3. Turingin koneet

Turingin kone on alkuaan matemaattisen logiikan tarpeisiin kehitelty laskennan malli. Tarkoituksena oli vangita mahdollisimman laajasti, millaisia asioita voidaan (periaatteessa) laskea "mekaanisesti". Malli on sittemmin osoittautunut sopivaksi myös "oikeiden" tietokoneiden ymmärtämiseen.

Tämän luvun jälkeen opiskelija

- osaa esittää yksinkertaisia algoritmeja täsmällisesti käyttäen Turingin konetta ja sen muunnelmia
- hieman monimutkaisemmille algoritmeille osaa kuvata periaatetasolla toteutuksen Turingin koneella
- tuntee Churchin-Turingin teesin ja osaa sen avulla selittää Turingin koneen yhteyden yleiseen algoritmin käsitteeseen.

Turingin koneen rajoituksia tarkastellaan lähemmin seuraavassa luvussa.

Turingin kone [Sipser luku 3.1]

Turingin kone (Turing machine, TM) on automaatti, jossa on rajoittamaton määrä muistia. Toisin kuin pinoautomaatissa, muistialkioita voi käsitellä mielivaltaisessa järjestyksessä.

Turingin koneen peruskomponentit ovat

1. äärellinen joukko *tiloja* (state), kuten DFA:ssa ja PDA:ssa,
2. rajoittamattoman pituinen *nauha* (tape), joka aluksi sisältää syötteen ja laskennan aikana toimii apumuistina ja
3. liikuteltava *nauhapää* (tape head, read/write head), joka osoittaa seuraavaksi vuorossa olevaa symbolia nauhalla.

Turingin koneen yksityiskohdat voidaan määritellä monella eri tavalla, ja laskentavoimaltaan samaan lopputulokseen voidaan päätyä myös aivan toisennäköisistä lähtökohdista. Otamme tässä perustaksi mahdollisimman yksinkertaisen mallin.

Kuten muutkin käsittelemämme automaatit, Turingin kone saa syötteenä merkkijonon, jonka sitten hyväksyy tai hylkää:

1. Aluksi syöte on nauhan alussa ja nauhan loppu sisältää pelkkiä *tyhjämerkkejä* ␣ (blank). Nauhapää osoittaa nauhan alkuun.
2. Yhdessä laskenta-askelessa
   - kone lukee nauhapään alla olevan symbolin,
   - valitsee seuraavan tilan,
   - *kirjoittaa* nauhapään kohdalle uuden symbolin (vanha häviää) ja
   - siirtää nauhapäätä *vasemmalle tai oikealle*.
3. Tilojen joukossa on *hylkäävä* ja *hyväksyvä* lopputila, joihin päätyminen lopettaa laskennan välittömästi.

Jos laskenta ei koskaan päädy hyväksyvään tai hylkäävään tilaan, se on *silmukassa*.

Tarkastellaan esimerkkinä kielen $A = \{ \mathrm{a}^n\mathrm{b}^n\mathrm{c}^n \mid n \in \mathbb{N} \}$ tunnistamista Turingin koneella. Periaatteena on pyöriä silmukassa, jonka jokaisella kierroksella nauhalta *yliviivataan* yksi kappale kutakin syötemerkkiä:

1. Kelaa nauhaa oikealle, kunnes löytyy jokin muu merkki kuin yliviivattu a. Jos löytyi muuta kuin a, siirry kohtaan 5. Muuten viivaa yli löytynyt a.
2. Kelaa nauhaa oikealle, kunnes löytyy jokin muu merkki kuin a tai yliviivattu b. Jos löytyi muuta kuin b, niin *hylkää*. Muuten viivaa yli löydetty b.
3. Kelaa nauhaa oikealle, kunnes löytyy jokin muu merkki kuin b tai yliviivattu c. Jos löytyi muuta kuin c, niin *hylkää*. Muuten viivaa yli löytynyt c.
4. Kelaa nauha alkuun ja siirry kohtaan 1.
5. Jos nauhalla on jäljellä yliviivaamattomia merkkejä, niin *hylkää*, muuten *hyväksy*.

Huomaa, että tähän ei pinoautomaatti pysty (kieli A ei ole yhteydetön).

Täsmällisemmin Turingin kone on seitsikko $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, missä

1. Q on äärellinen tilajoukko,
2. Σ on syöteaakkosto, joka **ei** sisällä tyhjämerkkiä $\sqcup$,
3. Γ on *nauha-aakkosto*, jolle $\sqcup \in \Gamma$ ja $\Sigma \subset \Gamma$,
4. $\delta \colon Q' \times \Gamma \to Q \times \Gamma \times \{\, \mathsf{L}, \mathsf{R} \,\}$ on siirtymäfunktio, missä $Q' = Q - \{\, q_{\text{accept}}, q_{\text{reject}} \,\}$,
5. $q_0 \in Q$ on alkutila,
6. $q_{\text{accept}} \in Q$ on *hyväksyvä* tila ja
7. $q_{\text{reject}} \in Q$ on *hylkäävä* tila, jolla $q_{\text{reject}} \neq q_{\text{accept}}$.

Perustulkinta: Jos $\delta(q, a) = (q', b, D)$, niin tilasta q symbolillä a kone siirtyy tilaan q', kirjoittaa symbolin b (joka korvaa vanhan symbolin a) ja siirtää nauhapäätä yhden askelen suuntaan D (missä $\mathsf{L} =$ vasen ja $\mathsf{R} =$ oikea).

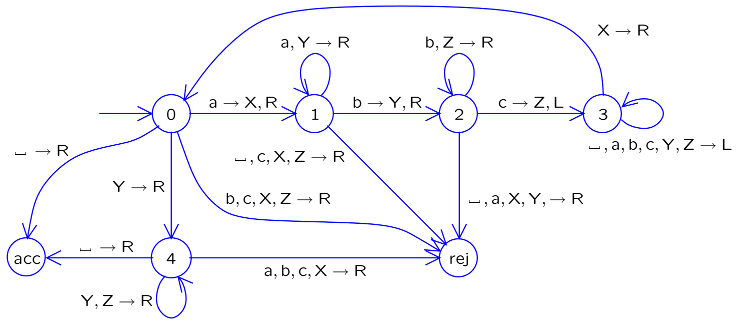
Jos laskenta päätyy tilaan q_{accept} tai q_{reject}, se pysähtyy välittömästi ja syöte on vastaavasti hyväksytty tai hylätty.

Esitetään kielen $A = \{\, \mathsf{a}^n \mathsf{b}^n \mathsf{c}^n \mid n \in \mathbb{N} \,\}$ tunnistava kone tässä formalismissa. Syöteaakkosto on siis $\Sigma = \{\, \mathsf{a}, \mathsf{b}, \mathsf{c} \,\}$. Nauha-aakkostoksi valitsemme $\Gamma = \{\, \sqcup, \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X}, \mathsf{Y}, \mathsf{Z} \,\}$, missä X, Y ja Z esittävät yliviivattuja a-, b- ja c-symboleja.

Siirtymäfunktio voidaan esittää kaaviona, jossa

- $q \xrightarrow{a \to b, D} q'$ tarkoittaa $\delta(q, a) = (q', b, D)$ ja
- $q \xrightarrow{a,b,c,d \to D} q'$ tarkoittaa $\delta(q, x) = (q', x, D)$ kun $x \in \{\, a, b, c, d \,\}$.

Kaaviota voidaan yksinkertaistaa jättämällä pois tila q_{reject} ja siihen johtavat siirtymät, mutta tässä näin ei ole tehty. (Kaavio seuraavalla sivulla.)



Turingin koneen *tilanne* (configuration) koostuu kolmesta komponentista:

1. koneen tila,
2. nauhan sisältö ja
3. nauhapään sijainti.

Koodaamme tilanteen merkkijonoksi uqv, missä $q \in Q$, $u = u_1 \dots u_m \in \Gamma^*$ ja $v = v_1 \dots v_n \in \Gamma^*$ joillain $m, n \in \mathbb{N}$. Tulkinta:

1. kone on tilassa q,
2. nauhan sisältö on $u_1 \dots u_m v_1 \dots v_n \sqcup \sqcup \sqcup \dots$ ja
3. nauhapää osoittaa symbolia v_1.

Koneen *alkutilanne* syötteellä w on $q_0 w$.

Koneen *lopputilanteita* eli *pysähtymistilanteita* (halting configuration) ovat

- *hyväksyvät* tilanteet $u q_{\text{accept}} v$ ja
- *hylkäävät* tilanteet $u q_{\text{reject}} v$

kaikilla $u, v \in \Gamma^*$.

Jos tilanteesta uqv kone menisi seuraavaksi tilanteeseen $u'q'v'$, niin tilanne uqv *johtaa suoraan* (yields) tilanteen $u'q'v'$, mitä merkitään

$$uqv \vdash u'q'v'.$$

Esitetään siirtymäfunktion semantiikka tätä formalismia käyttäen. Olkoot $u, v \in \Gamma^*$ ja $a, b \in \Gamma$.

- Jos $\delta(q, a) = (q', b, \mathsf{R})$, niin $uqav \vdash ubq'v$.
- Edellisen erikoistapaus: jos $\delta(q, \llcorner\!\lrcorner) = (q', b, \mathsf{R})$, niin $uq \vdash ubq'$ (sillä uq on sama tilanne kuin $uq\llcorner\!\lrcorner$).
- Jos $\delta(q, a) = (q', b, \mathsf{L})$, niin $ucqav \vdash uq'cbv$ kaikilla $c \in \Gamma$.
- Jos $\delta(q, a) = (q', b, \mathsf{L})$, niin $qav \vdash q'bv$ (ts. jos nauhapää yrittäisi siirtyä alkukohdasta vasemmalle, se jää paikalleen).

Esimerkkinä sivun 231 koneen hyväksyntä syötteelle aabbcc:

$$\begin{array}{lll}
q_0\mathsf{aabbcc} & \vdash & \mathsf{X}q_1\mathsf{abbcc} \\
& \vdash & \mathsf{Xa}q_1\mathsf{bbcc} \\
& \vdash & \mathsf{XaY}q_2\mathsf{bcc} \\
& \vdash & \mathsf{XaYb}q_2\mathsf{cc} \\
& \vdash & \mathsf{XaY}q_3\mathsf{bZc} \\
& \vdash & \mathsf{Xa}q_3\mathsf{YbZc} \\
& \vdash & \mathsf{X}q_3\mathsf{aYbZc} \\
& \vdash & q_3\mathsf{XaYbZc} \\
& \vdash & \mathsf{X}q_0\mathsf{aYbZc} \\
& \vdash & \mathsf{XX}q_1\mathsf{YbZc} \\
& \vdash & \mathsf{XXY}q_1\mathsf{bZc} \\
& \vdash & \mathsf{XXYY}q_2\mathsf{Zc} \\
& \vdash & \mathsf{XXYYZ}q_2\mathsf{c} \\
& \vdash & \mathsf{XXYY}q_3\mathsf{ZZ} \\
& \vdash & \mathsf{XXY}q_3\mathsf{YZZ} \\
& \vdash & \mathsf{XX}q_3\mathsf{YYZZ} \\
& \vdash & \mathsf{X}q_3\mathsf{XYYZZ} \\
& \vdash & \mathsf{XX}q_0\mathsf{YYZZ} \\
& \vdash & \mathsf{XXY}q_4\mathsf{YZZ} \\
& \vdash & \mathsf{XXYY}q_4\mathsf{ZZ} \\
& \vdash & \mathsf{XXYYZ}q_4\mathsf{Z} \\
& \vdash & \mathsf{XXYYZZ}q_4\llcorner\!\lrcorner \\
& \vdash & \mathsf{XXYYZZ}\llcorner\!\lrcorner\, q_{\mathsf{accept}}\llcorner\!\lrcorner
\end{array}$$

Jos on olemassa tilanteet $u_1q_1v_1, \ldots, u_nq_nv_n$, missä $u_iq_iv_i \vdash u_{i+1}q_{i+1}v_{i+1}$ kaikilla $1 \leq i \leq n-1$, sanomme että tilanne $u_1q_1v_1$ *johtaa* tilanteeseen $u_nq_nv_n$ ja merkitsemme

$$u_1q_1v_1 \vdash^* u_nq_nv_n.$$

Jos $q_0w \vdash^* uq_{\mathsf{accept}}v$ joillain $u, v \in \Gamma^*$, niin kone *hyväksyy* merkkijonon w.

Jos $q_0w \vdash^* uq_{\mathsf{reject}}v$ joillain $u, v \in \Gamma^*$, niin kone *hylkää* merkkijonon w.

Jos kone hyväksyy tai hylkää merkkijonon w, sanomme, että se *pysähtyy* (halts) syötteellä w. Muuten se *jää silmukkaan* (loops), mikä voi ilmetä mielivaltaisen monimutkaisena loputtomana tilannejonona.

Turingin koneen M *tunnistama kieli* on

$$L(M) = \{\, w \in \Sigma^* \mid M \text{ hyväksyy } w\text{:n} \,\}.$$

Jos $A = L(M)$, kieli A on *Turing-tunnistettava* tai lyhyesti *tunnistettava* (recognizable). Historiallisista syistä Turing-tunnistettavia kieliä sanotaan myös *rekursiivisesti numeroituviksi* (recursively enumerable); nimitys liittyy vain epäsuorasti ohjelmointikielissä käytettävään rekursioon.

Huom. komplementti $\overline{L(M)}$ sisältää ne merkkijonot, joilla M hylkää *tai* jää silmukkaan.

Jos M pysähtyy kaikilla syötteillä, se on *ratkaisija* (decider). Ratkaisijoita sanotaan myös *totaalisiksi* Turingin koneiksi. Jos $A = L(M)$, missä M on ratkaisija, niin A on *Turing-ratkeava* tai lyhyesti *ratkeava* (decidable). Ratkeavia kieliä sanotaan myös *rekursiivisiksi* (recursive).

(Osoitamme myöhemmin, että A on ratkeava, jos ja vain jos sekä A että $\overline{A}$ ovat tunnistettavia.)

Esimerkki 3.1: Kieli $B = \{\, ww \mid w \in \{\, 0,1 \,\}^* \,\}$ voidaan tunnistaa seuraavalla periaatteella:

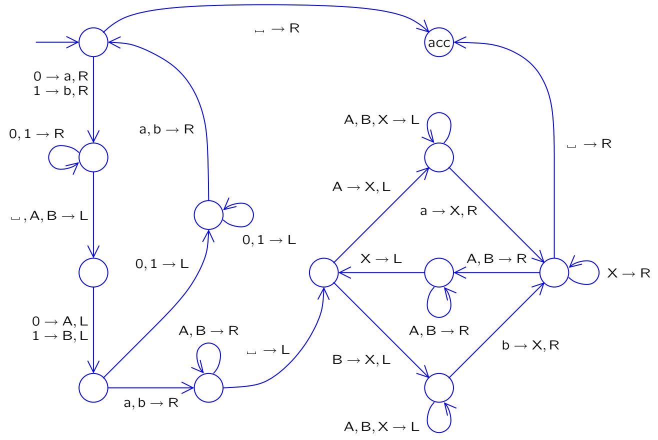
Vaihe I: Merkkijonon alkupuoliskossa korvaa $0 \mapsto$ a ja $1 \mapsto$ b ja loppupuoliskossa $0 \mapsto$ A ja $1 \mapsto$ B. Esim. 00101100 tulee muotoon aabaBBAA. Parittoman mittaiset syötteet hylätään. Toteutus:

1. Vaihda nauhalta ensimmäinen 0- tai 1-symboli vastaavasti a:ksi tai b:ksi.
2. Vaihda nauhalta viimeinen 0- tai 1-symboli vastaavasti A:ksi tai B:ksi. Jos nollia tai ykkösiä ei löytynyt, *hylkää*.
3. Jos nollia tai ykkösiä vielä on, palaa kohtaan 1. Muuten siirry vaiheeseen II.

Vaihe II: Lopusta alkaen vertaa, että kutakin A-symbolia vastaa a-symboli ja B-symbolia vastaa b-symboli. Toteutus:

4. Etsi nauhan oikeanpuoleisin A tai B ja viivaa yli.
   - Jos se oli A, etsi oikeanpuoleisin a tai b. Jos tämä on a, viivaa yli. Muuten hylkää.
   - Jos se oli B, etsi oikeanpuoleisin a tai b. Jos tämä on b, viivaa yli. Muuten hylkää.
5. Jos A:t ja B:t loppuivat, *hyväksy*. Muuten palaa kohtaan 4.

Alla olevaan kaavioon ei selkeyden vuoksi ole piirretty hylkäävää tilaa q_{reject}. Kaikki kuvasta puuttuvat siirtymät vievät siihen.



Kuten edellisestä esimerkistä havaitaan, Turingin koneista (kuten muistakin automaateista) tulee nopeasti hyvin monimutkaisia. Monimutkaisten Turingin koneiden kaavioesitysten piirtäminen ei ole kiinnostavaa tämän kurssin kannalta (eikä yleensä muutenkaan).

Turingin kone on tarkoitettu yleiseksi laskentaformalismiksi, jota käyttäen voidaan esittää mikä tahansa algoritminen laskenta. Mallin motivaatio on siis erilainen kuin tietoisesti rajatuissa malleissa (DFA, PDA jne.).

Seuraavassa esitellään esimerkinomaisesti Turingin koneiden "ohjelmointitekniikoita" tavoitteena vakuuttua, että *mikä tahansa algoritmi* voidaan koodata Turingin koneeksi.

Kun olemme vakuuttuneet tästä, voimme tyytyä kirjoittamaan esim. pseudokoodia ja unohtaa Turingin kone -toteutuksen yksityiskohdat.

(Jos taas emme olisi vakuuttuneet tästä, Turingin koneiden piirtelemisessä ei kuitenkaan olisi mitään järkeä, koska tällöin emme olisi vakuuttuneet, että Turingin kone kelpaa malliksi sille, mitä haluamme mallintaa.)

Esimerkki 3.2: [Sipser Ex. 3.11] Tarkastellaan kertolaskun tarkastamista eli kielen

$$C = \{\, \mathrm{a}^i \mathrm{b}^j \mathrm{c}^k \mid i, j, k \geq 1 \text{ ja } k = i \cdot j \,\}$$

tunnistamista. Turingin kone toimii seuraavasti:

1. Selaa syöte vasemmalta oikealle ja tarkista, että se on muotoa $\mathrm{a}^+\mathrm{b}^+\mathrm{c}^+$. Jos ei ole, niin *hylkää*.

2. Palauta nauhapää nauhan alkuun.

3. Viivaa yli vasemmanpuoleisin a ja selaa seuraavaan b:hen. Kelaa edestakaisin b- ja c-osien välillä merkaten aina yksi b ja yksi c kerrallaan. Jos c:t loppuvat, *hylkää*. Muuten kun b:t on kaikki merkattu, siirry kohtaan 4.

4. Poista merkit kaikista b:istä. Jos a:ita on jäljellä, mene kohtaan 3. Muuten jos kaikki c:t on merkattu, niin *hyväksy*, muuten *hylkää*.

Joitain toteutusteknisiä yksityiskohtia:

- Kohta 1 on helppo, koska koneen riittää toimia kuin äärellinen automaatti.
- Kohtaa 2 varten nauhan alkukohta pitää merkata esim. vaihtamalla ensimmäinen a erikoismerkiksi.
- Kohdassa 3 symbolin b "merkkaaminen" tapahtuu esim. kirjoittamalla sen tilalle b̆, missä b̆ on nauha-aakkostoon lisätty uusi symboli.

Jatkossa oletamme, että Turingin kone osataan tarvittaessa toteuttaa edellisen tapaisen (tai vielä korkeammantasoisen) kuvauksen perusteella. □

Olemme siis todenneet, että edellä esitetyt kielet A, B ja C ovat Turing-tunnistettavia. Koska kaikki esitetyt koneet pysähtyvät kaikilla syötteillä, kielet ovat vieläpä ratkeavia.

Turingin koneen muunnelmia [Sipser luku 3.2]

Turingin koneen yksityiskohdat voidaan määritellä monella tavalla, jotka antavat samat tunnistettavien ja ratkeavien kielten luokat.

Muunnelma 1: Sallitaan siirtymäfunktion olla muotoa $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{\,\mathsf{L}, \mathsf{R}, \mathsf{S}\,\}$. "Suunta" S tarkoittaa, että nauhapää pysyy paikoillaan (stay put). Koska kukin S-siirtymä voidaan korvata peräkkäisillä R- ja L-siirtymillä (lisäten tarvittaessa uusi tila), niiden salliminen ei lisäisi mallin laskentavoimaa.

Muunnelma 2: Nauha jatkuu äärettömän pitkälle myös vasemmalle. Syötteellä $w = w_1 \ldots w_n$ nauhan sisältö on aluksi $\ldots \sqcup\sqcup\sqcup w_1 \ldots w_m \sqcup\sqcup\sqcup \ldots$ ja nauhapää osoittaa symbolia w_1. Kahteen suuntaan äärettömän nauhan sisältö $\ldots a_{-2}a_{-1}a_0a_1a_2 \ldots$, missä $a_i \in \Gamma$, voidaan esittää "perusmallissa" muodossa $(a_1, a_0)(a_2, a_{-1})(a_3, a_{-2}) \ldots$, missä nauha-aakkostona on Γ^2. Tästä on helppo nähdä, että mallin laskentavoima ei taaskaan kasva.

Tärkeämpiä muunnelmia ovat *moninauhaiset* ja *epädeterministiset* koneet.

Moninauhainen Turingin kone (multitape TM)

Moninauhaisessa Turingin koneessa on useita nauhoja ja kullakin oma nauhapäänsä. Nauhoille voidaan kirjoittaa ja nauhapäitä siirtää toisistaan riippumatta.

Siirtymäfunktio k-nauhaiselle Turingin koneelle on muotoa

$$\delta\colon Q' \times \Gamma^k \to Q \times \Gamma^k \times \{\,\mathsf{L}, \mathsf{R}, \mathsf{S}\,\}^k.$$

Siirtymä $\delta(q, a_1, \ldots, a_k) = (q', b_1, \ldots, b_k, D_1, \ldots, D_k)$ tarkoittaa, että jos

- kone on tilassa q ja
- kaikilla $1 \le i \le k$ nauhapään i kohdalla on symboli a_i

niin

- kone siirtyy tilaan q' ja
- kaikilla $1 \le i \le k$
  - nauhalle i kirjoitetaan symboli b_i ja
  - nauhapää i siirtyy suuntaan D_i.

Alkutilanteessa ykkösnauha sisältää syötteen ja muut ovat tyhjiä.

Koneet ovat *ekvivalentit*, jos ne tunnistavat saman kielen.

Lause 3.3: [Sipser Thm. 3.13] Jokaiselle moninauhaiselle Turingin koneelle on olemassa ekvivalentti yksinauhainen Turingin kone.

Todistus: Olkoon M k-nauhainen kone, jonka nauha-aakkosto on Γ. Muodostamme ekvivalentin yksinauhaisen koneen S, jonka nauha-aakkostoksi tulee

$$\Gamma \cup \{\, \breve{a} \mid a \in \Gamma \,\} \cup \{\, \# \,\}.$$

Tässä $\breve{a} \notin \Gamma$ on symbolin a merkattu versio kaikilla $a \in \Gamma$, ja $\# \notin \Gamma$ uusi välimerkki.

Perusidea esimerkin valossa: Oletetaan, että kolminauhaisen koneen nauhojen sisällöt ovat

$$\begin{array}{ll} \text{nauha 1:} & \mathsf{a}\underline{\mathsf{a}}\mathsf{bB}\sqcup\sqcup\sqcup\ldots \\ \text{nauha 2:} & \underline{\sqcup}\sqcup\sqcup\ldots \\ \text{nauha 3:} & \mathsf{001}\underline{\mathsf{0}}\mathsf{01}\sqcup\sqcup\sqcup\ldots \end{array}$$

missä nauhapään sijainti on alleviivattu. Tämä esitetään yhdellä nauhalla muodossa

$$\text{nauha:}\quad \mathsf{a}\breve{\mathsf{a}}\mathsf{bB}\#\breve{\sqcup}\#\mathsf{001}\breve{\mathsf{0}}\mathsf{01}\#\sqcup\sqcup\sqcup\ldots \qquad .$$

Tarkemmin S on kone, joka syötteellä $w = w_1 \ldots w_n$ toimii seuraavasti:

1. Alusta nauhan sisällöksi $\#\breve{w}_1 w_2 w_3 \ldots w_n \#\breve{\sqcup}\#\breve{\sqcup}\# \ldots \# \sqcup \sqcup \sqcup \ldots$ missä #-merkkejä on $k+1$ kappaletta.
2. Simuloi yksi koneen M laskenta-askel seuraavasti:
   (a) Selaa koko nauha ensimmäisestä viimeiseen #-merkkiin. Pane muistiin, mitkä merkatut symbolit ($\breve{a}$) esiintyivät ja missä järjestyksessä. Vaihtoehtoja on äärellinen määrä $|\Gamma|^k$, joten ne voidaan koodata koneen S tiloihin.
   (b) Päätä koneen M uusi tila siirtymäfunktion mukaisesti.
   (c) Selaa nauha takaisin alkuun ja samalla muuta merkittyjä symboleita koneen M siirtymäfunktion mukaisesti. Jos jokin nauhapään merkki siirtyy oikealle #-merkin päälle, tee tilaa siirtämällä koko nauhan loppuosa askel oikealle.
3. Jos koneen M uusi tila on hyväksyvä, niin *hyväksy*. Jos koneen M uusi tila on hylkäävä, niin *hylkää*. Muuten jatka kohdasta 2.

$\square$

Korollaari 3.4: [Sipser Cor. 3.15] Kieli on Turing-tunnistettava (-ratkeava), jos ja vain jos jokin moninauhainen Turingin kone tunnistaa (vast. ratkaisee) sen.

Todistus: "Vain jos" -suunta on ilmeinen. "Jos"-suunta seuraa edellisestä konstruktiosta, jossa simuloiva S pysähtyy, jos ja vain jos simuloitava M pysähtyy. $\square$

Reunahuomautus: Oletetaan, että k-nauhainen kone M syötteellä w pysähtyy T askelessa. Jos $T \geq w$, millekään nauhalle ei tule yli T symbolia. Siis simuloivassa koneessa S nauhalle tulee $O(kT)$ symbolia.

Yhden simulointiaskelen toteuttamiseksi S joutuu pahimmillaan siirtämään nauhan sisältöä oikealle k kertaa. Siis yksi koneen M askel vie pahimmillaan $O(k^2T)$ koneen S askelta, ja koko laskenta $O(k^2T^2)$ askelta.

Sanomme, että Turingin kone toimii *polynomisessa ajassa*, jos syötteen x käsittelemiseen kuluvien laskenta-askelien määrä on $O(|x|^k)$ jollain $k \in \mathbb{N}$. Näemme siis, että jos kieli voidaan tunnistaa *polynomisessa ajassa* moninauhaisella koneella, se voidaan tunnistaa polynomisessa ajassa myös yksinauhaisella koneella.

Epädeterministinen Turingin kone

Kuten muidenkin automaattien tapauksessa, epädeterministisessä Turingin koneessa siirtymäfunktio antaa *joukon* mahdollisia seuraajia. Se siis on tyyppiä

$$\delta \colon Q' \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\, \mathsf{L}, \mathsf{R} \,\}).$$

Merkintä $\vdash$ yleistetään vastaavasti, esim. $uqav \vdash ubq'b$ pätee, jos $(q', b, \mathsf{R}) \in \delta(q, a)$. Epädeterministinen kone N hyväksyy syötteen, jos *jokin* mahdollinen laskenta johtaa hyväksyvään tilanteeseen.

Huom. Määritelmässä on oleellinen epäsymmetria. Kun $A = L(N)$, niin

- jos $w \in A$, niin *ainakin yksi* laskenta hyväksyy, mutta muut saavat johtaa hylkäämiseen tai silmukkaan;
- jos $w \notin A$, niin *mikään* laskenta ei johda hyväksymiseen, vaan kaikkien on johdettava hylkäämiseen tai silmukkaan.

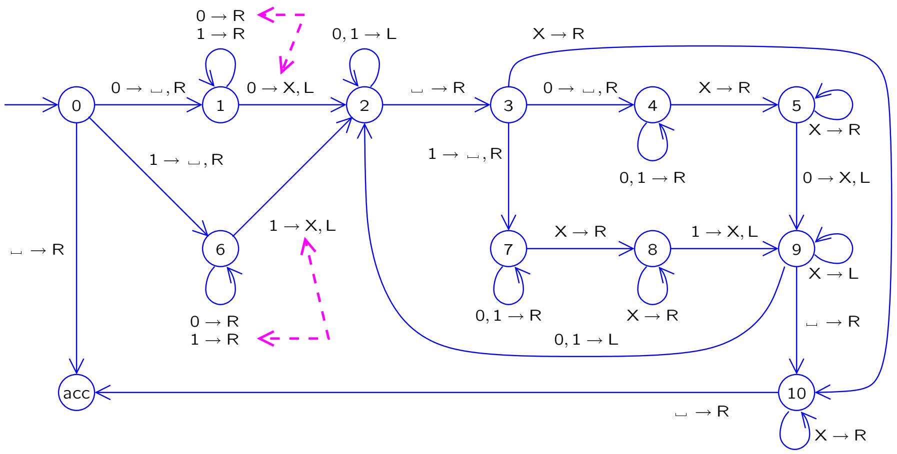
Jos määritelmä olisi löysästi "jonkin laskennan pitää johdaa oikeaan lopputulokseen", niin mikä tahansa kieli voitaisiin tunnistaa triviaalisti asettamalla kaikilla $a \in \Sigma$

$$\delta(q_0, a) = \{\, (q_{\mathsf{accept}}, a, \mathsf{R}), (q_{\mathsf{reject}}, a, \mathsf{R}) \,\}.$$

Esimerkki 3.5: Tuttu kieli $B = \{\, ww \mid w \in \{\, 0, 1 \,\}^* \,\}$ voidaan tunnistaa epädeterministisellä Turingin koneella seuraavasti:

1. Laita syötteen ensimmäinen symboli muistiin ja pyyhi se pois.
2. *Valitse epädeterministisesti* jokin symboli syötteestä toisen puoliskon alkumerkiksi. Jos se ei ollut sama kuin ensimmäinen symboli, niin *hylkää*. Muuten viivaa se yli.
3. Palaa syötteen alkuun.
4. Jos alussa ei ole enää yliviivaamattomia merkkejä, mene kohtaan 6. Muuten laita syötteen seuraava merkki muistiin ja pyyhi se pois.
5. Selaa nauhaa oikealle ohi kaikkien yliviivattujen symbolien. Jos niitä seuraava symboli ei ole muistissa oleva, *hylkää*. Muuten viivaa yli tämä symboli ja palaa kohtaan 3.
6. Jos kaikki symbolit on yliviivattu, *hyväksy*, muuten *hylkää*.

Erona deterministiseen ratkaisuun (ss. 237–238) meidän ei tarvitse etsiä syötteen keskikohtaa. Riittää arvata **ja tarkastaa**.



Sama kaaviona. Hylkäävä tila siirtymineen jätetty merkitsemättä. Kaksi epädeterminististä siirtymäparia merkitty nuolilla. □

Lause 3.6: [Sipser Thm. 3.16] Jokaiselle epädeterministiselle Turingin koneelle on olemassa ekvivalentti deterministinen Turingin kone.

Todistus: Olkoon N epädeterministinen Turingin kone. Muodostamme deterministisen Turingin koneen D, joka kokeilee järjestyksessä kaikkia mahdollisia koneen N laskentoja annetulla syötteellä. Jos hyväksyvä laskenta löytyy, D hyväksyy. Muuten D jää silmukkaan. Siis $L(D) = L(N)$.

Perusidea on kuvitella puu, jonka solmuina on koneen N tilanteita:

- juurena alkutilanne annetulla syötteellä ja
- solmun uqv lapsina kaikki tilanteet $u'q'v'$, joilla $uqv \vdash u'q'v'$.

Kone D käy puuta läpi leveyssuuntaisesti. Tätä varten haluamme indeksoida puun solmut.

Olkoon b yläraja solmun lapsien lukumäärälle:

$$b = \max_{q,a} |\delta(q, a)|.$$

Vakion b tarkalla arvolla ei ole tässä merkitystä, mutta selvästi $b \leq 3|Q||\Gamma|$.

Numeroidaan nyt jokaisen solmun lapset (eli jokaisen tilanteen seuraajatilanteet) numeroilla joukosta $\{1, \ldots, b\}$ (joista osa voi jäädä käyttämättä). Jos uqv ja $u'q'v'$ ovat kaksi saman solmun lasta, voidaan esim. sopia, että uqv saa pienemmän numeron, jos

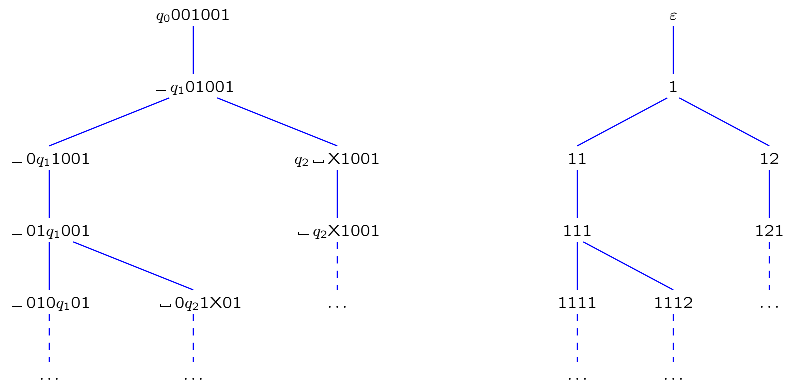
- jos q:n järjestysnumero on pienempi kuin q':n tai
- $q = q'$ ja u on aakkosjärjestyksessä ennen kuin u' tai
- $q = q'$ ja $u = u'$ ja v on aakkosjärjestyksessä ennen kuin v'.

Tästä saadaan puun jokaiselle solmulle osoite joukosta $\{1, \ldots, b\}^*$:

- juuren osoite on ε ja
- jos solmun osoite on $p_1 \ldots p_k$, niin sen lapsi numero q saa osoitteen $p_1 \ldots p_k q$.

Kaikki joukon $\{1, \ldots, b\}^*$ alkiot eivät ole minkään solmun osoitteita.

Esimerkki 3.7: Tarkastellaan sivun 249 epädeterministisen Turingin koneen laskentaa syötteellä 001001. Laskentapuun alkuosa ja vastaava solmujen numerointi ovat seuraavat:



Puussa ei ole esim. solmua numero 122, koska solmun 12 tilanteella ␣$Xq_2$1001 on vain yksi mahdollinen seuraaja. □

Muodostamme nyt kolminauhaisen koneen D, joka toimii kuten edellä on esitetty. Lauseen 3.3 nojalla tästä saadaan edelleen yksinauhainen kone.

Nauha 1 sisältää syötteen. Nauhalle 1 ei koskaan kirjoiteta.

Nauha 2 on työnauha, jonka sisältö seuraa koneen N nauhan sisältöä laskennan eri vaiheissa.

Nauha 3 sisältää aakkoston $\{1,\ldots,b\}$ merkkijonon, joka tulkitaan osoitteeksi koneen N laskentapuuhun.

Koneen D periaate on seuraava:

- Muodosta nauhalle 3 aakkoston $\{1,\ldots,b\}$ merkkijonoja pituusjärjestyksessä. Kullakin pituudella muodostetaan kaikki jonot esimerkiksi aakkosjärjestyksessä.
- Kullakin muodostetulla merkkijonolla etsi laskentapuusta vastaava solmu. Jos se vastaa hyväksyvää tilannetta, niin *hyväksy*. Jos solmua ei ole tai se vastaa muuta kuin hyväksyvää tilannetta, siirry seuraavaan merkkijonoon.

Nauhan 3 osoittaman solmun etsiminen tapahtuu seuraavasti:

1. Olkoon nauhan 3 merkkijono $p_1 \ldots p_n$.
2. Kopio nauhan 1 sisältö nauhan 2 sisällöksi ja aseta nauhapää nauhan 2 alkuun. Aseta koneen N simulaatio alkutilaan.
3. Simuloi koneen N laskentaa n askelta käyttäen nauhaa 2 koneen N työnauhana.
   - Laskenta-askelella numero i valitse uudeksi tilanteeksi nykyisen tilanteen seuraaja numero p_i.
   - Jos nykyisellä tilanteella on vähemmän kuin p_i seuraajaa, keskeytä laskenta; etsittyä solmua ei ole.

□

Korollaari 3.8: [Sipser Cor. 3.18] Kieli on Turing-tunnistettava, jos ja vain jos jokin epädeterministinen Turingin kone tunnistaa sen.

Todistus: Suunta "vain jos" on ilmeinen. Suunta "jos" seuraa edellisestä lauseesta. □

Epädeterministinen Turingin kone on *ratkaisija* (eli *totaalinen* kone), jos jokaisella syötteellä jokainen laskenta pysähtyy.

Korollaari 3.9: [Sipser Cor. 3.19] Kieli on ratkeava, jos ja vain jos jokin epädeterministinen Turingin kone ratkaisee sen.

Todistus: Edellisen lauseen konstruktiota on helppo täydentää siten, että D havaitsee, jos kaikki koneen N laskennat ovat pysähtyneet. □

Algoritmin määritelmä [Sipser luku 3.3]

Mitä algoritmilla yleensä tarkoitetaan

periaatteessa: yksiselitteisesti kuvattu jono (tietojenkäsittely)operaatioita, jotka voidaan toteuttaa mekaanisesti

käytännössä: luonnollista kieltä, pseudokoodia yms. käyttävä esitys, jonka pätevä ohjelmoija osaa koodata ilman suurempia ongelmia.

Tämä tarkkuustaso ei ole riittävä, jos halutaan tutkia laskettavuuden rajoja.

Erityisesti kun haluamme väittää jostain ongelmasta, että sille *ei ole olemassa* ratkaisualgoritmia, niin mitä oikeastaan väitämme? Huomaa, että tämä on oleellisesti eri asia kuin todeta, että ongelmalle ei ole *keksitty* algoritmia (toistaiseksi).