# Ohjelmistotekniikan matemaattiset menetelmät tentin 16.12.2003 kysymykset, vastaukset, arvosteluperusteet ja vastausohjeita

Antti Valmari
TTY / Ohj

20. tammikuuta 2004

Käyn tässä tekstissä perusteellisesti läpi opintojakson 8100500 Ohjelmistotekniikan matemaattiset menetelmät 16.12.2003 pidetyn tentin kysymykset, vastaukset ja arvosteluperusteet sekä pohdin, miten vastaukset voi löytää. Myös kerron joidenkin tehtävien taustasta, sekä annan yleisiä vastausvinkkejä.

Vastauksen voi usein löytää monella tavalla, ja samalle tehtävälle saattaa olla useita oikeita vastauksia. Tässä tekstissä annan pääsääntöisesti vain yhden. Vastauksen löytämistavoissa suosin niitä, jotka edellyttävät mahdollisimman vähän aiempien kysymysten vastausten osaamista. Näin siksi, että opiskelijan asema tentissä on sitä parempi, mitä vähemmän kyvyttömyys vastata johonkin tehtävään haittaa myöhempiin tehtäviin vastaamista.

Arvosteluperusteet koskevat vain kyseistä tenttiä. Täysin moitteettomat vastaukset tuottavat tietysti aina täydet pisteet, mutta hieman virheellisten vastausten pisteytys voi vaihdella tenttikerrasta toiseen, samoin vastausten, jotka ovat selkeästi väärin mutta joissa on silti jotakin oikeansuuntaista. Jos esimerkiksi samantapainen huolimattomuusvirhe toistuu useassa tehtävässä, ei välttämättä ole mielekästä vähentää siitä joka tehtävän kohdalla yhtä paljon pisteitä. Jos olen erityisesti varoittanut jostakin virhetyypistä juuri ennen tenttiä, on todennäköistä, että vähennän sellaisista virheistä siinä tentissä enemmän pisteitä kuin aiemmin.

Tehtävän pisteet jakautuvat tasan alakohtien kesken. Tehtävissä 9 ja 10 täysin oikea vastaus oli 6 pisteen arvoinen ja tehtävän 1 alakohdissa 4 pisteen arvoinen. Muissa tehtävissä täysin oikea vastaus oli 3 pisteen arvoinen. Vajaita mutta nollaa suurempia pistemääriä jaoin vain niissä tapauksissa, joissa vastaus jakaantuu luonnostaan osiin jotka voi pisteyttää erikseen, tai joissa vastaus oli selvästi periaatteeltaan oikein mutta sisälsi olennaisen huolimattomuusvirheen. Tästä syystä vajaat nollaa suuremmat pistemäärät ovat melko harvinaisia.

Tenttipaperin alussa oli seuraava kohta:

> Kirjoita lukukelpoista! Vastaukselta ei vaadita enempää kuin mihin tila riittää. Sivuja on kaksi.
> Opiskelijanro _____ nimi _____
> Sähköposti _____ allekirjoitus _____

**Kirjoita lukukelpoista:** Olen yleensä saanut kaikista eteeni tulleista vastauspapereista selvää, mutta joskus joudun tavaamaan aika lailla. On tilanteita, joissa hyvinkin pieni epäselvyys kirjoituksessa voi ratkaista, onko vastaus oikea. Esimerkiksi käsin kirjoitetut kirjaimet *n* ja *h* muistuttavat helposti toisiaan. Sen vuoksi vältän symbolien *n* ja *h* käyttämistä muuttujina samassa tehtävässä, ja muutenkin koetan laatia kysymykset siten, että epäselvyyksien mahdollisuus olisi mahdollisimman pieni. Aina en siinä kuitenkaan onnistu.

Jos olen varma siitä, mitä opiskelija on tarkoittanut, tulkitsen epäselvän kohdan sen mukaisesti. Muussa tapauksessa valitsen yleensä opiskelijalle epäedullisemman vaihtoehdon. Näin teen siksi, että mielestäni on kirjoittajan vastuulla, että hänen sanomansa on yksikäsitteinen. Se on myös tämän opintojakson hengen mukaista: tavoitteenahan on vähentää monikäsitteisyyden aiheuttamia ongelmia ohjelmistotyössä.

Osa vastauspapereista on hyvin sotkuisia. Jos on liian vaikeaa kirjoittaa vastaus suoraan ruutuun, se kannattaa suunnitella suttupaperilla ja kopioida sieltä vastauspaperiin kun se on valmis.

**Vastaukselta ei vaadita enempää kuin mihin tila riittää:** Haluan vastaukset kysymyspaperille eikä erilliselle paperille siksi, että silloin eri vastaajien vastaukset ovat aina samassa kohdassa paperia, joten voin etsiskelemättä käydä saman tehtävän kaikki vastaukset läpi ennen seuraavaan tehtävään siirtymistä. Tämä vähentää arvosteluvirheiden vaaraa ja lisää todennäköisyyttä sille, että annan samanlaisille osittain oikeille vastauksille saman pistemäärän. Lisäksi tilan rajallisuus estää minua liiaksi paisuttamasta kysymysten määrää.

Vastaukselle varattu tila ei riipu pelkästään mallivastaukseni tarvitsemasta tilasta, vaan siihen vaikuttaa paljon se, miten saan ladottua tekstin ja vastausruudut sivulle järkevästi. Vastaustilan kokoon vaikuttaa hieman myös pyrkimykseni olla antamatta sen välityksellä vihjettä siitä, mihin kohtaan tarvitaan pitkä ja mihin lyhyt vastaus.

Mallivastaukseni mahtuu annettuun tilaan siististi ja sullomatta. On luonnollista, että opiskelija ei välttämättä saa asiaa muotoiltua yhtä tiiviisti, varsinkin kun mitään tärkeäksi koettua ei tietenkään uskalla jättää pois. Vastaus saa siis venyä annetun tilan yli.

Vastaus ei kuitenkaan saa venyä miten paljon tahansa. Asian osaamiseen kuuluu myös kyky erottaa olennainen epäolennaisesta. Kohtuuton venyminen ei tähän mennessä ole ollut ongelma, joten en kirjoita tästä asiasta tällä kertaa tämän enempää.

Aiempi vihje vastauksen suunnittelemisesta suttupaperilla pätee tässäkin.

**Opiskelijanumero:** Opiskelijanumero kannattaa kirjoittaa selvästi. Tenttipaperin tapauksessa opiskelijanumeroa voi verrata nimeen, minkä ansiosta huomaan virheelliset opiskelijanumerot ennen tuloslistan antamista eteenpäin. Virheellisistä numeroista aiheutuu kuitenkin ylimääräistä työtä.

Silloin kun annatte opiskelijanumeronne ilman että samassa yhteydessä annatte nimenne, on vaarana, että suorituksenne kirjaantuu väärälle opiskelijalle. Näin voi olla esimerkiksi laskuharjoitusten yhteydessä. Kyllä nämä yleensä saadaan selvitettyä jälkikäteen, mutta siitä aiheutuu kaikille osapuolille huolta ja vaivaa.

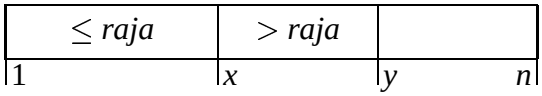
**Nimi:** Nimi tarvitaan varmistamaan yhdessä opiskelijanumeron kanssa, että suoritus kirjataan oikealle opiskelijalle.

**Sähköposti:** Tämä kohta on yleensä tarpeeton, mutta on mukana siltä varalta, että ilmenee erityinen syy kysyä jotakin. Tässäkin kohti kannattaa kirjoittaa selvästi, sillä yhdenkin merkin virhe tekee sähköpostiosoitteesta toimimattoman.

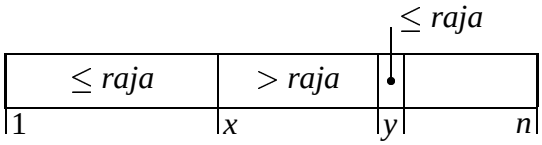
**1.** (12p) Täydennä seuraava ohjelma mielekkäillä (vrt. annetut esimerkit) tilapredikaateilla.

{ $n \geq 1$ }
$x := 1$; $raja := A[n]$
**for** $y := 1$ **to** $n$ **do**
    { $\forall i; 1 \leq i < x : A[i] \leq raja \ \wedge \ \forall i; x \leq i < y : A[i] > raja$ }
    **if** $A[y] \leq raja$ **then** $apu := A[x]$; $A[x] := A[y]$; $A[y] := apu$;
        {_____}
        $x := x + 1$
        {_____}
    **endif**
**endfor**
{_____}

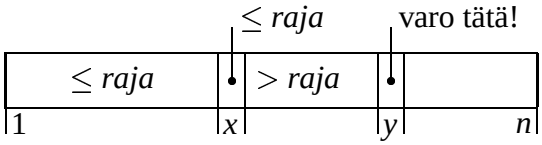
Tässä tehtävässä pääsee parhaiten eteenpäin, kun tulkitsee **for**-silmukan sisällä olevan valmiina annetun kaavan. Se sanoo, että taulukko koostuu "pienten" alkioiden alkuosasta ja "suurten" alkioiden loppuosasta. Pieniä ovat ne, jotka ovat enintään *raja*, ja muut ovat suuria. Alkuosa ulottuu paikasta 1 paikkaan $x-1$ ja loppuosa paikasta $x$ paikkaan $y-1$.




Ensimmäiselle vastausviivalle tultaessa on tapahtunut seuraava: $A[y]$ on todettu pieneksi ...




... ja sen jälkeen vaihdettu $A[x]$:n kanssa.




Tämän vaikutus annettuun kaavaan kannattaa analysoida tapauksittain, missä tapaukset edustavat taulukon eri paikkoja.

1. Muissa paikoissa kuin $x$ ja $y$ ei ole tapahtunut muutoksia, joten niitä koskevat väittämät ovat yhä voimassa:

$$\forall i\,;\, 1 \leq i < x : A[i] \leq raja \wedge \forall i\,;\, x < i < y : A[i] > raja \ .$$

2. Paikassa $x$ on pieni alkio (nimittäin entinen $A[y]$, joka havaittiin **if**-testissä pieneksi):

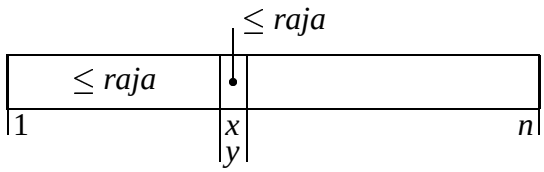
$$A[x] \leq raja \ .$$

3. Paikassa $y$ on alkuperäinen $A[x]$. Alkuperäinen kaava takaa, että jos $x < y$, niin $x$ sijaitsee suurten alkioiden alueella, joten vaihdon jälkeen $A[y] > raja$. Ei kuitenkaan ole välittömästi selvää, että $x < y$ pätee.

   Pieni mietiskely paljastaa, että tässä kohdassa ohjelmaa aina varmasti $x \leq y$. Nimittäin, silmukan ensimmäisen kierroksen alussa molemmilla on arvo 1. Kierroksen aikana $y$ kasvaa varmasti yhdellä, ja $x$ joko kasvaa yhdellä tai säilyy muuttumattomana sen mukaan suoritetaanko **then**-haara. Siis $x$ joko seuraa $y$:tä tai jää siitä jälkeen — edelle se ei voi päästä. (Vielä tässä vaiheessa ei välttämättä ole selvää, onko **then**-haaran suorittaminen koskaan mahdollista, ja onko sen suorittamatta jättäminen koskaan mahdollista. Sillä ei kuitenkaan ole merkitystä. Riittää, että huomataan, että ainakaan mitään muita mahdollisuuksia ei ole kuin että $x$ kasvaa saman verran tai vähemmän kuin $y$.)

   Jäljellä on siis tapaus $x = y$. Se on niin rajoitettu, että se on helppo miettiä erikseen. Silloin $A[x]$ ja $A[y]$ ovat sama alkio, ja alkioiden vaihto ei vaikuta mitään. Tiedetään siis, että $A[x] = A[y] \leq raja$.

   Huomaamme, että edeltävä kuvasarja on kohdan $y$ osalta harhaanjohtava: se houkuttelee uskomaan, että $A[y] > raja$. Kuten edellä näimme, niin onkin silloin kun $x < y$, mutta ei silloin kuin $x = y$. Tapauksessa $x = y$ kuvan pitää näyttää tältä:



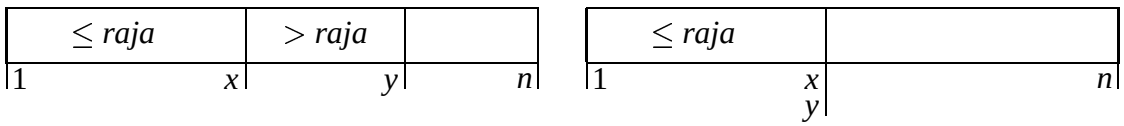

   Niin hyviä kuin kuvat ovatkin, niiden kanssa kannattaa olla varovainen, ettei vahingossa jätä erikoistapauksia huomiotta.

Nyt pitää koota tiedot yhteen. Kohdan 1 ensimmäinen osakaava on helppo yhdistää kohdan 2 tuloksen kanssa pienten alkioiden aluetta koskevaksi väittämäksi $\forall i\,;\, 1 \leq i \leq x : A[i] \leq raja$.

Suurten alkioiden aluetta koskevalle osakaavalle $\forall i\,;\, x < i < y : A[i] > raja$ pitäisi varmaan myös tehdä jotakin. Kohdan 1 nojalla se on voimassa sellaisenaan, mutta kannattaa miettiä, voidaanko myös alkiosta $A[y]$ sanoa jotakin. Kohdan 3 ensimmäinen osatapaus $x < y$ sallisi kirjoittaa $\forall i\,;\, x < i \leq y : A[i] > raja$. Kohdan 3 toinen osatapaus $x = y$ saattaa näyttää olevan sen kanssa ristiriidassa koska se väittää että $A[y] \leq raja$, mutta ei hätää: kun $x = y$, on väli $x < i \leq y$ tyhjä (eli isojen alkioiden osataulukko on tyhjä), joten $\forall i\,;\, x < i \leq y : A[i] > raja$ ei itse asiassa väitä yhtään mitään. Se ei väitä että $A[y] > raja$, koska $y$ ei täytä ehtoa $x < y \leq y$. Koska se ei väitä mitään, se on totta.

Kaava $\forall i\,;\, x < i \leq y : A[i] > raja$ on siis totta molemmissa tapauksissa. Muita tapauksia ei enää ole, joten voimme kirjoittaa vastaukseksi

$$\forall i\,;\, 1 \leq i \leq x : A[i] \leq raja \wedge \forall i\,;\, x < i \leq y : A[i] > raja \ .$$



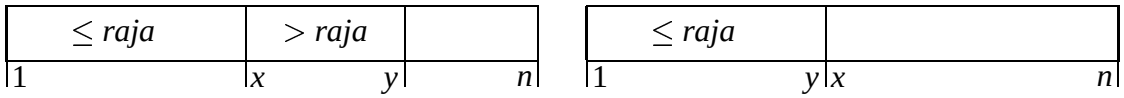
* * *

Tästä toiselle vastausviivalle edettäessä $x$ kasvaa yhdellä. Se, mikä oli ennen $x$:n arvo saadaan nyt aikaiseksi kirjoittamalla $x-1$. Tekemällä tämä edellisen kohdan vastaukselle saadaan

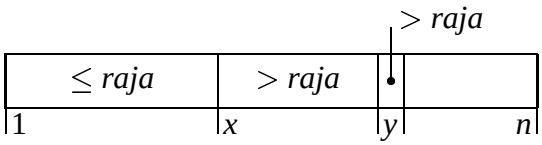
$$\forall i\,;\, 1 \le i \le x-1 : A[i] \le raja \wedge \forall i\,;\, x-1 < i \le y : A[i] > raja \ .$$

Koska taulukoiden indeksit ovat kokonaislukuja, $i \le x-1$ tarkoittaa samaa kuin $i < x$ ja $x-1 < i$ tarkoittaa samaa kuin $x \le i$. Niinpä edellinen kaava sievenee muotoon

$$\forall i\,;\, 1 \le i < x : A[i] \le raja \wedge \forall i\,;\, x \le i \le y : A[i] > raja \ .$$

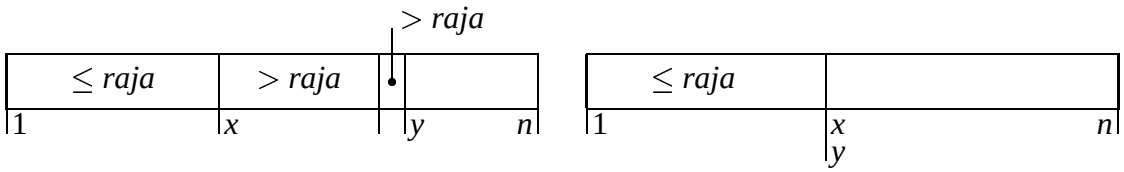



Tässä vaiheessa tarjoutuu mahdollisuus tarkastaa tähänastinen päättely. Seuraavaksi suoritus tulee ulos **if**-lauseesta. **If**-lauseen läpi voidaan mahdollisesti mennä myös niin, että **then**-haaraan ei mennä. Siinä tapauksessa taulukon sisältö ei muutu mitenkään, mutta saadaan lisätieto, että ehto $A[y] \le raja$ ei pätenyt. Siis $A[y] > raja$.

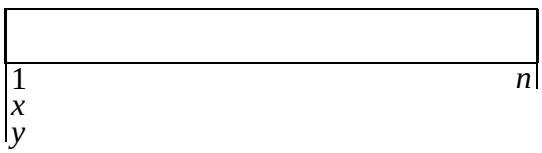



Kun sen lisää alkuperäiseen kaavaan saa $\forall i\,;\, 1 \le i < x : A[i] \le raja \wedge \forall i\,;\, x \le i \le y : A[i] > raja$, siis edellisen kohdan vastauksen! Tämä on hyvä merkki: mahtaneeko olla pelkkää sattumaa, että samaan kaavaan päädytään molempia reittejä?

Seuraavaksi ohjelma siirtyy **for**-silmukan alkuun ja samassa yhteydessä kasvattaa muuttujan $y$ arvoa yhdellä. Koska huomasimme, että on helppoa laskea miten sellainen kasvatus vaikuttaa kaavaan, teemme niin. Tulokseksi tulee $\forall i\,;\, 1 \le i < x : A[i] \le raja \wedge \forall i\,;\, x \le i < y : A[i] > raja$, siis alunperin annettu kaava. Tämä sopii yhteen sen kanssa, että suoritus jatkaa seuraavaksi kyseisen kaavan kohtaan ohjelmaa, paitsi viimeisen kierroksen tapauksessa.




Päättelyn voi vielä täydentää tarkastamalla, onko annettu kaava voimassa silloin kun silmukkaan tullaan ensimmäisen kerran. Sillä hetkellä $x = y = 1$, joten kumpikin väli on tyhjä, joten kaava ei väitä mitään, ja on siis tosi. (Kun kaava on tosi jostakin hyvin yksinkertaisesta syystä, kuten siksi, että se ei kyseisessä tilanteessa väitä mitään, on tapana sanoa, että kaava on *triviaalisti* tosi.)



Tällainen tarkastus menee jossain määrin kauas tentin kysymyksistä, ja on joka tapauksessa tentin laatijan vastuulla, että valmiina annettu kaava on oikein. Tarkastuksen avulla voi kuitenkin testata omaa ymmärtämystään tehtävän tilanteesta.

* * *

Ohjelman loppuun joudutaan, kun oltaisiin valmiita jatkamaan seuraavalle kierrokselle, mutta kierrosten määrä on tullut täyteen. On siis helppo arvata, että jokin alkuperäisen kaavan kaltainen kaava pätee, mutta $y$:n arvolla, joka olisi seuraavana vuorossa. Tämä arvo on tietenkin $n+1$. Sijoittamalla se $y$:n paikalle saamme $\forall i\,;\, 1 \le i < x : A[i] \le raja \wedge \forall i\,;\, x \le i < n+1 : A[i] > raja$ eli

$$\forall i\,;\, 1 \le i < x : A[i] \le raja \wedge \forall i\,;\, x \le i \le n : A[i] > raja \ .$$

Muuttujan $y$ mainitseminen loppukaavassa olisi virhe, koska $y$:tä ei välttämättä ole enää edes olemassa tässä kohdassa ohjelmaa.

Tällainen päättely on sikäli virhealtis, että alkuperäisen kaavan ei tarvitse koskaan päteä niissä suorituksissa, jotka eivät koskaan kulje kaavan kautta. Nyt tätä ongelmaa ei kuitenkaan ole, sillä ohjelman alussa luvataan, että $n \ge 1$, joten **for**-silmukka suoritetaan ainakin kerran.

* * *

Pisteytin tämän siten, että $\wedge$-merkin vasemmanpuoleinen osa kutakin kaavaa tuotti korkeintaan 2 pistettä, ja samoin oikea puoli. Kummassakin puolikkaassa paikkansa pitämätön kaava tuotti 0 pistettä. Puolikkaasta sai yhden pisteen, jos esitti edellä olevan päättelyn näkökulmasta mielekkään, paikkansa pitävän kaavan, joka ilmaisee vähemmän kuin edellä ollut vastaava kaava.

**2.** (3p) Mikä on suurin arvo, joka muuttujalla $x$ voi olla tehtävän 1 ohjelman lopussa? _____

Vaikka ei ymmärtäisi yhtään mitään edellisestä tehtävästä, voi havaita, että $x$:n arvo saadaan mahdollisimman suureksi järjestämällä niin, että **then**-haaraan mennään mahdollisimman usein. Pitää siis saada $A[y] \le raja$ voimaan mahdollisimman monella $y$:n arvolla. Muuttujan *raja* arvo on peräisin alkuperäisen taulukon kohdasta $n$. Kannattaa siis valita taulukko, jonka jokainen alkio on enintään yhtäsuuri kuin alkuperäinen $A[n]$. Alkiot saattavat vaihtaa paikkojaan **then**-haaran alussa, mutta tämä ei sotke asiaa, koska $A[y] \le raja$ toteutuu tällaisella taulukolla riippumatta siitä, mikä alkio on kulkeutunut paikkaan $y$. Tällä taulukolla $x$ käyttää hyväkseen joka ikisen mahdollisuutensa kasvaa. Se kasvaa kaikkiaan $n$ kertaa, joten lopullinen arvo on $n+1$.

Tällaista päättelyä voi usein helpottaa valitsemalla alkiot ahtaammin kuin mihin päättelyn alku antaisi mahdollisuuden. Tässä tapauksessa voi valita, että taulukon jokainen alkio on keskenään yhtäsuuri. Silloin ei tarvitse murehtia alkioiden siirtymisen vaikutusta edes sen vertaa kuin edellä, koska silloin alkioiden vaihdolla ei ole yhtään mitään vaikutusta. Lisärajoitus ei tässä tapauksessa sulje pois mitään tapaa saada $x$:lle vielä suurempi loppuarvo, koska joka ikinen mahdollisuus kasvattaa $x$:ää saadaan lisärajoituksen vallitessakin käytettyä hyväksi.

Pisteytin tämä kaikki tai ei mitään -periaatteella.

**3.** (9p) Mikä on pienin arvo, joka muuttujalla $x$ voi olla tehtävän 1 ohjelman lopussa? _____

Miksi se on mahdollinen ja pienin mahdollinen? _____

_____

Tässä täytyy koettaa saada **then**-haaraan meno mahdollisimman harvinaiseksi.

Tehtävän 1 yhteydessä näimme, miten voidaan päätellä, että aina silmukan alussa pätee $x \leq y$. Tästä huomaamme, että ohjelma ei koske alkioon $A[n]$ ennen kuin vasta viimeisellä kierroksella, koska sitä ennen $y < n$ ja siis myös $x < n$. Niinpä $A[n]$ säilyttää arvonsa kunnes testiin $A[y] \leq raja$ tullaan $y$:n arvolla $n$. Silloin testi vääjäämättä toteutuu, koska aiemmin on sijoitettu $raja := A[n]$, eikä myöskään $raja$:n arvo ole sen jälkeen muuttunut. Niinpä $x$ kasvaa alkuarvostaan ykkösestä ainakin kerran.

Taulukon muiden alkioiden alkuarvot voidaan valita suuremmiksi kuin $raja$ valitsemalla ne suuremmiksi kuin kohdan $n$ alkuarvo. Taulukkoon kohdistuvista sijoituksista riittää ymmärtää, että alkioon $A[n]$ ei kosketa ennen viimeistä kierrosta, joten sen $raja$:n kanssa yhtäsuuri arvo ei voi kopioitua muualle taulukkoon ennen kuin muu taulukko on käyty läpi. Niinpä tällaisella taulukolla ennen viimeistä kierrosta $A[y] > raja$, joten $x$ ei kasva ennen kuin viimeisellä kierroksella. Näin saadaan $x$:n loppuarvoksi 2.

Ensimmäiselle vastausviivalle tulee siis kirjoittaa “2”.

Tämänkin arvostelin perusteella täydet 3 pistettä tai ei yhtään.

Tehtävässä kysyttiin myös perustelu sekä sille, että 2 on mahdollinen arvo, että sille, että mikään pienempi arvo ei ole mahdollinen. Edellä on perustelua, mutta aivan liian paljon vastaukselle varattuun tilaan nähden.

Sille, että $x$ voi lopuksi olla 2 riittää perusteluksi vastauksessa esimerkiksi “toteutuu taulukolla [1, 1, . . . , 1, 2]”. Jos kysymyksessä pyydetään osoittamaan, että jotakin voi tapahtua, niin usein on kätevintä vastata antamalla esimerkki syötteestä, jolla kyseinen asia tapahtuu.

Hyväksyin täyden pisteen arvoisena myös vastauksen “toteutuu, kun $n = 1$”. Se on sikäli vastausta “[ 1, 1, . . . , 1, 2 ]” huonompi, että sen mukaisia taulukoita on olemassa vain yhtä kokoa. Myöhemmissä opinnoissa tietorakenteiden yhteydessä käy ilmi, että ohjelman suoritusta kuvaavat luvut annetaan usein syötteen koon — tässä tapauksessa siis luvun $n$ — funktiona. Vastaus $n = 1$ ei anna vastausta $n$:n funktiona, vaan ainostaan yhdellä $n$:n arvolla. Vielä tällä kurssilla ei vastaukselta kuitenkaan saa vaatia tällaisia hienouksia.

Tästä saattoi saada nollasta kolmeen pistettä.

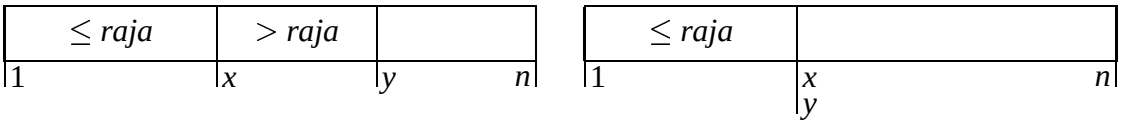
Sille, että 2 on pienin mahdollinen loppuarvo riitti perusteluksi mikä tahansa huomautus, jossa vedottiin siihen, että $x$ kasvaa varmasti ainakin viimeisellä kierroksella. Oikea vastaus tuotti 3 pistettä. Pisteitä tuli 2 silloin kun vastaus vihjasi, että $x$ kasvaa varmasti, mutta ei käyttänyt sitä perusteluna.

**4.** (6p) Merkitään symbolilla $\ell(i)$ sitä paikkaa, johon taulukon $A$ paikassa $i$ oleva alkio joutuu tehtävän 1 ohjelman suorituksen seurauksena. Oletetaan, että $1 \leq i < j \leq n$ ja $A[i] = A[j] \leq raja$ juuri ennen **for**-silmukkaa. Voiko $\ell(i) > \ell(j)$ (ts. voiko alkioiden $A[i]$ ja $A[j]$ keskinäinen järjestys vaihtua ohjelman suorituksen seurauksena)? _____

Perustele. _____

Tähän tehtävään vastaamiseksi ohjelman toiminta pitää ymmärtää aika hyvin (paitsi tietysti ensimmäisessä kohdassa saattaa osua oikeaan arvaamalla). Muistamme tehtävän 1 selityksestä, että aina **for**-silmukan alussa pätee $x \leq y$. Palauttakaamme mieliin tehtävässä 1 annettu kaava ja sitä havainnollistavat kuvat:

$$\forall i\,;\, 1 \leq i < x : A[i] \leq raja \;\wedge\; \forall i\,;\, x \leq i < y : A[i] > raja \;\; .$$




Ainoa kohta, missä ohjelma edes yrittää muuttaa taulukon $A$ sisältöä on $A[x]$:n ja $A[y]$:n vaihto **then**-haaran alussa.

Jos $x = y$, niin vaihdolla ei ole mitään vaikutusta.

Jos $x < y$, niin $A[y]$ on itse enintään *raja*:n suuruinen, ja siirtyy *raja*:a suurempien alkioiden yli. Myös vastakkaiseen suuntaan siirtyvä $A[x]$ on suurempi kuin *raja*.

Niinpä *raja*:a pienempi alkio ei missään oloissa siirry toisen *raja*:a pienemmän alkion yli eikä vaihda paikkaa sellaisen kanssa. Oikea vastaus on siis “ei”.

Perusteluksi riitti jälleen mikä tahansa, josta pystyin varmistumaan, että opiskelija on ymmärtänyt perussyyn miksi järjestys ei vaihdu. Esimerkiksi “*raja*:a pienemmät tai yhtäsuuret alkiot siirtyvät vain *raja*:a suurempien alkioiden yli”.

Kyllä-ei -kohta tuotti tietenkin joko 0 tai 3 pistettä. Perustelukohdasta saattoi saada pisteitä myös siltä väliltä.

**5.** (6p) Sama kysymys kuin edellä, mutta nyt $A[i] = A[j] > raja$. Vastaus: _____

Perustele. _____

Tämä tehtävä on sikäli edellistä helpompi, että kokeilemalla ohjelman toimintaa pienillä taulukoilla saattaa hyvässä lykyssä löytää sellaisen, jolla samansuuruiset *raja*:a suuremmat alkiot vaihtavat keskinäistä järjestystään. Sellaisen voi löytää myös miettimällä ohjelman toimintaa edellisen tehtävän selityksen tyyliin: on mahdollista suunnitella tilanne, jossa $A[x]$ hyppää itsensä suuruisen alkion yli. Samaa, *raja*:a suurempaa alkiota täytyy olla kaksi kappaletta peräkkäin, jotta olisi ylihyppääjä ja yli hypätty. Sen jälkeen täytyy olla enintään *raja*:n suuruinen alkio, jotta olisi jotakin, minkä kanssa ylihyppääjä vaihtaa paikkaa. Äkkipäätä miettien muuta ei tarvitakaan.

Nämä ehdot täyttää esimerkiksi taulukko $[2,2,1]$. Koeajamalla ohjelman mielessään kyseisellä taulukolla voi varmistua siitä, että kakkoset todellakin vaihtavat keskinäistä järjestystään.

Oikea vastaus on siis “kyllä voi”, ja perusteluksi riittää “niin käy taulukolla $[2,2,1]$”.

Perustelun arvostelussa oli käytössä koko skaala 0, 1, 2, 3. Kyllä-ei -kohdassa pistemäärä oli luonnollisesti joko 0 tai 3.

En sakottanut, jos perusteluna oli esimerkiksi taulukko $[3,2,1]$, jossa 3 ja 2 ovat *raja*:a suurempia ja vaihtavat keskinäistä järjestystään. Väärinhän tällainen perustelu on, koska se rikkoo tehtävässä asetetun ehdon $A[i] = A[j]$. Se käyttäytyy kuitenkin aivan samalla tavalla kuin $[2,2,1]$ ja siitä on helpompi nähdä, että vaihdos todella tapahtuu (lopputulos on $[1,2,3]$), joten päätin hyväksyä sen täysillä pisteillä. Tämä on yksi niistä tulkinnoista, joista en todellakaan lupaa, että linja tulee aina olemaan yhtä suopea.

Vastaus tyyppiä “sama, *raja*:a suurempi alkio kahdesti peräkkäin” ei tuottanut pisteitä, koska taulukko $[3,3,2,2]$ on sen mukainen, mutta siinä kolmosten keskinäinen järjestys ei vaihdu. Myöskin “taulukon toiseksi ja kolmanneksi viimeinen ovat viimeistä suurempia” ilman esimerkkitaulukkoa tuotti nollan, koska taulukko $[4,3,3,2]$ on sen mukainen, mutta siinä eivät kolmoset liiku.

* * *

Joku ihmetteli, mistä näin konstikas koodinpätkä on peräisin. Se on alkuosa N. Lomuton keksimästä ositusalgoritmista, joka on tarkoitettu osaksi Quicksortia. Poimin sen kirjan [1] Quicksortia käsittelevän luvun harjoitustehtävästä. Kirjan uudempi laitos [2] on ylentänyt algoritmin osaksi Quicksortin esitystä varsinaisessa tekstissä ja samalla tehnyt siihen pienen parannuksen. Quicksort lienee maailman kuuluisin ja eniten käytetty järjestämisalgoritmi ja [1] on eräs maailman eniten käytettyjä tietorakenteiden oppikirjoja, joten koodinpätkää ei todellakaan voi väittää epärealistiseksi.

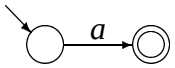
**6.** (9p) Olkoon $\Sigma = \{a, b, c\}$. Piirrä mahdollisimman pienet deterministiset äärelliset automaatit siten, että niiden hyväksymät kielet $A$, $B$ ja $C$ ovat kuvausten mukaiset.

| $\varepsilon \notin A$ ja $A$ on ääretön | $B$ on äärellinen ja $aab \in B$ | $C = \Sigma^* - \{aa\}$ |
|---|---|---|
| | | |

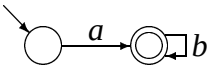
Jokaisella äärellisellä automaatilla on ainakin alkutila. Koska $\varepsilon$ eli tyhjä merkkijono ei kuulu kieleen $A$, ei ensimmäisen kohdan automaatin alkutila voi olla lopputila.

Jokin lopputila automaatissa on kuitenkin oltava, koska sen tulee hyväksyä ainakin yksi merkkijono. (Itse asiassa sen tulee hyväksyä äärettömän monta merkkijonoa.)

Lopputilaan täytyy päästä, muuten sillä ei ole merkitystä. Niinpä vedämme alkutilasta sinne kaaren. Kaarelle täytyy antaa jokin aakkonen nimeksi, ei väliä mikä.




Tässä vaiheessa automaatti hyväksyy vain yhden merkkijonon. Nyt ei enää tarvitse muuta kuin muuttaa määrä äärettömäksi. Se onnistuu esimerkiksi tekemällä lopputila osaksi silmukkaa. Siis vedämme jonkinnimisen kaaren lopputilasta joko alkutilaan tai itseensä, miten huvittaa.



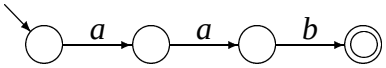

Jos vastauksessa oli enemmän kaaria kuin tarpeen, annoin vain 2 pistettä, koska tehtävässä kysyttiin mahdollisimman pientä automaattia.

* * *

Koska kielen $B$ on oltava äärellinen, ei automaatissa saa olla silmukoita missään kohdassa, jolla on hyväksytyn kielen kannalta merkitystä. (Tilalla on merkitystä, jos sinne pääsee alkutilasta ja

siitä pääsee ainakin yhteen lopputilaan.) Merkityksettömiä tiloja emme halua piirtää, koska automaatti ei ole mahdollisimman pieni jos siinä on tarpeetonta roinaa. Siis vältämme silmukoita kokonaan. Tämä mielessä pitäen käymme työhön.

Kannattaa aloittaa huolehtimalla, että automaatti hyväksyy *aab*:n. Silmukoita ei saanut piirtää, joten piirrämme suoran pötkön tiloja.

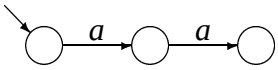



Muuta ei tarvitakaan.

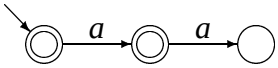
Ei ole merkitystä sillä, ovatko muut tilat kuin oikeanpuolimmaisin lopputiloja.

* * *

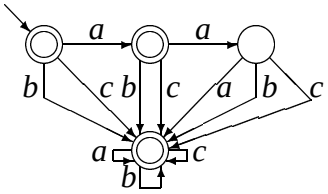
Tämä on edellisiä hankalampi. Merkkijono *aa* on erikoisasemassa, koska se on ainoa annetusta aakkostosta muodostettavissa oleva, joka ei kuulu kieleen *C*. Täytyy siis luoda rakennelma, joka käsittelee kyseisen jonon erikoistapauksena. Sen on oltava suora pötkö, koska jos siinä olisi silmukka, saataisiin silmukkaa kiertämällä muitakin jonoja erikoiskäsittelyn piiriin.




Hetkinen, nyt ε ja *a* hylätään, vaikka ne piti hyväksyä! Korjataan:




Kaikki muut jonot pitää hyväksyä. Luodaan tätä varten yhteinen hyväksymistila. Missä tahansa onkin tila josta puuttuu lähtökaari jollakin nimellä, vedetään kyseisestä tilasta kaari yhteiseen hyväksymistilaan kyseisellä nimellä, jotta kyseinen merkki ei tilan kohdalla esiintyessään aiheuttaisi automaatista “putoamista” ja siten merkkijonon hylkäämistä, vaan johtaisi hyväksymiseen.




Kaaren puuttuminen vei pisteen, samoin jos lopputilan paikalla oli epälopputila.

**7.** (3p) Kieli “ne aakkostosta $\{a,b\}$ muodostetut, pituudeltaan parittomat merkkijonot, joiden keskimmäinen merkki on *b*” ei ole säännöllinen. Kuvittele, että *k*-tilainen NFA hyväksyy ainakin kaikki ko. kieleen kuuluvat jonot, ja osoita, että se hyväksyy muutakin.

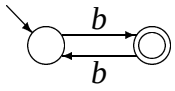
Tehtävä liittyy pumppauslemmaan. Yritin tehtävän muotoilulla pakottaa vastaamaan tavalla, joka osoittaa ymmärrystä siitä, miten pumppauslemmaa käytetään ja/tai miksi se pätee. Pelkkä maininta että pumppauslemmalla on jotain tekemistä tehtävän kanssa ei tuottanut pisteitä.

Koska automaatissa on vain $k$ tilaa, niin viimeistään $k$ merkkiä luettuaan automaatin täytyy palata tilaan, jossa se on ollut aiemminkin. Nimittäin, jos yritämme välttää paluuta aikaisempaan tilaan, niin tarvitsemme alkutilan lisäksi eri tilan, jossa ollaan ensimmäisen merkin lukemisen jälkeen; taas eri tilan, jossa ollaan toisen merkin lukemisen jälkeen ja niin edelleen. Kun $k-1$ merkkiä on luettu, on kaikki tilat käytetty. (Huomaa, että tämä ei osoita, että automaatti toistaa tilan *tasan* $k$ merkkiä luettuaan, vaan *viimeistään* $k$ merkkiä luettuaan. Toistohan voidaan aloittaa jo ennen kuin on aivan pakko. Sillä tavalla voidaan jättää yksi tila säästöön otettavaksi käyttöön vasta kun $k$:s merkki luetaan, niin että $k$:s merkki viekin uuteen tilaan.)

Oletetaan, että jokin $k$-tilainen automaatti hyväksyy ainakin kaikki kuvatun kaltaiset jonot. Silloin se hyväksyy myös jonon $a^k b a^k$. Äskeisen perusteella automaatti palaa aikaisempaan tilaan viimeistään $k$ merkkiä luettuaan. Se tapahtuu luettaessa merkkijonon alkuosaa $a^k$. Jos jätetään kyseisen tilan kahden käynnin välinen silmukka pois, päästään samaan tilaan kuin merkkijono $a^k$ lukemalla, mutta luettu onkin jokin $a^h$ missä $h < k$. Siitä voidaan jatkaa kuten merkkijonon $a^k b a^k$ tapauksessa. Silloin päädytään lopputilaan, ja on luettu $a^h b a^k$. Koska $h < k$, ei siinä ole $b$ keskellä. Automaatti siis hyväksyy jonon $a^h b a^k$, joka ei ole kuvauksen mukainen.

Pisteiden saamisen kannalta oli olennaista, että vastauksessa oli annettu jokin merkkijono jonka avulla yllä kuvatun kaltainen päättely menee läpi, ja todettu, että jokin pätkä sen alkuosasta voidaan toistaa tai poistaa siten että automaatti hyväksyy niin syntyvän jonon. Jonon tuli olla sellainen, että toiston tai poiston seurauksena syntyvässä jonossa ei varmasti ole sekä pituus pariton että $b$ keskellä. Tämän vuoksi esimerkiksi vastaukset, joissa jono saattoi koostua pelkistä $b$-kirjaimista eivät tuottaneet pisteitä.

Päättelyn täytyy todellakin taata että muodostettavassa jonossa ei ole $b$ keskellä, muuten se ei ole pätevä. Tätä havainnollistaa se, että kieli “ne aakkostosta $\{b\}$ muodostetut, pituudeltaan parittomat merkkijonot, joiden keskimmäinen merkki on $b$” *on* säännöllinen. Sen hyväksyy oheinen äärellinen automaatti:




Jos vastauksen ajatuksenjuoksua seuraamalla pystyy “todistamaan” myös tämän kielen epäsäännölliseksi, on vastaus ilman muuta virheellinen.

**8.** (3p) Ilmaise 7-tehtävän kieli yhteysriippumattomalla kieliopilla tai BNF:llä.

---

Otetaan lähtökohdaksi $b$, ja sitten kasvatetaan jonoa yhtäaikaa molemmista päistä. Siis:

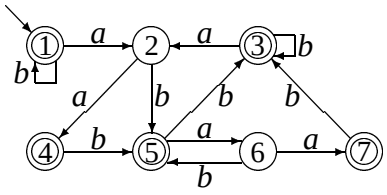
$$K ::= b \mid aKa \mid aKb \mid bKa \mid bKb \ .$$

Vastaukset tyyppiä $K ::= AbL$ ($A$ = alku, $L$ = loppu) eivät toimi. Nimittäin,

- Koska *aba* kuuluu kieleen, $A$:n täytyy voida tuottaa $a$.
- Koska *aaabaaa* kuuluu kieleen, $L$:n täytyy voida tuottaa *aaa*.

$\Rightarrow$ Niinpä *AbL* voi tuottaa jonon *abaaa*.

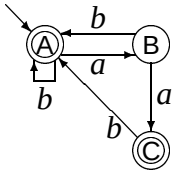
**9.** (6p) Minimoi oheinen äärellinen automaatti. Piirrä vastauksesi ruutuun. Näytä, miten minimoidun automaatin tilat vastaavat alkuperäisen automaatin tiloja.




Minimoinnin aluksi tilat on jaettava kahteen lohkoon: lopputilat ja muut. Kuvan automaatille näin saadaan lohkot $\{1,3,4,5,7\}$ ja $\{2,6\}$.

Seuraavaksi lohkoja on pilkottava sen mukaan, ovatko niiden tilat samaa mieltä siitä, mihin lohkoon joudutaan milläkin aakkosella. Nopeasti huomataan, että tilat 4 ja 7 eroavat tiloista 1, 3 ja 5 sikäli, että vain jälkimmäisistä lähtee *a*-kaaria. Lohko $\{1,3,4,5,7\}$ halkeaa siis osiin $\{1,3,5\}$ ja $\{4,7\}$.

Tämän jälkeen ei enää löydy syytä halkaista lohkoja. Saadaan kuvan mukainen vastaus.




A = $\{1,3,5\}$
B = $\{2,6\}$
C = $\{4,7\}$

**10.** (6p) Ilmaise tehtävän 9 automaatin tilan 6 hyväksymä kieli säännöllisenä lausekkeena.

Tehtävässä 9 annettua kuvaa katsomalla vastaus on työläs muodostaa, mutta apuna voidaan käyttää tehtävän vastausta. Lohkomisen perusideana on, että kun lohko on valmis, sen jokainen tila hyväksyy saman kielen. Myös koko lohko hyväksyy saman kielen. Riittää siis selvittää, minkä kielen hyväksyy vastauksen se lohko, johon tila 6 kuuluu — siis lohko B.

Tilasta B päästään lopputiloihin joko menemällä *a*:lla tilaan C, tai menemällä tilaan A joko *b*:llä tai *ab*:llä. Tilassa A voi pyöriä paikallaan *b*:llä. Näitä vastaa säännöllinen lauseke $a \mid (b \mid ab)b^*$. Tämän voi kirjoittaa hieman sievempään muotoon $(a \mid b)b^*$.

Tässä ei kuitenkaan ole vielä koko totuus. Nimittäin, tilasta A voi palata tilaan B. Tämä tekee mahdolliseksi kiertää kuvaa ympäri ennen lopullista siirtymistä lopputilaan. Kaikki mahdolliset tavat tehdä kierros esittää lauseke $(b \mid ab)b^*a$. Lauseke, joka sallii nolla tai useampia kierroksia ja sitten vie lopputilaan saadaan pistämällä tämän toisto ja edellä johdettu kaava peräkkäin:

$$((b \mid ab)b^*a)^*(a \mid b)b^* \ .$$

*Aikalogiikka* määritellään tässä tentissä kieliopilla

*Kaava* ::= [ *Kaava* “$\mathcal{U}$” ] *K*1 *K*2 ::= [ *K*2 “$\wedge$” ] *K*3

*K*1 ::= [ *K*1 “$\vee$” ] *K*2 *K*3 ::= [ “$\neg$” | “$\Box$” | “$\Diamond$” | “$\bigcirc$” ] ( *P* | “(” *Kaava* “)” ),

missä *P* on propositiosymboli tai vakio T (eli True) tai F (eli False). Operaattoreiden presedenssi ja liitännäisyys noudattavat tätä kielioppia.

Aikalogiikan kaava puhuu päättymättömän ajanhetkien jonon $s_0 s_1 s_2 \cdots$ ominaisuuksista. Kyseessä voi olla esimerkiksi päivien jono, jolloin $s_0$ = tänään, $s_1$ = huomenna, $s_2$ = ylihuomenna jne. Sanotaan, että kaava *pätee*, joss se pätee hetkellä $s_0$. Se, että kaava $\varphi$ pätee hetkellä $s_i$ merkitään $pät(\varphi, i)$, ja määritellään uusille operaattoreille seuraavasti:

$pät(\bigcirc\varphi, i) \iff pät(\varphi, i+1)$

$pät(\Diamond\varphi, i) \iff \exists j : j \geq i \wedge pät(\varphi, j)$

$pät(\Box\varphi, i) \iff \forall j : (j \geq i \rightarrow pät(\varphi, j))$

$pät(\varphi\ \mathcal{U}\ \psi, i) \iff \exists j : j \geq i \wedge pät(\psi, j) \wedge \forall k \in \{i, \ldots, j-1\} : pät(\varphi, k)$

Operaattoreilla “$\wedge$”, “$\vee$” ja “$\neg$” on propositiologiikasta tuttu merkitys. Esimerkiksi $\bigcirc$(Sataa $\wedge$ $\bigcirc\neg\Diamond$Sataa) tarkoittaa, että huomenna sataa, mutta sen jälkeen ei enää koskaan sada.

Tehtävien 11, 12, 13 ja 18 ratkaisemiseksi riittää, että hallitsee kurssin lausekkeita ja niihin liittyviä puita koskevat asiat. Tehtävissä 14, 15, 16 ja 17 on tarpeen ymmärtää yllä annetusta selostuksesta myös aikalogiikan merkitystä kuvaava osuus.

Ennen kuin käydään ratkaisemaan tehtäviä, palautetaan mieliin, että “[ ... ]” on valinnaisuutta tarkoittava lyhennemerkintä. Siis esimerkiksi sääntö *Kaava* ::= [ *Kaava* “$\mathcal{U}$” ] *K*1 tarkoittaa itse asiassa sääntöä *Kaava* ::= *K*1 | *Kaava* $\mathcal{U}$ *K*1, mikä puolestaan tarkoittaa samaa kuin kaksi sääntöä *Kaava* ::= *K*1 ja *Kaava* ::= *Kaava* “$\mathcal{U}$” *K*1.

**11.** (3p) Mitkä aikalogiikan operaattorit ovat unaarioperaattoreita?

Unaarioperaattoreita ovat ne, joilla on vain yksi argumentti. Sekä kieliopista etta *pät*-kaavoista näkee, että $\Box$, $\Diamond$ ja $\bigcirc$ ovat unaarioperaattoreita, ja $\mathcal{U}$ on binäärioperaattori eli kaksiargumenttinen operaattori. Operaattorin $\neg$ näkee unaarioperaattoriksi ja operaattorit $\wedge$ ja $\vee$ binäärioperaattoreiksi sekä kieliopista että huomautuksesta, että niillä on propositiologiikasta tuttu merkitys.

Jos vastauksesta puuttui $\neg$, niin annoin 2 pistettä. Ajattelin, että vastaaja oli ehkä tulkinnut, että vanhastaan tuttuja propositiologiikan operaattoreita ei lasketa aikalogiikan operaattoreiksi.

**12.** (3p) Mitkä aikalogiikan operaattorit ovat infix?

Infix-operaattori on argumenttiensa välissä. Unaarioperaattori ei voi olla infix, koska ei ole “argumenttien väliä” jos on vain yksi argumentti. Jäljelle jääviä ehdokkaita infix-operaattoreiksi ovat edellisen tehtävän selityksessä binäärioperaattoreiksi luokitellut operaattorit. Niistä jokainen voidaan havaita infix-operaattoriksi kieliopista.

Vaikkei osaisi lukea kielioppia, niin esimerkeistä näkyy, että $\mathcal{U}$ ja $\wedge$ ovat infix. $\wedge$ ja $\vee$ ovat normaalissa käytössään infix. Tässä vaiheessa on helppo arvata, että $\vee$ on infix myös aikalogiikassa.

Tulkinnasta, että propositiologiikan operaattorit eivät ole aikalogiikan operaattoreita ei sakotettu tässä kohdassa enää toistamiseen. Sen sijaan sakotettiin, jos edellisen tehtävän vastaus oli sen mukainen mutta tämän ei.

**13.** (3p) Mitenpäin liitännäinen aikalogiikan "$\mathcal{U}$" on? _____

Asian voi katsoa kieliopista, koska tehtävässä on nimenomaan annettu siihen lupa. (Mistään muualta sitä ei näekään.) Sieltä näkyy, että ainoa tapa tuottaa operaattori $\mathcal{U}$ on *Kaava* ::= *Kaava* $\mathcal{U}$ $K1$. $K1$ ei voi sisältää $\mathcal{U}$-operaattoria (paitsi sulkujen sisällä), mutta *Kaava* voi. Niinpä esimerkiksi $P\ \mathcal{U}\ Q\ \mathcal{U}\ R$ jäsentyy $\underline{P\ \mathcal{U}\ Q}\ \mathcal{U}\ \underline{R}$.

$\mathcal{U}$ on siis vasemmalle liitännäinen.

**14.** (3p) Operaattorin "$\bigcirc$" nimeksi sopii "seuraavaksi" tai "huomenna". Mikä sopisi operaattorin "$\Box$" nimeksi? _____

Aikalogiikan esittelyssä määriteltiin, että hetkellä $i$ esitetty väite $\bigcirc\varphi$ tarkoittaa samaa kuin hetkellä $i+1$ esitetty väite $\varphi$. Vastaavasti hetkellä $i$ esitetty väite $\Box\varphi$ väittää, että $\varphi$ pätee hetkinä $i$, $i+1$, $i+2$ jne., siis jokaisena hetkenä hetkestä $i$ alkaen. Sen nimeksi sopii siis "tästä lähtien", "siitä lähtien" tai "vastedes". Jostain syystä sen englanninkieliseksi nimeksi on vakiintunut "always" (vaikka myös nimi "henceforth" on vilahtanut kirjallisuudessa), joten sen suora suomennos "aina" kelpasi myös vastaukseksi täysin pistein. Nimellä "aina" se suomeksi tunnetaankin.

Vastaukset "kaikki seuraavat", "jälkeen" ja "myöhemmin" tuottivat nolla pistettä, koska ne eivät pidä sisällään tarkastelupäivää. Kaava $\Box$Sataa ei sano "huomisesta alkaen joka päivä sataa", vaan "tästä päivästä alkaen …". "Tulevaisuudessa" tuotti nolla pistettä, koska se ei sisällä sitä että asia on voimassa *jokainen* hetki *tarkasteluhetkestä alkaen*: "tulevaisuudessa ihmiset matkustelevat muille planeetoille". Sen sijaan "tänään ja tulevaisuudessa" kuulosti minusta jo niin oikealta, että annoin siitä kaksi pistettä.

**15.** (3p) Kirjoita aikalogiikan kaavana "sataa äärettömän monena päivänä". _____

Tässä pääsee ehkä helpoiten alkuun miettimällä väitteen vastakohtaa: "sataa äärellisen monena päivänä". Jos niin on, niin sitten on olemassa päivä, jona ei sada ja jonka jälkeen ei enää koskaan sada. (Olisi helpompi sanoa, että on olemassa viimeinen päivä, jona sataa, mutta se ei olisi oikein. Myös se, että ei koskaan sada toteuttaa väitteen "sataa äärellisen monena päivänä".)

Mutta tällaista päivää ei siis saa olla. Eletään mitä päivää tahansa, niin "ei enää koskaan sada" ei saa olla totta, vaan tulevaisuudessa (heti tai myöhemmin) täytyy olla sadepäivä. "Tulevaisuudessa sataa" voidaan ilmaista kaavalla $\Diamond$Sataa. Tämä vaaditaan jokaiselta päivältä, joten vastaukseksi tulee $\Box\Diamond$Sataa.

Vastaus $\Box$Sataa ei ole oikein, koska se väittää, että joka ikinen päivä sataa. Annoin siitä nolla pistettä. Jos esimerkiksi joka toinen päivä sataa ja joka toinen ei, niin silloin sataa äärettömän monena päivänä, vaikka ei sada joka päivä.

**16.** (3p) Koulumatematiikan kertolasku voidaan esittää muiden matemaattisten operaattoreiden avulla: $a \cdot b = \frac{(a+b)^2-(a-b)^2}{4}$. Myös aikalogiikassa operaattori $\Box$ voidaan esittää muiden operaattoreiden avulla. Kirjoita kaava, joka tekee niin. $\Box\varphi \Leftrightarrow$ _____

Määritelmän $pät(\Box\varphi, i) \iff \ldots$ oikea puoli alkaa operaattorilla $\forall$. Operaattorin $\Diamond$ vastaava kohta alkaa houkuttelevasti $\exists$, ja $\forall$ ja $\exists$ saadaan toisistaan De Morganin kaavalla. Siis

$$\begin{array}{rcl@{\qquad}rcl}
 & pät(\Box\varphi, i) & & \Leftrightarrow & \forall j : (j \geq i \to pät(\varphi, j)) \\
\Leftrightarrow & \neg\neg\forall j : (j \geq i \to pät(\varphi, j)) & & \Leftrightarrow & \neg\exists j : \neg(j \geq i \to pät(\varphi, j)) \\
\Leftrightarrow & \neg\exists j : \neg(\neg(j \geq i) \vee pät(\varphi, j)) & & \Leftrightarrow & \neg\exists j : \neg\neg(j \geq i) \wedge \neg pät(\varphi, j) \\
\Leftrightarrow & \neg\exists j : j \geq i \wedge pät(\neg\varphi, j) & & \Leftrightarrow & \neg pät(\Diamond\neg\varphi, i) \\
\Leftrightarrow & pät(\neg\Diamond\neg\varphi, i) \ . & & &
\end{array}$$

Siis $\Box\varphi \Leftrightarrow \neg\Diamond\neg\varphi$.

Vastauksen voi löytää myös miettimällä ilmausten merkityksiä: “aina sataa” tarkoittaa samaa kuin “ei ole niin, että joskus ei sada”.

**17.** (12p) Sievennä. $\mathsf{F}\ \mathcal{U}\ \varphi \Leftrightarrow$ _____ $\mathsf{T}\ \mathcal{U}\ \varphi \Leftrightarrow$ _____ $\varphi\ \mathcal{U}\ \mathsf{F} \Leftrightarrow$ _____ $\varphi\ \mathcal{U}\ \mathsf{T} \Leftrightarrow$ _____

Se, että $\varphi\ \mathcal{U}\ \psi$ pätee hetkellä $i$ on määritelty tarkoittamaan $\exists j : j \geq i \wedge pät(\psi, j) \wedge \forall k \in \{i, \ldots, j-1\} : pät(\varphi, k)$. Kun $\varphi$:n paikalle sijoitetaan $\mathsf{F}$ ja $\psi$:n paikalle $\varphi$ saadaan $\exists j : j \geq i \wedge pät(\varphi, j) \wedge \forall k \in \{i, \ldots, j-1\} : pät(\mathsf{F}, k)$. Väittämä $\mathsf{F}$ ei päde missään tilassa (väittämä pätee niissä tiloissa, joissa sen totuusarvoksi tulee $\mathsf{T}$, mutta väittämän $\mathsf{F}$ totuusarvo on tietenkin aina $\mathsf{F}$), joten saamme edellisen muotoon $\exists j : j \geq i \wedge pät(\varphi, j) \wedge \forall k \in \{i, \ldots, j-1\} : \mathsf{F}$. Kysymys on siis siitä, onko olemassa sellaista arvoa $j$:lle, että $j \geq i$ ja $pät(\varphi, j)$ ja $\forall k \in \{i, \ldots, j-1\} : \mathsf{F}$.

Jos $j > i$, niin $\forall k \in \{i, \ldots, j-1\} : \mathsf{F} \Leftrightarrow \mathsf{F}$, ja samalla koko kaava on auttamatta $\mathsf{F}$. Mutta jos $j = i$, niin $\forall k \in \{i, \ldots, j-1\} : \mathsf{F} \Leftrightarrow \mathsf{T}$, koska silloin väitetään jotakin tyhjän joukon alkioista. Siinä tilanteessa kaava sievenee muotoon $pät(\varphi, i)$. Kaivattu $j$ on siis olemassa, jos ja vain jos $\varphi$ pätee hetkellä $i$. Siis vastaus on $\varphi$.

Toinen tapa on ajatella kaavojen merkitystä. Kaava $\varphi\ \mathcal{U}\ \psi$ tarkoittaa, että jonain hetkenä nyt tai tulevaisuudessa $\psi$ pätee, ja sitä edeltävinä hetkinä nykyhetkestä alkaen $\varphi$ pätee. Tässä tapauksessa $\varphi$:n paikalla on $\mathsf{F}$, joka ei päde koskaan. Edeltäviltä hetkiltä vaaditaan siis mahdotonta. Niinpä ainoa keino on, että edeltäviä hetkiä ei ole. Siis pitää olla niin, että $\psi$:n paikalla oleva kaava eli $\varphi$ pätee nyt heti.

Kun $\varphi$:n paikalle sijoitetaan $\mathsf{T}$ ja $\psi$:n paikalle $\varphi$ saadaan $\exists j : j \geq i \wedge pät(\varphi, j) \wedge \forall k \in \{i, \ldots, j-1\} : pät(\mathsf{T}, k)$. Väite $\mathsf{T}$ pätee joka tilanteessa, joten $\forall k \in \{i, \ldots, j-1\} : pät(\mathsf{T}, k) \Leftrightarrow \mathsf{T}$. Koko kaavasta saadaan $\exists j : j \geq i \wedge pät(\varphi, j) \wedge \mathsf{T}$ eli $\exists j : j \geq i \wedge pät(\varphi, j)$, mikä on kaavan $\Diamond\varphi$ määritelmä.

Kun $\varphi$:n paikalle sijoitetaan $\varphi$ ja $\psi$:n paikalle $\mathsf{F}$ saadaan $\exists j : j \geq i \wedge pät(\mathsf{F}, j) \wedge \forall k \in \{i, \ldots, j-1\} : pät(\varphi, k)$. Koska $pät(\mathsf{F}, j)$ tuottaa aina $\mathsf{F}$ (eli $\mathsf{F}$ ei päde koskaan), on $\wedge$-merkkien välinen osuus ja samalla koko kaava yhtä kuin $\mathsf{F}$.

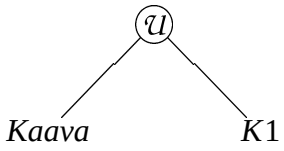
Kun $\varphi$:n paikalle sijoitetaan $\varphi$ ja $\psi$:n paikalle $\mathsf{T}$ saadaan $\exists j : j \geq i \wedge pät(\mathsf{T}, j) \wedge \forall k \in \{i, \ldots, j-1\} : pät(\varphi, k)$. Osuus $pät(\mathsf{T}, j)$ tuottaa $\mathsf{T}$, ja voidaan siis poistaa. Jäljelle jää $\exists j : j \geq i \wedge \forall k \in \{i, \ldots, j-1\} : pät(\varphi, k)$. Valitsemalla $j = i$ saadaan $\exists$-kvantifikaattorin jälkeinen kaavan osa toteutumaan aina, joten koko väite sievenee muotoon $\mathsf{T}$.

**18.** (6p) Piirrä lausekepuu ja jäsennyspuu kaavalle $\bigcirc P\ \mathcal{U}\ Q \wedge R\ \mathcal{U}\ R$.

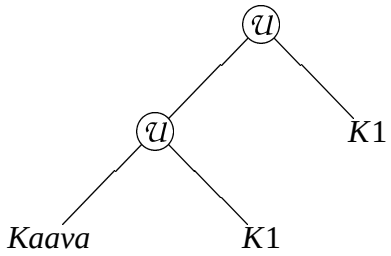
Lausekepuu esittää puumuodossa sen minkä lauseke esittää tekstimuodossa. Lausekkeen operaattoreista tulee puun solmuja, joilla on lapsinaan muita solmuja lausekkeen rakenteen mukaan. Puun lehtinä ovat ne perusosat, joihin operaattorit kohdistuvat. Tässä tapauksessa siis lehtiä ovat $P$, $Q$ ja kaksi $R$:ää, ja muina solmuina $\bigcirc$, $\mathcal{U}$, $\wedge$ ja $\mathcal{U}$.

Operaattorisolmut pitää saada oikeaan asemaan suhteessa toisiinsa ja lehtiin. Asemat määräytyvät presedenssi- ja liitännäisyyssäännöistä. Aikalogiikan esittelyssä sanottiin, että ne ovat kieliopin mukaiset.

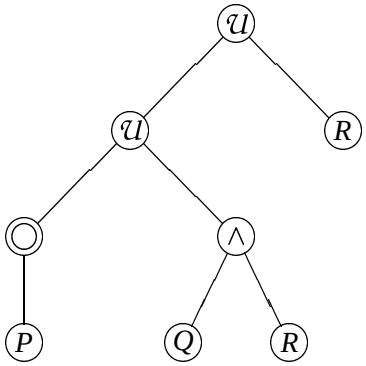
Kieliopin "ylin" sääntö on *Kaava* ::= *K*1 tai *Kaava* ::= *Kaava* "$\mathcal{U}$" *K*1. Tehtävässä annetussa lausekkeessa esiintyy $\mathcal{U}$. Sitä ei voi tuottaa muuten kuin näistä säännöistä jälkimmäisellä. Niinpä lauseke pitää jakaa ylimmällä tasolla osiin seuraavasti:




Lausekkeen toinen $\mathcal{U}$ voidaan tuottaa *Kaava*:sta, mutta *K*1:stä sitä ei saa (paitsi säännöillä *K*3 ::= ... "(" *Kaava* ")", mutta silloin lausekkeeseen ilmestyvät sulut, joita siinä ei todellisuudessa ole). Siispä rakenne jatkuu



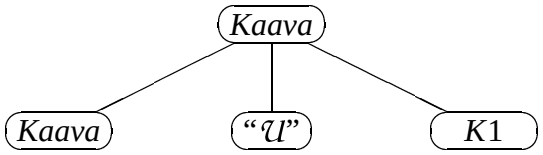

Kielioppia voisi käyttää tiedon lähteenä tästä eteenkinpäin, mutta se ei ole tarpeen, sillä loput lausekkeesta $\bigcirc P\ \mathcal{U}\ Q \wedge R\ \mathcal{U}\ R$ voi sovittaa piirrokseen vain yhdellä tavalla. Koska $\bigcirc P$ on ennen ensimmäistä $\mathcal{U}$-operaattoria ja operaattorin kohde roikkuu operaattorista, on ensimmäisen $\mathcal{U}$-operaattorin vasemmaksi lapseksi ripustettava $\bigcirc$ ja sen ainoaksi lapseksi $P$. Vastaavalla tavalla $Q \wedge R$ ripustetaan keskelle kuvaa $\wedge$ ylimmäiseksi, ja oikeaan reunaan tulee $R$.



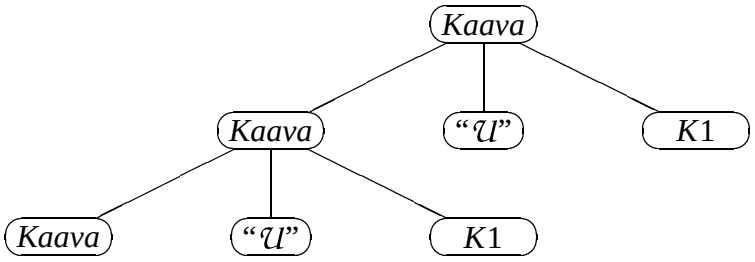

Jos kuva oli piirretty ikään kuin operaattorin $\mathcal{U}$ presedenssi olisi korkeampi tai sama kuin operaattorin $\wedge$, niin yksi piste katosi. Samoin kävi, jos sitovuuden suunta oli väärin.

* * *

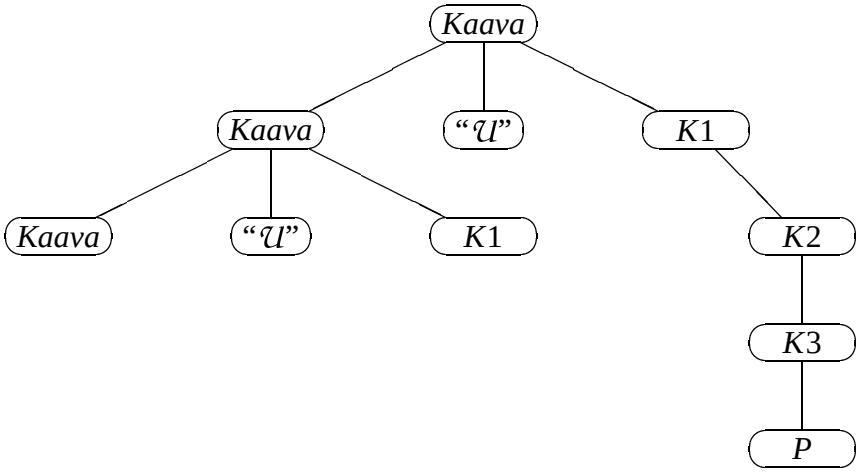
Jäsennyspuun muodostus etenee sovittamalla kielioppisääntöjä annettuun lausekkeeseen aloittamalla *Kaava*:sta. Koska ainoa taso, jolla $\mathcal{U}$ voi ilmestyä kaavaan on sääntö *Kaava* ::= *Kaava* "$\mathcal{U}$" *K*1, on pakko käyttää sitä jotta saataisiin aikaan kaavassa esiintyvä $\mathcal{U}$. Näin saadaan



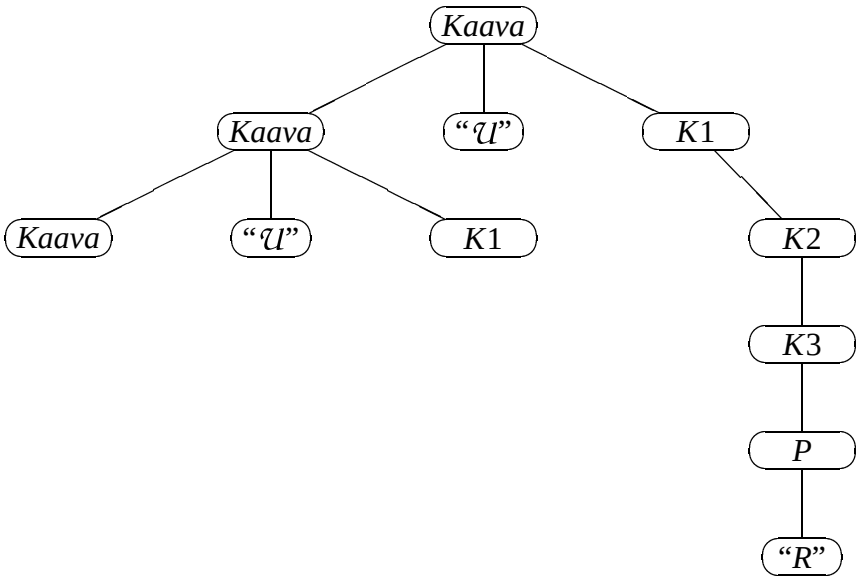

Lausekkeessa olevan toisen $\mathcal{U}$-operaattorin täytyy piiloutua alas vasemmalle ilmestyneeseen *Kaava*:an. Kaivetaan se esiin soveltamalla äskeistä sääntöä toistamiseen.



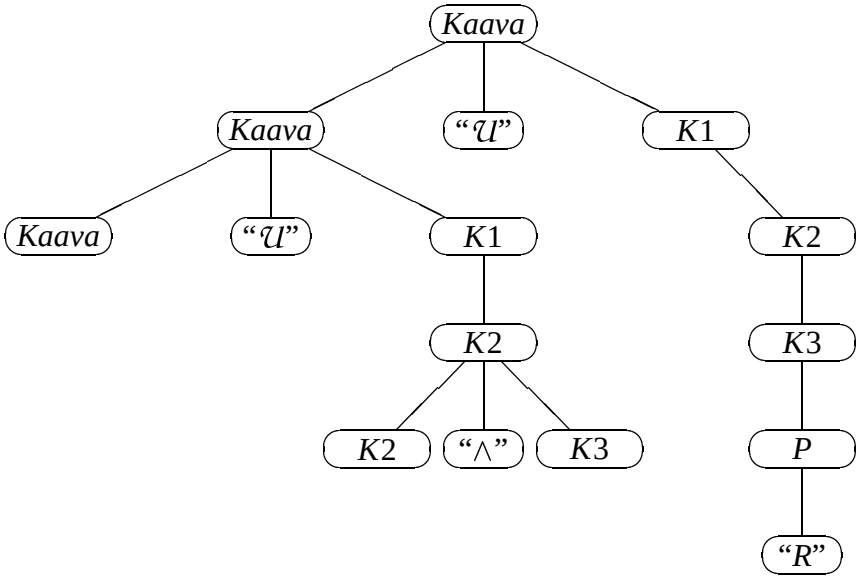

Seuraavaksi voidaan vaikka purkaa oikeanpuoleinen *K*1 vaiheittain auki säännöillä *K*1 ::= *K*2, *K*2 ::= *K*3 ja *K*3 ::= *P*.



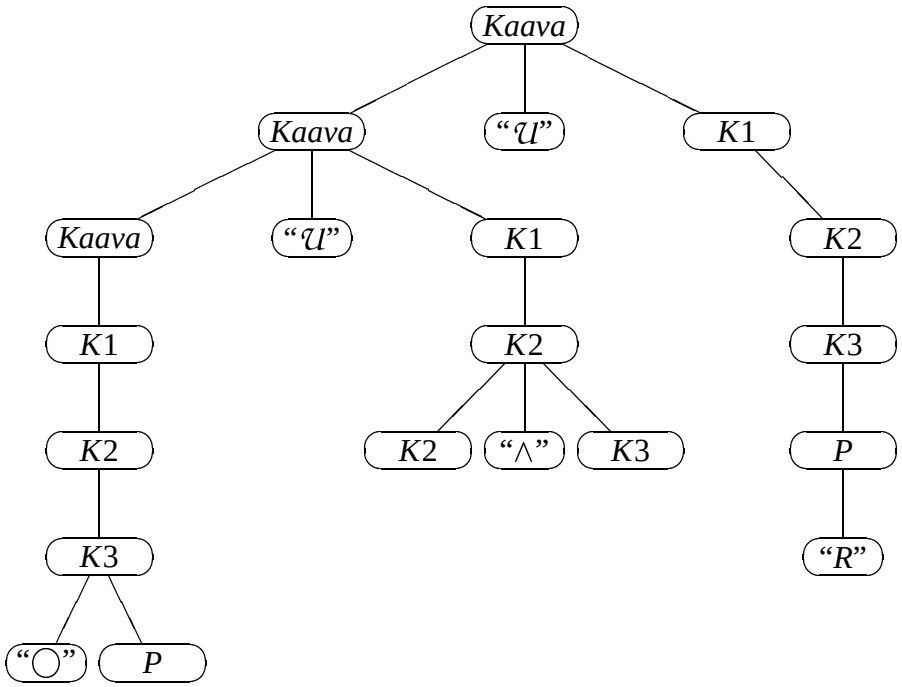

Lisätään tähän tieto, että *P* on propositiosymboli. Annetun kaavan viimeinen symboli on "*R*", joten *P*:n täytyy edustaa "*R*":ää.



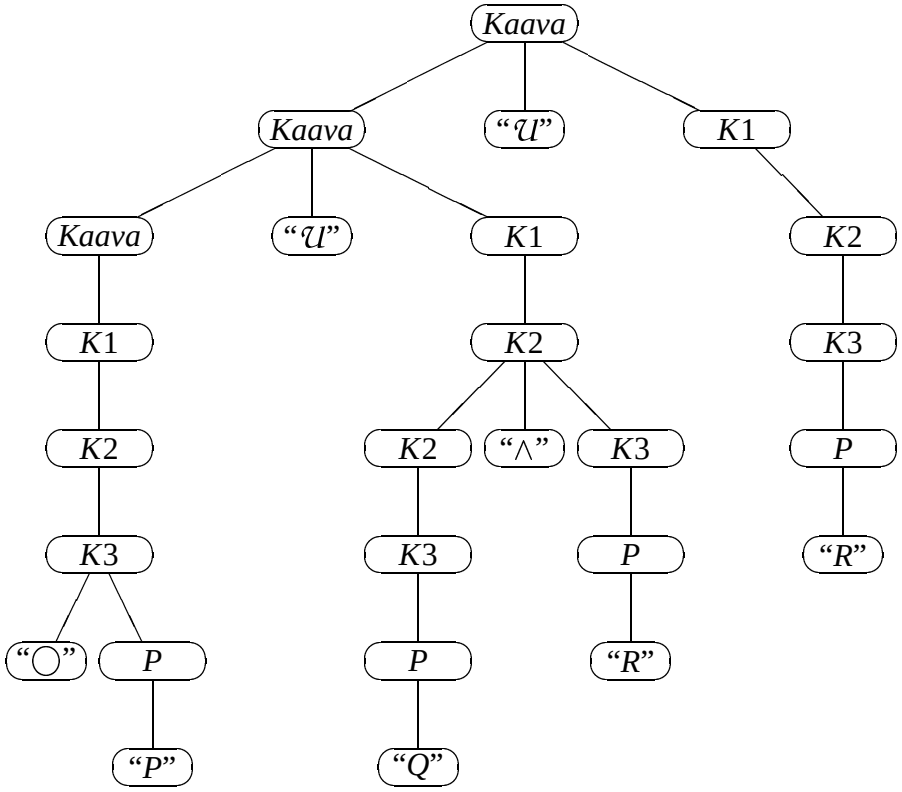

Keskialueelle pitää tavalla tai toisella tuottaa “$\wedge$”. Se onnistuu purkamalla $K1$ ensin $K2$:ksi ja sitten soveltamalla sääntöä $K2 ::= K2$ “$\wedge$” $K3$.




Vasemmanpuoleisesta *Kaava*:sta pitää jatkaa säännöllä *Kaava* ::= $K1$, koska enää ei tarvita lisää $\mathcal{U}$-operaattoreita. $K1$:stä voidaan jatkaa tuttuun tapaan kunnes vastaan tulee $K3$. Siitä pitää saada $\bigcirc P$. Tämä onnistuu kun seuraavaksi käytetään sääntöä $K3 ::= \bigcirc P$.




Sitten voidaan viedä työ loppuun soveltamalla suoraviivaisia sääntöjä kunnes kaikki keskeneräiset haarat ovat päättyneet propositioiden nimiin.




* * *

Aikalogiikan juuret ovat matemaatikoiden ja filosofien tutkimissa modaalilogiikoissa. Tietojenkäsittelyn tutkimukseen aikalogiikan toivat israelilaiset Amir Pnueli ja Zohar Manna vuonna

1977. Aikalogiikka osoittautui hyväksi työkaluksi puhelinkeskusten ja hissinohjausohjelmistojen kaltaisten, yhtä aikaa moneen suuntaan kuulolla olevien ohjelmistojen ymmärtämiseksi. Pnueli sai vuonna 1996 tietojenkäsittelyalan arvostetuimman palkinnon eli Turing-palkinnon aikalogiikan tuomisesta tietojenkäsittelyyn ja ohjelmien ja järjestelmien oikeaksi osoittamisen tutkimuksesta.

# Viitteet

[1] T. H. Cormen, C. E. Leiserson & R. L. Rivest: *Introduction to Algorithms*. The MIT Press, 1990.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest & C. Stein. *Introduction to Algorithms, 2nd edition*. The MIT Press, 2001.