Sovellettu predikaattilogiikka

Lisäaineistoa 2024

Antti Valmari

Jyväskylän yliopisto
Informaatioteknologian tiedekunta

1 Logiikka yhtälöiden 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 määritelty temppu, ovat lopputuloksetkin yhtäsuuret
  - (esimerkiksi nollalla jakaminen ei ole määritelty)
  - 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 $(3x-12)+12=12$

- 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
- vähennyslaskun määritelmästä seuraa $(3x-12)+12=3x$
- 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 määritelty 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 pitkähkö

Miksi edellä puhuttiin "määritellyistä" 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 $x=4 \Rightarrow 3x=12$
- $3x-12=0 \Leftarrow x=4$ voidaan tietenkin tarkastaa myös sijoittamalla
  - $x=4 \Rightarrow 3x-12=3\cdot 4-12=12-12=0$

Toimiiko $\Leftrightarrow$ aina, kun $=$:n molemmille puolille tehdään sama määritelty 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: menisinkö yliopistolle polkupyörällä?

- sataa
- jos laitan sadevaatteet, niin kastun hiestä
- jos en laita sadevaatteita, niin kastun sateesta

$\Rightarrow$ jos menen pyörällä, niin kastun joka tapauksessa

- havainto: eri haaroissa päättely voi vedota aivan eri asioihin

Esimerkki päättelyn jakamisesta haaroihin: puolitushaun **if-then-else**

$\{\, 1 \le a < y \le n+1 \,\}$

3	$v := (a+y) \operatorname{div} 2$	$\{\, 1 \le a \le v < y \le n+1 \,\}$
4	**if** $A[v] < x$ **then** $a := v+1$	$\{\, 1 < a \le y \le n+1 \,\}$
5	**else** $y := v$	$\{\, 1 \le a \le y < n+1 \,\}$

$\{\, 1 \le a \le y \le n+1 \,\}$

- rivin 4 alussa $1 \le v < y \le n+1$ ja luvut kokonaisia, joten $1 < v+1 \le y \le n+1$
  - jos mennään **then**-haaraan, niin sen lopussa $1 < a \le y \le n+1$
- rivin 4 alussa $1 \le a \le v < n+1$
  - jos mennään **else**-haaraan, niin sen lopussa $1 \le a \le y < n+1$

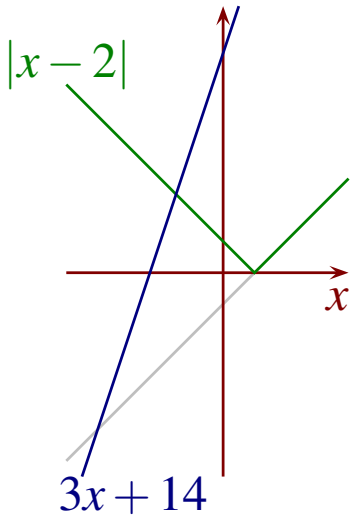
$\Rightarrow$ molemmissa tapauksissa rivin 6 alussa $1 \le a \le y \le n+1$

Esimerkki päättelyn jakamisesta haaroihin: ratkaise $|x-2| = 3x+14$

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

$\Rightarrow$ joko $x-2<0$ ja $-(x-2) = 3x+14$, tai $x-2 \geq 0$ ja $x-2 = 3x+14$

- tapaus $x-2<0$: $-(x-2) = 3x+14$
  $\Leftrightarrow -x+2 = 3x+14$
  $\Leftrightarrow 2-14 = 3x+x$
  $\Leftrightarrow -12 = 4x$
  $\Leftrightarrow x = -3$
- tapaus $x-2 \geq 0$: $x-2 = 3x+14$
  $\Leftrightarrow -2-14 = 3x-x$
  $\Leftrightarrow -16 = 2x$
  $\Leftrightarrow x = -8$
  $\Leftrightarrow$ **F**



- **F** tarkoittaa "false" eli epätosi eli että mikään luku ei kelpaa x:ksi
- $x = -8 \Leftrightarrow$ **F** ei normaalisti ole pätevä, koska sen vasen puoli on totta mutta oikea puoli ei ole kun $x = -8$
- nyt $x = -8$ ei kuitenkaan kelpaa vastaesimerkiksi lähtöoletuksen $x-2 \geq 0$ vuoksi

Päättelyaskeleen pätevyys voi siis riippua asiayhteydestä

- $x = -8 \Leftrightarrow$ **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. **puheenaihe** (**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.
- puheenaiheiden rajojen rikkominen johtaa usein sekopäisiin ilmauksiin
  - esim. Jussin äiti + 3 = aurinkoinen sää

2. tilapäiset lähtöoletukset
   - esim. tapauksen ehto $x-2 \geq 0$
   - tilapäisten merkintöjen ominaisuudet, esim. annetaan nimi luvulle, 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 puheenaiheen 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 puheenaiheen 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

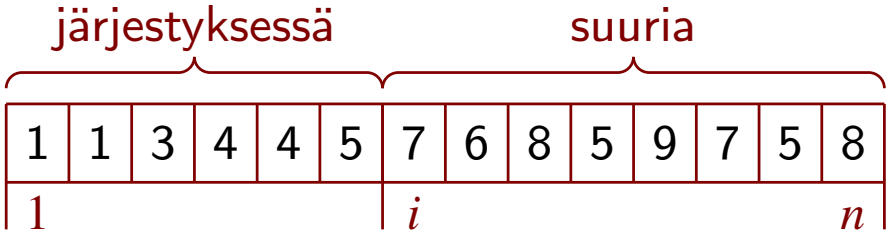
- niinkin olisi voitu tehdä
- käytännön päättelyissä tehdään usein pitkiä loikkia
  - esim. $3x - 12 = 0 \Leftrightarrow 3x = 12$ esitetään yleensä korkeintaan yhtenä askeleena, vaikka se koostuu monesta pikkuaskeleesta, kuten näimme
- on vaikea vetää raja sille, kuinka pitkä loikka hyväksytään pätevästi 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 pätevästi 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!

$$\begin{aligned} & x = x+1 && \text{vähennetään kummaltakin puolelta } x \\ \Rightarrow\; & 0 = 1 && \text{lisätään kummallekin puolelle } 1 \\ \Rightarrow\; & 1 = 2 \end{aligned}$$

$\Rightarrow$ 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

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. $|x-2|=3x+14$
  $\Leftrightarrow\ x-2<0\wedge-(x-2)=3x+14\ \vee\ x-2\geq 0\wedge x-2=3x+14$
  $\Leftrightarrow\ x<2\wedge x=-3\ \vee\ x\geq 2\wedge x=-8$
  $\Leftrightarrow\ x=-3\ \vee\ \mathbf{F}$
  $\Leftrightarrow\ x=-3$
- 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 $(x+2)(x+2)=(x+2)x+(x+2)\cdot 2$ ja $(x+2)x=xx+2x$ ja $(x+2)\cdot 2=x\cdot 2+2\cdot 2$ ja $2x+2x=(2+2)x=4x$

  $\Rightarrow (x+2)^2=(x+2)(x+2)=(x+2)x+(x+2)\cdot 2=$
  $xx+2x+x\cdot 2+2\cdot 2=x^2+2x+2x+4=x^2+4x+4$
  - lisäksi käytettiin tietoa $-4-12=-16$
- $(x+2)^2-16=0 \Leftrightarrow (x+2)^2=16$ ei sisällä uutta
- $(x+2)^2=16 \Leftrightarrow x+2=-4 \vee x+2=4$
  - 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

- jos
  - $osakaava1 \Leftrightarrow osakaava2$,
  - osakaavoissa 1 ja 2 ei ole määrittelemättömiä toimintoja, ja
  - $isompi\_kaava$:ssa ei ole symboleita $\forall$ eikä $\exists$

  niin $osakaava1$:n saa korvata $osakaava2$:lla $isompi\_kaava$:ssa
- koska $x+2=-4 \Leftrightarrow x=-6$ ja $x+2=4 \Leftrightarrow x=2$, pätee

$$x+2=-4 \vee x+2=4 \Leftrightarrow x=-6 \vee x+2=4$$
$$\Leftrightarrow x=-6 \vee x+2=4 \Leftrightarrow x=-6 \vee x=2$$

$$\Rightarrow \quad x+2=-4 \vee x+2=4 \Leftrightarrow x=-6 \vee x=2$$

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

$$\begin{aligned} & x=-6 \vee x=2 \\ \Leftrightarrow\ & x+6=0 \vee x-2=0 \\ \Leftrightarrow\ & (x+6)(x-2)=0 \\ \Leftrightarrow\ & x^2+6x+x\cdot(-2)+6\cdot(-2)=0 \\ \Leftrightarrow\ & x^2+4x-12=0 \end{aligned}$$

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 ja yleisemminkin tieteissä 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 (ja jopa keksiä!) 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
- siis
  - ratkaisua etsiessä ei tarvitse noudattaa matematiikan sääntöjä, saa arvata jne.
  - löytynyt ratkaisu pitää kirjoittaa muotoon, joka noudattaa matematiikan sääntöjä

Miksi $|x-2| = 3x+14 \Leftrightarrow \ldots \Leftrightarrow x = -3$ on pätevä?

$$
\begin{aligned}
& |x-2| = 3x+14 \\
\Leftrightarrow\ & x-2 < 0 \wedge -(x-2) = 3x+14 \ \vee\ x-2 \geq 0 \wedge x-2 = 3x+14 && (1)\\
\Leftrightarrow\ & x < 2 \wedge x = -3 \ \vee\ x \geq 2 \wedge x = -8 && (2)\\
\Leftrightarrow\ & x = -3 \ \vee\ \mathbf{F} && (3)\\
\Leftrightarrow\ & x = -3 && (4)
\end{aligned}
$$

- vaihe 1 seuraa suoraan itseisarvon määritelmästä
- osakaavojen korvaamisen yhtäpitävillä ehdot toteutuvat
  - nollalla jakoa, neliöjuuria eikä muitakaan määrittelemättömiä toimintoja esiinny
  - symboleita $\forall$ ja $\exists$ ei esiinny
- vaiheessa 2 ratkaistiin
  - epäyhtälöt $x-2<0$ ja $x-2 \geq 0$
  - yhtälöt $-(x-2) = 3x+14$ ja $x-2 = 3x+14$
- $x < 2 \wedge x = -3 \Leftrightarrow x = -3$, koska $-3 < 2$ on totta
  - x:n arvo -3 tekee molemmista puolista totta ja muut x:n arvot epätotta
- $x \geq 2 \wedge x = -8 \Leftrightarrow \mathbf{F}$, koska $-8 \geq 2$ ei ole totta
  - x:n arvo -8 tekee $(x \geq 2)$:sta epätotta ja muut x:n arvot $(x = -8)$:sta epätotta

  $\Rightarrow$ vasen puoli on epätosi kaikilla x:n arvoilla, joten $\mathit{vasen} \Leftrightarrow \mathbf{F}$

$\Rightarrow$ vaihe 3

- vaihe 4 seuraa siitä, että aina pätee $\mathit{väite} \vee \mathbf{F} \Leftrightarrow \mathit{väite}$

Ruudun 17 esimerkeistä on vielä yksi jäljellä

$$
\begin{aligned}
& 2x+3y=6 \ \wedge \ y-x=7 \\
\Leftrightarrow \ & 2x+3y=6 \ \wedge \ y=x+7 && (1)\\
\Leftrightarrow \ & 2x+3(x+7)=6 \ \wedge \ y=x+7 && (2)\\
\Leftrightarrow \ & 5x+21=6 \ \wedge \ y=x+7 && (3)\\
\Leftrightarrow \ & x=-3 \ \wedge \ y=4 && (4)
\end{aligned}
$$

- nytkin osakaavojen korvaamisen yhtäpitävillä ehdot toteutuvat
  - missään ei $\sqrt{negatiivinen}$, $\frac{jotain}{0}$, …, $\forall$ eikä $\exists$
- vaihe 1 seuraa siitä, että $y-x=7 \ \Leftrightarrow \ y=x+7$
- kaavassa, jossa ei esiinny $\forall$ eikä $\exists$, saa lausekkeen korvata yhtäsuurella

$\Rightarrow$ vaihe 3: $2x+3(x+7)=5x+21$, joten
$2x+3(x+7)=6 \wedge y=x+7 \ \Leftrightarrow \ 5x+21=6 \wedge y=x+7$

- vaihe 4 voidaan perustella muun muassa seuraavasti
  - x:n arvolla -3 sekä $5x+21=6$ että $x=-3$ tuottavat **T** eli tosi (true), joten

  $\Rightarrow$ ylempi rivi sievenee $5x+21=6 \wedge y=x+7 \ \Leftrightarrow \ \mathbf{T} \wedge y=-3+7 \ \Leftrightarrow \ y=4$
  - alempi rivi sievenee $x=-3 \wedge y=4 \ \Leftrightarrow \ \mathbf{T} \wedge y=4 \ \Leftrightarrow \ y=4$

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

  $\Rightarrow$ molemmat rivit tuottavat **F**

  $\Rightarrow$ *ylempi* $\Leftrightarrow$ *alempi*

Miksi vaihe 2 on pätevä?

- $2x+3y=6 \wedge y=x+7 \;\Leftrightarrow\; 2x+3(x+7)=6 \wedge y=x+7$
- 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 puoli on tosi
  - $2x+3y=6 \wedge y=x+7 \;\Rightarrow\; 2x+3(x+7)=6$ on pätevä
- kaavassa, jossa ei esiinny $\forall$ eikä $\exists$, saa lausekkeen korvata yhtäsuurella, vaikka tämä yhtäsuuruus ei olisi yleispätevä vaan nojautuu voimassa oleviin oletuksiin
- jos $y=x+7$, niin vaiheen 2 väitteen vasen puoli sievenee samanlaiseksi kuin oikea puoli, kun ensimmäisen y:n tilalle sijoitetaan $x+7$
- muussa tapauksessa $y \neq x+7$, joten vaiheen 2 väitteen kumpikin puoli tuottaa **F**

$\Rightarrow$ vaiheen 2 väitteen molemmat puolet tuottavat aina saman totuusarvon

Myös osakaavan korvaamisessa yhtäpitävällä riittää yhtäpitävyyden osalta, että se pätee voimassa olevilla oletuksilla

Eräs hyvin paljon käytetty päättelysääntö voidaan johtaa helposti korvaamalla kaavassa lauseke yhtäsuurella

- jos $isompi\_lauseke$ ei sisällä määrittelemättömiä toimintoja, niin $isompi\_lauseke(lauseke1) = isompi\_lauseke(lauseke1)$
  - tämä on kaava
  - siinä ei esiinny $\forall$ eikä $\exists$, koska ne eivät voi esiintyä lausekkeissa
- jos $lauseke1 = lauseke2$, niin jälkimmäisen $lauseke1$:n korvaaminen tuottaa $isompi\_lauseke(lauseke1) = isompi\_lauseke(lauseke2)$

$\Rightarrow$ Lausekkeessa, jossa ei esiinny määrittelemättömiä toimintoja, saa osalausekkeen korvata yhtäsuurella, vaikka tämä yhtäsuuruus nojautuisi voimassa oleviin oletuksiin

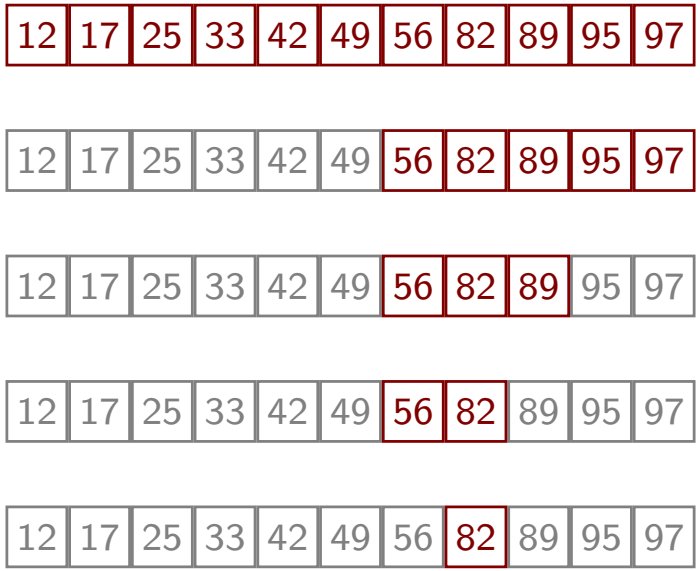
Esimerkki: yhtälöpari $2x+3y=6 \wedge y-x=7$ voidaan ratkaista myös seuraavasti

- ratkaistaan yhtälö $y-x=7 \Leftrightarrow y=x+7$
- oletuksella $y=x+7$ pätee $2x+3y=2x+3(x+7)=2x+3x+3\cdot 7=5x+21$, joten $2x+3y=6 \Leftrightarrow 5x+21=6 \Leftrightarrow x=-3$
- oletuksella $x=-3$ pätee $y=x+7=-3+7=4$
- niinpä $2x+3y=6 \wedge y-x=7 \Leftrightarrow x=-3 \wedge y=4$

2 Puolitushaun todistus

Puolitushaku (**binary search**)

- 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!



Onko helppoa?

- 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 ≤ n}
1    a := 1; y := n+1                          {1 ≤ a ≤ y ≤ n+1}
2    while a < y do                            {1 ≤ a < y ≤ n+1}
3        v := (a+y) div 2                      {1 ≤ a ≤ v < y ≤ n+1}
4        if A[v] < x then a := v+1             {1 ≤ a ≤ y ≤ n+1}
5        else y := v                           {1 ≤ a ≤ y ≤ n+1}
6        {1 ≤ a ≤ y ≤ n+1}
7    {1 ≤ a = y ≤ n+1}
```

- jokaisessa taulukossa on ainakin nolla alkiota, joten aina $n \geq 0$
  $\Rightarrow$ rivin 1 lopussa $1 \leq n+1$
- 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 $1 \leq a \leq y \leq n+1$
- rivin 1 jälkeen $a = 1 \leq n+1 = y$
  $\Rightarrow$ $1 \leq a \leq y \leq n+1$
  - $1 = a \leq y = n+1$ 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 $a < y$
  $\Rightarrow a < \frac{a+y}{2} < y$
  - rivin lopussa lisäksi $v = \frac{a+y}{2}$
  - 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 A, v ja x eivät muuttuneet rivillä 4, sen lopussa $a = v+1$, $a-1 = v$ 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

Takaako järjestyksessä oleminen, että on vain yksi luvatunlainen kohta?

- jos niitä on kaksi, niin olkoot ne a_1 ja a_2, missä $a_1 < a_2$

$\Rightarrow 1 \leq a_1 < a_2 \leq n+1 \Rightarrow 1 < a_2 \wedge a_1 < n+1 \Rightarrow A[a_1] \geq x \wedge A[a_2 - 1] < x$

$\Rightarrow A[a_1] > A[a_2 - 1] \Rightarrow$ taulukko on epäjärjestyksessä

3 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
  $\Rightarrow$ *Atomi* alkaa joko numerolla, kirjaimella tai `(`:lla
  - *Tekijä* alkaa `+`:lla, `-`:lla tai sillä millä *Atomi* alkaa

  $\Rightarrow$ *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

  $\Rightarrow$ *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 ;`
  $\Rightarrow$ 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 *Lauseke*, *Termi*, *Tekijä* ja *Atomi*
- ellei toisin sanota, koko määritelmän kieli on ensimmäisenä määritelty kieli
  - esimerkissä *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. *muuttuja* = *muuttuja* + *luku*;
  - miten tekstialkio muodostuu merkeistä, esim. *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 $\mathscr{L}(K)$ tarkoittamaan kieltä, jonka $K ::= \ldots$ tuottaa

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

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

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

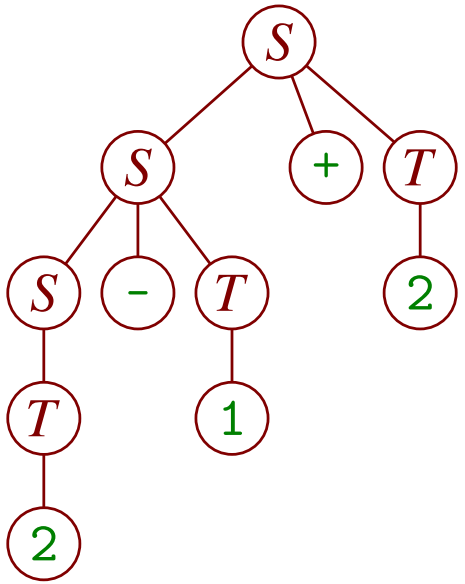
$\mathscr{L}(A)$	a		[aa]				[a[aa]]	[[aa]a]	[[aa][aa]]	...
$\mathscr{L}(B)$		aa		a[aa]	[aa]a	[aa][aa]				a[a[aa]] ...

- jos $A ::= aA$, niin $\mathscr{L}(A) = \emptyset$ 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 $T ::= 1 \mid 2 \mid \texttt{(}S\texttt{)}$, niin
  - $\mathscr{L}(S) = \{1, 2, 1\texttt{+}1, 1\texttt{+}2, 2\texttt{+}1, 2\texttt{+}2, 1\texttt{-}1, \ldots, \texttt{(}1\texttt{)}, \texttt{(}2\texttt{)}, \ldots, 2\texttt{-}1\texttt{+}2, \texttt{(}2\texttt{-(}1\texttt{+}2\texttt{))}, \ldots\}$
  - $\mathscr{L}(T) = \{1, 2, \texttt{(}1\texttt{)}, \texttt{(}2\texttt{)}, \texttt{(}1\texttt{+}1\texttt{)}, \texttt{(}1\texttt{+}2\texttt{)}, \ldots, \texttt{(}2\texttt{-}1\texttt{+}2\texttt{)}, \texttt{(}2\texttt{-(}1\texttt{+}2\texttt{))}, \ldots\}$

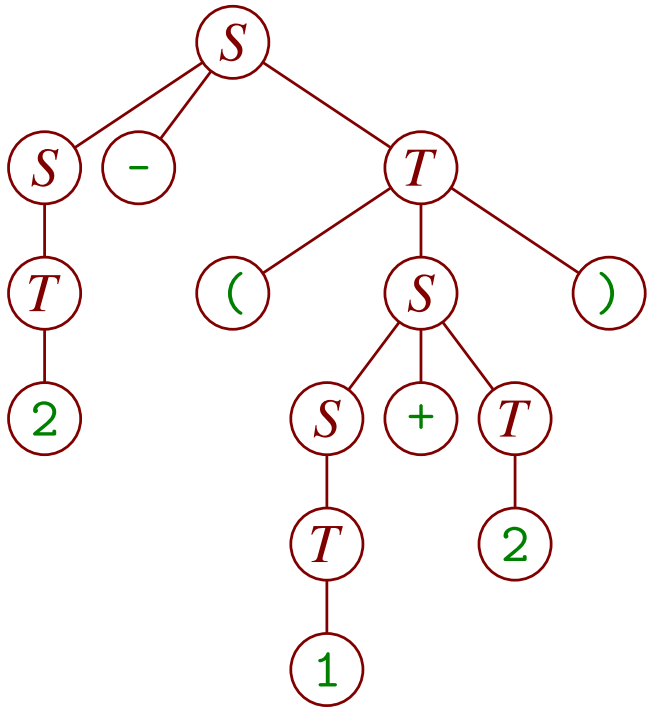
Kielen BNF-kielioppia voi käyttää

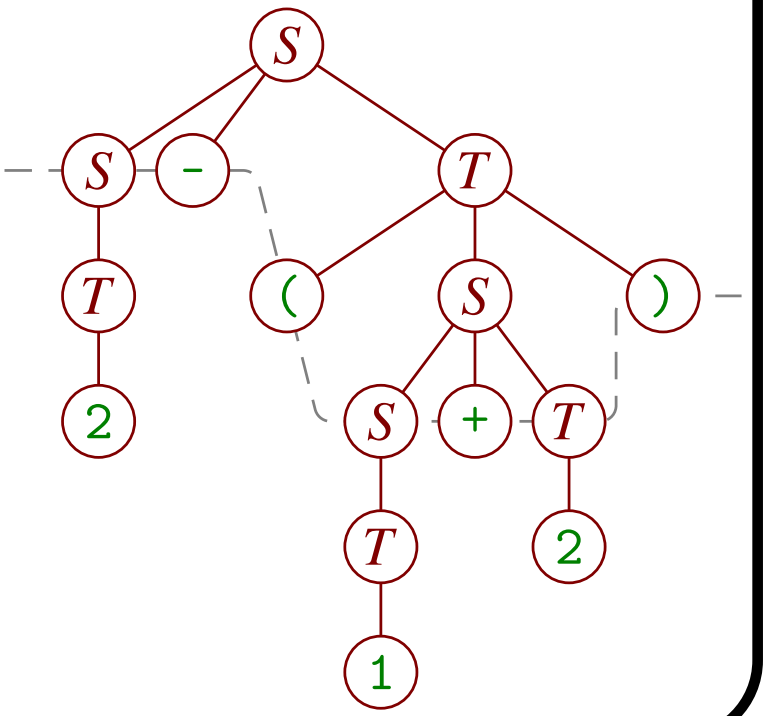
- alhaalta ylös
  - $2 \in \mathscr{L}(T) \;\Rightarrow\; 2 \in \mathscr{L}(S)$
  - $2 \in \mathscr{L}(S)$ ja $1 \in \mathscr{L}(T) \;\Rightarrow\; 2\texttt{-}1 \in \mathscr{L}(S)$
  - $2\texttt{-}1 \in \mathscr{L}(S)$ ja $2 \in \mathscr{L}(T) \;\Rightarrow\; 2\texttt{-}1\texttt{+}2 \in \mathscr{L}(S)$
- ylhäältä alas, vasemmanpuoleisin ensin
  - $S \rightsquigarrow S\texttt{+}T \rightsquigarrow S\texttt{-}T\texttt{+}T \rightsquigarrow T\texttt{-}T\texttt{+}T \rightsquigarrow$ $2\texttt{-}T\texttt{+}T \rightsquigarrow 2\texttt{-}1\texttt{+}T \rightsquigarrow 2\texttt{-}1\texttt{+}2$
- ylhäältä alas, oikeanpuoleisin ensin
  - $S \rightsquigarrow S\texttt{+}T \rightsquigarrow S\texttt{+}2 \rightsquigarrow S\texttt{-}T\texttt{+}2 \rightsquigarrow S\texttt{-}1\texttt{+}2 \rightsquigarrow T\texttt{-}1\texttt{+}2 \rightsquigarrow 2\texttt{-}1\texttt{+}2$
- 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

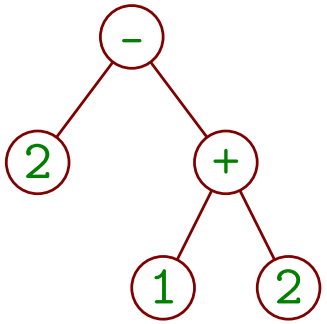
- alhaalta ylös
  - $2 \in \mathscr{L}(T) \Rightarrow 2 \in \mathscr{L}(S)$
  - $1 \in \mathscr{L}(T) \Rightarrow 1 \in \mathscr{L}(S)$
  - $2 \in \mathscr{L}(T)$
  - $1 \in \mathscr{L}(S)$ ja $2 \in \mathscr{L}(T) \Rightarrow 1+2 \in \mathscr{L}(S) \Rightarrow (1+2) \in \mathscr{L}(T)$
  - $2 \in \mathscr{L}(S)$ ja $(1+2) \in \mathscr{L}(T) \Rightarrow 2-(1+2) \in \mathscr{L}(S)$
- 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

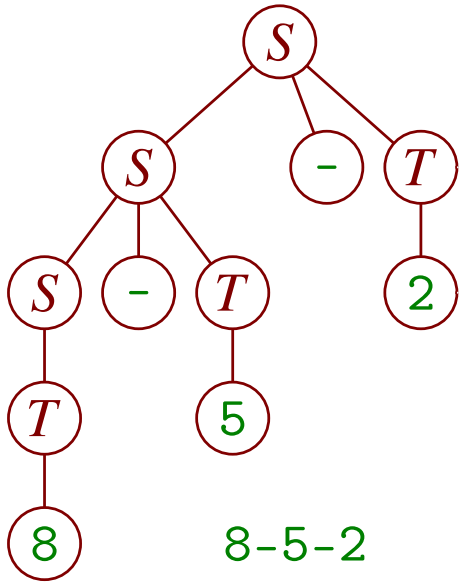
- lausekepuu esittää asian loogisen rakenteen
- jäsennyspuu esittää miten merkkijono saadaan kieliopista



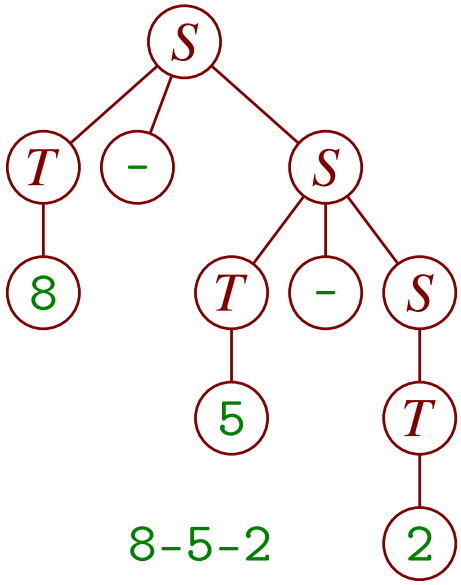
- jäsennyspuu voi sisältää apukielten nimiä, jäsennystä ohjaavia välimerkkejä yms.
  - esim. sulkeet, sijoituslauseen lopussa oleva `;`, sana `else`
- jäsennyspuulla ei välttämättä ole yhteyttä asian loogiseen rakenteeseen, koska voidaan käyttää "vääriä" jäsennyspuita tuottavaa kielioppia
- jotta voisi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

Eri kieliopit voivat tuottaa saman kielen eri jäsennyspuilla

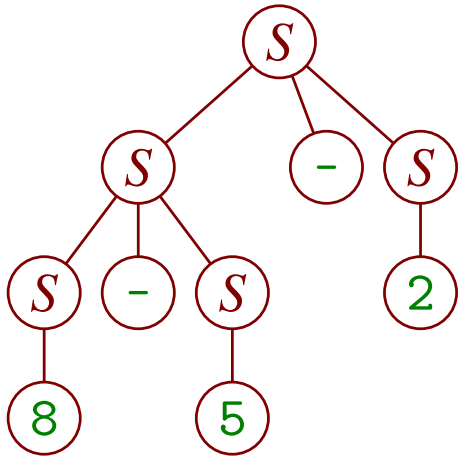
- $S ::= T \mid S\texttt{+}T \mid S\texttt{-}T$ ja $T ::= \texttt{0} \mid \cdots \mid \texttt{9} \mid \texttt{(}S\texttt{)}$
- $S ::= T \mid T\texttt{+}S \mid T\texttt{-}S$ ja $T ::= \texttt{0} \mid \cdots \mid \texttt{9} \mid \texttt{(}S\texttt{)}$
- $S ::= S\texttt{+}S \mid S\texttt{-}S \mid \texttt{0}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



8-5-2



8-5-2



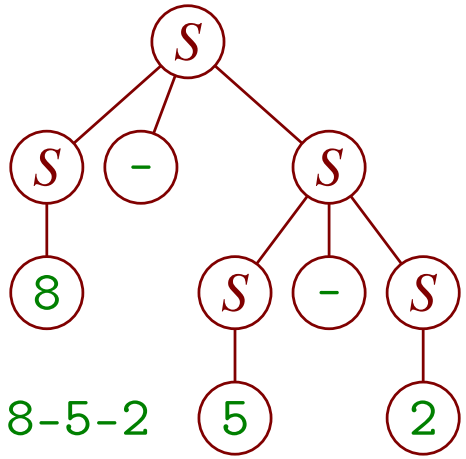
8-5-2

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

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

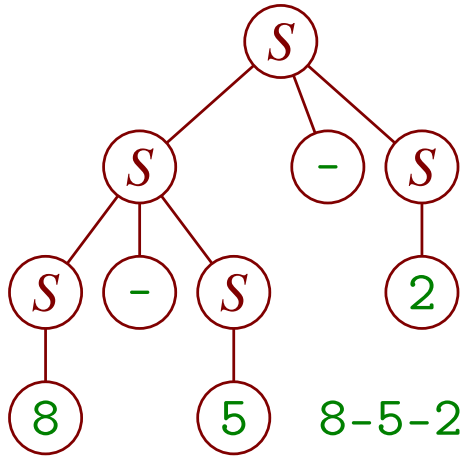
$\Rightarrow$ esim. $S ::= T \mid S{+}T \mid S{-}T$
eikä $S ::= T \mid T{+}S \mid T{-}S$

$\Rightarrow$ 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. $S ::= S{+}S \mid S{-}S \mid 0 \mid \cdots \mid 9 \mid (S)$
- 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 alkusulkeella `(` täytyy olla vastaava loppusulje `)`
  - 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

4 Pysähtymistesteriä ei ole

Yksinkertaisuuden vuoksi käytämme kuviteltua ohjelmointikieltä

Pysähtymistesteri

- olkoot $prog$ ja inp (äärellisiä!) merkkijonoja
- olkoon halts$(prog, inp)$ ohjelma, joka vastaa true jos ja vain jos merkkijonon $prog$ sisältö on seuraavanlainen ohjelma:
  - sillä on tasan yksi syöteparametri, ja se on tyypiltään merkkijono
  - se pysähtyy lopulta, jos se ajetaan syötteenä inp
- muutoin halts$(prog, inp)$ vastaa false
  - jos $prog$ ei ole ohjelma
  - jos $prog$ on ohjelma, mutta sillä on 0 tai > 1 syöteparametria
  - jos $prog$ on yksiparametrinen ohjelma, mutta parametri ei ole merkkijono
  - jos $prog$ on sellainen ohjelma, mutta ajo $prog(inp)$ ei pysähdy
- halts$(prog, inp)$ ei itse jää ikuiseen silmukkaan millään $prog$ ja inp
- halts$(prog, inp)$ on (yksiparametristen) ohjelmien **pysähtymistesteri**

Todistus 1: "Lyhyiden" ohjelmien tuottamien merkkijonojen tulostaminen

- tarkastellaan seuraavaa ohjelmaa

```
tulosta()
merkkijono := ""
while pituus(merkkijono) ≤ 1000···0 do        k nollaa
    if halts(merkkijono, "") then suorita merkkijono("")
    merkkijono := seuraava(merkkijono)
print "Valmis!"
```

- se kokeilee kaikki merkkijonot, joiden pituus on enintään 10^k
  - testaa kunkin, onko se tyhjällä syötteellä pysähtyvä ohjelma
  - ajaa ne, jotka ovat
  - niiden tulostukset tulevat peräkkäin, ja niiden perään "Valmis!"

$\Rightarrow$ sen oma tulostus on pitempi kuin minkään niistä

- se pysähtyy tyhjällä syötteellä
- sen oma pituus on vakio $+ k$

25764185	25764186	25764187	25764188
100000	1000000	10000000	100000000

$\Rightarrow$ tarpeeksi suurella k
  - se tulostaa merkkijonon, jota mikään enintään 10^k pitkä ei tulosta
  - se on enintään 10^k pitkä $\nearrow$

$\Rightarrow$ tyhjän syötteen pysähtymistesteriä ei ole olemassa

Todistus 2: Itseensä viittaava kutsu

- tarkastellaan seuraavaa ohjelmaa

  diag(inp)

  if halts(inp, inp) **then while** true **do** $i := -i$

- diag(inp) on yksiparametrinen ohjelma, jonka parametri on merkkijono
- mitä tekee kutsu diag(diag)?
  - alkajaisiksi se kutsuu halts(diag, diag)
  - halts($prog, inp$) on pysähtymistesteri

  $\Rightarrow$ halts(diag, diag) vastaa true, jos ja vain jos diag(diag) pysähtyy
  - jos halts(diag, diag) vastaa true, niin diag(diag) hyppää **while**-silmukkaan

  $\Rightarrow$ ei pysähdy
  - jos halts(diag, diag) vastaa false, niin diag(diag) ohittaa **while**-silmukkan

  $\Rightarrow$ pysähtyy

$\Rightarrow$ diag(diag) *tekee päinvastoin kuin* halts(diag, diag) *väittää sen tekevän*

$\Rightarrow$ halts(diag, diag) vastaa väärin

$\Rightarrow$ halts($prog, inp$) ei olekaan pysähtymistesteri $\nearrow$

- yksiparametristen ohjelmien pysähtymistesteriä ei ole olemassa
- todistuksen juoni lyhyesti: diag(diag) käyttää halts(diag, diag) ennustamaan oman tulevaisuutensa, ja vastauksen saatuaan tekee päinvastoin

Mitä tämä tulos tarkoittaa?

- tuskin kukaan odottaa, että tietokoneohjelmat löytäisivät jokaiseen kysymykseen oikean vastauksen
  - mikä on elämän tarkoitus? print 42
  - mikä on ensi viikon oikea lottorivi?
  - tulisiko minun juoda teetä vai kahvia?
- kysymys "pysähtyykö $prog$ jos se käynnistetään syötteenään inp" tuntuisi kuitenkin kuuluvan tietokoneohjelmien pätevyysalueelle
  - oikea vastaus on olemassa: $prog(inp)$ joko pysähtyy lopulta tai ei pysähdy
  - $prog$ ja inp ovat tietokoneen tutkittavissa olevassa muodossa
- **pysähtymisfunktio** $Ohjelmat \times Syötteet \to \{\mathsf{false}, \mathsf{true}\}$ on olemassa (on hyvin määritelty), mutta ei ole olemassa ohjelmaa, joka laskee sen
- syötteellinen kyllä–ei-kysymys on **ratkeamaton (undecidable)**, jos ja vain jos mikään tietokoneohjelma ei vastaa siihen oikein jokaisella syötteellä

$\Rightarrow$ "pysähtyykö annettu ohjelma annetulla syötteellä" on ratkeamaton
  - "annettu ohjelma" ja "annettu syöte" ovat kysymyksen syötteet

Epätäydellisiä pysähtymistestereitä on olemassa

- esim. simuloi $prog(inp)$ enintään 10^{10^n} askelta, missä $n = |prog|$
  - jos $prog(inp)$ pysähtyy siihen mennessä, vastaa "kyllä"
  - jos $prog(inp)$ jää tunnistettuun ikuiseen silmukkaan, vastaa "ei"
  - muutoin vastaa "en tiedä"
- jokainen epätäydellinen pysähtymistesteri epäonnistuu joillain $prog$ ja inp

$\Rightarrow$ ratkeamattomuus ei välttämättä tarkoita epäonnistumista *kaikilla* syötteillä
  - määritelmä tarkoittaa vain, että epäonnistutaan *ainakin yhdellä* syötteellä
  - jatkossa nähdään, että siitä seuraa epäonnistuminen äärettömän monella

Nimen diag syy

- em. todistus on ns. **diagonalisointitodistus**
- nimitys on peräisin G. Cantorin todistuksesta, että reaalilukuja ei voi luetella päättymättömällä luettelolla
  - jos voisi, myös välin $0 \leq x < 1$ reaaliluvut voisi luetella
  - päättymättömiä 9-jonoja ei ole pakko käyttää ($0{,}0999\ldots = 0{,}1000\ldots$)
  - jos otetaan mikä tahansa päättymätön luettelo ilman päättymättömiä 9-jonoja, niin kulkemalla sen lävistäjää eli diagonaalia pitkin ja korvaamalla $0 \mapsto 1$, muut numerot $\mapsto 0$ saadaan luku, joka ei ole luettelossa

    0,32156271...
    0,56438342...
    0,01010101...
    ...

    0,001...

- pysähtymistesterin tapauksessa
  - ulottuvuudet ovat *prog* ja *inp*
  - matriisin alkiot ovat false tai true

Epätäydellisen pysähtymistesterin täydentäminen

- vastatkoon $\text{halts}_1(prog, inp)$ kullekin $prog$ ja inp "kyllä", "ei" tai "en tiedä"
  - jos vastaus on "kyllä" tai "ei", sen on oltava oikein
- $\text{diag}_1(inp)$ on esim. **if** $\text{halts}_1(inp, inp) =$ "kyllä" **then while** true **do** $i := -i$
  - tapauksesta "en tiedä" voidaan tehdä pysähtyvä tai pysähtymätön
- voidaan tehdä parempi testeri, joka vastaa oikein kun $prog = inp = \text{diag}_1$ ja muutoin vastaa samoin kuin $\text{halts}_1(prog, inp)$

  $\underline{\text{halts}_2(prog, inp)}$

  if $prog = \text{diag}_1 \wedge inp = \text{diag}_1$ **then return** "kyllä"
  else return $\text{halts}_1(prog, inp)$

  - $\text{halts}_1(\text{diag}_1, \text{diag}_1)$ ei voi vastata "kyllä" eikä "ei", koska $\text{diag}_1(\text{diag}_1)$ tekisi päinvastoin

  $\Rightarrow$ se vastaa "en tiedä"
  $\Rightarrow$ $\text{diag}_1(\text{diag}_1)$ pysähtyy
  $\Rightarrow$ $\text{halts}_2(\text{diag}_1, \text{diag}_1)$:n vastaus "kyllä" on oikein
- mutta $\text{halts}_2(\text{diag}_2, \text{diag}_2)$ vastaa "en tiedä", missä $\text{diag}_2(inp)$ on **if** $\text{halts}_2(inp, inp) =$ "kyllä" **then while** true **do** $i := -i$
- myös $\text{halts}_1(\text{diag}_2, \text{diag}_2)$ vastaa "en tiedä", koska halts_2 vastaa samoin kuin halts_1, kun $prog \neq \text{diag}_1$ tai $inp \neq \text{diag}_1$
- samalla tavalla voidaan tehdä halts_3, diag_3, halts_4, diag_4, ...

$\Rightarrow$ saadaan päättymätön jono toinen toistaan parempia epätäydellisiä pysähtymistestereitä halts_i ja ohjelmia diag_j

- diag_j on tehty siten, että halts_j vastaa sille "en tiedä"
- halts_{j+1} on tehty vastaamaan "kyllä" diag_j:lle
- halts_{i+1} vastaa kullekin syötteelle "kyllä" tai samoin kuin halts_i

$\Rightarrow$ $\text{halts}_i(\text{diag}_j, \text{diag}_j)$ vastaa "kyllä", kun $j < i$ ja "en tiedä", kun $j \geq i$

$\Rightarrow$ jokaisella epätäydellisellä pysähtymistesterillä on äärettömasti syötteitä, joille se vastaa "en tiedä"

Mahdollisten erilaisten syötteiden määrän merkitys

- rajoitutaan yksinkertaisuuden vuoksi kyllä–ei-kysymyksiin
- jos mahdollisia erilaisia syötteitä on vain äärellinen määrä, niin oikeat vastaukset voi taulukoida

$\Rightarrow$ on olemassa ohjelma, joka tuottaa jokaiselle syötteelle oikean vastauksen

stupid_halts($prog, inp$)

if $prog = \ldots \wedge inp = \ldots$ **then return** "ei"
else if $prog = \ldots \wedge inp = \ldots$ **then return** "kyllä"
else if $prog = \ldots \wedge inp = \ldots$ **then return** "ei"
...
else return "en tiedä"

$\Rightarrow$ tehtävä voi olla ratkeamaton vain, jos erilaisia syötteitä on äärettömästi

- jos ohjelma epäonnistuu vain äärellisen monella syötteellä, siihen voidaan lisätä vastauksen katsominen taulukosta niillä syötteillä
  - saadaan ohjelma, joka ratkaisee tehtävän kaikilla syötteillä

$\Rightarrow$ ratkeamattomuudesta seuraa epäonnistuminen *äärettömän monella* syötteellä

- jos syötettä ei ole ja kysymys on hyvin määritelty, niin tehtävä on ratkeava
  - esim. Goldbachin hypoteesi: onko totta, että ei ole olemassa parillista lukua > 2, jota ei voi esittää kahden alkuluvun summana
  - sen ratkaisee jompikumpi seuraavista kahdesta ohjelmasta:

    print ”kyllä” print ”ei”
  - ongelmana on, että ei tiedetä, kumpi!
  - helpon ohjelman todistaminen oikeaksi voi olla vaikeaa
- ”ohjelmaa ei ole” on eri asia kuin ”ei tiedetä, mikä ohjelma on oikea”

Laskeeko ohjelma vastauksen itse vai onko ohjelmoija antanut valmiin vastauksen?

- stupid_halts on selvästi saanut vastaukset valmiina
- Bellman–Ford selvittää ”itse”, onko graafissa negatiivinen silmukka
- ohjelman toiminta voi olla yhdistelmä laskentaa ja taulukosta katsomista
- jos mahdollisia erilaisia syötteitä on äärettömästi,
  ohjelman on laskettava suurin osa vastauksista ”itse”
  - vain äärellinen määrä vastauksia voi olla annettu valmiiksi

$\Rightarrow$ siksikin on teorian kannalta tärkeää, että erilaisia syötteitä on äärettömästi

Huomautus äärellisyydestä ja äärettömyydestä

- ratkeamattomuuden teoriassa ohjelma ja syöte ovat *äärellisiä* merkkijonoja
  - ohjelmoija voi kirjoittaa vain äärellisen määrän koodia
  - jollei syöte ole äärellinen, ohjelma ei koskaan saa luettua sitä kokonaan
- erilaisia äärellisiä merkkijonoja on äärettömästi
  - esim. a, aa, aaa, ...
  - myös erilaisia ohjelmia on äärettömästi
- muualla tietojenkäsittelyteoriassa käsitellään äärettömiäkin merkkijonoja
  - esim. $\texttt{aaa}\cdots$, $\texttt{ababab}\cdots$, $\texttt{3,14159}\cdots$
  - funktio $\mathbb{N} \mapsto \Sigma$
- ratkeamattomuuden teoriassa ohjelma ja syöte eivät voi olla äärettömiä, mutta muuten voivat olla miten suuria tahansa
  - äärellisiä mutta **rajattomia** (**unbounded**)
- todellisessa maailmassa ohjelma ja syöte eivät voi olla rajattomiakaan
  $\Rightarrow$ saattaa tuntua, että ratkeamattomuuden teoriassakin pitäisi olettaa raja
  - olisiko $1\,000\,000$ tai $10^{1\,000\,000}$ sopiva raja ohjelman ja syötteen koolle?
  - nykyisten tietokoneiden muistin määrä on hyvin suuri
  - rajattomuus on kelvollinen likiarvo hyvin suurelle
- ratkeamattomuuden teoriassa oletetaan myös muisti ja aika rajattomiksi
  - kunakin ajanhetkenä vain äärellinen määrä muistia on käytössä
  - laskennan edetessä tämä määrä voi kasvaa rajatta

5 Reaalilukujen lakeja

Yhteenlaskun, nollan, vastaluvun ja vähennyslaskun lait

$$
\begin{aligned}
a+b &= b+a && \textbf{vaihdannaisuus (commutativity)}\\
(a+b)+c &= a+(b+c) && \textbf{liitännäisyys (associativity)}\\
a+0 &= a && \textbf{nollan (zero) perusominaisuus}\\
a+(-a) &= 0 && \textbf{vastaluvun (additive inverse) perusominaisuus}\\
a-b &= a+(-b) && \textbf{vähennyslaskun (subtraction) määritelmä}
\end{aligned}
$$

- kaikki reaalilukujen yhteen- ja vähennyslaskun ominaisuudet voidaan johtaa näistä
- lait pätevät jokaisella reaaliluvulla a, b ja c
- etumerkki-"$-$" on eri laskutoimitus kuin vähennyslasku-"$-$"
  - toisella on argumentti vain perässä: $-b$
  - toisella sekä edessä että perässä: $a-b$
- 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 $x=-a$:

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

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

- tosin tuttuihinkin asioihin voidaan tarvita pitkiä päättelyitä!
- esimerkki: todistamme $a+b+c=c+b+a$
  - "$+$" sitoo vasemmalle

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

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

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

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

- 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

	ab	$=$	ba	**vaihdannaisuus (commutativity)**
	$(ab)c$	$=$	$a(bc)$	**liitännäisyys (associativity)**
	1	$\neq$	0	ykkönen ei ole nolla
	$a \cdot 1$	$=$	a	**ykkösen (one) perusominaisuus**
jos $a \neq 0$, niin	$a \cdot \frac{1}{a}$	$=$	1	**käänteisluvun (multiplicative inverse) per.**
jos $b \neq 0$, niin	$\frac{a}{b}$	$=$	$a \cdot \frac{1}{b}$	**jakolaskun (division) määritelmä**
	$a(b+c)$	$=$	$ab+ac$	**osittelulaki (distributivity)**

- kaikki reaalilukujen kerto- ja jakolaskun ominaisuudet voidaan johtaa näistä
- lait pätevät jokaisella reaaliluvulla a, b ja c
- 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 **T** vain yhtäaikaa
  - 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:

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

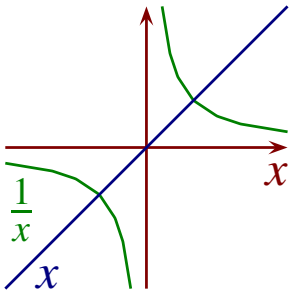
Siis kun epäyhtälön puolet kerrotaan

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

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

- tapaus $x < 0$: $x < 0 \wedge \frac{1}{x} \leq x \Leftrightarrow x < 0 \wedge 1 \geq x^2 \Leftrightarrow -1 \leq x < 0$
  - $\Rightarrow$: kerrotaan x:llä, $x^2 \leq 1 \Rightarrow -1 \leq x \leq 1$
  - $\Leftarrow$: kerrotaan $\frac{1}{x}$:llä, $x^2 \leq 1 \Leftarrow -1 \leq x \leq 1$
- 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

Loppu