

# Muodolliset koneet ja laskettavuus

Juuso Valli

13. 12. 2016

## Tiivistelmä

Tässä tutkielmassa käsitellään laskettavuuden peruskäsitteitä, tutustutaan rekisteri- ja Turingin koneisiin sekä osoitetaan näiden olevan laskentakyvyltään identtisiä.

## Sisältö

1	Algoritmit ja muodolliset koneet	2
2	Laskettavuus	3
3	Muodolliset kielet	3
4	Automaatit	3
5	Rekisterikoneet	6
6	Turingin koneet	7
7	Funktioiden esittäminen muodollisella koneella	8
8	Rekisterikoneiden ja Turingin koneiden välinen yhteys	9
A	Lähteet	11

# 1 Algoritmit ja muodolliset koneet

Algoritmillä tarkoitetaan äärellistä luetteloa ohjeita tai komentoja, joita noudattamalla voidaan ratkaista jokin tietty ongelma. Ratkaisulla tarkoitetaan yleensä vastausta kyllä-ei kysymykseen tai jonkinlaista numeerista tulosta annetun syötteen perusteella. Eukleideen algoritmi kahden luvun suurimman yhteisen tekijän selvittämiseen on eräs vanhimmista tunnetuista algoritmeista. Algoritmit ovat historiallisesti olleet merkittävä osa käytännön matematiikkaa; tavallisimpien algoritmien tuntemus on mahdollistanut monimutkaisten ongelmien matemaattisen ratkaisemisen ilman syvällistä ymmärrystä. Mikäli oppilas pystyy oppimaan algoritmin, hän pystyy yksinkertaisia ohjeita noudattamalla ratkaisemaan kokonaisen luokan ongelmia. Tässä merkityksessä algoritmit eivät ole menettäneet arvoaan, matematiikan alkeisopetus nojaa edelleen vahvasti yksinkertaisten algoritmien opetteluun. Algoritmien toinen vahvuus seuraa siitä tosiasiasta, että algoritmin suorittajan ei tarvitse olla ajatteleva ihminen. Sopivalla tavalla suunniteltu mekaaninen tai sähköinen kone pystyy myös noudattamaan algoritmin ohjeita. Tämä tarjoaa ihmisuorittajaan verrattuna lukuisia etuja. Koneen suorittamat algoritmit ovat vähemmän alttiita virheille ja useimmissa tapauksissa huomattavasti ihmistä nopeampia. Toisaalta ihmiset pystyvät käsittelemään paljon vapaamuotoisempia algoritmeja. Koneet yleensä pystyvät käsittelemään vain hyvin rajallista komentokantaa.

Algoritmin komentokannan rajoittamisesta on etua. Täysin vapaamuotoisia algoritmeja on vaikea tutkia; algoritmin oikeellisuus on vaikea todistaa täsmällisesti, algoritmin tulos voi vaihdella kommentojen tulkinnan mukaan, ja kahden algoritmin vertailu on lähes mahdotonta. Muodollisen määrittelyn kautta päästään eroon monista epävarmuuksista. Komennot ovat yksiselitteisiä, ja kahden algoritmin suoritusaikaa voidaan verrata laskemalla suoritettujen kommentojen lukumäärä. Algoritmin oikeellisuus on usein helpompi todistaa ja joskus voidaan osoittaa, että ratkaisualgoritmia ei ole olemassa tämän komentokannan puitteissa.

Komentokannan käsite on vahvasti sidoksissa siihen, millainen algoritmin suorittajan ajatellaan olevan. Vapaamuotoisen algoritmin suorittajan ajatellaan olevan ihminen. Muissa tapauksissa algoritmin suorittaja on jollakin tavalla määritelty formaalinen kone. Formaaleilla koneilla ei ole kaikenkattavaa määritelmää, vaan jokainen määritellään eri tavoin. Formaalista koneesta käytetään tässä merkintää  $M : S \rightarrow R$ , missä  $S$  on syöte ja  $R$  on algoritmin tulos. On tärkeä huomata että tässä  $M$  ei ole funktio  $S \rightarrow R$ , sillä formaalinen kone ei välttämättä koskaan lopeta suoritusta annetulla syötteellä. Kone  $M$  voidaan täydentää funktioksi  $S \rightarrow R \cup \{\infty\}$ , missä  $\infty \notin R$  tarkoittaa, että  $M$  ei pysähdy äärellisessä ajassa.

Tietyllä tavalla määriteltyjen formaalien koneiden avaruudesta käytetään tässä merkintää  $\hat{M}$ .

## 2 Laskettavuus

Eräs algoritmitutkimuksen peruskysymys on selvittää, mitkä ongelmat ovat ratkaistavissa algoritmisesti. Tämä vaatii ensin muodollisen määritelmän ongelmalle ja sen ratkaisulle.

**Määritelmä 1.** *Ongelma* on kuvaus  $P : S \rightarrow R$  missä  $S$  on ongelman syöteavaruus ja  $R$  odotettujen tulosten avaruus.

**Määritelmä 2.** Kone  $M$  ratkaisee ongelman  $\Leftrightarrow \forall s \in S : M(s) = P(s)$

Ongelman  $P$  sanotaan olevan ratkaistavissa konetyypillä  $\hat{M}$ , mikäli on olemassa sellainen  $M \in \hat{M}$  joka ratkaisee  $P$ :n. Mikäli ongelma  $P$  on ratkaistavissa jollakin konetyypillä, kutsutaan sitä laskettavissa olevaksi. On tärkeää huomata, että ongelma on ratkaistavissa mikäli voidaan osoittaa, että sille on olemassa ratkaisu, vaikka ratkaisua ei suoraan tunnettaisikaan. Kaikille ongelmille ei luonnollisesti löydy ratkaisualgoritmia.

## 3 Muodolliset kielet

Muodolliset kielet liittyvät läheisesti sellaisiin muodollisiin koneisiin, joiden syöte on merkkijono ja jotka joko hyväksyvät tai hylkäävät saamansa syöteen. Tällaisia ovat esimerkiksi äärelliset automaatit ja Turingin koneet. Olkoon  $\Sigma$  äärellinen aakkosto.  $\Sigma$ :n alkioiden äärellisiä jonoja kutsutaan sanoiksi. Tyhjästä sanastaa käytetään merkintää  $\epsilon$ . Kaikkien sanojen joukosta käytetään termiä  $\Sigma^*$ . Muodollinen kieli  $L$  on  $\Sigma^*$ :n osajoukko.  $L$ :n alkoita kutsutaan sanoiksi.  $L$ :n ajatellaan usein määräytyvän joistakin säännöistä, mutta tämä ei ole välttämätöntä.

Olkoon  $M : \Sigma^* \rightarrow \{\text{'kyllä'}, \text{'ei'}\}$  muodollinen kone ja olkoon  $w \in L$ .  $M(w)$  merkitsee koneen  $M$  antamaa tulosta syötteellä  $w$ . Kone  $M$  tunnistaa kielen  $L(M) = \{w \in \Sigma^* | M(w) = \text{'kyllä'}\}$ .

## 4 Automaatit

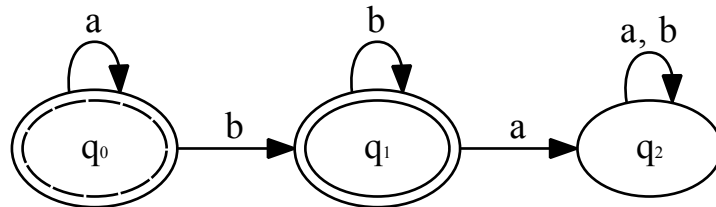
Automaatit ovat yksinkertaisia muodollisia koneita. Ne joko hyväksyvät tai hylkäävät annetun äärellisen merkkijonon. Muodollisesti määriteltynä automaatti  $M$  on viisikko  $(Q, \Sigma, \delta, q_0, F)$ , jonka osat ovat

- äärellinen tilajoukko  $Q$
- äärellinen aakkosto  $\Sigma$
- siirtymäfunktio  $\delta : Q \times \Sigma \rightarrow Q$
- alkutila  $q_0 \in Q$
- hyväksyvien tilojen joukko  $F \subset Q$ .

Automaatti on aina jossakin tilassa  $q \in Q$ , alkutilanteessa  $q = q_0$ . Se lukee syötteestä yhden symbolin kerrallaan, olkoon tämä symboli  $\sigma \in \Sigma$ . Koneen nykyisen tilan ja luetun merkin perusteella se siirtyy tilaan  $\delta(q, \sigma)$ , ja siirtyy lukemaan seuraavaa symbolia. Kun koko syöte on käyty läpi tarkistetaan, onko automaatin sen hetkinen tila joukossa  $F$ . Jos näin on, palautetaan vastaukseksi 'kyllä', muutoin 'ei'.

Tällä tavoin määritelty muodollinen kone on monilta ominaisuuksiltaan käyttökelpoinen; sen suoritus kestää aina lineaarisen ajan, ja sen suorittamiseen tarvittavan muistin määrä on vakio.

**Esimerkki 3.**  $M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_0, q_1\})$  jossa siirtymäfunktio  $\delta$  kuvataan seuraavalla graafilla:



Kuva 1: Automaatin tilakaavio

Siirtymäfunktio voidaan myös kuvata taulukkona:

	$q_0$	$q_1$	$q_2$
$a$	$q_0$	$q_2$	$q_2$
$b$	$q_1$	$q_1$	$q_2$

Tämä kone hyväksyy vain merkkijonot, jotka ovat muotoa  $a^n b^m$ ,  $n, m \in \mathbb{N}$ .

Jos syötteenä annetaan esimerkiksi merkkijono aaabb, suoritus etenisi seuraavasti:

Tila	Merkki	Jäljellä oleva syöte
$q_0$	a	aabb
$q_0$	a	abb
$q_0$	a	bb
$q_0$	b	b
$q_1$	b	
$q_1$		

Koska  $q_1 \in F$ , kone antaa tulokseksi 'kyllä'.

Toisaalta jos syötteenä olisi aba, suoritus etenisikin seuraavasti:

Tila	Merkki	Jäljellä oleva syöte
$q_0$	a	ba
$q_0$	b	a
$q_1$	a	
$q_2$		

Koska  $q_2 \notin F$ , kone antaisi tulokseksi 'ei'.

**Lause 4.** *Olkoon  $M$  esimerkissä 3 määritelty automaatti.*

$$L(M) = \{a^n b^m \mid n, m \in \mathbb{N}\}.$$

*Todistus.* Olkoon sana  $w \in \Sigma^*$ . Jos  $w \in L(M)$ , niin automaatti selvästi palauttaa tuloksen 'kyllä'. Jos  $w \notin L(M)$ , niin sen täytyy sisältää ainakin yksi  $ba$ -pari. Tämä riittää siirtämään automaatin mistä hyvänsä tilasta tilaan  $q_2$ . Mikään merkki ei siirrä automaattia tilasta  $q_2$  mihinkään muuhun tilaan, joten suorituksen päätyttyä kone on edelleen tilassa  $q_2$ , ja palauttaa siis tulokseksi 'ei'.  $\square$

Äärellisillä automaateilla on kuitenkin monia rajoitteita. On runsaasti esimerkkejä ongelmista, jotka intuitiivisesti olisivat ratkaistavissa jollakin algoritmilla, mutta joita ei voi laskea äärellisen automaatin avulla. Tällaisia ongelmia ovat esimerkiksi 'Onko annettu suljelauseke oikein muodostettu?' tai 'Onko merkkijono muotoa  $a^n b^n$ ?'. Onkin selvää, että äärelliset automaatit eivät ole riittävän yleisiä toteuttaakseen kaikkia algoritmeja.

**Lause 5.** *Ei ole olemassa äärellistä automaattia  $M$ , joka tunnistaa kielen  $L = \{a^n b^n \mid n \geq 0\}$ .*

*Todistus.* Olkoon  $M$  äärellinen automaatti, jolla on  $n$  tilaa. Mikä tahansa merkkijono, jonka pituus on vähintään  $n$  aiheuttaa sen, että jokin koneen tiloista toistetaan. Merkitään tätä tilaa  $S$ . Siirtymä tämän tilan ensimmäisestä esiintymästä toiseen vastaa jotakin käsitellyn merkkijonon osaa. Olkoon tämä

osa  $y$ . Jos kone on tilassa  $S$  ja se saa syötteekseen  $y:n$ , päättyy se taas tilaan  $S$ . Tästä seuraa, että jos automaatti hyväksyy sanan  $w = xyz$ , täytyy sen hyväksyä sana  $xy^iz, i \geq 0$ .

Sovelletaan tätä nyt kielen  $L$  tapaukseen. Olkoon  $w = a^m b^m, m > n$ . Tällöin on olemassa sellainen jako  $i+j+k = m, j > 0$ , että kone  $M$  tunnistaa sanat  $a^i a^{j \cdot p} a^k b^m, p \in \mathbb{N}$ . Näistä kuitenkin vain yksi kuuluu kieleen  $L$ , joten  $L(M) \neq L$ .  $\square$

## 5 Rekisterikoneet

Shepherdsonin ja Sturgisin rekisterikoneet ovat rakenteellisesti hyvin yksinkertaisia. Koneella on  $R$  rekisteriä, joista kukin muistaa yhden luonnollisen luvun. Rekisterin  $n$  arvoon viitataan merkinnällä  $\bar{n}$ . Suoritettava algoritmi esitetään numeroituna luettelona, joka koostuu hyvin yksinkertaisista komennoista. Merkitään näitä komentoja  $\hat{1}, \hat{2}, \dots, \hat{M}$ . Algoritmin alkuarvot ovat tallennettu rekistereihin. Rekisterikone tunnistaa kolme erilaista komentoa:  $(n, j)$ ,  $(n, j, k)$  sekä lopetuskomennon  $\hat{M}$ . Kahden ensimmäisen komennon merkitykset ovat:

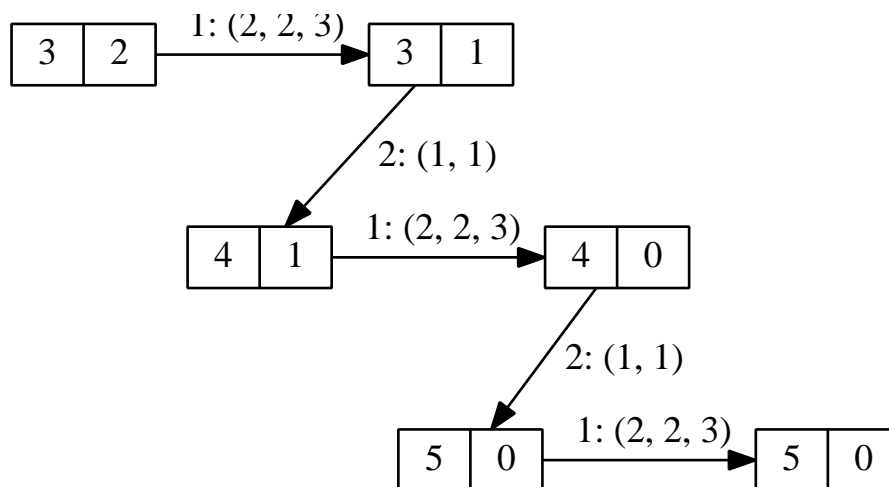
- $(n, j)$     Lisää 1 rekisteriin  $n$  ja siirry komentoon  $j$ .
- $(n, j, k)$     Jos  $\bar{n} > 0$ , vähennä 1 rekisteristä  $n$  ja siirry komentoon  $j$ ;  
muutoin siirry komentoon  $k$ .

Näillä sinänsä yksinkertaisilla säännöillä on mahdollista suorittaa monimutkaisiakin laskutoimituksia.

**Esimerkki 6.** Määritellään rekisterikone joka suorittaa yhteenlaskun.

$R = 2, M = 3$   
 $\hat{1}(2, 2, 3)$   
 $\hat{2}(1, 1)$   
 $\hat{3}\text{Seis}$

Olkoon rekisterien arvot alkutilanteessa  $(3, 2)$ . Ohjelman suoritus etenisi seuraavalla tavalla:



Kuva 2: Esimerkki rekisterikoneen toiminnasta

## 6 Turingin koneet

Turing kone on ensimmäinen tunnettu muodollinen kone. Alan Turing esitteli sen vuonna 1937, kun käytti sitä apuvälineenä ratkaistakseen David Hilbertin vuonna 1928 esittämän päätösongelman. Turingin koneet poikkeavat rakenteellisesti rekisterikoneista huomattavasti ja muistuttavat enemmänkin äärellisiä tilakoneita. Muodollisesti määriteltynä Turingin koneella on seitsemän komponenttia  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , jossa

- äärellinen tilajoukko  $Q$
- nauha-aakkosto  $\Gamma$
- syöteaakkosto  $\Sigma \subset \Gamma$
- siirtymäfunktio  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
- alkutila  $q_0$
- 'tyhjä' nauhasymboli  $B \in \Gamma - \Sigma$
- hyväksyvien tilojen joukko  $F \subseteq Q$ .

Turingin koneiden ajatellaan käsittelevän kaksisuuntaisesti ääretöntä nauhaa, jonka jollekin äärelliselle alueelle on kirjoitettu symboleja. Muu nauha sisältää vain symbolia  $B$  ('blank' eli tyhjä). Turingin koneet toimivat kuten

äärelliset automaattit lukuun ottamatta kahta merkittävää eroa. Ne voivat siirtyä nauhaa sekä eteen- että taaksepäin, ja ne voivat muuttaa nauhan sisältöä.

Alkutilanteessa kone aloittaa syötteen vasemmasta reunasta, tilassa  $q_0 \in Q$ . Kone lukee yhden merkin ja päättää sen perusteella mihin tilaan siirtyä, minkä merkin kirjoittaa nauhalle luetun merkin tilalle ja siirtyäkö oikealle vai vasemmalle. Suoritus päättyy vasta, kun siirtymäfunktio  $\delta$  palauttaa arvon  $\emptyset$ . Sen jälkeen verrataan senhetkistä tilaa joukkoon  $F$  kuten äärellisten automaattien tapauksessa.

**Esimerkki 7.** Olkoon  $M$  Turingin kone siten, että  $M = (\{q_0, q_a, q_b, q_r, q_e\}, \{a, b\}, \{a, b, B\}, \delta, q_0, B, \{s\})$  missä  $\delta$  toimii seuraavan taulukon mukaisesti:

	a	b	B
$q_0$	$(q_a, B, R)$	$(q_e, b, R)$	$\emptyset$
$q_a$	$(q_a, a, R)$	$(q_a, b, R)$	$(q_b, B, L)$
$q_b$	$(q_e, a, R)$	$(q_r, B, L)$	$(q_e, B, R)$
$q_r$	$(q_r, a, L)$	$(q_r, b, L)$	$(q_0, B, R)$
$q_e$	$\emptyset$	$\emptyset$	$\emptyset$

Tällä tavoin määritelty Turingin kone tunnistaa kielen  $\{a^n b^n | n \geq 0\}$ , jota äärelliset automaattit eivät pysty tunnistamaan.

## 7 Funktioiden esittäminen muodollisella koneella

Useimmissa tapauksissa ei riitä, että algoritmi tuottaa kyllä/ei -vastauksen, vaan algoritmin toivotaan tuottavan esimerkiksi laskutoimituksen tuloksen. Näissä tapauksissa ei ole tärkeää se, missä tilassa kone on lopettaessaan suorituksen, vaan se, mitä rekistereissä tai nauhalla on. Lopetustilan tarkastelu ei tällaisissa tapauksissa ole tarpeellista, vaan voimme pitää kaikkia lopetus-tiloja hyväksyttävinä.

Se, miten algoritmi palauttaa vastauksen, on sopimuskysymys. Turingin koneen tapauksessa vastaus luetaan nauhalta suorituksen päätyttyä, ja rekisterikoneen tapauksessa tulos tallennetaan yleensä rekisteriin yksi.



## 8 Rekisterikoneiden ja Turingin koneiden välinen yhteys

Vaikka Turingin koneet ja rekisterikoneet ovat rakenteellisesti hyvin erilaisia, ovat ne kuitenkin osoittautuneet laskukyvyltään ekvivalenteiksi. Jos ongelma on ratkaistavissa rekisterikoneella, se on myös ratkaistavissa Turingin koneella ja päinvastoin. Se, että näinkin erilaiset koneet ovat yhtä ilmaisuvoimaisia, viittaa siihen, että olemme lähestymässä käyttökelpoista tapaa määritellä laskettavuuden käsite. Laskettavuus samaistetaan usein laskettavuuteen Turingin koneilla.

**Määritelmä 8.** Olkoon sana  $w = \Sigma^*$  jonka pituus on  $n$ . Olkoon  $w_i$  sanan  $i$ :des merkki, ja olkoon kuvaus  $f : \Sigma \rightarrow \mathbb{N}$  injektio. Sanan  $w$  *alkulukukoodaus* on bijektio  $p : \Sigma^* \rightarrow \mathbb{N}$  missä

$$p(w) = \prod_{i=1}^n p_i^{f(w_i)}$$

ja  $p_i$  on  $i$ :s alkuluku.

Kuvauksen  $f$  valinta ei ole tärkeää, kunhan samassa yhteydessä käytetään aina samaa kuvausta. Mikäli kuvausta ei erikseen määritellä, oletetaan yleensä sen kuvaavan aakkoston positiivisiin kokonaislukuihin samassa järjestyksessä missä merkit lueteltiin.

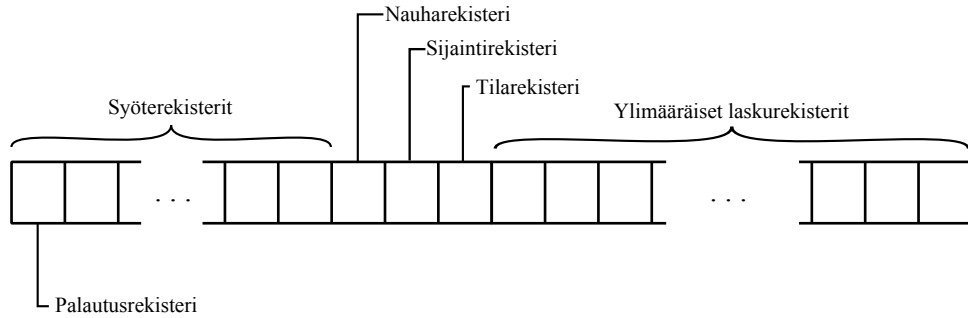
**Esimerkki 9.** Olkoon  $\Sigma = a, b, c$  ja  $w = abac$ .  $w$ :n alkulukukoodaus olisi tässä tapauksessa  $p(w) = p_1^{f(a)} \times p_2^{f(b)} \times p_3^{f(a)} \times p_4^{f(c)} = 2^1 \times 3^2 \times 5^1 \times 7^3 = 30870$ .

**Lause 10.** *Olkoon  $f$   $k$ :n muuttujan funktio. Funktio  $f$  on laskettavissa rekisterikoneella jos ja vain jos se on laskettavissa Turingin koneella.*

*Todistus.* Osoitetaan ensin, että annetun Turingin koneen  $M$  toimintaa voidaan simuloida sopivasti rakennetulla rekisterikoneella. Jos tämä pitää paikkansa, mikä tahansa Turingin koneella ratkaistavissa oleva ongelma voidaan ratkaista myös rekisterikoneella tarvittaessa simuloimalla ko. Turingin koneen toimintaa. Tämän jälkeen osoitetaan, että sama pätee myös toiseen suuntaan.

Olkoon  $M$  Turingin kone.  $M$ :n tämän hetkinen konfiguraatio koostuu nauhan sisällöstä, tällä hetkellä tutkittavasta symbolista sekä koneen tilasta. Vaikka nauha on ääretön, voimme mallintaa sitä äärellisenä, sillä vain äärellisen moni merkki on epätyhjä. Nauha-aakkosto on äärellinen, joten nauhan sen hetkinen tila voidaan ilmaista yhtenä lukuna alkulukukoodauksella.

Tallennetaan tämä ensimmäiseen rekisteriin, jota ei käytetä syötteen vastaanottamiseen. Seuraavaan rekisteriin tallennetaan tutkittavan merkin sijaintia vastaava luonnollinen luku, ja sitä seuraavaan koneen tämän hetkistä tilaa vastaava luku.

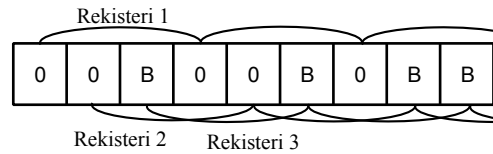


Kuva 3: Rekistereiden valinta

Tämän jälkeen voidaan kuvailla rekisterikoneen varsinainen toiminta. Jokaista paria  $(q_i, a)$ ,  $q_i \in Q, a \in \Gamma$  kohden on komentosarja, joka muokkaa rekistereitä niin, että se vastaa sitä mitä koneessa  $M$  olisi tapahtunut. Jos  $\delta(q_i, a) = \emptyset$  koneen tulisi pysähtyä, mutta tässä tapauksessa tulee ensin purkaa ja kopioida nauharekisterin sisältö tulosrekisteriin. Jos  $\delta(q_i, a) = (q_j, b, L)$  tai  $(q_j, b, R)$  tulee kirjoittaa komentosarja joka muokkaa kolmea tilarekisteriä koneen  $M$  tavoin. Tällaisten komentosarjojen tuottaminen täsmällisesti on varsin työlästä, mutta ei kovin vaikeaa.

Näiden lisäksi joudumme myös määrittelemään tällaisen koneen aloitustilan. Tämä tapahtuu asettamalla syötenauhaa vastaava luku nauharekisteriin, 1 käsiteltävän merkin numeroksi sekä 0 koneen tilaksi.

Käänteisesti tulee rakentaa sellainen Turingin kone, joka pystyy simuloimaan annetun rekisterikoneen toimintaa. Olkoon rekisterikoneella  $R$  rekisteriä. Tällöin tallennetaan yhden rekisterin sisältö koodattuna joka  $R$ :nteen muistipaikkaan valitusta rajasta eteenpäin. Esimerkiksi jos  $R = 2$ , joka toinen nauhan muistipaikka kuuluisi rekisteriin 1 ja joka toinen rekisteriin 2. Rekisterien sisältö voidaan koodata nauhalle monin eri tavoin. Yksinkertaisuuden vuoksi käytetään tässä unaarista lukujärjestelmää.



Kuva 4: Rekisterien sijoittelu nauhalle tapauksessa  $R = 3$ . Rekistereissä ovat kuvattuna arvot  $(3, 2, 0)$ , nauha on muualla tyhjä.

Ensimmäiseksi annettu syöte tulee muuntaa edellä kuvattuun muotoon. Tarvittavan aliohjelman teko on helppoa.

Sen jälkeen kirjoitetaan aliohjelmat, jotka suorittavat samat toiminnot kuin  $(n, j)$  ja  $(n, j, k)$ . Kannattanee pitää tarkoitukseen varattua merkkiä rekisterialueen vasemmalla puolen, jotta se löytyy helpommin. Tässä tarkoitukseen käytetään merkkiä 1. Olkoon Turingin koneen lukupää tällä merkillä seuraavia aliohjelmia varten, ja olkoon se komentoa  $(n, j)$  vastaavassa tilassa. Aluksi kone siirtyy  $n - 1$  merkkiä oikealle, jolloin se saavuttaa rekisteriä  $n$  vastaavan alueen. Tämän jälkeen se siirtyy oikealle  $R$ :n pituisin askelin, kunnes se löytää tyhjän merkin. Kone asettaa sen 0:ksi, palaa takaisin vasemmalle merkin 1 luokse ja muuttaa tilansa rekisterikoneen komentoa  $\hat{j}$  vastaavaksi. Tällä tavoin rekisterin  $n$  arvoa on kasvatettu yhdellä. Olkoon kone nyt komentoa  $(n, j, k)$  vastaavassa tilassa. Kone siirtyy ensin  $n - 1$  merkkiä oikealle. Jos sillä kohdalla oleva merkki on tyhjä, palataan kohtaan 1 ja siirrytään rekisterikoneen komentoa  $\hat{k}$  vastaavaan tilaan. Jos merkki ei ollut tyhjä, siirrymme oikealle  $R$ :n pituisin askelin kunnes löydämme tyhjän merkin. Siirrymme  $R$  merkkiä vasemmalle, merkitsemme sen tyhjäksi, siirrymme takaisin merkin 1 luo ja siirrymme komentoa  $\hat{j}$  vastaavaan tilaan.

Kun komento 'Seis' saavutetaan, ensimmäisen rekisterin sisältö talletetaan ja muu nauha tyhjennetään.

□

Tässä todistuksessa havaitaan myös, että mikäli algoritmin suoritus yhdellä koneella ei koskaan pääty, myöskään toisella koneella simuloitu suoritus ei koskaan pääty.

## A Lähteet

J. Truss: Discrete Mathematics For Computer Scientists, Addison-Wesley, 1999.