Tietojenkäsittelytiede **27** Joulukuu 2007 sivut 18–46
Toimittajat: Antti Valmari

Sanat ja automaatit*

Juhani Karhumäki
Matematiikan laitos ja Turun Tietotekniikan Keskus TUCS
Turun yliopisto
karhumak@utu.fi

Tiivistelmä

Automaatit muodostavat tietojenkäsittelytieteen perusteiden kulmakiven. Sanat, siis merkkijonot, puolestaan ovat automaattien, ja laajemminkin kaiken informaation käsittelyn, keskeisin peruskäsite. Näihin liittyvä tutkimus – automaattien teoria ja sanojen kombinatoriikka – ovat nykyisin laaja-alaisia läheisesti toisiinsa ja moniin muihin aloihin liittyviä tutkimuskohteita. Tämän artikkelin tarkoituksena on esitellä näiden tutkimusalojen kehitystä ja nykytilaa, painottaen – luonnollisesti – kirjoittajan henkilökohtaisia näkemyksiä.

1 Johdanto

Käytännön elämässä *automaatilla* tarkoitetaan laitetta, joka saatuaan tietyn syötteen antaa (yleensä) halutun tulosteen. Myös nykyaikaiset tietokoneet voidaan ajatella automaatteina tässä mielessä, jolloin siis koneen ohjelmisto tulee ajatella osana syötettä. Matemaattisesti automaatti on tämän ajatuksen abstraktio, täsmällisesti määritelty suure, joka pystyy suorittamaan tiettyjä laskuja, siis liittämään annettuun syötteeseen yksikäsitteisen tulosteen. Tavallisesti automaatista edellytetään, että se on äärellinen.

Matemaattinen automaatti on siis abstrakti laskemisen malli. *Automaattien teorian* keskeinen kysymys on, mitä ongelmia eri tyyppisillä automaateilla voidaan ratkaista tai laskea. Esimerkiksi, mitä voidaan laskea, jos käytössä on ainoastaan kiinteä äärellinen muisti? Tätä tutkii äärellisten automaattien teoria.

Erilaisia automaattimalleja on tutkittu sangen intensiivisesti 50-luvun lopulta lähtien, siis noin 50 vuotta. Itse asiassa, se mitä voidaan periaatteessa algoritmisesti laskea selvitettiin jo aikaisemmin – jo ennen ensimmäisten tietokoneiden rakentamista – itävaltalaisen K. Gödelin ja englantilaisen A. Turingin sekä heidän aikalaistensa toimesta. Teoreetikot olivat, kuten usein, aikaansa edellä. Osoittautui, että kaikki, mikä on algoritmisesti laskettavissa, siis tietokoneella toteutettavissa, voidaan ratkaista hyvin yksinkertaisella matemaattisella mallilla, niin kutsutulla *Turingin koneella*. Kone siis on yksinkertainen, mutta sen laskut voivat olla äärimmäisen monimutkaisia.

Kuluneiden vuosikymmenien aikana tietämys erilaisten automaattityyppien laskentakyvystä on täsmentynyt, mutta

*Artikkeli on alunperin kirjoitettu vuonna 2002 Tietojenkäsittelytieteen seuran toteutumatta jäänyttä 20-vuotisjuhlakirjaa varten.

OHJE KIRJAPAINOLLE: B5-arkin vasen- ja yläreuna kohdistetaan A4-arkin vasempaan ja yläreunaan. Näin pitäisi marginaaliksi tulla taitteen puolella noin 33 mm ja muualla noin 22 mm.

yhä vielä muutamat teorian syntyaikoina esitetyt kysymykset ovat vastausta vailla. Itse asiassa, tietyistä ongelmista tiedetään yhä kovin vähän, tai oikeastaan ei juuri mitään. Tällainen on muun muassa tunnettu $\mathbf{P} = \mathbf{NP}$ -probleema. Se mainitaan yhtenä seitsemästä tärkeimmästä 2000-luvun matematiikan ongelmasta (millennium probleemasta). Näiden kunkin ratkaisusta on luvattu miljoonan dollarin palkinto.

Automaatit, kuten myös tietokoneet, operoivat merkkijonoilla, eli matemaattisesti *sanoilla*. Itse asiassa silloinkin, kun tietokone laskee luvuilla, nämä esitetään sille merkkijonoina, yleensä binäärisinä bittijonoina. Luku 131 on tietokoneelle jono 10000011 ja luku 10^{80}, joka on kertaluvultaan maailmankaikkeuden atomien lukumäärää vastaava, on tietokoneelle vain 266 mittainen bittijono. Tämä selittää osaltaan miksi tietokoneilla voidaan operoida käsittämättömän suurilla luvuilla.

Sanoihin voidaan liittää monenlaisia matemaattisia ongelmia. Voidaan kysyä, miten muodostaa tehokkaasti äärettömiä sanoja, tai montako n:n mittaista keskenään erilaista osasanaa annettu ääretön sana sisältää, tai onko sana lopulta jaksollinen. Jaksollisuuskysymykset ovat keskeisiä, ja sovellusten kannalta tärkeitä, sanojen ongelmia: Milloin kaksi sanaa kommutoi? Sisältääkö annettu sana osasanana neliöitä, siis sanan toistoa, tai kuinka korkeaa astetta olevan toiston annettu sana sisältää osasanana? Nämä ovat esimerkkejä *sanojen kombinatoriikan* tutkimusongelmista.

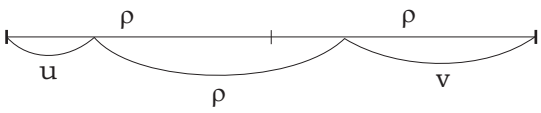
Tämä työ esittelee joitakin automaatteihin ja sanoihin liittyviä ongelmia sekä saavutettuja tuloksia. Valitut ongelmat ovat historiallisesti keskeisiä, mutta eivät pyri olemaan millään muotoa kattavia. Luvussa 2 esitetään tarvittavat määritelmät ja joitakin esimerkkejä lukijan motivoimiseksi. Luvussa 3 luodaan katsaus automaattien teorian saavutuksiin. Luvussa 4 esitellään automaattien teorian joitakin sovelluksia. Lopulta luvussa 5 tarkastellaan tyypillisiä sanojen kombinatoriikan ongelmia ja saavutuksia sekä niiden merkitystä esimerkiksi algoritmitutkimukselle.

2 Peruskäsitteet ja esimerkkejä

Tässä luvussa esitämme tarvittavat määritelmät ja käsitteet sekä esimerkkejä. Lisätietoja automaateista löytyy kirjoista [38], [36] ja [74], sekä sanojen kombinatoriikasta lähteistä [58], [59] ja [14].

Olkoon A äärellinen joukko, jota sanomme *aakkostoksi*, esimerkiksi $A = \{a, b\}$. Aakkoston A *sanalla* tarkoitamme mielivaltaisen mittaista A:n alkioiden jonoa. Sana voi olla myös (vasemmalta oikealle) ääretön. Sanan symbolien lukumäärä on sanan pituus. Jonoa, jonka pituus on nolla, kutsutaan *tyhjäksi sanaksi*. Siitä käytetään merkintää 1, tai usein myös ε. Kaikkien aakkoston A äärellisten sanojen joukkoa merkitään A^*:lla ja sovitaan, että $A^+ = A^* \setminus \{1\}$. Vastaavasti aakkoston A äärettömien sanojen joukkoa merkitään symbolilla A^ω. Sanojen *ketjuttaminen* eli *tulo* tekee joukosta A^* monoidin, toisin sanoen joukon jossa on määritelty assosiatiivinen binäärioperaatio ja joka sisältää neutraalialkion. Matemaattisesti A^+ on puoliryhmä. Edelleen, A^* ja A^+ ovat *vapaita*, tai tarkemmin sanoen aakkoston A *vapaasti generoimia*, toisin sanoen jokainen niiden alkio voidaan esittää yksikäsitteisesti aakkoston A alkioiden tulona. Siten sanojen teoria on matemaattisesti vapaiden puoliryhmien teoriaa.

Sana u on sanan w *osasana* (vastaa-



Kuva 1: Sanan esiintyminen neliönsä osasanana.

vasti *alkuosa*), jos on olemassa sanat x ja y niin, että $w = xuy$ (vastaavasti $w = uy$). Kutsumme sanaa w *neliöksi* tai *k:nneksi potenssiksi*, jos se voidaan kirjoittaa muodossa $w = u^2$ tai $w = u^k$, missä u^2 on uu, ja u^k on u toistettuna k kertaa. Käsite yleistyy luonnollisella tavalla murtopotensseille, esimerkiksi $abaabaab = (aba)^{2^2/3}$. Yllä mainittua lukua k sanassa w kutsutaan toiston u *kertaluvuksi*. Sana on *primitiivinen*, jos se ei ole minkään aidon osasanansa kokonaislukupotenssi. Edelleen sana w on *neliövapaa* tai *k-vapaa*, jos se ei sisällä osasanana neliötä tai k-toistoa, toisin sanoen sitä ei voida kirjoittaa muodossa $w = xu^2y$ tai $w = xu^ky$. Nämä käsitteet voidaan määritellä myös äärettömille sanoille.

Seuraavat esimerkit valaisevat sanoihin liittyviä probleemoja.

Esimerkki 1. Tarkastellaan puoliryhmän $\{a,b\}^*$ morfismeja

$$T : \begin{matrix} a \mapsto ab \\ b \mapsto ba \end{matrix} \qquad F : \begin{matrix} a \mapsto ab \\ b \mapsto a \end{matrix}.$$

Täten esimerkiksi a on $T(a)$:n alkuosa, $T(a)$ on $T^2(a)$:n alkuosa ja niin edelleen. Näinollen jono $(T^n(a))_{n\geq 0}$ suppenee kohden yksikäsitteistä ääretöntä sanaa

$$\alpha_T = \lim_{n\to\infty} T^n(a) = abbabaabbaababba\ldots.$$

Vastaavasti

$$\alpha_F = \lim_{n\to\infty} F^n(a) = abaababaabaab\ldots.$$

Yllä siis määrittelimme äärettömät sanat morfismin kiintopisteenä, toisin sanoen sanoina, jotka morfismi kuvaa itsekseen. Kyseiset sanat *Thue–Morse sana* α_T ja *Fibonaccin sana* α_F ovat tärkeitä esimerkkejä äärettömistä sanoista. Edellisellä on ominaisuus, että se ei sisällä osasanana kuutiota tai sanaa, joka olisi muotoa $uu\mathrm{pref}_1(u)$, missä $\mathrm{pref}_1(u)$ on sanan u ensimmäinen symboli. Tämän tärkeän tuloksen todisti Thue yli 100 vuotta sitten vuonna 1906. Fibonaccin sana puolestaan on niin sanotusti $(2+\varphi)^+$-vapaa, eli pitää sisällään mitä tahansa $(2+\varphi)$:tä pienempää lukua korkeampia toistoja, muttei $(2+\varphi)$:ta korkeampia toistoja. Tässä φ on kultaisen leikkauksen suhdeluku $\frac{1}{2}(1+\sqrt{5})$. Fibonaccin sana on sikäli merkittävä, että se on lähes universaali vastaesimerkki osoitettaessa sanoihin liittyviä tuloksia optimaalisiksi. Fibonaccin sanan nimellä on luonteva selitys: se voidaan määritellä myös rekursion

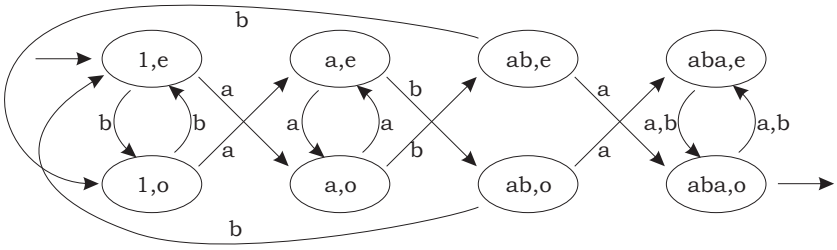
$$F_0 = a,\ F_1 = ab,\ F_{i+1} = F_iF_{i-1},\ i \geq 1$$

raja-arvona

$$\alpha_F = \lim_{i\to\infty} F_i.$$

Esimerkki 2. Olkoon ρ primitiivinen sana. Oletetaan edelleen, että ρ esiintyy sanan ρ^2 osasanana, toisin sanoen on olemassa sanat u ja v niin, että $\rho^2 = u\rho v$. Tilannetta voidaan havainnollistaa kuvalla 1. Tällöin helposti nähdään, että

$$uv = \rho = vu,$$



Kuva 2: Esimerkki äärellisestä automaatista.

mikä sanojen perustuloksen nojalla tarkoittaa, että u ja v ovat saman sanan potensseja. Näinollen, koska ρ on primitiivinen, ainoa mahdollisuus on, että $u = 1$ tai $v = 1$. Toisin sanoen primitiivinen sana ei voi esiintyä neliössään osasanana muuten kuin triviaalilla tavalla. Tämä tärkeä lemma on esimerkki merkkijonoalgoritmien yhteydessä käyttökelpoisista tuloksista.

Siirrymme seuraavaksi tarkastelemaan automaatteja. Niiden avulla määritellään tai hyväksytään *kieliä*, toisin sanoen sanajoukon A^* osajoukkoja. *Äärellinen automaatti* $\mathcal{A}$ aakkostossa A koostuu äärellisestä tilajoukosta Q, yksikäsitteisestä alkutilasta $q_0 \in Q$ ja lopputilojen joukosta $F \subseteq Q$, sekä automaatin toiminnan määrittelevästä *transitiofunktiosta* $\delta : Q \times A \to Q$. Transitiofunktio δ laajennetaan funktioksi $Q \times A^* \to Q$ ehdoilla

$$\begin{aligned} \delta(p,1) &= p \quad \text{ja} \\ \delta(q,wa) &= \delta(\delta(q,w),a), \\ &\qquad \text{kun } a \in A \text{ ja } w \in A^*. \end{aligned}$$

Tällöin määritellään automaatin $\mathcal{A}$ *hyväksymä* kieli kaavalla

$$L(\mathcal{A}) = \{w \in A^* \mid \delta(q_0, w) \in F\}.$$

Kieltä kutsutaan *säännölliseksi*, jos sen hyväksyy jokin äärellinen automaatti.

Esimerkki 3. Kirjoittamalla ehdon $\delta(q,a) = p$ asemasta $q \xrightarrow{a} p$ automaattia voidaan havainnollistaa graafilla, jossa nuolet on leimattu aakkoston symboleilla. Edelleen alkutila havainnollistetaan sisääntulevalla nuolella ja lopputilat ulosmenevillä nuolilla. Saamme esimerkiksi kuvassa 2 esitetyn automaatin.

Automaatti siis hyväksyy ne sanat, jotka saadaan graafin tilasta $(1,e)$ tilaan (aba,o) johtavien polkujen leimoina. Väitämme, että

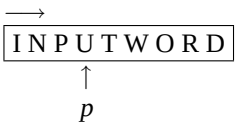
$$L(\mathcal{A}) = \{w \in \{a,b\}^* \mid w\text{:n pituus on pariton ja se sisältää } aba\text{:n osasanana }\}.$$

Tämä nähdään, kun huomataan, että horisontaalisesti automaatti testaa *aba*:n esiintymisen osasanana ja vertikaalisesti se laskee modulo 2. Tilat havainnollistavat luonnollisella tavalla automaatin äärellistä muistia. Esimerkiksi tila (ab,o) kertoo, että tähän asti luettu sana päättyy *ab*:hen, eikä ole vielä sisältänyt osasanaa *aba* ja että sanan pituus on pariton.

Äärellisen automaatin määritelmää voidaan yleistää monella tavalla ilman, että se vaikuttaa hyväksyttyjen kielten perheeseen. Transitiofunktion voidaan sallia olla vain osittainen funktio, siis joillakin arvoilla määrittelemättä, transitioiden leimoina voi olla kirjainten asemasta mielivaltaiset sanat, tai automaatti voi olla myös *epädeterministinen*. Viimeksi mainitussa tapauksessa automaatti siis voi

siirtyä tilasta p lukiessaan a:n joko tilaan p_1 tai p_2. Tällöin sanan hyväksyminen määritellään *eksistentiaalisesti*: on olemassa polku alkutilasta lopputilaan.

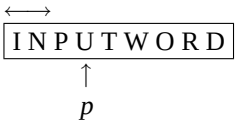
Kiintoisa yleistys on niin kutsuttu *2-suuntainen äärellinen automaatti*. Tavanomaisessa äärellisessä automaatissa lasku etenee niin, että syötesana luetaan askel askeleelta vasemmalta oikealle. Tällaisen laskun vaihetta voidaan kuvata kuviolla:



Tässä siis automaatti lukee symbolia U tilassa p, ja siirtyy askeleen oikealle (ja uuteen tilaan). Kaksisuuntaisessa automaatissa tilanne on aivan sama, paitsi että syötesanalla voidaan siirtyä myös vasemmalle (tai pysyä paikallaan). Siis transitiot ovat muotoa

$$(p,a) \longrightarrow (q,d), \text{ missä } d \in \{-1,0,1\},$$

ja yllä olevan laskun havainnollistus on:



Kaksisuuntainen automaatti hyväksyy sanan, jos automaatti päätyy lopputilaan ja samalla siirtyy sanan oikean laidan yli yllä olevissa havainnollistuksissa. Jälleen voidaan todistaa, että hyväksyttyjen kielten perhe säilyy säännöllisten kielten perheenä, näin siinäkin tapauksessa, että 2-suuntainen automaatti on epädeterministinen. Kuten näemme hetken kuluttua, 2-suuntaisen automaatin määrittelyssä on oleellista, että automaatti *ei laskun aikana muuta* alkuperäistä syötettään.

Edellä määritellyille automaateille yhteistä on, että niiden tulostus on hyvin yksinkertainen eli “hyväksyy”/“ei hyväksy”, siis 1/0. Edelleen, tuloste saadaan vasta laskun lopussa. Luonnollinen ajatus on sallia osa tulostusta laskun kussakin askeleessa. Tämä toteutuu, kun korvataan automaatin transitiot $p \xrightarrow{a} q$ transitioilla

$$p \xrightarrow{u,v} q, \tag{1}$$

missä $u \in A^*$ on tässä transitiossa luettu syötteen osa ja $v \in B^*$ on tässä vaiheessa annettu tuloste. Näin päädytään *äärellisen transduktorin* käsitteeseen. Yleisesti siis sallitaan yhdessä askeleessa luetun sanan u olla mielivaltainen; jos vaaditaan, että u on aakkoston A kirjain puhutaan *yleistetystä jonokoneesta*.

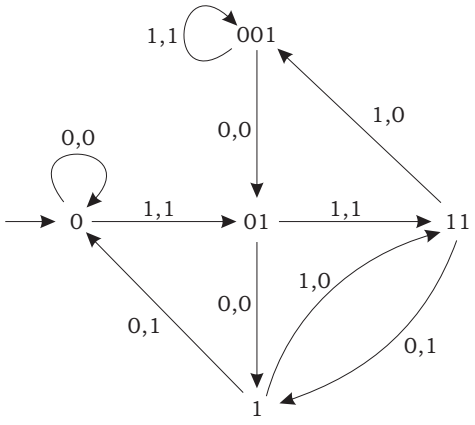
Yleistetty jonokone on *deterministinen*, jos (1):ssä pari (v,q) määräytyy yksikäsitteisesti parista $(p,u) = (p,a)$. Deterministinen yleistetty jonokone siis *laskee* (osittaisen) funktion $A^* \to B^*$, kun taas äärellinen transduktori laskee $A^* \times B^*$:n alijoukon tai moniarvoisen funktion $A^* \to B^*$.

Äärellisten transduktorien teoria on monessa mielessä oleellisesti monimutkaisempi kuin äärellisten automaattien teoria. Erikoisesti tämä johtuu siitä, että näissä transitioiden leimat ovat tulomonoidissa $A^* \times B^*$, joka ei ole, päinvastoin kuin A^*, vapaa. Yhteistä äärellisten automaattien ja äärellisten transduktorien teorioille on, että ne formuloivat sen, mitä *äärellisellä muistilla voidaan toteuttaa*. Seuraavat esimerkit havainnollistavat, mitä näin voidaan laskea.

Esimerkki 4. Osoitetaan, miten binääriluvun kertominen luvulla viisi voidaan suorittaa deterministisellä yleistetyllä jonokoneella. Oletetaan tässä, että $\text{bin}(n)$ on n:n *käänteinen* binääriesitys, siis esimerkiksi $\text{bin}(2) = 01$. On siis laskettava

$$\text{bin}(n) \mapsto \text{bin}(5n).$$

Laskun suorittaa kuvassa 3 esitetty deterministinen yleistetty jonokone. Konstruktio perustuu ajatukseen, että kone muis-



Kuva 3: Deterministinen yleistetty jonokone, joka kertoo luvulla viisi.

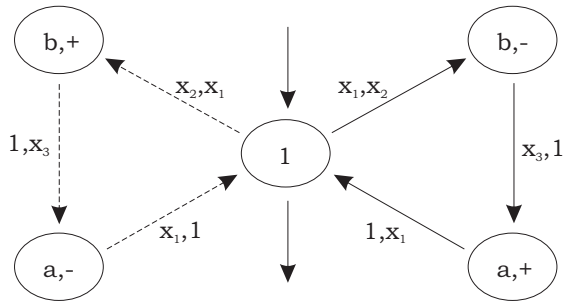
taa kussakin vaiheessa lasketun muistinumeron tiloissaan. Siten esimerkiksi tilasta 11 luettaessa 1 lasketaan $11+101=0001$, joten tulostetaan 0 ja laitetaan muistiin 001. Näinollen syötteen loputtua on muistissa viimeisen vaiheen muistinumero, joten se on tulostettava jotenkin. Luonteva tapa on asettaa syötteen loppuun reunamerkki #, ja tämän lukeminen tulostaa muistin sekä vie automaatin lopputilaan f. Siis koneeseen lisätään esimerkiksi transitio $11 \xrightarrow{\#,11} f$. Näin konstruoitu kone siis suorittaa laskun $\mathrm{bin}(n)\# \mapsto \mathrm{bin}(5n)$.

Tässä esimerkissä on oleellista, että syöte on käännetty binääriesitys – muuten laskua on mahdotonta suorittaa deterministisellä yleistetyllä jonokoneella. Vastaavasti kahden mielivaltaisen binääriluvun tulo on mahdotonta laskea äärellisellä transduktorilla, siis käyttäen vain äärellistä muistia.

Esimerkki 5. Tarkastellaan sanajoukkoa $X=\{a,ab,ba\}$. Merkitään $x_1=a$, $x_2=ab$ ja $x_3=ba$, sekä $\Xi=\{x_1,x_2,x_3\}$. Tällöin esimerkiksi sanalla *ababa* on kaksi eri *X-tekijöihinjakoa*, toisin sanoen se saadaan kahdella tavalla X:n alkioiden tulona: $a.ba.ba = ab.ab.a$. Miten voidaan määrätä kaikki sellaiset sanat, joille on kaksi eri X-tekijöihinjakoa, tai tarkemmin, kaikki joukon Ξ^* parit (η,ϕ), jotka vastaavat jonkin sanan eri tekijöihinjakoja? Ratkaisun tarjoaa kuvan 4 äärellinen transduktori.

Tässä siis kone lukee muuttujia x_i ja muistaa tiloissaan kumpi tekijöihinjako on pidemmällä ja kuinka paljon. Esimerkiksi vertailtaessa x_1:stä ja x_2:sta huomataan, että $x_1 b = x_2$, joten muistiin tulee $(b,-)$ kertomaan, että x_1:llä alkava tekijöihinjako on b:n verran jäljessä. Katkoviivoilla merkitty osa on automaatin toisen osan kanssa symmetrinen. Yllä on huomioitu vain ne X-tekijöihinjaot, jotka alkavat ja päättyvät eri X:n sanoilla. Selvästi konstruktio yleistyy mielivaltaiselle äärelliselle joukolle $X \subseteq A^*$: *aakkoston A sanojen kaksois-X-tekijöihinjaot voidaan laskea äärellisellä transduktorilla.*

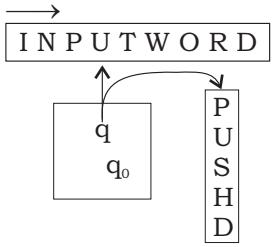
Edellä olevat esimerkit osoittivat, kuinka äärellinen automaatti onnistuu laskuissa, jos on muistettava vain äärellinen



Kuva 4: Kaksi eri tekijöihinjakoa etsivä äärellinen transduktori.

määrä bittejä. Näinollen esimerkiksi kielen $\{a^n b^n \mid n \geq 1\}$, joka on täysin realistinen osa tavanomaisia ohjelmointikieliä, hyväksyminen ei onnistu äärellisellä automaatilla – formaalinen todistus on yksinkertainen lokeroperiaatteen sovellus.

Yllä olevan kielen hyväksymiseksi määrittelemme äärellisten automaattien yleistyksenä niin kutsutut *pinoautomaatit*. Nämä ovat äärellisiä automaatteja, joihin on lisätty potentiaalisesti rajaton *pinomuisti*, toisin sanoen muisti, jota voidaan täyttää ja purkaa vain toisesta päästä. Pinoautomaatin havainnollistus on:



Formaalisesti transitiot ovat muotoa

$$(p, a, Z) \longrightarrow (q, \gamma), \qquad (2)$$

missä p ja q ovat automaatin vanha ja uusi tila, a on luettu syötesymboli (joka voi olla myös tyhjä sana) ja Z on pinomuistin päällimmäinen symboli, joka korvataan tässä laskuaskeleessa pinomuistin sanalla γ. Pinoautomaatissa *hyväksyminen* voi tapahtua vaihtoehtoisilla ekvivalenteilla tavoilla, esimerkiksi vaatimalla, että kun syöte on luettu automaatti *päätyy lopputilaan*. Tällaisella automaatilla kielen $\{a^n b^n \mid n \geq 1\}$ hyväksyminen on helppoa: kone lukee a:t pinomuistiin, ja poimii ne sieltä b:itä lukiessaan.

Pinoautomaatti on määrittelynsä perusteella epädeterministinen. Deterministinen variantti saadaan, kun vaaditaan, että kussakin vaiheessa laskua voidaan soveltaa vain yhtä (2):n transitiota. Automaatin sallitaan yhä lukea tyhjä sana. *Deterministiseltä pinoautomaatilta* siis vaaditaan:

(i) transition (2) q ja γ määräytyvät yksikäsitteisesti kolmikosta (p, a, Z); ja

(ii) jos automaatissa on transitio $(p, 1, Z) \rightarrow (q, \gamma)$ ei siinä ole transitiota millekään kolmikolle (p, a, Z), missä $a \in A$.

Selvästi syöte w määrää yksikäsitteisen deterministisen pinoautomaatin laskun, joka on siis *hyväksyvä* silloin, kun lasku johtaa alkutilasta lopputilaan. Kielen $\{a^n b^n \mid n \geq 1\}$ yllä hyväksyvä pinoautomaatti on selvästi deterministinen.

Pinoautomaatti voi siis vertailla kahta lukua, esimerkiksi hyväksyessään kielen $\{a^n b^n \mid n \geq 1\}$. Kolmea lukua se ei sen sijaan pysty vertailemaan, mikä voidaan formalisoida osoittamalla, että kieltä $\{a^n b^n c^n \mid n \geq 1\}$ ei voida hyväksyä pinoautomaatilla. Tämän kielen hyväksyminen luonnollisesti onnistuu pinoautomaatilla, jossa on kaksi pinoa. Näinollen tällainen automaatti on tavanomaista pinoautomaattia tehokkaampi kielten tunnistaja. Itse asiassa tällaisella automaatilla voidaan hyväksyä mikä tahansa algoritmisesti määritelty kieli, toisin sanoen ne formalisoivat algoritmikäsitteen.

Tavanomaisempi, ekvivalentti formalisointi saavutetaan niin kutsuttujen Turingin koneiden avulla. *Turingin kone* saadaan hyvin luonnollisena 2-suuntaisen äärellisen automaatin yleistyksenä: sallitaan, että kone kussakin vaiheessa korvaa lukemansa symbolin uudella (tai myös vanhalla) symbolilla. Siten Turingin koneen transitiot ovat muotoa

$$(p,a) \to (q,b,d)\,, \quad \text{missä } d \in \{-1,0,1\}. \tag{3}$$

Tässä siis p ja q vastaavat koneen vanhaa ja uutta tilaa, a ja b syötteessä tarkasteltavassa kohdassa olleita symboleja ennen ja jälkeen transition sovellusta ja d lukupään siirtymää. Koska kone pystyy vaihtamaan syötteessä olleita symboleja on syöte ajateltava (kahteen suuntaan) äärettömänä sanana, jossa muut paitsi äärellinen määrä symboleja ovat niin kutsuttuja *blankkosymboleja* $*$. Nämä siis voidaan laskun aikana muuntaa miksi tahansa symboliksi, yksi kerrallaan. Kone *hyväksyy* syötteen, jos lasku päätyy lopputilaan (jolloin alkuperäisen syötteen ei enää tarvitse olla näkyvissä).

Turingin konetta voidaan modifioida mitä moninaisimmilla tavoilla. Voidaan sallia *epädeterministisyys*, voidaan vaatia, että kone operoi vain *toiseen suuntaan äärettömällä nauhalla*, voidaan sallia syötenauhan ohella äärellinen määrä muita niin kutsuttuja *työnauhoja*. Kaikki nämä johtavat samaan luokkaan hyväksyttyjä kieliä, eli niin kutsuttuun *rekursiivisesti numeroituvien* kielten joukkoon. Keskeistä yllä olevassa määrittelyssä on, että ei vaadita, että kone pysähtyy kaikilla syötteillä. Tämä on oleellista, sillä muuten hyväksyttyjen kielten joukko pienenee oleellisesti. *Kaikilla syötteillä pysähtyvät* Turingin koneet nimittäin hyväksyvät vain niin kutsutut *rekursiiviset* kielet. Esimerkin transitioista, jotka johtavat päättymättömään laskuun, tarjoavat transitiot

$(p,a) \to (r,a,1)$
$(r,x) \to (r,x,1)$ kaikilla symboleilla x.

Tällöin koneen lukiessa a:n tilassa p se siirtyy tilaan r, jossa lasku jatkuu loputtomasti oikealle.

Turingin koneesta saadaan transduktorin tavoin (osittaisia) funktioita $A^* \to B^*$ laskeva laite spesifioimalla miten *tuloste* saadaan laskun päätyttyä. Se voi olla esimerkiksi nauhalla siinä vaiheessa oleva sana.

Turingin kone siis koostuu äärellisestä määrästä kohdan (3) mukaisia viisikkoja (p,a,q,b,d). Näinollen se on varsin yksinkertaisesti määritelty käsite. Kuitenkin se pystyy suoriutumaan hämmästyttävän vaativista algoritmisista laskuista:

$$f : A^* \to B^*,\ w \mapsto f(w),$$

missä siis funktion f sallitaan olla vain osittainen, mutta vaaditaan, että se on algoritmisesti laskettavissa. Itse asiassa 1940-luvulla kiteytynyt *Church–Turingin teesi* sanoo, että kaikki mikä on algoritmisesti laskettavissa on Turingin koneilla laskettavissa. Tämä tietysti edellyttää, että algoritmin syöte ja tuloste on sopivasti *koodattu koneen aakkostoon A*.

Turingin konetta voidaan siis pitää *intuitiivisen algoritmikäsitteen abstraktina mallina*: mikä voidaan algoritmisesti, siis tietokoneohjelmalla, laskea, voidaan laskea Turingin koneella. Päättelyä voidaan viedä vielä askel pidemmälle. Voidaan nimittäin konstruoida kiinteä Turingin kone $\mathcal{M}_u$, joka saatuaan syötteenä mielivaltaisen Turingin koneen $\mathcal{M}$ (sopivasti koodattuna) ja tämän syötteen w (niin ikään sopivasti koodattuna) suorittaa koneen $\mathcal{M}$ mukaisen laskun syötteellä w ja tulostaa tämän tulosteen (jälleen sopivasti koodattuna). Toisin sanoen, jos $\mathcal{M}$:n koodaus koneen $\mathcal{M}_u$ aakkostoon on $c(\mathcal{M})$ ja $\mathcal{M}$:n sanan w vastaava koodaus on $c(w)$, niin $\mathcal{M}_u$ suorittaa laskun

$$c(\mathcal{M}) * c(w) \mapsto c(\mathcal{M}(w)),$$

jos $\mathcal{M}$ syötteellä w laskee

$$w \mapsto \mathcal{M}(w).$$

Konetta $\mathcal{M}_u$ kutsutaan *universaaliksi Turingin koneeksi*. Sitä voidaan pitää *tietokoneen abstraktina mallina*: saatuaan ohjelman, siis algoritmin eli koneen $\mathcal{M}$ koodauksen, ja tämän algoritmin syötteen w, $\mathcal{M}_u$ toteuttaa algoritmin laskun.

On oleellista huomata, että $\mathcal{M}_u$ koostuu vain *kiinteästä* määrästä kohdan (3) viisikkoja. Hämmästyttäväksi tämän tekee se, että näiden määrä voi olla varsin pieni, eli tarkemmin sanoen on olemassa universaaleja Turingin koneita ([52]), joissa on

- 2-alkioinen aakkosto ja 19 tilaa,
- 3-alkioinen aakkosto ja 10 tilaa,
- 4-alkioinen aakkosto ja 7 tilaa,
- 5-alkioinen aakkosto ja 5 tilaa,
- 6-alkioinen aakkosto ja 4 tilaa,
- 9-alkioinen aakkosto ja 3 tilaa, sekä
- 19-alkioinen aakkosto ja 2 tilaa.

Tapauksessa $(5,5)$ siis universaali kone koostuu vain 25 transitiosta (p,a,q,b,d). Church–Turingin teesi sanoo, että koko *konstruktiivinen* matematiikka on koodattu näihin sääntöihin!

Algoritmikäsitteen formalisoinnin ja Church–Turingin teesin hyväksymisen syvällisiä seurauksia on *algoritmisen ratkeamattomuuden* käsite. Nimittäin väite, että jotakin tehtävää ei voida laskea Turingin koneella, on matemaattisesti legitimoitu väite, joka *voidaan monessa tapauksessa todistaa oikeaksi*. Church–Turingin teesin nojalla tämä tarkoittaa, että kyseistä laskua ei voida algoritmisesti suorittaa – se on siis tietokoneiden saavuttamattomissa niiden tehosta riippumatta!

Turingin koneen hyväksyvässä laskussa laskuaskelten tai käytettyjen nauhan muistipaikkojen lukumäärää ei mitenkään rajoiteta – näiden tulee olla ainoastaan äärellisiä. Rajoittamalla niitä syötteen koosta riippuvilla funktioilla päädytään *algoritmien kompleksisuusteoriaan*. Jos Turingin kone $\mathcal{M}$ jokaisella pituutta n olevalla syötteellä antaa tuloksen korkeintaan $f(n)$ laskuaskeleessa sanotaan, että koneen *aikakompleksisuus* on $f(n)$. Vastaavasti määritellään *tilakompleksisuus* tarkastelemalla koneen laskun aikana käyttämien syötenauhan muistipaikkojen lukumäärää. Nämä käsitteet voidaan määritellä yhtä hyvin deterministisille kuin epädeterministisillekin Turingin koneille, jolloin siis epädeterminististen koneiden osalta hyväksyminen on, äärellisen epädeterministisen automaatin mukaisesti, eksistentiaalista: on olemassa tässä resurssissa toimiva hyväksyvä lasku.

Erikoisen tärkeitä luokkia ovat luokat, jossa kompleksisuusfunktio on joko logaritminen tai polynomiaalisesti rajoitettu. Merkitään esimerkiksi logaritmitilassa deterministisesti toimivien Turingin koneiden hyväksymää kieliperhettä **LOGSPACE**:llä ja polynomiajassa epädeterministisesti toimivien Turin-

gin koneiden hyväksymää kieliperhettä **NPTIME**:llä. Algoritmien kompleksisuusteorian peruskysymykset sisältyvät kaavaan:

$$\begin{aligned}\mathbf{LOGSPACE} &\subseteq \mathbf{NLOGSPACE} \\ &\subseteq \mathbf{PTIME} \subseteq \mathbf{NPTIME} \\ &\subseteq \mathbf{PSPACE} \subseteq \mathbf{NPSPACE}\end{aligned} \tag{4}$$

Esitetyt sisältymiset ovat joko triviaaleja tai teorian helppoja perustuloksia. Näiden lisäksi tiedetään, että viimeinen sisältyminen on itse asiassa yhtäsuuruus, ja että kaksi ensimmäistä kieliperhettä sisältyvät aidosti kahteen viimeiseen. Muilta osin sisältymisten aitous on selvittämättä, ja ne edustavat algoritmisen kompleksisuusteorian suuria avoimia kysymyksiä. Kysymys "Onko $\mathbf{PTIME} = \mathbf{NPTIME}$?" on aiemmin mainittu $\mathbf{P} = \mathbf{NP}$ -probleema.

Historiallisesti merkittävät *lineaarisesti rajoitetut automaatit* saadaan Turingin koneista vaatimalla, että lasku toimii syötteeseen w nähden lineaarisessa tilassa, siis tilassa $O(|w|)$. Näin määräytyvät deterministiset ja epädeterministiset lineaarisesti rajoitetut automaatit. Näiden osalta, päinvastoin kuin äärellisten automaattien, pinoautomaattien tai Turingin koneiden kohdalla, *ei tiedetä* ovatko deterministinen ja epädeterministinen variantti yhtä tehokkaita kielten tunnistajia. Tämä on yhteydessä kaavan (4) avoimiin kohtiin.

3 Katsaus automaattien teoriaan

Tämän luvun tavoitteena on luoda lyhyt katsaus automaattien teoriaan kuluneiden 50 vuoden ajalta. Katsaus ei pyri olemaan mitenkään kattava, pikemminkin on pyritty valitsemaan ja erittelemään teorian kehityksen kannalta keskeisiä suuntaviivoja sekä probleemoja ja niiden ratkaisuja.

Automaattien teorian synty. Automaattien teoria syntyi 1950-luvulla. Tähän vaikutti ennen muuta kaksi seikkaa. Ensinnäkin, insinöörit laatiessaan elektronisia kytkentäpiirejä tarvitsivat matemaattisen mallin näiden analysointiin ja rakentamiseen. Toiseksi tuolloin ilmestyi muutama perustavaa laatua oleva teoreettinen artikkeli. Näistä ensimmäinen oli S. Kleenen vuonna 1956 julkaisema työ [49], joka

> *karakterisoi äärellisten automaattien hyväksymät kielet matemaattisten sulkeumaominaisuuksien avulla, rationaalisina kielinä.*

Usein tätä työtä pidetään automaattien teorian alkuna.

Toinen automaattien teorian historiaa laajalti muokannut työ oli M. Rabinin ja D. Scottin vuonna 1959 kirjoittama työ äärellisistä automaateista ja niiden ratkeavuusprobleemoista ([70]). Työn erityisanti oli sen matemaattinen selkeys; se kiinnitti terminologian, esitteli useita ongelmia ja osoitti muun muassa että

> *Epädeterministiset ja deterministiset automaatit ovat ekvivalentteja kielten hyväksymisen kannalta.*

Samoihin aikoihin Shepherdson osoitti ([79]), että myöskään

> *2-suuntaiset äärelliset automaatit eivät pysty hyväksymään muita kuin säännöllisiä kieliä.*

Näin muutamassa vuodessa säännöllisten kielten perheelle oli löydetty neljä toisistaan poikkeavaa karakterisaatiota. Matemaattisesti tämä tarkoittaa, että oli löydetty jotain tärkeää, esityksestä riippumatonta. Monet myöhemmät karakterisaatiot, kuten [9] ja [27] yhä vahvistivat tätä näkemystä.

Mainittujen töiden ohella teorian käynnistäjinä, hieman eri näkökulmasta, toimivat N. Chomskyn 50-luvun lopulla kirjoittamat työt *formaalisista kieliopeista*. Näiden lähtökohtana oli pyrkimys luonnollisten kielten rakenteen mallintamiseen. Tarkastelut huipentuivat 60-luvun alussa hänen M.P. Schützenbergerin kanssa kirjoittamaansa merkittävään työhön [15].

Teorian intesiivivaihe. Heti syntymänsä jälkeen automaattien teorian tutkimus laajeni ripeästi. Osaltaan tähän vaikutti teorian vakuuttava alku, mutta paljon enemmän ajan tarve: *tietojenkäsittelytiede oli syntymässä*. Tarvittiin täsmällisiä matemaattisia laskettavuuden malleja, tarvittiin teoriaa ohjelmointikielten rakenteen kuvaamiseen, tarvittiin teoria tietokoneohjelmien kääntämiseen konekielelle. Näihin kysymyksiin automaateilla ja Chomskyn generoivilla kieliopeilla oli paljon annettavaa. Oli syntynyt *automaattien ja formaalisten kielten teoria*. Vuonna 1972 käynnistynyt kongressi Automata, Languages and Programming, lyhyesti ICALP, kehittyi nopeasti teoreettisen tietojenkäsittelytieteen johtavaksi eurooppalaiseksi kongressiksi. Se on pidetty kolmesti Suomessa, vuosina 1977 ja 2004 Turussa sekä vuonna 1988 Tampereella.

Myös Suomessa alan tutkimus käynnistyi ripeästi. Akateemikko Arto Salomaa käynnisti automaattien teorian seminaarin Turun yliopistossa vuonna 1963. Tästä lähtien seminaari on jatkunut – hieman vaihtelevin muodoin – jo yli 40 vuotta. Näinä vuosina Turusta on väitellyt yli 20 henkilöä automaattien teoriasta tai lähialoilta.

Käynnistyttyään automaattien teorian tutkimus oli hyvin intensiivistä ja nopeasti uutta etsivää. Teorian perustulokset todistettiin ripeästi, mutta samalla paljastui, yllättävien tulosten ohella, että monet alan keskeisistä kysymyksistä ovat hyvin vaikeita ja matemaattisesti haastavia. Esittelemme tässä joitakin tällaisia tuloksia ja ongelmia.

Tutkimuksen alkuvaiheen keskeisiä kysymyksiä oli teorian ja/tai sovellusten kannalta tärkeiden kieliperheiden etsiminen ja niiden perusominaisuuksien tutkiminen. Näin päädyttiin *Chomskyn hierarkiaan*, joka alkaa säännöllisistä kielistä ja päätyy CF- ja CS-kieliperheiden kautta rekursiivisesti numeroituviin kieliin. Kullekin näistä luokista esitettiin sekä automaattiteoreettiset että kielioppeihin perustuvat karakterisaatiot. Automaattikarakterisaatiot tarjoavat edellä määritellyt äärelliset automaatit, pinoautomaatit, epädeterministiset lineaarisesti rajoitetut automaatit sekä Turingin koneet.

Yksi merkittävistä teorian alkuajan saavutuksia – ja yllätyksiä – oli huomata, että monet yksinkertaiset ja käytännön, esimerkiksi ohjelmointikielten teorian, motivoimat kysymykset ovat algoritmisesti ratkeamattomia. Tällaisia probleemoja olivat muun muassa:

(i) pinoautomaattien ekvivalenssiprobleema, toisin sanoen sen selvittäminen hyväksyykö kaksi annettua pinoautomaattia saman kielen;

(ii) yleistettyjen jonokoneiden ekvivalenssiprobleema;

tai hieman intuitiivisemmalla tasolla:

(iii) Toimiiko annettu tietokoneohjelma oikein?

Paljastui, että Postin vuonna 1946 algoritmisesti ratkeamattomaksi todistama kombinatorinen sanoihin liittyvä kysymys, niin kutsuttu *Postin vastaavuusprobleema* ([69]), on erittäin sovelias työkalu osoitettaessa formaalisten kielten probleemoja ratkeamattomiksi.

Tutkimuksen intensiivivaiheen yksi keskeinen tavoite oli identifioita teorian kannalta merkittäviä avoimia kysymyksiä. Tällaisiksi osoittautuivat muun muassa:

Probleema 1 *Moninauhaisten deterministisen äärellisten automaattien ekvivalenssiprobleema.*

Probleema 2 *Deterministisen pinoautomaattien ekvivalenssiprobleema.*

Probleema 3 (LBA-probleema)
Voidaanko jokainen epädeterministisellä lineaarisesti rajoitetulla automaatilla hyväksyttävä kieli hyväksyä myös vastaavalla deterministisellä automaatilla?

Probleema 4 *Oletetaan, että kielen L hyväksyy n-tilainen 2-suuntainen epädeterministinen automaatti. Voiko tämän kielen hyväksyvä deterministinen 2-suuntainen automaatti vaatia eksponentiaalisesti enemmän tiloja?*

Probleemoista kahdelle ensimmäiselle on löydetty myönteinen vastaus, kuten tulemme näkemään, kun taas kaksi jälkimmäistä ovat yhä avoinna. Probleeman 4 kysymys tavanomaisille 1-suuntaisille automaateille selvitettiin aivan teorian alkuaikoina.

Myös tärkeä $\mathbf{P} = \mathbf{NP}$ -probleema syntyi teorian tässä vaiheessa. Turingin koneen tultua hyväksytyksi laskennan täsmällisesti määritellyksi perusmalliksi oli luonnollista, että laskennallinen kompleksisuus, siis se kuinka paljon laskuaskeleita tai muistipaikkoja jonkin tehtävän ratkaiseminen vaatii, nousi keskeiseksi tutkimuskohteeksi.

Syntyi *algoritmien kompleksisuusteoria*. Varsin nopeasti paljastui, että resurssin vähäinenkin lisääminen tiettyyn laskennan malliin lisää aidosti ratkaistavien probleemojen lukumäärää. Esimerkiksi ajassa (tai tilassa) $O(n^2 \log n)$ ja $O(n^2)$ deterministisellä Turingin koneella ratkaistavissa olevat probleemajoukot eivät ole samat. Toisaalta – jälleen yllättäen – paljastui, että eri laskentamallien, esimerkiksi epädeterministisen ja deterministisen, vertailu voi olla äärimmäisen hankalaa. Tästä on yllämainittu Probleema 3 konkreettinen esimerkki. Toisaalta W. Savitchin vuonna 1970 todistama tulos ([77]), jonka mukaan

> *epädeterministisen Turingin koneen tilassa $S(n) \in \Omega(\log n)$ hyväksymä kieli voidaan hyväksyä deterministisellä Turingin koneella tilassa $O(S(n)^2)$,*

on esimerkki saavutetuista positiivisista tuloksista. Tämä tulos osoittaa myös kaavan (4) viimeisen sisältymisen itse asiassa yhtäsuuruudeksi.

Algoritmien kompleksisuusteorian alkuajan tutkimus huipentui $\mathbf{P} = \mathbf{NP}$ -probleeman formulointiin:

Probleema 5 *Ovatko kaikki polynomiajassa epädeterministisellä Turingin koneella ratkeavat probleemat ratkaistavissa myös deterministisellä Turingin koneella polynomiajassa?*

Probleeman merkitys paljastui S. Cookin todistaessa vuonna 1971, että

> *on olemassa niin kutsuttuja* **NP***-täydellisiä ongelmia,*

ja R. Karpin osoittaessa vuonna 1972, että niitä on itse asiassa varsin paljon ([16] ja [47]). Tällainen ongelma on esimerkiksi kysymys, voidaanko annettu tasokartta värittää kolmella värillä polynomiaikaisella algoritmilla niin, että naapureilla on aina eri väri. Sittemmin **NP**-täydellisiä probleemoja on löydetty useita satoja ([32]). Menemättä yksityiskohtaisemmin **NP**-täydellisyyteen, mainitsemme tässä, että mainittu miljoonan dollarin kysymys "Onko $\mathbf{P} = \mathbf{NP}$?" palautuu

NP-täydellisten probleemojen kompleksisuuden selvittämiseen: jos jokin **NP**-täydellinen probleema voidaan ratkaista polynomiajassa (deterministisellä Turingin koneella) ovat luokat **PTIME** ja **NPTIME** samat, muutoin ei.

Teorian vakiintumisen vaihe. Vajaassa 20 vuodessa, 1970-luvun alkuvuosiin mennessä, automaattien ja formaalisten kielten teoriasta oli tullut vakiintunut kypsä matemaattinen teoria. Perustulokset oli todistettu ja haastavat probleemat vuosiksi eteenpäin oli formuloitu. Tutkimus jatkui edelleen intensiivisenä, mutta nyt monille osa-alueille sirpaloituneena. Edelliselle vuosikymmenelle leimallinen yhtenäisyys oli katoamassa.

Syntyi monia uusia osa-alueita, kuten *formaalisten potenssisarjojen teoria* (esimerkiksi [75]), *L-systeemien teoria* ([72]), *puuautomaattien teoria* ([33]), *soluautomaattien teoria*, *rinnakkaislaskennan teoria* ja niin edelleen. Monet tutkimussuunnista erkanivat varsin kauas klassisesta automaattien teoriasta. Algoritmien kompleksisuusteoria, ja laajemminkin algoritmien teoria, ovat hyviä esimerkkejä tästä.

Uudet teoriat toivat mukanaan uusia kysymyksiä ja uusia työvälineitä vanhojen ratkaisemiseksi. Formaalisten potenssisarjojen teoria on tästä hyvä esimerkki. Siinä tarkastellaan äärellisiä automaatteja, mutta nyt ollaan kiinnostuneita, ei ainoastaan siitä mitä hyväksytään, vaan kuinka monta kertaa annettu sana hyväksytään. Tarkastellaan siis epädeterminististiä automaatteja. Teoria oli ensi sijassa matemaattisesti perusteltu, mutta se avasi aivan uusia mahdollisuuksia. Paljastui muun muassa, että jopa eräät äärellisiin automaatteihin liittyvät kysymykset ovat algoritmisesti ratkeamattomia, kuten esimerkiksi ([29]):

> *Hyväksyykö annettu äärellinen epädeterministinen automaatti jokaisen hyväksymänsä sanan useammin kuin toinen annettu samanlainen automaatti?*

Formaalisten potenssisarjojen teoria osoittautui myös keskeiseksi ratkaistaessa probleema 1 ([35]):

> *Deterministisen moninauhaisten äärellisten automaattien ekvivalenssiprobleema on algoritmisesti ratkeava.*

Teorian kehittyessä motivaatiota haettiin myös uusilta aloilta. Tästä esimerkkinä ovat niin kutsutut *Lindenmayerin systeemit*, eli *L-systeemit* ([72]). Ne esiteltiin nauhamaisten organismien kasvun ja kehityksen kuvaamiseen ([57]). Formaalisten kielten teorian kannalta ne toivat esiin uudelleenkirjoituksen samanaikaisuuden, *paralleelisuuden*: kukin symboli korvattiin uudella sanalla laskun joka vaiheessa. Yksinkertaisin esimerkki L-systeemeistä on niin sanottu *D0L-systeemi*, missä vapaiden puoliryhmien morfismia $h : A^* \to A^*$ käytetään generoidun sanajonon

$$w,\ h(w),\ h^2(w),\ \ldots \tag{5}$$

tai generoidun kielen $\{h^n(w) \mid n \geq 0\}$ määrittelyyn. Uudelleenkirjoituksen samanaikaisuus on tässä ilmeistä, ensimmäisessä vaiheessa sanan w kukin kirjain korvataan samanaikaisesti sanalla $h(a)$, näin saadaan sana $h(w)$.

Jonoa (5) kutsutaan (morfismin h ja sanan w määräämäksi) *D0L-jonoksi*. L-systeemien merkitys biologisten ilmiöiden mallintajana voidaan kyseenalaistaa, mutta niiden merkitystä uusien haasteiden ja merkittävien matemaattisten tulosten innoittajana on vaikea yliarvioida. Erikoisesti niin kutsuttu *D0L-jonojen ekvivalenssiprobleema*, siis algoritminen kysymys siitä, onko kaksi (5):n mukaista jonoa samat, osoittautui erikoisen haastavaksi.

Vaikka probleema osoitettiin ratkeavaksi ([21] ja [28]), on yhä avoinna minkäänlaisen kohtuullisen rajan löytyminen sille, kuinka pitkälle (5):n jonoja tulee verrata niiden identtisyyden selvittämiseksi. Raja voi vallan hyvin olla niinkin pieni kuin 2 kertaa aakkoston A koko – keinoja sen osoittamiseksi ei vain ole tiedossa. Muiden D0L-jonon motivoimien ongelmien osalta viittaamme työhön [41].

Mainitsimme jo, että teorian alkuajan keskeisistä ongelmista probleema 1 ratkaistiin 90-luvun alussa. Probleeman 2 kompleksista ratkaisua juoduimme odottamaan aivan viime vuosituhannen lopulle, jolloin G. Senizergues ([78]) osoitti:

> *Determinististen pinoautomaattien ekvivalenssiprobleema on ratkeava.*

Probleema 3, kuten monet muutkin algoritmien kompleksisuusteorian ongelmat, on yhä selvittämättä. Kuitenkin tähän liittyen N. Immerman ja R. Szelepcsenyi ([40] ja [83]) osoittivat vuonna 1988, että

> *Tilassa $S(n) \in \Omega(\log n)$ epädeterministisellä Turingin koneella hyväksyttyjen kielten perhe on suljettu komplementinottoon nähden. Erikoisesti näin on CS-kieliperheen osalta.*

On hämmästyttävää, että tämän selkeästi automaattien teorian helmiin kuuluvan tuloksen löytyminen kesti yli neljännesvuosisadan. Probleeman 4 osalta tilanne on yhä täysin avoin, vain kovin spesifisiä osittaistuloksia on onnistuttu todistamaan, katso esimerkiksi [63].

Vuosikymmeniä jatkunut intensiivinen automaattien ja formaalisten kielten tutkimus huipentui vuonna 1997 ilmestyneeseen käsikirjaan Handbook of Formal Languages [73]. Tämä kahden alan voimahahmon, G. Rozenbergin ja A. Salomaan, toimittama yli 2000-sivuinen 3-osainen merkkiteos kokosi yksiin kansiin 40 vuoden aikana kertyneen tiedon.

Uudet haasteet. Automaattien teorian saavutukset kuluneiden vuosikymmenien aikana ovat kiistattomat. Teoriaa ei kuitenkaan voida pitää valmiina – monet peruskysymykset, kuten probleema 4, ovat yhä vastausta vailla. Toisaalta monet esitetyistä vielä ratkaisemattomista kysymyksistä ovat ilmeisesti erittäin vaikeita, ja siten nykyisessä meritoitumista korostavassa tiedemaailmassa usein vähän houkuttelevia. Onko siis kiinnostus automaattien ja formaalisten kielten teoriaan hiipumassa?

Monet merkit osoittavat, että vastaus tähän on selkeästi “ei”. Näkemystä tukee muun muassa se, että alalta ilmestyy jatkuvasti uusia oppikirjoja johtavien tutkijoiden kirjoittamina, esimerkkejä näistä ovat [51] ja [81].

Taustalla on myös konkreettisempia ja vankempia tosiseikkoja. *Uudet laskettavuuden* mallit – kvanttilaskenta ja DNA-laskenta – tarjoavat runsaasti haastavia uusia ongelmia myös klassisille automaattiteoreetikoille. *Automaattien teorian soveltuvuus* mitä moninaisemmille tutkimusaloille on niin ikään korostunut viime vuosina. Edelleen tietojenkäsittelyn teoriassa viime aikoina painottuneet *interaktiivisuus* ja *probabilistisuus* ovat yhä vaillinaisesti ymmärrettyjä – ja tutkittuja – myös äärellisten automaattien yhteydessä. Esimerkiksi, mitä voidaan sanoa Las Vegas -tyyppisistä äärellisistä automaateista ([39])? *Alternoivat äärelliset automaatit* ([12]) – jotka tarjoavat jälleen uuden karakterisaation säännöllisille kielille – muodostavat toisen tärkeän luokan äärellisiä automaatteja.

Automaattien teorian sovelluksia tarkastellaan lähemmin – muttei suinkaan

kattavasti – seuraavassa luvussa. Tarkasteltujen esimerkkien ohella tärkeäksi sovellusalueeksi on noussut automaattien käyttö mallien testauksessa (model checking) ja systeemin verifioinnissa (system verification). Tässä erikoisesti äärettömillä sanoilla laskevat automaatit ovat nousseet keskeiseksi tutkimuskohteeksi ([84]), sekä perustutkimuksen että sovellusten näkökulmasta.

Kvanttilaskenta ja *DNA-laskenta* ([37] ja [67]), tarjoavat kaksi uutta haastavaa laskennan mallia. Edellinen perustuu kvanttimekaniikkaan, ja siten aivan erilaisiin fysikaalisiin ilmiöihin kuin klassiset tietokoneet – tai niiden abstrakteina malleina esitetyt Turingin koneet. Kvanttilaskennan – matemaattisena tutkimuskohteena – voidaan katsoa käynnistyneen P. Shorin mullistavasta työstä vuodelta 1994 ([80]), jossa osoitettiin, että

> *Matematiikan vanhimpiin kuuluva kysymys luvun tekijöihinjaosta voidaan ratkaista polynomiajassa kvanttilaskennan operaatioilla – siis kvanttitietokoneella, jos sellainen onnistutaan rakentamaan.*

Kvanttilaskennan alkeisoperaatiot, Hilbertin avaruuden unitaarimuunnokset, ovat sangen erilaisia kuin tavanomaisen laskennan bittioperaatiot. Näinollen niihin perustuvien laskennan mallien tutkiminen on matemaattisesti luonnollista ja hyvin haastavaa.

DNA-laskennan voidaan katsoa käynnistyneen L. Adlemanin niinikään vuonna 1994 julkaisemasta työstä ([1]), jossa

> *ratkaistiin kokeellisesti DNA-molekyylien avulla yhden* **NP***-täydellisen probleeman – niin kutsutun Hamiltonin polun probleeman – pieni instanssi.*

Laskennan malli perustuu DNA-molekyylien kaksoisrakenteeseen sisältyvän informaation hyväksikäyttöön. Näinollen DNA-laskennassa operoidaan konkreettisesti sanoilla käyttäen aivan uusia molekyylitason operaatioita laskuaskelina. Teoria on lähtökohdiltaan hyvin puhdasta formaalisten kielten teoriaa. Näinollen klassisin formaalisten kielten metodein on voitu osoittaa DNA-laskennan universaalisuus, toisin sanoen se, miten varsin yksinkertaisilla – luonnon motivoimilla – laskuoperaatioilla saavutetaan Turingin koneiden laskukapasiteetti ([67]).

Automaattien ja sanojen tutkijoille DNA-molekyylit tarjoavat myös toisen kiintoisan näkökulman. Tällöin pyrkimyksenä ei ole käyttää DNA:n kapasiteettia laskemiseen vaan tavoitteena on mallintaa ja ymmärtää “Miten luonto laskee?”, katso [71], toisin sanoen miten solut käyttävät DNA:n kaksoiskierrettä informaation käsittelyyn.

Automaattien teorian nykytilaa ja tulevaisuuden näkymiä on esitelty laajemmin äskettäin ilmestyneessä kirjassa A Half-Century of Automata Theory ([76]).

4 Esimerkkejä äärellisten automaattien sovelluksista

Tässä luvussa esitellään joitakin äärellisten automaattien sovelluksia. Esimerkeillä pyritään korostamaan sovellusten monipuolisuutta ja ajankohtaisuutta. Ne kaikki perustuvat 90-luvun tutkimuksiin.

Sovellus 1. Intuitiivisesti äärellinen automaatti formalisoi sen, mitä äärellisellä muistilla voidaan tehdä. Näinollen on varsin odotettua, että äärelliset automaatit ovat oiva työväline monilla diskreetin matematiikan aloilla, esimerkiksi algebrassa. Automaattisten ryhmien teoria on tästä hyvä esimerkki ([30]). Tähän liittyen tarkastelemme tässä lyhyesti yhtä keskeistä

algebran kysymystä:

> *Selvitettävä, onko kaksi vapaan puoliryhmän A^* äärellisesti generoitua alipuoliryhmää M ja N isomorfiset?*

Voimme siis olettaa, että M:n ja N:n generoivat äärelliset kielet X ja Y, toisin sanoen $M = X^*$ ja $N = Y^*$, ja on selvitettävä, toteuttavatko X ja Y (mahdollisen uudelleennimeämisen jälkeen) samat relaatiot. Merkitään näitä relaatioita symboleilla $\mathcal{R}(X)$ ja $\mathcal{R}(Y)$. Ne siis ovat, esimerkin 5 mukaisesti, joukon Ξ^* relaatioita, siis $\Xi^* \times \Xi^*$:n osajoukkoja, missä Ξ on X:n ja Y:n yhteinen kopio (vertaa esimerkki 5). Edelleen esimerkin 5 mukaisesti $\mathcal{R}(X)$ ja $\mathcal{R}(Y)$ ovat rationaalisia relaatioita, siis äärellisen transduktorin hyväksymiä. Näinollen probleema palautuu äärellisten transduktorien ekvivalenssiprobleemaan. Mutta sehän on, kuten näimme, algoritmisesti ratkeamaton!

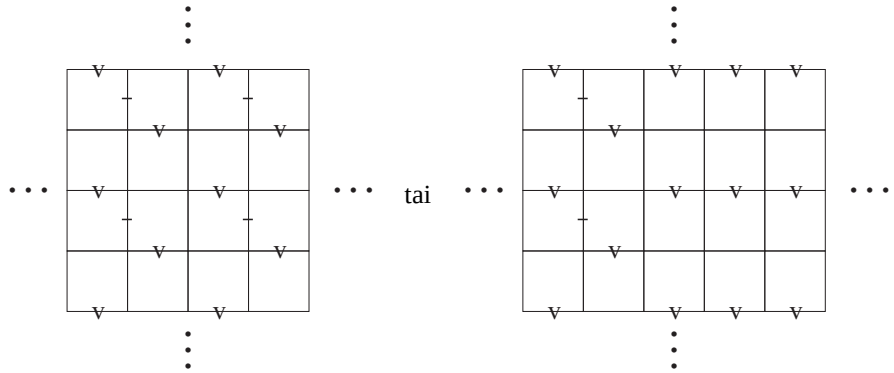
Pelastukseksi nousee sanojen kombinatoriikka. $\mathcal{R}(X)$ on ääretön yhtälöryhmä muuttujinaan Ξ, ja X on sen ratkaisu. Sanojen fundamentaalisia ominaisuuksia on, katso Ehrenfeuchtin kompaktisuusominaisuus luvussa 5, että mielivaltainen yhtälöryhmä on ekvivalentti jonkin äärellisen osansa kanssa. Tässä *ekvivalenttisuus* tarkoittaa, että alkuperäisellä ryhmällä ja sen osalla on tarkalleen samat ratkaisut. Siten alkuperäinen ongelma ratkeaa, kun konstruoidaan sekä $\mathcal{R}(X)$:lle että $\mathcal{R}(Y)$:lle ekvivalentit äärelliset yhtälöryhmät $\mathcal{R}_0(X)$ ja $\mathcal{R}_0(Y)$, ja tarkastetaan, että X ja Y molemmat ovat kummankin ratkaisuja. Tämä tietysti edellyttää, että esimerkiksi $\mathcal{R}_0(X)$ voidaan konstruoida efektiivisesti $\mathcal{R}(X)$:stä. Näin on äärellisten automaattien niin kutsutun pumppauslemman johdosta.

Yllä hahmoteltu algebran varsin keskeinen probleema on ratkaistu elegantisti käyttämällä hyväksi automaattien teorian ja sanojen kombinatoriikan perustuloksia. Yksityiskohtaisemmin ratkaisuun voi tutustua artikkelissa [13]. On mielenkiintoista huomata, että vastaava probleema multiplikatiivisille matriisipuoliryhmille on algoritmisesti ratkeamaton ([11]).

Sovellus 2. Toisena sovelluksena tarkastelemme *tason laatoitukseen* liittyvää ongelmaa. Tässä on annettuna joukko äärellisen montaa eri tyyppiä olevia neliölaattoja, esimerkiksi tyypit

ja kysytään, voidaanko näillä peittää koko taso niin, että naapurit ovat aina keskenään yhteensopivia, toisin sanoen niillä on sama väri. Kyseisillä laatoilla tämä on selvästi mahdollista, kuten esimerkiksi kuva 5 havainnollistaa. Näistä vasemmanpuoleinen laatoitus on sekä vertikaalisesti että horisontaalisesti jaksollinen, kun taas toinen on vain vertikaalisesti jaksollinen. Tässä tapauksessa on myös helppo konstruoida laatoitus, joka ei ole lainkaan *jaksollinen*, toisin sanoen mikään siirto ei säilytä laatoitusta muuttumattomana.

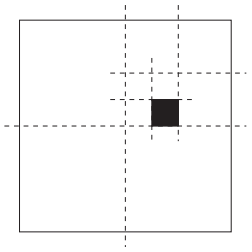
Mielenkiintoinen ei-triviaali kysymys on, onko olemassa laattajoukkoja, joilla taso voidaan peittää vain *jaksottomalla* tavalla. Kysymykseen antoi myönteisen vastauksen R. Berger ([3]) vuonna 1966. Ratkaisu oli sangen monimutkainen, ja vaati yli 2000 tyyppiä olevia laattoja! Sittemmin tarvittavien laattatyyppien määrää on onnistuttu pienentämään. Toistaiseksi paras ratkaisu on saavutettu äärellisten automaattien (ja lukuteorian) sovelluksena: J. Kari osoitti, että erityyppisten laattojen lukumäärää voidaan pudottaa neljääntoista ([46]) (itse asiassa näistäkin laattatyypeistä yksi on tarpeeton, [20]).



Kuva 5: Kaksi tason laatoitusta.

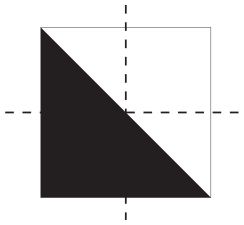
Sovellus 3. Äärellinen automaatti on säännöllisen, yleensä äärettömän, kielen kompakti esitystapa. Tähän itsestään selvyyteen perustuu ajatus niiden käytöstä *kuvien generoimiseen* ja *tiivistämiseen*. Tämä voidaan tehdä useilla eri tavoilla. Työ [17] esittelee erään äärellisiin automaatteihin perustuvan metodin tekstin tiivistämiseen. Tässä esittelemme toisen, formaalisiin potenssisarjoihin perustuvan metodin.

Tarkastellaan neliömuotoista kuvaa, joka on jaettu yhä uudelleen ja uudelleen pienempiin osaneliöihin:

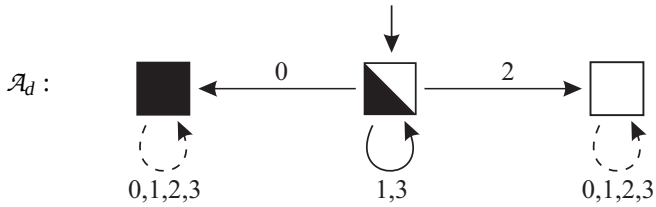


Merkitään ensimmäisen tason osaneliöitä alhaalta vasemmalta lähtien myötäpäivään symboleilla 0, 1, 2 ja 3. Jatkamalla samoin saadaan kunkin tason kullekin osaneliölle osoite: kuvan tummennettua neliötä vastaa sana 203. Annetulle kuvalle saadaan approksimaatio, kun suoritetaan sivujen puolittamista tarpeeksi monta kertaa, ja kootaan koko kuva näin saaduista pienistä osakuvista. Monissa tapauksissa tämä voidaan tehdä tehokkaasti äärellisellä automaatilla. Tarkastellaan tästä yksinkertaista esimerkkiä.

Esimerkki 6. Tarkastellaan kuvaa



Kun tähän sovelletaan yllä olevaa jakometodia, saadaan 4 pienempää neliötä, joista kaksi on identtiset. Osaneliötä ja niiden osoitteita yhdistää kuvan 6 graafi (leimatut nuolet). Tähän voidaan vielä lisätä ne nuolet, jotka saadaan puolittamalla yllä olevat sivut kokonaan mustalle ja kokonaan valkealle kuvalle. Merkitään näin saatuja nuolia katkoviivoilla. Näin on määräytynyt äärellinen automaatti, jonka alkutilaksi valitaan alkuperäistä kuvaa vastaava tila ◺. Konstruktiosta seuraa, että osaneliön, jonka osoite on sana w,



Kuva 6: Esimerkin 6 osaneliöitä yhdistävä graafi.

sisältämä kuva saadaan tilana, johon automaatti johtaa lukiessaan sanan w. Täten koko kuvan approksimaatio saadaan lukemalla kaikki tietynmittaiset sanat.

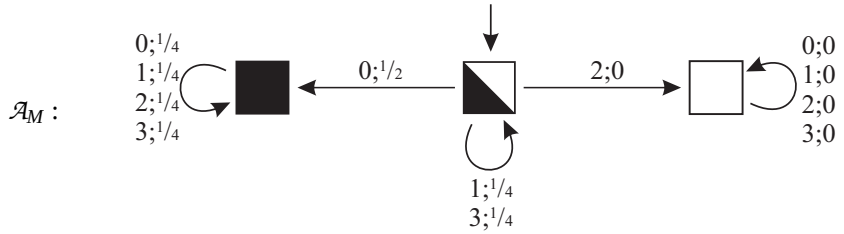
Yllä esiteltyyn automaattiin voidaan lisätä tummuuden skaalaus liittämällä transitioihin sopivat painot. Esimerkiksi ensimmäisessä jaossa puolet tummuudesta menee tilaan ■ ja $\frac{1}{4}$ tilaan ◺ kummallakin syötteellä. Näin saadaan $\mathcal{A}_d$:lle variantti, joka on kuvassa 7. Kyseessä on siis äärellinen automaatti, jossa transitiot on varustettu painoilla, niin kutsuttu *painotettu äärellinen automaatti*. Tässä esimerkiksi sanan 012 painoksi saadaan $\frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{4} = \frac{1}{32}$, mikä tarkoittaa, että kuvan väriaineesta yksi kolmaskymmeneskahdesosa on tässä ruudussa.

Painotettuja äärellisiä automaatteja kuvien käsittelyssä on tutkittu muun muassa töissä [23] ja [24]. Erikoisen kiintoisaa tässä on se, että kuvia voidaan useissa tapauksissa muokata muokkaamalla niiden tiivistettyä esitystä, siis kuvat koodavia automaatteja, katso [22].

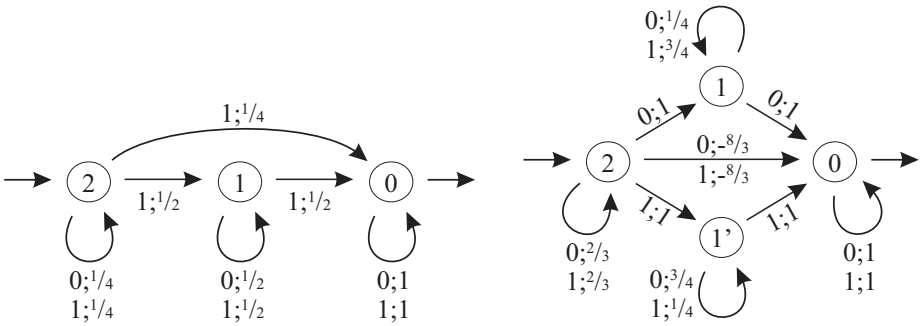
Esitämme lopuksi vielä pari esimerkkiä, jotka valottavat äärellisten automaattien tehokkuutta kuvien generoijina. Tarkastelemme tässä yksiulotteisia kuvia, siis funktioita $[0,1] \to [0,1]$. Funktion arvo annetussa pisteessä w kertoo kuvan tummuuden tässä pisteessä. Yksiulotteisessa tapauksessa yllä kuvattu neliön osittaminen vastaa tietysti janan jakamista vasempaan ja oikeaan puolikkaaseen. Tällöin siis argumentti w, toisin sanoen tarkasteltavan osajanan osoite, on binäärisana.

Esimerkki 7. Tarkastellaan kuvan 8 painotettuja automaatteja. Näissä esimerkeissä on käytetty lopputilaa. Siis syötteeseen w liittyvä arvo on kaikkien w:llä leimattujen alkutilasta lopputilaan johtavien polkujen painojen summa. Tässä siis polun paino on tämän polun transitioiden painojen tulo.

Kiintoisaa yllä olevassa esimerkissä



Kuva 7: Esimerkin 6 skaalattu versio.



Kuva 8: Kaksi painotettua automaattia.

on, että vasemmanpuoleinen laskee funktion $f(x) = x^2$, siis paraabelin, kun taas oikeanpuoleinen laskee välillä $[0,1]$ jatkuvan funktion, jolla ei ole derivaattaa missään pisteessä ([26])! Kuitenkin näiden, monessa mielessä varsin erilailla käyttäytyvien, funktioiden arvojen (likimääräinen) laskeminen automaatin avulla on laskennallisesti oleellisesti yhtä vaikeaa!

5 Katsaus sanojen kombinatoriikkaan

Tässä luvussa esitellään sanojen kombinatoriikan syntyä ja kehitystä sekä joitakin alan keskeisiä kysymyksiä ja saavutettuja huipputuloksia. Lisäksi tarkastellaan sanojen kombinatoriikkaa tietojenkäsittelyn näkökulmasta.

Teorian synty. Luonteenomaista sanojen kombinatoriikalle on sen moninaiset yhteydet eri tieteenaloihin. Yhteys tietojenkäsittelyyn on luonnollinen ja vahva, mutta ei suinkaan dominoiva. Yhteys fysiikkaan ilmenee muun muassa kvasikristallien ja diskreettien dynaamisten systeemien kautta, yhteys biologiaan puolestaan toteutuu DNA-jonon välityksellä. Niinikään monet sanojen kombinatoriikan kysymykset saavat motivaationsa matematiikan eri osa-alueilta, kuten esimerkiksi algebrasta ja todennäköisyyslaskennasta.

Yllä sanotusta johtuen sanojen kombinatoriikan synnyn jäljittäminen ei ole aivan helppoa: teoria on syntynyt useita kertoja eri probleemojen motivoimana. Vanhimpia merkittäviä sanoihin liittyviä tutkimuksia ovat A. Thuen 1900-luvun alussa julkaisemat työt toistovapaista ääerettömistä sanoista, vertaa esimerkki 1 ja [85]. Ne olivat hämmästyttävän laajoja ja poikkeuksellisen originelleja. Thuen motivaationa oli – ymmärtääkseni – vain tieteellinen uteliaisuus. Myöhemmin 1930-luvulla Morse ja Hedlund, vertaa [65], nyt symbolisen dynamiikan motivoimina, todistivat monien uusien merkittävien tulosten ohella osan Thuen tuloksista uudelleen. Kolmantena sanojen kombinatoriikan alkuna voidaan pitää P. Novikovin ja S. Adianin 1960-luvulla kehittämää ratkaisua Burnsiden probleemalle ryhmäteoriassa: yli 300-sivuinen ratkaisu sisälsi implisiittisesti huomattavan osan sanojen kombinatoriikan perustuloksista ([66]). Ja listaa voisi jatkaa.

Systemaattisesti tutkituksi tieteenalaksi sanojen kombinatoriikan nosti M. P. Schützenberger 1960-luvulla. Tähän liittyy läheisesti, mutta vain osana laajem-

paa teoriaa, *(vaihtelevanpituisten) koodien teorian* synty. Tutkimusta suuntaavia yleisiä merkittäviä töitä olivat muun muassa [55] ja [61]. Teoria laajeni ja kehittyi varsin nopeasti – alan tärkeimmät perustulokset kattava kirja "Combinatorics on Words" ilmestyi 1982 ([58]).

Kuluneiden 25 vuoden aikana sanojen kombinatoriikka on vakiinnuttanut paikkansa Euroopan johtavissa tietojenkäsittelyn teorian kongresseissa. Alan omaa kongressia – WORDS – on järjestetty vuodesta 1997 joka toinen vuosi. Vuonna 2003 kokous oli Turussa, ja viimeisin, yli 100 osanottajaa kerännyt tapahtuma järjestetiin Marseilles'ssa. Sanojen kombinatoriikan "käynnistänyt" monografia "Combinatorics on Words" on äskettäin saanut seurakseen jatko-osat "Algebraic Combinatorics on Words" ([59]) ja "Applied Combinatorics on Words" ([60]). Näistä jälkimmäinen keskittyy nimenomaan sanojen algoritmisiin ominaisuuksiin. Muita alan kokooma-artikkeleja ovat luku "Combinatorics of Words" ([14]) kirjassa Handbook of Formal Languages ja katsaus [5]. Sanojen kombinatoriikan historia on kuvailtu äskettäin työssä [6].

Tyypillisiä ongelmia. Tarkastelemme tässä lyhyesti kolmea tyypillistä sanojen kombinatoriikan tutkimuskohdetta.

Tutkimuskohde 1 (Toistovapaat sanat). Kuten mainittiin alan vanhimpia tuloksia ovat Thuen vuonna 1906 julkaisemat tulokset toistovapaista sanoista. Esimerkin 1 tuloksen ohella Thue todisti myös, että on olemassa ääretön sana 3-kirjaimisessa aakkostossa, joka ei pidä osasananaan neliötä, toisin sanoen on neliövapaa. Tällainen saadaan muun muassa Thuen sanasta α_T morfismin $\varphi : \{a,b,c\}^* \to \{a,b\}^*$, $\varphi(a) = abb$, $\varphi(b) = ab$ ja $\varphi(c) = a$ kantakuvana, siis

$$\varphi^{-1}(\alpha_T) = abcacbabcbacabca\ldots .$$

Monet Thuen tuloksista on uudelleenkeksitty moneen kertaan, johtuen siitä, että toistovapaat sanat ovat tärkeitä vastaesimerkkejä monissa sovelluksissa. Varsinainen renessanssi toistovapaiden sanojen tutkimuksessa alkoi vuonna 1979 J. Berstelin julkaisemasta työstä [4]. Saavutettuja tuloksia esitellään esimerkiksi kirjojen [58] ja [59] asiaa käsittelevissä luvuissa. Erikoismaininnan ansaitsee V. Keräsen vuonna 1990 keksimä ratkaisu eräälle – usein Erdösin probleeman nimellä tunnetulle – keskeiselle kysymykselle ([48]):

> *Onko olemassa 4-kirjaimisen aakkoston ääretöntä sanaa, joka ei sisällä välittömästi peräkkäisinä osasanoina kommutatiivisesti ekvivalentteja sanoja, toisin sanoen joka on* Abelin toistovapaa*?*

Tässä sanat ovat kommutatiivisesti ekvivalentteja, jos ne sisältävät kutakin kirjainta yhtä monta kappaletta. Vastaus on myönteinen, muttei lainkaan triviaali todistaa.

Tutkimuskohde 2 (Kommutointi ja defektilauseet). Sanojen kombinatoriikan yksi perustuloksista on karakterisaatio sille milloin kaksi sanaa kommutoi:

> *Sanat x ja y kommutoivat, toisin sanoen $xy = yx$, tarkalleen silloin, kun ne ovat saman sanan potensseja.*

Tulos lienee ollut tuttu – ainakin implisiittisesti – jo Thuelle, vaikkakin sen vanhinta todistusta on vaikea varmuudella löytää: se on todistettu muun muassa töissä [82], [55] ja [7]. Mainittu tulos on esimerkki tärkeästä defektilauseesta:

Jos n sanaa $x_1, \ldots, x_n$ toteuttaa eitriviaalin yhtälön voidaan ne ilmaista korkeintaan $n-1$ sanan tuloina, toisin sanoen on olemassa F niin, että $\{x_1, \ldots, x_n\} \subseteq F^*$ ja $card(F) \leq n-1$.

Defektilausetta voidaan pitää sanojen eräänlaisena – joskin heikkona – dimensio-ominaisuutena. Itse asiassa defektilauseella on useita hieman toisistaan poikkeavia formulaatioita, vertaa [14]. Vaikka defektilauseiden todistukset eivät ole hankalia, voidaan niihin liittyen formuloida avoimia ongelmia – kuten usein sanoihin liittyvien kysymysten yhteydessä. Tällainen on muun muassa:

Probleema 6 *Olkoon $X = \{x,y,z\} \subseteq A^+$ ja oletetaan, että X toteuttaa kolmen yhtälön riippumattoman yhtälöryhmän. Ovatko tällöin x, y ja z saman sanan potensseja?*

Probleema vaatii pari selventävää kommenttia. Ensinnäkin yhtälöiden oletetaan olevan vakiovapaita, siis koostuvan ainoastaan sanoista x, y ja z (tai tarkemmin ottaen näistä ajateltuina muuttujina). Toiseksi *riippumattomuus* tarkoittaa, että ryhmä ei ole ekvivalentti minkään osaryhmänsä kanssa, toisin sanoen näillä ei ole tarkalleen samoja ratkaisuja. Kolmanneksi lukua kolme ei voida korvata luvulla kaksi: yhtälöpari $xzy = yzx$ ja $xzzy = yzzx$ on riippumaton, mutta sanojen x, y ja z ei tarvitse olla saman sanan potensseja. Lopuksi vielä on syytä huomata, että probleema pyrkii formuloimaan kahden sanan kommutoinnin yleistyksen.

Tutkimuskohde 3 (Jaksollisuus). Jaksollisuus on sanojen, ja yleensäkin matemaattisen tutkimuksen, peruskäsitteitä. Sanojen kommutointi, kuten näimme, liittyi tähän läheisesti. Itse asiassa mainittu kommutointiin liittyvä tulos voidaan yleistää:

Jos sanoilla x^n ja y^m on pituutta $|x|+|y|-syt(|x|,|y|)$ oleva yhteinen alkuosa, ovat x ja y saman sanan potensseja.

Tuloksen todistivat N. Fine ja H. Wilf vuonna 1965 ([31]), ja sitä kutsutaan yleensä *Finen ja Wilfin jaksollisuuslemmaksi*. Heidän formulointinsa koski jaksollisia reaalifunktioita – eihän sanojen kombinatoriikka vielä tuolloin esiintynyt omana alanaan. Tuloksen luonnollinen ympäristö on kuitenkin sanojen maailmassa.

Jaksollisuuslemmassa mainittu raja – kuinka pitkälle alkuosien tulee yhtyä – on optimaalinen. Tässäkin esimerkin 1 Fibonaccin sanat F_4^ω ja F_5^ω toimivat vastaesimerkkinä:

$$\underbrace{abaababa.abaababa.aba}_{pituus\ =\ |F_4|+|F_5|-2}\,|\,ababa.aba\ldots$$
$$\underbrace{abaababa\ abaab.abaaba}\,|\,baabaab.a\ldots$$

Ylläolevat laskelmat voidaan yleistää päätelmäksi: peräkkäiset Fibonaccin sanat ovat optimaalisia yhteisen jakson väistäviä sanapareja. Tällaisiin sanoihin liittyvä viimeaikainen tutkimus on osoittautunut erittäin hedelmälliseksi ([25]). Sillä on läheiset yhteydet niin kutsuttuihin Sturmin sanoihin, katso luku 2 kirjassa [59].

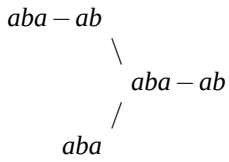
Jaksollisuuteen liittyviä muita fundamentaalisia tuloksia ovat niin kutsuttu kriittinen tekijöihinjakolause, katso [58] ja [59], sekä äskettäin todistettu hämmästyttävä karakterisaatio lopulta jaksollisille sanoille ([64]). Tarkastelemme jälkimmäistä hieman lähemmin. Aloitamme esimerkillä.

Esimerkki 8. Olkoon $X = \{ab, aba\}$, siis jälleen kaksi peräkkäistä Fibonaccin sanaa. Tarkastellaan X^ω:n sanoja, jotka on konstruoitu siten, että

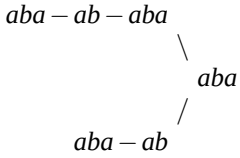
(i) *ab* ei esiinny kahta kertaa peräkkäin; ja

(ii) *aba* ei esiinny kolmea kertaa peräkkäin.

Selvästi tällaisia sanoja saadaan ylinumeroituva määrä, eikä niiden tarvitse olla – kuten on helppo nähdä – lopulta jaksollisia, siis muotoa uv^ω. Kuitenkin niillä on tietty lokaalinen jaksollisuusominaisuus: tällaisen sanan jokaisen vähintään kuuden mittaisen alkuosan lopussa on (jonkinmittainen) neliö. Tämä nähdään oheisista kuvioista, joissa sanoja *ab* ja *aba* on jatkettu kaikilla mahdollisilla tavoilla vasemmalle noudattaen sääntöjä (i) ja (ii):



ja



Tässä esimerkiksi jälkimmäisessä kuviossa *aba*:n *b*:hen päättyvä sana päättyy joko neliöön $(ab)^2$ tai $(abaab)^2$.

Esimerkin 8 tarkasteluja ei voida yleistää kuutioille. Nimittäin, jos alkuosat päättyvät aina kuutioon on sana lopulta jaksollinen. Tämä on erikoistapaus työssä [64] todistetusta perustavaalaatua olevasta karakterisaatiosta:

> *Ääretön sana w on lopulta jaksollinen jos ja vain jos jokainen riittävän pitkä w:n alkuosa päättyy vähintään kertalukua* $1+\varphi=\varphi^2$, *missä* $\varphi=(1+\sqrt{5})/2$, *olevaan toistoon.*

Jälleen esimerkin 1 sana α_F osoittaa, että yllä olevan tuloksen raja on optimaalinen. Intuitiivisesti tämä tulos voidaan tulkita täsmällisenä rajana milloin *ennustettavuus muuttuu kaoottisuudeksi*. Nimittäin tulos sanoo, että jos sanassa on tarpeeksi (mutta ei yhtään liikaa) lokaalista säännöllisyyttä, on se myös globaalisti säännöllinen eli jaksollinen ja siten ennustettava.

Yllä oleva tulos ei rajoita mitenkään alkuosien lopussa esiintyvien toistojen pituuksia, vaan ainoastaan niiden kertaluvun, toisin sanoen sen, montako kertaa sanaa toistetaan. Jos myös toistojen pituudet rajoitetaan tullaan uusiin ongelmiin, jotka on tyhjentävästi selvitetty A. Lepistön työssä [56].

Yhteys tietojenkäsittelyyn ja algoritmiikkaan. Sanat ovat keskeinen osa tietojenkäsittelyn peruskäsitteistä. Jokainen automaattien teorian kirja alkaa sanojen perusmääritelmillä. Toisaalta sanojen kombinatoriset ominaisuudet ovat usein vain kovin pinnallisesti käytössä automaattien ja laskettavuuden teoriassa. Ensimmäinen automaattien teorian oppikirja, jossa esiteltiin sanojen kombinatoriikkaa lähemmin on M. Harrisonin kirja “Introduction to Formal Language Theory” [36]. Tilanne muuttuu kun siirrytään algoritmiikkaan, ja etenkin merkkijonoalgoritmeihin. Tästä Crochemoren ja Rytterin kirja “Text Algorithms” [19] on oivallinen esimerkki.

Mallinsovitusongelmat, siis kysymykset, joissa on selvitettävä pitääkö annettu teksti (lue: sana) sisällään annettua mallia (lue: osasanaa), ovat selkeitä algoritmisia probleemoja, joissa tehokkaat algoritmit nojaavat sanojen jaksollisuusominaisuuksiin, vertaa [50] ja [8] tai näiden tulosten tarkastelu kirjassa [19]. Itse asiassa, aina kun on todistettu jokin sanojen jaksollisuuteen liittyvä perustulos, kuten Finen ja Wilfin jaksollisuuslemma, kriittinen tekijöihinjakolause tai yllä mainittu ääret-

tömien sanojen lopulta jaksollisuuden karakterisoiva tulos, näiden seurauksena on saatu myös joiltakin osin parannettu tai ainakin uusi merkkijonoalgoritmi, vertaa esimerkiksi [18] tai [64].

Tarkastelimme esimerkissä 2 primitiivisiä sanoja, ja esitimme niille karakterisaation. Karakterisaatio palauttaa kysymyksen "Onko annettu sana primitiivinen?" hahmonsovitusprobleemaan "Esiintyykö w osasanana sanassa ww?" ja voidaan siten ratkaista lineaariajassa tunnetuilla hahmonsovitusalgoritmeilla. Suoraan sanojen ominaisuuksiin perustuva lineaariaikaisen algoritmin konstruointi ei ole lainkaan ilmeinen.

Sanojen kombinatoriikka, Combinatorics on Words, on äskettäin luokiteltu referaattisarjassa Mathematical Reviews omaksi kokonaisuudekseen, joka on sijoitettu lukuun Computer Science alaotsakkeen Discrete Mathematics Related to Computer Science alle. Tämäkin korostaa sanojen merkitystä tietojenkäsittelyssä.

Teorian huippukohtia ja haasteita. Vaikka sanojen kombinatoriikka on yhä varsin nuori tutkimuskohde, ovat eräät sen tulokset merkittäviä laajemminkin matemaattisen tutkimuksen historiassa. Mainitsemme seuraavassa kaksi esimerkkiä.

Sanayhtälön ratkeavuus. Probleeman määrittelemiseksi olkoon Ξ äärellinen joukko muuttujia ja Σ annettu äärellinen aakkosto. Sanayhtälö on pari (u,v), tavallisesti kirjoitettuna muodossa $u = v$, missä u ja v ovat aakkoston $\Xi \cup \Sigma$ sanoja. Yhtälön ratkaisu on morfismi $\varphi : (\Xi \cup \Sigma)^* \to \Sigma^*$, joka kiinnittää vakiot $a \in \Sigma$, toisin sanoen $\varphi(a) = a$ ja toteuttaa $\varphi(u) = \varphi(v)$. Probleemana on selvittää onko annetulla sanayhtälöllä lainkaan ratkaisua. Yksi viime vuosikymmenien suuria diskreetin matematiikan saavutuksia on G. S. Makaninin vuonna 1976 todistama tulos [62]:

> *Sanayhtälön ratkeavuus on algoritmisesti selvitettävissä.*

Yllä sanayhtälöiden sallitaan sisältävän vakioita $a \in \Sigma$, jotta probleema ei olisi triviaali: muuten olisi aina triviaaliratkaisu, jossa muuttujat saavat arvon 1.

Makaninin ratkaisua on usein pidetty sanojen kombinatoriikan syvällisimpänä tuloksena. Alkuperäisen algoritmin kompleksisuus oli tähtitieteellinen. Äskettäin W. Plandowski esitti probleemalle kokonaan uuden ratkaisun [68]:

> *Sanayhtälön ratkeavuus on luokassa* **PSPACE**.

Probleema siis pysyy yhä käytännössä ratkeavien probleemojen ulkopuolella (teoreetikkojen arvomaailmassa), mutta ei ole sen vaikeampi kuin esimerkiksi äärellisten epädeterminististen automaattien ekvivalenssiprobleema, joka on niinikään luokassa **PSPACE**. Haastava avoin kysymys on:

Probleema 7 *Onko sanayhtälön ratkeavuusongelma luokassa* **NP***, tai ekvivalentisti* **NP***-täydellinen?*

Probleeman ekvivalentti formulointi seuraa siitä, että sanayhtälöiden ratkeavuusongelma on yleistys tunnetusta **NP**-vaikeasta ongelmasta.

Ehrenfeuchtin kompaktisuusominaisuus. Myös toinen tässä esiteltävä sanojen perustulos käsittelee sanayhtälöitä, nyt kuitenkin yhtälöryhmiä, missä yhtälöt ovat myös vakiovapaita. Sanomme, että kaksi yhtälöryhmää S ja S' ovat *ekvivalentit* jos niillä on tarkalleen samat ratkaisut Σ^*:ssa. A. Ehrenfeucht esitti 1970-luvun alussa otaksuman, että

> *Mielivaltainen yhtälöryhmä S Σ^*:ssä on ekvivalentti jonkin äärellisen aliyhtälöryhmansä S_0 kanssa.*

Yhtälöita siis on yleisesti ääretön määrä, mutta, luonnollisesti, niiden sallitaan sisältävän muuttujia vain kiinteästä äärellisestä joukosta Ξ. Ehrenfeuchtin otaksuma siis formuloi sanoille syvällisen kompaktisuusominaisuuden. Ja osoittautui, että otaksuma on oikea:

Ehrenfeuchtin otaksuma on tosi.

Todistuksen esittivät samanaikaisesti M. Albert ja J. Lawrence [2] sekä G. Guba [34]. Molemmat todistukset perustuvat jo yli 100 vuotta vanhaan niin kutsuttuun Hilbertin kantalauseeseen. Guban todistuksessa keskeistä sijaa näyttelee tunnettu tosiasia, että sanamonoidi A^* voidaan upottaa multiplikatiiviseen kokonaislukualkioisten 2×2 matriisien puoliryhmään $M_{2\times 2}(\mathbb{Z})$, toisin sanoen on olemassa injektiivinen morfismi

$$i : A^* \hookrightarrow M_{2\times 2}(\mathbb{Z}). \qquad (6)$$

Yhteys (6) on monessa suhteessa tärkeä. Tässä se antaa mahdollisuuden käyttää lukujen ja matriisien tuloksia sanojen perusominaisuuksien selvittämiseen. (Katso myös "Huomioita oikoluvun yhteydessä" kirjoituksen lopussa.) Myös (6):n käänteinen käyttö on usein hedelmällistä: sanoille tyypilliset ratkeamattomuustulokset voidaan kääntää eräiksi matriiseja koskeviksi ratkeamattomuustuloksiksi. Jo mainittu tulos, joka sanoo, että äärellisesti generoitujen multiplikatiivisten matriisipuoliryhmien isomorfisuusprobleema on ratkeamaton, on tällainen.

Toisaalta (6) tarjoaa myös runsaasti avoimia kysymyksiä. Konkreettinen ja äärimmäisen yksinkertaisen tuntuinen on (katso [13] ja myös lukua "Huomioita oikoluvun yhteydessä"):

Probleema 8 *Generoivatko matriisit*

$$\begin{pmatrix} 2/3 & 0 \\ 0 & 1 \end{pmatrix} \textit{ja} \begin{pmatrix} 3/5 & 1 \\ 1 & 1 \end{pmatrix}$$

vapaan multiplikatiivisen puoliryhmän?

Toinen, ja moninverroin keskeisempi, avoin kysymys on:

Probleema 9 *Voidaanko Ehrenfeuchtin kompaktisuustuloksen äärellisen alisysteemin S_0 kokoa jotenkin rajoittaa muuttujien lukumäärän funktiona?*

Tästä yleisestä probleemasta tiedetään, että S_0 voi olla kertalukua $\Omega((\mathrm{card}(\Xi))^4)$ ([45] tai [14]).

Sanoista äärellisiin sanajoukkoihin. Kuten näimme, sanojen teoria tunnetaan monilta osin varsin hyvin, vaikkakin eräät keskeiset kysymykset ovat yhä selvittämättä. Siirryttäessä sanoista äärellisiin sanajoukkoihin, siis äärellisiin kieliin, tilanne muuttuu oleellisesti: tietyistä kysymyksistä ei tiedetä juuri mitään. Havainnollistamme tätä parilla esimerkillä.

Tarkastellaan yhtälöitä

$$xy = yx \qquad \text{ja} \qquad xz = zy.$$

Näistä edellinen, *kommutointiyhtälö*, määrittelee sen milloin alkiot x ja y kommutoivat ja toinen, *konjugaattiyhtälö*, sen milloin x ja y ovat *konjugaatteja*, eli toteuttavat tämän yhtälön jollakin z:n arvolla. Sanojen osalta molempien yhtälöiden täydelliset ratkaisut tunnetaan, eivätkä ne ole kovin hankalia. Yhtälöitä voidaan tarkastella myös kieliyhtälöinä. Tällöin kommutointitapauksessa tiedetään jotakin, katso esimerkiksi [42] ja [44], mutta ei esimerkiksi karakterisaatiota milloin kaksi äärellistä kieltä kommutoi – ja tämä voikin olla hyvin hankala kysymys. Konjugaattiyhtälöstä ei tiedetä juuri mitään. Jopa algoritminen peruskysymys, ovatko kaksi äärellistä kieltä konjugaatteja, on ratkaisua vailla.

Mainitut ja vastaavat äärellisiä kieliä koskevat kysymykset näyttävät erittäin

vaikeilta. Tätä tukee muun muassa oheinen äskettäin todistettu ratkeamattomuustulos [43]:

> *On algoritmisesti ratkeamatonta, ovatko annetut äärelliset sijoitukset* $\varphi, \psi : \{a,b,c\}^* \to \{0,1\}^*$ *ekvivalentit kielellä* ab^*c*, toisin sanoen, onko voimassa* $\varphi(ab^ic) = \psi(ab^ic)$ *kaikilla i:n arvoilla?*

Tässä siis kunkin kirjaimen x, ja myös sanan, kuvat $\varphi(x)$ ja $\psi(x)$ ovat äärellisiä kieliä. Tuloksen helppona seurauksena saadaan äärellisten transduktorien ekvivalenssiprobleemalle hämmästyttävä terästys:

> *2-tilaisten yleistettyjen jonokoneiden, joiden syöteaakkosto on unaarinen, ekvivalenssiprobleema on ratkeamaton.*

Huomioita oikoluvun yhteydessä

Artikkeli on alkujaan kirjoitettu runsaat viisi vuotta sitten. Kuitenkin siihen nyt tehdyt päivitykset ovat kovin vähäiset. Tekstissä on korjattu ja täsmennetty vain joitakin kohtia, lähinnä luvun 5 kohdan "Teorian synty" viimeinen kappale on muokattu tämänpäivän tilannetta vastaavaksi.

Joillakin artikkelissa tarkastelluilla aloilla on saavutettu merkittäviä tuloksia kuluneiden viiden vuoden aikana. Tämän artikkelin kannalta keskeisiä saavutuksia ovat seuraavat. Kieliyhtälöiden kommutointiin, siis työn viimeiseen teemaan, liittyen M. Kunc [53] osoitti hämästyttävän tuloksen, jonka mukaan maksimaalinen kieli, joka kommutoi annetun äärellisen kielen kanssa, ei ole välttämättä edes rekursiivisesti numeroituva. Täten hän ratkaisi yli 30 vuotta vanhan niin kutsutun *Conwayn ongelman*, joka kysyi, onko tällainen kieli aina säännöllinen (jopa lähtien annetusta säännöllisestä kielestä). Niinikään hän osoitti äskettäin työssään [54], että yllä mainitusta äärellisiä sijoituksia koskevasta vahvasta ratkeamattomuustuloksesta voidaan jättää "reunamerkki" c pois ratkeamattomuuden muuttumatta. Monessa mielessä yhtä yllättävä on J. Cassaignen [10] probleemalle 8 antama ratkaisu: puoliryhmä ei ole vapaa – lyhin ei-triviaali identiteetti koostuu 22 mittaisista tuloista. Kiintoisaa tässä on, että tulojen pituus on juuri niillä rajoilla, mikä voidaan vielä löytää käytännössä tietokoneella.

Kiitokset

Tekijä on kiitollinen lehden toimittajalle Antti Valmarille erittäin huolellisesta artikkeliin paneutumisesta ja lukuisista arvokkaista täsmennyksistä.

Viitteet

[1] L. Adleman, Molecular computation of solutions to combinatorial problems, Science **226**, 1021–1024, 1994.

[2] M.H. Albert and J. Lawrence, A proof of Ehrenfeucht's Conjecture, Theoret. Comput. Sci. **41**, 121–123, 1985.

[3] R. Berger, The undecidability of the domino problem, Mem. Amer. Math. Soc. **66**, 1966.

[4] J. Berstel, Mots sans carré et morphismes itérés, Discr. Math. **29**, 235–244, 1979.

[5] J. Berstel and J. Karhumäki, Combinatorics on words – a tutorial, in: G. Paun, G. Rozenberg and A. Salomaa (eds), *Current Trends in Theoretical Computer Science, The Challenge of the New Century*, World Scientific, 415–476, 2004.

[6] J. Berstel and D. Perrin, The origins of combinatorics on words, European J. Combin. **28**, 996–1022, 2007.

[7] E.K. Blum, A note on free semigroups with two generators, Bull. AMS **71**, 678–679, 1965.

[8] R.S. Boyer and J.S. Moore, A fast string-searching algorithm, Comm. ACM **20**, 762–772, 1977.

[9] R. Büchi, Weak second-order arithmetic and finite automata, Z. Math. Logic Grundl. Math. **6**, 66–92, 1960.

[10] J. Cassaigne, personal communication, 2007.

[11] J. Cassaigne, T. Harju, and J. Karhumäki, On the undecidability of freeness of matrix semigroups, Inter. J. Alg. and Comput. **9**, 295–305, 1999.

[12] A.K. Chandra, D. Kozen, and L. Stockmayer, Alternation, J. ACM **28**, 114–133, 1981.

[13] C. Choffrut, T. Harju, and J. Karhumäki, A note on decidability questions on presentations of word semigroups, Theoret. Comput. Sci. **183**, 83–92, 1997.

[14] C. Choffrut and J. Karhumäki, Combinatorics of Words, in: G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages*, Vol. 1, Springer, 329–438, 1997.

[15] N. Chomsky and M.P. Schützenberger, The algebraic theory of context free languages, in: P. Braffort and D. Hirschberg (eds.), *Computer Programming and Formal Systems*, North Holland, 118–161, 1963.

[16] S. Cook, The complexity of theorem-proving procedures, FOCS, 151–156, 1971.

[17] M. Crochemore, F. Mignosi, A. Restivo, and S. Salemi, Text compression using anti-dictionaries, Springer LNCS **1644**, 261–270, 1999.

[18] M. Crochemore and D. Perrin, Two-way string matching, J. ACM **38**, 651–675, 1991.

[19] M. Crochemore and W. Rytter, *Text Algorithms*, Oxford University Press, 1994.

[20] K. Culik II, An aperiodic set of 13 tiles, Discr. Math. **160**, 245–251, 1996.

[21] K. Culik II and I. Fris, The decidability of the sequence equivalence problem for D0L languages, Inform. and Control **35**, 20–39, 1977.

[22] K. Culik II and I. Fris, Weighted finite transducers in image processing, Discr. Appl. Math. **58**, 223–237, 1995.

[23] K. Culik II and J. Karhumäki, Finite automata computing real functions, SIAM J. Comput. **23**, 789–814, 1994.

[24] K. Culik II and J. Kari, Digital Images and Formal Languages, in: A. Salomaa and G. Rozenberg (eds.), *Handbook of Formal Languages*, Vol. 3, 599–616, Springer-Verlag, 1997.

[25] A. de Luca, Sturmian words: structure, combinatorics, and their arithmetics, Theoret. Comput. Sci. **183**, 45–82, 1997.

[26] D. Derencourt, J. Karhumäki, M. Latteux, and A. Terlutte, On continuous functions computed by finite automata, Theor. Inform. and Appl. **28**, 387–403, 1994.

[27] A. Ehrenfeucht, R. Parikh, and G. Rozenberg, Pumping lemmas for regular languages, SIAM J. Comput. **10**, 536–541, 1981.

[28] A. Ehrenfeucht and G. Rozenberg, Elementary homomorphisms and a solution to the D0L sequence equivalence problem, Theoret. Comput. Sci. **7**, 169–183, 1978.

[29] S. Eilenberg, *Automata, Languages and Machines*, Vol. A, Academic Press, New York, 1974.

[30] D. Epstein, J. Cannon, D. Holt, S. Levy, M. Paterson, and W. Thurston, *Word processing in groups*, Jones and Bartlett, Boston, 1992.

[31] N.J. Fine and H.S. Wilf, Uniqueness theorem for periodic functions, Proc. Am. Math. Soc. **16**, 109–114, 1965.

[32] M.R. Garey and D.S. Johnson, *Computers and Intractrability: A Guide to the Theory of* **NP***-Completeness*, W.H. Freeman, 1979.

[33] F. Gecseg and M. Steinby, *Tree Automata*, Akademiai Kiado, 1984.

[34] V.S. Guba, The equivalence of infinite systems of equations in free groups and semigroups with finite subsystems, Mat. Zametki **40**, 321–324, 1986. In Russian.

[35] T. Harju and J. Karhumäki, The equivalence problem for multitape automata, Theoret. Comput. Sci. **78**, 347–355, 1991.

[36] M. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley, 1978.

[37] M. Hirvensalo, *Quantum Computing*, Springer, 2001. Second edition, 2004.

[38] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.

[39] J. Hromkovic and G. Schnitger, On the power of Las Vegas for one-way communication complexity, OBDD's, and finite automata, Inf. and Comput. **169**, 284–296, 2001.

[40] N. Immerman, Nondeterministic space is closed under complementation, SIAM J. Comp. **17**, 935–938, 1981.

[41] J. Karhumäki, The impact of the D0L problem, in: G. Rozenberg and A. Salomaa (eds.), Current Trends in Theoretical Computer Science, World Scientific, 586–594, 1993.

[42] J. Karhumäki, Challenges of commutation, an advertisement, Springer LNCS **2138**, 15–83, 2001.

[43] J. Karhumäki and L.P. Lisovik, A surprising undecidability result: The equivalence problem for finite substitutions on ab^*c, Int. J. Found. Comput. Sci. **14**, 699–710, 2003.

[44] J. Karhumäki and I. Petre, Two problems in commutation of languages, in: G. Paun, G. Rozenberg and A. Salomaa (eds.), *Current Trends in Theoretical Computer Science, The Challenge of the New Century*, World Scientific, 477–494, 2004.

[45] J. Karhumäki and W. Plandowski, On the size of independent systems of equations in semigroups, Theoret. Comput. Sci. **168**, 105–119, 1996.

[46] J. Kari, A small aperiodic set of Wang tiles, Discr. Math. **160**, 259–264, 1996.

[47] R. Karp, Reducibility among combinatorial problems, in: J. Thatcher and R. Miller (eds.), *Complexity of Computer Computations*, Plenum Press, 85–103, 1972.

[48] V. Keränen, Abelian squares are avoidable on 4 letters, Springer LNCS **623**, 41–52, 1992.

[49] S. Kleene, Representation of events in nerve nets and finite automata, in: C.E. Shannon and J. McCarthy (eds.), *Automata Studies*, Princeton University Press, 3–42, 1956.

[50] D. Knuth, D.E. Morris, and V.R. Pratt, Fast pattern matching in strings, SIAM J. Comput. **6**, 323–350, 1977.

[51] D. Kozen, *Automata and Computability*, Springer, 1997.

[52] M. Kudlek and Y. Rogozhin, New small universal circular Post Machines, Springer LNCS **2138**, 217–226, 2001.

[53] M. Kunc, The power of commuting with finite sets of words, Theor. Comput. Syst. **40**, 521–551, 2007.

[54] M. Kunc, The simplest language where the equivalence of finite substitutions is undecidable, Springer LNCS **4639**, 365–375, 2007.

[55] A. Lentin and M.P. Schützenberger, A combinatorial problem in the theory of free monoids, in: R.C. Bose and T.E. Dowling (eds.), Combinatorial Mathematics, North Carolina Press, Chapel Hill, 112–144, 1967.

[56] A. Lepistö, On Relations Between Local and Global Periodicity, Ph.D. thesis, TUCS Dissertations 43, 2002.

[57] A. Lindenmayer, Mathematical models of cellular interaction in development, I and II, J. Theoret. Biol. **18**, 280–315, 1968.

[58] M. Lothaire, *Combinatorics on Words*, Addison-Wesley, 1983.

[59] M. Lothaire, *Algebraic Combinatorics on Words*, Cambridge University Press, 2002.

[60] M. Lothaire, *Applied Combinatorics on Words*, Cambridge University Press, 2005.

[61] R. Lyndon and M.P. Schützenberger, The equation $a^m = b^n c^p$ in a free group, Michigan Math J. **9**, 289–298, 1962.

[62] G.S. Makanin, The problem of solvability of equation in a free semigroup, Mat. Sb. **103**, 147–236, 1977 (English transl. in Math USSR Sb. **32**, 129–198).

[63] S. Micali, Two-way deterministic finite automata are exponentially more succinct than sweeping automata. Inform. Proc. Letters **12**(2), 103–105, 1981.

[64] F. Mignosi, A. Restivo, and S. Salemi, A periodicity theorem on words and applications, Springer LNCS **969**, 337–348, 1995.

[65] M. Morse and G. Hedlund, Symbolic dynamics, Am. J. Math. **60**, 815–866, 1938.

[66] P.S. Novikov and S.I. Adian, On infinite periodic groups, Izvestiya of the Academy of Sciences of the USSR **32**, 212–244, 251–524, 709–731, 1968.

[67] Gh. Paun, G. Rozenberg, and A. Salomaa, *DNA Computing, New Computing Paradigms*, Springer, 1998.

[68] W. Plandowski, Satisfiability of word equations with constants is in **PSPACE**, Proc. of FOCS, 495–500, 1999.

[69] E. Post, A variant of a recursively unsolvable problem, Bull. AMS **52**, 264–268, 1946.

[70] M. Rabin and D. Scott, Finite Automata and their decision problems, IBM J. Res. **3**, 115–125, 1959.

[71] G. Rozenberg, Gene Assembly in Ciliates: Computing by Folding and Recombination, in: A. Salomaa, D. Wood, S. Yu (eds.), *A half-century of automata theory: Celebration and Inspiration*, World Scientific, 93–130, 2001.

[72] G. Rozenberg and A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, 1988.

[73] G. Rozenberg and A. Salomaa (eds.) *Handbook of Formal Languages*, Vol.I–II–III. Springer-Verlag, 1997.

[74] A. Salomaa, Formal Languages, Academic Press, 1973.

[75] A. Salomaa and M. Soittola, *Automata-Theoretic Aspects of Formal Power Series*, Springer, 1978.

[76] A. Salomaa, D. Wood, and S. Yu (eds.), *A Half-Century of Automata Theory*, World Scientific, 2001.

[77] W. Savitch, Relationship between nondeterministic and deterministic tape classes, JCSS **4**, 177–192, 1970.

[78] G. Senizergues, L(A)=L(B)? decidability results from formal systems, Theoret. Comput. Sci. **251**, 1–166, 2001.

[79] J.C. Shepherdson, The reduction of two-way automata to one-way automata, IBM J. Res. **3**, 198–200, 1959.

[80] P. Shor, Algorithms for quantum computation: discrete log and factoring, FOCS, 20–22, 1994.

[81] M. Sipser, *Introduction to the Theory of Computation*, PWS Publishing Company, 1997.

[82] D. Skordev and Bl. Sendov, On equations in words, Z. Math. Logic Grundlagen Math. **7**, 289–297, 1961.

[83] R. Szelepcsenyi, The method of forcing for nondeterministic automata, Acta Informatica **26**, 279–284, 1988.

[84] W. Thomas, Languages, Automata and Logic, in: A. Salomaa and G. Rozenberg (eds.), *Handbook of Formal Languages*, Vol. 3, 389–456, Springer-Verlag, 1997.

[85] A. Thue, Über unendliche Zeichenreihen, Kra. Vidensk. Selsk. Skrifter. I. Mat.-Nat. Kl., Christiana, Nr. **7**, 1906.