FORMAALIT KIELET

Keijo Ruohonen

2008

Sisältö

Esipuhe

Tämä moniste on tarkoitettu TTY:n kurssin „MAT-41180 Formaalit kielet" perusmateriaaliksi. Monisteessa käydään läpi peruskonstruktiot nimenomaan kielten ja kielioppien kannalta. Rinnakkaiskurssissa „MAT-41176 Automaattiteoria" käsitellään osin samaa asiaa automaattien, vaativuuden ja laskettavuuden kannalta.

Formaalit kielet ovat saaneet alkunsa matematiikan symbolisesta merkintäformalismista, erityisesti kombinatoriikassa ja symbolisessa logiikassa. Myöhemmin kuvaan tulivat mukaan myös erilaiset tiedon salauksessa, siirrossa, pakkauksessa ja virheiden korjauksessa tarvittavat koodit—joilla kaikilla on ollut oma vaikutuksensa myös formaalien kielten teoriaan—sekä erityisesti laskennan teorian erilaiset mallit.

Kuitenkin vasta Noam Chomskyn uraauurtavat ideat luonnollisten kielten tutkimuksessa sekä Marcel-Paul Schützenberger'n algebrallis-kombinatorinen lähestymistapa antoivat formaalien kielten teorialle varsinaisen sysäyksen. Myös ohjelmointikielillä on ollut vahva vaikutus. Teorian „kulta-aikana", 1960- ja 1970-luvuilla, luotiin paljolti pohja sen nykyiselle muodolle.[1] Nykyään formaalien kielten perusteiden voi sanoa vakiintuneen melko yhtenevään muotoon, mikä näkyy vertailtaessa uusia ja vanhempia oppikirjojakin.

Monisteessa esitetään klassinen Chomsky-tyyppinen kielten teoria, kuitenkin jättäen automaattikonstruktiot ja erityisesti laskettavuus vähemmälle. Mukana ovat myös katsaukset matemaattiseen kooditeoriaan sekä Lindenmayerin systeemeihin. Jossain määrin valtakirjallisuudesta poikkeavasti esitetään myös katsaus formaaleihin potenssisarjoihin, jotta nähtäisiin tapoja yleistää kielen käsite kuitenkin pitäen sen sanarakenne samana.[2]

Keijo Ruohonen

[1] Alan huippunimistä mainittakoon erityisesti suomalainen akateemikko Arto Salomaa.

[2] Tapoja yleistää kieli muuttamalla sanarakennetta, esimerkiksi graafiksi tai kuvaksi tai moniulotteiseksi, ei tässä käsitellä.

Luku 1

SANAT JA KIELET

1.1 Sanat ja aakkostot

Sana on äärellinen jono termejä, ns. *symboleja* tai *kirjaimia,* jotka voidaan valita tietystä äärellisestä joukosta, ns. *aakkostosta.* Tavallisia aakkostoja ovat esimerkiksi suomenkielen kirjaimet (+ sanaväli, välimerkit jne.) tai bitit 0 ja 1. Yhden pituinen sana, jossa on siis vain yksi symboli, samaistetaan tähän symboliin. Erityinen sana on ns. *tyhjä sana,* jossa ei ole lainkaan symboleja, merkitään Λ (tai λ tai ε tai 1).

Sanan w *pituus* on siinä olevien symbolien lukumäärä, merkitään $|w|$. Tyhjän sanan pituus on 0. Jos aakkostossa on k kirjainta, on n-pituisia sanoja kaikkiaan k^n kpl. Enintään n-pituisia sanoja on siis

$$\sum_{i=0}^{n} k^i = \frac{k^{n+1}-1}{k-1} \quad \text{kpl},$$

jos $k > 1$, ja $n+1$ kpl, jos $k = 1$. Kaikkiaan sanoja on numeroituvasti ääretön määrä eli ne voidaan kirjoittaa listaksi, vaikkapa ensin pituuden mukaan.

Sanojen perusoperaatio on ns. *katenaatio,* ts. sanojen kirjoittaminen peräkkäin eli yhdyssanaksi. Sanojen w_1 ja w_2 katenaatiota merkitään w_1w_2:lla. Esimerkkejä aakkostossa $\{a, b, c\}$:

$$\begin{array}{lcll} w_1 = aacbba & , & w_2 = caac & , \quad w_1w_2 = aacbbacaac \\ w_1 = aacbba & , & w_2 = \Lambda & , \quad w_1w_2 = w_1 = aacbba \\ w_1 = \Lambda & , & w_2 = caac & , \quad w_1w_2 = w_2 = caac \end{array}$$

Katenaatio on liitännäinen eli assosiatiivinen, ts.

$$w_1(w_2w_3) = (w_1w_2)w_3.$$

Tämän seurauksena toistetut katenaatiot voidaan aina kirjoittaa ilman sulkuja. Katenaatio ei yleensä ole vaihdannainen eli kommutatiivinen. Yleensä siis

$$w_1w_2 \neq w_2w_1.$$

Mutta ei toki aina, ja yksikirjaimisessa aakkostossa katenaatio on tietysti vaihdannainen.

Sanan w ns. n:s (*katenaatio*)*potenssi* on

$$w^n = \underbrace{ww\cdots w}_{n \text{ kpl}}.$$

Erikseen määritellään $w^1 = w$ ja $w^0 = \Lambda$, ja aina $\Lambda^n = \Lambda$.

Sanan $w = a_1 a_2 \cdots a_n$ *peilikuva* on sana

$$\hat{w} = a_n \cdots a_2 a_1,$$

erityisesti $\hat{\Lambda} = \Lambda$. Ilmeisesti $\widehat{w_1 w_2} = \hat{w}_2 \hat{w}_1$. Sana u on sanan w *alkusana* eli *prefiksi* (vast. *loppusana* eli *suffiksi*), jos $w = uv$ (vast. $w = vu$) jollekin sanalle v. Sana u on sanan w *osasana,* jos $w = v_1 u v_2$ joillekin sanoille v_1 ja v_2. Sana u on sanan w *harva osasana,* jos

$$w = w_1 u_1 w_2 u_2 \cdots w_n u_n w_{n+1},$$

missä $u = u_1 u_2 \cdots u_n$, joillekin sanoille $w_1, w_2, \ldots, w_{n+1}$ ja $u_1, u_2, \ldots, u_n$.

1.2 Kielet

Kieli on joukko jonkin aakkoston sanoja. Erityisiä kieliä ovat *äärelliset kielet,* joissa on vain äärellinen määrä aakkoston sanoja, *vastaäärelliset kielet,* joista puuttuu vain äärellinen määrä aakkoston sanoja, ja *tyhjä kieli* $\emptyset$, jossa ei ole sanoja lainkaan. Usein samaistetaan yhden sanan w muodostama kieli $\{w\}$ sanaan w ja merkitään myös kieltä w:llä.

Kielille käytetään joukko-opin merkintöjä: $\subseteq$ (sisältyminen), $\subset$ (aito sisältyminen), $\cup$ (yhdiste), $\cap$ (leikkaus), $-$ (erotus) ja $\overline{}$ (komplementti aakkoston kaikkien sanojen suhteen). Sanan w kuulumista kieleen L merkitään $w \in L$:llä. Huomaa myös „negeeratut" versiot $\not\subseteq$, $\not\subset$ ja $\notin$.

Aakkoston Σ kaikkien sanojen muodostamaa kieltä, mukana Λ, merkitään Σ^*:llä. Kaikkien muiden Σ:n sanojen paitsi tyhjän sanan Λ muodostamaa kieltä merkitään Σ^+:lla. Siispä $\overline{L} = \Sigma^* - L$ ja $\Sigma^+ = \Sigma^* - \{\Lambda\}$.

Lause 1. *Aakkoston kieliä on ylinumeroituvasti ääretön määrä, ts. niitä ei voida kirjoittaa listaksi.*

Todistus. Tehdään vastaoletus: Kaikki kielet (yli jonkin aakkoston Σ) voidaan kirjoittaa listaksi $L_1, L_2, \ldots$ Muodostetaan nyt kieli L seuraavasti: Kirjoitetaan kaikki aakkoston Σ sanat listaksi $w_1, w_2, \ldots$ Valitaan sana w_i kieleen L tarkalleen silloin, kun se ei ole kielessä L_i. Ilmeisesti kieli L ei silloin ole mikään listan $L_1, L_2, \ldots$ kielistä. Vastaoletus on näin ollen väärä. □

Lauseen todistuksessa esiintyvä menettely on esimerkki ns. *lävistäjämenetelmästä.* Erilaisia tapoja, joilla kieliä voidaan määritellä, on toisaalta enintään numeroituvasti äärellinen määrä, sillä määritelmät voidaan esittää sanallisesti ja laittaa sen jälkeen aakkosjärjestykseen. Formaalien kielten teoriassa kielten määritelmät ja kielten tutkiminen niiden määritelmien kautta on keskeistä, joten vain (mitättömän pieni) osa kaikista mahdollisista kielistä tulee käsittelyn piiriin!

Kielille voidaan määritellä muitakin operaatioita kuin ym. joukko-opilliset operaatiot. Kielten L_1 ja L_2 *katenaatio* on

$$L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1 \text{ ja } w_2 \in L_2\}.$$

Kielen L ns. n:s (*katenaatio*)*potenssi* on

$$L^n = \{w_1 w_2 \cdots w_n \mid w_1, w_2, \ldots, w_n \in L\},$$

erityisesti $L^1 = L$ ja $L^0 = \{\Lambda\}$. Myös $\emptyset^0 = \{\Lambda\}$! Kielen L ns. *katenaatiosulkeuma* on

$$L^* = \bigcup_{n=0}^{\infty} L^n,$$

ts. muodostetaan sanoja katenoimalla kaikin tavoin L:n sanoja, mukana myös Λ. Vastaavasti määritellään

$$L^+ = \bigcup_{n=1}^{\infty} L^n,$$

jossa on mukana Λ vain jos se on jo L:ssä. (Vrt. edellä olleet Σ^* ja Σ^+.) Siispä $\emptyset^* = \{\Lambda\}$, mutta $\emptyset^+ = \emptyset$. Itse asiassa $L^+ = L^*L = LL^*$.

Kielten L_1 ja L_2 *osamäärät, vasen* ja *oikea,* ovat

$$L_1 \backslash L_2 = \{w_2 \mid w_1 w_2 \in L_2 \text{ jollekin sanalle } w_1 \in L_1\}$$

(otetaan L_2:n sanoista pois kaikin tavoin prefikseinä olevia L_1:n sanoja) ja

$$L_1 / L_2 = \{w_1 \mid w_1 w_2 \in L_1 \text{ jollekin sanalle } w_2 \in L_2\}$$

(otetaan L_1:n sanoista pois kaikin tavoin suffikseina olevia L_2:n sanoja). Huomaa, että prefiksinä tai suffiksina voi olla myös Λ. Kielen L *peilikuva* on kieli $\hat{L} = \{\hat{w} \mid w \in L\}$.

Kielten määrittelyssä on kaksi peruskoneistoa: *kieliopit,* jotka tuottavat kielen sanat, ja *automaatit,* jotka tunnistavat kielen sanat. Lisäksi on vielä muita tapoja määritellä kieli, esimerkiksi säännöllisten kielten määrittely säännöllisen lausekkeen avulla.

"Some people, when confronted with
a problem, think "I know, I'll use
regular expressions." Now they
have two problems."

(Jamie Zawinski)

Luku 2

SÄÄNNÖLLISET KIELET

2.1 Säännölliset lausekkeet ja kielet

Säännöllinen lauseke on lauseke, joka määrittelee kielen käyttäen joukko-opin yhdistettä, jota merkitään tässä +:lla, katenaatiota ja katenaatiosulkeumaa. Näitä operaatioita yhdistellään tiettyjen sääntöjen mukaan, tarpeen vaatiessa käytetään sulkumerkkejä (ja). Lausekkeen muodostaminen aloitetaan aakkoston symboleista, $\emptyset$:sta ja Λ:sta, joukkosulut { ja } jäävät pois.

Säännöllisten lausekkeiden määrittelemät kielet ovat ns. *säännölliset kielet.* Merkitään aakkoston Σ säännöllisten kielten perhettä $\mathcal{R}_\Sigma$:lla, tai vain $\mathcal{R}$:llä, jos aakkosto on selvä.

Määritelmä. *$\mathcal{R}$ on kieliperhe, joka toteuttaa alla olevat ehdot:*

1. *Kieli $\emptyset$ on $\mathcal{R}$:ssä ja vastaava säännöllinen lauseke on $\emptyset$.*
2. *Kieli $\{\Lambda\}$ on $\mathcal{R}$:ssä ja vastaava säännöllinen lauseke on Λ.*
3. *Jokaiselle symbolille a kieli $\{a\}$ on $\mathcal{R}$:ssä ja vastaava säännöllinen lauseke on a.*
4. *Jos L_1 ja L_2 ovat $\mathcal{R}$:n kieliä ja r_1 sekä r_2 ovat vastaavat säännölliset lausekkeet, niin*
   (a) *kieli $L_1 \cup L_2$ on $\mathcal{R}$:ssä ja vastaava säännöllinen lauseke on $(r_1 + r_2)$.*
   (b) *kieli $L_1 L_2$ on $\mathcal{R}$:ssä ja vastaava säännöllinen lauseke on $(r_1 r_2)$.*
5. *Jos L on $\mathcal{R}$:n kieli ja r on vastaava säännöllinen lauseke, niin L^* on $\mathcal{R}$:ssä ja vastaava säännöllinen lauseke on (r^*).*
6. *Vain ne kielet, jotka saadaan yo. säännöillä* 1.–5., *ovat säännöllisiä.*

Jotteivat lausekkeet kasvaisi kovin pitkiksi, käytetään joitakin lyhennysmerkintöjä, esimerkiksi

$$(rr) =_{\text{merk.}} (r^2) \quad , \quad (r(rr)) =_{\text{merk.}} = (r^3) \quad \text{ja}$$
$$(r(r^*)) =_{\text{merk.}} (r^+).$$

Lisäksi yo. säännöt tuottavat täysin sulutettuja säännöllisiä lausekkeita. Jos sovitaan operaatioiden suoritusjärjestykseksi

$$^* \quad , \quad \text{katenaatio} \quad , \quad +,$$

voidaan sulkuja jättää paljolti pois ja kirjoittaa esimerkiksi $a+b^*c$ lausekkeen $(a+((b^*)c))$ sijasta. Myös on tapana usein samaistaa säännöllinen lauseke sen määrittelemään kieleen, esimerkiksi kirjoitettaessa $r_1 = r_2$ tarkoitetaan, että vastaavat säännölliset kielet ovat samat, itse lausekkeet voivat hyvinkin olla erilaiset. Siispä esimerkiksi

$$(a^*b^*)^* = (a+b)^*.$$

Määritelmästä seuraa suoraan, että kahden säännöllisen kielen yhdiste ja katenaatio ovat myös säännöllisiä, ja edelleen että säännöllisen kielen katenaatiosulkeuma sekä peilikuva ovat säännöllisiä.

2.2 Äärellinen automaatti

Automaatteja käytetään kielen sanojen *tunnistamiseen.* Tällöin automaatti „käsittelee" sanaa ja saatuaan käsittelyn valmiiksi „päättää" onko sana kielessä vai ei. Automaatti on *äärellinen,* jos sillä on äärellinen muisti eli automaatin voidaan ajatella olevan aina jossakin äärellisen monesta (muisti)tilasta. Formaalisti äärellinen deterministinen automaatti määritellään antamalla tilat, kielen symbolit, alkutila, tilasiirtymät sekä hyväksymiskriteeri,

Määritelmä. Äärellinen (deterministinen) automaatti(DFA) *on viisikko* $M = (Q, \Sigma, q_0, \delta, A)$*, missä*

- $Q = \{q_0, q_1, \ldots, q_m\}$ *on äärellinen* tilojen joukko*, jonka alkioita kutsutaan* tiloiksi;
- Σ *on* syötesymbolien aakkosto *(kielen aakkosto);*
- q_0 *on* alkutila ($q_0 \in Q$);
- δ *on* (tila)siirtofunktio, *joka kuvaa jokaisen parin* (q_i, a)*, missä* q_i *on tila ja* a *on syötesymboli, tarkalleen yhdeksi tilaksi* q_j: $\delta(q_i, a) = q_j$;
- A *on* lopputilojen joukko ($A \subseteq Q$).

Automaatti M saa *syötteekseen* sanan

$$w = a_1 \cdots a_n,$$

jonka lukemisen se aloittaa alusta. Aluksi M on alkutilassa q_0 ja lukee w:n ensimmäistä symbolia a_1. Seuraavan tilan q_i määrää siirtofunktio:

$$q_i = \delta(q_0, a_1).$$

Yleisesti, jos M on tilassa q_j ja lukee symbolia a_i, sen seuraava tila on $\delta(q_j, a_i)$ ja se siirtyy lukemaan syötteen seuraavaa symbolia a_{i+1}, jos sellaista on. Jos M:n tila, kun syötteen viimeinen symboli a_n on luettu, on lopputila eli A:ssa, M *hyväksyy* w:n, muuten se *hylkää* w:n. Erityisesti M hyväksyy Λ:n, jos alkutila q_0 on myös lopputila.

Automaatin M *tunnistama* kieli muodostuu sen hyväksymistä sanoista, merkitään $L(M)$.

Sana $w = a_1 \cdots a_n$, syöte tai muu, määrää automaatin M ns. *tilasiirtoketjun* tilasta q_{j_0} tilaan q_{j_n}:

$$q_{j_0}, q_{j_1}, \ldots, q_{j_n},$$

missä aina $q_{j_{i+1}} = \delta(q_{j_i}, a_{i+1})$. Vastaavasti voidaan määritellä siirtofunktion yleistys δ^* sanoille rekursiivisesti seuraavasti:

1. $\delta^*(q_i, \Lambda) = q_i$
2. Sanalle $w = ua$, missä a on symboli, $\delta^*(q_i, w) = \delta\big(\delta^*(q_i, u), a\big)$.

Näin ollen sana w hyväksytään tarkalleen siinä tapauksessa, että $\delta^*(q_0, w)$ on lopputila, ja $L(M)$muodostuu tarkalleen niistä sanoista w, joille $\delta^*(q_0, w)$ on lopputila.

Lause 2. (i) *Jos kielet L_1 ja L_2 ovat tunnistettavissa (omilla) äärellisillä automaateilla M_1 ja M_2, niin myös $L_1 \cup L_2$, $L_1 \cap L_2$ ja $L_1 - L_2$ ovat tunnistettavissa äärellisillä automaateilla.*

(ii) *Jos kieli L on tunnistettavissa äärellisellä automaatilla M, niin myös $\overline{L}$ on tunnistettavissa äärellisellä automaatilla.*

Todistus. (i) Tässä voidaan olettaa, että L_1 ja L_2 ovat samassa aakkostossa. Ellei näin ole, muodostetaan alkuperäisten aakkostojen yhdiste. Edelleen voidaan olettaa, että automaattien M_1 ja M_2 syöteaakkosto on tämä yhteinen aakkosto Σ, kuten on helppo todeta. Muodostetaan M_1:stä ja M_2:sta „tuloautomaatti" seuraavasti. Jos

$$M_1 = (Q, \Sigma, q_0, \delta, A)$$

ja

$$M_2 = (S, \Sigma, s_0, \gamma, B),$$

niin tuloautomaatti on

$$M_1 \times M_2 = \big(Q \times S, \Sigma, (q_0, s_0), \sigma, C\big),$$

missä lopputilojen joukko C valitaan sopivasti. Tilojen joukko $Q \times S$ muodostuu tilapareista (q_i, s_j), missä q_i on Q:ssa ja s_j on S:ssä. Jos $\delta(q_i, a) = q_k$ ja $\gamma(s_j, a) = s_\ell$, niin

$$\sigma\big((q_i, s_j), a\big) = (q_k, s_\ell).$$

Jos halutaan tunnistaa $L_1 \cup L_2$, otetaan C:hen ne parit (q_i, s_j), missä q_i on A:ssa tai/ja s_j on B:ssä, ts. ainakin toinen automaateista päätyy lopputilaan luettuaan syötteen. Jos taas halutaan tunnistaa $L_1 \cap L_2$, otetaan C:hen ne parit (q_i, s_j), missä q_i on A:ssa ja s_j on B:ssä, ts. kumpikin automaateista päätyy lopputilaan luettuaan syötteen. Vielä, jos halutaan tunnistaa $L_1 - L_2$, valitaan C:hen ne parit (q_i, s_j), missä q_i on A:ssa ja s_j ei ole B:ssä, ts. M_1 päätyy lopputilaan luettuaan syötteen, mutta M_2 ei.

(ii) Komplementin $\overline{L}$ tunnistava automaatti saadaan M:stä yksinkertaisesti vaihtamalla sen lopputilojen joukko komplementikseen. □

Äärellinen automaatti voidaan graafisesti esittää ns. *tiladiagrammina.* Tila esitetään ympyränä, jonka sisällä on tilan tunnus, ja lopputila erikoisesti kaksoisympyränä:

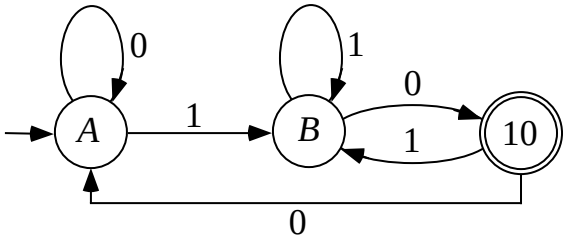
Tilasiirtymä $\delta(q_i, a) = q_j$ esitetään a:lla merkityn nuolen avulla ja alkutila pelkällä tulevalla nuolella:

Tällainen esitys on itse asiassa nuolimerkitty digraafi, ks. kurssi Graafiteoria.

Esimerkki. *Automaattia* $\big(\{A, B, 10\}, \{0, 1\}, A, \delta, \{10\}\big)$*, missä* δ*:n määrittelee tilasiirtotaulu*

δ	0	1
A	A	B
B	10	B
10	A	B

vastaa tiladiagrammi



Automaatin tunnistama kieli on säännöllinen kieli $(0 + 1)^*10$*.*

Yleensäkin *äärellisten automaattien tunnistamat kielet ovat tarkalleen kaikki säännölliset kielet* (ns. Kleenen lause). Tämä todistetaan kahdessa osassa. Ensimmäinen osa voidaan todistaa[1] heti, toinen osa vähän myöhemmin.

Lause 3. *Äärellisen automaatin tunnistama kieli on säännöllinen.*

Todistus. Katsotaan äärellistä automaattia

$$M = (Q, \Sigma, q_0, \delta, A).$$

M:n tilasiirtoketju on *polku,* jos siinä ei esiinny mikään tila monta kertaa. Edelleen tilasiirtoketju on q_i*-kierto,* jos sen ensimmäinen ja viimeinen tila on q_i eikä q_i esiinny ketjussa muualla. Vielä q_i-kierto on q_i*-piiri,* jos ainoa siinä monta kertaa esiintyvä tila on q_i. Huomaa, että polkuja ja piirejä on äärellinen määrä, ketjuja ja kiertoja yleensä sen sijaan on ääretön määrä. Tila q_i on sekä polku („tyhjä polku") että q_i-piiri („tyhjä piiri").

Jokaiseen tilasiirtoketjuun liittyy yksi tai useampia sen aiheuttavia sanoja, kuitenkin äärellinen määrä. Merkitään kaikkiin mahdollisiin q_i-kiertoihin liittyvien sanojen muodostamaa kieltä R_i:llä. Tyhjää piiriä vastaa kieli $\{\Lambda\}$.

Näytetään ensin, että R_i on säännöllinen kieli. Tämä tehdään induktiolla kierroissa esiintyvien eri tilojen lukumäärän suhteen. Merkitään $R_{S,i}$:llä sellaisia q_i-kiertoja vastaavien sanojen muodostamaa kieltä, joissa esiintyy vain Q:n osajoukon S tiloja, mukana tietysti aina q_i. Ilmeisesti silloin $R_i = R_{Q,i}$. Induktio tapahtuu S:n alkioluvun s suhteen ja näyttää jokaisen $R_{S,i}$:n säännölliseksi.

Induktion lähtökohta, $s = 1$*:* Nyt $S = \{q_i\}$, mahdolliset kierrot ovat q_i sekä q_i, q_i ja kieli $R_{S,i}$ on äärellinen ja siis säännöllinen ($R_{S,i}$:ssä on Λ ja mahdollisesti joitain yksittäisiä symboleja).

Induktio-oletus: Väitetty tulos pitää paikkansa, kun $s < h$, missä $h \geq 2$.

Induktioväite: Väitetty tulos pitää paikkansa, kun $s = h$.

Induktioväitteen todistus: Jokainen q_i-kierto, jossa esiintyy vain S:n tiloja, voidaan kirjoittaa—mahdollisesti monellakin eri tavalla—muotoon

$$q_i, q_{i_1}, K_1, \ldots, q_{i_n}, K_n, q_i,$$

[1]Todistus voidaan muuntaa matriisimuotoiseksi algoritmiksi, ns. *Kleenen algoritmiksi,* joka on sukua graafiteorian Warshallin ja Floydin algoritmeille, ks. kurssi Graafiteoria.

missä $q_i, q_{i_1}, \ldots, q_{i_n}, q_i$ on q_i-piiri ja q_{i_j}, K_j muodostuu q_{i_j}-kierroista, joissa esiintyy vain $S - \{q_i\}$:n tiloja. Merkitään piiriä $q_i, q_{i_1}, \ldots, q_{i_n}, q_i$ itseään $\mathcal{C}$:llä. Sitä vastaa tietty äärellinen määrä sanoja

$$a_{j0}a_{j1}\cdots a_{jn} \quad (j = 1, \ldots, \ell),$$

jotka aiheuttavat tilasiirtoketjuna juuri piirin $\mathcal{C}$. Kaikkia mahdollisia q_{i_j}, K_j:ssä olevia q_{i_j}-kiertoja vastaavien sanojen muodostama kieli $R_{S-\{q_i\},i_j}$ on Induktio-oletuksen mukaisesti säännöllinen, merkitään vastaavaa säännöllistä lauseketta lyhyesti r_j:llä. Samoin on säännöllinen kaikkia mahdollisia esitettyä muotoa $q_i, q_{i_1}, K_1, \ldots, q_{i_n}, K_n, q_i$ olevia q_i-kiertoja vastaavien sanojen muodostama kieli

$$\sum_{j=1}^{\ell} a_{j0}r_1^* a_{j1}r_2^* \cdots r_n^* a_{jn} =_{\text{merk.}} r_{\mathcal{C}}.$$

Jos sellaiset q_i-piirit, joissa esiintyy vain S:n tiloja, ovat $\mathcal{C}_1, \ldots, \mathcal{C}_m$, niin kysytty säännöllinen kieli $R_{S,i}$ on $r_{\mathcal{C}_1} + \cdots + r_{\mathcal{C}_m}$.

Itse lauseen todistus on nyt hyvin samantapainen kuin yo. induktiotodistus. Jokainen alkutilasta lopputilaan johtava ketju nimittäin joko muodostuu q_0-kierroista (jos alkutila on samalla lopputila) tai on muotoa

$$q_{i_0}, K_0, q_{i_1}, K_1, \ldots, q_{i_n}, K_n,$$

missä $i_0 = 0$, q_{i_n} on lopputila, $q_{i_0}, q_{i_1}, \ldots, q_{i_n}$ on polku ja q_{i_j}, K_j muodostuu q_{i_j}-kierroista. Kuten edellä, vastaavien sanojen muodostama kieli todetaan säännölliseksi. □

Huomautus. *Koska tiladiagrammiin tulee aika paljon nuolia, sallitaan usein* osittainen *tiladiagrammi, jossa ei ole kaikkia tilasiirtymiä vastaavia nuolia. Automaatin toiminta sen kohdatessa tilasiirtymän, jota ei ole, tulkitaan yksinkertaisesti niin, ettei syötesanaa tämän tilanteen jälkeen enää hyväksytä. Vastaavaa siirtofunktiota sanotaan* osittaiseksi funktioksi, *sillä se ei ole aina määritelty. On varsin helppo todeta, että tämä ei lisää millään tavoin äärellisten automaattien tunnistuskykyä, sillä jokainen „vajaa" äärellinen automaatti voidaan aina korvata ekvivalentilla „täydellisellä". Lisätään vain yksi ylimääräinen „hylkytila", johon aina siirrytään, kun tilansiirtoa ei muuten ole määritelty ja josta ei pääse kuin itseensä.*

Automaatissa voi myös olla „turhia tiloja", ts. sellaisia tiloja, joihin ei voi mitenkään päästä alkutilasta. Nämä voidaan ilman muuta poistaa.

2.3 Sanojen erottaminen ja pumppaus

Kieli L *erottaa* sanat w ja v, jos on sellainen sana u, että tarkalleen toinen sanoista wu ja vu on L:ssä. Jos L ei erota sanoja w ja v, niin sanat wu ja vu ovat u:sta riippuen joko molemmat L:ssä tai molemmat $\overline{L}$:ssä.

Äärellisen automaatin tunnistaman kielen erottelukyvyllä on yhteyksiä automaatin rakenteeseen:

Lause 4. *Jos äärellinen automaatti $M = (Q, \Sigma, q_0, \delta, A)$ tunnistaa kielen L ja sanoille w ja v*

$$\delta^*(q_0, w) = \delta^*(q_0, v),$$

niin L ei erota sanoja w ja v.

Todistus. Kuten helposti voi todeta, yleisesti

$$\delta^*(q_i, xy) = \delta^*\big(\delta^*(q_i, x), y\big).$$

Siispä

$$\delta^*(q_0, wu) = \delta^*\big(\delta^*(q_0, w), u\big) = \delta^*\big(\delta^*(q_0, v), u\big) = \delta^*(q_0, vu).$$

Riippuen siitä onko kyseessä lopputila vai ei sanat wu ja vu ovat L:ssä tai $\overline{L}$:ssä. □

Seuraus. *Jos kieli L erottaa mitkä tahansa kaksi n sanasta $w_1, \ldots, w_n$, L:ää ei voi tunnistaa äärellisellä automaatilla, jossa on vähemmän kuin n tilaa.*

Todistus. Jos äärellisellä automaatilla $M = (Q, \Sigma,\ q_0, \delta,\ A)$ on tiloja vähemmän kuin n kpl, on tiloista

$$\delta^*(q_0, w_1)\ , \ldots,\ \delta^*(q_0, w_n)$$

ainakin kaksi samaa. □

Aakkoston ns. *palindromien* muodostama kieli L_{pal} on esimerkki kielestä, jonka tunnistamiseen tarvitaan *ääretön määrä* tiloja (olettaen, että aakkostossa on ainakin kaksi symbolia). Sana w on palindromi, jos $\hat{w} = w$. L_{pal} nimittäin pystyy erottamaan kaikki sanat, kahdesta sanastahan voidaan aina yksi täydentää palindromiksi ja toinen ei-palindromiksi. Sama koskee lukuisia muitakin kieliä, mm. ns. *neliöiden* muodostamaa kieltä L_{sqr}, jossa ovat mukana tarkalleen kaikki muotoa w^2 olevat sanat.

Erottelukyky liittyy läheisesti myös tilaluvultaan pienimmän äärellisen automaatin konstruktioon eli ns. automaatin *minimointiin,* josta lisää myöhemmin.

Katsotaan vielä yo. todistuksen tapaista tilannetta, missä äärellisellä automaatilla on tarkalleen n tilaa ja hyväksyttävä syöte on pituudeltaan vähintään n:

$$x = a_1 a_2 \cdots a_n y,$$

missä $a_1, \ldots, a_n$ ovat syötesymboleja ja y on sana. Tiloista

$$q_0 = \delta^*(q_0, \Lambda)\ ,\ \delta^*(q_0, a_1)\ ,\ \delta^*(q_0, a_1 a_2)\ , \ldots,\ \delta^*(q_0, a_1 a_2 \cdots a_n)$$

ainakin kaksi on samaa, sanotaan

$$\delta^*(q_0, a_1 a_2 \cdots a_i) = \delta^*(q_0, a_1 a_2 \cdots a_{i+p}).$$

Merkitään lyhyyden vuoksi

$$u = a_1 \cdots a_i \quad , \quad v = a_{i+1} \cdots a_{i+p} \quad \text{ja} \quad w = a_{i+p+1} \cdots a_n y.$$

Ilmeisesti nyt sanat $uv^m w$ $(m = 0, 1, \ldots)$ tulevat myös hyväksytyiksi! Tämä tulos tunnetaan nimellä

Pumppauslemma (*uvw*-lemma). *Jos kieli L voidaan tunnistaa n-tilaisella äärellisellä automaatilla, $x \in L$ ja $|x| \geq n$, niin x voidaan kirjoittaa muotoon $x = uvw$, missä $|uv| \leq n$, $v \neq \Lambda$ ja sanat $uv^m w$ ovat* kaikki *L:ssä.*

2.4 Epädeterminististinen äärellinen automaatti

Epädeterministisyydellä tarkoitetaan tietyille valinnoille jätettyä vapautta, ts. mikä tahansa mahdollisista useista vaihtoehdoista voidaan valita. Sallitut vaihtoehdot on kuitenkin määriteltävä ja niitä on yleensä äärellinen määrä. Toiset valinnat ovat kuitenkin parempia kuin toiset, ts. voi olla, että tavoitteeseen päästään vain tietyin sopivin valinnoin.

Äärellisen automaatin tapauksessa *epädeterministisyys* tarkoittaa tilasiirroissa olevaa valintaa, kussakin tilanteessa saattaa olla monia vaihtoehtoisia tiloja, joihin voidaan siirtyä, ja jo alkutiloja voi olla monta. Tämä ilmaistaan siten, että siirtofunktion arvot ovat tilojen joukkoja, joissa ovat kulloinkin mukana tarkalleen kaikki vaihtoehtoiset seuraavat tilat. Ko. joukko voi olla myös tyhjä, jolloin tilasiirtoa ei ole, vrt. eo. huomautus osittaisista tiladiagrammeista.

Edellä äärellinen automaatti oli aina deterministinen. Jatkossa pitää mainita tarkasti automaatin tyyppi.

Formaalisti *epädeterministinen äärellinen automaatti* (*NFA*) on viisikko $M = (Q, \Sigma, S, \delta, A)$, missä

- Q, Σ ja A ovat kuten deterministiselle äärelliselle automaatille;
- S on *alkutilojen joukko;*
- δ on (*tila*)*siirtofunktio*, joka kuvaa jokaisen parin (q_i, a), missä q_i on tila ja a on syötesymboli, tarkalleen yhdeksi Q:n tilojen joukoksi T: $\delta(q_i, a) = T$.

Huomaa, että sekä S että T edellä voivat olla tyhjiä joukkoja. Kaikkien Q:n osajoukkojen joukkoa merkitään tavallisesti 2^Q:lla.

Siirtofunktio δ voidaan välittömästi yleistää siten, että sen ensimmäinen argumentti on tilajoukko:

$$\hat{\delta}(\emptyset, a) = \emptyset \quad \text{ja} \quad \hat{\delta}(U, a) = \bigcup_{q_i \in U} \delta(q_i, a).$$

Edelleen voidaan määritellä $\hat{\delta}^*$ samaan tapaan kuin edellä määriteltiin δ^*:

$$\hat{\delta}^*(U, \Lambda) = U \quad \text{ja} \quad \hat{\delta}^*(U, ua) = \hat{\delta}\big(\hat{\delta}^*(U, u), a\big).$$

M *hyväksyy* sanan w, jos tilajoukossa $\hat{\delta}^*(S, w)$ on ainakin yksi lopputila. Λ hyväksytään, jos ainakin yksi lopputila on S:ssä. Kaikki M:n hyväksymät sanat muodostavat M:n *tunnistaman* kielen $L(M)$.

Epädeterministinen äärellinen automaatti on deterministisen äärellisen automaatin yleistys, kun jälkimmäisessä aina tila q_i samaistetaan joukkoon $\{q_i\}$. Kuitenkaan se ei ole kielentunnistuskyvyiltään yhtään sen voimakkaampi:

Lause 5. *Jos kieli voidaan tunnistaa epädetermisellä äärellisellä automaatilla, se voidaan tunnistaa myös deterministisellä äärellisellä automaatilla, ja on siis säännöllinen.*

Todistus. Otetaan kieli L, jonka tunnistaa epädeterministinen äärellinen automaatti $M = (Q, \Sigma, S, \delta, A)$. Muodostetaan deterministinen äärellinen automaatti $M_1 = (Q_1, \Sigma, q_0, \delta_1, A_1)$, missä

$$Q_1 = 2^Q \quad , \quad q_0 = S \quad , \quad \delta_1 = \hat{\delta}$$

ja A_1 muodostuu tarkalleen niistä tilojen joukoista, joissa on ainakin yksi A:n tila. M_1:n tilat ovat siis M:n tilojen joukot.

Ilmeisesti nyt $\delta_1^*(q_0, w) = \hat{\delta}^*(S, w)$, joten M ja M_1 hyväksyvät tarkalleen samat sanat, ja näin ollen myös M_1 tunnistaa kielen L. □

Hieman toisenlainen epädeterministisyys saadaan, kun sallitaan ns. Λ-*siirrot.* Tällöin epädeterministisen äärellisen automaatin siirtofunktio δ on määritelty myös kaikille pareille (q_i, Λ), missä q_i on tila. Tuloksena on *epädeterminististinen äärellinen automaatti, jossa on* Λ*-siirtoja* (Λ*-NFA*). Siirto $\delta(q_i, \Lambda) = T$ tulkitaan siten, että automaatti voi siirtyä tilasta q_i mihin tahansa T:ssä olevaan tilaan lukematta uutta syötesymbolia. Jos $\delta(q_i, \Lambda) = \emptyset$ tai $\delta(q_i, \Lambda) = \{q_i\}$, ei tilasta q_i ole Λ-siirtoja muihin tiloihin.

Muiden kuin Λ-siirtojen osalta δ yleistetään tilojen joukoille kuten edellä. Λ-siirtojen osalta se voidaan yleistää tilajoukoille samaan tapaan:

$$\hat{\delta}(\emptyset, \Lambda) = \emptyset \quad \text{ja} \quad \hat{\delta}(U, \Lambda) = \bigcup_{q_i \in U} \delta(q_i, \Lambda).$$

Edelleen se voidaan yleistää Λ-siirtojen osalta „tähtifunktioksi”: $\hat{\delta}^*(U, \Lambda) = V$, jos

- U:n tilat ovat V:ssä;
- $\hat{\delta}(V, \Lambda)$:n tilat ovat V:ssä;
- jokainen V:n tila on U:n tila tai siihen päästään soveltamalla toistuvasti Λ-siirtoja jostain U:n tilasta lähtien.

Ja lopulta voidaan nyt yleistää δ muidenkin kuin vain Λ-siirtojen osalta:

$$\hat{\delta}^*(U, ua) = \hat{\delta}^*\big(\hat{\delta}\big(\hat{\delta}^*(U, u), a\big), \Lambda\big).$$

Huomaa erityisesti, että syötesymbolille a saadaan

$$\hat{\delta}^*(U, a) = \hat{\delta}^*\big(\hat{\delta}\big(\hat{\delta}^*(U, \Lambda), a\big), \Lambda\big),$$

ts. ensin tehdään Λ-siirtoja, sitten a:n aiheuttama „varsinainen” tilasiirto ja lopuksi taas Λ-siirtoja.

Λ-NFA:n hyväksymät sanat ja tunnistama kieli määritellään samoin kuin edellä. Mutta nytkään ei saada lisää tunnistusvoimaa:

Lause 6. *Jos kieli voidaan tunnistaa* Λ*-NFA:lla, se voidaan tunnistaa myös epädeterministisellä äärellisellä automaatilla, ja edelleen deterministisellä äärellisellä automaatilla ja on siis taas säännöllinen.*

Todistus. Otetaan kieli L, jonka tunnistaa Λ-NFA $M = (Q, \Sigma, S, \delta, A)$. Muodostetaan epädeterministinen äärellinen automaatti $M_1 = (Q, \Sigma, S_1, \delta_1, A)$ ilman Λ-siirtoja, missä

$$S_1 = \hat{\delta}^*(S, \Lambda) \quad \text{ja} \quad \delta_1(q_i, a) = \hat{\delta}^*\big(\{q_i\}, a\big).$$

Ilmeisesti nyt $\hat{\delta}_1^*(S_1, w) = \hat{\delta}^*(S, w)$, joten M ja M_1 hyväksyvät tarkalleen samat sanat, ja näin ollen myös M_1 tunnistaa kielen L. Huomaa erityisesti, että jos M hyväksyy Λ:n, niin jostain S:n tilasta päästään Λ-siirroilla lopputilaan ja ko. lopputila on silloin S_1:ssä. □

Epädeterministinenkin äärellinen automaatti—Λ-siirroilla tai ilman—voidaan esittää ilmeisellä tavalla tiladiagrammin avulla. Jos tilojen välillä on useita samansuuntaisia nuolia, on ne usein tapana yhdistää yhdeksi nuoleksi kooten ko. nuolten merkit listaksi.

2.5 Kleenen lause

Edellä Lauseessa 3 todettiin, että deterministisen äärellisen automaatin tunnistama kieli on aina säännöllinen, ja myöhemmin vielä, että sama pätee myös epädeterministisessä tapauksessa. Tämä tulos pätee myös kääntäen.

Kleenen lause. *Säännölliset kielet ovat tarkalleen kaikki äärellisten automaattien tunnistamat kielet.*

Todistus. Pitää vielä näyttää, että jokainen säännöllinen kieli voidaan tunnistaa äärellisellä automaatilla. Ajatellen säännöllisen lausekkeen rakennetta, pitää ensin näyttää, että „peruskielet" $\emptyset$, $\{\Lambda\}$ ja $\{a\}$, missä a on symboli, voidaan tunnistaa äärellisellä automaatilla. Tämä on helppoa. Toiseksi pitää näyttää, että jos kielet L_1 ja L_2 voidaan tunnistaa äärellisellä automaatilla, niin samoin voidaan kielet $L_1 \cup L_2$ ja L_1L_2. Yhdisteelle tämä tehtiin jo Lauseessa 2. Ja kolmanneksi pitää näyttää, että jos kieli L on tunnistettavissa äärellisellä automaatilla, niin samoin on L^+, ja siis myös $L^* = L^+ \cup \{\Lambda\}$.

Otetaan sitten tilanne, missä kielet L_1 ja L_2 voidaan tunnistaa epädeterministisillä äärellisillä automaateilla

$$M_1 = (Q_1, \Sigma_1, S_1, \delta_1, A_1) \quad \text{ja} \quad M_2 = (Q_2, \Sigma_2, S_2, \delta_2, A_2),$$

vastaavasti. Voidaan olettaa, että $\Sigma_1 = \Sigma_2 = \Sigma$ (lisätään vain tyhjiä siirtoja). Edelleen voidaan ilmeisesti olettaa, että tilojen joukot Q_1 ja Q_2 ovat erilliset. Uusi automaatti, joka tunnistaa L_1L_2:n, on

$$M = (Q, \Sigma, S_1, \delta, A_2),$$

missä

$$Q = Q_1 \cup Q_2$$

ja δ saadaan seuraavasti:

$$\delta(q,a) = \begin{cases} \delta_1(q,a), \text{ jos } q \in Q_1 \\ \delta_2(q,a), \text{ jos } q \in Q_2 \end{cases} \quad \text{ja} \quad \delta(q,\Lambda) = \begin{cases} \delta_1(q,\Lambda) & , \text{jos } q \in Q_1 - A_1 \\ \delta_1(q,\Lambda) \cup S_2, & \text{jos } q \in A_1 \\ \delta_2(q,\Lambda) & , \text{jos } q \in Q_2. \end{cases}$$

Lopputilaan voidaan nyt päästä alkutilasta vain siirtymällä Λ-siirrolla M_1:n jostain lopputilasta johonkin M_2:n alkutilaan ja tämä siirto tapahtuu vaiheessa, jossa M_1 on hyväksynyt luetun prefiksin. Lopputilaan pääsemiseksi pitää jäljellä olevan suffiksin olla L_2:ssa.

Vielä katsotaan tilannetta, jossa kieli L voidaan tunnistaa epädeterministisellä äärellisellä automaatilla

$$M = (Q, \Sigma, S, \delta, A).$$

Silloin L^+ voidaan tunnistaa automaatilla

$$M' = (Q, \Sigma, S, \delta', A),$$

missä

$$\delta'(q,a) = \delta(q,a) \quad \text{ja} \quad \delta'(q,\Lambda) = \begin{cases} \delta(q,\Lambda) & , \text{jos } q \notin A \\ \delta(q,\Lambda) \cup S, & \text{jos } q \in A. \end{cases}$$

Lopputilasta voidaan nyt siirtyä aina Λ-siirrolla alkutilaan, mikä mahdollistaa toistuvan katenaation. Jos syöte jaetaan osasanoihin sen mukaan missä nämä Λ-siirrot sattuvat, ovat osasanat kielessä L. □

Kleenen lauseen ja muiden eo. lauseiden avulla saadaan säännöllisille kielille karakterisaatiot sekä säännöllisten lausekkeiden kautta että erilaisten äärellisten automaattien (DFA, NFA ja Λ-NFA) tunnistamina kielinä. Nämä eri karakterisaatiot ovat luonteeltaan erilaisia ja eri tilanteissa käyttökelpoisia. Missä säännöllinen lauseke sopii hyvin, voi automaatti olla vaikea käsiteltävä, ja toisaalta automaateilla voidaan helposti tehdä sellaista, mikä on säännöllistä lauseketta käyttäen vaikeaa. Tämä näkyy mainittujen lauseiden todistuksissakin, ajattele vain miten vaikeaa olisi näyttää, että kahden säännöllisen kielen leikkaus on säännöllinen kieli, suoraan käyttäen säännöllisiä lausekkeita.

2.6 Automaatin minimointi

Monet äärelliset automaatit tunnistavat saman säännöllisen kielen L. Tilaluvultaan pienin L:n tunnistava deterministinen äärellinen automaatti on *minimaalinen äärellinen automaatti.* Minimaalinen äärellinen automaatti voidaan löytää tutkimalla kielen L rakennetta. Tätä varten pitää tietysti L:n olla säännöllinen ja jollain tavoin annettu, mutta katsotaan asiaa ensin aivan yleisesti. Aakkosto on Σ.

Pykälässä 3 määriteltiin kielen L erottamat sanat. Merkitään nyt $w \not\equiv_L v$, jos kieli L erottaa sanat w ja v, ja vastaavasti $w \equiv_L v$, jos L ei erota sanoja w ja v. Viimemainitussa tapauksessa sanotaan, että sanat w ja v ovat *L-ekvivalentit.* Ilmeisesti voidaan sopia, että aina $w \equiv_L w$. Edelleen, jos $w \equiv_L v$, niin myös $v \equiv_L w$.

Apulause. *Jos $w \equiv_L v$ ja $v \equiv_L u$, niin silloin myös $w \equiv_L u$. (Kuten on tapana sanoa, $\equiv_L$ on* transitiivinen.*)*

Todistus. Jos $w \equiv_L v$ ja $v \equiv_L u$ ja z on jokin sana, niin on kaksi mahdollisuutta. Jos vz on L:ssä, niin myös wz ja uz ovat L:ssä. Jos taas vz ei ole L:ssä, niin myöskään wz ja uz eivät ole L:ssä. Siispä $w \equiv_L u$. □

Tästä seuraa, että Σ^*:n sanat jakautuvat ns. *L-ekvivalenssiluokkiin:* Sanat w ja v ovat samassa luokassa, jos ne ovat L-ekvivalentit. Luokkaa, jossa on sana w, merkitään $[w]$:llä. „Edustajaksi" voidaan valita mikä tahansa toinenkin luokan sana v: Jos $w \equiv_L v$, niin $[w] = [v]$. Huomaa, että jos $w \not\equiv_L u$, niin $[w]$ ja $[u]$ eivät leikkaa. Jos niissä nimittäin olisi yhteinen sana v, niin $w \equiv_L v$ ja $v \equiv_L u$ ja Apulauseen nojalla $w \equiv_L u$.

L-ekvivalenssiluokkien lukumäärää kutsutaan kielen L *indeksiksi.* Se voi olla ääretönkin, mutta Lauseesta 4 seuraa suoraan, että

Lause 7. *Jos kielen tunnistaa deterministinen äärellinen automaatti, jossa on n tilaa, niin kielen indeksi on enintään n.*

Toisaalta

Lause 8. *Jos kielen L indeksi on n, niin se voidaan tunnistaa n-tilaisella deterministisellä äärellisellä automaatilla.*

Todistus. Otetaan kieli L, jonka indeksi on n, ja sen eri ekvivalenssiluokat

$$[x_0], [x_1], \ldots, [x_{n-1}],$$

missä $x_0 = \Lambda$.

Deterministinen äärellinen automaatti, jonka on määrä tunnistaa L, on $M = \big(Q, \Sigma, [\Lambda], \delta, A\big)$, missä

$$Q = \big\{[x_0], [x_1], \ldots, [x_{n-1}]\big\},$$

A:han otetaan ne ekvivalenssiluokat $[x_i]$, joissa on L:n sanoja, ja

$$\delta\big([x_i], a\big) = [x_i a].$$

Vm. δ:n määrittely on OK, sillä jos $x \equiv_L y$, niin ilmeisesti myös $xa \equiv_L ya$. Vastaava δ^* saadaan silloin suoraan:

$$\delta^*\big([x_i], y\big) = [x_i y].$$

Edelleen $L(M)$ muodostuu niistä sanoista w, joille

$$\delta^*\big([\Lambda], w\big) = [\Lambda w] = [w]$$

on M:n lopputila eli sisältää L:n sanoja.

Ilmeisesti $L \subseteq L(M)$, sillä jos $w \in L$, niin $[w]$ on M:n lopputila. Toisaalta, jos $[w]$:ssä on L:n sana v, niin w itsekin on L:ssä, muutenhan $w\Lambda \notin L$ ja $v\Lambda \in L$ ja L erottaisi w:n ja v:n. Ts. jos $w \in L$, niin $[w] \subseteq L$. Siispä $L(M) = L$. □

Seuraus. *Minimaalinen äärellinen automaatti kielelle L sisältää L:n indeksin osoittaman määrän tiloja.*

Seuraus (Myhill–Nerode-lause). *Kieli on säännöllinen tarkalleen silloin, kun sen indeksi on äärellinen.*

Jos säännöllinen kieli L annetaan deterministisen äärellisen automaatin $M = (Q, \Sigma, q_0, \delta, A)$ tunnistamana kielenä, niin minimointi pitää luonnollisesti tehdä M:stä lähtien. Aluksi poistetaan M:stä tilat, joihin ei voi päästä alkutilasta tilasiirroilla. Näin voidaan olettaa, että kaikki M:n tilat ovat muotoa $\delta^*(q_0, w)$ jollekin sanalle w.

Minimointia varten jaetaan M:n tilat *M-ekvivalenssiluokkiin* seuraavasti. Tilat q_i ja q_j eivät ole M-ekvivalentit, jos on sellainen sana u, että tiloista $\delta^*(q_i, u)$ ja $\delta^*(q_j, u)$ toinen on lopputila ja toinen ei, merkitään $q_i \not\equiv_M q_j$. Muussa tapauksessa q_i ja q_j ovat *M-ekvivalentit,* merkitään $q_i \equiv_M q_j$. Ilmeisesti aina $q_i \equiv_M q_i$. Edelleen, jos $q_i \equiv_M q_j$, niin myös $q_j \equiv_M q_i$. Ja jos $q_i \equiv_M q_j$ ja $q_j \equiv_M q_k$, niin myös $q_i \equiv_M q_k$. Tilojen ekvivalenssiluokat muodostuvat nyt keskenään M-ekvivalenteista tiloista ja ne ovat erilliset. (Vrt. L-ekvivalenssiluokat ja ekvivalenssi $\equiv_L$.) Merkitään tilan q_i edustamaa M-ekvivalenssiluokkaa $\langle q_i \rangle$:llä. Huomaa, että on samantekevää mikä luokan edustaja valitaan. Merkitään kaikkien M-ekvivalenssiluokkien joukkoa $\mathcal{Q}$:lla.

M-ekvivalenssi ja L-ekvivalenssi ovat tekemisissä toistensa kanssa, sillä $\big\langle \delta^*(q_0, w) \big\rangle = \big\langle \delta^*(q_0, v) \big\rangle$ tarkalleen silloin, kun $[w] = [v]$. Koska kaikki tilat voidaan saavuttaa tilasiirroilla alkutilasta lähtien, on näin ollen M-ekvivalenssiluokkia yhtä paljon kuin L-ekvivalenssiluokkia, eli L:n indeksin osoittama määrä. Edelleen M-ekvivalenssiluokat ja L-ekvivalenssiluokat voidaan asettaa yksikäsitteisesti vastaamaan toisiaan:

$$\big\langle \delta^*(q_0, w) \big\rangle \rightleftharpoons [w],$$

ja erityisesti $\langle q_0 \rangle \rightleftharpoons [\Lambda]$.

Minimaalinen automaatti, vastaten Lauseen 8 todistuksen konstruktiota, on nyt

$$M_{\min} = \big(\mathcal{Q}, \Sigma, \langle q_0 \rangle, \delta_{\min}, \mathcal{A}\big),$$

missä $\mathcal{A}$ muodostuu niistä M-ekvivalenssiluokista, joissa on lopputiloja, ja $\delta_{\min}$:n määrittelee

$$\delta_{\min}\big(\langle q_i \rangle, a\big) = \big\langle \delta(q_i, a) \big\rangle.$$

Huomaa, että jos M-ekvivalenssiluokassa on yksikin lopputila, niin kaikki sen tilat ovat lopputiloja. Huomaa myös, että jos $q_i \equiv_M q_j$, niin $\delta(q_i, a) \equiv_M \delta(q_j, a)$, ts. $\delta_{\min}$ on hyvin määritelty.

Samantapainen konstruktio voidaan tehdä myös lähtien epädeterministisestä äärellisestä automaatista, Λ-siirroilla tai ilman.

2.7 Ratkeavuuskysymyksiä

Säännöllisille kielille melkeinpä kaikki niitä koskevat karakterisaatioprobleemat ovat algoritmisesti ratkeavia. Katsotaan tavallisimmat probleemat (säännölliselle kielelle L tai säännöllisille kielille L_1 ja L_2):

- *Tyhjyysprobleema:* Onko L tyhjä kieli $\emptyset$?

  Äärellisestä automaatista on varsin helppo todeta, onko sillä jokin tilasiirtoketju alkutilasta lopputilaan vai ei.

- *Sisältymisprobleema:* Sisältyykö kieli L_1 kieleen L_2?

  Ilmeisesti $L_1 \subseteq L_2$ tarkalleen silloin, kun $L_1 - L_2 = \emptyset$.

- *Ekvivalenssiprobleema:* Onko $L_1 = L_2$?

  Ilmeisesti $L_1 = L_2$ tarkalleen silloin, kun $L_1 \subseteq L_2$ ja $L_2 \subseteq L_1$.

- *Äärellisyysprobleema:* Onko L äärellinen kieli?

  Äärellisestä automaatista on varsin helppo todeta onko sillä miten tahansa pitkiä tilasiirtoketjuja alkutilasta lopputilaan vai ei. Vrt. Lauseen 3 todistus.

- *Jäsenyysprobleema:* Onko sana w kielessä L?

  Äärellisellä automaatilla on helppo tarkistaa, hyväksyykö se syötesanan.

2.8 Jonokoneet ja transduktorit (katsaus)

Jonokone on deterministinen äärellinen automaatti varustettuna tulostuksella. Formaalisti *jonokone* (*SM*) on kuusikko

$$S = (Q, \Sigma, \Delta, q_0, \delta, \tau),$$

missä Q, Σ, q_0 ja δ ovat kuten deterministiselle äärelliselle automaatille, Δ on *tulosteaakkosto* ja τ on *tulostusfunktio,* joka kuvaa jokaisen parin (q_i, a) joksikin Δ:n symboliksi. Lopputiloja ei tässä tarvita.

δ laajennetaan „tähtifunktioksi" δ^* tavalliseen tapaan. τ taas laajennetaan seuraavasti:

1. $\tau^*(q_i, \Lambda) = \Lambda$
2. Sanalle $w = ua$, missä a on symboli,

$$\tau^*(q_i, ua) = \tau^*(q_i, u)\tau\big(\delta^*(q_i, u), a\big).$$

Syötesanaa w vastaava tulostesana on $\tau^*(q_0, w)$. Jonokone S kuvaa kielen L kieleksi

$$S(L) = \big\{\tau^*(q_0, w) \;\big|\; w \in L\big\}.$$

Automaattikonstruktiota käyttäen on melko yksinkertaista todeta, että jonokone kuvaa aina säännöllisen kielen säännölliseksi kieleksi.

Yleistetty jonokone (*GSM*)[2] on muuten kuin jonokone, mutta tulostusfunktion arvot ovat Δ^*:n sanoja. Jälleen voidaan todeta, että yleistetty jonokone kuvaa aina säännöllisen kielen säännölliseksi.

[2]Jotkut määrittelevät GSM:ään lopputilatkin, jolloin se ei kuvaa kaikkia sanoja.

Jos yleistetyssä jonokoneessa on vain yksi tila, sanotaan sen antamaa sanojen (ja kielten) kuvausta *morfismiksi.* Koska on vain yksi tila, ei sitä tarvitse merkitä näkyviin lainkaan:

$$\tau^*(\Lambda) = \Lambda \quad \text{ja} \quad \tau^*(ua) = \tau^*(u)\tau(a).$$

Ilmeisesti tällöin kaikille Σ^*:n sanoille u ja v

$$\tau^*(uv) = \tau^*(u)\tau^*(v).$$

Morfismista on ainakin helppo nähdä, että se kuvaa säännöllisen kielen säännölliseksi: Kuvataan vain vastaava säännöllinen lauseke morfismilla.

Jonokoneesta ja yleistetystä jonokoneesta on olemassa epädeterministiset versiot. Yleisempi käsite on kuitenkin ns. transduktori. Formaalisti *transduktori* on viisikko

$$T = (Q, \Sigma, \Delta, S, \delta),$$

missä Q, Σ ja Δ ovat kuten jonokoneelle, S on alkutilojen joukko ja δ on *siirto-tulostus-funktio,* joka kuvaa kunkin parin (q_i, a) muotoa (q_j, u) olevien parien äärelliseksi joukoksi. Tulkinta on se, että saadessaan tilassa q_i syötteen a voi T siirtyä tulostaen u:n mihin tahansa sellaiseen tilaan q_j, että pari (q_j, u) on $\delta(q_i, a)$:ssa.

„Tähtifunktion" $\hat{\delta}^*$ määritteleminen on nyt vähän työläämpää (sivuutetaan) ja kielen L *transduktio* on

$$\bigcup_{w \in L} \{u \mid (q_i, u) \in \hat{\delta}^*(S, w) \text{ jollekin } q_i\text{:lle}\}.$$

Joka tapauksessa transduktori kuvaa aina säännöllisen kielen säännölliseksi kieleksi, ts. säännöllisyys säilyy transduktioissakin.

Yksitilaisen transduktorin antamaa kuvausta eli transduktiota kutsutaan usein *äärelliseksi sijoitukseksi.* Kuten morfismille, on helppo todeta, että äärellinen sijoitus säilyttää säännöllisyyden: kuvataan säännöllinen lauseke ko. äärellisellä sijoituksella.

Luku 3

KIELIOPIT

3.1 Uudelleenkirjoitusjärjestelmät

Uudelleenkirjoitusjärjestelmä ja erikoisesti kielioppi antaa säännöt, joita loputtomasti toistaen saadaan tuotetuksi kielen sanat, lähtien tietystä lähtösanasta. Kieleen voidaan valita vain tietyt tuotetut sanat. Tämä toiminto on tavallaan duaalinen verrattuna automaattiin, joka tunnistaa kielen.

Määritelmä. Uudelleenkirjoitusjärjestelmä[1] *(*RWS*) on pari* $R = (\Sigma, P)$*, missä*

- Σ *on aakkosto;*
- $P = \{(p_1, q_1), \ldots, (p_n, q_n)\}$ *on äärellinen* Σ^**:n sanojen järjestettyjen parien, ns.* produktioiden *joukko. Produktio* (p_i, q_i) *kirjoitetaan yleensä muotoon* $p_i \to q_i$*.*

R *tuottaa suoraan* sanasta w sanan v, jos voidaan kirjoittaa $w = rp_is$ ja $v = rq_is$ jollekin produktiolle (p_i, q_i), merkitään

$$w \Rightarrow_R v.$$

$\Rightarrow_R$:stä muodostetaan vastaava „tähtirelaatio"[2] $\Rightarrow_R^*$ seuraavasti (vrt. siirtofunktion laajentaminen „tähtifunktioksi"):

1. Kaikille sanoille w on $w \Rightarrow_R^* w$.
2. Jos $w \Rightarrow_R v$, niin myös $w \Rightarrow_R^* v$.
3. Jos $w \Rightarrow_R^* v$ ja $v \Rightarrow_R^* u$, niin myös $w \Rightarrow_R^* u$.
4. $w \Rightarrow_R^* v$ vain jos tämä seuraa edellisistä kohdista.

Jos nyt $w \Rightarrow_R^* v$, niin sanotaan, että w *tuottaa* v:n. Tällöin joko $v = w$ tai/ja on ketju

$$w = w_0 \Rightarrow_R w_1 \Rightarrow_R \cdots \Rightarrow_R w_\ell = v,$$

ns. v:n *johto* w:stä. ℓ on johdon *pituus.*

[1] Uudelleenkirjoitusjärjestelmiä kutsutaan myös *puoli-Thue-systeemeiksi.* Varsinaisessa *Thuen systeemissä* vaaditaan lisäksi, että jos $p \to q$ on produktio, niin samoin on $q \to p$, ts. kukin produktio $p \leftrightarrow q$ on kaksisuuntainen.

[2] Jota kutsutaan sen *refleksiivis-transitiiviseksi sulkeumaksi.*

Sinällään RWS R ei tee muuta kuin tuottaa sanoista toisia sanoja. Jos kiinnitetään jokin joukko A ns. *lähtösanoja* eli *aksioomia*, voidaan määritellä RWS:n *generoima kieli*

$$L_{\text{gen}}(R, A) = \{v \mid w \Rightarrow_R^* v \text{ jollekin sanalle } w \in A\}.$$

Usein A:ssa on vain yksi sana tai se on äärellinen tai ainakin säännöllinen. Tällöin RWS toimii „kieliopin tapaan".

RWS voidaan saada toimimaan myös „automaatin tapaan" ottamalla käyttöön sallittujen *loppusanojen* kieli T, jolloin RWS R *tunnistaa kielen*

$$L_{\text{rec}}(R, T) = \{w \mid w \Rightarrow_R^* v \text{ jollekin sanalle } v \in T\}.$$

Usein T on säännöllinen. Tavallinen valinta on $T = \Delta^*$ jollekin Σ:n osajoukolle Δ. Tällöin Δ:n symbolit ovat ns. *loppusymbolit* eli *terminaalit* ja $\Sigma - \Delta$:n symbolit taas ns. *välisymbolit* eli *nonterminaalit.*

Esimerkki. *Deterministisestä äärellisestä automaatista* $M = (Q, \Sigma, q_0, \delta, B)$ *saadaan RWS kahdellakin tavalla. Tässä oletetaan, että* $\Sigma \cap Q$ *on tyhjä.*

Ensinnäkin muodostetaan RWS $R_1 = (\Omega, P_1)$*, jossa* $\Omega = \Sigma \cup Q$ *ja* P_1*:ssä ovat tarkalleen kaikki produktiot*

$$q_i a \to q_j \ , \quad \text{missä} \quad \delta(q_i, a) = q_j,$$

sekä lisäksi produktiot

$$a \to q_0 a \ , \quad \text{missä} \quad a \in \Sigma.$$

Kun valitaan T*:ksi kieli* $B + \Lambda$ *tai* B*, riippuen siitä onko* $L(M)$*:ssä* Λ *vai ei, on*

$$L_{\text{rec}}(R_1, T) \cap \Sigma^* = L(M).$$

Tyypillinen, kielen sanan $w = a_1 \cdots a_m$ *tunnistava johto on muotoa*

$$w \Rightarrow_{R_1} q_0 w \Rightarrow_{R_1} q_{i_1} a_2 \cdots a_m \Rightarrow_{R_1}^* q_{i_m},$$

missä q_{i_m} *on lopputila. Äärelliset automaatit ovat siis oleellisesti uudelleenkirjoitussysteemejä!*

Toinen tapa saada M*:stä RWS on ottaa* $R_2 = (\Omega,\ P_2)$*, jossa* P_2*:ssa ovat tarkalleen kaikki produktiot*

$$q_i \to a q_j \ , \quad \text{missä} \quad \delta(q_i, a) = q_j,$$

ja lisäksi produktio $q_i \to \Lambda$ *jokaiselle lopputilalle* q_i*. Tällöin*

$$L_{\text{gen}}\big(R_2, \{q_0\}\big) \cap \Sigma^* = L(M).$$

Tässä automaatista on oleellisesti tehty kielioppi!

Uudelleenkirjoitusjärjestelmien tuotto- ja generoimis/tunnistusmekanismia voidaan varioida monin eri tavoin.

Esimerkki. (**Markovin normaalialgoritmi**) *RWS:n produktiot annetaan järjestyksessä listana*

$$P : p_1 \to q_1, \ldots, p_n \to q_n$$

ja lisäksi valitaan P*:stä osajoukko* F*, ns.* loppuproduktiot*. Tuotossa sovitaan, että aina käytetään listan ensimmäistä soveltuvaa produktiota ja sitä sovelletaan sanassa alusta lukien ensimmäiseen soveltuvaan paikkaan. Eli jos* $p_i \to q_i$ *on ensimmäinen soveltuva*

produktio, niin sitä on käytettävä vasemmalta lukien ensimmäiseen uudelleenkirjoitettavan sanan osasanaan p_i. Johto pysähtyy, *kun ei löydy soveltuvaa produktiota tai kun on käytetty loppuproduktiota. Lähtien sanasta w normaalialgoritmi joko pysähtyy ja tuottaa yksikäsitteisen sanan v tai ei pysähdy lainkaan. Edellisessä tapauksessa sana v käsitetään* syötteen w aiheuttamaksi tulostukseksi, *jälkimmäisessä tapauksessa taas tulostusta ei tule. Normaalialgoritmeilla on universaalinen laskentakyky, eli niillä voi laskea sen mikä laskettavissa on. Niitä voi myös käyttää tunnistukseen: Sana (syöte) tunnistetaan, mikäli siitä lähtevä johto pysähtyy. Normaalialgoritmeilla on myös universaalinen tunnistuskyky.*

3.2 Kieliopit

Kielioppi on sellainen erikoistapaus uudelleenkirjoitusjärjestelmästä, jossa aakkosto on jaettu kahtia, *terminaaleihin* eli *loppusymboleihin* (eli *vakioihin*) ja *nonterminaaleihin* eli *välisymboleihin* (eli *muuttujiin*), ja valitaan yksi välisymboli aksioomaksi (vrt. edellä).

Määritelmä. Kielioppi[3] *on nelikkö $G = (\Sigma_\mathrm{N},\ \Sigma_\mathrm{T},\ X_0, P)$, jossa Σ_N on välisymbolien aakkosto, Σ_T on loppusymbolien aakkosto, $X_0 \in \Sigma_\mathrm{N}$ on aksiooma ja P muodostuu produktioista $p_i \to q_i$, missä p_i:ssä on ainakin yksi välisymboli.*

Jos G on kielioppi, niin $(\Sigma_\mathrm{N} \cup \Sigma_\mathrm{T}, P)$ on RWS, ns. *G:n indusoima RWS*. Merkitään $\Sigma = \Sigma_\mathrm{N} \cup \Sigma_\mathrm{T}$. „Perinteisesti" merkitään loppusymboleja pienillä kirjaimilla $(a, b, c, \ldots$ jne.) ja välisymboleja isoilla kirjaimilla $(X, Y, Z, \ldots$ jne.). G:n indusoimasta RWS:stä saadut $\Rightarrow$ ja $\Rightarrow^*$ antavat G:lle vastaavasti $\Rightarrow_G$:n ja $\Rightarrow_G^*$:n. G:n *generoima kieli* on silloin

$$L(G) = \{w \mid X_0 \Rightarrow_G^* w \text{ ja } w \in \Sigma_\mathrm{T}^*\}.$$

Kielioppi G on

- *yhteydetön* eli *CF-kielioppi,* jos jokaisessa produktiossa $p_i \to q_i$ vasen puoli p_i koostuu yhdestä ainoasta välisymbolista. Uudelleenkirjoitus ei nyt riipu siitä, missä „yhteydessä" eli ympäristössä välisymboli esiintyy.

- *lineaarinen,* jos se on CF ja jokaisen produktion oikealla puolella esiintyy enintään yksi välisymboli. CF-kielioppi, joka ei ole lineaarinen, on *epälineaarinen.*

- *yhteydellinen* eli *CS-kielioppi*[4]*,* jos jokaisessa produktiossa $p_i \to q_i$ on

  $$p_i = u_i X_i v_i \quad \text{ja} \quad q_i = u_i w_i v_i,$$

  missä $u_i, v_i \in \Sigma^*$, $X_i \in \Sigma_\mathrm{N}$ ja $w_i \in \Sigma^+$. Poikkeuksena mahdollisesti produktio $X_0 \to \Lambda$, mikäli X_0 ei esiinny minkään produktion oikealla puolella. Tällä tuodaan tarvittaessa tyhjä sana kieleen $L(G)$. Uudelleenkirjoitus riippuu nyt siitä, missä „yhteydessä" eli ympäristössä välisymboli X_i esiintyy.

- *pituutta kasvattava,* jos jokaisessa produktiossa $p_i \to q_i$ on $|p_i| \leq |q_i|$, paitsi mahdollisesti produktiossa $X_0 \to \Lambda$, mikäli X_0 ei esiinny minkään produktion oikealla puolella.

[3] Tarkemmin sanoen *generatiivinen* kielioppi. On olemassa myös ns. *analyyttinen* kielioppi, joka toimii duaalisesti automaatin tapaan.

[4] Jotkut määrittelevät CS-kieliopin yksinkertaisesti vain pituutta kasvattavaksi kieliopiksi. Kieliperheeseen tämä ei vaikuta.

Esimerkki. *Lineaarinen kielioppi*

$$G = \big(\{X\}, \{a, b\}, X, \{X \to \Lambda, X \to a, X \to b, X \to aXa, X \to bXb\}\big)$$

generoi palindromikielen L_{pal} *aakkostossa* $\{a, b\}$*. Kielioppi ei ole pituutta kasvattava (miksei?).*

Esimerkki. *Kielioppi*

$$\begin{aligned} G = \big(\{X_0, \$, X, Y\}, \{a\}, X_0, \{X_0 \to \$X\$, \$X \to \$Y, YX \to XXY,\\ Y\$ \to XX\$, X \to a, \$ \to \Lambda\}\big) \end{aligned}$$

generoi kielen $\{a^{2^n} \mid n \geq 0\}$*.* $\$$ *on reunamerkki ja* Y *„kulkee" vasemmalta oikealle kaksinkertaistaen* X*:iä. Jos produktioita* $X \to a$ *ja* $\$ \to \Lambda$ *käytetään „ennen aikojaan", ei ole mahdollista poistaa* Y*:tä. Kielioppi ei ole CF, CS eikä pituutta kasvattava.*

3.3 Chomskyn hierarkia

Ns. *Chomskyn hierarkiassa* kieliopit jaetaan neljään tyyppiin:

- *Tyyppi 0:* Ei mitään rajoituksia.
- *Tyyppi 1:* CS-kieliopit.
- *Tyyppi 2:* CF-kieliopit.
- *Tyyppi 3:* Lineaariset kieliopit, joissa produktiot ovat muotoa $X_i \to wX_j$ tai $X_j \to w$, missä X_i ja X_j ovat välisymboleja ja $w \in \Sigma_{\text{T}}^*$, ns. *oikealta lineaariset kieliopit.*[5]

Tyyppien 1 ja 2 kielioppien generoimina saadaan vastaavasti ns. *CS-kielet* ja *CF-kielet,* merkitään $\mathcal{CS}$ ja $\mathcal{CF}$.

Tyypin 0 kielioppien generoimia kieliä kutsutaan *laskettavasti numeroituviksi kieliksi* (*CE-kieliksi*), merkitään $\mathcal{CE}$. Nimi tulee siitä, että kunkin CE-kielen sanat voidaan algoritmisesti luetella listana, ts. on algoritmi, joka loputtomasti toimiessaan listaa kaikki kielen sanat, algoritmiksi käy tietysti itse kieliopin johtomekanismikin. Toisaalta muita kuin CE-kieliä ei tällä tavoin voida algoritmisesti listata. Tämä johtuu käytetystä ja yleisesti hyväksytystä algoritmin määrittelystä!

Tyypin 3 kielioppien generoimat kielet ovat tuttuja:

Lause 9. *Tyypin 3 kielioppien generoimat kielet ovat tarkalleen kaikki säännölliset kielet* $\mathcal{R}$*.*

Todistus. Tämä oli oleellisesti esimerkkinä Pykälässä 1. Jotta päästään oikealta lineaariseen kielioppiin, otetaan vain aksioomaksi q_0. Näytettäessä toisaalta, että oikealta lineaarinen kielioppi generoi säännöllisen kielen, käytetään kieliopin toimintaa matkivaa Λ-NFA:ta (jätetään harjoitukseksi). □

[5] On tietysti olemassa myös *vasemmalta lineaarinen* kielioppi, jossa sallitut produktiot ovat muotoa $X_i \to X_j w$ ja $X_j \to w$. Tyyppi 3 voitaisiin yhtä hyvin määritellä sitä käyttäen.

Chomskyn hierarkia voidaan näin myös käsittää kieliperheiden hierarkiaksi:

$$\mathcal{R} \subset \mathcal{CF} \subset \mathcal{CS} \subset \mathcal{CE}.$$

Kuten on todettu, palindromikieli L_{pal} vähintään kaksikirjaimisessa aakkostossa on CF, mutta ei säännöllinen. Myös muut sisältymiset ovat aitoja, kuten tullaan toteamaan.

Säännölliset kielet ovat suljettuja monien kielten operaatioiden suhteen, ts. operoitaessa säännöllisiin kieliin saadaan aina tuloksena säännöllinen kieli. Tällaisia operaatioita olivat mm. joukko-opin operaatiot, katenaatio, katenaatiosulkeuma ja peilikuva. Myös muut Chomskyn hierarkian kieliperheet ovat suljettuja sangen monien operaatioiden suhteen. Tämä itse asiassa tekee näistä perheistä luontevia kielten luokittelun takia: Laajempi perhe sisältää aina jollain tavalla radikaalisti erilaisia kieliä, ei vain entisistä joillain operaatioilla saatuja. Muut kuin $\mathcal{R}$ eivät kuitenkaan ole suljettuja edes kaikkien eo. operaatioiden suhteen, hankalia ovat leikkaus ja komplementti.

Apulause. *Kielioppi voidaan aina korvata saman tyypin kieliopilla, joka generoi saman kielen ja jossa ei esiinny loppusymboleja produktioiden vasemmilla puolilla.*

Todistus. Jos kielioppi on $G = (\Sigma_{\text{N}}, \Sigma_{\text{T}}, X_0, P)$, niin uusi kielioppi on $G' = (\Sigma'_{\text{N}}, \Sigma_{\text{T}}, X_0, P')$, jossa

$$\Sigma'_{\text{N}} = \Sigma_{\text{N}} \cup \Sigma'_{\text{T}} \quad , \quad \Sigma'_{\text{T}} = \{a' \mid a \in \Sigma_{\text{T}}\}$$

(Σ'_{T} on Σ_{T}:n erillinen „varjoaakkosto") ja P' saadaan P:stä vaihtamalla kunkin produktion jokainen loppusymboli a vastaavaksi „pilkulliseksi" symboliksi a' sekä lisäämällä produktiot $a' \to a$. □

Lause 10. *Kukin Chomskyn hierarkian kieliperhe on suljettu operaatioiden $\cup$, katenaatio, * ja $^+$ suhteen.*

Todistus. Asia tuli jo todetuksi perheen $\mathcal{R}$ tapauksessa. Jos kielet L ja L' generoidaan saman tyypin kieliopeilla

$$G = (\Sigma_{\text{N}}, \Sigma_{\text{T}}, X_0, P) \quad \text{ja} \quad G' = (\Sigma'_{\text{N}}, \Sigma'_{\text{T}}, X'_0, P'),$$

niin voidaan ensinnäkin ilmeisesti olettaa, että $\Sigma_{\text{N}} \cap \Sigma'_{\text{N}} = \emptyset$, ja toiseksi, että produktioiden vasemmilla puolilla ei esiinny loppusymboleja (eo. Apulause).

Silloin $L \cup L'$:n generoi saman tyypin kielioppi

$$H = (\Delta_{\text{N}}, \Delta_{\text{T}}, Y_0, Q),$$

missä

$$\Delta_{\text{N}} = \Sigma_{\text{N}} \cup \Sigma'_{\text{N}} \cup \{Y_0\} \quad , \quad \Delta_{\text{T}} = \Sigma_{\text{T}} \cup \Sigma'_{\text{T}},$$

Y_0 on uusi välisymboli ja Q saadaan seuraavasti:

1. Otetaan kaikki P:n ja P':n produktiot.
2. Jos kyseessä ovat Tyypin 1 kieliopit, poistetaan mahdolliset produktiot $X_0 \to \Lambda$ ja $X'_0 \to \Lambda$.
3. Lisätään produktiot $Y_0 \to X_0$ sekä $Y_0 \to X'_0$.
4. Jos kyseessä ovat Tyypin 1 kieliopit ja L:ssä tai L':ssä on Λ, lisätään produktio $Y_0 \to \Lambda$.

Kielen LL' taas generoi kielioppi F, kun kohdat 3. ja 4. korvataan kohdilla

3'. Lisätään produktio $Y_0 \to X_0 X_0'$. Jos kyseessä ovat Tyypin 1 kieliopit ja kielessä L on Λ, lisätään produktio $Y_0 \to X_0'$, ja vastaavasti, jos kielessä L' on Λ, lisätään produktio $Y_0 \to X_0$.

4'. Jos kyseessä ovat Tyypin 1 kieliopit ja L:ssä sekä L':ssä on Λ, lisätään produktio $Y_0 \to \Lambda$.

Jälleen tyyppi säilyy. Huomaa miten on tärkeää, että esitetyt oletukset ovat voimassa, muuten vierekkäiset johdot voisivat Tyyppien 0 ja 1 tapauksessa sekaantua toisiinsa.

Jos G on Tyyppiä 2, niin L^*:n generoi yksinkertaisesti saman tyypin kielioppi

$$K = \big(\Sigma_{\mathrm{N}} \cup \{Y_0\}, \Sigma_{\mathrm{T}}, Y_0, Q\big),$$

missä Q saadaan P:stä lisäämällä siihen produktiot

$$Y_0 \to \Lambda \quad \text{ja} \quad Y_0 \to Y_0 X_0.$$

Korvaamalla produktio $Y_0 \to \Lambda$ produktiolla $Y_0 \to X_0$, saadaan L^+.

Tyypille 1 konstruktio on hieman mutkikkaampi. Jos G on Tyyppiä 1, lisätään vielä yksi uusi välisymboli Y_1, ja Q saadaan seuraavasti: Poistetaan P:stä mahdollinen produktio $X_0 \to \Lambda$ ja lisätään produktiot

$$Y_0 \to \Lambda \quad , \quad Y_0 \to X_0 \quad \text{sekä} \quad Y_0 \to Y_1 X_0.$$

Lisätään vielä jokaista loppusymbolia a kohti produktiot

$$Y_1 a \to Y_1 X_0 a \quad \text{ja} \quad Y_1 a \to X_0 a.$$

Jättämällä produktio $Y_0 \to \Lambda$ tarvittaessa pois saadaan L^+. Huomaa miten jälleen oletus, ettei produktioiden vasemmilla puolilla ole loppusymboleja, estää vierekkäisten johtojen sekaantumisen toisiinsa, sillä uusi johto voidaan aloittaa vasta, kun viereinen jo alkaa loppusymbolilla.

Tyypille 0 konstruktio on hyvin samantapainen kuin Tyypille 1. □

Varsin helppo on muuten vielä todeta, että kukin hierarkian perhe on suljettu peilikuvan suhteen.

Edellä olevasta saadaan muitakin kieliperheitä kuin Chomskyn hierarkiaan kuuluvat, mm.

- lineaaristen kielioppien generoimat *lineaariset kielet* (perhe $\mathcal{LIN}$),
- CE-kielten komplementit, ns. *co–CE-kielet* (perhe co$-\mathcal{CE}$), ja
- $\mathcal{CE}$:n ja co$-\mathcal{CE}$:n leikkaus, ns. *laskettavat kielet* (perhe $\mathcal{C}$).

Laskettavat kielet ovat kielet, joiden jäsenyysprobleema voidaan algoritmisesti ratkaista, listataan vain vuorotellen kielen ja sen komplementin sanoja ja katsotaan kumpaan tutkittu sana kuuluu.

Yo. kieliperheissä ei ole erikseen mukana pituutta kasvattavien kielioppien generoimien kielien perhettä, sillä se on $\mathcal{CS}$.

Lause 11. *Jokainen pituutta kasvattava kielioppi voidaan korvata CS-kieliopilla, joka generoi saman kielen.*

Todistus. Katsotaan ensin tapaus, jossa pituutta kasvattavassa kieliopissa $G = (\Sigma_\mathrm{N}, \Sigma_\mathrm{T}, X_0, P)$ on vain yksi ei-sallittu pituutta kasvattava produktio $p \to q$, ts. kielioppi

$$G' = \big(\Sigma_\mathrm{N}, \Sigma_\mathrm{T}, X_0, P - \{p \to q\}\big)$$

on CS-kielioppi.

Eo. Apulauseen nojalla voidaan olettaa, että G:n produktioiden vasemmilla puolilla ei ole loppusymboleja. Näytetään miten G muunnetaan CS-kieliopiksi $G_1 = (\Delta_\mathrm{N}, \Sigma_\mathrm{T}, X_0, Q)$. Merkitään sitä varten

$$p = U_1 \cdots U_m \quad \text{ja} \quad q = V_1 \cdots V_n,$$

missä U_i:t ja V_j:t ovat välisymboleja ja $n \geq m \geq 2$. Valitaan uudet välisymbolit $Z_1, \ldots, Z_m$ ja otetaan Δ_N:ksi $\Sigma_\mathrm{N} \cup \{Z_1, \ldots, Z_m\}$. Q:hun otetaan P:n produktiot lukuunottamatta tietysti produktiota $p \to q$. Vm. produktio korvataan lisätyillä produktioilla

$$\begin{aligned}
\underline{U_1} U_2 \cdots U_m &\to \underline{Z_1} U_2 \cdots U_m, \\
Z_1 \underline{U_2} U_3 \cdots U_m &\to Z_1 \underline{Z_2} U_3 \cdots U_m, \\
&\vdots \\
Z_1 \cdots Z_{m-1} \underline{U_m} &\to Z_1 \cdots Z_{m-1} \underline{Z_m V_{m+1} \cdots V_n}, \\
\underline{Z_1} Z_2 \cdots Z_m V_{m+1} \cdots V_n &\to \underline{V_1} Z_2 \cdots Z_m V_{m+1} \cdots V_n, \\
&\vdots \\
V_1 \cdots V_{m-1} \underline{Z_m} V_{m+1} \cdots V_n &\to V_1 \cdots V_{m-1} \underline{V_m} V_{m+1} \cdots V_n.
\end{aligned}$$

(Alleviivaus osoittaa uudelleenkirjoituksen.) Tuloksena on CS-kielioppi G_1, joka generoi saman kielen kuin G. Huomaa, miten aina on sovellettava peräkkäin koko yo. produktioketjua, jotta päästään eteenpäin. Jos ketjun aikana voitaisiin soveltaa muita produktioita, niitä voidaan silloin soveltaa jo ennen ketjua tai vasta sen jälkeen.

Yleinen pituutta kasvattava kielioppi G muunnetaan CS-kieliopiksi nyt seuraavasti. Jälleen voidaan ottaa vain tapaus, jossa produktioiden vasemmilla puolilla ei ole loppusymboleja. Merkitään nyt G':lla kielioppia, joka saadaan poistamalla G:stä kaikki ei-sallitut produktiot. Lisätään sitten poistetut produktiot yksi kerrallaan kielioppiin G' muuntaen se joka askeleella ekvivalentiksi CS-kieliopiksi yllä kuvatulla tavalla. Tuloksena on CS-kielioppi, joka generoi saman kielen kuin G. □

Luku 4

CF-KIELET

4.1 Sanan jäsennys

CF-kieliopin produktiot, joissa on vasemmalla puolella sama välisymboli, on usein tapana kirjoittaa yhteen. Jos esimerkiksi kaikki ne produktiot, joissa on vasemmalla puolella välisymboli X, ovat

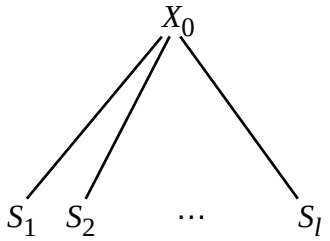
$$X \to w_1 \quad , \ldots , \quad X \to w_t,$$

kirjoitetaan nämä muodossa

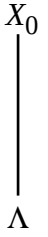
$$X \to w_1 \mid w_2 \mid \cdots \mid w_t.$$

Silloin on tietysti varottava käyttämästä pystyviivaa $\mid$ myös kieliopin symbolina!

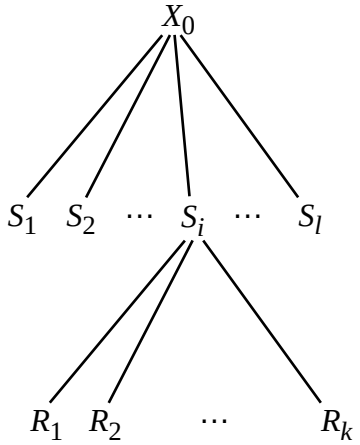
Tarkastellaan CF-kielioppia $G = (\Sigma_{\mathrm{N}}, \Sigma_{\mathrm{T}}, X_0, P)$ ja merkitään $\Sigma = \Sigma_{\mathrm{N}} \cup \Sigma_{\mathrm{T}}$. G:n johtoon $X_0 \Rightarrow_G^* w$ voidaan aina liittää ns. *jäsennyspuu.* Puun pisteet merkitään Σ:n symboleilla tai Λ:lla. Aksioomalla X_0 merkitään puun *juuri.* Puu konstruoidaan seuraavasti. Lähdetään juuripisteestä. Jos ensimmäinen käytetty produktio on $X_0 \to S_1 \cdots S_\ell$, missä $S_1, \ldots, S_\ell \in \Sigma$, jatketaan puuta ℓ:llä pisteellä, jotka merkitään vasemmalta oikealle symbolein $S_1, \ldots, S_\ell$:



Jos taas ensimmäinen käytetty produktio on $X_0 \to \Lambda$, jatketaan puuta yhdellä pisteellä, joka merkitään Λ:lla:



Mikäli johdon toinen produktio kohdistuu sen toisen sanan symboliin S_i ja on $S_i \to R_1 \cdots R_k$, jatketaan puuta sen vastaavasta S_i:llä merkitystä pisteestä k:lla pisteellä, jotka merkitään taas vasemmalta oikealle symbolein $R_1, \ldots, R_k$ (tapaus $S_i \to \Lambda$ vastaavasti):



Tällä tavoin jatketaan puun konstruktiota, kunnes se on saatu kokonaan. Puuta voidaan aina jatkaa mistä tahansa „vapaasta" välisymbolista, ei pelkästään viimeksi saaduista. Huomaa, että kun puun piste on merkitty loppusymbolilla tai Λ:lla, ei siitä enää voida puuta jatkaa. Tällaisia pisteitä kutsutaan puun *lehdiksi.* Johdon tuottama sana luetaan puun lehdistä katenoiden vasemmalta oikealle.

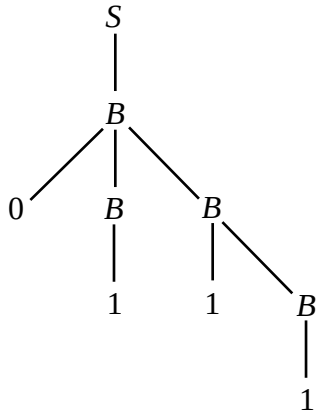
Esimerkki. *Kieliopin*

$$G = \big(\{A, B, S\}, \{0, 1\}, S, \{S \to A \mid B, A \to 0 \mid 0A \mid 1AA \mid AA1 \mid A1A,$$
$$B \to 1 \mid 1B \mid 0BB \mid BB0 \mid B0B\}\big)$$

johtoa

$$S \Rightarrow B \Rightarrow 0BB \Rightarrow 0B1B \Rightarrow 011B \Rightarrow 0111$$

vastaa jäsennyspuu



Kielioppi muuten generoi tarkalleen kaikki sanat, joissa on eri määrä 1*:siä ja* 0*:ia.*

Jäsennyspuut tuovat mieleen luonnollisten kielten kieliopeista tutun lauseenjäsennyksen ja myös tiettyjen ohjelmointikielten ohjelmien jäsennyksen.

Esimerkki. *Englanninkielessä yksinkertaiset lauseenjäsennyssäännöt voisivat olla vaikkapa muotoa*

$$\begin{aligned}
\langle \text{declarative sentence} \rangle &\to \langle \text{subject} \rangle \langle \text{verb} \rangle \langle \text{object} \rangle \\
\langle \text{subject} \rangle &\to \langle \text{proper noun} \rangle \\
\langle \text{proper noun} \rangle &\to \textit{Alice} \mid \textit{Bob} \\
\langle \text{verb} \rangle &\to \textit{reminded} \\
\langle \text{object} \rangle &\to \langle \text{proper noun} \rangle \mid \langle \text{reflexive pronoun} \rangle \\
\langle \text{reflexive pronoun} \rangle &\to \textit{himself} \mid \textit{herself}
\end{aligned}$$

joista tunnistaa välittömästi CF-kieliopin. Suomenkieli on tietysti vaikeampi sijapäätteiden yms. takia.

Esimerkki. *C-kielessä yksinkertaiset syntaksisäännöt voisivat olla vaikkapa*

$$\begin{aligned}
\langle \text{statement} \rangle &\to \langle \text{statement} \rangle \langle \text{statement} \rangle \mid \langle \textbf{for}\text{-statement} \rangle \mid \langle \textbf{if}\text{-statement} \rangle \mid \cdots \\
\langle \textbf{for}\text{-statement} \rangle &\to \textbf{for} \ (\ \langle \text{expression} \rangle\ ;\ \langle \text{expression} \rangle\ ;\ \langle \text{expression} \rangle\ )\ \langle \text{statement} \rangle \\
\langle \textbf{if}\text{-statement} \rangle &\to \textbf{if} \ (\ \langle \text{expression} \rangle\ )\ \langle \text{statement} \rangle \\
\langle \text{compound} \rangle &\to \{\ \langle \text{statement} \rangle\ \}
\end{aligned}$$

jne., joista taas tunnistaa CF-kieliopin rakenteen.

Johto on *vasen johto,* jos siinä aina jatketaan vasemmalta lukien ensimmäisestä soveltuvasta välisymbolista. Jokainen johto voidaan aina korvata vasemmalla johdolla. Tämä on selvää siksikin, että jäsennyspuissa ei näy se, missä järjestyksessä produktioita on sovellettu, ja jokaista jäsennyspuuta vastaa näin aina myös jokin vasen johto.

CF-kielioppi G on *moniselitteinen,* jos jollekin $L(G)$:n sanalle on kaksi erilaista vasenta johtoa tai ekvivalentisti kaksi erilaista jäsennyspuuta. CF-kielioppi, joka ei ole moniselitteinen, on *yksiselitteinen.* Luonnollisten kielten jäsennystä vastaavat kieliopit ovat moniselitteisiä, asiayhteydestä selviää lauseen merkitys. Ohjelmointikielissä moniselitteisyyttä ei saisi esiintyä.

Moniselitteisyys on paremminkin kieliopin kuin kielen ominaisuus. On kuitenkin CF-kieliä, joita ei voida lainkaan generoida yksiselitteisillä CF-kieliopeilla, ns. *luontaisesti moniselitteiset CF-kielet.*

Esimerkki. *Kielioppi*

$$G = \big(\{S, T, F\}, \{a, +, \times, (,)\}, S, \{S \to S + T \mid T, T \to T \times F \mid F, F \to (S) \mid a\}\big)$$

generoi yksinkertaisia aritmeettisia lausekkeita. a on lukujen yms. „paikanpitäjä". Todetaan, että G on yksiselitteinen. Näytetään tämä täydellisellä induktiolla johdetun lausekkeen pituuden ℓ suhteen.

Induktion lähtökohta on $\ell = 1$, joka on selvä, sillä ainoa tapa johtaa a on

$$S \Rightarrow T \Rightarrow F \Rightarrow a.$$

Oletetaan sitten, että pituuteen $p - 1$ asti kaikkien $L(G)$:n sanojen vasemmat johdot ovat yksiselitteisiä, ja tarkastellaan pituutta p olevan $L(G)$:n sanan w vasenta johtoa.

Otetaan ensin tapaus, jossa w:ssä on ainakin yksi symboli $+$, joka ei ole sulkujen sisällä. T:n ja F:n kautta johdetut $+$:n esiintymät ovat sulkujen sisällä, joten mainittu $+$

voidaan johtaa vain alkuproduktiolla $S \to S + T$, jossa oleva $+$ on sanan w viimeinen $+$, joka ei ole sulkujen sisällä. w:n vasen johto on siis muotoa

$$S \Rightarrow S + T \Rightarrow^* u + T \Rightarrow^* u + v = w.$$

Siinä olevat „alijohdot" $S \Rightarrow^* u$ ja $T \Rightarrow^* v$ ovat vasempia johtoja ja siten induktio-oletuksen nojalla yksiselitteisiä, joten myös w:n vasen johto on yksiselitteinen. Huomaa, että v:kin on kielessä $L(G)$ ja sen vasen johto $S \Rightarrow T \Rightarrow^* v$ on yksiselitteinen.

Samaan tapaan käsitellään tapaus, jossa w:ssä on ainakin yksi $\times$ sulkujen ulkopuolella, mutta ei $+$. Silloin tämän $\times$:n tuottaa joko S tai T. S:stä lähtevän johdon alkupää on $S \Rightarrow T \Rightarrow T \times F$ ja T:stä lähtevän $T \Rightarrow T \times F$. Jälleen esiintyvä $\times$ on w:n viimeinen sulkujen ulkopuolella oleva $\times$ ja vasen johto on muotoa

$$S \Rightarrow T \Rightarrow T \times F \Rightarrow^* u \times F \Rightarrow^* u \times v = w,$$

josta induktio-oletuksen nojalla päätellään, että w:llä on vain yksi vasen johto.

Vielä tarkistetaan tapaus, jossa jokainen $+$ sekä $\times$ on sulkujen sisällä. Silloin w:n johdon alku on

$$S \Rightarrow T \Rightarrow F \Rightarrow (S),$$

joten w on muotoa (u). Koska myös u kuuluu $L(G)$:hen, on sen vasen johto induktio-oletuksen mukaan yksiselitteinen, joten sama pätee w:lle.

4.2 Normaalimuodot

CF-kielioppien muotoa voidaan monin tavoin rajoittaa pienentämättä generoitujen kielten perhettä. Sellaisenaan CF-kielioppi ei esimerkiksi ole CS eikä pituutta kasvattava, mutta se voidaan korvata tällaisella CF-kieliopilla.

Lause 12. *Jokainen CF-kieli voidaan generoida pituutta kasvattavalla CF-kieliopilla.*

Todistus. Konstruoidaan CF-kieliopin $G = (\Sigma_\mathrm{N}, \Sigma_\mathrm{T}, X_0, P)$ kanssa saman kielen generoiva pituutta kasvattava CF-kielioppi

$$G' = \big(\Sigma_\mathrm{N} \cup \{S\}, \Sigma_\mathrm{T}, S, P'\big).$$

Jos $\Lambda \in L(G)$, otetaan S:lle produktiot $S \to \Lambda \mid X_0$, muuten vain produktio $S \to X_0$. Muiden produktioiden saamiseksi määritellään ensin rekursiivisesti välisymbolijoukko Δ_Λ:

1. Jos P:ssä on produktio $Y \to \Lambda$, niin $Y \in \Delta_\Lambda$.
2. Jos P:ssä on produktio $Y \to w$, missä $w \in \Delta_\Lambda^+$, niin $Y \in \Delta_\Lambda$.
3. Vain ne välisymbolit ovat Δ_Λ:ssa, jotka sinne saadaan kohtien 1. ja 2. kautta.

Muut kuin välisymbolia S koskevat P':n produktiot saadaan nyt P:n produktioista seuraavasti:

(i) Poistetaan kaikki muotoa $Y \to \Lambda$ olevat produktiot.

(ii) Jokaista produktiota $Y \to w$ kohti, missä w:ssä esiintyy ainakin yksi Δ_Λ:n symboli, lisätään kaikki produktiot, jotka saadaan siitä poistamalla w:stä yksi tai useampi Δ_Λ:n symboli, mutta ei sen kaikkia symboleita.

On ilmeistä, että $L(G') \subseteq L(G)$, sillä G:n vastaavissa johdoissa voidaan esiintyvät Δ_Λ:n symbolit tarvittaessa pyyhkiä pois. Toisaalta jokaisesta G:n johdosta saadaan vastaava G':n johto. Λ:n johdon tapaus on selvä, joten siirrytään ei-tyhjän sanan v johtoon. Asia on selvä, jos käytetyt produktiot ovat P':ssä. Muussa tapauksessa näytetään, miten v:n jäsennyspuusta G:ssä ($= \mathcal{T}$) saadaan sen jäsennyspuu G':ssa ($= \mathcal{T}'$). Nyt $\mathcal{T}$:ssä täytyy olla Λ-lehtiä. Puun pistettä, josta lähtee vain Λ-lehtiin päättyviä jatkohaaroja, sanotaan Λ-pisteeksi. Seurataan Λ-lehdestä reittiä taaksepäin niin kauan kuin siinä on vain Λ-pisteitä. Tällöin ei voida päästä aksioomaan asti, sillä muutoin olisi kyseessä Λ:n johto. Poistetaan puusta $\mathcal{T}$ kaikki reitin pisteet kaikkine jälkeläisineen. Jos Λ-lehtiä on jäljellä, toistetaan operaatio. Tuloksena on G':n jäsennyspuu $\mathcal{T}'$, jonka lehdistä voi lukea sanan v. □

Tässä vaiheessa saadaan edellisen lauseen ja Lauseen 11 seurauksena keskeinen Chomskyn hierarkian tulos:

Seuraus. $\mathcal{CF} \subseteq \mathcal{CS}$

Produktio $X \to Y$ on *yksikköproduktio,* jos Y on välisymboli. Hyvin samantapaisella päättelyllä kuin edellä voidaan todistaa

Lause 13. *Jokainen CF-kieli voidaan generoida CF-kieliopilla, jossa ei ole yksikköproduktioita. Lisäksi voidaan olettaa, että kielioppi on pituutta kasvattava.*

Todistus. Esitetään vain todistuksen pari pääkohtaa. Merkitään Δ_X:llä kaikkien niiden välisymbolien ($\neq X$) joukkoa, jotka saadaan välisymbolista X vain yksikköproduktioita käyttäen. Kieliopin $G = (\Sigma_\mathrm{N}, \Sigma_\mathrm{T}, X_0, P)$ kanssa saman kielen generoiva CF-kielioppi

$$G' = (\Sigma_\mathrm{N}, \Sigma_\mathrm{T}, X_0, P'),$$

jossa ei ole yksikköproduktioita, saadaan, kun P muunnetaan P':ksi seuraavalla menettelyllä:

1. Etsitään G:n välisymboleille X joukot Δ_X.
2. Poistetaan P:stä yksikköproduktiot.
3. Jos $Y \in \Delta_X$ ja P:ssä on produktio $Y \to w$, joka ei ole yksikköproduktio, lisätään P':uun produktio $X \to w$.

Ilmeisesti, jos G on pituutta kasvattava, niin G' on myös sitä. □

CF-kielioppi on ns. *Chomskyn normaalimuodossa,* jos sen produktiot ovat muotoa

$$X \to YZ \quad \text{tai} \quad X \to a,$$

missä X, Y ja Z ovat välisymboleja ja a on loppusymboli, poikkeuksena mahdollisesti produktio $X_0 \to \Lambda$, mikäli aksiooma X_0 ei esiinny minkään produktion oikealla puolella.

Kieliopin muuntaminen Chomskyn normaalimuotoon aloitetaan muuntamalla se pituutta kasvattavaksi kieliopiksi, jossa ei ole yksikköproduktioita (Lause 13). Seuraavaksi muunnetaan kielioppia siten, että ainoat produktiot, joissa esiintyy loppusymboleja, ovat muotoa $X \to a$, missä a on loppusymboli, ks. Pykälän 3.2 Apulause ja sen todistus. Sen jälkeen produktiot ovatkin joko mainittua muotoa $X \to a$ tai muotoa

$$X \to Y_1 \cdots Y_k,$$

missä $Y_1, \ldots, Y_k$ ovat välisymboleja, poikkeuksena mahdollinen produktio $X_0 \to \Lambda$. Mainittua muotoa $X \to Y_1 \cdots Y_k$ oleva produktio korvataan useilla normaalimuotoisilla produktioilla

$$\begin{aligned} X &\to Y_1 Z_1 \\ Z_1 &\to Y_2 Z_2 \\ &\vdots \\ Z_{k-3} &\to Y_{k-2} Z_{k-2} \\ Z_{k-2} &\to Y_{k-1} Y_k \end{aligned}$$

missä $Z_1, \ldots, Z_{k-2}$ ovat uusia välisymboleja, joita ei saa käyttää muiden produktioiden korvaamisessa. Näin on saatu

Lause 14. *Jokainen CF-kieli voidaan generoida Chomskyn normaalimuotoisella CF-kieliopilla.*

Toinen klassinen normaalimuoto on *Greibachin normaalimuoto.* CF-kielioppi on Greibachin normaalimuodossa, jos sen produktiot ovat muotoa

$$X \to aw,$$

missä a on loppusymboli ja w on tyhjä tai muodostuu välisymboleista. Jälleen poikkeuksena on mahdollinen produktio $X_0 \to \Lambda$, mikäli aksiooma X_0 ei esiinny minkään produktion oikealla puolella. Jokainen CF-kielioppi voidaan saattaa myös Greibachin normaalimuotoon, mutta sen todistaminen on hankalampaa, ks. esimerkiksi Simovici & Tenney.

Greibachin normaalimuotoinen kielioppi muistuttaa oikealta lineaarista kielioppia siinä, että se generoi vasemmissa johdoissa sanat vasemmalta oikealle. Sinällään oikealta lineaarinen kielioppi ei kuitenkaan ole Greibachin normaalimuotoa.

4.3 Pinoautomaatti

Kieliä, joiden indeksi on ääretön, ei voida tunnistaa äärellisellä automaatilla. Toisaalta aivan yleisen äärettömän muistin hallinta on vaikeaa—ja johtaa vallan toisenlaiseen teoriaankin—joten yleensä rajoitutaan *potentiaalisesti äärettömään muistiin.* Potentiaalisesti äärettömästä muistista on „käytössä" aina vain jokin äärellinen osa ja loput muistipaikat ovat tietyssä vakiotilassa („tyhjiä"). Riippuen siitä, millä tavalla muistia otetaan käyttöön ja miten sitä käytetään, saadaan erilaisia automaatteja.

Koska on CF-kieliä, joiden indeksi on ääretön—esimerkiksi palindromikielet, jos aakkosto ei ole yksikirjaiminen—tarvitaan CF-kielten tunnistamiseen ääretöntilainen automaatti. Sen muisti on tietysti potentiaalisesti ääretön ja aivan erityistä tyyppiä, nimittäin ns. *pinomuisti.* Pinomuistin sisältö voidaan kuvata sanalla, josta on näkyvissä eli luettavissa ja muutettavissa vain ensimmäinen symboli. Alussa pinossa on jokin pohjasymboli. Pinomuistin lisäksi automaatilla on käytössä „tavallinen" äärellinen tilamuisti, kuten Λ-NFA:lla.

Määritelmä. Pinoautomaatti *(*PDA*) on seitsikko* $M = (Q, \Sigma, \Gamma, S, Z, \delta, A)$*, missä*

- $Q = \{q_1, \ldots, q_m\}$ *on äärellinen* tilojen joukko, *jonka alkioita kutsutaan* tiloiksi;
- Σ *on* syöteaakkosto, *kielen aakkosto;*

- Γ *on äärellinen* pinoaakkosto *eli pinossa esiintyvien symbolien joukko;*
- $S \subseteq Q$ *on* alkutilojen joukko;
- $Z \subseteq \Gamma$ *on pinon* pohjasymbolien joukko;
- δ *on* siirtofunktio, *joka kuvaa jokaisen kolmikon* (q_i, a, X)*, missä* q_i *on tila,* a *on syötesymboli tai* Λ *ja* X *on pinosymboli, tarkalleen yhdeksi äärelliseksi parien* (q, α) *joukoksi* T*, missä* q *on tila ja* α *on pinoaakkoston sana; vrt.* Λ*-NFA;*
- $A \subseteq Q$ *on* lopputilojen joukko.

Jotta PDA:n tilastruktuurin käsittely voidaan määritellä, otetaan käyttöön kolmikot (q_i, x, α), missä q_i on tila, x on syötteen lukematon osa (suffiksi) ja α on pinon sisältö sanana, „päällimmäinen" symboli vasemmalla. Näitä kolmikoita kutsutaan M:n *konfiguraatioiksi.*

Nyt ei voida aivan helposti määritellä „hattufunktiota" ja „tähtifunktiota" kuten tehtiin Λ-NFA:lle, koska muisti on jaettu kahteen osaan, tilaan ja pinoon. Määritelläänkin asia käyttäen konfiguraatioita. Konfiguraation (q_j, y, β) sanotaan olevan konfiguraation (q_i, x, α) *suora seuraaja,* merkitään

$$(q_i, x, \alpha) \vdash_M (q_j, y, \beta),$$

jos

$$x = ay \quad , \quad \alpha = X\gamma \quad , \quad \beta = \epsilon\gamma \quad \text{ja} \quad (q_j, \epsilon) \in \delta(q_i, a, X).$$

Huomaa, että tässä a on joko syötesymboli tai Λ. Edelleen määritellään vastaava „tähtirelaatio" $\vdash_M^*$ seuraavasti:

1. $(q_i, x, \alpha) \vdash_M^* (q_i, x, \alpha)$
2. Jos $(q_i, x, \alpha) \vdash_M (q_j, y, \beta)$, niin myös $(q_i, x, \alpha) \vdash_M^* (q_j, y, \beta)$.
3. Jos $(q_i, x, \alpha) \vdash_M^* (q_j, y, \beta)$ ja $(q_j, y, \beta) \vdash_M (q_k, z, \gamma)$, niin myös $(q_i, x, \alpha) \vdash_M^* (q_k, z, \gamma)$.
4. $(q_i, x, \alpha) \vdash_M^* (q_j, y, \beta)$ vain, jos se seuraa edellisistä kohdista 1.–3.

Jos $(q_i, x, \alpha) \vdash_M^* (q_j, y, \beta)$, sanotaan, että (q_j, y, β) on (q_i, x, α):n *seuraaja.*

PDA M *hyväksyy*[1] syötesanan w, jos

$$(q_i, w, X) \vdash_M^* (q_j, \Lambda, \alpha)$$

jollekin alkutilalle $q_i \in S$, pohjasymbolille $X \in Z$, lopputilalle $q_j \in A$ ja pinolle α. M:n *tunnistama kieli* $L(M)$ muodostuu tarkalleen kaikista sen hyväksymistä sanoista.

Näin määritelty pinoautomaatti on luonteeltaan epädeterminstinen. Siirtovalintoja on siis yleisesti useita. Erityisesti on mahdollista, että mitään seuraavaa siirtoa ei ole, jolloin PDA:n toiminta pysähtyy ilman hyväksymistä, ellei olla lopputilassa ja syöte ole luettu loppuun. Näin käy esimerkiksi, kun pino on tyhjä.

[1] Tämä on ns. *hyväksyminen lopputilalla.* Pinon sisällöllä ei siis ole vaikutusta hyväksymiseen. Toinen hyväksymistapa on *hyväksyminen tyhjällä pinolla.* Tällöin syöte w hyväksytään, jos $(q_i, w, X) \vdash_M^* (q_j, \Lambda, \Lambda)$ jollekin alkutilalle q_i, jollekin pohjasymbolille X ja jollekin tilalle q_j. Lopputiloja ei siis tarvita. Ei ole kovin vaikea todeta, että nämä kaksi eri tunnistustapaa antavat saman kieliperheen. Ks. Apulauseen todistus jäljempänä.

Lause 15. *Jokainen CF-kieli voidaan tunnistaa PDA:lla. Lisäksi PDA:ssa tarvitaan tällöin vain kolme tilaa, alkutila, välitila ja lopputila, sekä yksi pohjasymboli.*

Todistus. Asian yksinkertaistamiseksi voidaan olettaa, että CF-kielioppi $G = (\Sigma_\mathrm{N}, \Sigma_\mathrm{T}, X_0, P)$ on valmiiksi Chomskyn normaalimuodossa.[2] Konstruoidaan PDA

$$M = \big(\{A, V, T\}, \Sigma_\mathrm{T}, \Sigma_\mathrm{N} \cup \{U\}, \{A\}, \{U\}, \delta, \{T\}\big),$$

missä δ saadaan seuraavilla säännöillä:

- Jos $X \to YZ$ on G:n produktio, niin $(V, YZ) \in \delta(V, \Lambda, X)$.
- Jos $X \to a$ on G:n produktio, missä $a \in \Sigma_\mathrm{T}$ tai $a = \Lambda$, niin $(V, \Lambda) \in \delta(V, a, X)$.
- Vielä tarvitaan alkusiirto $\delta(A, \Lambda, U) = \big\{(V, X_0U)\big\}$ sekä loppusiirto $\delta(V, \Lambda, U) = \big\{(T, \Lambda)\big\}$.

Pinosymbolit ovat siis G:n välisymboleja. G:n vasemmat johdot ja M:n laskut vastaavat tarkasti toisiaan: Kun G sanan $w = uv$ vasemmassa johdossaan on menossa sanassa $u\alpha$, missä $u \in \Sigma_\mathrm{T}^*$ ja $\alpha \in \Sigma_\mathrm{N}^+$, on M:n konfiguraatio $(V, v, \alpha U)$. Sanaa w itseään vastaa hyväksyvä loppukonfiguraatio (T, Λ, Λ). □

Käänteinenkin tulos pätee. Sitä varten tarvitaan ensin aputulos, jolla rajoitetaan PDA:n rakennetta samantapaiseksi kuin yo. todistuksessa, muuttamatta tietysti sen tunnistamaa kieltä.

Apulause. *Jokainen PDA voidaan korvata ekvivalentilla PDA:lla, jonka pino tyhjenee tarkalleen silloin, kun se tulee lopputilaan.*

Todistus. Jos asianlaita ei PDA:lle $M = (Q, \Sigma, \Gamma, S, Z, \delta, A)$ ole kuten vaaditaan, tehdään eräitä muutoksia. Otetaan ensinnäkin uusi pohjasymboli U ja lisätään tilasiirrot

$$(q_i, XU) \in \delta(q_i, \Lambda, U) \quad (q_i \in S \text{ ja } X \in Z).$$

Edelleen lisätään uudet tilat V ja T ja tilasiirrot

$$(V, X) \in \delta(q_i, \Lambda, X) \quad (q_i \in A \text{ ja } X \in \Gamma),$$

$$\delta(V, \Lambda, X) = \big\{(V, \Lambda)\big\} \quad (X \in \Gamma)$$

ja

$$\delta(V, \Lambda, U) = \big\{(T, \Lambda)\big\}.$$

Uusi pohjasymbolien joukko on nyt $\{U\}$ ja uusi lopputilajoukko on $\{T\}$. □

Lause 16. *PDA:n tunnistama kieli on aina CF-kieli.*

Todistus. Otetaan tarkasteltavaksi PDA $M = (Q, \Sigma, \Gamma, S, Z, \delta, A)$ ja näytetään, että $L(M)$ on CF. Voidaan olettaa, että M on edellisen Apulauseen mainitsemaa muotoa. Silloin M hyväksyy syötteen tarkalleen silloin, kun sen pino tyhjenee syötteen tultua luetuksi loppuun. Ideana on sisällyttää tila johdossa vasempana olevaan kieliopin välisymboliin. Uudet välisymbolit olisivat jotain muotoa $[X, q_i]$, missä $X \in \Gamma$ ja $q_i \in Q$. Tilaa voidaan

[2] Riittäisi itse asiassa olettaa, että jos produktion oikealla puolella on loppusymboleja, niin niitä on yksi ja se on ensimmäinen symboli. Jos lähdettäisiinkin Greibachin normaalimuotoisesta CF-kieliopista, saataisiin PDA, jossa on vain kaksi tilaa ja jossa ei ole Λ-siirtoja.

päivittää uudelleenkirjoitettaessa. Pulmana on kuitenkin pinon päällimmäisen symbolin pyyhkiminen eli korvaaminen Λ:lla, silloin tilaa ei voida päivittää. Tätä varten tuodaan mukaan vielä uuden tilan q_j „arvaus", ja välisymbolit ovatkin kolmikoita

$$[q_i, X, q_j],$$

missä $X \in \Gamma$ ja $q_i, q_j \in Q$. Merkitään

$$\Delta = \big\{[q_i, X, q_j] \;\big|\; q_i, q_j \in Q \text{ ja } X \in \Gamma\big\}.$$

Produktiot saadaan seuraavilla säännöillä, missä a on joko syötesymboli tai Λ:

- Jos $(q_j, Y_1 \cdots Y_\ell) \in \delta(q_i, a, X)$, missä $\ell \geq 2$ ja $Y_1, \ldots, Y_\ell \in \Gamma$, niin vastaavat produktiot ovat
$$[q_i, X, p_\ell] \to a[q_j, Y_1, p_1][p_1, Y_2, p_2] \cdots [p_{\ell-1}, Y_\ell, p_\ell]$$
kaikille valinnoille $p_1, \ldots, p_\ell \in Q$. Huomaa, miten aina välisymboleissa vasemmanpuoleisen kolmas komponentti on sama kuin oikeanpuoleisen ensimmäinen komponentti. Monet näistä arvauksista ovat tietysti „huteja".

- Jos $(q_j, Y) \in \delta(q_i, a, X)$, missä $Y \in \Gamma$, niin vastaavat produktiot ovat
$$[q_i, X, p] \to a[q_j, Y, p]$$
kaikille valinnoille $p \in Q$.

- Jos $(q_j, \Lambda) \in \delta(q_i, a, X)$, niin vastaava produktio on
$$[q_i, X, q_j] \to a.$$
Pinon päällä oleva symboli X voidaan siis simuloitaessa pyyhkiä pois vain, jos arvattu seuraava tila q_j on oikea, muuten vasen johto pysähtyy.

- Otetaan aksiooma X_0, joka ei ole Δ:ssa, ja sille produktiot
$$X_0 \to [q_i, X, q_j],$$
missä $q_i \in S$, $q_j \in A$ ja $X \in Z$.

Jokaista M:n sanan w hyväksyvää tyhjään pinoon päättyvää konfiguraatioketjua vastaa CF-kieliopin[3]

$$G = \big(\Delta \cup \{X_0\}, \Sigma, X_0, P\big)$$

sanan w vasen johto, missä produktiot P saadaan yltä. Vastaavasti jokaista G:n sanan w johtoa vastaa M:n sanan w hyväksyvä konfiguraatioketju. □

Usein rajoitetaan pino-operaatioiden tyyppiä. Pino-operaatio eli tilasiirto pinon osalta on

- **pop**, jos se on muotoa $(q_j, \Lambda) \in \delta(q_i, a, X)$.
- **push**, jos se on muotoa $(q_j, YX) \in \delta(q_i, a, X)$, missä Y on pinosymboli.
- **unit**, jos se on muotoa $(q_j, Y) \in \delta(q_i, a, X)$, missä Y on pinosymboli.

[3]Jos M:ssä ei ole Λ-siirtoja, G on helppo muuntaa Greibachin normaalimuotoon.

Lause 17. *Jokainen PDA voidaan korvata saman kielen tunnistavalla PDA:lla, jossa on vain pino-operaatioita* **pop**, **push** *ja* **unit**.

Todistus. Pulmallisia ovat tilasiirrot

$$(q_j, Y_1 \cdots Y_\ell) \in \delta(q_i, a, X),$$

missä $Y_1, \ldots, Y_\ell \in \Gamma$ ja $\ell \geq 2$. Muut siirrot ovat joko tyyppiä **pop** tai tyyppiä **unit**. Tällaisia tilasiirtoja varten lisätään tilajoukkoon tietyt muotoa $\langle q_j, Y_1 \cdots Y_i \rangle$ olevat tilat ja määritellään niille tilasiirrot. Ensinnäkin poistetaan yo. tilasiirto ja korvataan se tyyppiä **unit** olevalla siirrolla

$$\big(\langle q_j, Y_1 \cdots Y_{\ell-1} \rangle, Y_\ell\big) \in \delta(q_i, a, X).$$

Edelleen lisätään tyyppiä **push** olevat tilasiirrot

$$\delta\big(\langle q_j, Y_1 \cdots Y_i \rangle, \Lambda, Y_{i+1}\big) = \big\{\big(\langle q_j, Y_1 \cdots Y_{i-1} \rangle, Y_i Y_{i+1}\big)\big\} \quad (i = 2, \ldots, \ell - 1)$$

sekä vielä

$$\delta\big(\langle q_j, Y_1 \rangle, \Lambda, Y_2\big) = \big\{(q_j, Y_1 Y_2)\big\}.$$

Yksi siirto korvautuu nyt usealla siirrolla, jotka ovat haluttua tyyppiä. □

Deterministiseltä pinoautomaatilta (*DPDA*) vaaditaan neljä ehtoa:

- Alkutilojen joukossa on yksi tila tai se on tyhjä.
- Pohjasymbolien joukossa on vain yksi symboli.
- $\delta(q_i, a, X)$:ssä on aina enintään yksi alkio, ts. aina on joko tarkalleen yksi mahdollinen tilasiirto tai sitten ei yhtään. Tässä a on joko syötesymboli tai Λ.
- Jos $\delta(q_i, \Lambda, X)$ ei ole tyhjä, niin $\delta(q_i, a, X)$ on aina tyhjä, kun $a \in \Sigma$, ts. jos on Λ-siirto, niin ei ole muita siirtoja.

Deterministiset pinoautomaatit eivät pysty tunnistamaan kaikkia CF-kieliä, niiden tunnistamia kieliä kutsutaankin *deterministisiksi CF-kieliksi* (*DCF-kieliksi*). Esimerkiksi palindromikieli L_{pal}, jossa aakkostossa on ainakin kaksi symbolia, on CF-kieli, muttei DCF-kieli. DCF-kielet ovat generoitavissa yksiselitteisillä CF-kieliopeilla, ks. Lauseen 16 todistus.

Ajateltuna ilman pinoaan PDA muistuttaa paljon transduktoria. Luettu symboli muodostuu parista, jossa on syötesymboli tai Λ ja pinosymboli, ja vastaava tuloste on pinon päällimmäisen symbolin korvaava sana. Transduktorit muodostavatkin tärkeän työkalun pidemmälle menevässä CF-kielten teoriassa. (On myös olemassa pinotransduktoreita.)

4.4 Jäsennysalgoritmit (lyhyt katsaus)

Lauseen 15 todistuksen PDA suorittaa oleellisesti simuloimansa kieliopin sanan *top–down-jäsennyksen.* Ts. se etsii produktiojonon, jolla generoitu sana tuotetaan. Valitettavasti vain PDA on luonteeltaan epädeterministinen ja jäsennysalgoritmi ei saa sitä olla. Oikeanlaisen jäsentäjän saamiseksi pitääkin epädeterministisyys jollakin tavalla poistaa. Sen sijaan, että vain hyväksyisi tai hylkäisi, pitää PDA:n tässä vielä hyväksyessään „tulostaa" jäsennykseen tarvittava tieto.

Epädeterministisyys voidaan toisinaan poistaa esimerkiksi *ennakoimalla* (englanniksi „look-ahead"), toisin sanoen lukien syötettä eteenpäin ennen jäsennysaskelta. CF-kielioppi on LL(k)-*kielioppi,* jos top–down-jäsennyksessä riittää ennakoida k symbolia jäsentävän PDA:n seuraavan askeleen selvittämiseksi. Formaalisti LL(k)-kielioppi[4] on CF-kielioppi, joka toteuttaa seuraavan ehdon, missä $(w)_k$ tarkoittaa sanan w k-pituista prefiksiä ja $\Rightarrow_{\text{left}}$ vasenta johtoaskelta: Jos

$$X_0 \Rightarrow^*_{\text{left}} uXv \Rightarrow_{\text{left}} uwv \Rightarrow^*_{\text{left}} uz \quad \text{ja}$$
$$X_0 \Rightarrow^*_{\text{left}} uXv' \Rightarrow_{\text{left}} uw'v' \Rightarrow^*_{\text{left}} uz'$$

ja

$$(z)_k = (z')_k,$$

niin

$$w = w'.$$

Ns. *bottom–up-jäsennyksessä* redusoidaan sanaa korvaamalla siinä osasanana oleva kieliopin produktion oikea puoli sen vasemman puolen välisymbolilla. Redusointia jatketaan keräten jäsennystieto kunnes saavutetaan aksiooma. Myös tämä jäsennystapa voidaan toteuttaa PDA:lla.

Nopeaa jäsennystä on tutkittu hyvin paljon. Hyvä viite on suomalaisten alan asiantuntijoiden kaksiosainen kirja Sippu, S. & Soisalon-Soininen, E.: *Parsing Theory. Volume I: Languages and Parsing.* Springer–Verlag (1988) ja *Volume II:* LR(k) *and* LL(k) *Parsing* (1990). Alan klassinen viite on „lohikäärmekirja" Aho, A.V. & Sethi, R. & Ullman, J.D.: *Compilers: Principles, Techniques, and Tools.* Addison–Wesley (1985), josta on tullut uusittu painos 2006 (uutena tekijänä mukana on Monica Lam).

4.5 Pumppaus

Säännöllisen äärettömän kielen sanoja voidaan pumpata yhdestä kohdasta. Muitakin kuin säännöllisiä CF-kieliä voidaan pumpata vain yhdestä kohdasta, mutta ei kaikkia: Yleisesti CF-kieltä voidaan pumpata vain kahdesta kohdasta yhtaikaa.

Pumppaus on helpointa ajatella, jos CF-kielioppi on vaikkapa Chomskyn normaalimuodossa. Tämähän ei rajoita tilannetta mitenkään, pumppaus on kielen ominaisuus, ei kieliopin. Chomskyn normaalimuotoisen kieliopin jäsennyspuu on *binääripuu,* ts. jokaisesta puun pisteestä lähtee enintään kaksi haaraa. Puun *korkeus* on sen juuresta lehteen kulkevan pisimmän polun pituus.

Apulause. *Jos binääripuussa on enemmän kuin* 2^h *lehteä, puun korkeus on ainakin* $h+1$.

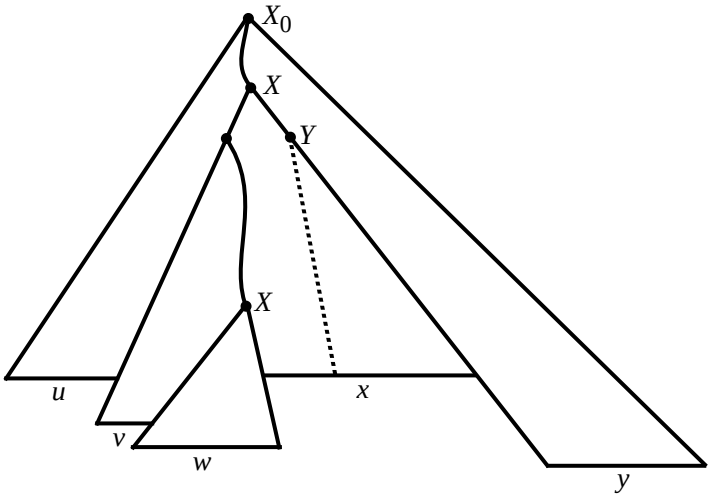
Todistus. Tulos pitää paikkansa, kun $h = 0$. Edetään induktiolla ja oletetaan, että tulos pitää paikkansa, kun $h \leq \ell$ ja katsotaan tapausta, missä $h = \ell + 1 \geq 1$. Kun puussa on ainakin kaksi lehteä, se voidaan jakaa ensimmäisen haarautumisen kautta kahteen binääripuuhun plus joihinkin edeltäviin pisteisiin (ainakin juuri on tällainen piste). Ainakin toisessa näistä puista on enemmän kuin $2^{\ell+1}/2 = 2^\ell$ lehteä ja sen korkeus on siis vähintään $\ell + 1$. Koko puun korkeus on näin ollen ainakin $\ell + 2$. □

Ja sitten pumppauksen perustulos:

[4]On myös olemassa vastaava oikeita johtoja käyttävä käsite, ns. LR(k)-*kielioppi.*

Pumppauslemma (*uvwxy*-lemma). *Jos CF-kieli L voidaan generoida Chomskyn normaalimuotoisella kieliopilla, jossa on p välisymbolia, $z \in L$ ja $|z| \geq 2^{p+1}$, niin z voidaan kirjoittaa muotoon $z = uvwxy$, missä $|vwx| \leq 2^{p+1}$, $vx \neq \Lambda$, $w \neq \Lambda$ ja sanat $uv^n wx^n y$ ovat kaikki L:ssä.*

Todistus. Sanan z jäsennyspuun korkeus on Apulauseen nojalla ainakin $p+1$. Otetaan tarkasteltavaksi pisin puun polku juuresta lehteen. Lehden lisäksi polussa on ainakin $p+1$ pistettä ja ne on merkitty välisymboleilla. Valitaan alimmaiset $p+1$ tällaista pistettä. Koska välisymboleja on p kpl, esiintyy jokin välisymboli X ainakin kahdesti valittujen pisteiden merkkinä. Valitaan näistä esiintymistä jotkin kaksi. Alemmasta X:n esiintymästä lähtevän alipuun lehdistä saadaan sana w ($\neq \Lambda$) ja ylemmästä vwx, ja edelleen z voidaan kirjoittaa muotoon $z = uvwxy$. Ks. skemaattinen kuva alla.



Ylemmästä X:n esiintymästä lähtevä alipuu voidaan tulkita vwx:n jäsennyspuuksi (binääripuu). Sen korkeus on ilmeisesti enintään $p+1$, joten Apulauseen nojalla siinä on enintään 2^{p+1} lehteä ja näin ollen $|vwx| \leq 2^{p+1}$. Ylemmällä X:n esiintymällä on kaksi jälkeläistä, toinen näistä on alemman X:n esiintymän esi-isä, toinen ei. Jälkimmäisen pisteen merkki on jokin välisymboli Y. Tästä pisteestä lähtevä alipuu on jonkin ei-tyhjän v:n tai x:n osasanan jäsennyspuu, riippuen siitä kummalla puolella X:n ylempää esiintymää Y on. Siispä $v \neq \Lambda$ tai/ja $x \neq \Lambda$.

Sanan z vasen johto on muotoa

$$X_0 \Rightarrow^* uXy' \Rightarrow^* uvXx'y' \Rightarrow^* uvwxy.$$

Silloin myös

$$X_0 \Rightarrow^* uXy' \Rightarrow^* uwy,$$
$$X_0 \Rightarrow^* uvXx'y' \Rightarrow^* uv^2Xx'^2y' \Rightarrow^* uv^2wx^2y$$

jne. ovat vasempia johtoja. □

Toisinaan myös pumppaus yhdessä paikassa on mahdollista, se vastaa tilannetta, jossa joko $v = \Lambda$ tai $x = \Lambda$.

Pumppausta käyttäen on helppo näyttää, että eräät kielet eivät ole CF-kieliä.

Esimerkki. *Kieli $L = \{a^{2^n} \mid n \geq 0\}$ on CS-kieli, kuten melko helposti on todettavissa (harjoituksena). Se ei kuitenkaan ole CF-kieli. Muutenhan se olisi generoitavissa Chomskyn normaalimuotoisella kieliopilla, jossa on p välisymbolia, ja kyllin pitkät osasanat olisivat näin ollen pumpattavissa. Tämä ei kuitenkaan ole mahdollista. Muutoin, jos $n = p+3$,*

niin voidaan kirjoittaa $2^{p+3} = m_1 + m_2$*, missä* $0 < m_2 \leq 2^{p+1}$*, ja myös sana* a^{m_1} *on kielessä* L *(valitaan* $m_2 = |vx|$*). Mutta*

$$m_1 \geq 2^{p+3} - 2^{p+1} > 2^{p+3} - 2^{p+2} = 2^{p+2}$$

eikä kielessä L *ole sanaa, jonka pituus olisi välillä* $2^{p+2} + 1, \ldots, 2^{p+3} - 1$*.*

Hieman vahvempi pumppaustulos, jonka todistus on Pumppauslemman todistuksen vähän terästetty versio, on

Ogdenin lemma. *Jos CF-kieli* L *voidaan generoida Chomskyn normaalimuotoisella kieliopilla, jossa on* p *välisymbolia,* $z \in L$ *ja* $|z| \geq 2^{p+1}$ *ja* z*:ssa on merkitty ainakin* 2^{p+1} *symbolia, niin* z *voidaan kirjoittaa muotoon* $z = uvwxy$*, missä* v*:ssä ja* x*:ssä on yhteensä ainakin yksi merkitty symboli,* vwx*:ssä on enintään* 2^{p+1} *merkittyä symbolia,* w*:ssä on ainakin yksi merkitty symboli, ja sanat* $uv^n wx^n y$ *ovat kaikki* L*:ssä.*

4.6 CF-kielten leikkaukset ja komplementit

CF-kielet eivät ole suljettuja leikkauksen suhteen. Esimerkiksi kielet

$$L_1 = \{a^i b^i a^j \mid i \geq 0 \text{ ja } j \geq 0\} \quad \text{ja} \quad L_2 = \{a^i b^j a^j \mid i \geq 0 \text{ ja } j \geq 0\}$$

ovat CF-kieliä, kuten on helposti todettavissa. Niiden leikkaus

$$L_1 \cap L_2 = \{a^i b^i a^i \mid i \geq 0\}$$

ei kuitenkaan ole CF-kieli, mikä voidaan helposti todeta Pumppauslemmalla. (Ko. leikkaus on kuitenkin CS-kieli.)

Joukko-opin sääntöjen mukaan

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}.$$

Koska CF-kielet ovat suljettuja yhdisteen suhteen, seuraa tästä, että *CF-kielet eivät ole suljettuja myöskään komplementin suhteen,* muutoinhan kielet $\overline{L_1}$ ja $\overline{L_2}$ olisivat CF-kieliä, ja samoin niiden yhdiste ja edelleen $L_1 \cap L_2$. Voidaan kuitenkin näyttää, että *DCF-kielet ovat suljettuja komplementin suhteen.*

Käyttäen hyvin samantyyppistä tilaparikonstruktiota kuin Lauseen 2 todistuksessa, voidaan todistaa

Lause 18. *CF-kielen ja säännöllisen kielen leikkaus on CF-kieli.*

Todistus. Tehdään CF-kielen L_1 tunnistavasta pinoautomaatista $M = (Q, \Sigma, \Gamma, S, Z, \delta, A)$ ja säännöllisen kielen L_2 tunnistavasta deterministisestä äärellisestä automaatista $M' = (Q', \Sigma, q_0', \delta', A')$ uusi PDA

$$M'' = (Q'', \Sigma, \Gamma, S'', Z, \delta'', A'').$$

Valitaan

$$\begin{aligned} Q'' &= \{(q_i, q_j') \mid q_i \in Q \text{ ja } q_j' \in Q'\}, \\ S'' &= \{(q_i, q_0') \mid q_i \in S\}, \\ A'' &= \{(q_i, q_j') \mid q_i \in A \text{ ja } q_j' \in A'\} \end{aligned}$$

ja määritellään δ'' säännöillä

- Jos $(q_j, \alpha) \in \delta(q_i, a, X)$ ja $\delta'(q'_k, a) = q'_\ell$, niin

$$\big((q_j, q'_\ell), \alpha\big) \in \delta''\big((q_i, q'_k), a, X\big),$$

  ts. syötesymbolin a lukeminen aiheuttaa kummassakin automaatissa oikean tilasiirron.

- Jos $(q_j, \alpha) \in \delta(q_i, \Lambda, X)$, niin

$$\big((q_j, q'_k), \alpha\big) \in \delta''\big((q_i, q'_k), \Lambda, X\big),$$

  ts. Λ-siirrossa vain PDA:ssa tapahtuu tilasiirto.

Tällöin M'' tunnistaa leikkauskielen $L_1 \cap L_2$. □

4.7 Ratkeavuuskysymyksiä. Postin vastaavuusprobleema

Säännöllisille kielille kutakuinkin kaikki karakterisaatioprobleemat ovat algoritmisesti ratkeavia. CF-kielille ei näin enää ole. Katsotaan ensin eräitä probleemoja, jotka ovat algoritmisesti ratkeavia.

- *Jäsenyysprobleema:* Onko sana w kielessä L?

  Generoidaan kieli L Chomskyn normaalimuotoisella kieliopilla. Silloin helppo tarkistaa, onko Λ kielessä L. Edelleen mahdollisia w:n vasempia johtoja on äärellinen määrä, sillä joka askeleella joko tulee uusi loppusymboli tai pituus kasvaa yhdellä. Tarkistetaan nämä mahdolliset johdot. Jäsenyys ja jäsennys ovat luonnollisesti tekemisissä keskenään, jos ajatellaan vain jonkin jäsennyspuun löytämistä tai sen toteamista, ettei jäsennystä ole. Tähän tarkoitukseen tunnetaan useita melko nopeita menetelmiä, mm. ns. *Earleyn algoritmi.*

- *Tyhjyysprobleema:* Onko L tyhjä kieli $\emptyset$?

  Pumppauslemman avulla on varsin helppo nähdä, että jos L ei ole tyhjä, niin siinä on sana, jonka pituus on enintään $2^{p+1} - 1$.

- *Äärellisyysprobleema:* Onko L äärellinen kieli?

  Jos CF-kieli on ääretön, sitä voi pumpata. Pumppauslemman avulla on silloin helppo todeta, että siinä on näin ollen sana, jonka pituus on välillä $2^{p+1}, \ldots, 2^{p+2} - 1$.

- *DCF-ekvivalenssiprobleema:* Ovatko DCF-kielet L_1 ja L_2 samat?

  Tämä probleema oli pitkään kuuluisa avoin probleema. Sen ratkaisi vasta melko äskettäin ranskalainen Géraud Sénizergues. Ratkaisu on muuten erittäin mutkikas.[5]

[5]Se selostetaan pitkässä artikkelissa SÉNIZERGUES, G.: $L(A) = L(B)$? Decidability Results from Complete Formal Systems. *Theoretical Computer Science* **251** (2001), 1–166. Huomattavasti lyhyempi ratkaisu on artikkelissa STIRLING, C.: Decidability of DPDA Equivalence. *Theoretical Computer Science* **255** (2001), 1–31.

Monet probleemat puolestaan ovat algoritmisesti ratkeamattomia. Useimmat näistä voidaan melko helposti palauttaa erään tietyn probleeman, ns. *Postin vastaavuusprobleeman* (*PCP*), algoritmiseen ratkeamattomuuteen. Jotta yleensä ottaen päästään käsittelemään algoritmista ratkeamattomuutta, pitää määritellä algoritmin käsite. Kutakuinkin täysin hyväksytty määrittely käyttää ns. *Turingin konetta,* jolla taas on läheinen yhteys Tyypin 0 kielioppeihin.[6] Asiaan palataan myöhemmin.

Postin vastaavuusprobleeman syötteenä on kaksi aakkostoa $\Sigma = \{a_1, \ldots, a_n\}$ ja Δ ja kaksi morfismia $\sigma_1, \sigma_2 : \Sigma^* \to \Delta^*$, ks. Pykälä 2.8. Nämä morfismit annetaan luettelemalla Σ:n symbolien kuvat (aakkoston Δ ei-tyhjät sanat[7]):

$$\sigma_1(a_i) = \alpha_i \quad , \quad \sigma_2(a_i) = \beta_i \quad (i = 1, \ldots, n).$$

Tehtävänä on selvittää onko sellaista sanaa $w \in \Sigma^+$, että $\sigma_1(w) = \sigma_2(w)$. Tuloste on vastaus „kyllä" tai „ei". Itse sanaa w kutsutaan probleeman *ratkaisuksi.*

Liittyen annettuun Postin vastaavuusprobleemaan määritellään jatkoa ajatellen aakkoston $\Sigma \cup \Delta \cup \{\#\}$ kielet

$$L_1 = \left\{\sigma_1(w)\#\hat{w} \;\middle|\; w \in \Sigma^+\right\} \quad \text{ja} \quad L_2 = \left\{\sigma_2(w)\#\hat{w} \;\middle|\; w \in \Sigma^+\right\},$$

missä # on uusi symboli. Ei ole kovin vaikea todeta, että nämä kielet samoin kuin niiden komplementit $\overline{L_1}$ sekä $\overline{L_2}$ ovat CF-kieliä, jolloin edelleen kielet

$$L_3 = L_1 \cup L_2 \quad \text{ja} \quad L_4 = \overline{L_1} \cup \overline{L_2}$$

ovat CF-kielten yhdisteinä CF-kieliä. Ilmeisesti annetulla Postin vastaavuusprobleemalla ei ole ratkaisua tarkalleen silloin, kun leikkaus $L_1 \cap L_2$ on tyhjä, eli tarkalleen silloin, kun L_4 muodostuu kaikista aakkoston $\Sigma \cup \Delta \cup \{\#\}$ sanoista. Viime mainittu seuraa, koska joukko-opillisesti $L_4 = \overline{L_1 \cap L_2}$.

Vielä voidaan todeta, että L_4 on säännöllinen tarkalleen silloin, kun $L_1 \cap L_2$ on säännöllinen, ja tämä taas on tosi tarkalleen silloin, kun $L_1 \cap L_2 = \emptyset$. Jos nimittäin $L_1 \cap L_2$ on säännöllinen ja siinä on sana

$$\sigma_1(w)\#\hat{w} = \sigma_2(w)\#\hat{w},$$

niin $w \neq \Lambda$ ja $\sigma_1(w) \neq \Lambda$ ja edelleen myös sanat

$$\sigma_1(w^n)\#\widehat{w^n} = \sigma_2(w^n)\#\widehat{w^n} \quad (n = 2, 3, \ldots)$$

ovat $L_1 \cap L_2$:ssa. Kyllin suurelle n:lle säännöllisten kielten Pumppauslemma on silloin sovellettavissa ja tuottaa selvästikin sanoja, jotka eivät ole kielissä L_1 ja L_2.

Seuraavat probleemat ovat näin edellä olevan mukaan algoritmisesti ratkeamattomia:

- *Leikkauksen tyhjyysprobleema:* Onko kahden annetun CF-kielen leikkaus tyhjä?
- *Universaalisuusprobleema:* Onko annetussa CF-kielessä kaikki aakkoston sanat?
- *Ekvivalenssiprobleema:* Ovatko annetut CF-kielet samat?

  Palautuu Universaalisuusprobleemaan, sillä kaikkien sanojen muodostama kieli on tietysti CF-kieli.

[6] Jos olisi algoritmi PCP:n ratkaisemiseksi, olisi myös algoritmi Turingin koneen pysähtymisprobleeman ratkaisemiseksi, ks. Lause 30. Tämä yhteys on varsin vaikea todistaa, ks. esimerkiksi MARTIN.

[7] Toisin kuin tässä, usein sallitaan myös tyhjä sana kuvana. Tällä ei ole suurempaa merkitystä.

- *Säännöllisyysprobleema* Onko annettu CF-kieli säännöllinen?

Pienellä lisäkonstruktiolla nähdään algoritmisesti ratkeamattomaksi myös

- *Yksiselitteisyysprobleema:* Onko annettu CF-kielioppi yksiselitteinen?

  Otetaan tarkasteltavaksi kielen L_3 generoiva CF-kielioppi

  $$G = \big(\{X_0, X_1, X_2\}, \Sigma \cup \Delta \cup \{\#\}, X_0, P\big),$$

  missä P:ssä ovat produktiot $X_0 \to X_1 \mid X_2$ sekä

  $$X_1 \to \alpha_i X_1 a_i \mid \alpha_i \# a_i \quad \text{ja} \quad X_2 \to \beta_i X_1 a_i \mid \beta_i \# a_i \quad (i = 1, \ldots, n).$$

  Jos nyt $\sigma_1(w) = \sigma_2(w)$ jollekin sanalle $w \in \Sigma^+$, niin L_3:n sanalle

  $$\sigma_1(w)\#\hat{w} = \sigma_2(w)\#\hat{w}$$

  on G:ssä kaksi erilaista vasenta johtoa. Jos toisaalta jollekin L_3:n sanalle $v\#\hat{w}$ on kaksi vasenta erilaista johtoa, niin yksi niistä alkaa $X_0 \Rightarrow_G X_1$:llä ja toinen puolestaan $X_0 \Rightarrow_G X_2$:lla ja $v = \sigma_1(w) = \sigma_2(w)$.

Myös CF-kielten luontainen moniselitteisyys on algoritmisesti ratkeamaton. Edelleen, sen lisäksi että CF-kielet eivät ole suljettuja leikkauksen eikä komplementin suhteen, mahdollinen sulkeutuvuuskaan ei ole algoritmista. Onko CF-kielten leikkaus CF-kieli tai onko CF-kielen komplementti CF-kieli ovat nimittäin nekin algoritmisesti ratkeamattomia kysymyksiä. Näiden todistukset ovat jo hankalampia, ks. esimerkiksi HOPCROFT & ULLMAN.

Luku 5

CS-KIELET

5.1 Lineaarisesti rajoitettu automaatti

Pinoautomaatin pinomuistin luku ja kirjoitus tapahtuu pinon yläpäässä, muualle pinoon ei ole pääsyä. Pinon korkeus suhteessa luetun sanan pituuteen voi periaatteessa olla miten tahansa suuri Λ-siirroista johtuen. Λ-siirrot voidaan kuitenkin kokonaan poistaa.[1] Näin ollen pinon korkeuden voitaisiin haluttaessa olettaa pysyvän verrannollisena syötteen pituuteen.

Kun muutetaan pinoautomaatin ideaa siten, että „kaikkialla pinossa" voidaan lukea ja kirjoittaa eli vaihtaa symboli toiseen ja myös syöte on kaikkialta koko ajan luettavissa, saadaan oleellisesti automaatti, jota kutsutaan lineaarisesti rajoitetuksi automaatiksi (LBA). Muistitilan pituus ei tässä saa olla suurempi kuin syötteen pituus. Koska muistissa voi käyttää symboleja enemmänkin kuin vain syötesymbolit, voi muistin kulloinkin tallentama informaatiomäärä olla suurempi kuin syötteen, mutta se on kuitenkin verrannollinen syötteen informaatiomäärään. Ts. on sellainen kerroin C, että

$$\text{muistin informaatio} \leq C \times \text{syötteen informaatio.}$$

Tästä tulee nimi „lineaarisesti rajoitettu".

LBA määritellään kuitenkin yleensä hieman toisin, käyttäen Turingin koneen tapaista formalismia.

Määritelmä. Lineaarisesti rajoitettu automaatti *(*LBA*) on kahdeksikko*

$$M = (Q, \Sigma, \Gamma, S, X_{\mathrm{L}}, X_{\mathrm{R}}, \delta, A),$$

missä

- $Q = \{q_1, \ldots, q_m\}$ *on äärellinen* tilojen joukko, *jonka alkioita kutsutaan* tiloiksi;
- Σ *on* syöteaakkosto *(kielen aakkosto);*
- Γ *on* nauha-aakkosto ($\Sigma \subseteq \Gamma$);
- $S \subseteq Q$ *on* alkutilojen joukko;
- $X_{\mathrm{L}} \in \Gamma - \Sigma$ *on* vasen reunamerkki;

[1] Ks. Lauseiden 15 ja 16 todistusten alaviitteet. Oleellisesti tämä vastaa kieliopin muuttamista Greibachin normaalimuotoon.

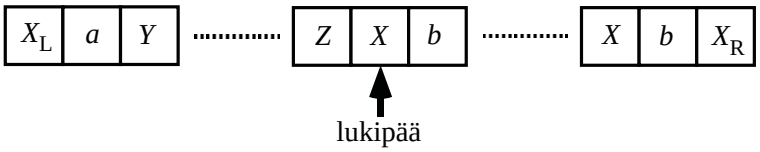
- $X_{\mathrm{R}} \in \Gamma - \Sigma$ *on* oikea reunamerkki ($X_{\mathrm{R}} \neq X_{\mathrm{L}}$);
- δ *on* siirtofunktio, *joka kuvaa jokaisen parin* (q_i, X)*, jossa* q_i *on tila ja* X *on nauhasymboli, tarkalleen yhdeksi äärelliseksi kolmikoiden* (q_j, Y, s) *joukoksi* $\delta(q_i, X)$*, missä* q_j *on tila,* Y *on nauhasymboli ja* s *on jokin luvuista* 0, $+1$ *tai* -1*; lisäksi oletetaan, että*
  - *jos* $q_i \in A$*, niin* $\delta(q_i, X) = \emptyset$*,*
  - *jos* $(q_j, Y, s) \in \delta(q_i, X_{\mathrm{L}})$*, niin* $Y = X_{\mathrm{L}}$ *ja* $s \neq -1$*, ja*
  - *jos* $(q_j, Y, s) \in \delta(q_i, X_{\mathrm{R}})$*, niin* $Y = X_{\mathrm{R}}$ *ja* $s \neq +1$*;*
- $A \subseteq Q$ *on* lopputilojen joukko.

LBA:n muistitila, ns. *nauha,* on aina muotoa

$$X_{\mathrm{L}} \alpha X_{\mathrm{R}}$$

oleva sana, missä $\alpha \in \Gamma^*$. Alussa nauha on $X_{\mathrm{L}} w X_{\mathrm{R}}$, missä w on syötesana. Edelleen $|\alpha| = |w|$ eli nauhan pituus on koko ajan $|w| + 2$.

LBA on aina tarkalleen yhdessä tilassa q_i ja *lukee* tarkalleen yhtä symbolia X nauhansa symboleista. Alussa LBA lukee vasenta reunasymbolia X_{L}. Havainnollisesti voi ajatella, että LBA:lla on „lukipää”, joka lukee nauhaa ja voi myös kirjoittaa siihen:



Siirtofunktio δ ilmoittaa, mitä LBA seuraavaksi tekee. Jos

$$(q_j, Y, s) \in \delta(q_i, X)$$

ja luettu symboli on nauhan ℓ:s symboli, niin LBA vaihtaa tilan q_i:stä q_j:hin, korvaa lukemansa nauhasymbolin X symbolilla Y (joka Y voi olla X) ja siirtyy lukemaan nauhan $\ell + s$:ttä symbolia. Tällainen operaatio on yksi LBA:n *siirto*. Huomaa, että LBA on luonteeltaan epädeterministinen: kussakin tilanteessa siirtoja voi olla valittavissa useampia tai ei yhtään.

Kuten PDA:lle, muistin tilanne annetaan konfiguraatiolla. *Konfiguraatio* on nelikkö (q_i, α, X, β), missä q_i on tila, $\alpha X \beta$ on nauha ja X on luettu nauhan symboli. Edellä olevaan verraten siis $\ell = |\alpha X|$. Konfiguraatio $(q_j, \alpha', Y, \beta')$ on konfiguraation (q_i, α, X, β) *suora seuraaja,* jos se saadaan yhdellä LBA:n siirrolla (q_i, α, X, β):sta, merkitään

$$(q_i, \alpha, X, \beta) \vdash_M (q_j, \alpha', Y, \beta').$$

Siirtoa $(q_j, Y, -1) \in \delta(q_i, X)$ vastaa siis esimerkiksi

$$(q_i, \alpha Z, X, \beta) \vdash_M (q_j, \alpha, Z, Y\beta),$$

missä $Z \in \Gamma$. „Tähtirelaatio” $\vdash_M^*$ määritellään aivan samoin kuin PDA:lle.

Alussa nauhalla on syötesana w ja vastaava konfiguraatio on $(q_i, \Lambda, X_{\mathrm{L}}, w X_{\mathrm{R}})$, missä q_i on jokin alkutila. LBA M *hyväksyy* syötteen w, jos

$$(q_i, \Lambda, X_{\mathrm{L}}, w X_{\mathrm{R}}) \vdash_M^* (q_j, \alpha, Y, \beta),$$

missä q_i on jokin alkutila ja q_j on jokin lopputila. Huomaa, että lopputilasta eteenpäin ei ole siirtoja, ts. LBA *pysähtyy.* LBA voi myös pysähtyä muuhunkin kuin lopputilaan, tällöin syötettä ei hyväksytä. LBA:n hyväksymien sanojen muodostama kieli on sen *tunnistama* kieli.

Koska yksinkertaisenkin LBA:n täydellinen määrittely on yleensä pitkä, on tapana esittää vain LBA:n toimintaidea, kuitenkin niin yksityiskohtaisesti, ettei sen toiminnan oikeellisuudesta jää epäilyksiä.[2]

Esimerkki. *Kielen* $L = \{a^{2^n} \mid n \geq 0\}$ *tunnistaa LBA*

$$M = \big(Q, \{a\}, \Gamma, \{q_0\}, @, \#, \delta, \{q_1\}\big),$$

jonka toiminta voidaan kuvata lyhyesti seuraavasti. Paitsi symboli a*,* Γ*:ssa on myös symboli* a'*. Ensimmäisenä toimenaan* M *vaihtaa vasemman* a*:n* a'*:ksi. Sen jälkeen* M *kaksinkertaistaa* a'*-symbolien lukumäärän käyttäen aikaisempia* a'*:ja laskurina. Jos* M *törmää kesken tällaisen kaksinkertaistamisprosessin oikeaan reunamerkkiin* $\#$*, se pysähtyy, tila ei tällöin ole lopputila* q_1*. Jos se taas törmää* $\#$*:oon saatuaan juuri valmiiksi kaksinkertaistamisen, se siirtyy lopputilaan* q_1*. Selvästi tässä tarvitaan vielä muitakin nauhasymboleja ja tiloja, jotta tällainen toiminto voidaan toteuttaa.*

Lause 19. *Jokainen CS-kieli voidaan tunnistaa LBA:lla.*

Todistus. Otetaan CS-kielioppi $G = (\Sigma_\mathrm{N}, \Sigma_\mathrm{T}, X_0, P)$ ja näytetään miten sen generoima kieli voidaan tunnistaa LBA:lla

$$M = (Q, \Sigma_\mathrm{T}, \Gamma, S, X_\mathrm{L}, X_\mathrm{R}, \delta, A).$$

Paitsi G:n loppusymboleja, Γ:ssa ovat myös symbolit

$$[x, a] \quad (x \in \Sigma_\mathrm{N} \cup \Sigma_\mathrm{T} \text{ ja } a \in \Sigma_\mathrm{T})$$

sekä vielä symbolit

$$[\Lambda, a] \quad (a \in \Sigma_\mathrm{T}).$$

Aluksi M vaihtaa syötteen kunkin symbolin a symboliksi $[\Lambda, a]$, paitsi syötteen vasemman symbolin b, jonka se vaihtaa symboliksi $[X_0, b]$. M tallettaa näin syötteen nauhasymbolien toisiin komponentteihin ja käyttää ensimmäisiä komponentteja G:n johdon simulointiin. Yksi simulointiaskel muodostuu seuraavista osista.

1. M käy läpi vasemmalta oikealle nauhan ja etsii äärellistä muistiaan käyttäen osasanaa, joka olisi jonkin G:n produktion vasen puoli. Tätä varten M pitää koko ajan äärellisessä muistissaan kyllin monta viimeksi lukemaansa symbolia.

2. Löydettyään jonkin produktion vasemman puolen M päättää käyttääkö tätä produktiota simuloinnissa vai ei (epädeterministisyys).

3. Jos M päättää käyttää simuloinnissa löytämäänsä sopivaa produktiota, se suorittaa G:n vastaavan johtoaskeleen. Jos tällöin sana pitenee, ts. produktion oikea puoli on vasenta pidempi, pitää jo johdetun sanan loppuosaa siirtää vastaavasti oikealle. Tämä vaatii useita askeleita. Mikäli tämä ei onnistu, menee johdettu sana liian pitkäksi ja M pysähtyy, mutta ei lopputilaan.

[2] „Designing Turing machines by writing out a complete set of states and a next-move function is a noticeably unrewarding task." (Hopcroft & Ullman).

4. Ellei johtoa voida enää jatkaa eli ei löydy produktioita, M tarkistaa, onko johdettu sana sama kuin syöte, joka oli tallennettu alempiin komponentteihin, ja siirtyy positiivisessa tapauksessa lopputilaan. Muutoin M pysähtyy ei-lopputilaan.

Selvästi tarvitaan jo mainittujen lisäksi paljon muitakin nauhasymboleja paikkamerkkeinä ja muuten, ja myös paljon tiloja. □

Lause 20. *LBA:n tunnistama kieli on CS-kieli.*

Todistus. Otetaan tarkasteltavaksi LBA

$$M = (Q, \Sigma, \Gamma, S, \$, \#, \delta, A).$$

$L(M)$:n generoiva pituutta kasvattava kielioppi $G = (\Sigma_{\mathrm{N}}, \Sigma, X_0, P)$ saadaan seuraavasti (ks. Lause 11). Ensinnäkin tarvitaan välisymbolit

$$[X, a] \quad (X \in \Gamma \text{ ja } a \in \Sigma)$$

ja sanan päissä

$$[\$, X, a] \quad \text{sekä} \quad [\#, X, a] \quad (X \in \Gamma \text{ ja } a \in \Sigma).$$

Tilasiirron simuloimista varten tarvitaan vielä välisymbolit

$$[X, q_i, a] \quad \text{sekä} \quad [X, q_i, s, a] \quad (X \in \Gamma, q_i \in Q, a \in \Sigma \text{ ja } s = 0, \pm 1)$$

ja sanan päissä

$$\begin{aligned} [\$, q_i, X, a] \quad &\text{sekä} \quad [\#, q_i, X, a], \\ [\$, X, q_i, a] \quad &\text{sekä} \quad [\#, X, q_i, a], \\ [\$, X, q_i, s, a] \quad &\text{sekä} \quad [\#, X, q_i, s, a] \quad (X \in \Gamma, q_i \in Q, a \in \Sigma \text{ ja } s = 0, \pm 1). \end{aligned}$$

Lisäksi tarvitaan vielä yksi välisymboli Y.

Ensin G generoi mielivaltaisen vähintään 3-pituisen syötesanan, joka talletetaan välisymbolien viimeiseen komponenttiin. (Lyhyemmät sanat hoidetaan erikseen.) Tähän tarvitaan produktiot

$$\begin{aligned} X_0 &\to Y[\#, a, a] \quad (a \in \Sigma), \\ Y &\to Y[a, a] \mid [\$, q_i, a, a][b, b] \quad (a, b \in \Sigma \text{ ja } q_i \in S). \end{aligned}$$

Sen jälkeen G simuloi M:ää välisymbolien ensimmäisissä komponenteissa. Huomaa, että reunamerkit on otettava syötteen päissä olevien symbolien sisään, koska niitä ei voida pyyhkiä pituutta kasvattavalla kieliopilla pois.

Siirtoa

$$(q_j, V, 0) \in \delta(q_i, U),$$

jossa lukipää ei liiku, simuloidaan produktioilla

$$\begin{aligned} [U, q_i, a] &\to [V, q_j, a], \\ [\$, q_i, X, a] &\to [\$, q_j, X, a] \quad (U = V = \$), \\ [\$, U, q_i, a] &\to [\$, V, q_j, a], \\ [\#, q_i, X, a] &\to [\#, q_j, X, a] \quad (U = V = \#), \\ [\#, U, q_i, a] &\to [\#, V, q_j, a]. \end{aligned}$$

Siirtoa

$$(q_j, V, +1) \in \delta(q_i, U),$$

jossa lukipää siirtyy oikealle, simuloidaan produktioilla

$$\begin{aligned}[U, q_i, a] &\to [V, q_j, +1, a],\\ [\$, U, q_i, a] &\to [\$, V, q_j, +1, a],\end{aligned}$$

joilla „ilmoitetaan", että lukipään siirto oikealle on tulossa, ja produktioilla

$$\begin{aligned}[V, q_j, +1, a][X, b] &\to [V, a][X, q_j, b],\\ [V, q_j, +1, a][\#, X, b] &\to [V, a][\#, X, q_j, b],\\ [\#, U, q_i, a] &\to [\#, q_j, V, a],\\ [\$, q_i, X, a] &\to [\$, X, q_j, a] \quad (U = V = \$),\\ [\$, V, q_j, +1, a][X, b] &\to [\$, V, a][X, q_j, b],\end{aligned}$$

jotka toteuttavat siirtymisen. Siirtoa, jossa lukipää siirtyy vasemmalle, simuloidaan vastaavanlaisilla produktioilla.

Vielä tarvitaan loppuproduktiot

$$\begin{aligned}[X, q_i, a] &\to a\ ,\\ [\$, q_i, X, a] &\to a\ , \qquad [\$, X, q_i, a] \to a\ ,\\ [\#, q_i, X, a] &\to a\ , \qquad [\#, X, q_i, a] \to a\ ,\end{aligned}$$

missä $q_i \in A$, ja

$$[X, a] \to a\ , \quad [\$, X, a] \to a\ , \quad [\#, X, a] \to a.$$

Kielen $L(M)$ sana w, jonka pituus on ≤ 2, otetaan mukaan produktiolla $X_0 \to w$. □

Lauseiden 19 ja 20 seurauksena *CS-kielet ovat tarkalleen kaikki lineaarisesti rajoitettujen automaattien tunnistamat kielet.*

LBA on *deterministinen,* jos $\delta(q_i, X)$:ssä on aina enintään yksi alkio, ts. joko siirtoa ei ole tai sitten vaihtoehtoja on vain yksi, ja alkutiloja on enintään yksi. Deterministisen LBA:iden tunnistamia CS-kieliä sanotaan *deterministisiksi CS-kieliksi* eli *DCS-kieliksi.* On varsin tunnettu avoin probleema, ovatko kaikki CS-kielet deterministisiä.

5.2 Normaalimuotoja

Pituutta kasvattava kielioppi on ns. *Kurodan normaalimuodossa,* jos sen produktiot ovat muotoa

$$X \to a\ , \quad X \to Y\ , \quad X \to YZ \quad \text{tai} \quad XY \to UV,$$

missä a on loppusymboli ja X, Y, Z, U ja V ovat välisymboleja. Poikkeuksena taas produktio $X_0 \to \Lambda$, mikäli aksiooma X_0 ei esiinny minkään produktion oikealla puolella.

Lause 21. *Jokainen CS-kieli voidaan generoida Kurodan normaalimuotoisella pituutta kasvattavalla kieliopilla.*

Todistus. Tunnistetaan ensin CS-kieli LBA:lla ja sen jälkeen muunnetaan LBA pituutta kasvattavaksi kieliopiksi Lauseen 20 todistuksen menetelmällä. Tuloksena on Kurodan normaalimuotoinen kielioppi, kuten helposti voi todeta. □

CS-kielioppi on ns. *Penttosen normaalimuodossa,*[3] jos sen produktiot ovat muotoa

$$X \to a\ , \quad X \to YZ \quad \text{tai} \quad XY \to XZ,$$

missä a on loppusymboli ja X, Y ja Z ovat välisymboleja. Ja poikkeuksena tietysti produktio $X_0 \to \Lambda$, mikäli aksiooma X_0 ei esiinny minkään produktion oikealla puolella. Jokainen CS-kielioppi voidaan viedä Penttosen normaalimuotoon, mutta tämä onkin jo varsin vaikea todistaa.

5.3 CS-kielten ominaisuuksia

CS-kielet ovat laskettavia, ts. niiden jäsenyysprobleema on algoritmisesti ratkaistavissa. Tämä on helposti nähtävissä. Sen selvittämiseksi onko sana w LBA:n M tunnistamassa kielessä vai ei, katsotaan vain M:n syötteellä w ottamia askeleita, kunnes joko M pysähtyy tai jokin konfiguraatio toistuu. Tämä tehdään kaikille siirtovaihtoehdoille (epädeterministisyys). Toisaalta

Lause 22. *On olemassa aakkoston $\{a\}$ laskettava kieli, joka ei ole CS-kieli.*

Todistus. Numeroidaan kaikki mahdolliset CS-kieliopit, joiden loppusymboliaakkosto on $\{a\}$: $G_1, G_2, \ldots$ Tämä voidaan tehdä vaikkapa seuraavasti. Korvataan i:s välisymboli kieliopissa jokaisessa esiintymässään $\#b_i$:llä, missä b_i on i:n binääriesitys. Sen jälkeen käytössä ovatkin vain symbolit

$$a\ 0\ 1\ ,\ \#\ (\ )\ \{\ \}\ \to$$

ja kieliopit voidaan järjestää leksikografiseen järjestykseen, eli ensin pituusjärjestykseen ja saman pituuden sisällä aakkosjärjestykseen.

Otetaan sitten aakkoston $\{a\}$ kieli

$$L = \left\{ a^n \;\middle|\; a^n \notin L(G_n) \right\}.$$

L on laskettava, sillä sanan a^m kuuluminen L:ään voidaan selvittää seuraavalla menettelyllä: (1) Etsitään kielioppi G_m. (2) Koska G_m:n jäsenyysprobleema on algoritmisesti ratkeava, jatketaan tutkimalla onko a^m kielessä $L(G_m)$ vai ei. Toisaalta L ei ole CS. Jos se nimittäin olisi, se olisi jokin kielistä $L(G_1), L(G_2), \ldots$, sanotaan $L = L(G_k)$. Mutta silloinhan sana a^k on L:ssä tarkalleen silloin, kun se ei ole L:ssä! □

Edellinen todistus on taas esimerkki lävistäjämenetelmästä, vrt. Lauseen 1 todistus. Itse asiassa on vaikea löytää „luontevia" esimerkkejä kielistä, jotka eivät ole CS-kieliä, käyttämättä lävistäjämenetelmää tai ottamatta käyttöön laskennallisen vaativuuden tuloksia. Ohjelmointikielissä esimerkiksi on usein piirteitä, jotka johtavat siihen, etteivät ne tarkkaan ottaen olekaan CF-kieliä. Kuitenkin ne ovat kutakuinkin poikkeuksetta tällöinkin CS-kieliä.

CS-kielten todettiin edellä Lauseessa 10 olevan suljettuja yhdisteen, katenaation ja katenaatiosulkeuman sekä myös peilikuvan suhteen. Toisin kuin CF-kielet, CS-kielet ovat suljettuja myös leikkauksen ja komplementin suhteen.

Lause 23. *$\mathcal{CS}$ on suljettu leikkauksen suhteen.*

[3] Tämä on ns. vasemmanpuoleinen normaalimuoto. On tietysti myös vastaava oikeanpuoleinen normaalimuoto. Alkuperäisviite on PENTTONEN, M.: One-Sided and Two-Sided Context in Formal Grammars. *Information and Control* **25** (1974), 371–392.

Todistus. Jos LBA:t M_1 ja M_2 tunnistavat CS-kielet L_1 ja L_2, vastaavasti, on helppo konstruoida kolmas LBA M, joka tunnistaa leikkauksen $L_1 \cap L_2$. Tällöin M simuloi sekä M_1:tä että M_2:ta yhtaikaa vuorotellen tallentaen nauhalleen näiden nauhat, lukipäiden paikat ja tilat. M hyväksyy syötteen, mikäli simuloinnissa sekä M_1 että M_2 hyväksyvät sen. □

$\mathcal{CS}$:n sulkeutuvuus komplementin suhteen oli eräs 80-luvun kuuluisimpia diskreetin matematiikan tuloksia. Sen todistivat toisistaan riippumatta samaan aikaan Neil Immerman ja Róbert Szelepcsényi.[4] Itse asiassa he todistivat paljon yleisemmän tilavaativuusluokkia koskevan tuloksen.

Immerman–Szelepcsényi-lause. *$\mathcal{CS}$ on suljettu komplementin suhteen.*

Todistus. Lähdetään liikkeelle LBA:sta M ja konstruoidaan toinen LBA M_{C}, joka tunnistaa $\overline{L(M)}$:n.

Ensimmäisenä toimenaan M_{C} laskee niiden M:n konfiguraatioiden lukumäärän N, jotka ovat jonkin alkukonfiguraation seuraajia syötteelle w. Tällaisten konfiguraatioiden pituus on $|w| = n$. Merkitään N_j:llä niiden konfiguraatioiden lukumäärää, jotka saadaan syötettä w vastaavista alkukonfiguraatioista enintään j askeleella. Kun löytyy ensimmäinen sellainen j:n arvo $j = j_{\max}$, että $N_{j_{\max}} = N_{j_{\max}+1}$, on ilmeisesti $N = N_{j_{\max}}$. Alussa N_0 on alkutilojen lukumäärä. Ei ole kovin vaikea nähdä, että luku N voidaan tallentaa nauhalle käyttäen vaikkapa desimaaliesitystä, jossa useampi desimaali on ahdettu yhteen nauhasymboliin. Samalla tekniikalla voidaan tallentaa useitakin enintään samaa suuruusluokkaa olevia lukuja.

Luvun N_j saatuaan M_{C} laskee seuraavan luvun N_{j+1}. M_{C} pitää muistissaan vain luvun N_j:n, ei sitä edeltäviä lukuja. M_{C} käy läpi kaikki n-pituiset M:n konfiguraatiot aakkosjärjestyksessä ja pitää muistissaan tätä varten aina vain konfiguraation ja tarvittaessa järjestyksessä sitä edeltävän konfiguraation, ei muita. Tutkittuaan konfiguraation κ_ℓ loppuun, M_{C} etsii tätä käyttäen järjestyksessä seuraavan konfiguraation $\kappa = \kappa_{\ell+1}$. Tarkoitus on tutkia, voisiko konfiguraation κ saada seuraajana jostain alkukonfiguraatiosta syötteelle w enintään $j + 1$ askeleella.

M_{C} pitää nauhalla yllä kahta laskuria λ_1 ja λ_2. Aina kun löytyy konfiguraatio κ, joka saadaan seuraajana jostain alkukonfiguraatiosta syötteelle w enintään $j + 1$ askeleella, lisätään λ_1:n arvoa 1:llä.

Tutkiakseen konfiguraatiota κ, M_{C} käy läpi n-pituisia konfiguraatioita κ' löytääkseen sellaisia, jotka ovat saatavissa jonkin w:n alkukonfiguraation seuraajina enintään j askeleella. Löydettyjen tällaisten konfiguraatioiden lukumäärää M_{C} ylläpitää laskurissa λ_2.

Kunkin konfiguraation κ' kohdalla M_{C} ensin arvaa onko ko. konfiguraatio niiden N_j konfiguraation joukossa, jotka saavutetaan enintään j askeleella jostain alkukonfiguraatiosta, vai ei. Jos arvaus on „ei”, M_{C} siirtyy seuraavaan kokeiltavaan konfiguraatioon lisäämättä laskuria λ_2. Jos taas arvaus on „kyllä”, M_{C} arvaa konfiguraatioon κ' jostain alkukonfiguraatiosta enintään j askeleella vievän konfiguraatioketjun. Tällainen arvaus tehdään siirto kerrallaan. Mikäli konfiguraatiot κ' on käyty läpi ja $\lambda_2 < N_j$, M_{C} pysähtyy ei-lopputilaan hyväksymättä syötettä w.

M_{C} tarkistaa onko arvattu κ':uun päättyvä konfiguraatioketju oikea. On kaksi vaihtoehtoa.

[4] Alkuperäisviitteet ovat Immerman, N.: Nondeterministic Space is Closed under Complementation. *SIAM Journal of Computing* **17** (1988), 275–303 ja Szelepcsényi, R.: The Method of Forced Enumeration for Nondeterministic Automata. *Acta Informatica* **26** (1988), 279–284.

(1) Jos konfiguraatioketju on oikea, M_{C} tarkistaa onko $\kappa = \kappa'$ tai onko $\kappa' \vdash_M \kappa$, ja positiivisessa tapauksessa lisää laskuria λ_1 yhdellä sekä siirtyy tutkimaan konfiguraatiota $\kappa_{\ell+2}$. Jos laskuri λ_2 saavuttaa maksimiarvonsa N_j eikä vieläkään ole löytynyt etsittyä konfiguraatiota, M_{C} päättelee, ettei konfiguraatiota κ voida saavuttaa enintään $j+1$ askeleella alkukonfiguraatiosta ja siirtyy tutkimaan konfiguraatiota $\kappa_{\ell+2}$ lisäämättä laskuria λ_1.

(2) Jos arvattu konfiguraatioketju ei ole oikea, M_{C} pysähtyy ei-lopputilaan hyväksymättä syötettä w.

Lopulta M_{C} on käynyt läpi kaikki mahdolliset konfiguraatiot κ_ℓ ja on saanut selville luvut N_j, ja näin myös luvun N. Huomaa, että aina siirtyessään tutkimaan uutta konfiguraatiota κ_ℓ LBA M_{C} nollaa laskurin λ_2, ja aina siirtyessään laskemaan uutta lukua N_j se nollaa laskurin λ_1.

M_{C}:n toiminnan toinen vaihe on käydä jälleen läpi kaikki n-pituiset konfiguraatiot pitäen yllä laskuria λ. Kun tarkasteltavana on konfiguraatio κ, M_{C} ensin arvaa onko se jonkin w:n alkukonfiguraation seuraaja vai ei. Jos arvaus on „ei”, M_{C} siirtyy tutkimaan listassa seuraavaa konfiguraatiota lisäämättä laskuria λ. Jos taas arvaus on „kyllä”, M_{C} arvaa konfiguraatioon κ jostain w:n alkukonfiguraatiosta vievän enintään $j_{\max}$ askeleen konfiguraatioketjun. Nyt on useita mahdollisuuksia:

1. Jos arvattu konfiguraatioketju ei ole oikea tai se on oikea, mutta κ on M:n hyväksyvä loppukonfiguraatio, M_{C} pysähtyy hyväksymättä syötettä w.

2. Jos arvattu konfiguraatioketju on oikea ja κ ei ole M:n hyväksyvä loppukonfiguraatio ja λ ei ole saavuttanut maksimiarvoaan N, M_{C} lisää laskuria λ yhdellä ja siirtyy tutkimaan listassa seuraavaa konfiguraatiota.

3. Jos arvattu konfiguraatioketju on oikea ja κ ei ole M:n hyväksyvä loppukonfiguraatio ja λ on saavuttanut maksimiarvon N, M_{C} hyväksyy syötteen w. □

CS-kieliin liittyvät karakterisaatioprobleemat ovat kutakuinkin kaikki algoritmisesti ratkeamattomia. Jäsenyysprobleemahan on, kuten todettu, sentään algoritmisesti ratkeava. Erityisesti *tyhjyys-, äärellisyys-, universaalisuus-, ekvivalenssi- ja säännöllisyysprobleemat ovat ratkeamattomia.* Universaalisuus, ekvivalenssi sekä säännöllisyys olivat ratkeamattomia jo CF-kielille, joten ne ovat sitä myös CS-kielille, ja koska CS-kielet ovat suljettuja komplementin suhteen, myös tyhjyys on niille ratkeamaton.

Äärellisyyden ratkeamattomuus vaatii lisäkonstruktion. Se voidaan tehdä vaikkapa Pykälässä 4.7 käsitellyn Postin vastaavuusprobleeman avulla. Kutakin PCP:n syötemorfismiparia

$$\sigma_1 : \Sigma^* \to \Delta^* \quad \text{ja} \quad \sigma_2 : \Sigma^* \to \Delta^*$$

kohti konstruoidaan LBA M, jonka syöteaakkosto on $\{a\}$ ja joka saatuaan syötteen a^n tarkistaa onko sellaista sanaa $w \in \Sigma^+$, että $|w| \leq n$ ja $\sigma_1(w) = \sigma_2(w)$. (Ei ole kovin vaikeaa nähdä, että tämä on mahdollista.) Jos mainitunlainen sana w on olemassa, M hylkää syötteen a^n, muuten ei. Näin ollen $L(M)$ on äärellinen tarkalleen silloin, kun on olemassa sellainen sana $w \in \Sigma^+$, että $\sigma_1(w) = \sigma_2(w)$.

Postin vastaavuusprobleeman avulla voidaan myös näyttää tyhjyyden ratkeamattomuus suoraan, ilman Immerman–Szelepcsényi-lausetta, sillä kieli

$$\bigl\{ w \bigm| w \in \Sigma^+ \text{ ja } \sigma_1(w) = \sigma_2(w) \bigr\}$$

on CS-kieli, kuten voi helposti todeta.

Luku 6

CE-KIELET

6.1 Turingin kone

Turingin kone (*TM*) on samantapainen kuin LBA, mutta siinä ei ole reunamerkkejä ja muistinauha on potentiaalisesti molempiin suuntiin ääretön. Nauha-aakkostossa on erikoinen merkki B, ns. *blanko*. Kussakin vaiheessa vain äärellisen moni nauhasymboli on jotain muuta kuin blanko. Alussa nauhalla on syötesana, jonka kahtapuolta on blankoja, ja lukipää lukee syötteen ensimmäistä symbolia. Tyhjän sanan tapauksessa lukipää lukee jotain blankoa.

Formaalisti Turingin kone on seitsikko

$$M = (Q, \Sigma, \Gamma, S, B, \delta, A),$$

missä Q, Σ, Γ, S, δ ja A ovat kuten LBA:lle ja B on blanko. Siirto määritellään kuten LBA:lle, mitään rajoituksia lukipään liikkeelle ei ole, koska ei ole reunamerkkejä. Ainoa lisäehto on, että jos $(q_j, Y, s) \in \delta(q_i, X)$, niin $Y \neq B$. Näin määritelty Turingin kone on luonteeltaan epädeterministinen. *Deterministinen Turingin kone* (*DTM*) määritellään kuten deterministinen LBA.

Turingin koneen konfiguraatio määritellään hieman toisin kuin LBA:lle. Jos lukipää ei lue blankoa, niin *konfiguraatio* on nelikkö (q_i, α, X, β), missä q_i on tila, $\alpha X \beta$ on pisin nauhalla oleva sana, joka ei sisällä blankoja, ja X on nauhasymboli, jota M ko. sanassa lukee. Jos taas lukipää lukee blankoa, on konfiguraatio (q_i, Λ, B, β) (vast. $(q_i, \alpha, B, \Lambda)$), missä β (vast. α) on pisin nauhalla oleva sana, joka ei sisällä blankoja. Erityisesti, jos nauhalla on vain blankoja eli nauha on tyhjä, on vastaava konfiguraatio $(q_i, \Lambda, B, \Lambda)$.

Sanan hyväksyminen ja kielen tunnistaminen määritellään kuten LBA:lle.

Lause 24. *Jokainen CE-kieli voidaan tunnistaa Turingin koneella.*

Todistus. Todistus on hyvin samankaltainen kuin Lauseen 19 todistus. Ainoa ero on, että Tyypin 0 kielioppi voi myös pyyhkiä pois symboleja eli sana voi uudelleenkirjoitettaessa lyhentyä. Tällöin jo johdetun sanan loppuosaa pitää siirtää vasemmalle. □

Lause 25. *Turingin koneiden tunnistamat kielet ovat CE-kieliä.*

Todistus. Todistus on kutakuinkin samanlainen kuin Lauseen 20 todistus. Mainittakoon vain pari yksityiskohtaa. Blankoja on mukana kaksi, yksi kummassakin uudelleenkirjoitettavan sanan päässä. Lopuksi ne pyyhitään pois. Lopuksi myös terminoidaan symbolit, joissa on tallella „syöte", ja muut pyyhitään pois. Tämä prosessi alkaa, kun simuloitava TM on lopputilassa ja etenee „aaltona" molempiin suuntiin. □

Turingin koneiden tunnistamat kielet ovat näin ollen tarkalleen kaikki CE-kielet.

Kielten tunnistamisen kannalta voitaisiin rajoittua deterministisiin Turingin koneisiin. Jokaista Turingin konetta M voidaan nimittäin simuloida DTM:llä M', joka tunnistaa saman kielen. Simuloivalla DTM:llä M' on i:nnessä vaiheessa aina nauhallaan peräkkäin jollain tavalla koodattuna kaikki ne M:n konfiguraatiot, jotka voidaan saada tiettyä syötettä w vastaavista alkukonfiguraatioista enintään i:llä askeleella. 0:nnessa vaiheessa (alussa) M' laskee nauhalleen kaikki syötettä w vastaavat M:n alkukonfiguraatiot, joita on yhtä paljon kuin alkutiloja. Siirryttäessä vaiheesta i vaiheeseen $i+1$ DTM M' yksinkertaisesti käy läpi viimeksi saamansa konfiguraatiot, jatkaa niistä kustakin yhdellä askeleella kaikilla mahdollisilla tavoilla ja tallettaa tulokset nauhalleen. M' pysähtyy löytäessään M:n loppukonfiguraation.

Perhe $\mathcal{CE}$ todettiin jo suljetuksi yhdisteen, katenaation ja katenaatiosulkeuman sekä peilikuvan suhteen. Aivan samalla tavalla kuin Lause 23, saadaan

Lause 26. *$\mathcal{CE}$ on suljettu leikkauksen suhteen.*

Huolimatta siitä, että $\mathcal{CE}$:n kielet voidaan tunnistaa deterministisillä Turingin koneilla, $\mathcal{CE}$ ei ole suljettu komplementin suhteen. Syynä on se, että (D)TM voi ottaa ääretömän monta askelta pysähtymättä lainkaan. Niin voi muuten LBA:kin, mutta sopivalla laskuritekniikalla tämä voidaan estää (harjoitus). Asian lähemmäksi tutkimiseksi otetaan käyttöön ns. universaali Turingin kone. Riittää tarkastella vain Turingin koneita, joiden syöteaakkosto on $\{0,1\}$.[1] Tällaiset Turingin koneet voidaan koodata binääriluvuiksi. Tämä voidaan tehdä vaikkapa antamalla ensin δ viisikkojen joukkona P:

$$(q_j, Y, s, q_i, X) \in P$$

tarkalleen silloin, kun $(q_j, Y, s) \in \delta(q_i, X)$. Sen jälkeen koodataan esiintyvät symbolit binäärimuotoon: Symbolit

$$(\ )\ \{\ \}\ ,\ 0\ 1\ +\ -$$

koodataan tässä järjestyksessä binäärisanoiksi 10^i ($i = 1, \ldots, 9$). i:s tila koodataan binäärisanaksi 110^i ja j:s nauhasymboli sanaksi 1110^j. Tämän jälkeen Turingin kone M voidaankin esittää binäärilukuna $\beta(M)$.

Universaali Turingin kone (*UTM*) on sellainen TM U, joka saatuaan syötteen[2] $w\$\beta(M)$, missä $w \in \{0,1\}^*$, simuloi Turingin konetta M syötteellä w, ts. U hyväksyy ko. syötteen, kun M hyväksyy syötteen w. Muut syötteet U hylkää.

Lause 27. *On olemassa universaali Turingin kone U.*

Todistus. Menemättä sen kummemmin yksityiskohtiin, U toimii seuraavasti. Ensin U tarkistaa, että syöte on oikeaa muotoa $w\$\beta(M)$. Myönteisessä tapauksessa U „dekoodaa" $\beta(M)$:n saadakseen käyttöönsä M:n siirtofunktion δ. Sen jälkeen U simuloi M:ää syötteellä w askel askeleelta käyden aina välillä hakemassa siirtosäännöt dekoodatusta δ:sta. Kun M menee johonkin lopputilaansa, niin samoin tekee U. □

Lause 28. *$\mathcal{CE}$ ei ole suljettu komplementin suhteen.*

[1] Aakkosto saisi kyllä olla mikä vaan, myös yksikirjaiminen!

[2] Kaikkien tätä muotoa olevien syötteiden muodostama kieli voidaan tunnistaa LBA:lla, kuten voi aika helposti todeta.

Todistus. „Lävistäjäkieli”

$$D = \{w \mid w\$w \in L(U)\}$$

on CE, kuten on helposti todettavissa—kopioidaan vain w ja simuloidaan U:ta. D:ssä ovat siis niiden Turingin koneiden koodit, jotka „hyväksyvät itsensä”. Kuitenkaan $\overline{D}$ ei ole CE. Jos se olisi, niin sen tunnistaisi jokin Turingin kone M. Mutta silloinhan sana $\beta(M)$ on kielessä D tarkalleen silloin, kun se ei ole siinä! □

Yo. todistuksessa on jälleen käytetty lävistäjämenetelmää, vrt. Lauseiden 1 ja 22 todistukset. Todistuksessa esiintyvä $\overline{D}$ on esimerkki kielestä, jolla on selvästi äärellinen määritelmä, mutta joka silti ei ole CE.

$\overline{D}$ on co–CE-kieli, joka ei ole CE. Vastaavasti kieli D on CE-kieli, joka ei ole co–CE. Perheiden $\mathcal{CE}$ ja co$-\mathcal{CE}$ leikkaus $\mathcal{C}$ (laskettavat kielet) sisältyy siis aidosti kumpaankin näistä perheistä.

6.2 Algoritminen ratkeavuus

Deterministiseen Turingin koneeseen voidaan liittää *tulostus.* Koneen pysähtyessä lopputilaan tuloste on nauhalla sopivasti erotettuna muista siellä olevista symboleista. Huomaa, että tulostetta ei aina tule, sillä kone voi pysähtyä ei-lopputilaan tai olla pysähtymättä kokonaan.

Yleisesti hyväksytyn käsityksen mukaan *algoritmeilla* tarkoitetaan tarkalleen niitä menetelmiä, jotka voidaan toteuttaa Turingin koneilla. Tämä tunnetaan ns. *Church–Turing-teesinä.* Algoritmisesti ratkeavat tehtävät ovat siis tarkalleen ne tehtävät, jotka ainakin periaatteessa voidaan ratkaista Turingin koneella.

Eräs seuraus Church–Turing-teesistä on, että on olemassa selvästi äärellisesti määriteltyjä tehtäviä, joita ei voida ratkaista algoritmisesti.

Lause 29. *On CE-kieli, jonka jäsenyysprobleema ei ole ratkaistavissa algoritmisesti. Huomaa, että algoritmin tulisi tällöin aina tulostaa kyllä/ei-vastaus riippuen siitä onko sana kielessä vai ei.*

Todistus. Lauseen 28 todistuksessa esiintyvä lävistäjäkieli D on CE-kieli, jonka jäsenyysprobleema ei ole algoritmisesti ratkeava. Muutenhan myös kielen $\overline{D}$ jäsenyysprobleeman voisi ratkaista deterministisellä Turingin koneella ja $\overline{D}$ olisi näin CE-kieli. □

Deterministisen Turingin koneiden *pysähtymisprobleemassa* on syötteenä sana w ja kysytään pysähtyykö deterministinen Turingin kone M saatuaan syötteen w.

Lause 30. *On Turingin kone, jonka pysähtymisprobleema on algoritmisesti ratkeamaton.*

Todistus. On helppo muuntaa Turingin konetta siten, että se pysähtyy vain lopputilaan lisäämällä tarvittaessa toiminto, joka jättää koneen äärettömään silmukkaan, mikäli se olisi pysähtymässä ei-lopputilaan. Tulos seuraa nyt edellisestä lauseesta. □

Kutakuinkin kaikki CE-kieliä koskevat karakterisaatioprobleemat ovat algoritmisesti ratkeamattomia. Itse asiassa jokainen ei-triviaali CE-kielten ominaisuus on algoritmisesti ratkeamaton. Ei-triviaali tarkoittaa ominaisuutta, joka on joillain mutta ei kaikilla CE-kielillä.

Ricen lause. *Jos O on ei-triviaali ominaisuus, niin se on CE-kielille algoritmisesti ratkeamaton. Syöte on tässä CE-kieli annettuna sen tunnistavan Turingin koneen muodossa.*

Todistus. Tyhjällä kielellä $\emptyset$ joko on ominaisuus O tai ei ole sitä. Vaihtamalla tarvittaessa ominaisuus O negaatiokseen, voidaan olettaa, että tyhjällä kielellä ei ole ominaisuutta O. Koska ominaisuus O on ei-triviaali, on myös sellainen CE-kieli L_1, jolla on ominaisuus O, ja sen tunnistava Turingin kone M_1.

Valitaan CE-kieli L, jonka jäsenyysprobleema on ratkeamaton, ja sen tunnistava Turingin kone M.

Otetaan kielen L aakkoston sana w ja määritellään sen avulla Turingin kone M_w seuraavasti. M_w:n syöteaakkosto on sama kuin M_1:n. Syötteen v saatuaan M_w aloittaa tutkimalla onko w kielessä L simuloiden Turingin konetta M. Myönteisessä tapauksessa M_w jatkaa simuloiden Turingin konetta M_1 syötteellä v, jonka se on huolellisesti tallentanut tätä varten, ja hyväksyy syötteen tarkalleen silloin kuin M_1:kin. Kielteisessä tapauksessa M_w ei hyväksy mitään. Huomaa, että jos M:n simulointi ei pysähdy, niin tulos on joka tapauksessa oikea! Huomaa myös, että vaikka ei olekaan Turingin konetta, joka selvittäisi pysähtyykö M syötteellä w vai ei, Turingin kone M_w voidaan konstruoida algoritmisesti.

Sana w on näin muunnettu koneeksi M_w, jolla on seuraava ominaisuus: Kielellä $L(M_w)$ on ominaisuus O tarkalleen silloin, kun $w \in L$. Koska vm. kuuluminen on algoritmisesti ratkeamaton, on sitä myös ominaisuus O. □

Ilmeisesti esimerkiksi seuraavat probleemat ovat suoraan Ricen lauseen nojalla algoritmisesti ratkeamattomia CE-kielille:

- Tyhjyysprobleema
- Äärellisyysprobleema
- Universaalisuusprobleema
- Säännöllisyysprobleema
- *Laskettavuusprobleema:* Onko annettu kieli laskettavissa?

Mutta myös esimerkiksi Ekvivalenssiprobleema on algoritmisesti ratkeamaton CE-kielille Ricen lauseen nojalla, koska jo „heikompi" Universaalisuusprobleema on ratkeamaton.

Turingin koneista on lukuisia variantteja, koneita, joilla on useita nauhoja tai moniulotteisia nauhoja jne. Millään näistä, eikä muillakaan algoritmimäärittelyillä, saada tunnistetuksi laajempaa kieliperhettä kuin $\mathcal{CE}$.

Toisaalta CE-kielet voidaan tunnistaa vieläkin rajoitetummilla Turingin koneilla kuin deterministisillä. DTM:n M sanotaan olevan *reversiibeli Turingin kone* (*RTM*), jos on toinen DTM M', jolla on se ominaisuus, että jos $\kappa_1 \vdash_M \kappa_2$ konfiguraatiolle κ_1 ja κ_2, niin (pienin teknisin muutoksin) $\kappa_2 \vdash_{M'} \kappa_1$. DTM M' siis toimii kuin M takaperin! Yves Lecerf todisti jo vuonna 1962, että jokainen CE-kieli voidaan tunnistaa RTM:llä.[3] Vastaavasti jokainen algoritmi voidaan korvata reversiibelillä algoritmilla. Tämä tulos on tullut hyvin tärkeäksi mm. kvanttilaskennassa.

6.3 Aikaluokat (lyhyt katsaus)

Chomskyn hierarkia merkitsee automaattipuolella lähinnä muistin erilaisten rajoitusten vaikutusta. Vastaavanlaisia rajoituksia voidaan asettaa *ajalle,* ts. askelien (siirtojen) lukumäärälle.

[3]Alkuperäisviite on LECERF, M.Y.: Machines de Turing réversibles. Récursive insolubilité en $n \in N$ de l'équation $u = \theta^n u$, où θ est un "isomorphisme de codes". *Comptes Rendus* **257** (1963), 2597–2600.

Askelten lukumäärää ei kannata rajoittaa yksityiskohtaisesti, vaan vain asymptoottisesti eli vain kyllin pitkille syötesanoille sekä vakiokertojaa vaille, sillä automaattia voidaan helposti „kiihdyttää lineaarisesti".

Aikaluokka $\mathcal{TIME}\big(f(n)\big)$ tarkoittaakin niiden kielten L perhettä, jotka voidaan tunnistaa seuraavasti. Joillekin vakioille C ja n_0 kieli L voidaan tunnistaa deterministisellä Turingin koneella, joka käyttää enintään $Cf(n)$ askelta jokaiselle n-pituiselle syötesanalle, kun $n \geq n_0$. Vastaavasti määritellään aikaluokka $\mathcal{NTIME}\big(f(n)\big)$, Turingin kone vain on epädeterministinen. Käyttäen eri funktioita $f(n)$ saadaan aikaluokille mutkikas ääretön hierarkia perheen $\mathcal{C}$ sisään.

Ajatellen käytännön toteutusta, eivät ilmeisestikään esimerkiksi aikaluokat $\mathcal{TIME}(2^n)$ ja $\mathcal{NTIME}(2^n)$ ole realistisia, sillä syötesanan käsittely vie liiaksi aikaa. Toisaalta luokat $\mathcal{TIME}(n^d)$ (polynomiaikaiset kielet) ovat tässä mielessä paremmin toteutettavissa. Kaikkien polynomiaikaisten kielten perhettä

$$\mathcal{P} = \bigcup_{d \geq 1} \mathcal{TIME}(n^d)$$

pidetäänkin yleisesti laskennallisesti tehokkaasti („tractable") tunnistettavien kielten perheenä.

Luokat $\mathcal{NTIME}(n^d)$ eivät ole samalla tavoin selvästi toteutettavissa kuin $\mathcal{P}$. Perhe

$$\mathcal{NP} = \bigcup_{d \geq 1} \mathcal{NTIME}(n^d)$$

voidaan määritellä, mutta on kuuluisa avoin probleema onko $\mathcal{P} = \mathcal{NP}$! Luokasta $\mathcal{NP}$ on lisäksi löytynyt „universaaleja" kieliä, ns. *$\mathcal{NP}$-täydellisiä kieliä,* joilla on se ominaisuus, että jos yksikin niistä on $\mathcal{P}$:ssä, niin $\mathcal{P} = \mathcal{NP}$.

Luku 7

KOODIT

7.1 Koodi. Schützenberger'n kriteeri

Kieli L on *koodi,* jos se on ei-tyhjä ja L^*:n sanat ovat yksikäsitteisesti purettavissa L:n sanojen katenaatioksi. Tätä purkuoperaatiota kutsutaan *dekoodaukseksi.* Tarkemmin sanoen, jos $u_1, \ldots, u_m$ ja $v_1, \ldots, v_n$ ovat L:n sanoja ja

$$u_1 \cdots u_m = v_1 \cdots v_n,$$

niin $u_1 = v_1$, mikä toistettuna takaa yksikäsitteisen dekoodauksen. Virheitä korjaavat koodit (ks. kurssi Koodausteoria) ja jotkut salauskoodit (ks. kurssi Matemaattinen kryptologia) ovat tällaisia koodeja, mutta sellaisina sangen erikoista tyyppiä. Sen sijaan tiedon pakkauksessa käytetyt koodit (ks. kurssi Informaatioteoria) ovat melko yleistä koodityyppiä.

Seuraavassa eräitä melko ilmeisiä huomioita:

- Koodissa ei ole tyhjää sanaa.
- Koodin ei-tyhjä alikieli on myös koodi, ns. *alikoodi.*
- Jos koodin aakkostossa on vain yksi symboli, niin koodissa on tarkalleen yksi sana.

Koska yksikirjaimiset koodit ovat näin yksinkertaisia, oletetaan jatkossa, että aakkostot ovat ainakin kaksikirjaimisia. Koodeilla ei yleensä ole kovin kummoisia sulkeumaominaisuuksia. Kahden koodin leikkaus on kuitenkin koodi, ellei se ole tyhjä. Koodin peilikuva ja potenssit ovat myös koodeja.

Koodi voidaan määritellä ja karakterisoida monenlaisin sen sanoja koskevin ehdoin. Kielen L sanotaan olevan *katenatiivisesti riippumaton,* jos

$$L \cap LL^+ = \emptyset,$$

ts. mitään kielen L sanaa ei voida esittää useamman L:n sanan katenaationa. Ilmeisesti jokainen koodi on katenatiivisesti riippumaton. Kääntäen tämä ei pidä yleisesti paikkaansa, joten tarvitaan lisäehtoja:

Schützenberger'n kriteeri. *Aakkoston Σ katenatiivisesti riippumaton ei-tyhjä kieli L on koodi tarkalleen silloin, kun kaikille sanoille $w \in \Sigma^*$ pätee seuraava ehto:*

(∗) *Jos on sellaiset L^*:n sanat t, x, y ja z, että $wt = x$ ja $yw = z$, niin $w \in L^*$.*

Todistus. Oletetaan ensiksi, että ehto (∗) on voimassa ja näytetään, että L on koodi. Asetetaan vastaoletus: L ei ole koodi. Silloin on sellaiset L:n sanat $u_1, \ldots, u_m$ ja $v_1, \ldots, v_n$ että

$$u_1 \cdots u_m = v_1 \cdots v_n,$$

mutta $u_1 \neq v_1$. Toinen sanoista u_1 ja v_1 on toisen aito prefiksi, olkoon vaikkapa $v_1 = u_1 w$, missä $w \neq \Lambda$. Nyt w on v_1:n suffiksi ja $u_2 \cdots u_m$:n prefiksi, joten ehdon (∗) nojalla w on L^+:ssa. Tällöin kuitenkin L on katenatiivisesti riippuva, mikä on ristiriita.

Toiseksi oletetaan, että L on koodi. Asetetaan jälleen vastaoletus: On sellaiset L^*:n sanat t, x, y ja z, että $wt = x$ ja $yw = z$, mutta $w \notin L^*$. Tällöin mikään näistä sanoista t, x, y, z ja w ei ole tyhjä. Kirjoitetaan y ja z L:n sanojen katenaatioiksi

$$y = u_1 \cdots u_m \quad \text{ja} \quad z = v_1 \cdots v_n,$$

jolloin

$$u_1 \cdots u_m w = v_1 \cdots v_n.$$

Ilmeisesti voidaan olettaa, että $u_1 \neq v_1$. Mutta nyt

$$v_1 \cdots v_n t = u_1 \cdots u_m w t = u_1 \cdots u_m x$$

ja $u_1 \neq v_1$ eikä L näin voi olla koodi, jälleen ristiriita. □

7.2 Sardinas–Patterson-algoritmi

Sinällään Schützenberger'n kriteeri ei anna algoritmia sen ratkaisemiseksi onko äärellinen kieli L koodi vai ei. Sen avulla voidaan kuitenkin johtaa[1] tällaisia algoritmeja, esimerkiksi klassinen *Sardinas–Patterson-algoritmi:*

1. Asetetaan $i \leftarrow 0$ sekä $L_i \leftarrow L$.
2. Asetetaan $i \leftarrow i + 1$ ja
$$L_i \leftarrow \{w \mid w \neq \Lambda \text{ ja } xw = y \text{ tai } yw = x \text{ joillekin sanoille } x \in L \text{ ja } y \in L_{i-1}\}.$$
3. Jos $L_i \cap L \neq \emptyset$, niin tulostetaan „ei" ja lopetetaan.
4. Jos $L_i = L_j$ jollekin indeksille $1 \leq j < i$, niin tulostetaan „kyllä" ja lopetetaan. Muutoin mennään kohtaan 2.

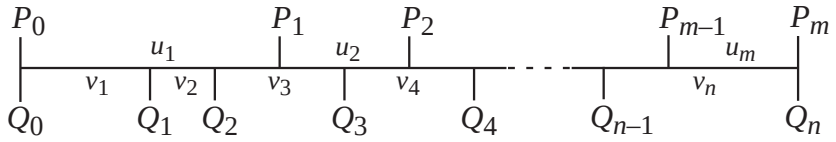
Lause 31. *Sardinas–Patterson-algoritmi tuottaa oikean tuloksen.*

Todistus. Osoitetaan ensin, että jos L ei ole koodi, niin jossain kielistä $L_1, L_2, \ldots$ on L:n sana. Oletetaan siis, ettei L ole koodi. Tapaus $\Lambda \in L$ on heti ilmeinen, sillä silloin $L_1 \cap L \neq \emptyset$. Muutoin on sellaiset L:n sanat $u_1, \ldots, u_m$ ja $v_1, \ldots, v_n$ että

$$u_1 \cdots u_m = v_1 \cdots v_n = w,$$

mutta $u_1 \neq v_1$. Havainnollistetaan tilannetta graafisesti:

[1] Ks. esimerkiksi Berstel & Perrin.



Kuviossa on merkitty osasanojen päitä pisteillä. Esimerkiksi aina

$$P_{i-1}P_i = u_i \quad \text{ja} \quad Q_{j-1}Q_j = v_j.$$

Alkupiste on $P_0 = Q_0$ ja loppupiste $P_m = Q_n$. Voidaan olettaa, ettei muita yhteisiä pisteitä ole, muuten korvataan w jollain prefiksillään. Jaetaan w osasanoihin käyttäen kaikkia pisteitä $P_1, \ldots, P_m, Q_1, \ldots, Q_n$. Muotoa P_iQ_j tai Q_jP_i olevat osasanat, missä P_i ja Q_j ovat peräkkäiset jakopisteet ja $i, j \geq 1$, ovat nyt yhdisteessä $L_1 \cup L_2 \cup \cdots$, kuten algoritmin määrittelystä on helppo nähdä rekursiivisesti aloittaen sanan w alusta.

Jos nyt loppupistettä $P_m = Q_n$ edeltävä piste on P_{m-1}, kuten yo. kuviossa, niin osasana $Q_{n-1}P_{m-j}$, missä P_{m-j} on Q_{n-1}:tä seuraava piste, on jossain kielessä L_i ($i \geq 1$). Mutta silloin $u_m = P_{m-1}P_m$ on kielessä L_{i+j}. Vastaavasti käsitellään tapaus, missä loppupistettä $P_m = Q_n$ edeltävä piste on Q_{n-1}.

Todetaan vielä, että jos L on koodi, niin missään kielistä $L_1, L_2, \ldots$ ei ole L:n sanoja. Tähän tarvitaan Schützenberger'n kriteeri.

Näytetään ensiksi, että jokaiselle sanalle $w \in L_i$ on $L^*w \cap L^* \neq \emptyset$, ts. on sellaiset sanat $s, t \in L^*$, että $sw = t$. Asia on selvä, kun $i = 0$. Jatketaan tästä induktiolla ja oletetaan, että jokainen L_{i-1}:n sana on esitetyllä tavalla jonkin L^*:n sanan suffiksi. Otetaan tarkasteltavaksi L_i:n sana w. L_i:n määrittelystä algoritmissa seuraa, että on sellaiset sanat $x \in L$ ja $y \in L_{i-1}$, että

$$xw = y \quad \text{tai} \quad yw = x.$$

Induktio-oletuksen mukaisesti puolestaan on sellaiset sanat $s, t \in L^*$, että $sy = t$. Siispä

$$sxw = sy = t \quad \text{tai} \quad sx = syw = tw.$$

Tulos pätee siis myös L_i:lle.

Oletetaan nyt, että kumminkin $w \in L \cap L_i$ jollekin indeksille $i \geq 1$, ja johdetaan ristiriita. Algoritmin mukaisesti silloin on sellaiset sanat $x \in L$ ja $y \in L_{i-1}$, että

$$yw = x \quad \text{tai} \quad xw = y.$$

Jos $yw = x$, niin silloin on $L^*y \cap L^* \neq \emptyset$ (edeltä) ja $L^* \cap yL^* \neq \emptyset$ ja Schützenberger'n kriteerin mukaisesti $y \in L^*$, mikä on ristiriita (tapaus $i = 1$ on selvä muutenkin). Siispä pätee $xw = y$, jolloin myös on oltava $i \geq 2$ (miksi?) ja $y \in L^*$. Määrittelyn mukaisesti on sellaiset sanat $x' \in L$ ja $y' \in L_{i-2}$, että

$$y'y = x' \quad \text{tai} \quad x'y = y'.$$

Edellinen vaihtoehto suljetaan pois kuten edellä Schützenberger'n kriteerillä. Jäljelle jää $x'y = y'$, jolloin on oltava $i \geq 3$ (miksi?) ja $y' \in L^*$.

Jne. Jatkaen tällä tavalla todetaan, että mikään i:n arvo ei käy.

Lopuksi todetaan, että jos L on koodi, niin algoritmi pysähtyy kohtaan 4. Kieliä L_i on nimittäin äärellinen määrä, koska L on äärellinen (ajattele sanojen pituuksia). □

Äärettömien kielten tapaus on mutkikkaampi. Se, onko annettu säännöllinen kieli koodi, on kyllä algoritmisesti ratkaistavissa. Tapaus, jossa kielessä on tyhjä sana, on tietysti selvä. Lähtien L:n tunnistavasta DFA:sta on melko helppoa muodostaa Λ-NFA M, joka hyväksyy tarkalleen kaikki sanat, mitkä voidaan kirjoittaa ainakin kahdella eri tavalla L:n sanojen katenaatioksi (vrt. Kleenen lauseen todistus). Kysymys onkin sitten säännöllisen kielen $L(M)$ tyhjyydestä.[2]

Toisaalta ei ole yleistä algoritmia sen ratkaisemiseksi onko annettu CF-kieli koodi. Tämä voidaan näyttää vaikkapa Pykälässä 4.7 käsitellyn Postin vastaavuusprobleeman avulla. Kutakin PCP:n syötemorfismiparia

$$\sigma_1 : \Sigma^* \to \Delta^* \quad \text{ja} \quad \sigma_2 : \Sigma^* \to \Delta^*$$

kohti konstruoidaan aakkoston $\Sigma \cup \Delta \cup \{\#, \$\}$ kieli

$$L = \left\{\sigma_1(w)\#\hat{w}\$ \;\middle|\; w \in \Sigma^+\right\} \cup \left\{\sigma_2(w)\#\hat{w}\$\sigma_1(v)\#\hat{v}\$ \;\middle|\; w, v \in \Sigma^+\right\}.$$

On helppo nähdä, että L on CF-kieli ja että se on koodi tarkalleen silloin, kun ei ole sellaista sanaa $w \in \Sigma^+$, että $\sigma_1(w) = \sigma_2(w)$.

7.3 Indikaattorisumma. Prefiksikoodit

Koodien, ja erityisesti virheitä korjaavien koodien, teorian keskeinen työkalu ovat erilaiset koodeihin liittyvät numeeriset summat, joilla teoriaa aritmetisoidaan, ks. kurssi Koodausteoria.

Aakkoston Σ kielen L ns. *indikaattorisumma* on

$$\mathrm{is}(L) = \sum_{w \in L} M^{-|w|},$$

missä M on Σ:n symbolien lukumäärä. Yleisesti indikaattorisumma voi olla ääretön, esimerkiksi

$$\mathrm{is}(\Sigma^*) = \sum_{n=0}^{\infty} M^n M^{-n} = \infty.$$

Koodeissa sanoja on harvassa ja tilanne on toinen:

Markov–McMillan-lause. *Jokaiselle koodille L on* $\mathrm{is}(L) \leq 1$.

Todistus. Tarkastellaan ensin äärellistä koodia $\{w_1, w_2, \ldots, w_k\}$, jonka koodisanojen pituudet ovat

$$l_1 \leq l_2 \leq \cdots \leq l_k.$$

Otetaan mielivaltainen positiivinen kokonaisluku r (lopulta $r \to \infty$). Silloin

$$\Big(\sum_{n=1}^{k} M^{-l_n}\Big)^r = \underbrace{\Big(\sum_{n=1}^{k} M^{-l_n}\Big) \cdots \Big(\sum_{n=1}^{k} M^{-l_n}\Big)}_{r \text{ kpl}} = \sum_{n_1=1}^{k} \sum_{n_2=1}^{k} \cdots \sum_{n_r=1}^{k} M^{-l_{n_1} - l_{n_2} - \cdots - l_{n_r}}.$$

[2] Tämä toimii tietysti myös äärellisille kielille, mutta Sardinas–Patterson on parempi!

Summa $l_{n_1} + l_{n_2} + \cdots + l_{n_r}$ on r:n koodisanan katenaation $w_{n_1} w_{n_2} \cdots w_{n_r}$ pituus. Indeksien $n_1, n_2, \ldots, n_r$ kulkiessa toisistaan riippumatta arvot $1, 2, \ldots, k$ saadaan kaikki r:n koodisanan katenaatiot, mahdolliset toistot mukaanlukien.

Merkitään nyt s_j:llä niiden r:stä koodisanasta eri tavoin katenoimalla saatujen sanojen lukumäärää, mahdolliset toistot mukaan lukien, joiden pituus on j. Eri tavoin katenoimalla ei kuitenkaan voida saada samaa sanaa, koska kyseessä on koodi, joten $s_j \leq M^j$. Ilmeisesti mahdolliset j:n arvot ovat

$$j = rl_1, rl_1 + 1, \ldots, rl_k.$$

Näin ollen

$$\Big(\sum_{n=1}^{k} M^{-l_n} \Big)^r = \sum_{j=rl_1}^{rl_k} s_j M^{-j} \leq \sum_{j=rl_1}^{rl_k} M^j M^{-j} = rl_k - rl_1 + 1 \leq rl_k.$$

Otetaan puolittain r:s juuri ja annetaan $r \to \infty$:

$$\sum_{n=1}^{k} M^{-l_n} \leq \sqrt[r]{rl_k} \to 1,$$

sillä

$$\lim_{r \to \infty} \ln \sqrt[r]{rl_k} = \lim_{r \to \infty} \frac{\ln r + \ln l_k}{r} = 0.$$

Koska arvion vasen puoli ei riipu r:stä, pätee arvio sille myös rajalla $r \to \infty$.

Äärettömille koodeille tulos seuraa edellisestä. Jos nimittäin tulos ei pitäisi paikkaansa jollekin äärettömälle koodille, se ei myöskään pidä paikkaansa sen jollekin äärelliselle alikoodille. □

Markov–McMillan-lause ei ole yleisesti käännettävissä, ts. on kieliä, joille $\mathrm{is}(L) \leq 1$ ja jotka eivät ole koodeja. Eräänlainen käänteinen tulos on kuitenkin olemassa:

Kraftin lause. *Jos* $\sum_{n=1,2,\ldots} M^{-l_n} \leq 1$*, niin on sellainen* M*-kirjaimisen aakkoston koodi, jonka sanojen pituudet ovat* $l_1, l_2, \ldots$ *Koodi voi olla ääretön tai äärellinen. Lisäksi ko. koodi voidaan valita* prefiksikoodiksi *eli koodiksi, jonka mikään sana ei ole toisen sen sanan prefiksi.*

Todistus. Ensiksi todetaan, että pelkkä prefiksiehto takaa, että kyseessä on koodi. Ts. jos kielen mikään sana ei ole sen toisen sanan prefiksi, niin kieli on koodi. Toiseksi todetaan, että summaehdosta seuraa, että

$$(*) \qquad \sum_{n=1}^{j} M^{l_j - l_n} \leq M^{l_j} \quad (j = 1, 2, \ldots).$$

Ilmeisesti voidaan olettaa, että $l_1 \leq l_2 \leq \cdots$

Seuraava prosessi, kyllin pitkään toteutettuna, tuottaa halutun prefiksikoodin.

1. Asetetaan $L \leftarrow \emptyset$, $i \leftarrow 1$ ja

$$W_j \leftarrow \Sigma^{l_j} \quad (j = 1, 2, \ldots).$$

Tällöin W_j:ssä on M^{l_j} sanaa.

2. Valitaan W_i:stä jokin sana w_i ja asetetaan

$$W_j \leftarrow W_j - w_i\Sigma^{l_j-l_i} \quad (j = i, i+1, \dots)$$

Ennen tätä operaatiota W_j:stä on poistettu yhteensä

$$M^{l_j-l_1} + M^{l_j-l_2} + \cdots + M^{l_j-l_{i-1}}$$

sanaa. Arvion $(*)$ nojalla sen jälkeen W_i voi olla tyhjä, mutta $W_{i+1}, W_{i+2}, \dots$ eivät.

3. Asetetaan $L \leftarrow L \cup \{w_i\}$ ja $i \leftarrow i+1$ ja mennään kohtaan 2. Jos kyseessä on äärellinen määrä pituuksia, lopetetaan kun se on käyty loppuun, muuten jatketaan äärettömän monta askelta. □

Seuraus. *Jokainen koodi voidaan sananpituudet säilyttäen korvata saman aakkoston prefiksikoodilla.*

Tämä tulos on tärkeä, sillä prefiksikoodi on hyvin helposti dekoodattavissa (miten?).

Koodi on *suffiksikoodi,* jos sen peilikuva on prefiksikoodi. Yo. tulokset pätevät myös suffiksikoodeille.

Koodin sanotaan olevan *maksimaalinen,* jos se ei ole minkään toisen koodin aito alikoodi.[3] Ts. jos siihen ei voida lisätä yhtäkään sanaa ilman, että se lakkaa olemasta koodi. Ilmeisesti

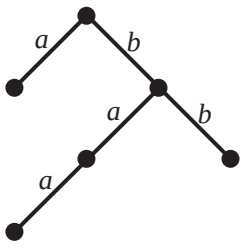
Seuraus. *Jos koodille L on* $\mathrm{is}(L) = 1$*, niin koodi on maksimaalinen.*

Käänteinen tulos ei päde yleisesti äärettömille koodeille, äärellisille kylläkin.

Prefiksikoodeilla on likeinen yhteys puurakenteisiin, vrt. jäsennyspuu Pykälässä 4.1. Puu T on ns. aakkoston Σ *prefiksipuu,* jos

- T:n oksat (viivat) on merkitty Σ:n symboleilla,
- jokaisesta pisteestä lähtevät oksat on merkitty eri symboleilla ja
- T:ssä on muitakin pisteitä kuin juuri (T voi olla ääretönkin).

Esimerkki aakkoston $\{a, b\}$ prefiksipuusta:



Prefiksipuu T määrää kielen $L(T)$, joka saadaan, kun luetaan polut juuresta lehtiin katenoiden oksien merkit. Yo. esimerkissä $L(T) = \{a, bb, baa\}$. Konstruktiosta johtuen jokaiselle prefiksipuulle T kieli $L(T)$ on prefiksikoodi. Myös käänteinen tulos pätee: Jokaiselle prefiksikoodille L on sellainen prefiksipuu T, että $L = L(T)$. Tämä puu saadaan seuraavasti:

[3] Kirjallisuudessa kutsutaan tällaista koodia toisinaan myös *täydelliseksi* ja varataan maksimaalisuus muuhun käyttöön.

1. Asetetaan T:ksi pelkkä juuripiste.
2. Lajitellaan L:n sanat leksikografiseen järjestykseen eli ensin pituuden mukaan ja saman pituuden sisällä aakkosjärjestykseen: $w_1, w_2, \ldots$ Asetetaan $i \leftarrow 1$.
3. Etsitään T:stä pisin sana u, luettuna juuresta johonkin pisteeseen, joka on w_i:n prefiksi, ts. $w_i = uv$. Jatketaan puuta T liittämällä siihen ko. pisteestä lähtevä polku, jonka oksat on merkitty järjestyksessä v:n symbolein. Alussa mainittu piste on juuri.
4. Jos kaikki L:n sanat on käyty läpi, lopetetaan ja tulostetaan T. Muutoin asetetaan $i \leftarrow i + 1$ ja mennään kohtaan 3.

7.4 Rajoitetun viipeen koodit

Ääretöntä koodia ei ilmeisestikään voi dekoodata yleistetyllä jonokoneella (GSM, ks. Pykälä 2.8). Toisaalta kaikkia äärellisiäkään koodeja ei voi sellaisella dekoodata. Esimerkiksi koodille $\{a, ab, bb\}$ sanasta ab^n ei voi dekoodata yhtään koodisanaa lukematta sitä ensin aivan loppuun asti.

Äärellinen koodi L on *viipeen p koodi,* jos aina kun

$$u_1, \ldots, u_p \in L \quad \text{ja} \quad v_1, \ldots, v_n \in L$$

ja $u_1 \cdots u_p$ on $v_1 \cdots v_n$:n prefiksi, on $u_1 = v_1$. Dekoodaukseen tarvitaan siis p koodisanan ennakointi („look-ahead"). Koodi, joka on jollekin p:lle viipeen p koodi, on ns. *rajoitetun viipeen koodi.* Huomaa, että viipeen 1 koodit ovat edellisen pykälän prefiksikoodit!

Viipeen p koodi L aakkostossa Σ voidaan dekoodata GSM:llä S seuraavalla tavalla.

1. Jos m on pisimmän L^p:n sanan pituus, niin S:n tilat ovat $\langle w \rangle$, missä w käy läpi kaikki enintään m-pituiset sanat. Alkutila on $\langle \Lambda \rangle$.
2. S:n syöte-/tulostusaakkosto on $\Sigma \cup \{\#\}$. Symboli # laitetaan syötesanan loppuun, jotta S tietää syötteen loppuneen ja tyhjentää muistinsa, ja dekoodattaessa erotetaan L:n sanat symbolilla #.
3. Jos S on tilassa $\langle w \rangle$ ja $w \notin L^p$, niin syötteellä $a \in \Sigma$ se siirtyy tilaan $\langle wa \rangle$ tulostaen Λ:n.
4. Jos S on tilassa $\langle u_1 u_2 \cdots u_p \rangle$, missä sanat $u_1, u_2, \ldots, u_p$ ovat koodisanoja, niin syötteellä $a \in \Sigma$ se siirtyy tilaan $\langle u_2 \cdots u_p a \rangle$ tulostaen $u_1\#$:n. (Jos $p = 1$, siirrytään vain tilaan $\langle a \rangle$.)
5. Jos S on tilassa $\langle u_1 \cdots u_k \rangle$, missä $k \geq 2$ ja sanat $u_1, \ldots, u_k$ ovat koodisanoja, niin syötteellä # se siirtyy tilaan $\langle \Lambda \rangle$ tulostaen $u_1\# \cdots \#u_k$:n.
6. Jos S on tilassa $\langle u \rangle$, missä $u \in L$ tai $u = \Lambda$, niin syötteellä # se siirtyy tilaan $\langle \Lambda \rangle$ tulostaen u:n.

Täydellisyyden vuoksi S:ään pitäisi vielä lisätä tilasiirrot ja tulostukset, jotka liittyvät muiden kuin L^*:n sanojen käsittelyyn.

Erityisen helposti ovat siis näin dekoodattavissa prefiksikoodit. Jos koodisanoilla on merkitystä vain niiden pituuden kautta, pyrittäessä esimerkiksi mahdollisimman lyhyisiin koodisanoihin, kannattaa käyttää prefiksikoodeja. Markov–McMillan-lauseen ja Kraftin lauseen seurauksena se on aina mahdollista.

7.5 Optimikoodit ja Huffmanin algoritmi

Optimikoodauksessa valitaan ensin aakkosto $\Sigma = \{c_1, c_2, \ldots, c_M\}$, koodisanojen lukumäärä k sekä niiden painot $P_1, P_2, \ldots, P_k$. Painot ovat ei-negatiivisia lukuja, joiden summa on $= 1$. Tarvittaessa ne voidaan tulkita todennäköisyyksiksi, frekvensseiksi tms. Jatkossa sovitaan, että

$$P_1 \geq P_2 \geq \cdots \geq P_k \ (\geq 0).$$

Ääritapauksia ovat $P_1 = 1, P_2 = \cdots = P_k = 0$ sekä $P_1 = \cdots = P_k = 1/k$.

Tehtävänä on valita koodin sanat $w_1, w_2, \ldots, w_k$, vastaten eo. painoja, siten, että koodin *keskipituus*

$$P_1|w_1| + P_2|w_2| + \cdots + P_k|w_k|$$

on pienin mahdollinen. Saatu koodi on ns. *optimikoodi.* Koska keskipituus riippuu koodisanoista vain niiden pituuden kautta, voidaan lisäksi olettaa, että saatu koodi on prefiksikoodi. Jatkossa merkitään $|w_i| = l_i$ $(i = 1, 2, \ldots, k)$ ja keskipituutta merkitään $\bar{l}$:llä. Kokonaislukuoptimointitehtäväksi puettuna optimikoodaus on silloin

$$\begin{cases} \bar{l} = \sum_{i=1}^{k} P_i l_i = \min! \\ \sum_{i=1}^{k} M^{-l_i} \leq 1 \\ l_1, l_2, \ldots, l_k \geq 1. \end{cases}$$

Optimikoodeilla on likeinen yhteys informaatioteoriaan[4] ja tiedonpakkaukseen. Niiden etsimiseksi onkin kehitetty lukuisia algoritmeja. Näistä tunnetuin lienee klassinen Huffmanin algoritmi. Käyttäen eo. merkintöjä, näytetään ensin

Apulause. *Optimaalisen prefiksikoodin sanat $w_1, w_2, \ldots, w_k$ voidaan valita siten, että*

- $l_1 \leq \cdots \leq l_{k-s} \leq l_{k-s+1} = \cdots = l_k$*, missä* $2 \leq s \leq M$ *ja* $s \equiv k \mod M - 1$,[5]
- $w_{k-s+1}, \ldots, w_k$ *eroavat toisistaan vain viimeisessä symbolissa, joka w_{k-s+i}:ssä on c_i, ja*
- *sanojen $w_{k-s+1}, \ldots, w_k$ yhteinen $l_k - 1$-pituinen prefiksi ei ole minkään sanan $w_1, \ldots, w_{k-s}$ prefiksi.*

Todistus. Jokin optimaalinen (prefiksi)koodi $\{w'_1, w'_2, \ldots, w'_k\}$ on tietenkin olemassa ja sen sanojen pituudet $l'_1, l'_2, \ldots, l'_k$. Jos nyt jollekin i:lle $l'_i > l'_{i+1}$, niin vaihdetaan w'_i ja w'_{i+1} keskenään. Koodin keskipituuden muutos on tällöin

$$P_i l'_{i+1} + P_{i+1} l'_i - (P_i l'_i + P_{i+1} l'_{i+1}) = (P_i - P_{i+1})(l'_{i+1} - l'_i) \leq 0.$$

[4]Klassisen *Shannonin koodauslauseen* mukaan $H \leq \bar{l} \leq H + 1$, missä

$$H = -P_1 \log_M P_1 - \cdots - P_k \log_M P_k$$

on ns.*entropia,* ks. kurssi Informaatioteoria.

[5]Yleisesti kongruenssi $a \equiv b \mod m$ tarkoittaa sitä, että $a - b$ on jaollinen m:llä.

Toisaalta koodi oli optimaalinen, joten ko. muutos on $= 0$ ja koodi pysyy vaihdonkin jälkeen optimaalisena (prefiksi)koodina. Toistamalla vaihto-operaatiota tarvittaessa kyllin monta kertaa voidaan olettaa, että saaduille pituuksille

$$l_1 \leq l_2 \leq \cdots \leq l_k.$$

Tällaisiakin optimikoodeja voi hyvinkin olla useita. Valitaankin jatkossa käytetty optimikoodi siten, että summa $l_1 + l_2 + \cdots + l_k$ on pienin mahdollinen, ja merkitään

$$(*) \qquad \Delta = M^{l_k} - \sum_{i=1}^{k} M^{l_k - l_i}.$$

Markov–McMillan-lauseen nojalla $\Delta \geq 0$. Jos toisaalta olisi $\Delta \geq M - 1$, niin pituudet $l_1, \ldots, l_{k-1}, l_k - 1$ toteuttaisivat Kraftin lauseen ehdon, mikä on mahdotonta, koskapa $l_1 + l_2 + \cdots + l_k$ oli pienin mahdollinen. Siispä $2 \leq M - \Delta \leq M$.

Merkitään nyt r:llä pituutta l_k olevien koodisanojen lukumäärää. Silloin $r \geq 2$. Muussa tapauksessa voitaisiin w_k:n viimeinen symboli poistaa ja saada prefiksikoodi, mikä on mahdotonta, koska $l_1 + l_2 + \cdots + l_k$ oli pienin mahdollinen. Yhtälöstä $(*)$ seuraa, että

$$\Delta \equiv -r \mod M \quad \text{eli} \quad r \equiv M - \Delta \mod M.$$

Koska toisaalta $2 \leq M - \Delta \leq M$, on r kirjoitettavissa muotoon

$$r = tM + (M - \Delta),$$

missä $t \geq 0$. Yhtälöstä $(*)$ seuraa edelleen, että

$$\Delta \equiv 1 - k \mod M - 1,$$

sillä

$$M \equiv 1 \mod M - 1 \quad \text{eli} \quad M - \Delta \equiv k \mod M - 1.$$

Näin nähdään, että $M - \Delta$ on juuri Apulauseessa mainittu luku s ja $r \geq s$.

Tarvittaessa indeksointia vaihtamalla voidaan olettaa, että koodin l_k-pituisista sanoista $w_{k-r+1}, \ldots, w_k$ sanat $w_{k-t}, \ldots, w_k$ ovat sellaisia, että niiden $l_k - 1$-pituiset prefiksit $z_1, \ldots, z_{t+1}$ ovat keskenään erilaiset. Muista, että $r = tM + s$. Korvataan nyt sanat $w_{k-r+1}, \ldots, w_k$ sanoilla

$$z_1c_1\ ,\ldots,\ z_1c_M\ ,\ z_2c_1\ ,\ldots,\ z_2c_M\ ,\ldots,\ z_tc_1\ ,\ldots,\ z_tc_M\ ,\ z_{t+1}c_1\ ,\ldots,\ z_{t+1}c_s.$$

Tämä ei hävitä koodin prefiksisyyttä eikä optimaalisuutta. Saatu koodi on vaadittu $\{w_1, w_2, \ldots, w_k\}$. □

Huffmanin algoritmi on rekursiivinen algoritmi. Rekursion etsimiseksi tarkastellaan sanoja $v_1, \ldots, v_{k-s+1}$, jotka saadaan Apulauseen koodista $\{w_1, w_2, \ldots, w_k\}$ seuraavasti:

(a) $v_1 = w_1, \ldots, v_{k-s} = w_{k-s}$ ja

(b) v_{k-s+1} saadaan, kun poistetaan w_{k-s+1}:stä sen viimeinen symboli $(= c_1)$.

Sanat $v_1, \ldots, v_{k-s+1}$ muodostavat prefiksikoodin. Ajatellaan siihen liittyviksi painoiksi vastaavasti

$$P_1, \ldots, P_{k-s}, P_{k-s+1} + \cdots + P_k.$$

Koodin keskipituus on tällöin

$$\overline{l'} = \sum_{i=1}^{k-s} P_i l_i + \sum_{i=k-s+1}^{k} P_i(l_k - 1) = \overline{l} - \sum_{i=k-s+1}^{k} P_i.$$

Koodi $\{v_1, \ldots, v_{k-s+1}\}$ on näin ollen myös optimaalinen. Muutoin olisi jokin pienemmän keskipituuden omaava prefiksikoodi $\{v'_1, \ldots, v'_{k-s+1}\}$ ja siitä saatavan prefiksikoodin

$$\{v'_1, \ldots, v'_{k-s}, v'_{k-s+1}c_1, \ldots, v'_{k-s+1}c_s\}$$

keskipituus olisi pienempi kuin

$$\overline{l'} + P_{k-s+1} + \cdots + P_k = \overline{l}.$$

Toisaalta, jos $\{v'_1, \ldots, v'_{k-s+1}\}$ on optimaalinen prefiksikoodi, vastaten mainittuja painoja

$$P_1, \ldots, P_{k-s}, P_{k-s+1} + \cdots + P_k,$$

niin sen keskipituus on $\overline{l'}$ ja

$$\{v'_1, \ldots, v'_{k-s}, v'_{k-s+1}c_1, \ldots, v'_{k-s+1}c_s\}$$

on optimaalinen prefiksikoodi, vastaten painoja $P_1, P_2, \ldots, P_k$. Muussa tapauksessa olisi Apulauseen antama optimikoodi, jonka keskipituus olisi pienempi kuin

$$\overline{l'} + P_{k-s+1} + \cdots + P_k,$$

ja tästä saataisiin vielä koodia $\{v'_1, \ldots, v'_{k-s+1}\}$:kin „optimaalisempi" koodi.

Huffmanin algoritmi on seuraava rekursio, joka yllä esitetyn mukaan tuottaa optimikoodin saatuaan syötteenä painot $P_1, P_2, \ldots, P_k$ vähenevässä järjestyksessä. Huomaa, ettei edellä mitenkään kielletty 0:nkaan esiintymistä painona.

1. Jos $k \leq M$, niin valitaan $w_1 = c_1, w_2 = c_2, \ldots, w_k = c_k$ ja lopetetaan.

2. Muutoin koodi $\{w_1, w_2, \ldots, w_k\}$ vastaten painoja $P_1, P_2, \ldots, P_k$ saadaan välittömästi, kunhan tunnetaan koodi $\{v_1, v_2, \ldots, v_{k-s+1}\}$:

   $$w_1 = v_1\ ,\ w_2 = v_2\ , \ldots,\ w_{k-s} = v_{k-s}$$

   ja

   $$w_{k-s+1} = v_{k-s+1}c_1\ , \ldots,\ w_k = v_{k-s+1}c_s.$$

3. Koodin $\{v_1, \ldots, v_{k-s+1}\}$ saamiseksi taas lasketaan s sekä asetetaan

   $$Q_1 \leftarrow P_1\ , \ldots,\ Q_{k-s} \leftarrow P_{k-s} \quad \text{ja} \quad Q_{k-s+1} \leftarrow P_{k-s+1} + \cdots + P_k.$$

   Edelleen asetetaan uusiksi todennäköisyyksien $P_1, P_2, \ldots, P_{k-s+1}$ arvoiksi $Q_1, Q_2, \ldots, Q_{k-s+1}$ vähenevässä järjestyksessä sekä $k \leftarrow k - s + 1$ ja palataan kohtaan 1.

Rekursio pysähtyy, koska annettujen painojen lukumäärää vähenee joka kierroksella ainakin yhdellä, ja on lopulta $\leq M$, jolloin prosessi pysähtyy kohtaan 1.

Huomautus. *Ensiksi valittava s toteuttaa ehdon $s \equiv k \mod M-1$. „Uusi k" on $k-s+1$ ja*

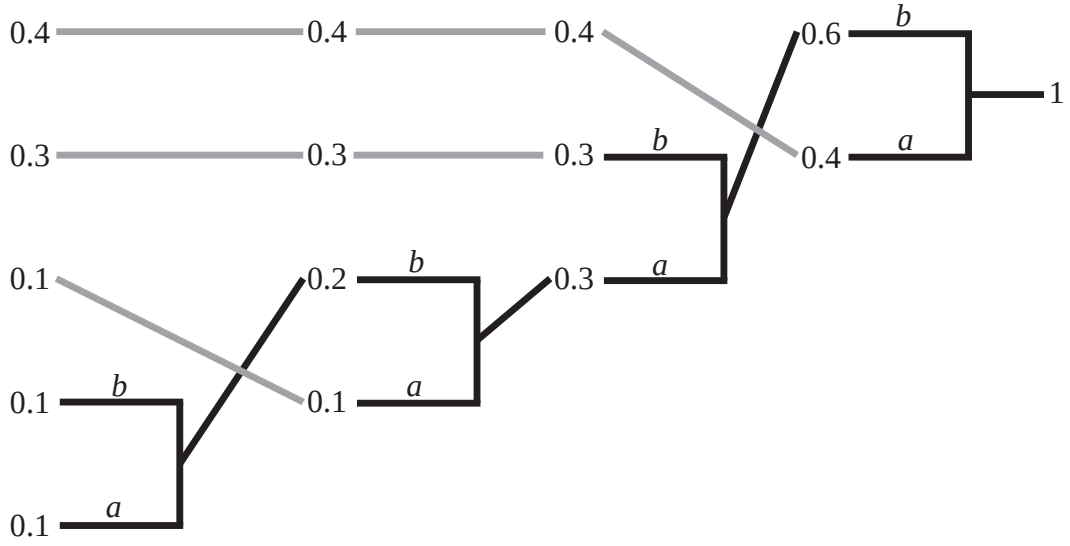
$$k - s + 1 \equiv 1 \mod M - 1.$$

Näin ollen seuraava s:n arvo onkin M. Itse asiassa näin jatkaen todetaan, että kaikki seuraavatkin uudet s:n arvot ovat $= M$ ja siis vain ensimmäinen s:n arvo pitää laskea! Huomaa myös, että tapauksessa $M = 2$ (binäärinen Huffmanin algoritmi) *on aina $s = 2$.*

Huffmanin algoritmi konstruoi oleellisesti optimikoodia vastaavan prefiksipuun, ns. *Huffmanin puun,* seuraavanlaisella „bottom-up"-menetelmällä:

- Puun oksat merkitään Σ:n symboleilla kuten edelläkin. Pisteet sen sijaan merkitään painoilla.
- Alussa on merkitty vain lehdet painoilla $P_1, P_2, \ldots, P_k$. Lehdet on myös merkitty käsittelemättömiksi.
- Kussakin vaiheessa lasketaan ensin s, joka määräytyy silloisten käsittelemättömien pisteiden lukumäärästä, ks. eo. Huomautus. Sitten lisätään puuhun uusi piste ja yhdistetään s pienintä painoa vastaavat käsittelemättömät pisteet tähän uuteen pisteeseen oksilla, jotka merkitään symboleilla $c_1, \ldots, c_s$. Samalla nämä s pistettä merkitään käsitellyiksi. Uudelle pisteelle annetaan paino, joka on ko. s pienimmän painon summa, merkitään se käsittelemättömäksi ja jatketaan prosessia.
- Menettely pysähtyy, kun pisteen painoksi tulee 1. Ko. piste on silloin puun juuri.

Tämä menettely tarjoaa „graafisen" tavan etsiä optimikoodi, ainakin pienille sanamäärille. Seuraavassa eräs tällainen „graafinen" Huffmanin puu:



Varsinaiset puun oksat on merkitty mustalla, harmaalla merkityt viivat ovat vain painojen lajittelua varten. Puu määrää aakkoston $\{a, b\}$ optimaalisen prefiksikoodin sanat $a, bb, baa, babb, baba$, vastaten painoja $0.4, 0.3, 0.1, 0.1, 0.1$.

Huffmanin algoritmin antama tulos ei ole yksikäsitteinen, johtuen esiintyvistä samoista painoista. Algoritmin haittapuolena on sen hankalahko implementoitavuus nopeaksi algoritmiksi.

Huffmanin algoritmilla voidaan ratkaista esimerkiksi ns. *kyselysysteemi.* Tehtävä on selvittää mikä mahdollisista vaihtoehdoista $V_1, \ldots, V_k$ on kyseessä käyttäen kysymyksiä,

joiden vastaus on aina jokin sovitusta M vastausvaihtoehdosta. (Usein „kyllä” tai „ei”, jos $M = 2$.) Eri vaihtoehtojen $\mathrm{V}_1,\ldots,\mathrm{V}_k$ esiintymisfrekvenssit (todennäköisyydet) $P_1, \ldots, P_k$ tiedetään. Miten kysymykset pitäisi valita, jotta niitä olisi keskimäärin vähiten?

Luku 8

LINDENMAYERIN SYSTEEMIT

8.1 Yleistä

Tähän asti käsitellyt uudelleenkirjoitusjärjestelmät ja kieliopit ovat jonollisia eli sekventiaalisia, niissä uudelleenkirjoitetaan vain yksi symboli tai osasana kerrallaan kullakin johdon askeleella. Vastaavia rinnakkaisia eli paralleeleja uudelleenkirjoitusjärjestelmiä on myös tutkittu paljon. Tunnetuimpia näistä ovat *Lindenmayerin systeemit* eli *L-systeemit.* Alunperin L-systeemit oli tarkoitettu mallintamaan kasvien tai yleisemminkin solukkojen kasvun morfologiaa. Nykyisin niitä käytetään melkeinpä pelkästään tietokonegrafiikassa kasvien malleina sekä fraktaalien generointiin eri tarkoituksiin.

Verrattuina kielioppeihin L-systeemeissä käytetään hieman poikkeavaa terminologiaa. Erityisesti mainittakoon seuraavat lyhenteet:

- 0: yhteydetön systeemi
- I: yhteydellinen systeemi
- E: käytetään loppusymboleja
- P: pituutta kasvattavat produktiot

8.2 Yhteydettömät L-systeemit

Formaalisesti *0L-systeemi* on kolmikko $G = (\Sigma, \alpha, P)$, missä Σ on (kielen) aakkosto, $\alpha \in \Sigma^*$ on ns. *aksiooma* ja P on produktioiden joukko. Erityisesti vaaditaan, että P:ssä on ainakin yksi muotoa $a \to w$ oleva produktio jokaiselle Σ:n symbolille a.

Johdot määritellään samoin kuin kieliopeille, paitsi että johdon askeleessa jokaiseen sanan symboliin on sovellettava jotain produktiota (rinnakkaisuus). Tällöin produktio voi hyvinkin olla identiteettiproduktio $a \to a$. G:n generoima kieli on

$$L(G) = \{w \mid \alpha \Rightarrow_G^* w\}.$$

0L-systeemi on

- *deterministinen* eli *D0L-systeemi,* jos jokaiselle aakkoston symbolille a on tarkalleen yksi produktio $a \to w$.
- *pituutta kasvattava* eli *propagoiva* eli *P0L-systeemi,* jos jokaisessa produktiossa $a \to w$ on $w \neq \Lambda$, erikoistapauksena PD0L-systeemi.

Vastaavia kieliperheitä merkitään $0\mathcal{L}$, $\mathcal{D}0\mathcal{L}$ jne.

Esimerkki. *Kielen* $\{a^{2^n} \mid n \geq 0\}$ *generoi yksinkertainen PD0L-systeemi* $G = \big(\{a\}, a, \{a \to a^2\}\big)$. *Huomaa, että tämä kieli ei ole CF, mutta se on CS.*

Lause 32. $0\mathcal{L} \subset \mathcal{CS}$

Todistus. Todistus on hyvin samantapainen kuin Lauseen 12. Merkitään 0L-systeemille $G = (\Sigma, \alpha, P)$

$$\Delta = \{a \in \Sigma \mid a \Rightarrow_G^* \Lambda\}.$$

Jokaiselle symbolille $a \in \Delta$ voidaan lisäksi määritellä luku

$$d_a = \min_{a \Rightarrow_G^n \Lambda} n.$$

On helppo konstruoida LBA M, joka tunnistaa $L(G)$:n. M simuloi G:n johtoja lähtien aksioomasta α. Kohdatessaan symbolin $a \in \Delta$ LBA M päättää valitseeko se jatkossa johdon, jossa $a \Rightarrow_G^* \Lambda$, vai ei. Edellisessä tapauksessa se pyyhkii a:n välittömästi nauhalta siirtäen sanan loppuosaa vasemmalle ja sen on silloin simuloitava G:n johtoa eteenpäin ainakin d_a askelta. Tämän M muistaa tiloissaan. Simulointia varten M pakkaa yhteen nauhasymboliin tarvittaessa useita Σ:n symboleja, jotta pois pyyhittävät Δ:n symbolitkin mahtuvat mukaan. Mikäli simulointi menee „leveäksi" eli ei mahdu sille varattuun tilaan, M pysähtyy ei-lopputilaan. Huomaa, että P0L-systeemin tapauksessa simulointi on erityisen helppoa.

Kieli $\{a^n b^n \mid n \geq 1\}$ on CF-kieli, joka ei ilmeisestikään ole 0L. □

Lisäämällä 0L-systeemiin $G = (\Sigma, \alpha, P)$ loppuaakkosto $\Sigma_\mathrm{T} \subseteq \Sigma$, saadaan *E0L-systeemi* $G' = (\Sigma, \Sigma_\mathrm{T}, \alpha, P)$. Tällöin generoitu kieli on

$$L(G') = \{w \mid \alpha \Rightarrow_{G'}^* w \text{ ja } w \in \Sigma_\mathrm{T}^*\}.$$

Esimerkki. *Kieli* $\{a^n b^n \mid n \geq 1\}$ *on E0L-kieli, sen generoi E0L-systeemi* $G = \big(\{a, b, c\}, \{a, b\}, c, P\big)$, *missä* P*:ssä ovat produktiot*

$$a \to a \quad , \quad b \to b \quad , \quad c \to acb \mid ab.$$

Lause 33. $\mathcal{CF} \subset \mathcal{E}0\mathcal{L} \subset \mathcal{CS}$

Todistus. CF-kieliopista on helppo tehdä vastaava E0L-systeemi, lisätään vain kaikille symboleille identiteettiproduktiot. Toisaalta on D0L-kieliä, jotka eivät ole CF (esimerkki edellä). Näin ollen $\mathcal{CF} \subset \mathcal{E}0\mathcal{L}$.

Lauseesta 32 seuraa suoraan, että $\mathcal{E}0\mathcal{L} \subseteq \mathcal{CS}$. Sen sijaan on vaikeampi löytää CS-kieltä, joka ei ole E0L. Eräs sellainen on ns. *Hermanin kieli*

$$H = \big\{w \;\big|\; |w|_a = 2^n, n = 0, 1, 2, \dots\big\}$$

aakkostossa $\{a, b\}$. Tässä $|w|_a$ tarkoittaa a-symbolien lukumäärää sanassa w. On helppo näyttää, että H on CS, mutta vaikeampi näyttää, että se ei ole E0L (sivuutetaan). □

Grafiikassa symbolit tulkitaan graafisina operaatioina, jotka toteutetaan sanan johdon antamassa järjestyksessä.

Esimerkki. *Lähtiessä kuva-alueena on neliö* $0 \leq x \leq 1$, $0 \leq y \leq 1$. *Aakkoston* $\{a, b, c\}$ *symbolit tulkitaan seuraavasti:*

- *a: pisteiden* $(1/3, 1/2)$ *ja* $(2/3, 1/2)$ *välinen jana.*
- *b: pisteiden* $(0, 0)$ *ja* $(1/3, 1/2)$ *välinen jana; skaalaus suorakulmioon* $0 \leq x \leq 1/3$, $0 \leq y \leq 1/2$.
- *c: pisteiden* $(2/3, 1/2)$ *ja* $(1, 1)$ *välinen jana; skaalaus + translaatio suorakulmioon* $2/3 \leq x \leq 1$, $1/2 \leq y \leq 1$.

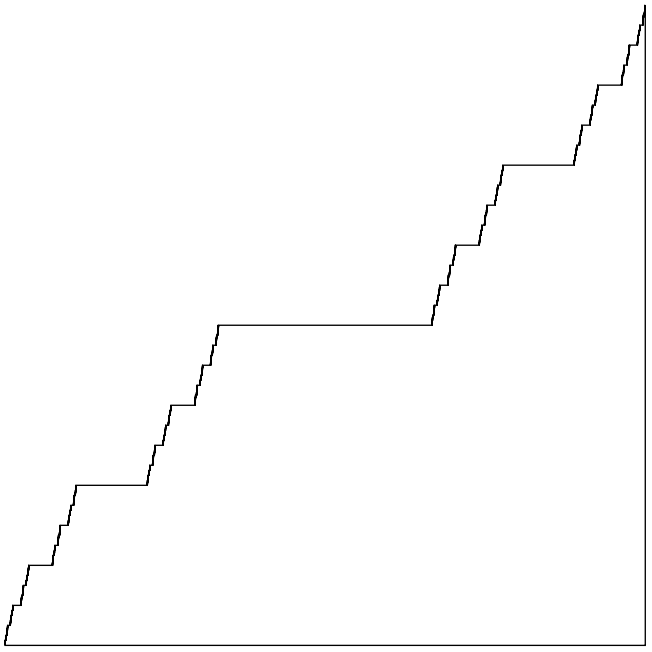
D0L-systeemi

$$G = \big(\{a, b, c\}, bac, \{a \to a, b \to bac, c \to bac\}\big)$$

generoi silloin sanat

$$bac\ ,\ bacabac\ ,\ bacabacabacabac\ ,\ldots$$

Tulkittuna graafisesti saadaan rajalla fraktaalinen kuvio, ns. pirunportaat:



Kuvaaja esittää muuten jatkuvaa funktiota, jolla on melkein kaikkialla derivaatta, mutta derivaatta on aina $= 0$.

Mallinnettaessa kasveja jms. operaatiot ovat kolmiulotteisia: rungon osa, lehti, haara, kukinto jne.

8.3 Yhteydelliset L-systeemit

IL-systeemi on viisikko $G = (\Sigma, X_{\mathrm{L}}, X_{\mathrm{R}}, \alpha, P)$, missä Σ on (kielen) aakkosto, $X_{\mathrm{L}}, X_{\mathrm{R}} \notin \Sigma$ ovat reunamerkit (vasen ja oikea), $\alpha \in \Sigma^*$ on ns. *aksiooma* ja P on produktioiden joukko. Produktiot ovat tässä erikoista tyyppiä

$$\langle u, a, v\rangle \to w,$$

missä $a \in \Sigma$ ja u on ns. *vasen ympäristö* ja v *oikea ympäristö.*

Asian tarkemmaksi selvittämiseksi merkitään d_{L}:llä (vast. d_{R}:llä) pisimmän esiintyvän vasemman ympäristön (vast. oikean ympäristön) pituutta sekä määritellään joukot

$$Y_{\mathrm{L}} = \left\{ X_{\mathrm{L}} x \;\middle|\; x \in \Sigma^* \text{ ja } |x| < d_{\mathrm{L}} \right\} \cup \Sigma^{d_{\mathrm{L}}} \quad \text{ja} \quad Y_{\mathrm{R}} = \left\{ y X_{\mathrm{R}} \;\middle|\; y \in \Sigma^* \text{ ja } |y| < d_{\mathrm{R}} \right\} \cup \Sigma^{d_{\mathrm{R}}}.$$

Nyt vaaditaan, että jokaista symbolia $a \in \Sigma$ sekä jokaista sanaa $u \in Y_{\mathrm{L}}$ ja jokaista sanaa $v \in Y_{\mathrm{R}}$ kohti on sellainen u:n suffiksi u' ja sellainen v:n prefiksi v' että jollekin w:lle $\langle u', a, v' \rangle \to w$ on P:ssä. Tämä ehto takaa, että uudelleenkirjoitus on kaikissa tilanteissa mahdollista. Mikäli mainittuja produktioita on vain yksi, on kyseessä deterministinen IL-systeemi eli *DIL-systeemi.*

Produktio $\langle u, a, v \rangle \to w$ tulkitaan siten, että sitä saa soveltaa a:n uudelleenkirjoitukseen vain siinä tilanteessa, missä a esiintyy uudelleenkirjoitettavassa sanassa osasanojen u ja v välissä tässä järjestyksessä. Kuten L-systeemeille yleensäkin, uudelleenkirjoitus on rinnakkaista, ts. sanan jokainen symboli on uudelleenkirjoitettava samanaikaisesti. Jos mitään symbolia ei uudelleenkirjoiteta Λ:ksi, ts. produktioissa $\langle u, a, v \rangle \to w$ aina $w \neq \Lambda$, niin kyseessä on ns. *PIL-systeemi.*

Generoitu kieli on

$$L(G) = \{ w \mid X_{\mathrm{L}} \alpha X_{\mathrm{R}} \Rightarrow_G^* X_{\mathrm{L}} w X_{\mathrm{R}} \}.$$

Huomaa, että reunamerkkejä ei uudelleenkirjoiteta, ne ovat vain osoittamassa sanan reunaa.

Lisäämällä IL-systeemiin $G = (\Sigma, X_{\mathrm{L}}, X_{\mathrm{R}}, \alpha, P)$ loppuaakkosto $\Sigma_{\mathrm{T}} \subseteq \Sigma$, saadaan ns. *EIL-systeemi* $G' = (\Sigma, \Sigma_{\mathrm{T}}, X_{\mathrm{L}}, X_{\mathrm{R}}, \alpha, P)$. Tällöin generoitu kieli on

$$L(G') = \{ w \mid X_{\mathrm{L}} \alpha X_{\mathrm{R}} \Rightarrow_{G'}^* X_{\mathrm{L}} w X_{\mathrm{R}} \text{ ja } w \in \Sigma_{\mathrm{T}}^* \}.$$

Lause 34. *$\mathcal{EPIL} = \mathcal{CS}$ ja $\mathcal{EIL} = \mathcal{CE}$.*

Todistus. Näiden todistus on hyvin samankaltainen kuin Lauseiden 19 ja 20. □

Koska yhteydelliset L-perheet näin yhtyvät Chomskyn hierarkiaan, ei niillä ole sellaista merkitystä kuin yhteydettömillä L-perheillä. Toisaalta IL-systeemit tarjoavat erikoisen rinnakkaisen vaihtoehdon laskentaan.

Mielenkiintoista on, että deterministisille IL-systeemeille saadaan vastaavat tulokset. Näiden todistus on hankalampi kuin Lauseen 34 (ei kuitenkaan kovin vaikea).[1]

Lause 35. *$\mathcal{EPDIL} = \mathcal{DCS}$ ja $\mathcal{EDIL} = \mathcal{CE}$.*

Näin perheille $\mathcal{DCS}$ ja $\mathcal{CE}$ saadaan myös deterministiset kielioppikarakterisaatiot. Samalla saadaan deterministinen rinnakkaislaskennan malli.

IL-systeemi on rakenteeltaan niin likellä Turingin konetta, että sitä voidaan pitää eräänä harvoista Turingin koneen rinnakkaislaskentavarianteista. Tilavaativuusmielessä tällaisesta rinnakkaisuudesta ei ole juurikaan etua, aikavaativuuden osalta asiaa ei taas tiedetä.

[1] Alkuperäisviite on väitöskirja VITÁNYI, P.M.B.: *Lindenmayer Systems: Structure, Languages, and Growth Functions.* Mathematisch Centrum. Amsterdam (1978).

Luku 9

FORMAALIT POTENSSISARJAT

9.1 Kieli formaalina potenssisarjana

Aakkoston Σ kielen L *karakteristinen funktio* on

$$\chi_L(w) = \begin{cases} 1, \text{ jos } w \in L \\ 0 \text{ muuten.} \end{cases}$$

Selvästi karakteristinen funktio χ_L määrittelee kielen L täysin. Edelleen voidaan todeta, että

$$\begin{aligned} \chi_{L_1 \cup L_2}(w) &= \max\big(\chi_{L_1}(w), \chi_{L_2}(w)\big), \\ \chi_{L_1 \cap L_2}(w) &= \min\big(\chi_{L_1}(w), \chi_{L_2}(w)\big) = \chi_{L_1}(w)\chi_{L_2}(w), \\ \chi_{L_1 L_2}(w) &= \max_{uv=w}\big(\chi_{L_1}(u)\chi_{L_2}(v)\big). \end{aligned}$$

Jos ajattelee maksimointia eräänlaisena summana, nämä operaatiot tuovat mieleen potenssisarjojen operaatiot: *summa* (kerrointen summa), *Hadamardin tulo* (kerrointen tulo) ja *Cauchyn tulo* (sarjojen tulo, konvoluutio). Niinpä onkin otettu käyttöön formaali merkintä

$$\sum_{w \in \Sigma^*} \chi_L(w) w,$$

ns. kielen L *formaali potenssisarja.* Tässä ajatellaan aakkoston $\Sigma = \{x_1, \ldots, x_k\}$ symboleita muuttujina, jotka eivät kommutoi.

9.2 Puolirenkaat

Sinällään kielen esittäminen formaalina potenssisarjana ei tuo muuta lisää kuin „tutun” merkinnän sarjamuodossa. Kielille tärkeää katenaatiosulkeumaa vastaavaa operaatiotakaan ei ollut edellä formaaleille potenssisarjoille. Formaalien potenssisarjojen anti syntyykin, kun sallitaan yleisemmät kertoimet. Yleisesti kerrointen ajatellaan tulevan tietyistä algebrallisista rakenteista, ns. puolirenkaista, joissa on määritelty yhteen- ja kertolasku ja joissa ovat voimassa tavalliset laskujen ominaisuudet. Lisäksi vaaditaan nolla-alkion ja ykkösalkion olemassaolo.

Puolirengas on algebrallinen rakenne $R = (C, +, \cdot, \mathbf{0}, \mathbf{1})$, missä C on ei-tyhjä joukko (alkiot) ja laskuoperaatioilla on halutut ominaisuudet—sen lisäksi, että operaatioiden tuloksen pitää olla yksikäsitteinen ja aina määritelty:

- Laskuoperaatiot + (*yhteenlasku*) ja · (*kertolasku*) ovat liitännäiset, ts.

$$a + (b + c) = (a + b) + c \quad \text{ja} \quad a \cdot (b \cdot c) = (a \cdot b) \cdot c.$$

  Tästä on se seuraus, että pitkät summat ja tulot voidaan suluttaa miten tahansa tuloksen muuttumatta, tai kirjoittaa kokonaan ilman sulkeita tyyliin

$$a_1 + a_2 + \cdots + a_n \quad \text{ja} \quad a_1 \cdot a_2 \cdot \cdots \cdot a_n.$$

  Edelleen voidaan ottaa käyttöön monikerta- ja potenssimerkinnät

$$na = \underbrace{a + a + \cdots + a}_{n \text{ kpl}} \quad \text{ja} \quad a^n = \underbrace{a \cdot a \cdot \cdots \cdot a}_{n \text{ kpl}}$$

  ja erikoisesti $1a = a$ sekä $a^1 = a$. Näille pätevät tavanomaiset laskusäännöt:

$$(n + m)a = (na) + (ma) \quad \text{ja} \quad a^{n+m} = a^n \cdot a^m.$$

- Yhteenlasku on vaihdannainen, ts. $a + b = b + a$.

  Usein myös kertolasku on vaihdannainen, ts. $a \cdot b = b \cdot a$. Tällöin puhutaan *vaihdannaisesta puolirenkaasta.*

- Kertolasku on osittuva yhteenlaskun suhteen, ts.

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c) \quad \text{ja} \quad (a + b) \cdot c = (a \cdot c) + (b \cdot c).$$

- $\mathbf{0} \in C$ on ns. *nolla-alkio,* jolle $a + \mathbf{0} = \mathbf{0} + a = a$. Monikertamerkinnässä vastaavasti $0a = \mathbf{0}$. Huomaa, että nolla-alkioita on vain yksi (totea!).

- $\mathbf{1} \in C$ on ns. *ykkösalkio,* jolle $\mathbf{1} \cdot a = a \cdot \mathbf{1} = a$. Potenssimerkinnässä vastaavasti $a^0 = \mathbf{1}$, myös $\mathbf{0}^0 = \mathbf{1}$. Lisäksi oletetaan, että $\mathbf{0} \neq \mathbf{1}$. (Puolirenkaassa on siis aina vähintään kaksi alkiota.) Huomaa, että ykkösalkioita on vain yksi (totea!).

- $\mathbf{0} \cdot a = a \cdot \mathbf{0} = \mathbf{0}$

Sulkeita voidaan jättää pois sopimalla, että kertolaskut suoritetaan ennen yhteenlaskuja. Kertolaskupiste jätetään paljolti merkitsemättä, kuten tavallista.

Tuttuja esimerkkejä puolirenkaista ovat $N = (\mathbb{N}, +, \cdot, 0, 1)$ (luonnolliset luvut tavallisin laskuoperaatioin) sekä $(\mathbb{R}_+, +, \cdot, 0, 1)$ (ei-negatiiviset reaaliluvut tavallisin laskuoperaatioin). Luonnollisesti myös kaikki kokonaisluvut $\mathbb{Z}$ ja kaikki reaaliluvut $\mathbb{R}$ tavallisin laskuoperaatioin muodostavat puolirenkaat. Kielen formaalissa potenssisarjassa kertoimet tulevat myös puolirenkaasta, ns. *Boolen puolirenkaasta* $B = (\mathbb{B}, \max, \min, 0, 1)$, jossa $\mathbb{B} = \{0, 1\}$ (bitit) ja jossa siis ei ole muita alkioita kuin nolla- ja ykkösalkio. Liitännäisyys ja osittuvuus ovat helppoja todeta. Kaikki nämä puolirenkaat ovat vaihdannaisia. Esimerkki ei-vaihdannaisesta puolirenkaasta on vaikkapa luonnollisten lukujen muodostamien $n \times n$-matriisien puolirengas $(\mathbb{N}^{n \times n}, +, \cdot, \mathbf{O}_n, \mathbf{I}_n)$, missä yhteen- ja kertolasku ovat tavanomaiset matriisioperaatiot, $\mathbf{O}_n$ on nollamatriisi ja $\mathbf{I}_n$ on identiteettimatriisi, ja vielä $n > 1$.

9.3 Yleinen formaali potenssisarja

Samaan tapaan määriteltynä kuin kielen formaali potenssisarja yleinen aakkoston $\Sigma = \{x_1, x_2, \ldots, x_k\}$ *formaali potenssisarja puolirenkaassa* $R = (C, +, \cdot, \mathbf{0}, \mathbf{1})$ on kuvaus $\kappa : \Sigma^* \longrightarrow C$. Sitä merkitään perinteisesti äärettömänä formaalina summana

$$\sum_{w \in \Sigma^*} \kappa(w)w,$$

jonka yhteenlaskettavia kutsutaan *termeiksi.* Edelleen tavalliseen tapaan termit $\kappa(w)w$, missä $\kappa(w) = \mathbf{0}$, voidaan jättää summasta pois. Tietyille termeille käytetään tavanomaisia lyhennysmerkintöjä:

$$\kappa(\Lambda)\Lambda = \kappa(\Lambda) \quad \text{ja} \quad \mathbf{1}w = w.$$

Termissä $\kappa(w)w$ oleva $\kappa(w)$ on sen *kerroin.* Kerrointen ympäriltä jätetään yleensä sulkeet pois, jos se ei aiheuta epäselvyyttä. Kaikkien tällaisten formaalien potenssisarjojen joukkoa merkitään $R\langle\langle\Sigma\rangle\rangle$:lla.

Puolirenkaan operaatioista + ja $\cdot$ saadaan formaaleille potenssisarjoille

$$S_1 = \sum_{w \in \Sigma^*} \kappa_1(w)w \quad \text{ja} \quad S_2 = \sum_{w \in \Sigma^*} \kappa_2(w)w$$

laskuoperaatiot, joista jo oli puhetta:

- $S_1 + S_2 = \sum_{w \in \Sigma^*} \big(\kappa_1(w) + \kappa_2(w)\big)w$ (*summa*).
- $S_1 \otimes S_2 = \sum_{w \in \Sigma^*} \kappa_1(w)\kappa_2(w)w$ (*Hadamardin tulo*).
- $S_1S_2 = \sum_{w \in \Sigma^*} \kappa(w)w$ (*Cauchyn tulo*), missä $\kappa(w) = \sum_{uv=w} \kappa_1(u)\kappa_2(v)$.

Cauchyn tuloa käyttäen saadaan edelleen sarjan

$$S = \sum_{w \in \Sigma^*} \kappa(w)w$$

Cauchyn potenssi $S^m = \sum_{w \in \Sigma^*} \kappa'(w)w$, missä

$$\kappa'(w) = \sum_{u_1u_2\cdots u_m = w} \kappa(u_1)\kappa(u_2)\cdots\kappa(u_m).$$

Huomaa erityisesti, että sarjan ensimmäinen potenssi S^1 on sama kuin sarja itse.

Formaalia potenssisarjaa, jossa vain äärellisen monta kerrointa on $\neq \mathbf{0}$, sanotaan *formaaliksi polynomiksi.* Äärellisten kielten formaalit potenssisarjat ovat juuri Boolen renkaan B formaalit polynomit. Formaalit polynomit, joissa on vain yksi termi, ts. jotka ovat muotoa aw jollekin alkiolle $a \in C$, $a \neq \mathbf{0}$, ja sanalle $w \in \Sigma^*$, ovat ns. *monomeja.* Edelleen formaalit sarjat, joissa vain $\kappa(\Lambda)$ on $\neq \mathbf{0}$, ovat *vakiosarjoja* ja samaistettavissa puolirenkaan alkioihin. Vakiosarjalle a Cauchyn tulo on yksinkertainen:

$$a\Big(\sum_{w \in \Sigma^*} \kappa(w)w\Big) = \sum_{w \in \Sigma^*} a\kappa(w)w.$$

Sarjan S nollanneksi potenssiksi S^0 sovitaan vakiosarja $\mathbf{1}$. Sarja, jossa kaikki kertoimet ovat $= \mathbf{0}$, on ns. *nollasarja* $\mathbf{0}$.

Sellaiset sarjat, joissa $\kappa(\Lambda) = \mathbf{0}$, ovat puolestaan ns. *kvasisäännöllisiä* sarjoja. Kvasisäännöllisellä formaalilla sarjalla

$$S = \sum_{w\in\Sigma^*} \kappa(w)w = \sum_{w\in\Sigma^+} \kappa(w)w$$

on ns. *kvasi-inverssit*

$$S^+ = \sum_{w\in\Sigma^+} \kappa^+(w)w \quad \text{ja} \quad S^* = \mathbf{1} + S^+$$

(huomaa, että $\kappa^+(\Lambda) = \mathbf{0}$), missä

$$\kappa^+(w) = \sum_{m=1}^{|w|} \sum_{u_1u_2\cdots u_m=w} \kappa(u_1)\kappa(u_2)\cdots\kappa(u_m).$$

Kvasi-inverssin perusominaisuus on

Lause 36. *Formaalin potenssisarjan S kvasi-inverssit toteuttavat yhtälöt*

$$S + SS^+ = S + S^+S = SS^* = S^*S = S^+.$$

Todistus. Todistetaan malliksi yhtälö $S + SS^+ = S^+$. Merkitään

$$SS^+ = \sum_{w\in\Sigma^*} \kappa'(w)w.$$

Silloin eo. määritelmien nojalla

$$\begin{aligned}\kappa'(w) &= \sum_{uv=w} \kappa(u) \sum_{m=1}^{|v|} \sum_{u_1\cdots u_m=v} \kappa(u_1)\cdots\kappa(u_m) = \sum_{uv=w} \sum_{m=1}^{|v|} \sum_{u_1\cdots u_m=v} \kappa(u)\kappa(u_1)\cdots\kappa(u_m) \\ &= \sum_{k=2}^{|w|} \sum_{u_1u_2\cdots u_k=w} \kappa(u_1)\kappa(u_2)\cdots\kappa(u_k).\end{aligned}$$

Viimeisen yhtäläisyyden näet käymällä läpi w:n eri suffiksit v. Näin ollen $\kappa(w) + \kappa'(w) = \kappa^+(w)$. □

Kielelle L, jossa ei ole tyhjää sanaa, sen formaali potenssisarja S on kvasisäännöllinen ja S^+ (vast. S^*) on L^+:n (vast. L^*:n) formaali potenssisarja. Kvasi-inversio on siis katenaatiosulkeumaa vastaava operaatio formaaleille potenssisarjoille.

Vaikkakaan puolirenkaassa alkioilla ei välttämättä ole vasta-alkioita, on toisinaan tapana kirjoittaa kvasi-inverssit muodossa

$$S^+ = \frac{S}{\mathbf{1} - S} \quad \text{ja} \quad S^* = \frac{\mathbf{1}}{\mathbf{1} - S}.$$

Vertaamalla yo. Cauchyn potenssiin voidaan vielä todeta, että

$$S^+ = \sum_{m=1}^{\infty} S^m \quad \text{ja} \quad S^* = \sum_{m=0}^{\infty} S^m.$$

Esimerkki. *Peruskursseilta tutut yhden muuttujan x potenssisarjat ovat ajateltavissa myös formaaleiksi potenssisarjoiksi puolirenkaassa $(\mathbb{Q}, +, \cdot, 0, 1)$ (rationaaliluvut) tai $(\mathbb{R}, +, \cdot, 0, 1)$ (reaaliluvut). Niinpä esimerkiksi funktion xe^x Maclaurinin sarja*

$$S = \sum_{m=1}^{\infty} \frac{1}{(m-1)!} x^m$$

on tulkittavissa tällaiseksi formaaliksi sarjaksi aakkostossa $\{x\}$. Sen kvasi-inverssi S^+ (vast. S^*) on funktion $xe^x/(1-xe^x)$ (vast. $1/(1-xe^x)$) Maclaurinin sarja, kuten voisi odottaa.

Huomaa myös, että $(S^+)^+ \neq S^+$. Kielillehän $(L^+)^+ = L^+$, joten kaikki kielten laskulait eivät siirry formaaleille potenssisarjoille!

Summa, Cauchyn tulo ja kvasi-inversio ovat formaalien potenssisarjojen ns. *rationaaliset operaatiot.* Ne aakkoston Σ formaalit potenssisarjat puolirenkaassa R, jotka saadaan rationaalisilla operaatioilla lähtien liikkeelle vakioista ja monomeista x_l, ovat ns. *R-rationaaliset potenssisarjat.* Aakkoston Σ kaikkien R-rationaalisten potenssisarjojen joukkoa merkitään $R_{\mathrm{rat}}\langle\langle\Sigma\rangle\rangle$:lla. Ottaen huomioon mainitut yhteydet kielten operaatioihin ja säännöllisten kielten määrittelyn säännöllisten lausekkeiden avulla nähdään

Lause 37. *Aakkoston Σ säännöllisten kielten formaalit potenssisarjat muodostavat tarkalleen joukon $B_{\mathrm{rat}}\langle\langle\Sigma\rangle\rangle$.*

Todistus. Ajatellen säännöllisten kielten määrittelyä säännöllisten lausekkeiden avulla Pykälässä 2.1 ja kvasi-inversiota pitää vain näyttää, että lausekkeissa voidaan aina muodostaa katenaatiosulkeuma muodossa $r^* = \Lambda + r_1^+$, missä r_1:tä vastaavassa kielessä ei ole tyhjää sanaa. Tämä seuraa suoraan laskusäännöistä, joilla tyhjä sana „pidetään erillään”:

$$\begin{aligned}
r_1 + (\Lambda + r_2) &= \Lambda + (r_1 + r_2)\ , \\
(\Lambda + r_1) + (\Lambda + r_2) &= \Lambda + (r_1 + r_2)\ , \\
r_1(\Lambda + r_2) &= r_1 + r_1 r_2\ , \\
(\Lambda + r_1) r_2 &= r_2 + r_1 r_2\ , \\
(\Lambda + r_1)(\Lambda + r_2) &= \Lambda + (r_1 + r_2 + r_1 r_2)\ , \\
(\Lambda + r)^* &= r^* = \Lambda + r^+.
\end{aligned}$$

Soveltaen näitä saadaan jokainen säännöllinen lauseke esitetyksi ekvivalentissa muodossa $\Lambda + r$ tai r, missä r:n esittämässä kielessä ei ole tyhjää sanaa. (Muista, että $\Lambda^* = \emptyset^* = \Lambda$.) □

Rationaaliset formaalit potenssisarjat ovat näin säännöllisten kielten vastine.[1]

Huomattakoon vielä, että jokainen $R\langle\langle\Sigma\rangle\rangle$:n formaali potenssisarja

$$S = \sum_{w \in \Sigma^*} \kappa(w) w$$

määrittää kielen

$$L(S) = \{ w \mid \kappa(w) \neq \mathbf{0} \},$$

[1] CF-kielten vastine, ns. *algebralliset formaalit potenssisarjat,* saataisiin ottamalla mukaan algebralliset operaatiot (polynomiyhtälön ratkaisut).

ns. S:n *tukikielen.* Nollasarjalle (kaikki kertoimet $= \mathbf{0}$) tukikieli on tyhjä ja polynomeille äärellinen.

Melko helposti on todettavissa, että $\big(R\langle\langle\Sigma\rangle\rangle, +, \cdot, \mathbf{0}, \mathbf{1}\big)$, missä $\cdot$ on Cauchyn tulo, on sekin puolirengas—ja sitä voitaisiin periaatteessa käyttää formaalin potenssisarjan kerroinpuolirenkaana!

9.4 Tunnistuvat formaalit potenssisarjat. Schützenberger'n esityslause

Säännöllisille kielille määrittely yhtäältä säännöllisten lausekkeiden avulla ja toisaalta äärellisten automaattien avulla muodostavat yhdessä vahvan koneiston, jolla voidaan todistaa kielten ominaisuuksia. Edellä aakkoston Σ säännöllisten kielten perheen vastineeksi todettiin $R_{\mathrm{rat}}\langle\langle\Sigma\rangle\rangle$ ja se määriteltiin rationaalisten operaatioiden kautta. Tämä vastaa säännöllisten lausekkeiden käyttöä. Automaattikarakterisaatio on myös olemassa, ns. tunnistuvuus.

Jotta päästään tunnistuvuuteen käsiksi, esitetään ensin malliksi epädeterminististen äärellinen automaatti (ilman Λ-siirtoja) $M = (Q, \Sigma, S, \delta, A)$ Boolen puolirenkaan B avulla. Automaatin tilat ovat $Q = \{q_1, \ldots, q_m\}$. Kutakin symbolia $x_l \in \Sigma$ asetetaan vastaamaan bittimatriisi $\mathbf{D}_l = (d_{ij}^{(l)})$, missä

$$d_{ij}^{(l)} = \begin{cases} 1, \text{ jos } q_j \in \delta(q_i, x_l) \\ 0 \text{ muuten.} \end{cases}$$

Edelleen alku- ja lopputilajoukkoja vastaamaan asetetaan m-vektorit (vaakavektorit) $\mathbf{s} = (s_1, \ldots, s_m)$ ja $\mathbf{a} = (a_1, \ldots, a_m)$, missä

$$s_i = \begin{cases} 1, \text{ jos } q_i \in S \\ 0 \text{ muuten} \end{cases} \quad \text{ja} \quad a_i = \begin{cases} 1, \text{ jos } q_i \in A \\ 0 \text{ muuten.} \end{cases}$$

Silloin ne tilat, joihin on päästävissä jostain alkutilasta, kun luetaan syötesymboli x_l, vastaavat 1-alkioita vektorissa $\mathbf{s}\mathbf{D}_l$. Yleisesti ne tilat, joihin on päästävissä alkutilasta syötteellä $w = x_{l_1}x_{l_2}\cdots x_{l_k}$, eli $\hat{\delta}^*(S, w)$, vastaavat 1-alkioita vektorissa

$$\mathbf{s}\mathbf{D}_{l_1}\mathbf{D}_{l_2}\cdots\mathbf{D}_{l_k}.$$

Merkitään tällöin lyhyyden vuoksi

$$\boldsymbol{\mu}(w) = \mathbf{D}_{l_1}\mathbf{D}_{l_2}\cdots\mathbf{D}_{l_k}$$

ja erikoisesti $\boldsymbol{\mu}(x_l) = \mathbf{D}_l$ sekä $\boldsymbol{\mu}(\Lambda) = \mathbf{I}_m$ (identiteettimatriisi, jonka alkiot ovat $\mathbb{B}$:ssä). Muista, että laskutoimitukset tehdään Boolen puolirenkaassa B. Matriisitulon liitännäisyys seuraa tavalliseen tapaan B:n operaatioiden liitännäisyydestä ja osittuvuudesta.

Syöte w hyväksytään nyt tarkalleen silloin, kun joukossa $\hat{\delta}^*(S, w)$ on lopputila, eli silloin kun

$$\mathbf{s}\boldsymbol{\mu}(w)\mathbf{a}^{\mathsf{T}} = 1.$$

Erikoisesti tyhjä sana Λ hyväksytään tarkalleen silloin, kun $\mathbf{s}\mathbf{a}^{\mathsf{T}} = 1$.

Kielen $L(M)$ formaali potenssisarja on siis

$$\sum_{w\in\Sigma^*} \mathbf{s}\boldsymbol{\mu}(w)\mathbf{a}^{\mathsf{T}} w.$$

Tällaisen formaalin potenssisarjan sanotaan olevan *B-tunnistuva.*

Yleisen aakkoston $\Sigma = \{x_1, \ldots, x_k\}$ tunnistuvan formaalin potenssisarjan määritelmä puolirenkaassa $R = (C, +, \cdot, \mathbf{0}, \mathbf{1})$ on vastaavanlainen. Formaali potenssisarja

$$S = \sum_{w \in \Sigma^*} \kappa(w) w$$

on *R-tunnistuva,* jos on olemassa sellainen luku $m \geq 1$ ja sellaiset R:n $m \times m$-matriisit $\mathbf{D}_1, \ldots, \mathbf{D}_k$ ja m-vaakavektorit $\mathbf{s}$ ja $\mathbf{a}$, että

$$\kappa(w) = \mathbf{s}\boldsymbol{\mu}(w)\mathbf{a}^{\mathsf{T}},$$

ns. *matriisiesitys*[2], missä

- $\boldsymbol{\mu}(w)$ on R:n $m \times m$-matriisi,
- $\boldsymbol{\mu}(\Lambda) = \mathbf{I}_m$ (alkioista $\mathbf{0}$ ja $\mathbf{1}$ muodostettu $m \times m$-identiteettimatriisi),
- $\boldsymbol{\mu}(x_l) = \mathbf{D}_l$ $(l = 1, \ldots, k)$ ja
- $\boldsymbol{\mu}$ toteuttaa ehdon
$$\boldsymbol{\mu}(uv) = \boldsymbol{\mu}(u)\boldsymbol{\mu}(v) \quad (u, v \in \Sigma^*).$$
Jälleen matriisitulon liitännäisyys seuraa R:n operaatioiden liitännäisyydesta ja osittuvuudesta.

Matriisiesitys on eräänlainen automaatti, joka tunnistaa sarjan. R-tunnistuvien potenssisarjojen joukkoa aakkostossa Σ merkitään $R_{\mathrm{rec}}\langle\langle\Sigma\rangle\rangle$:lla.

Edellä määritellyt sarjojen operaatiot säilyttävät tunnistuvuuden. Aloitetaan rationaalisista operaatioista:

Lause 38. (i) *Jos formaalit sarjat S_1 ja S_2 ovat R-tunnistuvia, niin samoin ovat $S_1 + S_2$ ja $S_1 S_2$.*

(ii) *Jos kvasisäännöllinen formaali potenssisarja S on R-tunnistuva, niin samoin ovat S^+ ja S^*.*

Todistus. Todistukset ovat paljolti samankaltaiset kuin Lauseen 2 ja Kleenen lauseen.

(i) Jos sarjat S_1 ja S_2 ovat tunnistuvia, niin niillä on matriisiesitykset

$$\kappa_1(w) = \mathbf{s}_1\boldsymbol{\mu}_1(w)\mathbf{a}_1^{\mathsf{T}} \quad \text{ja} \quad \kappa_2(w) = \mathbf{s}_2\boldsymbol{\mu}_2(w)\mathbf{a}_2^{\mathsf{T}},$$

vastaavasti. Summan $S_1 + S_2$ matriisiesitys

$$\gamma(w) = \mathbf{t}\boldsymbol{\nu}(w)\mathbf{b}^{\mathsf{T}}$$

saadaan, kun valitaan

$$\mathbf{t} = \left(\mathbf{s}_1 \,\middle|\, \mathbf{s}_2\right) \quad \text{ja} \quad \mathbf{b} = \left(\mathbf{a}_1 \,\middle|\, \mathbf{a}_2\right)$$

sekä

$$\boldsymbol{\nu}(x_l) = \left(\begin{array}{c|c} \boldsymbol{\mu}_1(x_l) & \mathbf{O} \\ \hline \mathbf{O} & \boldsymbol{\mu}_2(x_l) \end{array}\right) \quad (l = 1, \ldots, k)$$

[2]Huomaa, että $\mathbf{s}\boldsymbol{\mu}(w)\mathbf{a}^{\mathsf{T}}$ on matriisin $\boldsymbol{\mu}(w)$ alkioista laskettu lineaariyhdelmä kiintein vektoreista $\mathbf{s}$ ja $\mathbf{a}$ saatavin kertoimin. Toisinaan matriisiesitys määritelläänkin niin, että se on vain jokin kiintein kertoimin matriisin alkioista laskettu lineaariyhdelmä. Tämä johtaa samaan tunnistuvuuden käsitteeseen.

(lohkomuodossa). Tässä **O**:t ovat sopivan kokoisia nollamatriiseja, jotka muodostetaan R:n nolla-alkiota käyttäen. Silloin nimittäin

$$\begin{aligned}\boldsymbol{\gamma}(w) &= \left(\mathbf{s}_1 \,\middle|\, \mathbf{s}_2\right) \left(\begin{array}{c|c} \boldsymbol{\mu}_1(w) & \mathbf{O} \\ \hline \mathbf{O} & \boldsymbol{\mu}_2(w) \end{array}\right) \left(\begin{array}{c} \mathbf{a}_1^{\mathsf{T}} \\ \hline \mathbf{a}_2^{\mathsf{T}} \end{array}\right) = \left(\mathbf{s}_1 \,\middle|\, \mathbf{s}_2\right) \left(\begin{array}{c} \boldsymbol{\mu}_1(w)\mathbf{a}_1^{\mathsf{T}} \\ \hline \boldsymbol{\mu}_2(w)\mathbf{a}_2^{\mathsf{T}} \end{array}\right) \\ &= \kappa_1(w) + \kappa_2(w).\end{aligned}$$

Cauchyn tulo S_1S_2 on vähän mutkikkaampi. Merkitään $\mathbf{C} = \mathbf{a}_1^{\mathsf{T}}\mathbf{s}_2$. Nyt valitaankin

$$\mathbf{t} = \left(\mathbf{s}_1 \,\middle|\, \mathbf{0}\right) \quad \text{ja} \quad \mathbf{b} = \left(\mathbf{a}_2\mathbf{C}^{\mathsf{T}} \,\middle|\, \mathbf{a}_2\right),$$

missä $\mathbf{0}$ on $\mathbf{a}_2$:n pituinen nollavektori, sekä

$$\boldsymbol{\nu}(x_l) = \left(\begin{array}{c|c} \boldsymbol{\mu}_1(x_l) & \mathbf{C}\boldsymbol{\mu}_2(x_l) \\ \hline \mathbf{O} & \boldsymbol{\mu}_2(x_l) \end{array}\right) \quad (l = 1, \ldots, k).$$

Silloin

$$\boldsymbol{\nu}(w) = \left(\begin{array}{c|c} \boldsymbol{\mu}_1(w) & \boldsymbol{\eta}(w) \\ \hline \mathbf{O} & \boldsymbol{\mu}_2(w) \end{array}\right),$$

missä

$$\boldsymbol{\eta}(w) = \sum_{\substack{uv=w \\ v\neq\Lambda}} \boldsymbol{\mu}_1(u)\mathbf{C}\boldsymbol{\mu}_2(v).$$

Tämä todetaan induktiivisesti. Tulos pitää paikkansa, kun $w = \Lambda$ (summa on tyhjä ja $\boldsymbol{\eta}(\Lambda) = \mathbf{O}$). Toisaalta matriisin $\boldsymbol{\nu}(wx_l)$ oikea ylälohko on

$$\begin{aligned}\boldsymbol{\mu}_1(w)\mathbf{C}\boldsymbol{\mu}_2(x_l) + \left(\sum_{\substack{uv=w \\ v\neq\Lambda}} \boldsymbol{\mu}_1(u)\mathbf{C}\boldsymbol{\mu}_2(v)\right)\boldsymbol{\mu}_2(x_l) &= \boldsymbol{\mu}_1(w)\mathbf{C}\boldsymbol{\mu}_2(x_l) + \sum_{\substack{uv=w \\ v\neq\Lambda}} \boldsymbol{\mu}_1(u)\mathbf{C}\boldsymbol{\mu}_2(vx_l) \\ &= \sum_{\substack{uv=wx_l \\ v\neq\Lambda}} \boldsymbol{\mu}_1(u)\mathbf{C}\boldsymbol{\mu}_2(v).\end{aligned}$$

Matriisiesitys

$$\begin{aligned}\mathbf{t}\boldsymbol{\nu}(w)\mathbf{b}^{\mathsf{T}} &= \mathbf{s}_1\boldsymbol{\mu}_1(w)\mathbf{C}\mathbf{a}_2^{\mathsf{T}} + \sum_{\substack{uv=w \\ v\neq\Lambda}} \mathbf{s}_1\boldsymbol{\mu}_1(u)\mathbf{C}\boldsymbol{\mu}_2(v)\mathbf{a}_2^{\mathsf{T}} = \sum_{uv=w} \mathbf{s}_1\boldsymbol{\mu}_1(u)\mathbf{a}_1^{\mathsf{T}}\mathbf{s}_2\boldsymbol{\mu}_2(v)\mathbf{a}_2^{\mathsf{T}} \\ &= \sum_{uv=w} \kappa_1(u)\kappa_2(v)\end{aligned}$$

on silloin Cauchyn tulon matriisiesitys.

(ii) Jos kvasisäännöllinen formaali potenssisarja S on tunnistuva, niin sillä on matriisiesitys

$$\kappa(w) = \mathbf{s}\boldsymbol{\mu}(w)\mathbf{a}^{\mathsf{T}},$$

missä $\mathbf{s}\mathbf{a}^{\mathsf{T}} = \kappa(\Lambda) = \mathbf{0}$. Merkitään jälleen $\mathbf{C} = \mathbf{a}^{\mathsf{T}}\mathbf{s}$. Silloin

$$\mathbf{s}\mathbf{C} = \mathbf{0}.$$

Sarjan S^+ matriisiesitys saadaan nyt muodossa

$$\mathbf{s}\boldsymbol{\mu}^+(w)\mathbf{a}^{\mathsf{T}},$$

missä

$$\boldsymbol{\mu}^+(x_l) = \boldsymbol{\mu}(x_l) + \mathbf{C}\boldsymbol{\mu}(x_l) \quad (l = 1, \ldots, k).$$

Esitys on ilmeisesti oikea, kun $w = \Lambda$. Jos taas $w = x_{l_1}x_{l_2}\cdots x_{l_n}$, niin yo. yhtälön nojalla aukikertoen

$$\begin{aligned}\mathbf{s}\boldsymbol{\mu}^+(w)\mathbf{a}^\mathsf{T} &= \mathbf{s}\big(\boldsymbol{\mu}(x_{l_1}) + \mathbf{C}\boldsymbol{\mu}(x_{l_1})\big)\big(\boldsymbol{\mu}(x_{l_2}) + \mathbf{C}\boldsymbol{\mu}(x_{l_2})\big)\cdots\big(\boldsymbol{\mu}(x_{l_n}) + \mathbf{C}\boldsymbol{\mu}(x_{l_n})\big)\mathbf{a}^\mathsf{T} \\ &= \sum_{m=1}^{|w|} \sum_{u_1u_2\cdots u_m = w} \mathbf{s}\boldsymbol{\mu}(u_1)\mathbf{C}\boldsymbol{\mu}(u_2)\mathbf{C}\cdots\mathbf{C}\boldsymbol{\mu}(u_m)\mathbf{a}^\mathsf{T}.\end{aligned}$$

Aukikerrotussa tulossa nimittäin ne termit, joissa on $\mathbf{sC}$, ovat nolla-alkioita ja ne voidaan jättää pois. Jäljelle jäävät termit ovat juuri ne, jotka esiintyvät summassa. Näin ollen

$$\mathbf{s}\boldsymbol{\mu}^+(w)\mathbf{a}^\mathsf{T} = \sum_{m=1}^{|w|} \sum_{u_1u_2\cdots u_m = w} \kappa(u_1)\kappa(u_2)\cdots\kappa(u_m) = \kappa^+(w)$$

ja kvasi-inverssille S^+ on saatu matriisiesitys.

Vakiosarja $\mathbf{1}$ on tunnistuva (ks. alla), joten samoin on kvasi-inverssi $S^* = \mathbf{1} + S^+$. □

Välittömänä seurauksena saadaan

Seuraus. *R-rationaaliset formaalit potenssisarjat ovat R-tunnistuvia.*

Todistus. Edellisen lauseen ja rationaalisuuden määritelmän nojalla riittää osoittaa, että vakiosarjat ja monomit x_l ovat tunnistuvia. Vakiosarjojen osalta tämä on helppoa, vakion a matriisiesitys on

$$\kappa(w) = a\mu(w)\mathbf{1},$$

missä $\mu(x_l) = \mathbf{0}$ $(l = 1, \ldots, k)$. Muista, että $\mathbf{0}^0 = \mathbf{1}$. Monomin x_l matriisiesitys taas on

$$\kappa(w) = \begin{pmatrix}\mathbf{1} & \mathbf{0}\end{pmatrix}\boldsymbol{\mu}(w)\begin{pmatrix}\mathbf{0}\\ \mathbf{1}\end{pmatrix},$$

missä $\boldsymbol{\mu}(x_i)$ on 2×2-nollamatriisi, kun $i \neq l$, ja

$$\boldsymbol{\mu}(x_l) = \begin{pmatrix}\mathbf{0} & \mathbf{1}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}.$$ □

Käänteinenkin tulos pitää paikkansa, ts. R-tunnistuvat formaalit potenssisarjat ovat R-rationaalisia. Tämän todistus muistuttaa Lauseen 3 todistusta ja perustuu siihen, että puolirenkaan R:n alkioista muodostetut $m \times m$-matriisit muodostavat nekin puolirenkaan $R^{m\times m}$, kun laskuoperaatioina ovat matriisien summa ja tulo, nolla-alkiona ja ykkösalkiona toimivat R:n nolla-alkiosta $\mathbf{0}$ ja ykkösalkiosta $\mathbf{1}$ muodostetut nollamatriisi $\mathbf{O}_m$ ja identiteettimatriisi $\mathbf{I}_m$.

$R^{m\times m}\langle\langle\Sigma\rangle\rangle$:n formaalit potenssisarjat voidaan tulkita joko $m \times m$-matriisikertoimisiksi potenssisarjoiksi (tavallinen tulkinta), mutta myös $m \times m$-matriiseiksi, joiden alkiot ovat $R\langle\langle\Sigma\rangle\rangle$:n formaaleja potenssisarjoja. Potenssisarja-alkioisten matriisien matriisitulo on silloin sama kuin matriisikertoimisten potenssisarjojen Cauchyn tulo (miksi?). Todistusta varten tarvitaan aputulos:

Apulause. *Jos* $\mathbf{P} \in R^{m\times m}\langle\langle\Sigma\rangle\rangle$ *on kvasisäännöllinen ja* $\mathbf{Z}$ *sekä* $\mathbf{Q}$ *ovat* m*-vaakavektoreita, joiden alkiot ovat* $R\langle\langle\Sigma\rangle\rangle$*:ssä, niin yhtälön*

$$\mathbf{Z} = \mathbf{Q} + \mathbf{ZP}$$

ainoa ratkaisu on $\mathbf{Z} = \mathbf{QP}^*$*. Lisäksi, jos* $\mathbf{P}$*:n ja* $\mathbf{Q}$*:n alkiot ovat* R*-rationaalisia, niin samoin ovat* $\mathbf{Z}$*:n alkiot.*

Todistus. Lauseen 36 nojalla $\mathbf{P}^* = \mathbf{I}_m + \mathbf{P}^*\mathbf{P}$, joten $\mathbf{Z} = \mathbf{QP}^*$ on yhtälön ratkaisu. Toisaalta ratkaisu toteuttaa myös „iteroimalla" saadut yhtälöt

$$\mathbf{Z} = \mathbf{Q}\sum_{i=0}^{n-1}\mathbf{P}^i + \mathbf{ZP}^n \quad (n = 1, 2, \dots),$$

missä „jäännöstermi" $\mathbf{ZP}^n$ pitää sisällään kaikki termit, joissa sanan pituus on $\geq n$ (muista, että $\mathbf{P}$ on kvasisäännöllinen). Näin ollen ratkaisu on yksikäsitteinen.

Oletetaan sitten, että $\mathbf{P}$:n ja $\mathbf{Q}$:n alkiot ovat R-rationaalisia potenssisarjoja ja näytetään induktiolla m:n suhteen, että samoin ovat ratkaisun $\mathbf{Z}$ alkiot.

Induktion lähtökohta, tapaus $m = 1$, on selvä. Oletetaan (Induktio-oletus), että väite on oikea, kun $m = l - 1$, ja siirrytään tapaukseen $m = l$ (Induktioväite). Merkitään Induktioväitteen todistusta varten

$$\mathbf{Z} = (Z_1, \dots, Z_l) \quad \text{ja} \quad \mathbf{Q} = (Q_1, \dots, Q_l)$$

sekä vielä $\mathbf{P} = (P_{ij})$. Silloin

$$Z_l = R_l + Z_l P_{ll},$$

missä

$$R_l = Q_l + Z_1 P_{1l} + \cdots + Z_{l-1} P_{l-1,l}.$$

Ilmeisestikin R_l on R-rationaalinen potenssisarja, jos vain $Z_1, \dots, Z_{l-1}$ ovat sitä. Edellä olevan nojalla

$$Z_l = \begin{cases} R_l, \text{ jos } P_{ll} = \mathbf{0} \\ R_l P_{ll}^*, \text{ jos } P_{ll} \neq \mathbf{0}. \end{cases}$$

(Huomaa, että P_{ll} on kvasisäännöllinen.) Sijoittamalla saatu Z_l yhtälöön $\mathbf{Z} = \mathbf{Q} + \mathbf{ZP}$ saadaan yhtä alempaa dimensiota $m = l - 1$ oleva yhtälö, jonka kerroinmatriisi on kvasisäännöllinen $R^{(l-1)\times(l-1)}\langle\langle\Sigma\rangle\rangle$:n formaali potenssisarja. Induktio-oletuksen nojalla sen ratkaisun alkiot $Z_1, \dots, Z_{l-1}$ ovat R-rationaalisia potenssisarjoja. Näin ollen myös Z_l on R-rationaalinen. □

Näin saadaan Kleenen lausetta vastaava kuuluisa tulos potenssisarjoille:

Schützenberger'n esityslause. *R-rationaaliset formaalit potenssisarjat ovat tarkalleen kaikki R-tunnistuvat formaalit potenssisarjat.*

Todistus. Edellä olevan nojalla pitää siis vielä todistaa, että R-tunnistuva aakkoston $\Sigma = \{x_1, \dots, x_k\}$ formaali potenssisarja, joka $m \times m$-matriisiesityksessä on

$$S = \sum_{w\in\Sigma^*} \mathbf{s}\boldsymbol{\mu}(w)\mathbf{a}^{\mathsf{T}} w,$$

on R-rationaalinen. Puolirenkaan $R^{m\times m}\langle\langle\Sigma\rangle\rangle$ formaali potenssisarja

$$\mathbf{P} = \boldsymbol{\mu}(x_1)x_1 + \cdots \boldsymbol{\mu}(x_k)x_k$$

(polynomi) on kvasisäännöllinen ja sen alkiot ovat R-rationaalisia. Edelleen ilmeisestikin

$$\mathbf{P}^* = \sum_{w\in\Sigma^*} \boldsymbol{\mu}(w)w \quad \text{ja} \quad S = \mathbf{s}\mathbf{P}^*\mathbf{a}^\mathsf{T}.$$

Apulauseen mukaan $\mathbf{Z} = \mathbf{s}\mathbf{P}^*$ on yhtälön $\mathbf{Z} = \mathbf{s} + \mathbf{Z}\mathbf{P}$ ainoa ratkaisu, sen alkiot ovat R-rationaalisia ja samoin on näin ollen $S = \mathbf{s}\mathbf{P}^*\mathbf{a}^\mathsf{T}$. □

9.5 Tunnistuvuus ja Hadamardin tulo

Tunnistuvien formaalien potenssisarjojen Hadamardin tulo on aina myös tunnistuva. Tämä voidaan todistaa helposti matriisien Kroneckerin tuloja käyttäen.

Matriisien $\mathbf{A} = (a_{ij})$ ($n_1 \times m_1$-matriisi) ja $\mathbf{B} = (b_{ij})$ ($n_2 \times m_2$-matriisi) *Kroneckerin tulo*[3] on $n_1n_2 \times m_1m_2$-matriisi.

$$\mathbf{A}\otimes\mathbf{B} = \left(\begin{array}{c|c|c|c} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1m_1}\mathbf{B} \\ \hline a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2m_1}\mathbf{B} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline a_{n_11}\mathbf{B} & a_{n_12}\mathbf{B} & \cdots & a_{n_1m_2}\mathbf{B} \end{array}\right)$$

(lohkomuodossa). Erikoistapauksena saadaan kahden vektorin Kroneckerin tulo ($n_1 = n_2 = 1$ tai $m_1 = m_2 = 1$). Seuraavat Kroneckerin tulon perusominaisuudet ovat varsin helppoja todistaa. Tässä oletetaan, että esiintyvät matriisioperaatiot ovat määriteltyjä.

1. Liitännäisyys:
$$(\mathbf{A}\otimes\mathbf{B})\otimes\mathbf{C} = \mathbf{A}\otimes(\mathbf{B}\otimes\mathbf{C})$$
Tämän seurauksena useammat peräkkäiset Kroneckerin tulot voidaan kirjoittaa ilman sulkeita.

2. Kroneckerin tulojen kertolasku (seuraa jokseenkin suoraan lohkomatriisien kertolaskusta):
$$(\mathbf{A}_1\otimes\mathbf{B}_1)(\mathbf{A}_2\otimes\mathbf{B}_2) = (\mathbf{A}_1\mathbf{A}_2)\otimes(\mathbf{B}_1\mathbf{B}_2)$$

3. Kahden identiteettimatriisin Kroneckerin tulo on identiteettimatriisi.

4. Kroneckerin tulon inverssi (seuraa kertolaskulaista):
$$(\mathbf{A}\otimes\mathbf{B})^{-1} = \mathbf{A}^{-1}\otimes\mathbf{B}^{-1}$$

[3]Usein Kroneckerin tuloa kutsutaan myös *tensorituloksi,* joskus myös Hadamardin tuloksi.

5. Kroneckerin tulon transpoosi (seuraa suoraan lohkomatriisien transponoinnista):

$$(\mathbf{A} \otimes \mathbf{B})^{\mathsf{T}} = \mathbf{A}^{\mathsf{T}} \otimes \mathbf{B}^{\mathsf{T}}$$

Sama pätee kompleksialkioisten matriisien konjugaattitranspoosille:

$$(\mathbf{A} \otimes \mathbf{B})^{\dagger} = \mathbf{A}^{\dagger} \otimes \mathbf{B}^{\dagger}$$

6. Unitääristen matriisien Kroneckerin tulot ovat unitäärisiä. (Seuraa edellisistä.)

Huomaa erityisesti, että 1×1-matriisien Kroneckerin tulo on yksinkertaisesti skalaarien kertolasku. Formaalien potenssisarjojen S_1 ja S_2 matriisiesitykset

$$\kappa_1(w) = \mathbf{s}_1 \boldsymbol{\mu}_1(w) \mathbf{a}_1^{\mathsf{T}} \quad \text{ja} \quad \kappa_2(w) = \mathbf{s}_2 \boldsymbol{\mu}_2(w) \mathbf{a}_2^{\mathsf{T}},$$

vastaavasti, voidaankin nyt kertoa Kroneckerin tulona:

$$\begin{aligned} \kappa_1(w)\kappa_2(w) = \kappa_1(w) \otimes \kappa_2(w) &= \big(\mathbf{s}_1 \boldsymbol{\mu}_1(w) \mathbf{a}_1^{\mathsf{T}}\big) \otimes \big(\mathbf{s}_2 \boldsymbol{\mu}_2(w) \mathbf{a}_2^{\mathsf{T}}\big) \\ &= (\mathbf{s}_1 \otimes \mathbf{s}_2)\big(\boldsymbol{\mu}_1(w) \otimes \boldsymbol{\mu}_2(w)\big)(\mathbf{a}_1^{\mathsf{T}} \otimes \mathbf{a}_2^{\mathsf{T}}). \end{aligned}$$

Matriisiesitys $\kappa(w) = \mathbf{s}\boldsymbol{\mu}(w)\mathbf{a}^{\mathsf{T}}$ Hadamardin tulolle $S_1 \otimes S_2$ saadaan siis, kun valitaan

$$\mathbf{s} = \mathbf{s}_1 \otimes \mathbf{s}_2 \quad \text{ja} \quad \mathbf{a} = \mathbf{a}_1 \otimes \mathbf{a}_2$$

sekä

$$\boldsymbol{\mu}(x_l) = \boldsymbol{\mu}_1(x_l) \otimes \boldsymbol{\mu}_2(x_l) \quad (l = 1, \ldots, k).$$

Silloin nimittäin kertolaskusäännön nojalla

$$\boldsymbol{\mu}(w) = \boldsymbol{\mu}_1(w) \otimes \boldsymbol{\mu}_2(w).$$

Näin on saatu

Lause 39. *R-tunnistuvien formaalien potenssisarjojen Hadamardin tulo on R-tunnistuva.*

Schützenberger'n esityslauseen nojalla edelleen

Seuraus. *R-rationaalisten formaalien potenssisarjojen Hadamardin tulo on R-rationaalinen.*

9.6 Esimerkkejä formaaleista potenssisarjoista

9.6.1 Multikielet

Palataan Pykälän 4 epädeterministiseen äärelliseen automaattiin (ilman Λ-siirtoja) $M = (Q, \Sigma, S, \delta, A)$. Automaatin tilat olivat $\{q_1, \ldots, q_m\}$. Uutena asiana otetaankin nyt puolirenkaaksi $N = (\mathbb{N}, +, \cdot, 0, 1)$ eikä B:tä. Toiseksi automaatti ajatellaan *multiautomaatiksi,* jolloin

- siirtofunktion arvo $\delta(q_i, x_l, q_j)$—huomaa kolme argumenttia—antaa tiedon siitä miten monella eri tavalla voidaan tilasta q_i siirtyä tilaan q_j lukien x_l. Arvo voi olla $= 0$, jolloin tilasta q_i ei voi siirtyä lainkaan tilaan q_j lukien x_l.

- alku- ja lopputilajoukot ovat multijoukkoja, ts. tila voi esiintyä niissä monta kertaa.

Matriisiesityksessä symbolia $x_l \in \Sigma$ asetetaan vastaamaan kokonaisalkioinen $m \times m$-matriisi $\boldsymbol{\mu}(x_l) = \mathbf{D}_l = (d_{ij}^{(l)})$, missä $d_{ij}^{(l)} = \delta(q_i, x_l, q_j)$. Alku- ja lopputilajoukkoja vastaamaan asetetaan vastaavasti kokonaisalkioiset vaakavektorit $\mathbf{s} = (s_1, \ldots, s_m)$ sekä $\mathbf{a} = (a_1, \ldots, a_m)$, missä s_i ilmoittaa miten monella tavalla q_i on alkutila ja a_i miten monella tavalla q_i on lopputila. Jos $s_i = 0$, niin q_i ei ole lainkaan alkutila. Vastaavasti jos $a_i = 0$, q_i ei ole lopputila.

Silloin niiden tapojen lukumäärät, joilla eri tiloihin on päästävissä alkutiloista, kun luetaan syötesymboli x_l, ovat luettavissa vektorista $\mathbf{sD}_l$. Yleisesti niiden tapojen lukumäärät, joilla tiloihin on päästävissä alkutiloista syötteellä $w = x_{l_1} x_{l_2} \cdots x_{l_k}$, ovat luettavissa vektorista

$$\mathbf{sD}_{l_1}\mathbf{D}_{l_2} \cdots \mathbf{D}_{l_k} = \mathbf{s}\boldsymbol{\mu}(w)$$

Näin ollen niiden eri tapojen lukumäärä, joilla sana w voidaan hyväksyä—mukaanlukien 0, jos sanaa ei hyväksytä—on

$$\mathbf{s}\boldsymbol{\mu}(w)\mathbf{a}^{\mathsf{T}} = \kappa(w).$$

Formaali potenssisarja

$$S(M) = \sum_{w \in \Sigma^*} \kappa(w) w,$$

ns. M:n tunnistama *multikieli,* on silloin N-tunnistuva ja Schützenberger'n esityslauseen nojalla myös N-rationaalinen. $\kappa(w)$ on sanan w ns. *multiplisiteetti.*

Jokainen N-tunnistuva, ja siis myös N-rationaalinen, multikieli on ilmeisesti tulkittavissa erään multiautomaatin tunnistamaksi multikieleksi. Yleensäkin $N\langle\langle\Sigma\rangle\rangle$:n formaalit potenssisarjat ovat ajateltavissa aakkoston Σ multikieliksi, mutta niille ei silloin yleensä ole yhtä välitöntä tulkintaa.

9.6.2 Stokastiset kielet

Reaalialkioista vaakavektoria $\mathbf{v}$ sanotaan *stokastiseksi,* jos sen alkiot ovat ≥ 0 ja niiden summa on $= 1$. Reaalialkioinen matriisi on *stokastinen,* jos sen rivit ovat stokastisia vektoreita.

Muutetaan Pykälän 4 automaatti *stokastiseksi äärelliseksi automaatiksi* lisäämällä siihen todennäköisyydet:

- Alkutilavektorin tilalla on alkutilatodennäköisyyksien stokastinen vektori $\mathbf{s} = (s_1, \ldots, s_m)$, missä s_i on todennäköisyys $\mathsf{P}(q_i \text{ on alkutila})$.

- Lopputilavektorin tilalla on lopputilatodennäköisyyksien vektori $\mathbf{a} = (a_1, \ldots, a_m)$ (ei välttämättä stokastinen), missä a_i on todennäköisyys $\mathsf{P}(q_i \text{ on lopputila})$.

- Tilansiirtomatriisien tilalla ovat tilansiirtotodennäköisyyksien stokastiset matriisit $\mathbf{D}_l = (d_{ij}^{(l)})$. Edelleen

  $$d_{ij}^{(l)} = \mathsf{P}(\text{siirrytään tilaan } q_j, \text{ kun luetaan } x_l \mid \text{ollaan tilassa } q_i),$$

  ts. ehdollinen todennäköisyys, että luettaessa x_l siirrytään tilaan q_j sillä ehdolla, että ollaan tilassa q_i.

Todennäköisyyksien laskusääntöjen mukaan

$$\mathsf{P}(\text{siirrytään tilaan } q_j, \text{ kun luetaan } x_l) = \sum_{i=1}^{m} d_{ij}^{(l)} \mathsf{P}(\text{ollaan tilassa } q_i).$$

Näin ollen vektori $\mathbf{sD}_l$ antaa tilatodennäköisyydet, kun on luettu ensimmäinen symboli x_l. Yleisesti tilatodennäköisyydet sanan w lukemisen jälkeen antaa vektori

$$\mathbf{s}\boldsymbol{\mu}(w).$$

(Matemaattisesti tilansiirtoketju on ns. Markovin prosessi.) Edelleen todennäköisyyksien laskusääntöjen mukaan

$$\begin{aligned}
&\mathsf{P}(\text{sanan } w \text{ lukemisen jälkeen ollaan lopputilassa}) \\
&= \sum_{i=1}^{m} \mathsf{P}(\text{sanan } w \text{ lukemisen jälkeen ollaan tilassa } q_i)\mathsf{P}(q_i \text{ on lopputila}) \\
&= \mathbf{s}\boldsymbol{\mu}(w)\mathbf{a}^{\mathsf{T}} = \kappa(w).
\end{aligned}$$

Näin saadaan puolirenkaassa $(\mathbb{R}_+, +, \cdot, 0, 1)$ tunnistuva formaali potenssisarja

$$P = \sum_{w \in \Sigma^*} \kappa(w) w,$$

missä $\kappa(w)$ on todennäköisyys, että sana w on kielessä. Kyseessä on ns. *stokastinen kieli.*

Ilmeisesti stokastiset kielet eivät ole suljettuja summan, Cauchyn tulon eikä kvasi-inverssin suhteen, koska niille on $\kappa(w) \leq 1$. Toisaalta

Lause 40. *Stokastiset kielet ovat suljettuja Hadamardin tulon suhteen.*

Todistus. Pykälän 5 konstruktion perusteella pitää vain todeta, että stokastisten vektorien Kroneckerin tulo on stokastinen ja että sama pätee stokastisille matriiseille. Alkioiden ei-negatiivisuus ilmeisesti säilyy Kroneckerin tulossa. m-vektorille $\mathbf{v}$ ja $m \times m$-matriisille $\mathbf{M}$ stokastisuuden summaehto on kirjoitettavissa muotoon

$$\mathbf{v}\mathbf{1}_m = 1 \quad \text{ja} \quad \mathbf{M}\mathbf{1}_m = \mathbf{1}_m,$$

missä $\mathbf{1}_m$ on m-pystyvektori, jonka kaikki alkiot ovat $= 1$. Jos nyt $\mathbf{v}_1$ ja $\mathbf{v}_2$ ovat stokastisia vektoreita dimensioissa m_1 ja m_2, vastaavasti, niin

$$(\mathbf{v}_1 \otimes \mathbf{v}_2)\mathbf{1}_{m_1 m_2} = (\mathbf{v}_1 \mathbf{1}_{m_1}) \otimes (\mathbf{v}_2 \mathbf{1}_{m_2}) = 1 \cdot 1 = 1$$

(huomaa, että $\mathbf{1}_{m_1} \otimes \mathbf{1}_{m_2} = \mathbf{1}_{m_1 m_2}$). Näin ollen vektori $\mathbf{v}_1 \otimes \mathbf{v}_2$ on stokastinen. Vastaavalla tavalla näytetään, että stokastisten matriisien Kroneckerin tulo on stokastinen. □

Stokastisesta kielestä P saadaan „varsinainen kieli" asettamalla todennäköisyydelle kynnys. Esimerkiksi

$$\{w \mid \kappa(w) > 0.5\}$$

on tällainen kieli. Näitäkin kieliä kutsutaan stokastisiksi. Tällaisille stokastisille kielille saadaan toinenkin kuuluisa karakterisaatio. Ne nimittäin voidaan kirjoittaa muodossa

$$\{w \mid \kappa'(w) > 0\},$$

missä

$$\sum_{w\in\Sigma^*} \kappa'(w)w$$

on R-rationaalinen formaali potenssisarja reaalilukujen puolirenkaassa $R = \{\mathbb{R}, +, \cdot, 0, 1\}$ (tavallisin laskuoperaatioin). Kääntäen, jokainen vm. muotoa oleva kieli on stokastinen. Tulos tunnetaan *Turakaisen lauseena.*[4]

9.6.3 Pituusfunktiot

Kielen L *pituusfunktio* on kuvaus

$$\lambda_L(n) = L\text{:n } n\text{-pituisten sanojen lukumäärä.}$$

Vastaava formaali potenssisarja yli aakkoston $\{x\}$ puolirenkaassa $N = (\mathbb{N}, +, \cdot, 0, 1)$ on

$$S_L = \sum_{n=0}^{\infty} \lambda_L(n)x^n.$$

Lause 41. *Säännölliselle kielelle L formaali potenssisarja S_L on N-rationaalinen (ja siis myös N-tunnistuva).*

Todistus. Otetaan kielen L tunnistava deterministinen äärellinen automaatti $M = (Q, \Sigma, q_0, \delta, A)$. Muutetaan M äärelliseksi multiautomaatiksi $M' = \big(Q, \{x\}, S, \delta', B\big)$ seuraavasti. Tilansiirtoluku

$$\delta'(q_i, x, q_j) = n$$

on voimassa tarkalleen silloin, kun on n sellaista symbolia x_l, että $\delta(q_i, x_l) = q_j$. Erityisesti $\delta'(q_i, x, q_j) = 0$ tarkalleen silloin, kun kaikille symboleille x_l on $\delta(q_i, x_l) \neq q_j$. Edelleen S:ssä on vain M:n alkutila q_0 multiplisiteetilla 1 ja B:ssä vain A:n tilat kukin multiplisiteetilla 1.

Silloin ilmeisesti multikieli $S(M')$ on juuri mainittu potenssisarja S_L. □

9.6.4 Kvanttikielet

Kvanttiautomaatti aakkostossa $\Sigma = \{x_1, \ldots, x_l\}$ saadaan, kun valitaan luku m, kompleksialkioinen m-vaakavektori $\mathbf{s} = (s_1, \ldots, s_m)$, ns. *alkutila,* jolle

$$\|\mathbf{s}\|^2 = \mathbf{s}\mathbf{s}^\dagger = \sum_{i=1}^{m} |s_i|^2 = 1,$$

sekä kompleksialkioiset *tilansiirtomatriisit*

$$\mathbf{U}_l = \boldsymbol{\mu}(x_l)$$

vastaten Σ:n symboleja. Tilansiirtomatriisien on on oltava unitäärisiä matriiseja, ts. aina $\mathbf{U}_l^{-1} = \mathbf{U}_l^\dagger$, muutoin tilansiirto ei ole kvanttiteoreettisesti mielekäs operaatio.

[4] Alkuperäisviite on TURAKAINEN, P.: Generalized Automata and Stochastic Languages. *Proceedings of the American Mathematical Society* **21** (1969), 303–309.

Kvanttikieli[5] on formaali potenssisarja

$$\mathbf{Q} = \sum_{w \in \Sigma^*} \boldsymbol{\kappa}(w) w,$$

missä

$$\boldsymbol{\kappa}(w) = \mathbf{s}\boldsymbol{\mu}(w)$$

on kvanttiautomaatin *tila* sanan w lukemisen jälkeen. Jälleen tässä $\boldsymbol{\mu}$ määräytyy matriiseista $\boldsymbol{\mu}(x_l) = \mathbf{U}_l$ ja säännöistä $\boldsymbol{\mu}(\Lambda) = \mathbf{I}_m$ ja $\boldsymbol{\mu}(uv) = \boldsymbol{\mu}(u)\boldsymbol{\mu}(v)$. Huomaa, että tämä $\mathbf{Q}$ ei oikeastaan ole formaali potenssisarja, koska kertoimet ovat vektoreita. Jokainen sen komponentti on toisaalta tunnistuva formaali potenssisarja puolirenkaassa $(\mathbb{C}, +, \cdot, 0, 1)$ (kompleksiluvut tavallisin laskuoperaatioin).

Kvanttikielillä, kuten stokastisilla kielillä, on varsin huonot sulkeumaominaisuudet. Kvanttikieletkin ovat kuitenkin suljettuja Hadamardin tulon suhteen, sillä

$$(\mathbf{s}_1 \otimes \mathbf{s}_2)(\mathbf{s}_1 \otimes \mathbf{s}_2)^\dagger = (\mathbf{s}_1 \otimes \mathbf{s}_2)(\mathbf{s}_1^\dagger \otimes \mathbf{s}_2^\dagger) = (\mathbf{s}_1\mathbf{s}_1^\dagger) \otimes (\mathbf{s}_2\mathbf{s}_2^\dagger) = 1 \cdot 1 = 1$$

ja unitääristen matriisien Kroneckerin tulo on aina myös unitäärinen, vrt. edellinen pykälä. Hadamardin tulo vastaakin kvanttirekisterien yhdistämistä pidemmäksi rekisteriksi, ks. esimerkiksi Nielsen & Chuang. Tässä Hadamardin tulo on siis kerroinvektorien Kroneckerin tulo.

Lopputilakonstruktiota vastaa tietyssä mielessä kvanttifysikaalinen *mittaus.* Tila $\boldsymbol{\kappa}(w)$ kerrotaan jollain projektiomatriisilla $\mathbf{P}$.[6] $\mathbf{P}$:llä kertominen projisoi vektorin ortogonaalisesti tiettyyn $\mathbb{C}^m$:n aliavaruuteen H. Silloin kvanttifysiikan ns. *todennäköisyystulkinnassa*

$$\left\|\boldsymbol{\kappa}(w)\mathbf{P}\right\|^2$$

on todennäköisyys sille, että tila $\boldsymbol{\kappa}(w)$ on mittauksen jälkeen aliavaruudessa H. Huomaa, että myös $\boldsymbol{\mu}(w)$ on unitäärinen ja

$$\left\|\boldsymbol{\kappa}(w)\mathbf{P}\right\|^2 \leq \left\|\boldsymbol{\kappa}(w)\right\|^2 = \mathbf{s}\boldsymbol{\mu}(w)\boldsymbol{\mu}(w)^\dagger\mathbf{s}^\dagger = \mathbf{s}\mathbf{s}^\dagger = 1.$$

Mittauksen tarkoitus on selvittää onko tila aliavaruudessa H vai ei. Fysikaalisen mittauksen tulos on vastaus kyllä/ei. Kyllä-vastauksen voitaisiin ajatella merkitsevän syötteen hyväksymistä.

Eräänlaisiksi jonokoneiksi ajateltuna ja Hadamardin tulolla yhdistettyinä kvanttiautomaatit ovat huomattavasti perinteisiä klassisia tietokoneita nopeampia, nekin tietysti ovat oikeastaan (yleistettyjä) jonokoneita. Niin nopeita, että niillä voitaisiin murtaa monet yleisesti käytetyt kryptosysteemit, ks. kurssi Matemaattinen kryptologia ja viite Nielsen &Chuang. Toistaiseksi tosin vain varsin pieniä kvanttitietokoneita on voitu konstruoida fysikaalisesti.

[5] Alkuperäisviite on Moore,C. & Crutchfield, J.P.: Quantum Automata and Quantum Grammars. *Theoretical Computer Science* **237** (2000), 257–306.

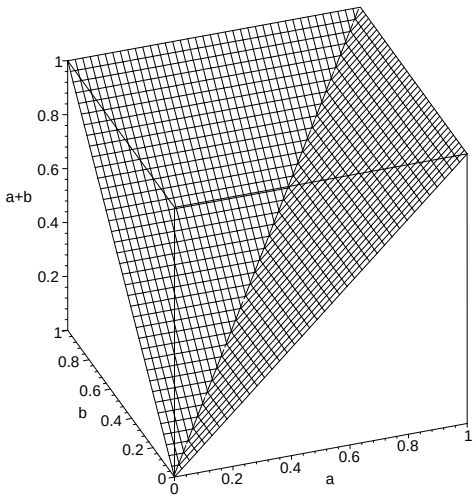
[6] Projektiomatriisi $\mathbf{P}$ on aina itseadjungoitu, ts. $\mathbf{P}^\dagger = \mathbf{P}$, sekä idempotentti, ts. $\mathbf{P}^2 = \mathbf{P}$. Lisäksi aina $\|\mathbf{x}\mathbf{P}\| \leq \|\mathbf{x}\|$, ortogonaaliprojektiossa pituus ei voi kasvaa.

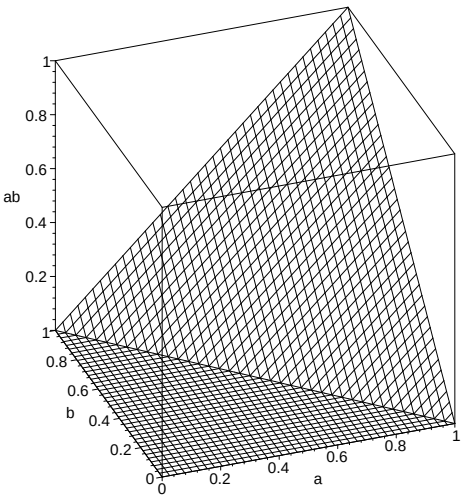
9.6.5 Sumeat kielet

Vaihtamalla „tavallinen” kaksiarvoinen logiikka moniarvoiseksi, saadaan hyvin monenlaisia Boolen puolirenkaan tapaisia puolirenkaita.[7] Esimerkkinä otetaan yksinkertainen puolirengas $S = \big\{[0,1], +, \cdot, 0, 1\big\}$, missä $[0,1]$ on reaalilukuväli ja operaatiot määritellään yhtälöillä

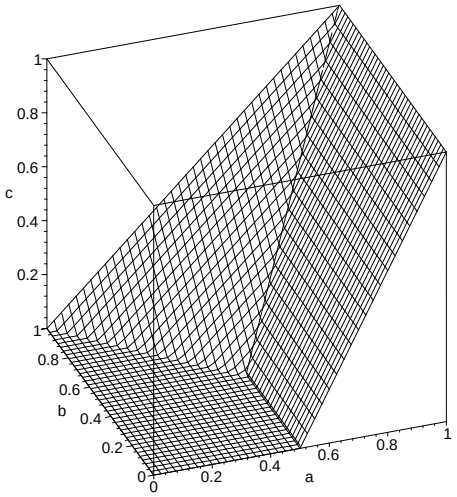
$$a + b = \max(a, b) \quad \text{ja} \quad a \cdot b = \max(0, a + b - 1).$$

Operaatioiden kuvaajapinnat ovat (Maple)





ja esimerkiksi lausekkeen $c = a \cdot \big(a + b \cdot (1 + a)\big)$ kuvaajapinta on



Helposti on todettavissa, että S todella on puolirengas (harjoituksena). Rajoittumalla vain alkioihin 0 ja 1 tästä syntyy Boolen puolirengas B.

Jokaiselle sanalle w luku $\kappa(w)$ antaa eräänlaisen „kuulumisasteen”, jolla sana kuuluu kieleen. S-tunnistuvan formaalin potenssisarjan matriisiesitys puolestaan antaa vastaavan *sumean automaatin,* jonka toiminta muistuttaa eo. stokastista automaattia.

[7] Itse asiassa jokaisesta moniarvologiikan määrittämästä ns. MV-algebrasta saa useita sellaisia.

KIRJALLISUUS

1. Berstel, J. & Perrin, D. & Reutenauer, C.: *Codes and Automata.* Cambridge University Press (2008)

2. Berstel, J. & Reutenauer, C.: *Noncommutative Rational Series with Applications* (2008) (Saatavana verkossa)

3. Harrison, M.A.: *Introduction to Formal Language Theory.* Addison–Wesley (1978)

4. Hopcroft, J.E. & Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation.* Addison–Wesley (1979)

5. Hopcroft, J.E. & Motwani, R. & Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation.* Addison–Wesley (2006)

6. Kuich, W. & Salomaa, A.: *Semirings, Automata, Languages.* Springer–Verlag (1986)

7. Lewis, H.R. & Papadimitriou, C.H.: *Elements of the Theory of Computation.* Prentice–Hall (1998)

8. Martin, J.C.: *Introduction to Languages and the Theory of Computation.* McGraw–Hill (2002)

9. McEliece, R.J.: *The Theory of Information and Coding.* Cambridge University Press (2004)

10. Meduna, A.: *Automata and Languages. Theory and Applications.* Springer–Verlag (2000)

11. Nielsen, M.A. & Chuang, I.L.: *Quantum Computation and Quantum Information.*Cambridge University Press (2000)

12. Rozenberg, G. & Salomaa, A. (toim.): *Handbook of Formal Languages. Vol. 1. Word, Language, Grammar.* Springer–Verlag (1997)

13. Salomaa, A.: *Formal Languages.* Academic Press (1973)

14. Salomaa, A.: *Jewels of Formal Language Theory.* Computer Science Press (1981)

15. Salomaa, A. & Soittola, M.: *Automata-Theoretic Aspects of Formal Power Series.* Springer–Verlag (1978)

16. Shallit, J.: *A Second Course in Formal Languages and Automata Theory.* Cambridge University Press (2008)

17. Simovici, D.A. & Tenney, R.L.: *Theory of Formal Languages with Applications.* World Scientific (1999)

18. Sipser, M.: *Introduction to the Theory of Computation.* Course Technology (2005)

Hakemisto