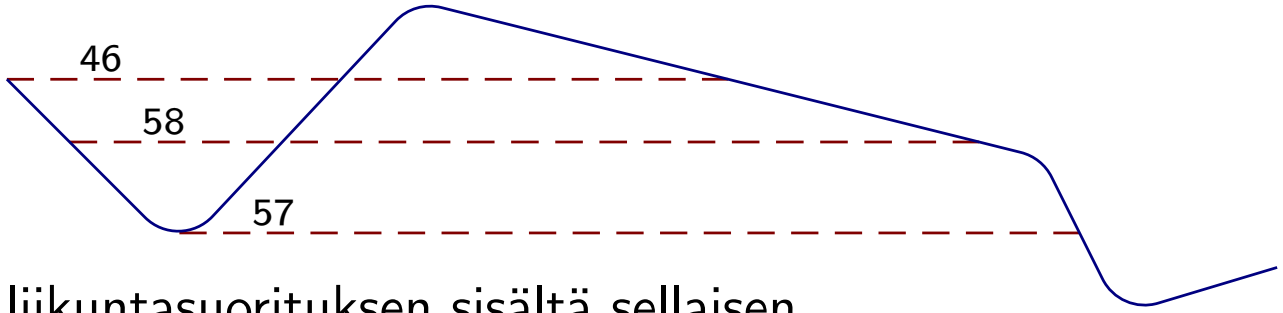
$\Rightarrow$ eksponentissa ei voida soveltaa yhtäsuuruutta $=_M$

- kantaluvussa voidaan: $n^m = \underbrace{n \cdots n}_{m \text{ kpl.}} =_M \underbrace{(n \bmod M) \cdots (n \bmod M)}_{m \text{ kpl.}} = (n \bmod M)^m$

7 Päättelyesimerkkejä

7.1 Takahuippu

Epätyhjästä taulukosta `korkeus` on löydettävä mahdollisimman pitkän osuuden, jonka loppukohta on vähintään yhtä korkealla kuin alkukohta, pituus



- esim. pitkän liikuntasuorituksen sisältä sellaisen osuuden pituus, jossa ei ole netto alamäkihyötyä
  - vaikka todellisuus on jatkuva käyrä, korkeustiedot ovat esim. 10m välein
- osuus *kelpaa*, jos ja vain jos sen loppukohta on ainakin yhtä korkealla kuin alkukohta
- vertaamme helppoa mutta hidasta algoritmia keskivaikeaan mutta nopeaan

Helppo algoritmi

- kokeillaan kaikki alkukohdat ja kaikki loppukohdat
- valitaan paras tulos
- yksinkertainen, tehtävää läheisesti muistuttava toimintaperiaate
  $\Rightarrow$ helppo vakuuttautua, että toimii oikein

Toteutus C++:lla

- `korkeus` indeksoidaan `0,...,nn-1`
- `ii` käy läpi kaikki mahdolliset alkukohdat: kaikki kohdat
- `jj` käy läpi kaikki mahdolliset loppukohdat: kaikki kohdat, joille `jj` $\geq$ `ii`
- `paras` pitää kirjaa parhaasta löytyneestä tuloksesta
  - sille on annettava aluksi jokin arvo, joka ei voi vääristää lopputulosta
  - ainakin 0:n pituinen kelpaava osuus on olemassa: pysytään alkukohdassa

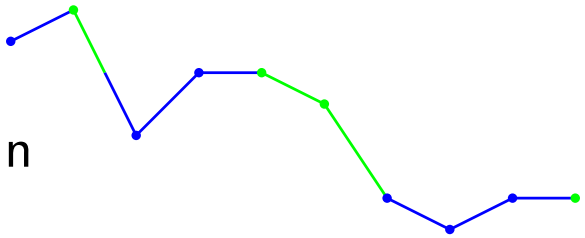
  $\Rightarrow$ voidaan sijoittaa aluksi `paras = 0`
  - `korkeus[ jj ] >= korkeus[ ii ]` testaa, että osuus kelpaa
  - `jj - ii > paras` testaa, että osuus on parempi kuin jo tutkitut

```
unsigned helppo( unsigned korkeus[], unsigned nn ){
  unsigned paras = 0;
  for( unsigned ii = 0; ii < nn; ++ii ){
    for( unsigned jj = ii; jj < nn; ++jj ){
      if( korkeus[ jj ] >= korkeus[ ii ] && jj - ii > paras ){
        paras = jj - ii;
      }
    }
  }
  return paras;
}
```

Ajan kulutus sekunteina

nn	1 000	3 162	10 000	31 623	100 000	316 228	1 000 000
helppo	0	0	0	2	23	238	2 344
nopea	0	0	0	0	0	0	0

Nopean algoritmin peruskäsite: takahuippu



- jos $0 \leq i < n$, niin *i:lle paras loppukohta* on mahdollisimman suuri j, jolle $i \leq j < n$ ja $\texttt{korkeus}[j] \geq \texttt{korkeus}[i]$
  - jos mahdollisimman pitkä osuus on $a \ldots \ell$, niin ℓ on a:lle paras loppukohta
- jos osuus $i \ldots j$ kelpaa ja on olemassa sellainen kohta j', että $j < j' < n$ ja $\texttt{korkeus}[j'] \geq \texttt{korkeus}[j]$, niin osuus $i \ldots j'$ kelpaa ja on pitempi kuin osuus $i \ldots j$

  $\Rightarrow$ parhaalla loppukohdalla i:lle on erityinen ominaisuus:
  sen jälkeen ei enää tule ainakin yhtä korkeita kohtia
- annamme nimen kohdille, joilla on tämä ominaisuus: *takahuippu*

$$\text{takahuippu}(j) \Leftrightarrow 0 \leq j < n \wedge \forall k; j < k < n : \texttt{korkeus}[k] < \texttt{korkeus}[j]$$

  $\Rightarrow$ *i:lle paras loppukohta on takahuippu*
- jos $0 \leq j < j' < n$ ja sekä j että j' on takahuippu, niin $\texttt{korkeus}[j] > \texttt{korkeus}[j']$
  - kaikki kohdat j:n jälkeen ovat alempana kuin j, joten myös j' on

  $\Rightarrow$ *seuraava takahuippu on edellistä alempana*

$\Rightarrow$ i:lle paras loppukohta löytyy selaamalla takahuippuja vasemmalta alkaen kunnes ne loppuvat tai niiden korkeus alittaa i:n korkeuden

- jos $0 \leq i < i' < n$, niin joko $\texttt{korkeus}[i'] \geq \texttt{korkeus}[i]$ tai $\texttt{korkeus}[i'] < \texttt{korkeus}[i]$
- jos $\texttt{korkeus}[i'] \geq \texttt{korkeus}[i]$, niin
  - jos $i' \ldots j$ on kelpaava osuus, niin $\texttt{korkeus}[i] \leq \texttt{korkeus}[i'] \leq \texttt{korkeus}[j]$

  $\Rightarrow$ $i \ldots j$ on kelpaava osuus

  $\Rightarrow$ i' ei voi olla mahdollisimman pitkän kelpaavan osuuden alkukohta
  - ei tarvitse etsiä i':lle parasta loppukohtaa
- jos $\texttt{korkeus}[i'] < \texttt{korkeus}[i]$, niin i':lle paras loppukohta on sama tai alempana kuin i:lle paras loppukohta
  - jokainen paras loppukohta on takahuippu
  - takahuiput ovat laskevassa järjestyksessä

  $\Rightarrow$ i':lle paras loppukohta on sama tai enemmän oikealla kuin i:lle paras loppukohta
  - jos se on sama, niin sitä ei tarvitse tutkia i':lle, koska i on sille parempi alkukohta

  $\Rightarrow$ voidaan jatkaa takahuippujen selaamista oikealle siitä mihin jäätiin
- viimeisen kohdan jälkeen ei tule mitään kohtia

  $\Rightarrow$ ei tule ainakin yhtä korkeita kohtia

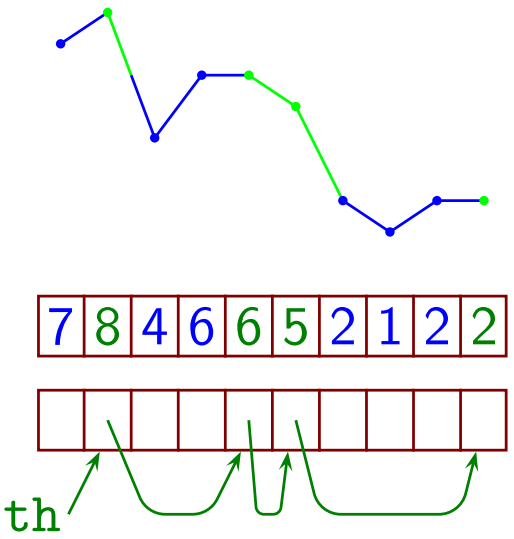
  $\Rightarrow$ *viimeisessä kohdassa on takahuippu*

Nopea algoritmi

- etsitään takahuiput ja talletetaan ne listaksi, jota voi selata vasemmalta oikealle
- selataan alkukohdat ja ym. lista tahdistetusti

Nopea algoritmi C++:lla

```
unsigned nopea( unsigned korkeus[], unsigned nn ){
```



- etsitään takahuiput selaamalla `korkeus` takaperin
  - `th` on korkein löydetty kohta
  - ensimmäiseksi löydetään viimeinen kohta, joten alustetaan `th = nn-1`
  - kohta on takahuippu, jos ja vain jos se on viimeinen kohta tai korkeampi kuin korkein aiemmin löydetty kohta
  - `seuraava_th[ ii ]` on takahuippua `ii` seuraava (eli uusin sitä ennen löydetty) takahuippu

```
  unsigned th = nn-1, *seuraava_th = new unsigned[ nn ];
  for( unsigned ii = nn; ii--; ){
    if( korkeus[ ii ] > korkeus[ th ] ){
      seuraava_th[ ii ] = th; th = ii;
    }
  }
```

- etsitään paras tulos selaamalla aloituskohdat ja takahuiput etuperin
  - `for`-silmukka selaa aloituskohdat
  - `while`-silmukka selaa takahuippuja kunnes ne loppuvat (`th == nn-1`) tai ohitetaan `ii`:lle paras loppukohta (`while`:n ehto ei toteudu)
  - pidetään kirjaa parhaasta selatun osuuden pituudesta

```
unsigned paras = 0;
for( unsigned ii = 0; ii < nn; ++ii ){
  while( korkeus[ th ] >= korkeus[ ii ] ){
    if( th - ii > paras ){ paras = th - ii; }
    if( th == nn-1 ){ delete[] seuraava_th; return paras; }
    th = seuraava_th[ th ];
  }
}
```

- on helppo nähdä, että algoritmi kokeilee vain korkeusehdon toteuttavia osuuksia ja valitsee kokeilemistaan parhaan
- lopussa on rivi, jolle ei koskaan tulla, mutta C++-kääntäjäni suuttui ilman sitä

```
delete[] seuraava_th; return paras; }
```

- voiko (jokainen) kaikkein paras osuus $a \ldots \ell$ jäädä kokeilematta?
  - `ii` kokeilee jokaisen alkukohdan

  $\Rightarrow$ `ii` kokeilee parhaan osuuden alkukohdan a
  - parhaan osuuden loppukohta ℓ on takahuippu
  - jos $\texttt{ii} < a$, niin $\texttt{korkeus}[\ell] < \texttt{korkeus}[\texttt{ii}]$ (miksi?)

  $\Rightarrow$ `th` ei ohita kohtaa ℓ kun $\texttt{ii} < a$

  $\Rightarrow$ suorituksen aikana on hetki, jolloin $\texttt{ii} = a$ ja $\texttt{th} = \ell$

  $\Rightarrow$ ohjelma tutkii jokaisen kaikkein parhaan osuuden

Lähdeviitteet

[ACM13] ACM & IEEE: Computer Science Curricula 2013 : Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, 2013
https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf

[AnW09] Glenda Anthony & Margaret Walshaw: Effective pedagogy in mathematics. UNESCO/IBE 2009 http://www.ibe.unesco.org/fileadmin/user_upload/Publications/Educational_Practices/EdPractices_19.pdf

[Ben86] Jon Bentley: Programming Pearls. Addison-Wesley 1986

[Blo06] Joshua Bloch: Extra, Extra - Read All About It: Nearly All Binary Searches and Mergesorts are Broken. Blogi 2.6.2006
https://ai.googleblog.com/2006/06/extra-extra-read-all-about-it-nearly.html

[Cha05] Robert N. Charette: Why Software Fails. IEEE Spectrum 42(9), 2005, 42–49
https://spectrum.ieee.org/computing/software/why-software-fails

[Gib10] Graham Gibbs: Using assessment to support student learning. Leeds Metropolitan University, 2010 http://eprints.leedsbeckett.ac.uk/2835/1/100317_36641_Formative_Assessment3Blue_WEB.pdf

[HiG07] James Hiebert & Douglas A. Grouws: The Effects of Classroom Mathematics Teaching

on Students' Learning. Frank K. Lester Jr. (toim.): Second Handbook of Research on Mathematics Teaching and Learning, 2007, 371–404

[HuN19] Thomas Huckle & Tobias Neckel: Bits and Bugs – A Scientific and Historical Review of Software Failures in Computational Science. SIAM 2019

[Kar17] Marjaana Karhunkorpi: Malmin Prisma hukkui vihapostiin. Aamulehti 16.2.2017 `https://www.aamulehti.fi/a/24283137`

[KSF01] Jeremy Kilpatrick & Jane Swafford & Bradford Findell (toim.): Adding It Up: Helping Children Learn Mathematics. National Academy of Sciences 2001 `http://nap.edu/9822`

[KB+05] Barbara Kitchenham & David Budgen & Pearl Brereton & Philip Woodall: An Investigation of Software Engineering Curricula. Journal of Systems and Software 74(3): 325–335, 2005

[Knu73] Donald Knuth: The Art of Computer Programming, Volume III: Sorting and Searching. Addison-Wesley 1973

[Lai17] Annu Laine: Pyörätuolissa istuvaa miestä nöyryytettiin Prisman kassalla. Aamulehti 15.2.2017 `https://www.aamulehti.fi/a/24277883`

[Let00] Timothy Lethbridge: What Knowledge Is Important to a Software Professional? IEEE Computer 33(5): 44–50, 2000

[McC11] Steve McConnell: Origins of 10X — How Valid is the Underlying Research? Blogi

9.1.2011 https://www.construx.com/blog/the-origins-of-10x-how-valid-is-the-underlying-research/

[Pat88] Richard Pattis: Textbook Errors in Binary Searching. Proc. 19st SIGCSE Technical Symposium on Computer Science Education, ACM 1988, 190–194

[PuA09] Antti Puhakka & Kirsti Ala-Mutka: Survey on the Knowledge and Education Needs of Finnish Software Professionals, Tampere University of Technology 2009

[Sur07] Sami Surakka: What Subjects and Skills Are Important for Software Developers? Communications of the ACM 50(1): 73–78, 2007