Muodolliset koneet ja laskettavuus

Juuso Valli

21. 4. 2013

Tiivistelmä

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

Sisältö

1 Laskettavuus

Eräs algoritmitutkimuksen peruskysymys on selvittää, mihin ongelmiin on mahdollista löytää ratkaisu algoritmisesti. Tämän selvittämiseksi on toisaalta tutkittava, onko sellaista ratkaisualgoritmia ylipäänsä olemassa, ja toisaalta onko sitä mahdollista suorittaa käytännössä. Ratkaisualgoritmin ole-

massaoloa ja käyttökelpoisuutta on kuitenkin vaikea osoittaa ilman että määritellään täsmällisesti, mitä algoritmilla tarkoitetaan.

Epämuodollisesti määriteltynä algoritmi on äärellinen luettelo komentoja, joita noudattamalla saadaan ratkaisu annettuun ongelmaan. Yleensä algoritmin määritelmään liittyy myös ne vaatimukset, että algoritmin suorituksen tulee päättyä äärellisen ajan kuluttua ja että sen tulee tuottaa samalle syötteelle sama tulos joka ajokerralla. Tämä määritelmä on kuitenkin hyvin tulkinnanvarainen esimerkiksi siksi, että siitä ei käy ilmi, mitä tarkoitetaan komennolla. Tämän vuoksi mahdollisten komentojen joukkoa pyritäänkin rajoittamaan äärelliseksi useimmiten muodollisten koneiden avulla. On tärkeää huomata, että vaikka mahdollisten komentojen joukkoa pienennetään, monimutkaisemmat komennot voidaan toteuttaa yhdistelemällä tämän rajatun joukon komentoja. Tavoitteena olisi löytää mahdollisimman pieni komentokanta, jonka avulla voisi toteuttaa kaikki muut komennot. Lienee mahdotonta osoittaa että tällainen kanta olisi optimaalinen, mutta muutamia hyviä ehdokkaita kuitenkin tunnetaan.

Kaikille ongelmille ei luonnollisesti löydy ratkaisualgoritmia. Mikäli sellainen algoritmi kuitenkin tunnetaan, tai voidaan osoittaa että sellainen on olemassa, ongelma katsotaan laskettavissa olevaksi. Koska epämuodollisen algoritmin olemassaolosta on vaikea osoittaa mitään suuntaan tai toiseen, tarkastellaankin usein vain tietyn tyyppisellä muodollisella koneella toteutettuja algoritmeja.

Teoreettisesti toimivan algoritmin toteuttamiskelpoisuutta tarkastellaan muun muassa tutkimalla sen suoritusaikaa syötteen pituuden funktiona. Algoritmia pidetään yleisesti toteutettavana, jos sen suoritusajan yläraja voidaan esittää syötteen polynomifunktiona. Tällaista algoritmia kutsutaan polynomiaikaiseksi. Muita kriteerejä on esimerkiksi algoritmin tarvitseman muistin määrä.

Muodollisilla koneilla ei ole kaikenkattavaa määritelmää, vaan jokainen määritellään eri tavoin. Seuraavassa osiossa tutustutaan muutamiin erityyppisiin muodollisiin koneisiin ja tutkitaan niiden välisiä yhteyksiä.

2 Muodolliset kielet

Muodolliset kielet liittyvät läheisesti sellaisiin muodollisiin koneisiin, jotka käsittelevät merkkijonoja. Tällaisia ovat esimerkiksi äärelliset automaatit ja Turingin koneet. Olkoon Σ äärellinen aakkosto. Muodollinen kieli L on Σ^*:n osajoukko. L:n alkoita kutsutaan sanoiksi. L:n ajatellaan usein määräytyvän joistakin säännöistä, mutta tämä ei ole välttämätöntä.

Olkoon M muodollinen kone ja olkoon $w \in L$. $M(w)$ merkitsee koneen

M antamaa tulosta syötteellä w. Kone M tunnistaa kielen $L(M) = \{w \in \Sigma^* | M(w) =' \text{kyllä}'\}$.

3 Automaatit

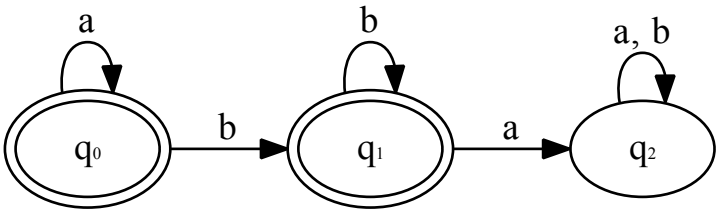
Automaatit ovat yksinkertaisia muodollisia koneita. Ne joko hyväksyvät tai hylkäävät annetun äärillisen merkkijonon. Muodollisesti määriteltynä automaatti M on viisikko $(Q, \Sigma, \delta, q_0, F)$, jonka osat ovat

- Äärellinen tilajoukko (Q)
- Äärellinen aakkosto (Σ)
- Siirtymäfunktio $(\delta : Q \times \Sigma \to Q)$
- 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; se suoritus kestää aina lineaarisen ajan, ja sen suorittamiseen tarvittavan muistin määrä on vakio.

Esimerkki 1. $M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_0, q_1\})$ *jossa siirtymäfunktio* δ *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 syötteeksi 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'.

Todistus. Olkoon sana $w \in \Sigma^*$. Olkoon $L \subset \Sigma^*$ siten että L sisältää täsmälleen muotoa a^*b^* olevat sanat. Jos $w \in L$, niin automaatti selvästi palauttaa tuloksen 'kyllä'. Jos $w \notin L$, 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'. □

Äärellisillä automaateilla on kuitenkin monia rajoitteita. On runsaasti esimerkkejä ongelmista, jotka intuitiivisesti olisivat ratkaistavissa, 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 kaikki algoritmit.

Teoreema 1. *Ei ole olemassa äärellistä automaattia M, joka tunnistaa kielen $L = \{a^n b^n | n \geq 0\}$.*

Todistus. Olkoon M äärellinen automaatti, jolla on n tilaa. Mikä tahansa merkkijono, jonka pituus on vähintään kuin n aiheuttaa sen, että jokin koneen tiloista toistetaan. Merkitään tätä tilaa S. Siirtymä tämän tilan ensimmäisestä ilmentymästä toiseen vastaa jotakin käsitellyn merkkijonon osaa. Olkoon tämä osa y. Jos kone on tilassa S ja se saa syötteekseen w:n, päätyy se taas tilaan S. Tästä seuraa, että jos automaatti hyväksyy sanan $w = xyz$, täytyy sen hyväksyä sana $xy^i z, i \geq 0$.

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 $a^i a^{j*p} a^k b^m, p \in \mathbb{N}$. Näistä kuitenkin vain yksi kuuluu kieleen L, joten $L(M) \neq L$. □

4 Rekisterikoneet

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

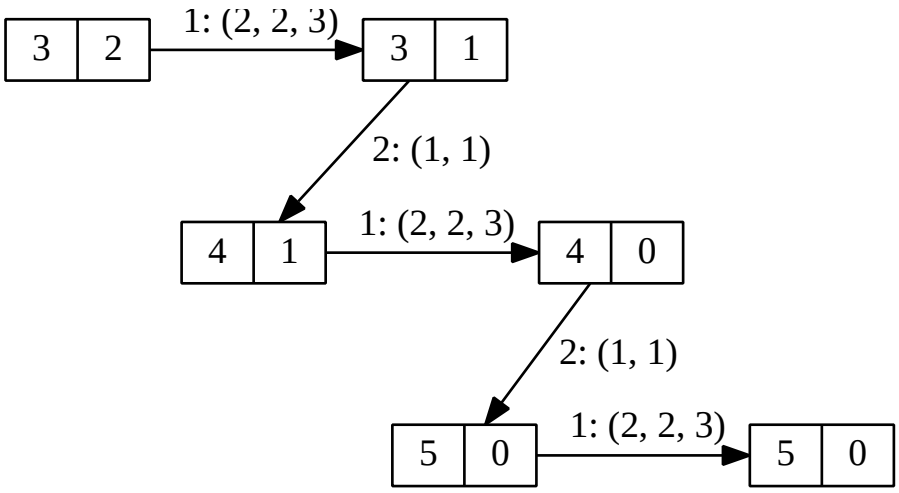
(n, j)	Lisää 1 rekisteriin n ja siirry komentoon j.
(n, j, k)	Jos $\bar{n} > 0$, vähennä 1 rekisteristä n ja siirry komentoon j; muutoin siirry komentoon k.

Näillä sinänsä yksinkertaisilla säännöillä on mahdollista suorittaa monimutkaisiakin laskutoimituksia.

Esimerkki 2. *Määritellään rekisterikone joka suorittaa yhteenlaskun.*

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

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



Kuva 2: Esimerkki rekisterikoneen toiminnasta

5 Turingin koneet

Turingin koneet ovat 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 (Γ)
- Syöteaakkosto ($\Sigma \subset \Gamma$)
- Siirtymäfunktio ($\delta : Q \times \Gamma \to Q \times \Gamma \times L, R \cup \emptyset$)
- 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 automaatit 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 δ palauttaa arvon $\emptyset$. Sen jälkeen verrataan senhetkistä tilaa joukkoon F kuten äärellisten automaattien tapauksessa.

Esimerkki 3. *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 δ 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 automaatit eivät pysty tunnistamaan.

6 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 tarkestelu ei tällaisissa tapauksissa ole tarpeellista, vaan voimme pitää kaikkia lopetustiloja hyväksyttävinä.

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

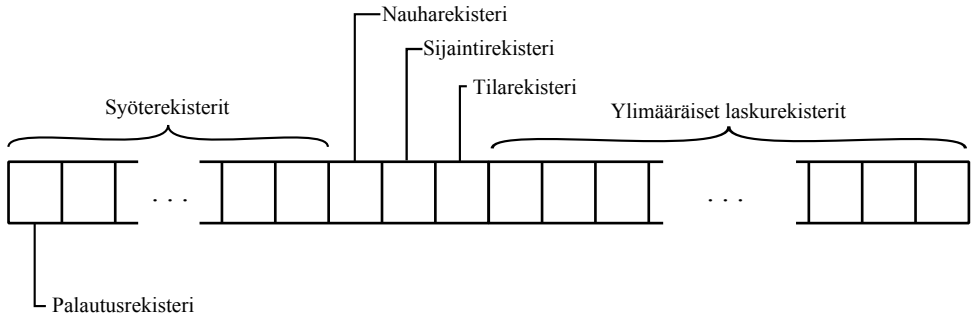
7 Rekisterikoneiden ja Turingin koneiden välinen yhteys

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 samaistetaankin usein laskettavuuteen Turingin koneilla.

Teoreema 2. *Olkoon f k:n muuttujan 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.

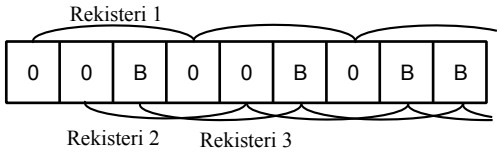


Kuva 3: Rekistereiden valinta

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 sisällön koodattuna joka R:nteen muistipaikkaan valitusta rajasta eteenpäin. Esimerkiksi jos $R = 2$, joka toinen nauhan muistipaikka kuuluisi rekisteriin 1 ja joka toinen rekisteriin 2. Yksinkertaisuuden vuoksi tässä käytetään additiivista koodausta, vaikka sen voisi tallettaa tehokkaamminkin, esimerkiksi multiplikatiivisesti.



Kuva 4: Rekisterien sijoittelu nauhalle tapauksessa R = 3. $(3, 2, 0)$

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 saavutta 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 $\hat{\text{j}}$ 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, 1999