

5.3 Ratkeavia ongelmia

- Deterministisen äärellisten automaattien (DFA) hyväksymisongelma: hyväksyykö annettu automaatti B merkkijonon w ?
- Ongelmaa vastaava formaali kieli on

$$A_{DFA} = \{ \langle B, w \rangle \mid B \text{ on DFA, joka hyväksyy } w:n \}$$

- Turingin kone voi helposti simuloida automaatin B toimintaa syötteellä w
- Jos simulointi päättyy hyväksyvään lopputilaan, hyväksyy kone syöteparin ja muutoin hylkää sen
- Automaatit voidaan koodata samaan tapaan kuin Turingin koneetkin

Lause 4.1 A_{DFA} on rekursiivinen kieli.

- Epädeterministiset äärelliset automaatit voidaan determinisoida, joten niille pätee vastaava lause
- Säännölliset lausekkeet puolestaan voidaan muuntaa epädeterministisiksi äärellisiksi automaateiksi, joten myös niille pätee vastaava tulos
- Toisenlainen ongelma on *tyhjiystestaus*: niiden automaattien A tunnistaminen, joiden hyväksymä kieli on tyhjä

$$E_{DFA} = \{ \langle A \rangle \mid A \text{ on DFA ja } L(A) = \emptyset \}$$

Lause 4.4 E_{DFA} on rekursiivinen kieli.

- Turingin kone voi toimia seuraavasti:
 1. Merkitsi automaatin alkutila
 2. Toista kunnes uusia tiloja ei enää tule merkittyä: Merkitse jokainen tila, johon on siirtymä aiemmin merkitystä tilasta
 3. Hyväksy syöte jos yksikään lopputiloista ei ole merkitty, muutoin hylkää se
- Myös kahden automaatin hyväksymän kielen ekvivalenssin testaaminen on ratkeava ongelma
- Ongelmaa vastaava kieli on

$$EQ_{DFA} = \{ \langle A, B \rangle \mid A \text{ ja } B \text{ ovat DFAita ja } L(A) = L(B) \}$$
- EQ_{DFA} :n rekursiivisuus seuraa lauseesta 4.4, kun siirrytään tarkastelemaan kielten $L(A)$ ja $L(B)$ *symmetristä erotusta*

$$\begin{aligned} L(C) &= (L(A) \setminus L(B)) \cup (L(B) \setminus L(A)) \\ &= (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B)) \end{aligned}$$

- $L(C) = \emptyset$ joss $L(A) = L(B)$
- Koska säännöllisten kielten luokka on suljettu yhdisteen, leikkauksen ja komplementoinnin suhteen, niin Turingin koneella voidaan muodostaa automaatti C annettuna A ja B
- Lauseen 4.4 perusteella voidaan testata onko $L(C)$ tyhjä kieli

Lause 4.5 EQ_{DFA} on rekursiivinen kieli.

- Siirryttäessä tarkastelemaan kontekstittomia kielioppeja, ei voida tarkastella kaikkia mahdollisia johtoja, koska niitä voi olla ääretön määrä

- Hyväksymisongelmassa voidaan siirtyä tarkastelemaan Chomskyn normaalimuotoon muutettua kielioppia, jolloin merkkijonon w , $|w| = n$, johdon pituus on $2n-1$
- Kontekstittomien kielioppien tyhjyytestaus voidaan ratkaista toisin
- Sen sijaan kontekstittomien kielioppien tuottamien kielten ekvivalenssin testaaminen on ratkeamaton ongelma
- Hyväksymisongelman ratkeavuuden perusteella kaikki kontekstittomat kielet ovat ratkeavia
- Näin ollen siis säännölliset kielet ovat (aito) osajoukko kontekstittomista, jotka ovat ratkeavia ja ratkeavat ovat aito osajoukko Turing-tunnistettavista kielistä

5.4 Äärettömistä joukoista

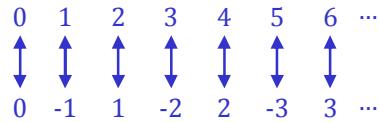
- Diagonalisoinnin kehitti 1873 Georg Cantor, joka tutki äärettömien joukkojen mahtavuuksien vertailua
- Äärellisillä joukoilla voimme laskea alkioden lukumäärät
- Ovatko numeroituvat joukot $\mathbb{N} = \{0, 1, 2, \dots\}$ ja $\mathbb{Z}^+ = \{1, 2, 3, \dots\}$ yhtä suuret?
 - \mathbb{N} on suurempi, koska se sisältää ylimääräisen alkion 0 ja muuten kaikki samat alkiot kuin \mathbb{Z}^+ (?)
 - Joukot ovat yhtä mahtavat, koska jokainen \mathbb{Z}^+ :n alkio voidaan kuvata \mathbb{N} :n alkioille kuvauksella $f(z) = z-1$ (?)

- Aiemmin jo käytimme tätä suuruusvertailua:
 - $|A| \leq |B|$ jos ja vain jos on olemassa injektio $f: A \rightarrow B$
 - injektio: $a_1 \neq a_2 \Rightarrow f(a_1) \neq f(a_2)$
- Aiemmin tarkastelimme joukkojen yhtäsuuruutta $|A| = |B|$ bijektiivisen $f: A \rightarrow B$ kuvauksen kautta
 - Bijektio = injektio + surjektio
 - Surjektio: $f(A) = B$ eli $\forall b \in B: \exists a \in A: b = f(a)$
- Bijektio asettaa joukkojen A ja B alkiot pareittain 1-1-suhteeseen
- $|\mathbb{N}| = |\mathbb{Z}^+|$
- Joukko, joka on yhtä suuri kuin \mathbb{N} on **numeroituva**

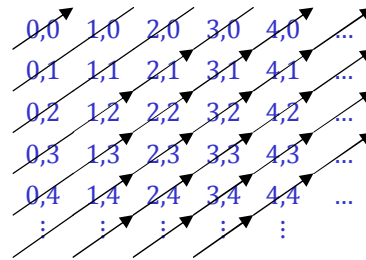
- Rationaalilukujen joukko

$$\mathbb{Q} = \{ m/n \mid m, n \in \mathbb{Z} \wedge n \neq 0 \}$$
- Kahden eri luonnollisen luvun välissä on aina äärettömästi rationaalilukuja
- Silti $|\mathbb{Q}| = |\mathbb{N}|$
- Kuvaus $f: \mathbb{Q} \rightarrow \mathbb{Z}^2, f(m/n) = (m, n)$
 - Luvuilla m ja n ei yhteisiä tekijöitä!
- Kuvaus f on funktio ja injektio, joten $|\mathbb{Q}| \leq |\mathbb{Z}^2| = |\mathbb{N}|$
- Toisaalta $\mathbb{N} \subseteq \mathbb{Q} \Rightarrow |\mathbb{N}| \leq |\mathbb{Q}|$
- $\therefore |\mathbb{Q}| = |\mathbb{N}|$

- $|\mathbb{Z}| = |\mathbb{N}|$



- $|\mathbb{N}^2| = |\mathbb{N}|$



- $|\mathbb{Z}^2| = |\mathbb{N}|$

- $|\mathbb{N}| = |\mathbb{N}^2|$
- $|\mathbb{N}^2| = |\mathbb{Z}^2|$ (helpohko)
- Seuraa transitivisuudesta

- $|\mathbb{N}| < |\mathbb{R}|$
- Sovelletaen Cantorin diagonalisointitekniikkaa välin $[0, 1[$ lukuihin x_1, x_2, x_3, \dots
- Olkoon väliin kuuluvien lukujen desimaalikehitelmät (ilman äärettömiä 9-jonoja)

$$x_i = \sum_{j \in \mathbb{Z}^+} d_{ij} \cdot 10^{-j}$$

- Muodostetaan uusi luku

$$x = \sum_{j \in \mathbb{Z}^+} d_j \cdot 10^{-j}$$

s.e.

$$d_j = \begin{cases} 0, & \text{jos } d_{jj} > 0 \\ 1, & \text{jos } d_{jj} = 0 \end{cases}$$

- Jos esim.

$$\begin{aligned}x_1 &= 0,23246\dots \\x_2 &= 0,30589\dots \\x_3 &= 0,21754\dots \\x_4 &= 0,05424\dots \\x_5 &= 0,99548\dots \\&\vdots\end{aligned}$$

niin $x = 0,01000\dots$

- Siis $x \neq x_i$ kaikilla i , joten oletus välin $[0, 1[$ lukujen numeroituvuudesta on väärä
- $|\mathbb{R}| = |[0, 1[\neq |\mathbb{N}|$

5.5 Universaalit Turingin koneet

- Aakkoston $\{0, 1\}$ **universaalikieli** U on

$$U = \{ \langle M, w \rangle \mid w \in L(M) \}.$$
- Kieli U sisältää tiedon kaikista aakkoston $\{0, 1\}$ RE-kielistä:
 - Olkoon $A \subseteq \{0, 1\}^*$ jokin RE-kieli ja M kielen A tunnistava standardimallinen Turingin kone. Tällöin

$$A = \{ w \in \{0, 1\}^* \mid \langle M, w \rangle \in U \}.$$
- Myös U on itsekin RE-kieli.
- Kielen U tunnistavia Turingin koneita sanotaan **universaaleiksi Turingin koneiksi**.

Lause 5.6 U on RE-kieli

Todistus. Kielen tunnistaa seuraava kolminauhainen Turingin kone M_U .

1. Ensin M_U tarkastaa, että nauhan 1 alussa olevassa syötteessä cw , c on kelvollinen Turingin koneen koodi. Jos syöte ei ole kelvollinen, M_U hylkää sen
2. Muutoin $w = a_1a_2\dots a_k \in \{0, 1\}^*$ kopioidaan nauhalle 2 muodossa $00010^{a_1+1}10^{a_2+1}1 \dots 10^{a_k+1}10000$
3. M_U :n on selvítettävä hyväksyisikö kone M ($c = \langle M \rangle$) syötteen w . Nauha 1 sisältää koneen M kuvauksen c , nauha 2 simuloi M :n nauhaa ja nauha 3 pitää yllä tietoa koneen M tilasta: $q_i \sim 0^{i+1}$

4. M_U toimii vaiheittain, simuloiden kussakin vaiheessa yhden koneen M siirtymän

1. Ensin M_U etsii nauhalta 1 M :n koodauksen sen kohdan, joka vastaa M :n simuloitua tilaa (nauha 3) ja nauhan 2 nauhapään kohdalla olevaa merkkiä
2. Olkoon löydetty koodinkohta

$$0^{i+1}10^{j+1}10^{r+1}10^{s+1}10^{t+1},$$

joka vastaa siirtymäntion δ sääntöä

$$\delta(q_p, a_j) = (q_r, a_s, \Delta_j).$$

nauha 3: $0^{i+1} \mapsto 0^{r+1}$

nauha 2: $0^{j+1} \mapsto 0^{s+1}$

Lisäksi nauhan 2 nauhapäätä siirretään yhden merkin koodin verran vasemmalle, jos $t = 0$, ja oikealle muuten

3. Kun nauhalla 1 ei ole yhtään simuloituaan tilaan q_i liittyvää koodia, kone M on tullut lopputilaan. Nyt $i = k+1$ tai $i = k+2$, missä q_k on viimeinen koodattu tila. Kone M_U siirtyy lopputilaan **accept** tai **reject**.

Selvästi kone M_U hyväksyy binäärijonon $\langle M, w \rangle$ jos ja vain jos $w \in L(M)$. \square

Lause 5.7 *Kieli U ei ole rekursiivinen*

Todistus. Oletetaan, että U :lla olisi totaalinen tunnistajakone M_U^T .

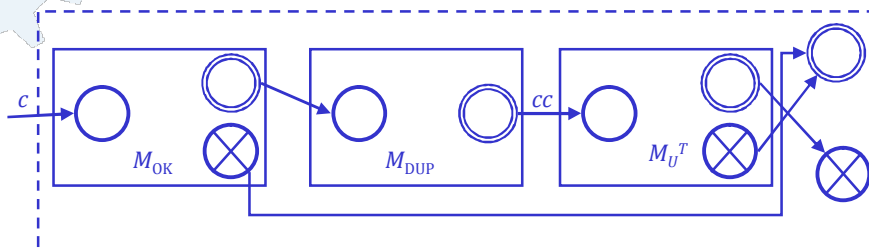
Tällöin "diagonaalikielille" D , joka ei ole RE-kieli, voitaisiin muodostaa tunnistajakone M_D koneesta M_U^T ja seuraavista totaalisista koneista.

- M_{OK} testaa onko syöteenä annettu merkkijono kelvollinen Turingin koneen koodi
- M_{DUP} duplikoi syötejonon c nauhalle: cc

Yhdistämällä koneet seuraavan kuvan esittämällä tavalla muodostuu kone M_D , joka on totaalinen, jos M_U^T on. Lisäksi

$$\begin{aligned} c \in L(M_D) \\ \Leftrightarrow c \notin L(M_{OK}) \vee cc \notin L(M_U^T) \\ \Leftrightarrow c \notin L(M_C) \\ \Leftrightarrow c \in D = \{c \mid c \notin L(M_C)\} \end{aligned}$$

Lemman 5.5 perusteella kieli D ei ole rekursiivinen, joten oheinen tulos on ristiriita. Näin ollen oletuksen tulee olla väärä ja kielen U tunnistavaa totaalista konetta M_U^T ei voi olla olemassa. \square



Diagonaalikielen tunnistava Turingin kone

Seuraus 5.8 $\bar{U} = \{ \langle M, w \rangle \mid w \notin L(M) \}$ ei ole RE-kieli

Todistus. $\bar{U} = \bar{U} \cup \text{ERR}$, missä ERR on helposti tunnistettava rekursiivinen kieli:

$\text{ERR} = \{ x \in \{0, 1\}^* \mid x \text{ ei sisällä alkuosanaan kelvollista Turingin koneen koodia } \}$.

Tehdään vastaoletus: \bar{U} on RE-kieli

- Lauseen 5.2 perusteella $\bar{U} \cup \text{ERR} = \bar{U}$ on RE-kieli
- U on RE-kieli (5.6) ja edellisen mukaan \bar{U} on RE-kieli, joten lauseen 5.3 perusteella U on rekursiivinen

Tämä on ristiriita lauseen 5.7 kanssa, joten vastaoletus ei päde. Siis \bar{U} ei ole RE-kieli. \square

5.6 Pysähtymisongelma

- Turingin koneiden **pysähtymisongelma** (halting problem) on kysymys

Pysähtyykö annettu Turingin kone M syötteellä w ?

- Ongelma on ratkeamaton. Jos se voitaisiin ratkaista, niin universaalikieli voitaisiin helposti tunnistaa totaalisesti

Lause 4.11 $H = \{ \langle M, w \rangle \mid M \text{ pysähtyy syötteellä } w \}$ on RE-kieli, mutta ei rekursiivinen

Todistus. H on RE-kieli: Lauseen 5.6 universaalikoneesta M_U on helppo muokata kone, joka simuloi koneen M laskentaa syötteellä w ja pysähtyy hyväksyvään lopputilaan, jos ja vain jos simuloitu laskenta pysähtyy.

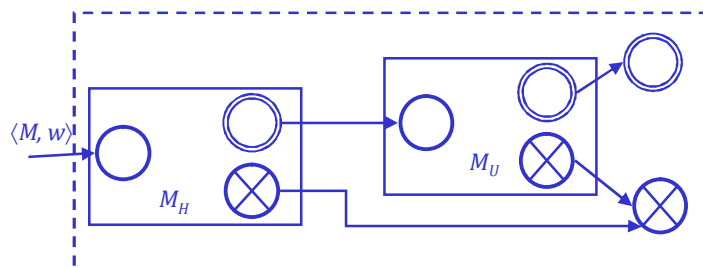
H ei ole rekursiivinen: vastaoletus: $H = L(M_H)$ totaalisella Turingin koneella M_H .

Nyt kielelle U voidaan muodostaa totaalinen tunnistaja yhdistämällä koneet M_U ja M_H seuraavan kuvan esittämällä tavalla.

Tällaisen koneen olemassaolo on ristiriita Lauseen 5.7 kanssa. Näin ollen vastaoletus on väärä ja H ei ole rekursiivinen. \square

Seuraus 5.9 $\hat{H} = \{ \langle M, w \rangle \mid M \text{ ei pysähdy syötteellä } w \}$ ei ole RE-kieli

Todistus. Kuten seuraus 5.8. \square



Universaalikielen U tunnistava totaalinen Turingin kone

5.7 Chomskyn kieliluokat

- Formaali kieli L voidaan tunnistaa Turingin koneella, jos ja vain jos se voidaan tuottaa rajoittamattomalla kielioilla
- Näin ollen rajoittamattomilla kielioilla tuotettavat kielet ovat RE-kielet. Toisaalta tämä on Chomskyn kielihierarkian luokka 0
- Chomskyn luokka 1 puolestaan on kontekstiset kielet. Voidaan osoittaa, että ne kaikki ovat rekursiivisia
- On olemassa rekursiivisia kieliä, joita ei voida tuottaa kontekstisilla kielioilla

☆ $D, \bar{U}, \hat{H}, \dots$

0: RE-kielet

Rekursiiviset kielet

U, H, \dots

1: Kontekstiset kielet

2: Kontekstittomat kielet

3: Säännölliset kielet

Äärelliset kielet

esim. $\{a^k b^k c^k \mid k \geq 0\}$

esim. $\{a^k b^k \mid k \geq 0\}$

esim. $\{a^k \mid k \geq 0\}$

Pysähtymisongelma ohjelmointikielellä

Turingin koneiden ja ohjelmointikielten vastaavuus:

Koneet ~ ohjelmointikieli
 Kone ~ ohjelma
 Koneen koodi ~ ohjelman konekieliesitys
 Universaalikone ~ konekielen tulkki

Esimerkiksi pysähtymisongelman ratkeamattomuuden ohjelmointikielitulkinta on:

“Ei ole olemassa totaalista Java-menetelmää, joka ratkaisisi, pysähtyykö annettu Java-menetelmä M annetulla syötteellä w ”.

Oletetaan, että on olemassa totaalinen Java-menetelmä h , joka palauttaa arvon `true` jos merkkijonon m esittämä menetelmä pysähtyy syötteellä w , ja `false` muuten:

```
boolean h(String m, String w)
```

Nyt voidaan toteuttaa menetelmä `hHat`

```
boolean hHat( String m )
{ if (h(m,m))
  while (true) ;}
```

Olkoon H menetelmän `hHat` merkkijonoesitys. `hHat` toimii seuraavasti:

`hHat(H)` pysähtyy $\Leftrightarrow h(H,H) = \text{false} \Leftrightarrow hHat(H)$ ei pysähdy

6. Palautuvuus

- Pysähtymisongelman ratkeamattomuuden todistus on esimerkki palautuksesta
- Jos pysähtymisongelma olisi ratkeava, niin myös universaalikieli olisi rekursiivinen
- Palautus itsessään ei kerro mitään kummankaan ongelman ratkeavuudesta, niillä vain on tämä kytkentä
- Tiedämme muuta kautta, että universaalikieli ei ole rekursiivinen
- Ongelman A ollessa palautettavissa ongelmaan B , A :n ratkaiseminen ei voi olla vaikeampaa kuin B :n
- Jos voimme palauttaa ratkeamattoman ongelman toiseen ongelmaan, on jälkimmäisenkin oltava ratkeamaton

Epätyhjyysongelma

Seuraava päätösongelma on ratkeamaton:

”Hyväksyykö annettu Turingin kone yhtään syötemerkkijonoa?”

$$NE = \{c \in \{0, 1\}^* \mid L(M_c) \neq \emptyset\}$$

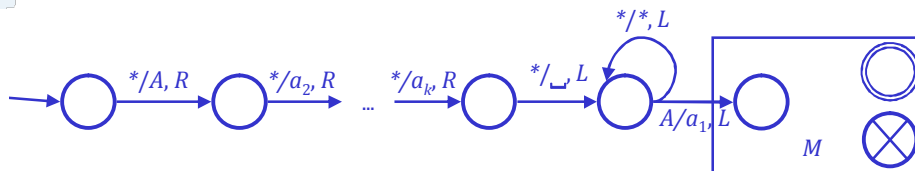
Lause (5.2) NE on RE -kieli, mutta ei rekursiivinen

Todistus. NE on RE -kieli osoitetaan harjoituksissa.

- Oletetaan sitten, että NE :llä olisi totaalinen tunnistajakone M_{NE}^T
- Tämän avulla voitaisiin muodostaa totaalinen tunnistajakone kielelle U
- Olkoon M mv. Turingin kone, jonka toimintaa syötteellä w tarkastellaan

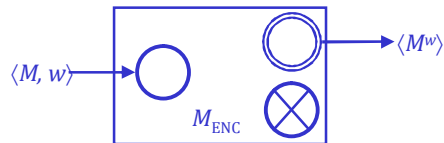
- Merkitään M^w :llä konetta, joka korvaa kulloisenkin todellisen syötteen aina merkkijonolla $w = a_1a_2...a_k$ ja toimii sitten kuten M
- Koneen M^w toiminta ei riipu laisinkaan sen todellisesta syötteestä.
 - Se joko hylkää tai hyväksyy kaikki syötteet:

$$L(M^w) = \begin{cases} \{0,1\}^* & \text{jos } w \in L(M) \\ \emptyset & \text{jos } w \notin L(M) \end{cases}$$

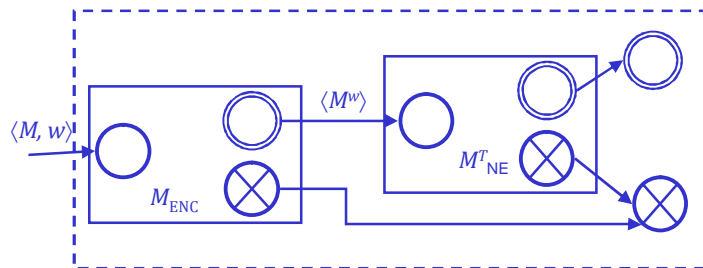


Turingin kone M^w

- Olkoon M_{ENC} kone, joka
 - saa syötteenään Turingin koneen M koodin $\langle M \rangle$ sekä binäärijonon w katenaation, $\langle M, w \rangle$, ja
 - jättää tuloksena nauhalle koneen M^w koodin $\langle M^w \rangle$



- Nyt kone M_{ENC} ja kielen NE totaalinen tunnistajakone M_{NE}^T yhdistämällä saataisiin seuraavan kuvan esittämä Turingin kone M_U^T



Universaalikielen U tunnistava totaalinen Turingin kone M_U^T

- M_U^T on totaalinen jos M_{NE}^T on, ja $L(M_U^T) = U$, koska

$$\begin{aligned} \langle M, w \rangle \in L(M_U^T) \\ \Leftrightarrow \langle M^w \rangle \in L(M_{NE}^T) = NE \\ \Leftrightarrow L(M^w) \neq \emptyset \\ \Leftrightarrow w \in L(M) \\ \Leftrightarrow \langle M, w \rangle \in U \end{aligned}$$

- Lauseen 5.7 perusteella U ei kuitenkaan ole rekursiivinen, joten koneen M_U^T olemassaolo on ristiriita
- Näin ollen kielellä NE ei voi olla totaalista tunnustajakonetta M_{NE}^T . Olemme siis todistaneet, että kieli NE ei ole rekursiivinen

□