

4.3 Algoritmeista

- Churchin ja Turingin formuloinnit laskennalle syntyivät Hilbertin vuonna 1900 esittämän kymmenennen ongelman seurauksena
- Oleellisesti Hilbert pyysi algoritmia polynomin kokonaislukujuuren määräämiseksi
- Nykyisin tiedämme tämän ongelman olevan algoritmisesti ratkeamattoman
- Ilman täsmällistä määritelmää on mahdollista esittää algoritmeja, muttei osoittaa niiden olemassaoloa mahdottomaksi
- Vasta 1970 Matijasevič osoitti, ettei ole olemassa algoritmia sen testaamiseksi, onko polynomilla kokonaislukujuuri

- Kielenä ilmaisten Hilbertin kymmenes ongelma on

$$D = \{ p \mid p \text{ on polynomi, jolla on kokonaislukujuuri} \}$$
- Jos ajatellaan vain yhden muuttujan polynomeja, niin on helppo nähdä, kuinka kieli D voidaan tunnistaa
- Ainoan muuttujan oikean arvon löytämiseksi käydään sen mahdolliset kokonaislukuarvot läpi $0, 1, -1, 2, -2, 3, -3, \dots$
- Jos polynomin arvoksi tulee 0 millä tahansa tutkitulla muuttujan arvolla, niin syöte hyväksytään
- Vastaavasti voidaan toimia, kun muuttujia on useita

- Yhden muuttujan polynomeilla juurien pitää olla välillä $\pm k(c_{\max} / c_1)$,
 - missä k on polynomin termien lukumäärä,
 - c_{\max} on itseisarvoltaan suurin kerroin ja
 - c_1 on korkea-asteisimman termin kerroin
- Jos juurta ei löydy näiden rajojen puitteissa, niin Turingin kone voi hylätä syötteen
- Matijasevič'n todistus osoittaa, ettei monen muuttujan polynomeille voi laskea vastaavia rajoja
- Kieli D on siis Turingin koneella tunnistettavissa, muttei ratkaistavissa (voi jäädä pysähtymättä)

5. Laskettavuusteoriaa

- Tarkastellaan ongelmien *algoritmista ratkeavuutta*
 - S.o. ratkeavuutta Turingin koneilla
- Erotamme tapaukset, joissa formaalit kielet ovat tunnistettavissa jollain Turingin koneella, ja ne, joissa koneen vaaditaan pysähtyvän kaikilla syöteillä
- Osoittautuu, että on monia luonnollisia ja mielenkiintoisia ongelmia, joita ei voida ratkaista Turingin koneilla.
- Näin ollen Churchin-Turingin teesin perusteella nämä ongelmat ovat myös tietokoneilla ratkeamattomia!

Määritelmä Turingin kone on **totaalinen**, jos se pysähtyy kaikilla syötteillä.

Formaali kieli on **rekursiivisesti lueteltava** (RE-kieli, Turing-tunnistettava), jos se voidaan tunnistaa jollakin Turingin koneella.

Formaali kieli on **rekursiivinen** (ratkeava), jos se voidaan tunnistaa jollakin totaalisella Turingin koneella.

- Kieltä A vastaava päätösongelma on **ratkeava**, jos A on rekursiivinen. Ongelma, joka ei ole ratkeava, on **ratkeamaton**
- Kieltä A vastaava päätösongelma on **osittain ratkeava**, jos A on RE-kieli
- Huom.: ratkeamaton ongelma voi olla osittain ratkeava.

5.1 Rekursiivisten ja RE-kielten perusominaisuuksia

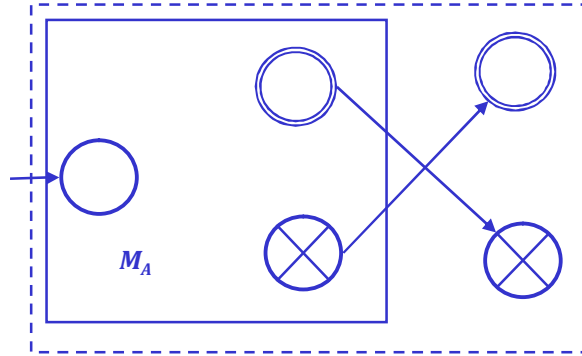
Lause 5.1 Olkoot $A, B \subseteq \Sigma^*$ rekursiivisia kieliä. Tällöin myös kielet

1. $\bar{A} = \Sigma^* \setminus A$,
2. $A \cup B$ ja
3. $A \cap B$

ovat rekursiivisia.

Todistus.

1. Olk. M_A kielen A tunnistava totaalinen Turingin kone. Vaihtamalla M_A :n hylkäävä ja hyväksyvä lopputila keskenään saadaan totaalinen Turingin kone, joka tunnistaa kielen \bar{A} .



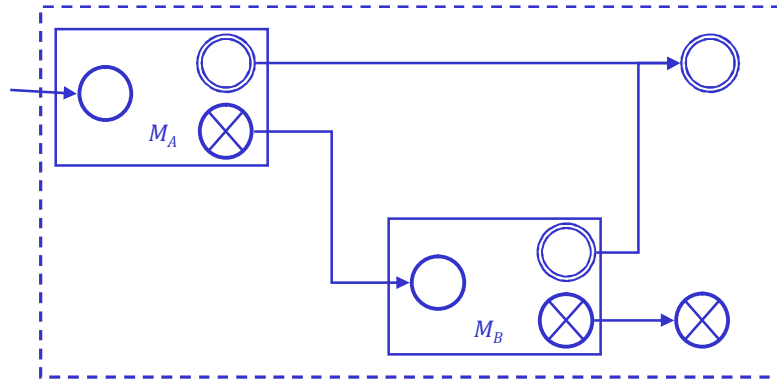
Kielen A komplementin tunnistajakone

2. Olkoot M_A ja M_B kielet A ja B tunnistavat totaaliset Turingin koneet.
 - Yhdistetään koneet siten, että ensin tarkastetaan hyväksyykö M_A syötteen.
 - Jos hyväksyy, niin yhdistetty konekin hyväksyy.
 - Jos M_A hylkää syötteen, niin se annetaan edelleen koneen M_B tarkastettavaksi.
 - Tässä tapauksessa M_B tekee lopullisen päätöksen. M_A :n tulee antaa M_B :lle alkuperäinen syöte.
 - Selvästi yhdistetty kone on totaalinen ja hyväksyy kielen $A \cup B = \{x \in \Sigma^* \mid x \in A \vee x \in B\}$

3. $(A \cap B)$:n rekursiivisuus seuraa edellisistä kohdista, sillä

$$A \cap B = \overline{\overline{A \cup B}}$$

Kielten A ja B yhdisteen tunnistajakone:



Lause 5.2 Olkoot $A, B \subseteq \Sigma^*$ RE-kieliä. Tällöin myös kielet $A \cup B$ ja $A \cap B$ ovat RE-kieliä.

Todistus. Laskuharjoitukset.

- Lauseen 5.3 seurauksena pätee seuraava tulos:

Lause 5.4 Olkoon $A \subseteq \Sigma^*$ RE-kieli, joka ei ole rekursiivinen. Tällöin \bar{A} ei ole RE-kieli.

Lause 5.3 Kieli $A \subseteq \Sigma^*$ on rekursiivinen $\Leftrightarrow A$ ja \bar{A} ovat RE-kieliä.

Todistus.

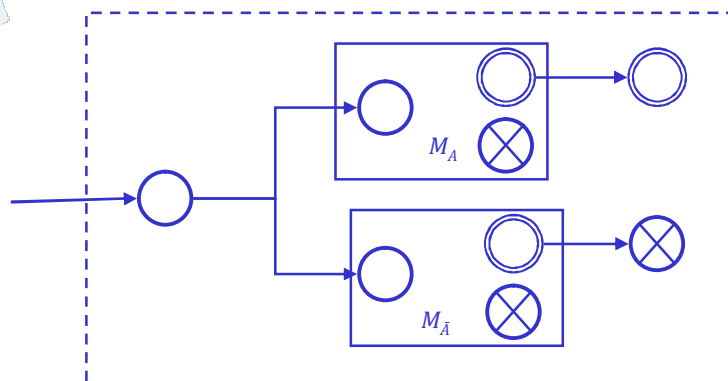
\Rightarrow Jos A on rekursiivinen, niin se on myös RE-kieli.
Lauseen 5.1(1) perusteella sama pätee myös \bar{A} :lle.

\Leftarrow Olkoot M_A ja $M_{\bar{A}}$ kielet A ja \bar{A} tunnistavat Turingin koneet.

Kaikilla $x \in \Sigma^*$ pätee joko $x \in A$ tai $x \in \bar{A}$. Siis joko kone M_A tai $M_{\bar{A}}$ pysähtyy syötteellä x .

Yhdistämällä koneet M_A ja $M_{\bar{A}}$ rinnakkain toimiviksi saadaan A :lle totaalinen tunnistajakone. \square

Totaalinen Turingin kone Kielen A tunnistamiseksi:



5.2 Turingin koneiden koodaus

Standardimallinen Turingin kone

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \text{accept}, \text{reject}),$$

missä $\Sigma = \{0, 1\}$, voidaan esittää binäärijonona seuraavasti:

- $Q = \{q_0, q_1, \dots, q_n\}$, missä $q_{n-1} = \text{accept}$ ja $q_n = \text{reject}$
- $\Gamma = \{a_0, a_1, \dots, a_m\}$, missä $a_0 = 0, a_1 = 1, a_2 = _$
- Merk. $\Delta_0 = L$ ja $\Delta_1 = R$
- Siirtymäfunktion δ säännön

$$\delta(q_i, a_j) = (q_r, a_s, \Delta_t)$$

koodi on

$$c_{ij} = 0^{i+1}10^{j+1}10^{r+1}10^{s+1}10^{t+1}$$

- Koko koneen M koodi $\langle M \rangle$ on

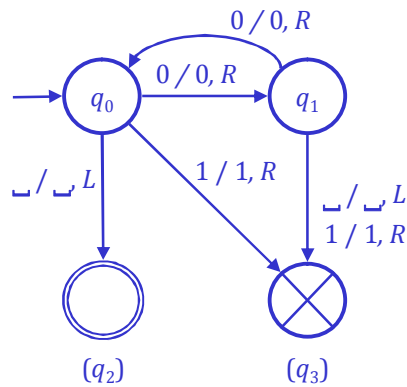
$$111c_{00}11c_{01}11 \dots 11c_{0m}11c_{10}11 \dots 11c_{n-2,0}11 \dots 11c_{n-2,m}111$$

- Esim. seuraavan koneen koodi on

$$\langle M \rangle = 11 \underbrace{1010100101001}_{\delta(q_0,0)=(q_1,0,R)} \underbrace{1010010000100100}_{\delta(q_0,1)=(q_3,1,R)} \dots 111$$

- Näin ollen jokaisella aakkoston $\{0, 1\}$ jonkin kielen tunnistavalla standardimallisella Turingin koneella M on siis binäärikoodi $\langle M \rangle$
- Toisaalta jokaiseen binäärijonoon c voidaan liittää jokin Turingin kone M_c

Kielen $\{0^{2k} \mid k \geq 0\}$ tunnistava Turingin kone:



- Kaikki binäärijonot eivät kuitenkaan ole Turingin koneiden koodeja
- Esimerkiksi **00**, **011110**, **111000111** ja **1110101010111** eivät ole edellisen koodauksen mukaisia Turingin koneiden koodeja
- Epäkelpoihin binäärijonoihin liitetään triviaali, kaikki syötteet hylkäävä kone M_{triv} :

$$M_c = \begin{cases} M, & \text{jos } c = \langle M \rangle \text{ on Turingin koneen } M \text{ koodi} \\ M_{\text{triv}}, & \text{muuten} \end{cases}$$

- Näin kaikki aakkoston $\{0, 1\}$ Turingin koneet voidaan luetteloida:

$$M_\epsilon, M_0, M_1, M_{00}, M_{01}, M_{10}, M_{000}, \dots$$

- Samalla saadaan luettelointi aakkoston $\{0, 1\}$ RE-kielille:

$$L(M_\epsilon), L(M_0), L(M_1), L(M_{00}), L(M_{01}), L(M_{10}), \dots$$

- Kielet voivat esiintyä luettelossa useammin kuin kerran
- Nyt diagonalisointitekniikalla voidaan todistaa, ettei päätösongelmaa
Hylkääkö annetun koodin c esittämä Turingin kone syötteen c ? vastaava kieli D ole RE-kieli
- Näin ollen D :tä vastaava päätösongelma on ratkeamaton

Lemma 5.5 Kieli $D = \{c \in \{0, 1\}^* \mid c \notin L(M_c)\}$ ei ole RE-kieli

Todistus. Oletetaan, että $D = L(M)$ jollakin standardimallisella Turingin koneella M .

Olkoon d koneen M binäärikoodi. Siis $D = L(M_d)$. Mutta nyt
$$d \in D \Leftrightarrow d \notin L(M_d) = D.$$

Tämä on ristiriita. Joten oletus ei pidä paikkaansa. Täten ei ole olemassa standardimallista Turingin konetta M s.e. $D = L(M)$.

Näin ollen D ei voi olla RE-kieli. □