Luku 5

Turingin koneet ja rajoittamattomat kielet

Tässä luvussa tutustumme kaikkein tärkeimpään konetyyppiin, Turingin koneisiin, joilla voidaan *Churchin-Turingin teesin* mukaan ratkaista mikä tahansa mekaanisesti ratkeava ongelma. Toisin sanoen Turingin koneet tunnistavat Chomskyn kielihierarkian vahvimman kielityypin, *rajoittamattomat kielet* (tyyppi 0). Turingin koneiden avulla voidaan siis tutkia, onko ongelma ylipäänsä ratkeava (keksitäänkö sen ratkaiseva Turingin kone) sekä analysoida ongelman vaativuutta (kuinka monta aika-askelta tai työnauhan solua kone tarvitsee työssään).

Rajoittamattomien kielten luokka voidaan jakaa kahteen luokkaan sen perusteella, pysähtyykö ongelman ratkaiseva Turingin kone kaikilla syötteillä (siis sekä hyväksyessään

että hylätessään merkkijonon se lopulta pysähtyy ja palauttaa vastauksen "kyllä" tai "ei") vai pysähtyykö se vain hyväksyvässä tapauksessa (vastaus "kyllä"). Ensimmäistä, aina pysähtyvien eli *totaalisten Turningin koneiden* tunnistamaa kieliluokkaa kustutaan *Rekursiivisiksi kieliksi* ja sitä vastaavat päätösongelmat ovat *ratkeavia* (*solvable, decidable*). Toista, vain myönteisissä tapauksissa pysähtyviä koneita vastaavaa kieliluokkaa kutsutaan *Rekursiivisesti numeroituviksi* kieliksi ja vastaavat päätösongelmat ovat *osittain ratkeavia* (*semidecidable*). Kieliluokkien ulkopuolelle jäävät täysin ratkeamattomat ongelmat. Huom! Myös osoittain ratkeavat ongelmat luetaan ratkeamattomiin ongelmiin (*unsolvable, undecidable*). [3]

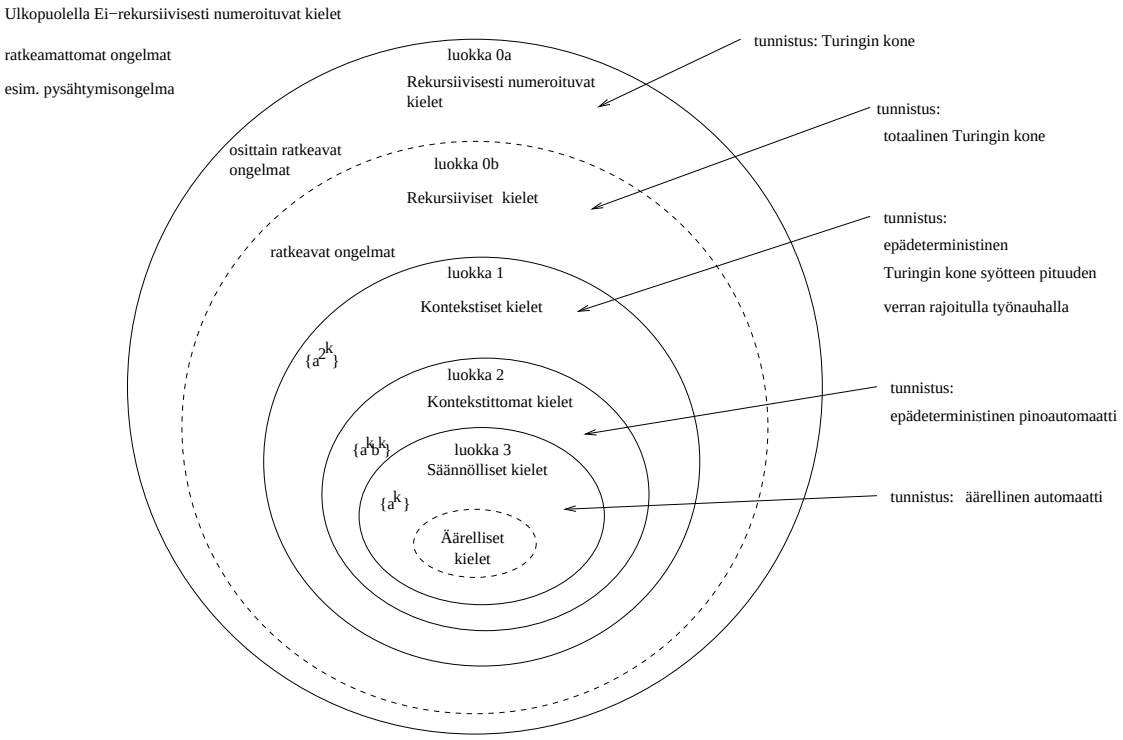
On syytä huomata, että olemme hypänneet kokonaan yhden Chomskyn luokan, nimittäin *kontekstillisten kielten* (luokka 1) yli. Tietojenkäsittelytieteen kannalta kontekstillisten kielten luokkaa ei ole pidetty merkittävänä, sillä kaikki kontekstilliset kielet voidaan tunnistaa Turingin koneiden avulla ja toisaalta törmäämme käytännössä harvoin ongelmiin, jotka olisivat (ts. vastaavat formaalit kielet olisivat) kontekstillisia, mutta eivät rajoittamattomia. Chomsky kuitenkin uskoi luonnollisten kielten kuuluvan juuri tähän luokkaan.

5.1 Churchin-Turingin teesi

Churchin-Turingin teesi: Mikä tahansa mekaanisesti ratkeava ongelma voidaan ratkaista Turingin koneella.

- kaikki algoritmisesti ratkeavat ongelmat
- Esimerkiksi ohjelmointikielet ja RAM-koneet laskentavoimaltaan ekvivalentteja Turingin koneiden kanssa
  - Kaikki "riittävän vahvat" ohjelmointikielet määrittävät täsmälleen saman ratkeavien ongelmien luokan
  - Perustelu: millä tahansa riittävän vahvalla ohjelmointikielellä voidaan kirjoittaa kääntäjä (oik. tulkkiohjelma) mille tahansa toiselle ohjelmointikielelle
  - Mikä tahansa Turingin kone voidaan toteuttaa tällaisen ohjelmointikielen ohjelmana ja kääntäen (sama pätee esim. RAM-koneelle) (ks. esim. Hopcroft-Motwani-Ullman, luvut 8.6.1 ja 8.6.2)

[3] Huom! Nimityksillä *rekursiivinen* ja *rekursiivisesti numeroituva* ei ole paljonkaan tekemistä tietojenkäsittelytieteilijän ymmärtämän rekursion kanssa. Termien hämäävä alkuperä johtuu matemaatikkojen (Gödel, Kleene) määrittelemistä rekursiivisista ja rekursiivisesti numeroituvista funktioista, joista kerrotaan lisää ekskursiossa **??**.



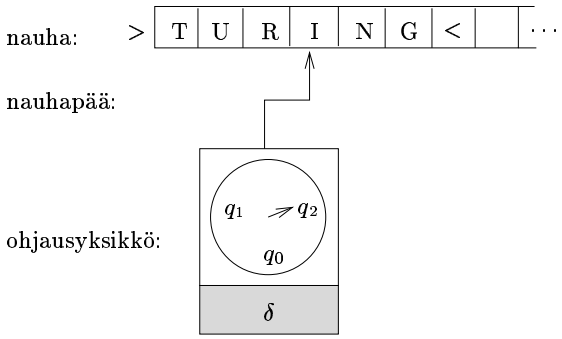
Kuva 5.1: Chomskyn kielihierarkia täydennettynä rekursiivisten ja rekursiivisisesti nuemroituvien kielten luokilla.

- Huom! Myös kaksipinoiset pinoautomaatit ovat yhtä vahvoja kuin Turingin koneet! (ks. esim. Hopcroft-Motwani-Ullman: teoreema 8.13)

- Chomskyn kielihierarkiassa rekursiivisesti lueteltavat kielet: jos Turingin kone ratkaisee tunnistusongelman osittain, niin mikään muukaan malli ei kykene ratkaisemaan sitä kuin osittain (ts. joillain syötteillä laskenta jatkuu ikuisesti...).

- Yleensä algoritmiksi kutsutaan vain sellaista algoritmia, jonka laskenta päättyy aina, kaikilla syötteillä – ts. rekursiiviset eli ratkeavat kielet. Ei ole järkeä tehdä ohjelmaa, joka ei koskaan pysähdy! Matematiikassakin termi ”funktio” on yleensä varattu vain laskettavissa olevalle (rekursiiviselle) funktiolle, ja osittain laskettavia funktioita kutsutaan osittais(rekursiivisiksi) funktioiksi.

- Huom! Vain teesi, ei teoreema (lause): kukaan ei ole voinut todistaa oikeaksi, mutta kaikki tunnetut laskentamallit ovat osoittautuneet ekvivalenteiksi ja yleisesti uskotaan pätevän kaikille laskennan malleille.



Kuva 5.2: Turingin kone.

- Jotkut ovat spekuloineet, olisivatko ns. kvanttitietokoneet vahvempia...

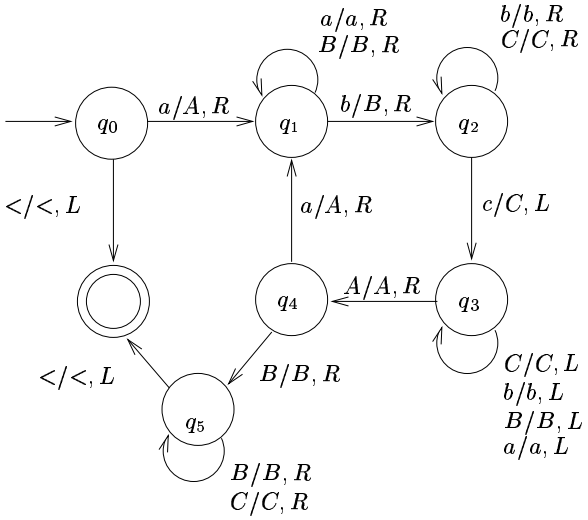
5.2 Turingin koneet

- kuten äärellinen automaatti, jolla on toiseen suuntaan ääretön työnauha, jota voi lukea ja kirjoittaa merkki kerrallaan
- aakkoston lisäksi nauhan alkua merkitsevä erikoismerkki > ja loppua merkitsevä erikoismerkki < (loppumerkki "liikkuu" eteenpäin, kun sen päälle kirjoitetaan aakkoston symboli ts. voit ajatella, että koko loppunauha on täynnä loppumerkkiä <<<<< ...
- Aluksi nauhalla on syötemerkkijono (ja loppu nauhasta tyhjä), nauhapää osoittaa ensimmäistä nauhapään paikkaa ja kone käynnistetään alkutilassa q_0
- Kullakin laskenta-askeleella se lukee nauhapään kohdalla olevan merkin, päättää siirtymäfunktion mukaisesti uuden tilansa, kirjoittaa nauhapään kohdalle uuden merkin ja siirtää nauhapäätä yhden askeleen vasemmalle tai oikealle (ensimmäisen paikan vasemmalle puolelle ei voi kuitenkaan mennä)
- Koneella on hyväksyvä lopputila q_{yes} ja hylkäävä lopputila q_{no} (kun kyseessä on kielen tunnistaminen, jota nyt käsittelemme). Kone pysähtyy, kun se saavuttaa lopputilan.
- Turingin kone eroaa äärellisestä automaatista siten että
  - työnauhalle voidaan kirjoittaa
  - työnauhalla voi liikkua sekä vasemmalle että oikealle

- työnauha on rajoittamattoman pitkä
- kun lopputila on saavutettu, kone pysähtyy

Esim. kielen $\{a^k b^k c^k | k \geq 0\}$ tunnistava Turingin kone

- Idea: kone pitää kirjaa tapaamistaan a:sta, b:stä ja c:stä muuttamalla ne yksi kerrallaan A:ksi, B:ksi ja C:ksi.
- muutettuaan viimeisen a:n A:ksi tarkistaa, ettei enää jäljellä b:tä tai c:tä.



Kuva 5.3: Kielen $\{a^k b^k c^k \mid k \geq 0\}$ tunnistava Turingin kone.

Huom! Sama voidaan ratkaista kaksipinoisella pinoautomaatilla (harjoitustehtävä) tai tietokoneohjelmana:

```
while ((c=getchar())!=EOF) {
    switch(c) {
        case 'a': A++;
        case 'b': B++;
        case 'c': C++;
        else: ERROR;
    }
}
if ((A==B) && (B==C))
    printf("Ok");
```

5.2.1 Formaali määrittely

Määritelmä: *Turingin kone* on seitsikko

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}}),$$

missä

- Q on koneen *tilojen* äärellinen joukko;
- Σ on koneen *syöteaakkosto*;
- $\Gamma \supseteq \Sigma$ on koneen *nauha-aakkosto*; ol. että $>, < \notin \Gamma$;
- $\delta : (Q - \{q_{\text{yes}}, q_{\text{no}}\}) \times (\Gamma \cup \{>, <\}) \to Q \times (\Gamma \cup \{>, <\}) \times \{L, R\}$ on koneen *siirtymäfunktio*;
- $q_0 \in Q$ on koneen *alkutila*;
- $q_{\text{yes}} \in Q$ on koneen *hyväksyvä* ja $q_{\text{no}} \in Q$ sen *hylkäävä lopputila*.

Siirtymäfunktion arvoilta

$$\delta(q, a) = (q', b, \Delta)$$

vaaditaan:

(i) jos $b = >$, niin $a = >$ (alkumerkkiä ei saa siirtää);

(ii) jos $a = >$, niin $b = >$ ja $\Delta = R$ (alkumerkin vasemmalle puolelle ei saa siirtyä);

(iii) jos $b = <$, niin $a = <$ ja $\Delta = L$ (loppumerkin saa kirjoittaa vain entisen loppumerkin päälle ja silloin täytyy siirtyä vasemmalle).

- Siirtymäfunktion arvon $\delta(q, a) = (q', b, \Delta)$ tulkinta:
  - Ollessaan tilassa q ja lukiessaan nauhamerkin (tai alku- tai loppumerkin) a, kone siirtyy tilaan q', kirjoittaa lukemaansa paikkaan merkin b, ja siirtää nauhapäätä yhden merkkipaikan verran suuntaan Δ ($L \sim$ "left", $R \sim$ "right").
  - Sallittuja kirjoitettavia merkkejä ja siirtosuuntia on rajoitettu, mikäli a = '$>$' tai '$<$' (yllä olevat kohdat (i)–(iii)), ja siirtymäfunktion arvo on aina määrittelemätön, kun $q = q_{\text{yes}}$ tai $q = q_{\text{no}}$. Joutuessaan jompaan kumpaan näistä tiloista kone pysähtyy heti.

- Koneen *tilanne* on nelikko

$$(q, u, a, v) \in Q \times \Gamma^* \times (\Gamma \cup \{\epsilon\}) \times \Gamma^*,$$

missä voi olla $a = \epsilon$, mikäli myös $u = \epsilon$ tai $v = \epsilon$.

- Tulkinta: kone on tilassa q, nauhan sisältö sen alusta nauhapään vasemmalle puolelle on u, nauhapään kohdalla on merkki a ja nauhan sisältö nauhapään oikealta puolelta käytetyn osan loppuun on v.

- Mahdollisesti on $a = \epsilon$, jos nauhapää sijaitsee aivan nauhan alussa ($u = \epsilon$) tai sen käytetyn osan lopussa ($v = \epsilon$). Ensimmäisessä tapauksessa ajatellaan, että kone "havaitsee" merkin '$>$' ja toisessa tapauksessa merkin '$<$'.

- *Alkutilanne syötteellä* $x = a_1 a_2 \ldots a_n$ on nelikko

$$(q_0, \epsilon, a_1, a_2 \ldots a_n).$$

- Tilannetta (q, u, a, v) merkitään yleensä yksinkertaisemmin $(q, u\underline{a}v)$, ja alkutilannetta syötteellä x yksinkertaisesti $(q_0, \underline{x})$.

- Tilanne (q, w) *johtaa suoraan* tilanteeseen (q', w'), merkitään

$$(q, w) \underset{M}{\vdash} (q', w'),$$

jos jokin seuraavista ehdoista täyttyy: kaikilla $q, q' \in Q$, $u, v \in \Gamma^*$, $a, b \in \Gamma$ ja $c \in \Gamma \cup \{\epsilon\}$:

jos $\delta(q, a) = (q', b, R)$, niin $(q, u\underline{a}cv) \underset{M}{\vdash} (q', ub\underline{c}v)$;

jos $\delta(q, a) = (q', b, L)$, niin $(q, uc\underline{a}v) \underset{M}{\vdash} (q', u\underline{c}bv)$;

jos $\delta(q, >) = (q', >, R)$, niin $(q, \underline{\epsilon}cv) \underset{M}{\vdash} (q', \underline{c}v)$;

jos $\delta(q, <) = (q', b, R)$, niin $(q, u\underline{\epsilon}) \underset{M}{\vdash} (q', ub\underline{\epsilon})$;

jos $\delta(q, <) = (q', b, L)$, niin $(q, uc\underline{\epsilon}) \underset{M}{\vdash} (q', u\underline{c}b)$;

jos $\delta(q, <) = (q', <, L)$, niin $(q, uc\underline{\epsilon}) \underset{M}{\vdash} (q', u\underline{c})$.

- Tilanteet, jotka ovat muotoa (q_{yes}, w) tai (q_{no}, w) eivät johda mihinkään muuhun tilanteeseen. Näissä tilanteissa kone *pysähtyy*.

- Tilanne (q, w) *johtaa tilanteeseen* (q', w'), merkitään

$$(q, w) \underset{M}{\vdash^*} (q', w'),$$

jos on olemassa tilannejono (q_0, w_0), (q_1, w_1), $\ldots$, (q_n, w_n), $n \geq 0$, siten että

$$(q, w) = (q_0, w_0) \underset{M}{\vdash} (q_1, w_1) \underset{M}{\vdash} \cdots \underset{M}{\vdash} (q_n, w_n) = (q', w').$$

- Turingin kone M *hyväksyy* merkkijonon $x \in \Sigma^*$, jos

$$(q_0, \underline{x}) \underset{M}{\vdash^*} (q_{\text{yes}}, w) \qquad \text{jollakin } w \in \Gamma^*;$$

muuten M *hylkää* x:n.

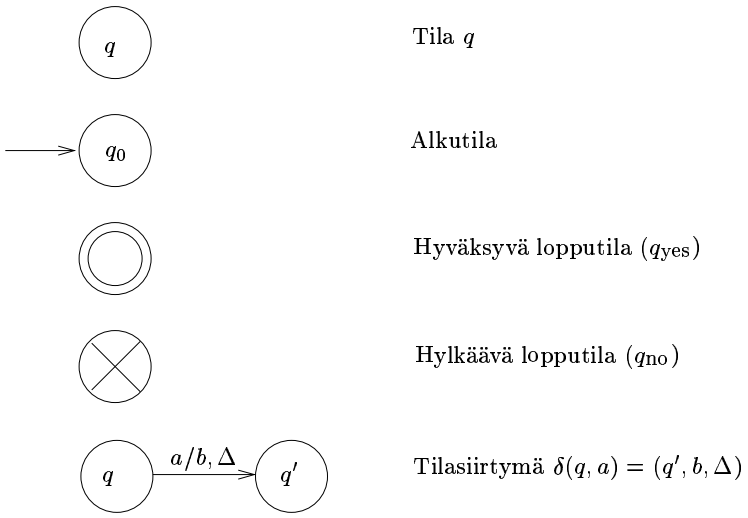
(ts. hyväksymiseen riittää, että pääsemme hyväksyvään lopputilaan, koko merkkijonoa ei välttämättä tarvitse edes lukea! Nauhalle saa luonnollisestikin jäädä merkkejä $\leftrightarrow$ automaatit)

- Koneen M *tunnistama kieli* koostuu syöteaakkoston merkkijonoista, joilla päästään alkutilasta hyväksyvään lopputilaan:

$$L(M) = \{x \in \Sigma^* \mid (q_0, \underline{x}) \underset{M}{\vdash^*} (q_{\text{yes}}, w) \text{ jollakin } w \in \Gamma^*\}.$$

5.2.2 Turingin koneen esitys siirtymäkaaviona

Turingin kone voidaa esittää antamalla sen siirtymäfunktio tai siirtymäkaaviona samaan tapaan kuin äärelliset automaatit.



Kuva 5.4: Turingin koneiden kaavioesityksen merkinnät.

Esimerkki: kieli $\{a^{2k} \mid k \geq 0\}$ voidaan tunnistaa Turingin koneella

$$M = (\{q_0, q_1, q_{\text{yes}}, q_{\text{no}}\}, \{a\}, \{a\}, \delta, q_0, q_{\text{yes}}, q_{\text{no}}),$$

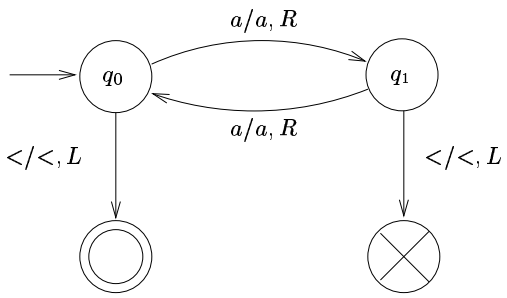
missä

$$\begin{aligned} \delta(q_0, a) &= (q_1, a, R), \\ \delta(q_1, a) &= (q_0, a, R), \\ \delta(q_0, <) &= (q_{\text{yes}}, <, L), \\ \delta(q_1, <) &= (q_{\text{no}}, <, L). \end{aligned}$$

Koneen M laskenta esimerkiksi syötteellä aaa etenee seuraavasti:

$$\begin{aligned} (q_0, \underline{a}aa) &\underset{M}{\vdash} (q_1, a\underline{a}a) \underset{M}{\vdash} (q_0, aa\underline{a}) \\ &\underset{M}{\vdash} (q_1, aaa\underline{\epsilon}) \underset{M}{\vdash} (q_{\text{no}}, aa\underline{a}). \end{aligned}$$

Kone pysähtyy tilassa q_{no}, joten $aaa \notin L(M)$.



Kuva 5.5: Kielen $\{a^{2k} \mid k \geq 0\}$ tunnistava Turingin kone.

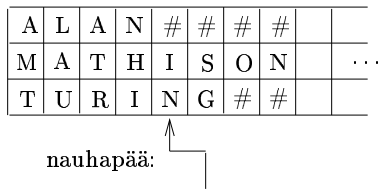
Esimerkki 2: Kielen $\{a^k b^k c^k \mid k \geq 0\}$ tunnistavan koneen laskenta syötteellä *aabbcc*:

$$
\begin{array}{lll}
(q_0, \underline{a}abbcc) & \vdash & (q_2, AABB\underline{C}c) \quad \vdash \\
(q_1, A\underline{a}bbcc) & \vdash & (q_2, AABBC\underline{c}) \quad \vdash \\
(q_1, Aa\underline{b}bcc) & \vdash & (q_3, AABB\underline{C}C) \quad \vdash \\
(q_2, AaB\underline{b}cc) & \vdash & (q_3, AAB\underline{B}CC) \quad \vdash \\
(q_2, AaBb\underline{c}c) & \vdash & (q_3, AA\underline{B}BCC) \quad \vdash \\
(q_3, AaB\underline{b}Cc) & \vdash & (q_3, A\underline{A}BBCC) \quad \vdash \\
(q_3, Aa\underline{B}bCc) & \vdash & (q_4, AA\underline{B}BCC) \quad \vdash \\
(q_3, A\underline{a}BbCc) & \vdash & (q_5, AAB\underline{B}CC) \quad \vdash \\
(q_3, \underline{A}aBbCc) & \vdash & (q_5, AABB\underline{C}C) \quad \vdash \\
(q_4, A\underline{a}BbCc) & \vdash & (q_5, AABBC\underline{C}) \quad \vdash \\
(q_1, AA\underline{B}bCc) & \vdash & (q_5, AABBCC\underline{\epsilon}) \quad \vdash \\
(q_1, AAB\underline{b}Cc) & \vdash & (q_{\text{yes}}, AABBC\underline{C}).
\end{array}
$$

5.3 Turingin koneiden laajennuksia

Turingin koneista on olemassa monia erilaisia variaatioita. Koneen työnauha voi esimerkiksi olla kumpaankin suuntaan ääretön tai sallitaan, että koneen nauhapää saa pysyä paikallaan. Seuraavassa esitellään kolme hyödyllistä variaatiota: moniuraiset koneet, joiden työnauha koostuu useasta rinnakkaisesta urasta, moninauhaiset koneet, joilla on useita toisistaan riippumattomia työnauhoja, sekä tärkeimpänä kaikista, epädeterministiset Turingin koneet. **Huom!** Mikä tahansa näistä variaatioista voidaan esittää standardimallisena Turingin koneena.

5.3.1 *Moniuraiset koneet



Kuva 5.6: Kolmeuraisen Turingin koneen nauha.

- Sallitaan, että Turingin koneen nauha koostuu k:sta rinnakkaisesta urasta, jotka kaikki kone lukee ja kirjoittaa yhdessä laskenta-askelessa.
- Koneen siirtymäfunktion arvot ovat tällöin muotoa:

$$\delta(q, (a_1, \ldots, a_k)) = (q', (b_1, \ldots, b_k), \Delta),$$

  missä $a_1, \ldots, a_k$ ovat urilta $1, \ldots, k$ luetut merkit, $b_1, \ldots, b_k$ niiden tilalle kirjoitettavat merkit, ja $\Delta \in \{L, R\}$ on nauhapään siirtosuunta.
- Laskennan aluksi tutkittava syöte sijoitetaan ykkösuran vasempaan laitaan; muille urille tulee sen kohdalle erityisiä tyhjämerkkejä $\#$.
- Merkitys: moninauhainen kone voidaan muuntaa yksinauhaiseksi muuntamalla se ensin vastaavaksi moniurakoneeksi ja sitten yksiuraiseksi standardikoneeksi.
- Formaalisti *k-urainen Turingin kone* on seitsikko

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}}),$$

missä muut komponentit ovat kuten standardimallissa, paitsi siirtymäfunktio:

$$\delta : (Q - \{q_{\text{yes}}, q_{\text{no}}\}) \times (\Gamma^k \cup \{>, <\}) \to \\ Q \times (\Gamma^k \cup \{>, <\}) \times \{L, R\}.$$

(Seuraajatilannerelaation $\underset{M}{\vdash}$, alkutilan jne. määritelmät ovat pieniä muutoksia lukuunottamatta samanlaiset kuin standardimallissa.)

Lause: Jos formaali kieli L voidaan tunnistaa k-uraisella Turingin koneella, se voidaan tunnistaa myös standardimallisella Turingin koneella.

Todistus.

- Olkoon $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}})$ k-urainen Turingin kone, joka tunnistaa kielen L.

- Vastaava standardimallinen kone $\widehat{M}$ muodostetaan seuraavasti:

$$\widehat{M} = (\hat{Q}, \Sigma, \hat{\Gamma}, \hat{\delta}, \hat{q}_0, q_{\text{yes}}, q_{\text{no}}),$$

missä $\hat{Q} = Q \cup \{\hat{q}_0, \hat{q}_1, \hat{q}_2\}$, $\hat{\Gamma} = \Sigma \cup \Gamma^k$ ja kaikilla $q \in Q$ on

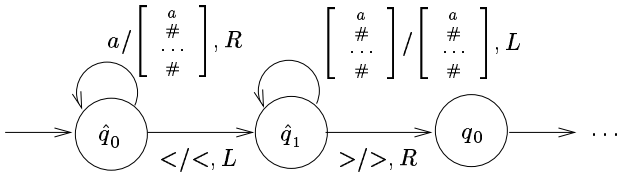
$$\hat{\delta}(q, \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix}) = (q', \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix}, \Delta), \\ \text{kun} \quad \delta(q, (a_1, \ldots, a_k)) = (q', (b_1, \ldots, b_k), \Delta).$$

- Koneen $\widehat{M}$ laskennan aluksi täytyy syötejono "nostaa" ykkösuralle, so. korvata nauhalla merkkijono $a_1 a_2 \ldots a_n$ merkkijonolla

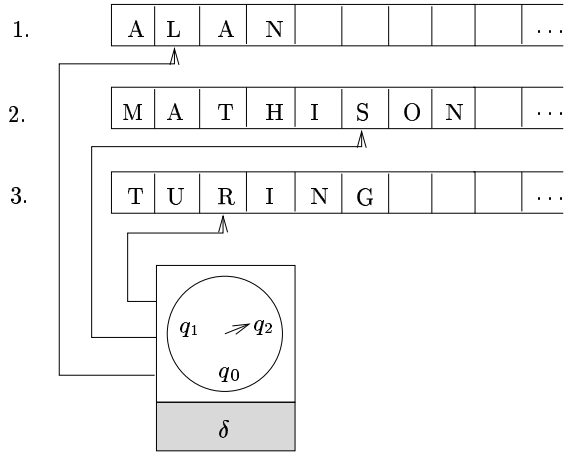
$$\begin{bmatrix} a_1 \\ \# \\ \vdots \\ \# \end{bmatrix} \begin{bmatrix} a_2 \\ \# \\ \vdots \\ \# \end{bmatrix} \ldots \begin{bmatrix} a_n \\ \# \\ \vdots \\ \# \end{bmatrix}.$$

- Tätä operaatiota varten liitetään M:stä kopioidun siirtymäfunktion osan alkuun vielä pieni "esiprosessori".

□



Kuva 5.7: Esiprosessori moniuraisen koneen sytteen nostamiseksi ykksuralle.



Kuva 5.8: Kolmenauhainen Turingin kone.

5.3.2 Moninauhaiset koneet

- Sallitaan, että Turingin koneella on k toisistaan riippumatonta nauhaa, joilla on kullakin oma nauhapäänsä.
- Kone lukee ja kirjoittaa kaikki nauhat yhdessä laskenta-askelessa.
- Laskennan aluksi syöte sijoitetaan ykkösnauhan vasempaan laitaan ja kaikki nauhapäät nauhojensa alkuun.
- Tällaisen koneen siirtymäfunktion arvot ovat muotoa

$$\delta(q, a_1, \ldots, a_k) = (q', (b_1, \Delta_1), \ldots, (b_k, \Delta_k)),$$

  missä $a_1, \ldots, a_k$ ovat nauhoilta $1, \ldots, k$ luetut merkit, $b_1, \ldots, b_k$ niiden tilalle kirjoitettavat merkit, ja $\Delta_1, \ldots, \Delta_k \in \{L, R\}$ nauhapäiden siirtosuunnat.
- Merkitys: usein on helpompi ratkaista ongelma moninauhaisella koneella – esim. yhteenlaskussa $a + b = c$ kirjoitetaan 1. nauhalle luvun a binääriesitys x, 2. nauhalle luvun b binääriesitys y ja lasketaan 3. nauhalle vastauksen c binääriesitys z. Ja mikäli ongelma kyetään ratkaisemaan monen nauhan avulla, se kyetään ratkaisemaan myös yksinauhaisella standardikoneella.

- Formaalisti *k-nauhainen Turingin kone* on seitsikko

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}}),$$

missä muut komponentit ovat kuten standardimallissa, paitsi siirtymäfunktio:

$$\delta : (Q - \{q_{\text{yes}}, q_{\text{no}}\}) \times (\Gamma \cup \{>, <\})^k \rightarrow \\ Q \times ((\Gamma \cup \{>, <\}) \times \{L, R\})^k.$$

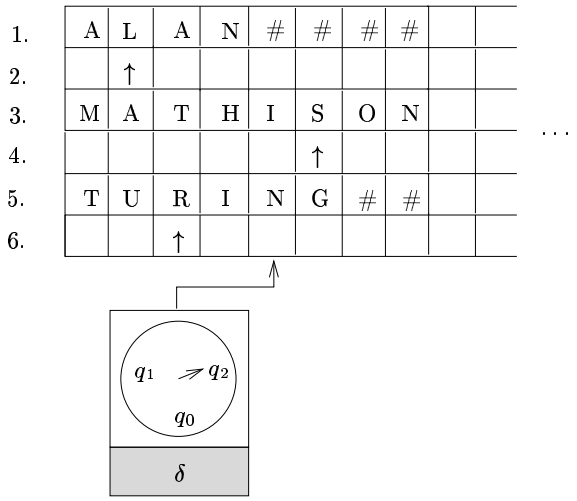
(Seuraajatilannerelaatio ym. peruskäsitteet määritellään pienin muutoksin entiseen tapaan.)

Lause: Jos formaali kieli L voidaan tunnistaa k-nauhaisella Turingin koneella, se voidaan tunnistaa myös standardimallisella Turingin koneella.

Todistus (idea).

- Olkoon $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}})$ k-nauhainen Turingin kone, joka tunnistaa kielen L.
- Konetta M voidaan simuloida $2k$-uraisella koneella $\widehat{M}$ siten, että koneen $\widehat{M}$ parittomat urat $1, 3, 5, \ldots, 2k-1$ vastaavat M:n nauhoja $1, 2, \ldots, k$, ja kutakin paritonta uraa seuraavalla parillisella uralla on merkillä $\uparrow$ merkitty vastaavan nauhan nauhapään sijainti.
- Simuloinnin aluksi syötemerkkijono sijoitetaan normaalisti koneen $\widehat{M}$ ykkösuralle. Ensimmäisessä siirtymässään $\widehat{M}$ merkitsee nauhapääosoittimet $\uparrow$ parillisten urien ensimmäisiin merkkipaikkoihin.
- Tämän jälkeen $\widehat{M}$ toimii "pyyhkimällä" nauhaa edestakaisin sen alku- ja loppumerkin välillä.
  - Vasemmalta oikealle pyyhkäisyllä $\widehat{M}$ kerää tiedot kunkin osoittimen kohdalla olevasta M:n nauhamerkistä.
  - Kun kaikki merkit ovat selvillä, $\widehat{M}$ simuloi yhden M:n siirtymän.
  - Takaisin oikealta vasemmalle suuntautuvalla pyyhkäisyllä kone kirjoittaa $\uparrow$-osoittimien kohdalle asianmukaiset uudet merkit ja siirtää osoittimia.

□



Kuva 5.9: Kolmenauhaisen Turingin koneen simulointi kuusiuraisella.

5.3.3 Epädeterministiset koneet

- tavalliset Turingin koneet ovat *deterministisiä* ts. koneen toiminta on yksikäsitteisesti määrättyä: kun tunnetaan nauhapään kohdalla oleva merkki ja koneen senhetkinen tila, määrää siirtymäfunktio yksikäsitteisesti nauhapään kohdalle kirjoitettavan merkin ja seuraavan tilan.

- Epädeterministisen Turingin koneen toiminta on *epädeterminististä* ts. se ei ole yksikäsitteisesti määriteltyä: kun tunnetaan nauhapään kohdalla oleva merkki ja senhetkinen tila, voi koneella olla useita vaihtoehtoja, mitä kirjoittaa nauhapään kohdalle ja mihin tilaan siirtyä seuraavaksi.

- Siirtymäfunktio on siis muotoa

$$\delta(q, a) = \{(q_1, b_1, \Delta_1), \ldots, (q_k, b_k, \Delta_k)\}$$

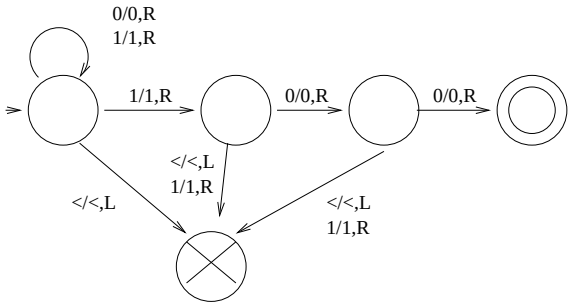
  Ts. ollessaan tilassa q ja lukiessaan merkin a kone voi toimia jonkin kolmikon (q_i, b_i, Δ_i) mukaisesti.

- Syötemerkkijon hyväksymiseksi riittää siis, että jokin koneen laskentapolku johtaa hyväksyvään lopputilaan.

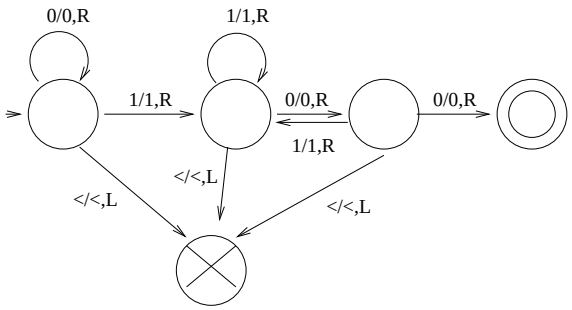
- Merkitys: usein helpompi keksiä epädeterministinen kone, joka ”arvaa” kulloinkin parhaan toiminta-askeleen. Mikäli halutaan vain tutkia ongelman vaikeutta, tämä riittää, sillä **mikä tahansa epädeterministinen kone voidaan esittää deterministisenä!** (aina voidaan simuloida deterministisellä koneella kaikkia mahdollisia laskentapolkuja.)

- **Huom!** Epädeterministiset koneet ovat tärkeitä myös määritettäessä ongelman epädeterministisiä vaativuusluokkia ja erityisesti ongelman *NP-täydellisyyttä*.

Esimerkki 1: Epädeterministinen kone, joka tutkii, sisältääkö annettu merkkijono osajonon 100:



Saman ongelman ratkaiseva deterministinen kone:



Määritelmä: Epädeterministinen Turingin kone on seitsikko

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}}),$$

missä muut komponentit ovat kuten deterministisessä standardimallissa, paitsi siirtymäfunktio:

$$\delta : (Q - \{q_{\text{yes}}, q_{\text{no}}\}) \times (\Gamma \cup \{>, <\}) \rightarrow \\ \mathcal{P}(Q \times (\Gamma \cup \{>, <\}) \times \{L, R\}).$$

- Siirtymäfunktion arvon

$$\delta(q, a) = \{(q_1, b_1, \Delta_1), \ldots, (q_k, b_k, \Delta_k)\}$$

tulkinta on, että ollessaan tilassa q ja lukiessaan merkin a kone voi toimia jonkin kolmikon (q_i, b_i, Δ_i) mukaisesti.

- Epädeterministisen koneen tilanteet, tilannejohdot jne. määritellään formaalisti samoin kuin deterministisenkin koneen tapauksessa, paitsi että ehdon $\delta(q, a) = (q', b, \Delta)$ sijaan kirjoitetaan $(q', b, \Delta) \in \delta(q, a)$ (siirtymäfunktio on siis joukkoarvoinen).

- Tämän muutoksen takia seuraajatilannerelaatio $\underset{M}{\vdash}$ ei ole enää yksiarvoinen: koneen tilanteella (q, w) voi nyt olla useita vaihtoehtoisia seuraajia, so. tilanteita (q', w'), joilla $(q, w) \underset{M}{\vdash} (q', w')$.

- Koneen M tunnistama kieli määritellään:

$$L(M) = \{x \in \Sigma^* \mid (q_0, \underline{x}) \underset{M}{\vdash}^* (q_{\text{yes}}, w) \text{ jollakin } w \in \Gamma^*\}.$$

- Epädeterministisen koneen M tapauksessa siis merkkijono x kuuluu M:n tunnistamaan kieleen, jos *jokin* M:n kelvollinen tilannejono johtaa alkutilanteesta syötteellä x hyväksyvään lopputilanteeseen.

Esimerkki 2: yhdistettyjen lukujen "tunnistaminen" epädeterministisillä Turingin koneilla.

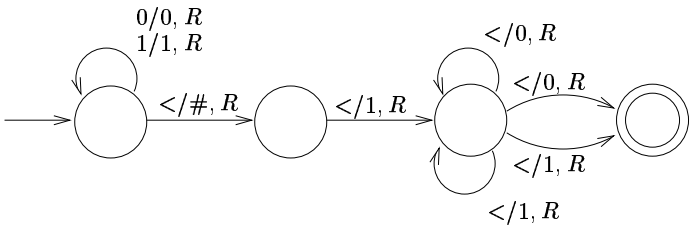
Ei-negatiivinen kokonaisluku n on *yhdistetty*, jos sillä on kokonaislukutekijät $p, q \geq 2$, joilla $pq = n$. Luku, joka ei ole yhdistetty, on *alkuluku*.

Oletetaan, että on jo suunniteltu deterministinen kone CHECK_MULT, joka tunnistaa kielen

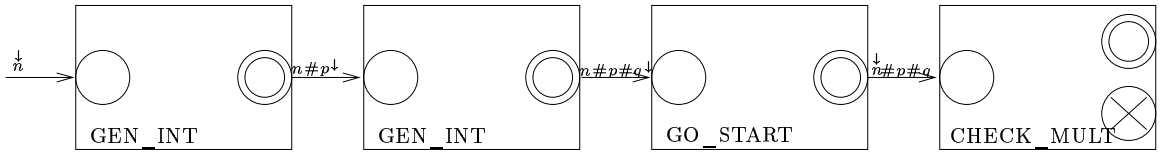
$$L(\text{CHECK\_MULT}) = \{n\#p\#q \mid n, p, q \text{ binäärilukuja, } n = pq\}.$$

Olkoon lisäksi GO_START deterministinen Turingin kone, joka siirtää nauhapään osoittamaan nauhan ensimmäistä merkkiä.

Olkoon edelleen GEN_INT mielivaltaisen ykköstä suuremman binääriluvun nauhan loppuun tuottava epädeterministinen Turingin kone.



Kuva 5.10: Epädeterministinen Turingin kone GEN_INT.



Kuva 5.11: Epädeterministinen Turingin kone TEST_COMPOSITE.

Epädeterministinen Turingin kone TEST_COMPOSITE, joka tunnistaa kielen

$$L(\text{TEST\_COMPOSITE}) = \{n \mid n \text{ on binäärimuotoinen yhdistetty luku}\}$$

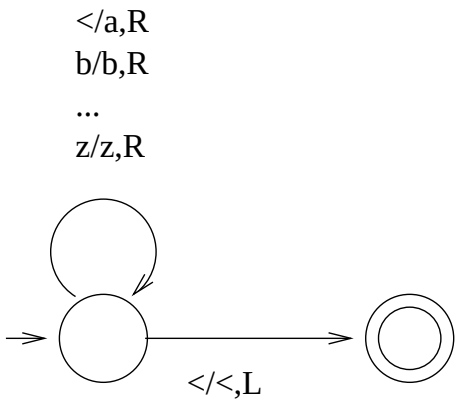
voidaan muodostaa näistä komponenteista yhdistämällä.

Yhdistetty kone hyväksyy syötteenä annetun binääriluvun n, jos ja vain jos on olemassa binääriluvut $p, q \geq 2$, joilla $n = pq$ — siis jos ja vain jos n on yhdistetty luku.

Esimerkki 3: Suunnitellaan epädeterministinen Turingin kone, joka tutkii, sisältäkö suunnattu verkko Hamiltonin kehän, ts. polun, joka kulkee verkon jokaisen solmun kautta kertaalleen ja palaa alkusolmuun.

Oletetaan, että solmut on nimetty kirjaimin $\Sigma = \{a, b, ..., z\}$. Koneen epädeterministinen komponentti $GUESS\_PATH$ arvaa solmujonon $a_1a_2...a_m$ ($a_i \in \Sigma$, $1 \leq i \leq m$), joka on Hamiltonin polku, mikäli sellainen on olemassa.

Sen jälkeen riittääkin tutkia deterministisesti, että polku todella löytyy verkosta, kulkee kaikkien solmujen kautta kertaalleen ja palaa lopuksi alkusolmuun. (Verkon vieruslistaesitys voidaan koodata esim. seuraavasti: n-solmuiselle verkolle muodostetaan n-nauhainen kone, jonka kukin nauha sisältää solmun a_i ja sen seuraajasolmut a_j ($a_i, a_j \in \Sigma$) Lisäksi tarvitaan työnauha arvattua polkua varten.)



Kuva 5.12: Epädeterministinen Turingin kone $GUESS\_PATH$.

Deterministisen ja epädeterministisen Turingin koneen ekvivalenssi

Lause: Jos formaali kieli L voidaan tunnistaa epädeterministisellä Turingin koneella, se voidaan tunnistaa myös standardimallisella deterministisellä Turingin koneella.

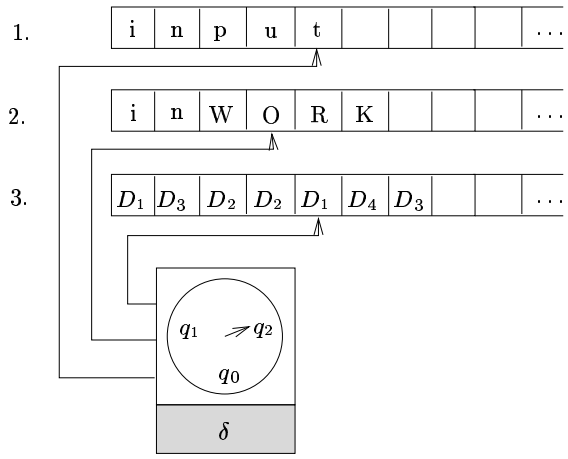
Todistus (idea).

- Olkoon $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}})$ epädeterministinen Turingin kone, joka tunnistaa kielen L.
- Konetta M voidaan simuloida kolmenauhaisella deterministisellä koneella $\widehat{M}$, joka käy systemaattisesti läpi M:n mahdollisia laskentoja (tilannejonoja), kunnes löytää hyväksyvän — jos sellainen on olemassa.
- Kone $\widehat{M}$ voidaan edelleen muuntaa standardimalliseksi edellisten lauseiden konstruktioilla.

Yksityiskohtaisemmin kuvattuna kone $\widehat{M}$ toimii seuraavasti:

- Nauhalla 1 $\widehat{M}$ säilyttää kopiota syötejonosta ja nauhalla 2 se simuloi koneen M työnauhaa. Kunkin simuloitavan laskennan aluksi $\widehat{M}$ kopioi syötteen nauhalta 1 nauhalle 2 ja pyyhkii pois nauhalle 2 edellisen laskennan jäljiltä mahdollisesti jääneet merkit.
- Nauhalla 3 $\widehat{M}$ pitää kirjaa vuorossa olevan laskennan "järjestysnumerosta". Tarkemmin sanoen, olkoon r suurin M:n siirtymäfunktion arvojoukon koko ($r = \max|\delta(q, a, \Delta)|$). Tällöin $\widehat{M}$:lla on erityiset nauhamerkit $D_1, \ldots, D_r$, joista koostuvia jonoja se generoi nauhalle 3 kanonisessa järjestyksessä (ϵ, D_1, D_2, $\ldots$, D_r, D_1D_1, D_1D_2, $\ldots$, D_1D_r, D_2D_1, $\ldots$).

- Kutakin generoitua jonoa kohden $\widehat{M}$ simuloi yhden M:n osittaisen laskennan, jossa epädeterministiset valinnat tehdään kolmosnauhan koodijonon ilmaisemalla tavalla.
- Esimerkiksi jos kolmosnauhalla on jono $D_1D_3D_2$, niin ensimmäisessä siirtymässä valitaan vaihtoehto 1, toisessa vaihtoehto 3, kolmannessa vaihtoehto 2. Ellei tämä laskenta johtanut M:n hyväksyvään lopputilaan, generoidaan seuraava koodijono $D_1D_3D_3$ ja aloitetaan alusta.
- Jos koodijono on epäkelpo, so. jos siinä jossakin kohden on tilanteeseen liian suuri koodi, simuloitu laskenta keskeytetään ja generoidaan seuraava jono.
- Selvästi tämä systemaattinen koneen M laskentojen läpikäynti johtaa koneen $\widehat{M}$ hyväksymään syötejonon, jos ja vain jos koneella M on syötteen hyväksyvä laskenta. Jos hyväksyvää laskentaa ei ole, kone $\widehat{M}$ ei pysähdy. □



Kuva 5.13: Epädeterministisen Turingin koneen simulointi deterministisellä koneella.

5.4 Rajoittamattomat ja kontekstiset kieliopit

Rajoittamattomat kieliopit (*unrestricted grammars*) eli yleiset muunnossysteemit (*string rewriting systems*)

- yleistetään kontekstittomia kielioppeja sallimalla produktiossa yhden välikkeen sijaan minkä tahansa välikkeistä ja päätteistä koostuvan merkkijonon korvaaminen toisella (mahd. tyhjällä merkkijonolla)

- esim. $aB \to BC|d$ ja $a \to B|\epsilon$ ovat nyt sallittuja sääntöjä, mutta $\epsilon \to A$ ei ole.
- säännöt siis muotoa $\omega \to \omega'$, missä $\omega, \omega' \in V^*$ (V koko aakkosto) ja $\omega \neq \epsilon$
- Tärkeä tulos: **Kieli voidaan tuottaa rajoittamattomalla kieliopilla jos ja vain jos se voidaan tunnistaa Turingin koneella.**

Määritelmä: Rajoittamaton kielioppi on nelikko

$$G = (V, \Sigma, P, S),$$

missä

- V on kieliopin aakkosto;
- $\Sigma \subseteq V$ on kieliopin *päätemerkkien* joukko; $N = V - \Sigma$ on *välikemerkkien* t. *-symbolien* joukko;
- $P \subseteq V^+ \times V^*$ on kieliopin *sääntöjen* t. *produktioiden* joukko ($V^+ = V^* - \{\epsilon\}$);
- $S \in N$ on kieliopin *lähtösymboli.*

Produktiota $(\omega, \omega') \in P$ merkitään tavallisesti $\omega \to \omega'$.

- Merkkijono $\gamma \in V^*$ *tuottaa* t. *johtaa suoraan* merkkijonon $\gamma' \in V^*$ kieliopissa G, merkitään
$$\gamma \underset{G}{\Rightarrow} \gamma'$$
jos voidaan kirjoittaa $\gamma = \alpha\omega\beta$, $\gamma' = \alpha\omega'\beta$ ($\alpha, \beta, \omega' \in V^*$, $\omega \in V^+$), ja kieliopissa on produktio $\omega \to \omega'$.
- Jos kielioppi G on yhteydestä selvä, merkitään yksinkertaisesti $\gamma \Rightarrow \gamma'$.
- Merkkijono $\gamma \in V^*$ *tuottaa* t. *johtaa* merkkijonon $\gamma' \in V^*$ kieliopissa G, merkitään
$$\gamma \underset{G}{\Rightarrow}^* \gamma'$$
jos on olemassa jono V:n merkkijonoja $\gamma_0, \gamma_1, \ldots, \gamma_n$ ($n \geq 0$), siten että
$$\gamma = \gamma_0 \underset{G}{\Rightarrow} \gamma_1 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} \gamma_n = \gamma'.$$
- Jälleen, jos G on yhteydestä selvä, merkitään yksinkertaisesti $\gamma \Rightarrow^* \gamma'$.

- Merkkijono $\gamma \in V^*$ on kieliopin G *lausejohdos*, jos on $S \underset{G}{\Rightarrow}^* \gamma$.
- Pelkästään päätemerkeistä koostuva G:n lausejohdos $x \in \Sigma^*$ on G:n *lause*.
- Kieliopin G *tuottama* t. *kuvaama kieli* $L(G)$ koostuu G:n lauseista, s.o.:

$$L(G) = \{x \in \Sigma^* \mid S \underset{G}{\Rightarrow}^* x\}.$$

Esimerkki: rajoittamaton kielioppi ei-kontekstittomalle kielelle $\{a^k b^k c^k \mid k \geq 0\}$.

$$\begin{aligned}
S &\rightarrow LT \mid \epsilon \\
T &\rightarrow ABCT \mid ABC \\
BA &\rightarrow AB \\
CB &\rightarrow BC \\
CA &\rightarrow AC \\
LA &\rightarrow a \\
aA &\rightarrow aa \\
aB &\rightarrow ab \\
bB &\rightarrow bb \\
bC &\rightarrow bc \\
cC &\rightarrow cc.
\end{aligned}$$

Esimerkiksi lauseen *aabbcc* johto:

$$\begin{aligned}
\underline{S} &\Rightarrow \underline{LT} \Rightarrow LABC\underline{T} \Rightarrow LAB\underline{CA}BC \Rightarrow LA\underline{BA}CBC \\
&\Rightarrow LAAB\underline{CB}C \Rightarrow \underline{LA}ABBCC \Rightarrow \underline{aA}BBCC \\
&\Rightarrow a\underline{aB}BCC \Rightarrow aa\underline{bB}CC \Rightarrow aab\underline{bC}C \\
&\Rightarrow aabb\underline{cC} \Rightarrow aabbcc.
\end{aligned}$$

Lause: Jos formaali kieli L voidaan tuottaa rajoittamattomalla kieliopilla, se voidaan tunnistaa Turingin koneella.

**Todistus:*

Olkoon $G = (V, \Sigma, P, S)$ kielen L tuottava rajoittamaton kielioppi. Muodostetaan kielen L tunnistava kaksinauhainen epädeterministinen Turingin kone M_G seuraavasti:

- Nauhalla 1 kone säilyttää kopiota syötejonosta.

- Nauhalla 2 on kullakin hetkellä jokin G:n lausejohdos, jota kone pyrkii muuntamaan syötejonon muotoiseksi.
- Toimintansa aluksi M_G kirjoittaa kakkosnauhalle kieliopin lähtösymbolin S.
- Koneen M_G laskenta koostuu vaiheista. Kussakin vaiheessa kone:

  (i) vie kakkosnauhan nauhapään epädeterministisesti johonkin kohtaan nauhalla;

  (ii) valitsee epädeterministisesti jonkin G:n produktion, jota yrittää soveltaa valittuun nauhankohtaan (produktiot on koodattu M_G:n siirtymäfunktioon);

  (iii) jos produktion vasen puoli sopii yhteen nauhalla olevien merkkien kanssa, M_G korvaa ao. merkit produktion oikean puolen merkeillä;

  (iv) vaiheen lopuksi M_G vertaa ykkös- ja kakkosnauhan merkkijonoja toisiinsa: jos jonot ovat samat, kone siirtyy hyväksyvään lopputilaan ja pysähtyy, muuten aloittaa uuden vaiheen (kohta (i)). □

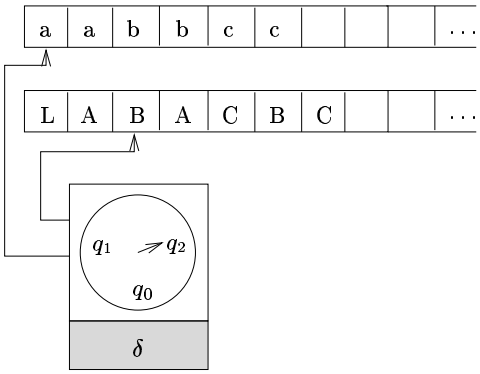
Lause: Jos formaali kieli L voidaan tunnistaa Turingin koneella, se voidaan tuottaa rajoittamattomalla kieliopilla.

**Todistus.* Olkoon $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}})$ kielen L tunnistava standardimallinen Turingin kone. Muodostetaan kielen L tuottava rajoittamaton kielioppi G_M seuraavasti.

Idea:

- Kieliopin G_M välikkeiksi otetaan (muiden muassa) kaikkia M:n tiloja $q \in Q$ edustavat symbolit.
- Koneen M tilanne $(q, u\underline{a}v)$ esitetään merkkijonona $[uqav]$.
- M:n siirtymäfunktion perusteella G_M:ään muodostetaan produktiot, joiden ansiosta
$$[uqav] \underset{G_M}{\Rightarrow} [u'q'a'v'] \quad \text{joss} \quad (q, uav) \underset{M}{\vdash} (q', u'a'v').$$
- Siten M hyväksyy syötteen x, jos ja vain jos
$$[q_0 x] \underset{G_M}{\Rightarrow}{}^* [uq_{\text{yes}}v]$$
joillakin $u, v \in \Sigma^*$.

Kaikkiaan kielioppiin G_M tulee kolme ryhmää produktioita:



Kuva 5.14: Rajoittamattoman kieliopin tuottaman kielen tunnistaminen Turingin koneella.

1. Produktiot, joilla lähtösymbolista S voidaan tuottaa mikä tahansa merkkijono muotoa $x[q_0x]$, missä $x \in \Sigma^*$ ja $[$, q_0 ja $]$ ovat G_M:n välikkeitä.
2. Produktiot, joilla merkkijonosta $[q_0x]$ voidaan tuottaa merkkijono $[uq_{\text{yes}}v]$, jos ja vain jos M hyväksyy x:n.
3. Produktiot, joilla muotoa $[uq_{\text{yes}}v]$ oleva merkkijono muutetaan tyhjäksi merkkijonoksi.

Kieleen $L(M)$ kuuluvan merkkijonon x tuottaminen tapahtuu tällöin seuraavasti:

$$S \overset{(1)}{\Rightarrow}{}^* x[q_0x] \overset{(2)}{\Rightarrow}{}^* x[uq_{\text{yes}}v] \overset{(3)}{\Rightarrow}{}^* x.$$

Täsmällisesti määritellään $G = (V, \Sigma, P, S)$, missä

$$V = \Gamma \cup Q \cup \{S, T, [,], E_L, E_R\} \cup \{A_a \mid a \in \Sigma\},$$

ja produktiot P muodostuvat seuraavista kolmesta ryhmästä:

1. Alkutilanteen tuottaminen:

$$\begin{array}{lcll} S & \to & T[q_0] & \\ T & \to & \epsilon & \\ T & \to & aTA_a & (a \in \Sigma) \\ A_a[q_0 & \to & [q_0A_a & (a \in \Sigma) \\ A_ab & \to & bA_a & (a, b \in \Sigma) \\ A_a] & \to & a] & (a \in \Sigma) \end{array}$$

2. M:n siirtymien simulointi ($a, b \in \Gamma$, $c \in \Gamma \cup \{[\}$) :

Siirtymät:			*Produktiot:*		
$\delta(q,a)$	$=$	(q',b,R)	qa	$\rightarrow$	bq'
$\delta(q,a)$	$=$	(q',b,L)	cqa	$\rightarrow$	$q'cb$
$\delta(q,>)$	$=$	$(q',>,R)$	$q[$	$\rightarrow$	$[q'$
$\delta(q,<)$	$=$	(q',b,R)	$q]$	$\rightarrow$	$bq']$
$\delta(q,<)$	$=$	(q',b,L)	$cq]$	$\rightarrow$	$q'cb]$
$\delta(q,<)$	$=$	$(q',<,L)$	$cq]$	$\rightarrow$	$q'c]$

3. Lopputilanteen siivous:

q_{yes}	$\rightarrow$	$E_L E_R$	
$q_{\text{yes}}[$	$\rightarrow$	E_R	
aE_L	$\rightarrow$	E_L	$(a \in \Gamma)$
$[E_L$	$\rightarrow$	ϵ	
$E_R a$	$\rightarrow$	E_R	$(a \in \Gamma)$
$E_R]$	$\rightarrow$	ϵ	

□

Esim. Tarkastellaan seuraavaa kasvikielioppia, johon on lisätty lähtösymboli S ja sille säänöt, jotta on saatu tavallinen rajoittamaton kielioppi. Kieliopin aakkosto on $V = \{S, SIEMEN, SIRKKALEHDET, LEHDET, VARSI, NUPPU, PUNKUKKA, SINKUKKA, PAIVA, YO\}$. (Huom! Koska tämä on kasvikielioppi, emme erottele pääte- ja välikesymboleja – ”jäsennys” jatkuu ikuisesti, ellei koko populaatio satu kuolemaan.)

$S \rightarrow$ SIEMEN PAIVA | SIEMEN YO
SIEMEN YO$\rightarrow$ SIRKKALEHDET YO
SRKKALEHDET PAIVA$\rightarrow$ LEHDET PAIVA| VARSI PAIVA
LEHDET PAIVA$\rightarrow$ VARSI PAIVA
VARSI PAIVA$\rightarrow$ NUPPU PAIVA| LEHDET PAIVA
NUPPU YO$\rightarrow$ PUNKUKKA YO| SINKUKKA YO
PUNKUKKA $\rightarrow$ SIEMEN | SIEMEN SIEMEN |ϵ
SINKUKKA $\rightarrow$ SIEMEN | SIEMEN SIEMEN |ϵ
PAIVA $\rightarrow$ YO
YO $\rightarrow$ PAIVA

Nyt lähtösymbolista S voidaan johtaa esim. seuraavat lausejohdokset:

$S \Rightarrow$ SIEMEN PAIVA$\Rightarrow$ SIEMEN YO $\Rightarrow$ SIRKKALEHDET YO $\Rightarrow$ SIRKKALEHDET PAIVA $\Rightarrow$ LEHDET PAIVA $\Rightarrow$ VARSI PAIVA $\Rightarrow$ NUPPU PAIVA $\Rightarrow$ NUPPU YO $\Rightarrow$ PUNKUKKA YO $\Rightarrow$ PUNKUKKA PAIVA $\Rightarrow$ PUNKUKKA YO $\Rightarrow$ SIEMEN SIEMEN YO $\Rightarrow$ ...

Kontekstiset kieliopit (*context-sensitive grammars*)

- Rajoittamaton kielioppi on *kontekstinen*, jos sen kaikki produktiot ovat muotoa $\omega \to \omega'$, missä $|\omega'| \geq |\omega|$, tai mahdollisesti $S \to \epsilon$, missä S on lähtösymboli.
- ts. lavennettaessa merkkijono voi vain kasvaa, ei koskaan pienentyä, paitsi lähtösymboli, joka saa tuottaa ϵ:in, jos ϵ kuuluu kieleen
- Lisäksi vaaditaan, että jos kieliopissa on produktio $S \to \epsilon$, niin lähtösymboli S ei esiinny minkään produktion oikealla puolella.
- Kontekstilliset kielet ovat siis rajoittamattomien kielten osaluokka!
- esim. säännöt
  tietojenkäsittelijä $\to$ *tietojenkäpistelijä*, *tieto* $\to$ *taito*
  $SUBJ$ *on* $PREDIKAT \to SUBJ$ *oli* $PREDIKAT | SUBJ$ *on ollut* $PREDIKAT |$
  $SUBJ$ *oli ollut* $PREDIKAT$

  ovat kontekstillisia
- Formaali kieli L on *kontekstinen*, jos se voidaan tuottaa jollakin kontekstisella kieliopilla.
- Normaalimuotolause: produktiot saadaan muotoon $S \to \epsilon$ ja $\alpha A \beta \to \alpha \omega \beta$, missä A on välike ja $\omega \neq \epsilon$. (Säännön $A \to \omega$ sovellus "kontekstissa" $\alpha \_ \beta$.)
- esim. $PAIVA\ SIEMEN \to PAIVA\ SIRKKALEHTI$, $PAIVA \to YO$: nyt sääntöä $SIEMEN \to SIRKKALEHTI$ saa soveltaa vain kontekstissa $PAIVA\_\epsilon$ (vain α muodostaa siis kontekstin ja β puuttuu)

Lause: Formaali kieli L on kontekstinen, jos ja vain jos se voidaan tunnistaa epädeterministisellä Turingin koneella, joka ei tarvitse enempää työtilaa kuin syötejonon pituuden verran — siis koneella, jolla ei ole muotoa $\delta(q, <) = (q', b, \Delta)$ olevia siirtymiä, missä $b \neq$ '$<$'.

Todistusidea: kontekstillisen kieliopin mukaisessa jäsennyksessä lähtösymboli voi vain kasvaa joka jäsennysaskeleella, joten jos lähdetään merkkijonosta kohti lähtösymbolia, voi symbolijono vain pienentyä joka jäsennysaskeleella. □

- Lauseen koneita sanotaan *lineaarisesti rajoitetuiksi automaateiksi* (*LBA=linear-bounded automaton*).
- Lineaarisesti rajoitetut automaatit tunnistavat siis täsmälleen kontekstilliset kielet.
- Avoin ongelma ("LBA = DLBA"): onko lauseessa vaadittu epädeterminismi välttämätöntä vai riittäisikö deterministinen kone?

Esimerkki Tiedämme, että kielen $L = \{a^k b^k c^k \mid k \geq 0\}$ voi tunnistaa syötteen vaatimassa tilassa (muutetaan merkkejä A:ksi, B:ksi ja C:ksi yksi kerrallaan), joten se on kontekstillinen. Voidaan antaa seuraava kielen kuvaava kontekstillinen kielioppi:

$S' \to S|\epsilon$
$S \to aSBc|abc$
$cB \to Bc$
$bB \to bb$

Esim. merkkijonon *aaabbbccc* johto:
$\underline{S} \Rightarrow a\underline{S}Bc \Rightarrow aa\underline{S}BcBc \Rightarrow aaab\underline{cB}cBc \Rightarrow aaa\underline{bB}ccBc \Rightarrow aaabbc\underline{cB}c \Rightarrow aaabb\underline{cB}cc \Rightarrow aaab\underline{bB}ccc \Rightarrow aaabbbccc$