T-79.1001/1002
Tietojenkäsittelyteorian perusteet T/Y

Harri Haanpää

Tietojenkäsittelyteorian laboratorio, TKK

Syksy 2007

T–79.1001/1002 Tietojenkäsittelyteorian perusteet T/Y
Introduction to Theoretical Computer Science T/Y

What can one do with a computer?

This course divides roughly into three parts. We examine three models of a computer, their power and limitations, i.e., the classes of languages that they can decide.

It turns out that each class of *automata* corresponds to a class of *grammars*. The classes of automata and grammars are:

1. Finite state automata & regular expressions
2. Pushdown automata & context-free grammars
3. Turing machines & unrestricted grammars

The first teaching period of the course (=Y version) covers finite state automata, regular expressions and context-free grammars. The second period covers the rest.

Course versions

There are two versions of the course for different kinds of students. If you are unsure, check the study guide or ask your study advisor.

T-79.1001 Introduction to TCS T, periods I-II, 4 cr
- compulsory in the computer science candidate program;
- replaces T-79.148, compulsory in many specialisations of the pre-2005 telecommunications engineering curriculum;
- in B1 module for studying Theoretical Computer Science as a minor;
- in the Discrete Mathematics B1 module

T-79.1002 Intro to TCS Y, period I, 2 cr
- compulsory in the 2005 data and communications candidate program;
- part of bridge studies for polytechnic graduates to enter computer science program

Practical arrangements

Registration: obligatory, by TOPI

Lectures: Tue 14–16 T1, in Finnish by Harri Haanpää

Tutorials: once a week, register by TOPI

- Thu 16-18 (in English)
- Mon 10-12
- Mon 12-14
- Mon 14-16 (period I only)
- Mon 16-18 (period I only)
- Tue 12-14

Computer exercises: obligatory, using WWW system "STRATUM"

Course home page:

`http://www.tcs.hut.fi/Studies/T-79.1001/`
`http://www.tcs.hut.fi/Studies/T-79.1002/`

Course newsgroup: `opinnot.tik.tkt` on `news.tky.hut.fi`

- Please ask (and answer!) your questions there!

To pass the course

1. Pass the computer exercises *before taking the exam*.
2. *After passing the computer exercises* take the exam (exams typically in May, Aug, Oct, Dec, Mar). Maximum score in exam is 60 for T course, 40 for Y course.
3. There are 3 homework problems a week; you gain bonus points for exam according to table:

# solved:	0–4	5–9	10–14	15–19	20–24	25–29	30–
T-79.1001	−2	−1	+0	+1	+2	+3	+4
T-79.1002	−1	+0	+1	+2	N/A		

4. By completing the computer assignments by 18 Oct 13:00, one can earn a bonus of 2 points for the exam.
5. Minimum score for various grades may vary. However, with 30 points (T) or 20 points (Y) passing is guaranteed.

Honour code

- Solve your computer assignments yourself
- Solve tutorial problems yourself or in small groups
  - . . . but do not tolerate free riders.
  - An honest try is better than a copied model solution.
- If someone asks for your help, do not give away the problem
  - . . . give a little hint to point him/her in the right direction.

Material

Lecture notes (in Finnish) and solutions of *demonstration* exercises (in Finnish) are distributed through Edita.

Recommended textbook Michael Sipser, Introduction to the Theory of Computation, PWS Publishing 1997. (Supplementary for Finnish-speaking students; likely necessary for non-Finnish-speaking students.)

Some additional material on the course WWW page.

TIETOJENKÄSITTELYTEORIA

- Matemaattinen oppi siitä, mitä tietokoneella on mahdollista tehdä ja kuinka tehokkaasti.
- Tarjoaa matemaattisia käsitteitä ja menetelmiä tietojenkäsittelyjärjestelmien mallintamiseen ja analysointiin sekä selkeiden ja tehokkaiden ratkaisujen laatimiseen.

Tietojenkäsittelyteorian osa-aloja

Laskettavuusteoria

Mitä tietokoneella voi tehdä periaatteessa?

- Turing, Gödel, Church, Post (1930-luku); Kleene, Markov (1950-luku).

Laskennan vaativuusteoria

Mitä tietokoneella voi tehdä käytännössä?

- Hartmanis, Stearns (1960-luku); Cook, Levin, Karp (1970-luku); Papadimitriou, Sipser, Håstad, Razborov ym. (1980-).

Automaatti- ja kielioppiteoria

Tietojenkäsittelyjärjestelmien perustyyppien ominaisuudet ja kuvausformalismit.

- Chomsky (1950-luku); Ginsburg, Greibach, Rabin, Salomaa, Schützenberger ym. (1960-luku)

Tietojenkäsittelyteorian osa-aloja

Ohjelmien oikeellisuus

Tietojenkäsittelyjärjestelmien matemaattisesti eksakti määrittely ja oikean toiminnan verifiointi.

- Dijkstra, Hoare (1960-luku); Manna, Pnueli, Scott ym. (1970-).

Muuta

- algoritmien suunnittelu ja analyysi (Knuth, Hopcroft, Tarjan ym.)
- kryptologia (Rivest, Shamir, Adleman ym.)
- rinnakkaisten ja hajautettujen järjestelmien teoria (Lamport, Lynch, Milner, Valiant ym.)
- koneoppimisteoria (Valiant ym.)
- jne.

Tällä kurssilla käsitellään lähinnä automaatteja ja kielioppeja sekä hieman laskettavuusteorian alkeita. Muita aiheita käsitellään Tietojenkäsittelyteorian laboratorion muilla kursseilla.

1. Matemaattisia peruskäsitteitä

1.1 Joukot

Joukko (engl. set) on kokoelma alkioita. Alkiot voidaan ilmoittaa joko luettelemalla, esim.

$$S = \{2, 3, 5, 7, 11, 13, 17, 19\}$$

tai jonkin säännön avulla, esim.

$$S = \{p \mid p \text{ on alkuluku, } 2 \leq p \leq 20\}.$$

Jos alkio a kuuluu joukkoon A, merkitään $a \in A$, päinvastaisessa tapauksessa $a \notin A$. (Esim. $3 \in S$, $8 \notin S$.)

Tärkeä erikoistapaus on *tyhjä joukko* (engl. empty set) $\emptyset$, johon ei kuulu yhtään alkiota.

Jos joukon A kaikki alkiot kuuluvat myös joukkoon B, sanotaan että A on B:n *osajoukko* (engl. subset) ja merkitään $A \subseteq B$. [Kirjallisuudessa esiintyy myös merkintä $A \subset B$.] Jos A *ei* ole B:n osajoukko merkitään $A \not\subseteq B$. Siis esim.

$$\{2,3\} \subseteq S, \qquad \{1,2,3\} \not\subseteq S.$$

Triviaalisti on voimassa $\emptyset \subseteq A$ kaikilla A.

Joukot A ja B ovat samat, jos niissä on samat alkiot, so. jos on $A \subseteq B$ ja $B \subseteq A$. Jos on $A \subseteq B$, mutta $A \neq B$, sanotaan että A on B:n *aito osajoukko* (engl. proper subset) ja merkitään $A \subsetneq B$. Edellä olisi siis voitu myös kirjoittaa $\{2,3\} \subsetneq S$ ja $\emptyset \subsetneq A$ jos $A \neq \emptyset$.

Joukon alkioina voi olla myös toisia joukkoja (tällöin puhutaan usein "joukkoperheestä"), esim.

$$X = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}.$$

Jonkin perusjoukon A kaikkien osajoukkojen muodostamaa joukkoperhettä sanotaan A:n *potenssijoukoksi* (engl. powerset) ja merkitään $\mathcal{P}(A)$:lla; esim. edellä on $X = \mathcal{P}(\{1,2\})$. [Koska n-alkioisen perusjoukon A potenssijoukossa on 2^n alkiota (HT), käytetään kirjallisuudessa potenssijoukolle myös merkintää 2^A.]

Huomaa, että $A \subseteq B$ jos ja vain jos $A \in \mathcal{P}(B)$.

Joukkoja voidaan kombinoida *joukko-operaatioilla*, joista tärkeimmät ovat:

yhdiste (engl. union)

$$A \cup B = \{x \mid x \in A \text{ tai } x \in B\},$$

esim. $\{1,2,3\} \cup \{1,4\} = \{1,2,3,4\}$.

leikkaus (engl. intersection)

$$A \cap B = \{x \mid x \in A \text{ ja } x \in B\},$$

esim. $\{1,2,3\} \cap \{1,4\} = \{1\}$.

erotus (engl. difference)

$$A - B = \{x \mid x \in A \text{ ja } x \notin B\},$$

esim. $\{1,2,3\} - \{1,4\} = \{2,3\}$.
[Erotukselle käytetään myös merkintää $A \setminus B$.]

Joukko-operaatioita koskevat tietyt laskulait, joista tärkeimmät ovat yhdisteen ja leikkauksen *liitännäisyys*:

$$A \cup (B \cup C) = (A \cup B) \cup C, \quad A \cap (B \cap C) = (A \cap B) \cap C$$

ja *vaihdannaisuus*:

$$A \cup B = B \cup A, \quad A \cap B = B \cap A$$

sekä näiden *osittelulait*:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C),$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C).$$

Jos kaikki tarkasteltavat joukot ovat jonkin yhteisen "universaalijoukon" U osajoukkoja, sanotaan erotusta $U - A$ joukon A *komplementiksi* (U:n suhteen) ja merkitään $\bar{A}$:lla.

Yhdiste-, leikkaus- ja komplementointioperaatioita yhdistävät tärkeät ns. *de Morganin kaavat*:

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

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

Lisäksi joukkojen erotus voidaan esittää leikkauksen ja komplementoinnin avulla seuraavasti:

$$A - B = A \cap \bar{B}.$$

Jos joukkoperheen $\mathcal{A}$ jäsenet on *indeksoitu*, esim.

$$\mathcal{A} = \{A_1, A_2, A_3, \ldots\},$$

niin yhdisteelle ja leikkaukselle voidaan käyttää lyhennemerkintöjä

$$\bigcup_{i \geq 1} A_i = A_1 \cup A_2 \cup A_3 \cdots \qquad \text{ja} \qquad \bigcap_{i \geq 1} A_i = A_1 \cap A_2 \cap A_3 \cdots$$

Indeksien ei tarvitse olla edes luonnollisia lukuja, vaan *indeksijoukkona* voi olla mikä tahansa joukko I. Tällöin käytetään merkintöjä

$$\mathcal{A} = \{A_i \mid i \in I\}$$

ja

$$\bigcup_{i \in I} A_i, \qquad \bigcap_{i \in I} A_i.$$

1.2 Relaatiot ja funktiot

Olkoot A ja B joukkoja. Alkioiden $a \in A$ ja $b \in B$ *järjestettyä paria* (engl. ordered pair) merkitään (a, b). Huomaa, että joukkoina on aina $\{a, b\} = \{b, a\}$, mutta jos $a \neq b$, niin järjestettyinä pareina on $(a, b) \neq (b, a)$.

Joukkojen A ja B *karteesinen tulo* (engl. Cartesian product) määritellään

$$A \times B = \{(a, b) \mid a \in A \text{ ja } b \in B\},$$

esim.

$$\begin{aligned} &\{1,2,3\} \times \{1,4\} \\ &\quad = \quad \{(1,1),(1,4),(2,1),(2,4),(3,1),(3,4)\}. \end{aligned}$$

Relaatio R joukolta A joukolle B on karteesisen tulon $A \times B$ osajoukko:

$$R \subseteq A \times B.$$

Jos $(a, b) \in R$, niin merkitään myös aRb ja sanotaan että alkio a on *relaatiossa (suhteessa) R* alkioon b. Tätä *infix*-merkintää käytetään varsinkin silloin, kun relaation nimenä on jokin erikoismerkki, esim. $\leq$, $\prec$, $\equiv$, $\sim$.

Jos relaation R *lähtöjoukko* (engl. domain) A ja *maalijoukko* (engl. range) B ovat samat, so. $R \subseteq A \times A$, sanotaan että R on relaatio *joukossa A*.

Relaation $R \subseteq A \times B$ *käänteisrelaatio* (engl. inverse relation) on relaatio $R^{-1} \subseteq B \times A$,

$$R^{-1} = \{(b,a) \mid (a,b) \in R\}.$$

Jos $R \subseteq A \times B$ ja $S \subseteq B \times C$ ovat relaatioita, niin niiden *yhdistetty relaatio* (engl. composite relation) $R \circ S \subseteq A \times C$ määritellään:

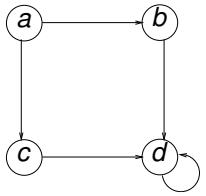
$$R \circ S = \{(a,c) \mid \exists b \in B \text{ s.e. } (a,b) \in R, (b,c) \in S\}.$$

Varsinkin jos joukot A ja B ovat äärellisiä, relaatiota $R \subseteq A \times B$ voi olla havainnollista tarkastella *suunnattuna verkkona* t. *graafina*, jonka *solmuina* ovat joukkojen A ja B alkiot ja solmusta $a \in A$ on *kaari* (“nuoli”) solmuun $b \in B$, jos ja vain jos $(a,b) \in R$.

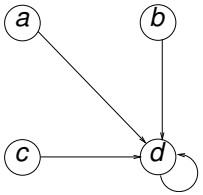
Olkoon esimerkiksi joukossa $A = \{a,b,c,d\}$ määritelty relaatio $R \subseteq A \times A$,

$$R = \{(a,b),(a,c),(b,d),(c,d),(d,d)\}.$$

Relaation R graafiesitys on



Relaation $R \circ R$ graafiesitys puolestaan on



Relaatio $f \subseteq A \times B$ on *funktio*, jos kukin $a \in A$ on relaatiossa f täsmälleen yhden $b \in B$ kanssa. Tällöin käytetään yleisten relaatiomerkintöjen sijaan tavallisesti merkintöjä $f : A \to B$ ja $f(a) = b$.

Funktioita koskee kaikki mitä edellä yleisesti on todettu relaatioista, mutta historiallisista syistä funktioiden yhdistäminen merkitään toisin päin kuin yleisten relaatioiden: jos $f : A \to B$ ja $g : B \to C$ ovat funktioita, niin niiden yhdistetty funktio määritellään kaavalla $(g \circ f)(a) = g(f(a))$, so. relaatioina

$$g \circ f = \{(a,c) \mid \exists b \in B \text{ s.e. } f(a) = b, g(b) = c\}.$$

Olkoon $f : A \to B$ funktio.

- f on *surjektio* (engl. onto map), jos jokainen $b \in B$ on jonkin $a \in A$ kuva, so. jos $f(A) = B$.
- f on *injektio* (engl. one-to-one map), jos kaikki $a \in A$ kuvautuvat eri alkioille, so. jos $a \neq a' \Rightarrow f(a) \neq f(a')$.
- f on *bijektio*, jos se on sekä injektio että surjektio, so. jos jokainen $b \in B$ on yhden ja vain yhden $a \in A$ kuva.

1.3 Ekvivalenssirelaatiot

Ekvivalenssirelaatiot ovat matemaattisesti täsmällinen muotoilu sille yleiselle idealle, että oliot ovat keskenään *samankaltaisia* jonkin kiinnostavan ominaisuuden X suhteen. Ominaisuuteen X perustuva ekvivalenssirelaatio osittaa tarkasteltavien olioiden joukon *ekvivalenssiluokkiin*, jotka vastaavat ominaisuuden X eri arvoja. (Kääntäen mielivaltainen olioiden joukon ositus Π määrää tietyn abstraktin samankaltaisuusominaisuuden, nim. sen että oliot ovat samankaltaisia jos ne sijoittuvat samaan osituksen Π luokkaan.)

Osoittautuu, että yleinen “samankaltaisuusrelaation” idea voidaan kiteyttää seuraaviin kolmeen ominaisuuteen.

Määritelmä 1.1 Relaatio $R \subseteq A \times A$ on

1. *refleksiivinen*, jos $aRa\ \forall a \in A$;
2. *symmetrinen*, jos $aRb \Rightarrow bRa\ \forall a, b \in A$;
3. *transitiivinen*, jos $aRb, bRc \Rightarrow aRc\ \forall a, b, c \in A$.

Määritelmä 1.2 Relaatio $R \subseteq A \times A$, joka toteuttaa edelliset ehdot 1–3 on *ekvivalenssirelaatio*. Alkion $a \in A$ *ekvivalenssiluokka* (relaation R suhteen) on

$$R[a] = \{x \in A \mid aRx\}.$$

Ekvivalenssirelaatioita merkitään usein R:n sijaan alkioiden samankaltaisuutta korostavilla symboleilla $\sim$, $\equiv$, $\simeq$ tms.

Esim. Olkoon

$$A = \{\text{kaikki 1900-luvulla syntyneet ihmiset}\}$$

ja aRb voimassa, jos henkilöillä a ja b on sama syntymävuosi. Tällöin R on selvästi ekvivalenssi, jonka ekvivalenssiluokat koostuvat keskenään samana vuonna syntyneistä henkilöistä. Luokkia on 100 kappaletta, ja “abstraktisti” ne vastaavat 1900-luvun vuosia 1900,...,1999.

Lemma 1.3 Olkoon $R \subseteq A \times A$ ekvivalenssi. Tällöin on kaikilla $a, b \in A$ voimassa:

$$R[a] = R[b] \quad \text{joss} \quad aRb.$$

Tod. Helppo; sivuutetaan. □

Lemma 1.4 Olkoon $R \subseteq A \times A$ ekvivalenssi. Tällöin R:n ekvivalenssiluokat muodostavat A:n *osituksen* erillisiin epätyhjiin osajoukkoihin, so.:

- $R[a] \neq \emptyset$ kaikilla $a \in A$;
- $A = \bigcup_{a \in A} R[a]$;
- jos $R[a] \neq R[b]$, niin $R[a] \cap R[b] = \emptyset$, kaikilla $a, b \in A$.

Tod. Helppo; sivuutetaan. □

Kääntäen jokainen perusjoukon A ositus erillisiin epätyhjiin luokkiin A_i, $i \in I$, määrää vastaavan ekvivalenssirelaation:

$$a \sim b \quad \Leftrightarrow \quad a \text{ ja } b \text{ kuuluvat samaan luokkaan } A_i.$$

1.5 Induktiopäättely

Olkoon $P(k)$ jokin luonnollisten lukujen ominaisuus. Jos on voimassa:

1. $P(0)$ ja
2. kaikilla $k \geq 0$:
$$P(k) \Rightarrow P(k+1),$$

niin $P(n)$ on tosi kaikilla $n \in \mathbb{N}$. □

Esimerkki.

Väite. Kaikilla $n \in \mathbb{N}$ on voimassa kaava

$$P(n): \quad (1+2+\cdots+n)^2 = 1^3+2^3+\cdots+n^3.$$

Todistus.

1. Perustapaus: $P(0): \quad 0^2 = 0.$

(jatkuu)

2. Induktioaskel: Oletetaan, että annetulla $k \geq 0$ kaava

$$P(k): \quad (1+2+\cdots+k)^2 = 1^3+2^3+\cdots+k^3$$

on voimassa. Tällöin on myös:

$$\begin{aligned}
&(1+2+\cdots+k+(k+1))^2 \\
&= (1+\cdots+k)^2 + 2(1+\cdots k)(k+1) + (k+1)^2 \\
&= 1^3+\cdots+k^3 + 2\cdot\frac{k(k+1)}{2}\cdot(k+1) + (k+1)^2 \\
&= 1^3+\cdots+k^3 + k(k+1)^2 + (k+1)^2 \\
&= 1^3+\cdots+k^3 + (k+1)^3.
\end{aligned}$$

On siis todettu, että kaavan $P(k)$ totuudesta seuraa kaavan $P(k+1)$ totuus, so. että $P(k) \Rightarrow P(k+1)$, kaikilla $k \geq 0$. Luonnollisten lukujen induktioperiaatteen 1.9 nojalla voidaan nyt päätellä, että kaava $P(n)$ on voimassa kaikilla $n \in \mathbb{N}$. □

1.6 Aakkostot, merkkijonot ja kielet

Automaattiteoria $\sim$ diskreetin signaalinkäsittelyn perusmallit ja -menetelmät

($\sim$ diskreettien I/O-kuvausten yleinen teoria)

Automaatin käsite on *matemaattinen abstraktio*. Yleisellä tasolla suunniteltu automaatti voidaan toteuttaa eri tavoin: esim. sähköpiirinä, mekaanisena laitteena tai (tavallisimmin) tietokoneohjelmana.

Tällä kurssilla keskitytään pääosin automaatteihin, joiden:

1. syötteet ovat äärellisiä, diskreettejä *merkkijonoja*
2. tulokset ovat muotoa “hyväksy”/”hylkää” ($\sim$ “syöte OK”/”syöte ei kelpaa”)

Yleistyksiä:

- äärettömät syötejonot ($\rightarrow$ “reaktiiviset” järjestelmät, Büchi-automaatit)
- funktioautomaatit ($\rightarrow$ Moore- ja Mealy-tilakoneet, Turingin funktiokoneet)

Peruskäsitteitä ja merkintöjä

Aakkosto (engl. alphabet, vocabulary): äärellinen, epätyhjä joukko *alkeismerkkejä* t. *symboleita*. Esim.:

- *binääriaakkosto* $\{0,1\}$;
- *latinalainen aakkosto* $\{\mathrm{A},\mathrm{B},\ldots,\mathrm{Z}\}$.

Merkkijono (engl. string): äärellinen järjestetty jono jonkin aakkoston merkkejä. Esim.:

- “01001”, “0000”: binääriaakkoston merkkijonoja;
- “TKTP”, “XYZZY”: latinalaisen aakkoston merkkijonoja.
- *tyhjä merkkijono* (engl. empty string). Tyhjässä merkkijonossa ei ole yhtään merkkiä; sitä voidaan merkitä ε:llä.

Merkkijonon x *pituus* $|x|$ on sen merkkien määrä. Esim.: $|01001| = 5$, $|\mathrm{TKTP}| = 4$, $|\varepsilon| = 0$.

Merkkijonojen välinen perusoperaatio on *katenaatio* eli jonojen peräkkäin kirjoittaminen. Katenaation operaatiomerkkinä käytetään joskus selkeyden lisäämiseksi symbolia $\widehat{\ }$. Esim.

- KALA$\widehat{\ }$KUKKO = KALAKUKKO;
- jos $x = 00$ ja $y = 11$, niin $xy = 0011$ ja $yx = 1100$;
- kaikilla x on $x\varepsilon = \varepsilon x = x$;
- kaikilla x, y, z on $(xy)z = x(yz)$;
- kaikilla x, y on $|xy| = |x| + |y|$.

Aakkoston Σ kaikkien merkkijonojen joukkoa merkitään Σ^*:lla. Esimerkiksi jos $\Sigma = \{0,1\}$, niin $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, \ldots\}$.

Mielivaltaista merkkijonojoukkoa $A \subseteq \Sigma^*$ sanotaan aakkoston Σ *(formaaliksi) kieleksi* (engl. formal language).

Automaatit ja formaalit kielet

Olkoon M automaatti, jonka syötteet ovat jonkin aakkoston Σ merkkijonoja, ja tulos on yksinkertaisesti muotoa "syöte hyväksytään"/"syöte hylätään". (Merk. lyhyesti $1/0$.)

Merkitään M:n syötteellä x antamaa tulosta $M(x)$:llä ja M:n hyväksymien syötteiden joukkoa A_M:llä, so.

$$A_M = \{x \in \Sigma^* \mid M(x) = 1\}.$$

Sanotaan, että automaatti M *tunnistaa* (engl. recognises) kielen $A_M \subseteq \Sigma^*$.

Automaattiteorian (yksi) idea: *automaatin M rakenne heijastuu kielen A_M ominaisuuksissa.*

Kääntäen: olkoon annettuna jokin toivottu I/O-kuvaus $f : \Sigma^* \to \{0,1\}$. Tarkastelemalla kieltä

$$A_f = \{x \in \Sigma^* \mid f(x) = 1\}$$

saadaan vihjeitä siitä, millainen automaatti tarvitaan kuvauksen f toteuttamiseen.

Vakiintuneita merkintöjä

Em. matemaattisille käsitteille käytetyt merkinnät ovat periaatteessa vapaasti valittavissa, mutta esityksen ymmärrettävyyden parantamiseksi on tapana pitäytyä tietyissä käytännöissä. Seuraavat merkintätavat ovat vakiintuneet:

Aakkostot: Σ, Γ, ... (isoja kreikkalaisia kirjaimia). Esim. binääriaakkosto $\Sigma = \{0,1\}$.

Aakkoston koko (tai yleisemmin joukon mahtavuus): $|\Sigma|$.

Alkeismerkit: a, b, c, ... (pieniä alkupään latinalaisia kirjaimia). Esim.: Olkoon $\Sigma = \{a_1, \ldots, a_n\}$ aakkosto; tällöin $|\Sigma| = n$.

Merkkijonot: u, v, w, x, y, ... (pieniä loppupään latinalaisia kirjaimia).

Merkkijonojen katenaatio: $x\widehat{\ }y$ tai vain xy.

Merkkijonon pituus: $|x|$. *Esimerkkejä:*

- $|abc| = 3$;
- olkoon $x = a_1 \ldots a_m$, $y = b_1 \ldots b_n$; tällöin $|xy| = m + n$.

Tyhjä merkkijono: ε.

Merkkijono, jossa on n kappaletta merkkiä a: a^n. Esimerkkejä:

- $a^n = \underbrace{aa\ldots a}_{n\,\text{kpl}}$;
- $|a^i b^j c^k| = i + j + k$.

Merkkijonon x toisto k kertaa: x^k. Esimerkkejä:

- $(ab)^2 = abab$;
- $|x^k| = k|x|$.

Aakkoston Σ kaikkien merkkijonojen joukko: Σ^*. Esim.:

- $\{a,b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$.

Merkkijonoinduktio

Automaattiteoriassa tehdään usein konstruktioita "induktiolla merkkijonon pituuden suhteen." Tämä tarkoittaa, että määritellään ensin toiminto tyhjän merkkijonon ε (tai joskus yksittäisen aakkosmerkin) tapauksessa. Sitten oletetaan, että toiminto on määritelty kaikilla annetun pituisilla merkkijonoilla u ja esitetään, miten se tällöin määritellään yhtä merkkiä pitemmillä merkkijonoilla $w = ua$.

Esimerkki. Olkoon Σ mielivaltainen aakkosto. Merkkijonon $w \in \Sigma^*$ *käänteisjono* (engl. reversal) w^R määritellään induktiivisesti säännöillä:

1. $\varepsilon^R = \varepsilon$;
2. jos $w = ua$, $u \in \Sigma^*$, $a \in \Sigma$, niin $w^R = a\widehat{\ }u^R$.

Induktiivista (“rekursiivista”) määritelmää voidaan tietenkin käyttää laskujen perustana; esim.:

$$\begin{array}{rclcl} (011)^R & = & 1\widehat{\ }(01)^R & = & 1\widehat{\ }(1\widehat{\ }0^R) \\ & = & 11\widehat{\ }(0\widehat{\ }\varepsilon^R) & = & 110\widehat{\ }\varepsilon^R \\ & = & 110\widehat{\ }\varepsilon & = & 110. \end{array}$$

Tärkeämpää on kuitenkin konstruktioiden ominaisuuksien todistaminen määritelmää noudattelevalla induktiolla. Esimerkki:

Väite. Olkoon Σ aakkosto. Kaikilla $x, y \in \Sigma^*$ on voimassa $(xy)^R = y^R x^R$.

Todistus. Induktio merkkijonon y pituuden suhteen.

1. Perustapaus $y = \varepsilon$: $(x\varepsilon)^R = x^R = \varepsilon^R x^R$.
2. Induktioaskel: Olkoon y muotoa $y = ua$, $u \in \Sigma^*$, $a \in \Sigma$. Oletetaan, että väite on voimassa merkkijonoilla x, u. Tällöin on:

$$\begin{array}{rcll} (xy)^R & = & (xua)^R & \\ & = & a\widehat{\ }(xu)^R & [R\text{:n määritelmä}] \\ & = & a\widehat{\ }(u^R x^R) & [\text{induktio-oletus}] \\ & = & (a\widehat{\ }u^R)x^R & [\widehat{\ }\text{:n liitännäisyys}] \\ & = & (ua)^R x^R & [R\text{:n määritelmä}] \\ & = & y^R x^R. & \square \end{array}$$

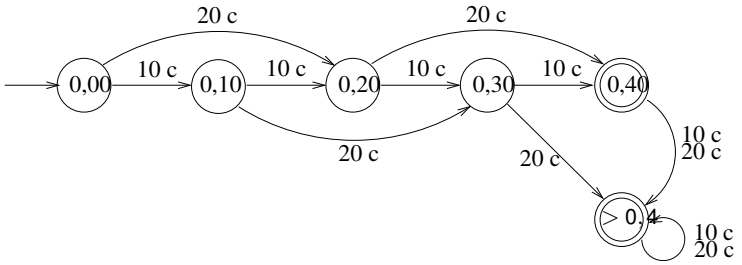
ÄÄRELLISET AUTOMAATIT JA SÄÄNNÖLLISET KIELET

2.1 Tilakaaviot ja tilataulut

Tarkastellaan aluksi tietojenkäsittelyjärjestelmiä, joilla on vain äärellisen monta mahdollista tilaa. Tällaisen järjestelmän toiminta voidaan kuvata *äärellisenä automaattina* t. *äärellisenä tilakoneena* (engl. finite automaton, finite state machine).

Äärellisillä automaateilla on useita vaihtoehtoisia esitystapoja: tilakaaviot, tilataulut, ...

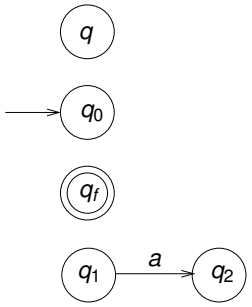
Esimerkki 1: Kahviautomaatti.



Em. tilakaavion esittämä automaatti ratkaisee päätösongelman “riittävätkö annetut rahat kahvin ostamiseen?”

Äärellisiä automaatteja voidaan yleensäkin käyttää yksinkertaisten päätösongelmien ratkaisujen mallintamiseen. Automaattimallista on muitakin kuin binäärivasteisten järjestelmien kuvaamiseen tarkoitettuja versioita (ns. Moore- ja Mealy-automaatit), mutta niitä ei käsitellä tällä kurssilla.

Tilakaavioiden merkinnät:



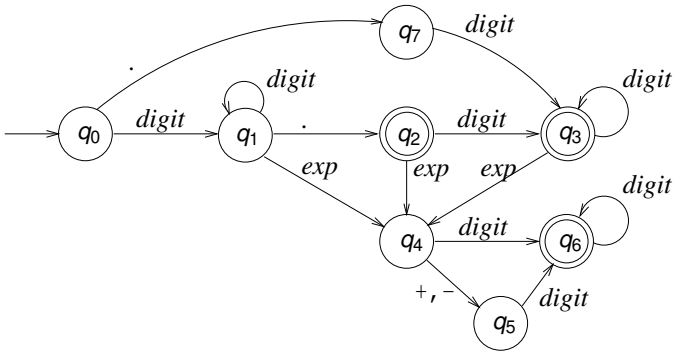
Automaatin tila nimeltä q

Alkutila

Lopputila: automaatti "hyväksyy" syötejonon, joss se jonon loppuessa on tällaisessa tilassa

Syötemerkin a aikaansaama siirtymä tilasta q_1 tilaan q_2

Esimerkki 2: C-kielen etumerkittömät reaaliluvut.



Käytetyt lyhenteet: *digit* $= \{0,1,\ldots,9\}$, *exp* $= \{\mathtt{E},\mathtt{e}\}$.

Äärellisen automaatin esitys *tilatauluna*: automaatin uusi tila vanhan tilan ja syötemerkin funktiona.

Esim. reaalilukuautomaatin tilataulu:

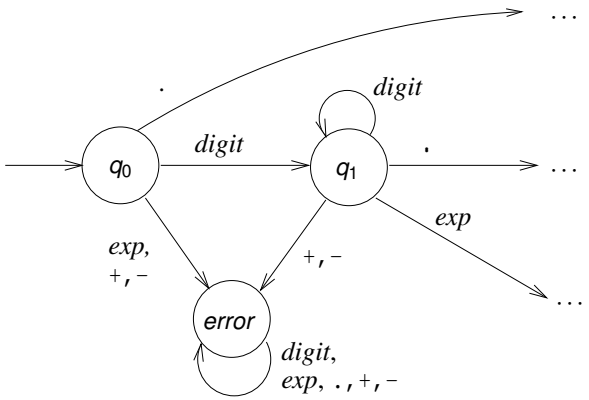
		digit	.	*exp*	+	−
→	q_0	q_1	q_7			
	q_1	q_1	q_2	q_4		
←	q_2	q_3		q_4		
←	q_3	q_3		q_4		
	q_4	q_6			q_5	q_5
	q_5	q_6				
←	q_6	q_6				
	q_7	q_3				

K: Mitä tilataulun tyhjät paikat tarkoittavat?

V: Tilataulun tyhjät paikat, tai vastaavasti tilakaavion "puuttuvat" kaaret, kuvaavat automaatin virhetilanteita. Jos automaatti ohjautuu tällaiseen paikkaan, syötejono ei kuulu automaatin hyväksymään joukkoon.

Muodollisesti automaatissa ajatellaan olevan erityinen virhetila, jota ei vain selkeyden vuoksi merkitä näkyviin.

Esim. reaalilukuautomaatin täydellinen kaavioesitys olisi:



ja reaalilukuautomaatin täydellinen tauluesitys olisi:

		digit	.	*exp*	+	−
→	q_0	q_1	q_7	*error*	*error*	*error*
	q_1	q_1	q_2	q_4	*error*	*error*
⋮	⋮	⋮	⋮	⋮	⋮	⋮
←	q_6	q_6	*error*	*error*	*error*	*error*
	error	*error*	*error*	*error*	*error*	*error*

2.2 Äärellisiin automaatteihin perustuva ohjelmointi

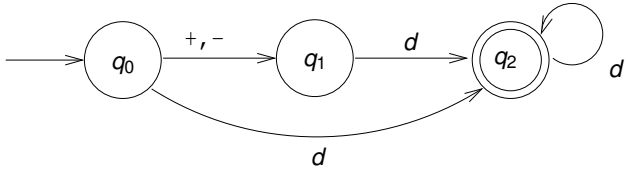
Annetun äärellisen automaatin pohjalta on helppo laatia automaatin toimintaa vastaava ohjelma. Esim. reaalilukuautomaattiin perustuva syötejonon syntaksitestaus:


```
from sys import stdin

q=0
for c in stdin.readline().strip("\n"):
    if q==0:
        if c.isdigit(): q=1
        elif c==".": q=7
        else: q=99
    elif q==1:
        if c.isdigit(): q=1
        elif c==".": q=2
        elif c=="E" or c=="e": q=4
        else: q=99
    elif q==2 or q==3:
        if c.isdigit(): q=3
        elif c=="E" or c=="e": q=4
        else: q=99
```

```
    elif q==4:
        if c.isdigit(): q=6
        elif c=="+" or c=="-": q=5
        else: q=99
    elif q==5 or q==6:
        if c.isdigit(): q=6
        else: q=99
    elif q==7:
        if c.isdigit(): q=3
        else: q=99

if q in [2,3,6]: print "On reaaliluku."
else: print "Ei ole reaaliluku."
```


Semanttisten toimintojen liittäminen äärellisiin automaatteihin

Esimerkki. Kahdeksanjärjestelmän lukuja tunnistava automaatti ja siihen perustuva syöteluvun arvonmääritys (“muuttaminen kymmenjärjestelmään”).



Lyhennysmerkintä $d = \{0,1,\ldots,7\}$.

Pelkän syntaksitestin toteutus:

```
from sys import stdin

q=0
for c in stdin.readline().strip("\n"):
    if q==0:
        if c=="+" or c=="-": q=1
        elif c in "01234567": q=2
        else: q=99
    elif q==1:
        if c in "01234567": q=2
        else: q=99
    elif q==2:
        if c in "01234567": q=2
        else: q=99
if q == 2 print "Oktaaliluku!"
else: print "Virheellinen oktaaliluku."
```

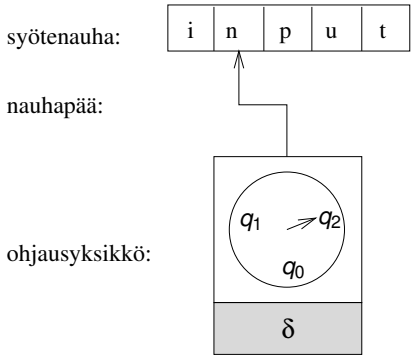

Täydennys syöteluvun arvon laskevilla operaatioilla (“luvun muuttaminen kymmenjärjestelmään”):

```
from sys import stdin

q=0
sgn=1                          # SEM: etumerkki
val=0                          # SEM: itseisarvo
for c in stdin.readline().strip("\n"):
    if q==0:
        if c=="+": q=1
        elif c=="-":
            sgn=-1                          # SEM
            q=1
        elif c in "01234567":
            val=int(c)                      # SEM
            q=2
        else: q=99
```

```
    elif q==1:
        if c in "01234567":
            val=int(c)                    # SEM
            q=2
        else: q=99
    elif q==2:
        if c in "01234567":
            val=8*val+int(c)              # SEM
            q=2
        else: q=99
if q == 2: print "Oktaaliluku", sgn*val    # SEM
else: print "Virheellinen oktaaliluku."
```


2.3 Äärellisen automaatin käsitteen formalisointi

Mekanistinen malli:



Äärellinen automaatti *M* koostuu äärellistilaisesta *ohjausyksiköstä*, jonka toimintaa säätelee automaatin *siirtymäfunktio* δ, sekä merkkipaikkoihin jaetusta *syötenauhasta* ja nämä yhdistävästä *nauhapäästä*, joka kullakin hetkellä osoittaa yhtä syötenauhan merkkiä.

Automaatin "toiminta":

Automaatti käynnistetään erityisessä *alkutilassa* q_0, siten että tarkasteltava syöte on kirjoitettuna syötenauhalle ja nauhapää osoittaa sen ensimmäistä merkkiä.

Yhdessä toiminta-askelessa automaatti lukee nauhapään kohdalla olevan syötemerkin, päättää ohjausyksikön tilan ja luetun merkin perusteella siirtymäfunktion mukaisesti ohjausyksikön uudesta tilasta, ja siirtää nauhapäätä yhden merkin eteenpäin.

Automaatti pysähtyy, kun viimeinen syötemerkki on käsitelty. Jos ohjausyksikön tila tällöin kuuluu erityiseen *(hyväksyvien) lopputilojen* joukkoon, automaatti *hyväksyy* syötteen, muuten *hylkää* sen.

Automaatin *tunnistama kieli* on sen hyväksymien merkkijonojen joukko.

Täsmällinen muotoilu:

Äärellinen automaatti on viisikko

$$M = (Q, \Sigma, \delta, q_0, F),$$

missä

- Q on automaatin *tilojen* äärellinen joukko;
- Σ on automaatin *syöteaakkosto*;
- $\delta : Q \times \Sigma \to Q$ on automaatin *siirtymäfunktio*;
- $q_0 \in Q$ on automaatin *alkutila*;
- $F \subseteq Q$ on automaatin *(hyväksyvien) lopputilojen* joukko.

Esimerkki.

Reaalilukuautomaatin formaali esitys:

$$M = (\{q_0, \ldots, q_7, \mathit{error}\}, \{\mathtt{0}, \mathtt{1}, \ldots, \mathtt{9}, \mathtt{.}, \mathtt{E}, \mathtt{e}, \mathtt{+}, \mathtt{-}\}, \delta, q_0, \{q_2, q_3, q_6\}),$$

missä δ on kuten aiemmin taulukossa; esim.

$$\delta(q_0, \mathtt{0}) = \delta(q_0, \mathtt{1}) = \cdots = \delta(q_0, \mathtt{9}) = q_1,$$
$$\delta(q_0, \mathtt{.}) = q_7, \quad \delta(q_0, \mathtt{E}) = \delta(q_0, \mathtt{e}) = \mathit{error},$$
$$\delta(q_1, \mathtt{.}) = q_2, \quad \delta(q_1, \mathtt{E}) = \delta(q_1, \mathtt{e}) = q_4,$$

jne.

Automaatin *tilanne* on pari $(q, w) \in Q \times \Sigma^*$; erityisesti automaatin *alkutilanne syötteellä* x on pari (q_0, x).

Intuitio: q on automaatin tila ja w on syötemerkkijonon jäljellä oleva, so. nauhapäästä oikealle sijaitseva osa.

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

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

jos on $w = aw'$ ($a \in \Sigma$) ja $q' = \delta(q, a)$. Tällöin sanotaan myös, että tilanne (q', w') on tilanteen (q, w) *välitön seuraaja*.

Intuitio: automaatti ollessaan tilassa q ja lukiessaan nauhalla olevan merkkijonon $w = aw'$ ensimmäisen merkin a siirtyy tilaan q' ja siirtää nauhapäätä yhden askelen eteenpäin, jolloin nauhalle jää merkkijono w'.

Jos automaatti M on yhteydestä selvä, relaatiota voidaan merkitä yksinkertaisesti

$$(q, w) \vdash (q', w').$$

Tilanne (q, w) *johtaa tilanteeseen* (q', w') t. tilanne (q', w') on tilanteen (q, w) *seuraaja*, merkitään

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

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

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

Erikoistapauksena $n = 0$ saadaan $(q, w) \underset{M}{\vdash^*} (q, w)$ millä tahansa tilanteella (q, w).

Jälleen, jos automaatti M on yhteydestä selvä, merkitään yksinkertaisesti

$$(q, w) \vdash^* (q', w').$$

Automaatti M *hyväksyy* merkkijonon $x \in \Sigma^*$, jos on voimassa

$$(q_0, x) \vdash_M^* (q_f, \varepsilon) \qquad \text{jollakin } q_f \in F;$$

muuten M *hylkää* x:n.

Toisin sanoen: automaatti hyväksyy x:n, jos sen alkutilanne syötteellä x johtaa, syötteen loppuessa, johonkin hyväksyvään lopputilanteeseen.

Automaatin M *tunnistama kieli* määritellään:

$$L(M) = \{x \in \Sigma^* \mid (q_0, x) \vdash_M^* (q_f, \varepsilon) \quad \text{jollakin } q_f \in F\}.$$

Esimerkki: merkkijonon "`0.25E2`" käsittely reaalilukuautomaatilla:

$$\begin{array}{lll} (q_0, \texttt{0.25E2}) & \vdash & (q_1, \texttt{.25E2}) \\ & \vdash & (q_2, \texttt{25E2}) \\ & \vdash & (q_3, \texttt{5E2}) \\ & \vdash & (q_3, \texttt{E2}) \\ & \vdash & (q_4, \texttt{2}) \\ & \vdash & (q_6, \varepsilon). \end{array}$$

Koska $q_6 \in F = \{q_2, q_3, q_6\}$, on siis $\texttt{0.25E2} \in L(M)$.

2.4 Äärellisten automaattien minimointi

Voidaan osoittaa, että jokaisella äärellisellä automaatilla on yksikäsitteinen ekvivalentti (so. saman kielen tunnistava) tilamäärältään minimaalinen automaatti.

Annetun äärellisen automaatin kanssa minimointi (ekvivalentin minimiautomaatin määrittäminen) on sekä käytännössä että teoreettiselta kannalta tärkeä tehtävä: siten voidaan esimerkiksi selvittää, tunnistavatko kaksi annettua automaattia saman kielen.

Tehtävä voidaan ratkaista seuraavassa esitettävällä tehokkaalla menetelmällä. Menetelmän perusideana on pyrkiä samaistamaan keskenään sellaiset syötteenä annetun automaatin tilat, joista lähtien automaatti toimii täsmälleen samoin kaikilla merkkijonoilla.

Olkoon

$$M = (Q, \Sigma, \delta, q_0, F)$$

äärellinen automaatti.

Laajennetaan M:n siirtymäfunktio yksittäisistä syötemerkeistä merkkijonoihin: jos $q \in Q$, $x \in \Sigma^*$, merkitään

$$\delta^*(q, x) = \text{ se } q' \in Q, \text{ jolla } (q, x) \vdash_M^* (q', \varepsilon).$$

M:n tilat q ja q' ovat *ekvivalentit*, merkitään

$$q \equiv q',$$

jos kaikilla $x \in \Sigma^*$ on

$$\delta^*(q, x) \in F \quad \text{jos ja vain jos} \quad \delta^*(q', x) \in F;$$

toisin sanoen, jos automaatti q:sta ja q':sta lähtien hyväksyy täsmälleen samat merkkijonot.

Lievempi ekvivalenssiehto: tilat q ja q' ovat *k-ekvivalentit*, merkitään

$$q \overset{k}{\equiv} q',$$

jos kaikilla $x \in \Sigma^*$, $|x| \leq k$, on

$$\delta^*(q,x) \in F \quad \text{jos ja vain jos} \quad \delta^*(q',x) \in F;$$

toisin sanoen, jos mikään enintään k:n pituinen merkkijono ei pysty erottamaan tiloja toisistaan.

Ilmeisesti on:

$$\begin{array}{lllll} \text{(i)} & q \overset{0}{\equiv} q' & \text{joss} & \text{sekä } q \text{ että } q' \text{ ovat lopputiloja} & \\ & & & \text{tai kumpikaan ei ole; ja} & (1) \\ \text{(ii)} & q \equiv q' & \text{joss} & q \overset{k}{\equiv} q' \text{ kaikilla } k = 0,1,2,\ldots & \end{array}$$

Esitettävä minimointialgoritmi perustuu syötteenä annetun automaatin tilojen k-ekvivalenssiluokkien hienontamiseen $(k+1)$-ekvivalenssiluokiksi kunnes saavutetaan täysi ekvivalenssi.

Algoritmi perustuu seuraavaan:

1. Kaksi (k-ekvivalenttia) tilaa $q_1 \overset{k+1}{\equiv} q_2$, jos ja vain jos $q_1 \overset{0}{\equiv} q_2$ ja $\delta(q_1,a) \overset{k}{\equiv} \delta(q_2,a)$ kaikilla $a \in \Sigma$.
2. Jos jollakin k:n arvolla pätee, että *kaikki* keskenään k-ekvivalentit tilat ovat myös $k+1$-ekvivalentteja, niin keskenään k-ekvivalentit tilat ovat keskenään ekvivalentteja.

Tod. Induktioaskel: oletetaan, että keskenään k-ekvivalentit tilat ovat p-ekvivalentteja jollakin p. Keskenään k-ekvivalentit tilat ovat siis $k+1$-ekvivalentteja, eli ym. perusteella niistä siirrytään kaikilla aakkosilla keskenään k-ekvivalentteihin (p-ekvivalentteihin) tiloihin; k-ekvivalentit tilat ovat siis $p+1$-ekvivalentteja. (Perustapaus esim. $p = 0$.)

Algoritmi MIN-FA

[Äärellisen automaatin minimointi]

Syöte: Äärellinen automaatti $M = (Q, \Sigma, \delta, q_0, F)$.

Tulos: M:n kanssa ekvivalentti äärellinen automaatti $\widehat{M}$, jossa on minimimäärä tiloja.

Menetelmä:

1. [Turhien tilojen poisto.] Poista M:stä kaikki tilat, joita ei voida saavuttaa tilasta q_0 millään syötemerkkijonolla.

2. [0-ekvivalenssi.] Osita M:n jäljelle jääneet tilat kahteen luokkaan: ei-lopputiloihin ja lopputiloihin.

3. [k-ekvivalenssi $\rightarrow$ $(k+1)$-ekvivalenssi.] Tarkasta, siirrytäänkö samaan ekvivalenssiluokkaan kuuluvista tiloista samoilla merkeillä aina samanluokkaisiin tiloihin. Jos kyllä, algoritmi päättyy ja minimiautomaatin $\widehat{M}$ tilat vastaavat M:n tilojen *luokkia*.

Muussa tapauksessa hienonna ositusta jakamalla kunkin äskeistä ehtoa rikkovan k-ekvivalenssiluokan tilat uusiin, pienempiin $(k+1)$-ekvivalenssiluokkiin sen mukaan, mihin luokkaan kustakin tilasta siirrytään milläkin aakkosella, ja toista kohta 3 uudella osituksella.

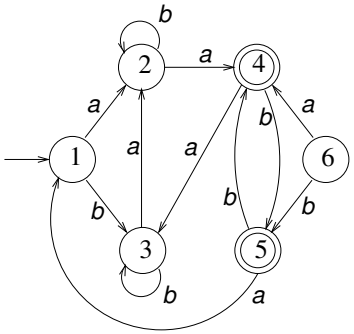
Lause 2.1 Algoritmi MIN-FA muodostaa annetun äärellisen automaatin M kanssa ekvivalentin äärellisen automaatin $\widehat{M}$, jossa on minimimäärä tiloja. Tämä automaatti on tilojen nimeämistä paitsi yksikäsitteinen.

- Algoritmin suoritus päättyy aina: kullakin askelen 3 suorituskerralla, viimeistä lukuunottamatta, vähintään yksi luokka ositetaan pienemmäksi.
- Askel 3 jakaa k-ekvivalenssiluokat $(k+1)$-ekvivalenssiluokiksi: kaksi tilaa ovat $(k+1)$-ekvivalentit, jos ja vain jos molemmat tai ei kumpikaan ovat hyväksyviä lopputiloja ja kummastakin siirrytään jokaisella aakkosella k-ekvivalentteihin tiloihin.
- Kun k-ekvivalenssiluokat ovat myös $k+1$-ekvivalenssiluokat, kunkin luokan tilat ovat k-ekvivalentteja kaikilla k eli ekvivalentteja (1.ii).
- Jokaisessa ekvivalenssiluokassa on ainakin yksi alkutilasta saavutettava tila, joten kaikki luokat ovat tarpeellisia.
- Yksikäsitteisyys todistettu opetusmonisteessa. □

Esimerkki. Tarkastellaan seuraavan automaatin minimointia:

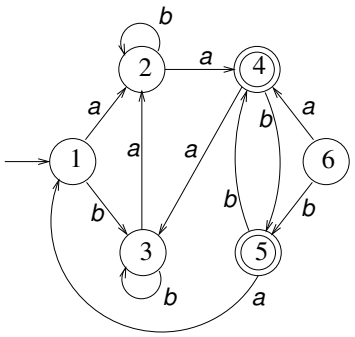
Vaiheessa 1 automaatista poistetaan tila 6, johon ei päästä millään merkkijonolla.

Vaiheessa 2 ositetaan automaatin tilat 1–5 ei-lopputiloihin (luokka I) ja lopputiloihin (luokka II), ja tarkastetaan siirtymien käyttäytyminen osituksen suhteen:



			a	b
I :	→	1	2, I	3, I
		2	4, II	2, I
		3	2, I	3, I
II :	←	4	3, I	5, II
	←	5	1, I	4, II

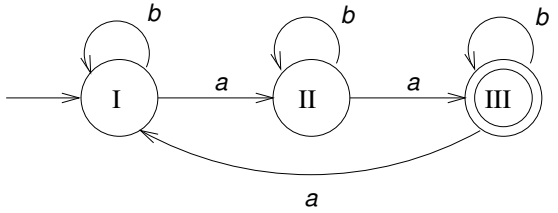
Luokassa I on nyt kahdentyyppisiä tiloja ({1,3} ja {2}), joten ositusta täytyy hienontaa ja tarkastaa siirtymät uuden osituksen suhteen:



			a	b
I :	→	1	2, II	3, I
		3	2, II	3, I
II :		2	4, III	2, II
III :	←	4	3, I	5, III
	←	5	1, I	4, III

Nyt kunkin luokan sisältämät tilat ovat keskenään samanlaisia, joten minimointialgoritmi päättyy.

Saadun minimiautomaatin tilakaavio on seuraava:

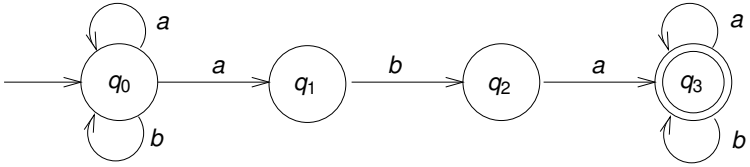


2.5 Epädeterministiset äärelliset automaatit

Epädeterministiset automaatit ovat muuten samanlaisia kuin deterministiset, mutta niiden siirtymäfunktio δ ei liitä automaatin vanhan tilan ja syötemerkin muodostamiin pareihin yksikäsitteistä uutta tilaa, vaan *joukon* mahdollisia seuraavia tiloja. Epädeterministinen automaatti hyväksyy syötteensä, jos *jokin* mahdollisten tilojen jono johtaa hyväksyvään lopputilaan.

Vaikka epädeterministisiä automaatteja ei voi sellaisinaan toteuttaa tietokoneohjelmina, ne ovat tärkeä päätösongelmien *kuvausformalismi*.

Esimerkki. Epädeterministinen automaatti, joka tutkii sisältääkö syötejono osajonoa *aba*:



Automaatti hyväksyy esim. syötejonon *aaba*, koska sen on mahdollista edetä seuraavasti:

$$(q_0, aaba) \vdash (q_0, aba) \vdash (q_1, ba) \vdash (q_2, a) \vdash (q_3, \varepsilon).$$

Automaatti voisi päätyä myös hylkäävään tilaan:

$$(q_0, aaba) \vdash (q_0, aba) \vdash (q_0, ba) \vdash (q_0, a) \vdash (q_0, \varepsilon),$$

mutta tällä ei ole merkitystä— voidaan ajatella, että automaatti osaa "ennustaa" ja valita aina parhaan mahdollisen vaihtoehdon.

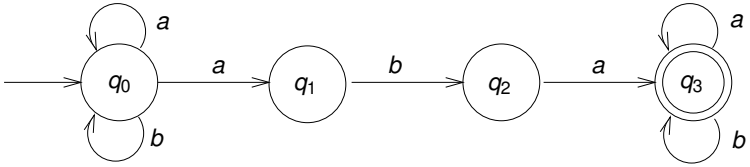
Määritelmä 2.2

Epädeterministinen äärellinen automaatti on viisikko

$$M = (Q, \Sigma, \delta, q_0, F),$$

missä

- Q on äärellinen *tilojen* joukko;
- Σ on *syöteaakkosto*;
- $\delta : Q \times \Sigma \to \boxed{\mathcal{P}(Q)}$ on automaatin joukkoarvoinen *siirtymäfunktio*;
- $q_0 \in Q$ on *alkutila*;
- $F \subseteq Q$ on *(hyväksyvien) lopputilojen* joukko.



Esimerkiksi *aba*-automaatin siirtymäfunktio:

		a	b
$\to$	q_0	$\{q_0, q_1\}$	$\{q_0\}$
	q_1	$\emptyset$	$\{q_2\}$
	q_2	$\{q_3\}$	$\emptyset$
$\leftarrow$	q_3	$\{q_3\}$	$\{q_3\}$

Taulukosta nähdään, että esimerkiksi $\delta(q_0, a) = \{q_0, q_1\}$ ja $\delta(q_1, a) = \emptyset$.

Epädeterministisen automaatin tilanne (q, w) voi *johtaa suoraan* tilanteeseen (q', w'), merkitään

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

jos on $w = aw'$ $(a \in \Sigma)$ ja $q' \in \delta(q, a)$. Sanotaan myös, että tilanne (q', w') on tilanteen (q, w) mahdollinen *välitön seuraaja*.

Useamman askelen mittaiset tilannejohdot, merkkijonojen hyväksyminen ja hylkääminen ym. käsitteet määritellään samoin sanoin kuin deterministisillä automaateilla. Koska yhden askelen johdon määritelmä kuitenkin nyt on toinen, niiden sisältö muovautuu hieman erilaiseksi.

Lause 2.2 Olkoon $A = L(M)$ jonkin epädeterministisen äärellisen automaatin M tunnistama kieli. Tällöin on olemassa myös deterministinen äärellinen automaatti $\widehat{M}$, jolla $A = L(\widehat{M})$.

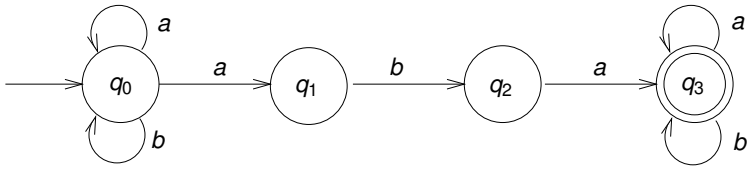
Todistus. Olkoon $A = L(M)$, $M = (Q, \Sigma, \delta, q_0, F)$. Todistuksen ideana on laatia deterministinen äärellinen automaatti $\widehat{M}$, joka simuloi M:n toimintaa kaikissa sen kullakin hetkellä mahdollisissa tiloissa *rinnakkain*.

Formaalisti: automaatin $\widehat{M}$ tilat vastaavat M:n tilojen *joukkoja*:

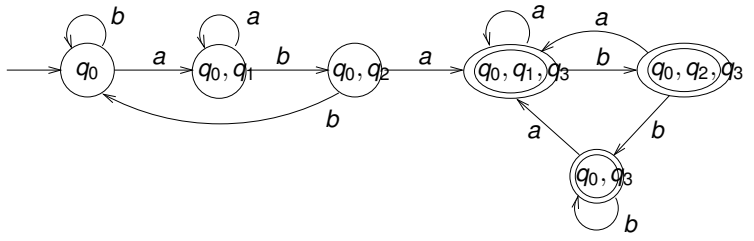
$$\widehat{M} = (\hat{Q}, \Sigma, \hat{\delta}, \hat{q}_0, \widehat{F}),$$

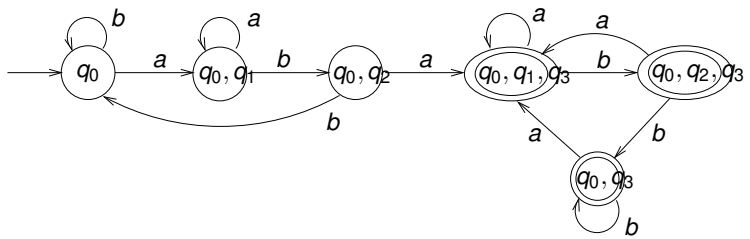
missä

$$\begin{aligned} \hat{Q} &= \mathcal{P}(Q) = \{S \mid S \subseteq Q\}, \\ \hat{q}_0 &= \{q_0\}, \\ \widehat{F} &= \{S \subseteq Q \mid S \text{ sisältää jonkin } q_f \in F\}, \\ \hat{\delta}(S, a) &= \bigcup_{q \in S} \delta(q, a). \end{aligned}$$

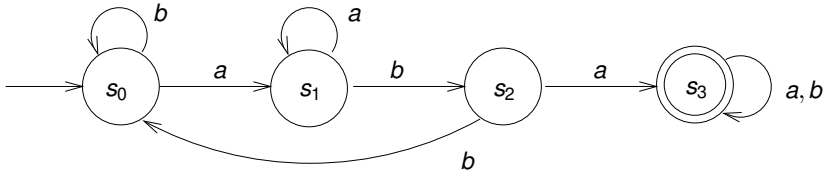


Esimerkiksi *aba*-automaattiin sovellettuna em. konstruktio tuottaa seuraavan deterministisen automaatin (vain alkutilasta saavutettavat tilat esitetty):





Minimoimalla ja nimeämällä tilat uudelleen tämä yksinkertaistuu muotoon:



[Todistus jatkuu.] Tarkastetaan, että automaatti $\widehat{M}$ todella on ekvivalentti M:n kanssa, so. että $L(\widehat{M}) = L(M)$.

Määritelmien mukaan on:

$$x \in L(M) \text{ joss } (q_0, x) \vdash_M^* (q_f, \varepsilon) \text{ jollakin } q_f \in F$$

ja

$$x \in L(\widehat{M}) \text{ joss } (\{q_0\}, x) \vdash_{\widehat{M}}^* (S, \varepsilon), \text{ missä } S \text{ sis. jonkin } q_f \in F.$$

Osoitetaan siis, että kaikilla $x \in \Sigma^*$ ja $q \in Q$ on:

$$(q_0, x) \vdash_M^* (q, \varepsilon) \text{ joss } (\{q_0\}, x) \vdash_{\widehat{M}}^* (S, \varepsilon) \text{ ja } q \in S. \qquad (2)$$

Väite (2):

$$(q_0, x) \vdash_M^* (q, \varepsilon) \text{ joss } (\{q_0\}, x) \vdash_{\widehat{M}}^* (S, \varepsilon) \text{ ja } q \in S.$$

Väitteen (2) todistus tehdään induktiolla merkkijonon x pituuden suhteen.

(i) Tapaus $|x| = 0$*:*

$(q_0, \varepsilon) \vdash_M^* (q, \varepsilon)$ joss $q = q_0$;

samoin $(\{q_0\}, \varepsilon) \vdash_{\widehat{M}}^* (S, \varepsilon)$ joss $S = \{q_0\}$.

Väite (2): $(q_0, x) \vdash_M^* (q, \varepsilon)$ joss $(\{q_0\}, x) \vdash_{\widehat{M}}^* (S, \varepsilon)$ ja $q \in S$.

(ii) Induktioaskel:

Olkoon $x = ya$; oletetaan, että väite (2) pätee y:lle. Tällöin:

$(q_0, x) = (q_0, ya) \vdash_M^* (q, \varepsilon)$ joss

$\exists q' \in Q$ s.e. $(q_0, ya) \vdash_M^* (q', a)$ ja $(q', a) \vdash_M (q, \varepsilon)$ joss

$\exists q' \in Q$ s.e. $(q_0, y) \vdash_M^* (q', \varepsilon)$ ja $(q', a) \vdash_M (q, \varepsilon)$ joss (ind.ol.)

$\exists q' \in Q$ s.e. $(\{q_0\}, y) \vdash_{\widehat{M}}^* (S', \varepsilon)$ ja $q' \in S'$ ja $q \in \delta(q', a)$ joss

$(\{q_0\}, y) \vdash_{\widehat{M}}^* (S', \varepsilon)$ ja $\exists q' \in S'$ s.e. $q \in \delta(q', a)$ joss

$(\{q_0\}, y) \vdash_{\widehat{M}}^* (S', \varepsilon)$ ja $q \in \bigcup_{q' \in S'} \delta(q', a) = \hat{\delta}(S', a)$ joss

$(\{q_0\}, ya) \vdash_{\widehat{M}}^* (S', a)$ ja $q \in \hat{\delta}(S', a) = S$ joss

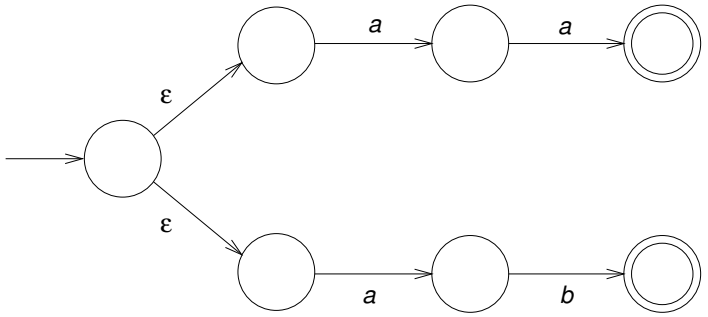
$(\{q_0\}, ya) \vdash_{\widehat{M}}^* (S', a)$ ja $(S', a) \vdash_{\widehat{M}} (S, \varepsilon)$ ja $q \in S$ joss

$(\{q_0\}, x) = (\{q_0\}, ya) \vdash_{\widehat{M}}^* (S, \varepsilon)$ ja $q \in S$. □

ε-automaatit

Jatkossa tarvitaan vielä yksi äärellisten automaattien mallin laajennus: epädeterministinen äärellinen automaatti, jossa sallitaan ε-*siirtymät*. Tällaisella siirtymällä automaatti tekee epädeterministisen valinnan eri jatkovaihtoehtojen välillä lukematta yhtään syötemerkkiä.

Esimerkiksi kieli $\{aa, ab\}$ voitaisiin tunnistaa seuraavalla ε-automaatilla:



Formaalisti: ε-automaatti on viisikko

$$M = (Q, \Sigma, \delta, q_0, F),$$

missä siirtymäfunktio δ on kuvaus

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q).$$

Muut määritelmät ovat kuten tavallisilla epädeterministisillä äärellisillä automaateilla, paitsi suoran tilannejohdon määritelmä: ε-automaattien tapauksessa relaatio

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

on voimassa, jos on

(i) $w = aw'$ ($a \in \Sigma$) ja $q' \in \delta(q, a)$; tai
(ii) $w = w'$ ja $q' \in \delta(q, \varepsilon)$.

Lemma 2.4

Olkoon $A = L(M)$ jollakin ε-automaatilla M. Tällöin on olemassa myös ε-siirtymätön epädeterministinen automaatti $\widehat{M}$, jolla $A = L(\widehat{M})$.

Todistus. Olkoon $M = (Q, \Sigma, \delta, q_0, F)$ jokin ε-automaatti. Automaatti $\widehat{M}$ toimii muuten aivan samoin kuin M, mutta se "harppaa" ε-siirtymien yli suorittamalla kustakin tilasta lähtien vain ne "aidot" siirtymät, jotka ovat siitä käsin jotakin ε-siirtymäjonoa pitkin saavutettavissa.

Formaalisti määritellään annetun tilan $q \in Q$ ε-*sulkeuma* $\varepsilon^*(q)$ automaatissa M kaavalla

$$\varepsilon^*(q) = \{q' \in Q \mid (q, \varepsilon) \underset{M}{\vdash}^* (q', \varepsilon)\},$$

so. joukkoon $\varepsilon^*(q)$ kuuluvat kaikki ne automaatin M tilat, jotka ovat saavutettavissa tilasta q pelkillä ε-siirtymillä.

Automaatin $\widehat{M}$ siirtymäsäännöt voidaan nyt kuvata seuraavasti:

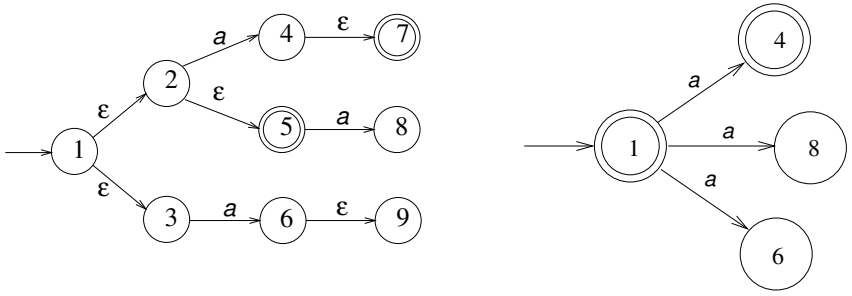
$$\widehat{M} = (Q, \Sigma, \hat{\delta}, q_0, \widehat{F}),$$

missä

$$\hat{\delta}(q, a) = \bigcup_{q' \in \varepsilon^*(q)} \delta(q', a);$$

$$\widehat{F} = \{q \in Q \mid \varepsilon^*(q) \cap F \neq \emptyset\}.$$

Poistamalla edellisen konstruktion mukaisesti ε-siirtymät ε-automaatista saadaan tavallinen epädeterministinen automaatti, esim.:



2.6 SÄÄNNÖLLISET LAUSEKKEET

Automaattimalleista poikkeava tapa kuvata yksinkertaisia kieliä.

Olkoot A ja B aakkoston Σ kieliä. Perusoperaatioita:

- *Yhdiste:* $A \cup B = \{x \in \Sigma^* \mid x \in A \text{ tai } x \in B\}$;
- *Katenaatio:* $AB = \{xy \in \Sigma^* \mid x \in A,\ y \in B\}$;
- *Potenssit:*

$$\begin{cases} A^0 &= \{\varepsilon\}, \\ A^k &= AA^{k-1} = \{x_1 \dots x_k \mid x_i \in A \quad \forall i = 1, \dots, k\} \quad (k \geq 1); \end{cases}$$

- *Sulkeuma t. "Kleenen tähti":*

$$\begin{aligned} A^* &= \bigcup_{k \geq 0} A^k \\ &= \{x_1 \dots x_k \mid k \geq 0,\ x_i \in A \quad \forall i = 1, \dots, k\}. \end{aligned}$$

Määritelmä 2.3

Aakkoston Σ *säännölliset lausekkeet* määritellään induktiivisesti säännöillä:

1. $\emptyset$ ja ε ovat Σ:n säännöllisiä lausekkeita;
2. a on Σ:n säännöllinen lauseke kaikilla $a \in \Sigma$;
3. jos r ja s ovat Σ:n säännöllisiä lausekkeita, niin $(r \cup s)$, (rs) ja r^* ovat Σ:n säännöllisiä lausekkeita;
4. muita Σ:n säännöllisiä lausekkeita ei ole.

Kukin Σ:n säännöllinen lauseke r *kuvaa* kielen $L(r)$, joka määritellään:

- $L(\emptyset) = \emptyset$;
- $L(\varepsilon) = \{\varepsilon\}$;
- $L(a) = \{a\}$ kaikilla $a \in \Sigma$;
- $L((r \cup s)) = L(r) \cup L(s)$;
- $L((rs)) = L(r)L(s)$;
- $L(r^*) = (L(r))^*$.

Aakkoston $\{a, b\}$ säännöllisiä lausekkeita:

$$r_1 = ((ab)b), \quad r_2 = (ab)^*,$$

$$r_3 = (ab^*), \quad r_4 = (a(b \cup (bb)))^*.$$

Lausekkeiden kuvaamat kielet:

$$\begin{aligned} L(r_1) &= (\{a\}\{b\})\{b\} = \{ab\}\{b\} = \{abb\}; \\ L(r_2) &= \{ab\}^* = \{\varepsilon, ab, abab, ababab, \dots\} \\ &= \{(ab)^i \mid i \geq 0\}; \\ L(r_3) &= \{a\}(\{b\})^* = \{a, ab, abb, abbb, \dots\} \\ &= \{ab^i \mid i \geq 0\}; \\ L(r_4) &= (\{a\}\{b, bb\})^* = \{ab, abb\}^* \\ &= \{\varepsilon, ab, abb, abab, ababb, \dots\} \\ &= \{x \in \{a, b\}^* \mid \text{ kutakin } a\text{-kirjainta } x\text{:ssä} \\ &\quad \text{seuraa 1 tai 2 } b\text{-kirjainta } \}. \end{aligned}$$

Sulkumerkkien vähentämissääntöjä:

Operaattoreiden prioriteetti:

$$* \quad \succ \quad \cdot \quad \succ \quad \cup$$

Yhdiste- ja tulo-operaatioiden assosiatiivisuus:

$$\begin{aligned} L(((r \cup s) \cup t)) &= L((r \cup (s \cup t))) \\ L(((rs)t)) &= L((r(st))) \end{aligned}$$

$\Rightarrow$ peräkkäisiä yhdisteitä ja tuloja ei tarvitse suluttaa.

Käytetään tavallisia kirjasimia, mikäli sekaannuksen vaaraa merkkijonoihin ei ole.

Yksinkertaisemmin siis:

$$r_1 = abb, \quad r_2 = (ab)^*, \quad r_3 = ab^*, \quad r_4 = (a(b \cup bb))^*.$$

Esimerkki: C-kielen etumerkittömät reaaliluvut

$$number = (dd^*.d^* \cup .dd^*)(e(+ \cup - \cup \varepsilon)dd^* \cup \varepsilon) \cup (dd^* e(+ \cup - \cup \varepsilon)dd^*),$$

missä d on lyhennemerkintä lausekkeelle

$$d = (\mathtt{0} \cup \mathtt{1} \cup \mathtt{2} \cup \mathtt{3} \cup \mathtt{4} \cup \mathtt{5} \cup \mathtt{6} \cup \mathtt{7} \cup \mathtt{8} \cup \mathtt{9})$$

ja e on lyhennemerkintä lausekkeelle

$$e = (\mathtt{E} \cup \mathtt{e}).$$

Usein merkitään myös lyhyesti $rr^* \equiv r^+$. Esim.:

$$(d^+.d^* \cup .d^+)(e(+ \cup - \cup \varepsilon)d^+ \cup \varepsilon) \cup (d^+ e(+ \cup - \cup \varepsilon)d^+).$$

Määritelmä 2.4

Kieli on *säännöllinen*, jos se voidaan kuvata säännöllisellä lausekkeella.

Säännöllisten lausekkeiden sieventäminen

Säännöllisillä kielillä on yleensä useita vaihtoehtoisia kuvauksia, esim.:

$$\begin{aligned} \Sigma^* &= L((a \cup b)^*) \\ &= L((a^* b^*)^*) \\ &= L(a^* b^* \cup (a \cup b)^* ba(a \cup b)^*). \end{aligned}$$

Määritelmä. Säännölliset lausekkeet r ja s ovat *ekvivalentit*, merk. $r = s$, jos $L(r) = L(s)$.

Lausekkeen sieventäminen = “yksinkertaisimman” ekvivalentin lausekkeen määrittäminen.

Säännöllisten lausekkeiden ekvivalenssitestaus on epätriviaali, mutta periaatteessa mekaanisesti ratkeava ongelma.

Sievennyssääntöjä:

$$
\begin{array}{rcl}
r \cup (s \cup t) & = & (r \cup s) \cup t \\
r(st) & = & (rs)t \\
r \cup s & = & s \cup r \\
r(s \cup t) & = & rs \cup rt \\
(r \cup s)t & = & rt \cup st \\
r \cup r & = & r
\end{array}
\qquad
\begin{array}{rcl}
r \cup \emptyset & = & r \\
\varepsilon r & = & r \\
\emptyset r & = & \emptyset \\
r^* & = & \varepsilon \cup r^* r \\
r^* & = & (\varepsilon \cup r)^*
\end{array}
$$

Mikä tahansa säännöllisten lausekkeiden tosi ekvivalenssi voidaan johtaa näistä laskulaeista, kun lisätään päättelysääntö:

jos $r = rs \cup t$, niin $r = ts^*$, edellyttäen että $\varepsilon \notin L(s)$.

Kahden lausekkeen ekvivalenssin toteamiseksi kannattaa usein päätellä erikseen kummankin kuvaaman kielen sisältyminen toiseen.

Merkitään lyhyesti: $r \subseteq s$, jos $L(r) \subseteq L(s)$.
Tällöin siis $r = s$ joss $r \subseteq s$ ja $s \subseteq r$.

Esimerkki: todetaan, että $(a^*b^*)^* = (a \cup b)^*$.

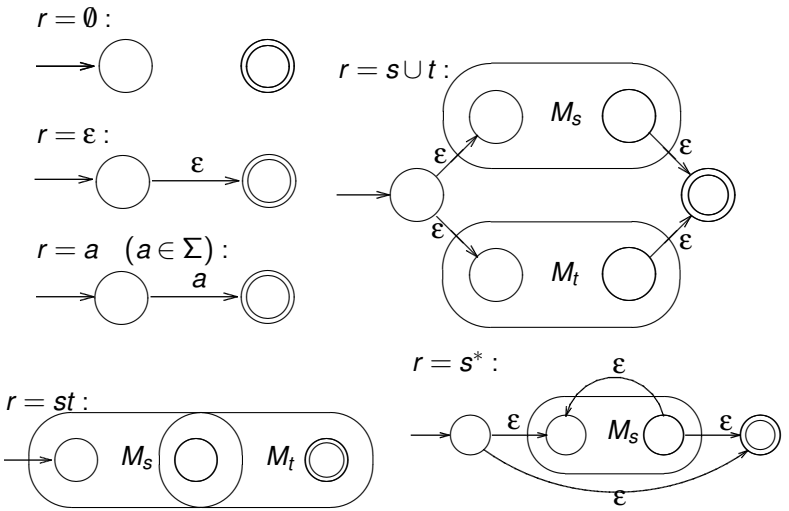
1. Selvästi $(a^*b^*)^* \subseteq (a \cup b)^*$, koska $(a \cup b)^*$ kuvaa *kaikkia* aakkoston $\{a, b\}$ merkkijonoja.
2. Koska selvästi $(a \cup b) \subseteq a^*b^*$, niin myös $(a \cup b)^* \subseteq (a^*b^*)^*$.

2.7 ÄÄRELLISET AUTOMAATIT JA SÄÄNNÖLLISET KIELET

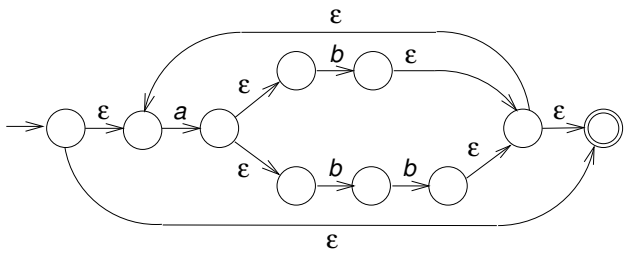
Lause 2.3 Jokainen säännöllinen kieli voidaan tunnistaa äärellisellä automaatilla.

Todistus. Seuraavan kalvon induktiivisen konstruktion avulla voidaan mielivaltaisen säännöllisen lausekkeen r rakennetta seuraten muodostaa ε-automaatti M_r, jolla $L(M_r) = L(r)$. Tästä automaatista voidaan poistaa ε-siirtymät Lemman 2.4 mukaisesti, ja tarvittaessa voidaan syntyvä epädeterministinen automaatti determinisoida Lauseen 2.2 konstruktiolla.

Esitettävästä konstruktiosta on syytä huomata, että muodostettaviin ε-automaatteihin tulee aina yksikäsitteiset alku- ja lopputila, eikä minkään osa-automaatin lopputilasta lähde eikä alkutilaan tule yhtään ko. osa-automaatin sisäistä siirtymää. □

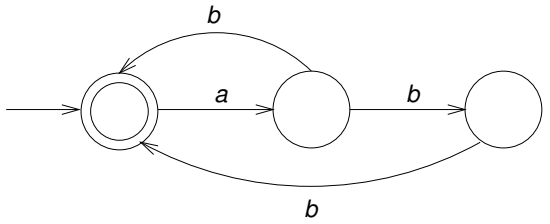


Esimerkiksi lausekkeesta $r = (a(b \cup bb))^*$ saadaan näiden sääntöjen mukaan seuraava ε-automaatti:



Automaatti on selvästi hyvin redundantti. Käsin automaatteja suunniteltaessa ne kannattaakin usein muodostaa suoraan.

Esim. lausekkeen $r = (a(b \cup bb))^*$ perusteella on helppo muodostaa seuraava yksinkertainen epädeterministinen tunnistaja-automaatti:



Lause 2.4

Jokainen äärellisellä automaatilla tunnistettava kieli on säännöllinen.

Todistus. Tarvitaan vielä yksi äärellisten automaattien laajennus: *lausekeautomaatissa* voidaan siirtymien ehtoina käyttää mielivaltaisia säännöllisiä lausekkeita.

Formalisointi: Merk. RE_Σ = aakkoston Σ säännöllisten lausekkeiden joukko. *Lausekeautomaatti* on viisikko

$$M = (Q, \Sigma, \delta, q_0, F),$$

missä siirtymäfunktio δ on äärellinen kuvaus

$$\delta : Q \times \mathrm{RE}_\Sigma \to \mathcal{P}(Q)$$

(so. $\delta(q, r) \neq \emptyset$ vain äärellisen monella parilla $(q, r) \in Q \times \mathrm{RE}_\Sigma$).

Yhden askelen tilannejohto määritellään:

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

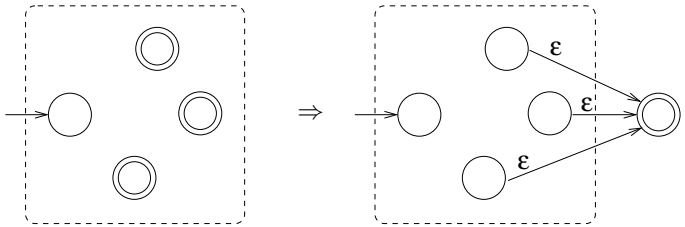
jos on $q' \in \delta(q, r)$ jollakin sellaisella $r \in \mathrm{RE}_\Sigma$, että $w = zw'$, $z \in L(r)$. Muut määritelmät samat kuin aiemmin.

Todistetaan: jokainen lausekeautomaatilla tunnistettava kieli on säännöllinen.

Olkoon M jokin lausekeautomaatti. Säännöllinen lauseke, joka kuvaa M:n tunnistaman kielen, muodostetaan kahdessa vaiheessa:

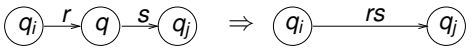
1. Tiivistetään M ekvivalentiksi enintään 2-tilaiseksi lausekeautomaatiksi seuraavilla muunnoksilla:

(i) Jos M:llä on useita lopputiloja, yhdistetään ne seuraavasti.

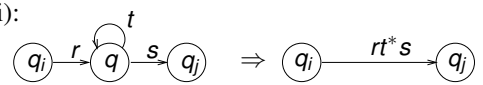


(ii) Poistetaan M:n muut kuin alku- ja lopputila yksi kerrallaan seuraavasti. Olk. q jokin M:n tila, joka ei ole alku- eikä lopputila; tarkastellaan kaikkia "reittejä", jotka M:ssä kulkevat q:n kautta. Olk. q_i ja q_j q:n välittömät edeltäjä- ja seuraajatila jollakin tällaisella reitillä. Poistetaan q reitiltä $q_i \longrightarrow q_j$ oheisen kuvan (i) muunnoksella, jos tilasta q ei ole siirtymää itseensä, ja kuvan (ii) muunnoksella, jos tilasta q on siirtymä itseensä:

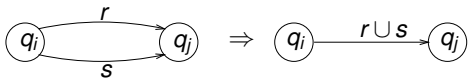
(i):



(ii):

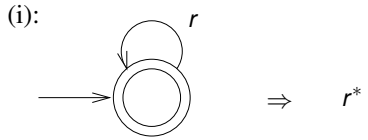


Samalla yhdistetään rinnakkaiset siirtymät seuraavasti:

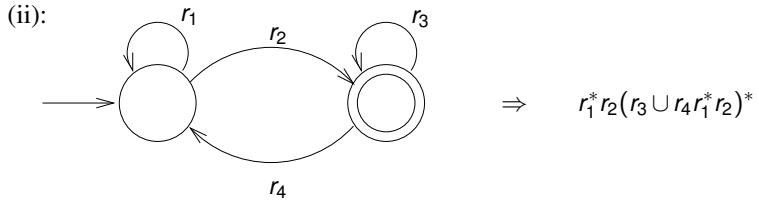


2. Tiivistyksen päättyessä jäljellä olevaa 2-tilaista automaattia vastaava säännöllinen lauseke muodostetaan seuraavan kuvan esittämällä tavalla:

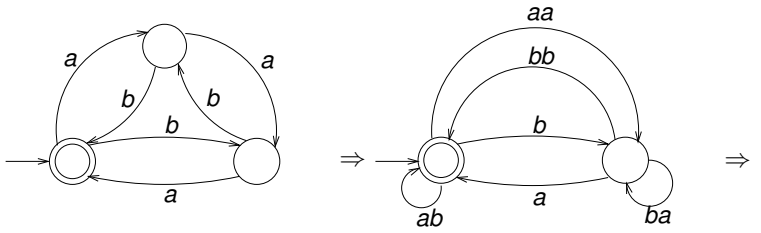
(i):

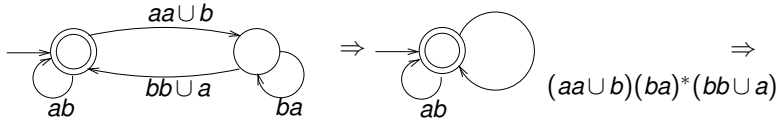


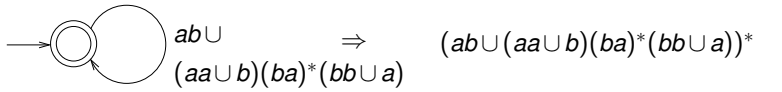
(ii):



Esimerkki:







2.8 Säännöllisten kielten rajoituksista

Kardinaliteettisyistä on oltava olemassa (paljon) ei-säännöllisiä kieliä: kieliä on ylinumeroituva määrä, säännöllisiä lausekkeita vain numeroituvasti.

Voidaanko löytää konkreettinen, *mielenkiintoinen* esimerkki kielestä, joka ei olisi säännöllinen? Helposti.

Säännöllisten kielten perusrajoitus: äärellisillä automaateilla on vain rajallinen "muisti". Siten ne eivät pysty ratkaisemaan ongelmia, joissa vaaditaan mielivaltaisen suurten lukujen tarkkaa muistamista.

Esimerkki: sulkulausekekieli

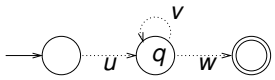
$$L_{\text{match}} = \{(^k)^k \mid k \geq 0\}.$$

Formalisointi: "pumppauslemma".

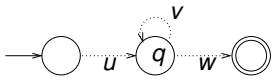
Lemma 2.6 (Pumppauslemma)

Olkoon A säännöllinen kieli. Tällöin on olemassa sellainen $n \geq 1$, että mikä tahansa $x \in A$, $|x| \geq n$, voidaan jakaa osiin $x = uvw$ siten, että $|uv| \leq n$, $|v| \geq 1$, ja $uv^i w \in A$ kaikilla $i = 0, 1, 2, \ldots$

Todistus. Olkoon M jokin A:n tunnistava deterministinen äärellinen automaatti, ja olkoon n M:n tilojen määrä. Tarkastellaan M:n läpikäymiä tiloja syötteellä $x \in A$, $|x| \geq n$. Koska M jokaisella x:n merkillä siirtyy tilasta toiseen, sen täytyy kulkea jonkin tilan kautta (ainakin) kaksi kertaa — itse asiassa jo x:n n:n ensimmäisen merkin aikana. Olkoon q ensimmäinen toistettu tila.



Olkoon u M:n käsittelemä x:n alkuosa sen tullessa ensimmäisen kerran tilaan q, v se osa x:stä jonka M käsittelee ennen ensimmäistä paluutaan q:hun, ja w loput x:stä. Tällöin on $|uv| \leq n$, $|v| \geq 1$, ja $uv^i w \in A$ kaikilla $i = 0, 1, 2, \ldots$ □



Esimerkki. Tarkastellaan em. sulkulausekekieltä (merk. '(' = a, ')' = b):

$$L = L_{\text{match}} = \{a^k b^k \mid k \geq 0\}.$$

Oletetaan, että L olisi säännöllinen. Tällöin pitäisi pumppauslemman mukaan olla jokin $n \geq 1$, jota pitempiä L:n merkkijonoja voidaan pumpata. Valitaan $x = a^n b^n$, jolloin $|x| = 2n > n$. Lemman mukaan x voidaan jakaa pumpattavaksi osiin $x = uvw$, $|uv| \leq n$, $|v| \geq 1$; siis on oltava

$$u = a^i,\ v = a^j,\ w = a^{n-(i+j)} b^n, \quad i \leq n-1,\ j \geq 1.$$

Mutta esimerkiksi "0-kertaisesti" pumpattaessa:

$$uv^0 w = a^i a^{n-(i+j)} b^n = a^{n-j} b^n \notin L.$$

Siten L ei voi olla säännöllinen.

3. KIELIOPIT JA MERKKIJONOJEN TUOTTAMINEN

Kielioppi = muunnossysteemi merkkijonojen (kielen "sanojen") tuottamiseen tietystä lähtöjonosta alkaen, osajonoja toistuvasti annettujen sääntöjen mukaan uudelleenkirjoittamalla.

Kielioppi on *yhteydetön*, jos kussakin uudelleenkirjoitusaskelessa korvataan yksi erityinen muuttuja- t. *välikesymboli* jollakin siihen liitetyllä korvausjonolla, ja korvaus voidaan aina tehdä symbolia ympäröivän merkkijonon rakenteesta riippumatta.

Sovelluksia: rakenteisten tekstien kuvaaminen (esim. ohjelmointikielten BNF-syntaksikuvaukset, XML:n DTD/Schema-määrittelyt), yleisemmin rakenteisten "olioiden" kuvaaminen (esim. syntaktinen hahmontunnistus).

Yhteydettömillä kieliopeilla voidaan kuvata (tuottaa) myös ei-säännöllisiä kieliä.

Esimerkki: yhteydetön kielioppi kielelle L_{match} (lähtösymboli S):

(i) $S \to \varepsilon$,
(ii) $S \to (S)$.

Esimerkiksi merkkijonon $((()))$ tuottaminen:

$$S \Rightarrow (S) \Rightarrow ((S)) \Rightarrow (((S))) \Rightarrow (((\varepsilon))) = ((())).$$

Toinen esimerkki: kielioppi C-tyyppisen ohjelmointikielen aritmeettisille lausekkeille (yksinkertaistettu).

$$\begin{array}{lcl|l} E & \to & T & E+T \\ T & \to & F & T*F \\ F & \to & a & (E). \end{array}$$

Esimerkiksi lausekkeen $(a+a)*a$ tuottaminen:

$$\begin{array}{lllllll} \underline{E} & \Rightarrow & \underline{T} & \Rightarrow & \underline{T}*F & \Rightarrow & \underline{F}*F \\ & \Rightarrow & (\underline{E})*F & \Rightarrow & (\underline{E}+T)*F & \Rightarrow & (\underline{T}+T)*F \\ & \Rightarrow & (\underline{F}+T)*F & \Rightarrow & (a+\underline{T})*F & \Rightarrow & (a+\underline{F})*F \\ & \Rightarrow & (a+a)*\underline{F} & \Rightarrow & (a+a)*a. & & \end{array}$$

Määritelmä 3.1

Yhteydetön kielioppi on nelikko

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

missä

- V on kieliopin aakkosto;
- $\Sigma \subseteq V$ on kieliopin *päätemerkkien* joukko; sen komplementti $N = V - \Sigma$ on kieliopin *välikemerkkien* t. *-symbolien* joukko;
- $P \subseteq N \times V^*$ on kieliopin *sääntöjen* t. *produktioiden* joukko;
- $S \in N$ on kieliopin *lähtösymboli*.

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

Merkkijono $\gamma \in V^*$ *tuottaa* t. *johtaa suoraan* merkkijonon $\gamma' \in V^*$ kieliopissa G, merkitään

$$\gamma \underset{G}{\Rightarrow} \gamma'$$

jos voidaan kirjoittaa $\gamma = \alpha A \beta$, $\gamma' = \alpha\omega\beta$ ($\alpha, \beta, \omega \in V^*$, $A \in N$), ja kieliopissa G on produktio $A \to \omega$.

Jos kielioppi G on yhteydestä selvä, voidaan merkitä $\gamma \Rightarrow \gamma'$.

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

$$\gamma = \gamma_0 \underset{G}{\Rightarrow} \gamma_1 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} \gamma_n = \gamma'.$$

Erikoistapauksena $n = 0$ saadaan $\gamma \underset{G}{\Rightarrow}^* \gamma$ millä tahansa $\gamma \in V^*$.

Jälleen, jos G on yhteydestä selvä, voidaan merkitä $\gamma \Rightarrow^* \gamma'$.

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* koostuu G:n lauseista:

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

Formaali kieli $L \subseteq \Sigma^*$ on *yhteydetön*, jos se voidaan tuottaa jollakin yhteydettömällä kieliopilla.

Esimerkiksi tasapainoisten sulkujonojen muodostaman kielen $L_{\text{match}} = \{(^k)^k \mid k \geq 0\}$ tuottaa kielioppi

$$G_{\text{match}} = (\{S, (,)\}, \{(,)\}, \{S \to \varepsilon, S \to (S)\}, S).$$

Yksinkertaisten aritmeettisten lausekkeiden muodostaman kielen L_{expr} tuottaa kielioppi

$$G_{\text{expr}} = (V, \Sigma, P, E),$$

missä

$$\begin{aligned} V &= \{E, T, F, a, +, *, (,)\}, \\ \Sigma &= \{a, +, *, (,)\}, \\ P &= \{E \to T,\ E \to E + T,\ T \to F,\ T \to T * F, \\ &\quad F \to a,\ F \to (E)\}. \end{aligned}$$

Toinen kielioppi kielen L_{expr} tuottamiseen on

$$G'_{\text{expr}} = (V, \Sigma, P, E),$$

missä

$$\begin{aligned} V &= \{E, a, +, *, (,)\}, \\ \Sigma &= \{a, +, *, (,)\}, \\ P &= \{E \to E + E,\ E \to E * E,\ E \to a,\ E \to (E)\}. \end{aligned}$$

Huom: Vaikka kielioppi G'_{expr} näyttää yksinkertaisemmalta kuin kielioppi G_{expr}, sen ongelmana on ns. rakenteellinen moniselitteisyys, mikä on monesti ei-toivottu ominaisuus.

Vakiintuneita merkintätapoja

Välikesymboleita: $A, B, C, \ldots, S, T$.

Päätemerkkejä: kirjaimet $a, b, c, \ldots, s, t$;
numerot $0, 1, \ldots, 9$;
erikoismerkit; lihavoidut tai alleviivatut varatut sanat (**if**, **for**, **end**, ...).

Mielivaltaisia merkkejä (kun välikkeitä ja päätteitä ei erotella): X, Y, Z.

Päätemerkkijonoja: u, v, w, x, y, z.

Sekamerkkijonoja: $\alpha, \beta, \gamma, \ldots, \omega$.

Produktiot, joilla on yhteinen vasen puoli A, voidaan kirjoittaa yhteen: joukon

$$A \to \omega_1,\ A \to \omega_2,\ \ldots A \to \omega_k$$

sijaan kirjoitetaan

$$A \to \omega_1 \mid \omega_2 \mid \ldots \mid \omega_k.$$

Kielioppi esitetään usein pelkkänä sääntöjoukkona:

$$\begin{array}{lcl|c|l} A_1 & \to & \omega_{11} & \ldots & \omega_{1k_1} \\ A_2 & \to & \omega_{21} & \ldots & \omega_{2k_2} \\ \vdots & & & & \\ A_m & \to & \omega_{m1} & \ldots & \omega_{mk_m}. \end{array}$$

Tällöin päätellään välikesymbolit edellisten merkintäsopimusten mukaan tai siitä, että ne esiintyvät sääntöjen vasempina puolina; muut esiintyvät merkit ovat päätemerkkejä. *Lähtösymboli* on tällöin *ensimmäisen säännön vasempana puolena* esiintyvä välike; tässä siis A_1.

Eräitä konstruktioita

Olkoon $L(T)$ välikkeestä T johdettavissa olevien päätejonojen joukko. Olkoon annettu produktiokokoelma P, jossa ei esiinny välikettä A, ja jolla B:stä voidaan johtaa $L(B)$ ja vastaavasti C:stä $L(C)$.

Lisäämällä P:hen jokin seuraavista produktioista saadaan uusia kieliä:

produktio	kieli
$A \to B \mid C$	yhdiste $L(A) = L(B) \cup L(C)$
$A \to BC$	katenaatio $L(A) = L(B)L(C)$, ja
$A \to AB \mid \varepsilon$ (vasen rekursio) tai $A \to BA \mid \varepsilon$ (oikea rekursio)	Kleenen sulkeuma $L(A) = L(B)^*$

Välikkeiden *keskeisupotus* on yhteydettömille kieliopeille ominainen konstruktio, joka tekee usein (muttei aina) kielestä epäsäännöllisen: lisäämällä produktio

$A \to BAC \mid \varepsilon$ saadaan

$$L(A) = \bigcup_{i=0}^{\infty} L(B)^i L(C)^i.$$

3.2 Säännölliset kielet ja yhteydettömät kieliopit

Yhteydettömillä kieliopeilla voidaan siis kuvata joitakin ei-säännöllisiä kieliä (esimerkiksi kielet L_{match} ja L_{expr}). Osoitetaan, että myös kaikki säännölliset kielet voidaan kuvata yhteydettömillä kieliopeilla. Yhteydettömät kielet ovat siten säännollisten kielten aito yliluokka.

Yhteydetön kielioppi on *oikealle lineaarinen*, jos sen kaikki produktiot ovat muotoa $A \to aB$ tai $A \to \varepsilon$, ja *vasemmalle lineaarinen*, jos sen kaikki produktiot ovat muotoa $A \to Ba$ tai $A \to \varepsilon$.

Osoittautuu, että sekä vasemmalle että oikealle lineaarisilla kieliopeilla voidaan tuottaa täsmälleen säännölliset kielet, minkä takia näitä kielioppeja nimitetään myös yhteisesti *säännöllisiksi*. Todistetaan tässä väite vain oikealle lineaarisille kieliopeille.

Lause 3.1

Jokainen säännöllinen kieli voidaan tuottaa oikealle lineaarisella kieliopilla.

Todistus. Olkoon L aakkoston Σ säännöllinen kieli, ja olkoon $M = (Q, \Sigma, \delta, q_0, F)$ sen tunnistava (deterministinen tai epädeterministinen) äärellinen automaatti. Muodostetaan kielioppi G_M, jolla on $L(G_M) = L(M) = L$.

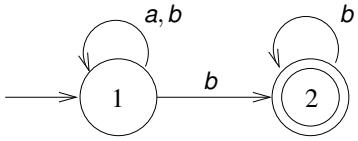
Kieliopin G_M pääteaakkosto on sama kuin M:n syöteaakkosto Σ, ja sen välikeaakkostoon otetaan yksi välike A_q kutakin M:n tilaa q kohden. Kieliopin lähtösymboli on A_{q_0}, ja sen produktiot vastaavat M:n siirtymiä:

(i) kutakin M:n lopputilaa $q \in F$ kohden kielioppiin otetaan produktio $A_q \to \varepsilon$;

(ii) kutakin M:n siirtymää $q \xrightarrow{a} q'$ (so. $q' \in \delta(q, a)$) kohden kielioppiin otetaan produktio $A_q \to aA_{q'}$.

Esimerkki.

Automaatti:



Vastaava kielioppi:

$$\begin{aligned} A_1 &\to aA_1 \mid bA_1 \mid bA_2 \\ A_2 &\to \varepsilon \mid bA_2. \end{aligned}$$

Konstruktion oikeellisuuden tarkastamiseksi merkitään välikkeestä A_q tuotettavien päätejonojen joukkoa

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

Induktiolla merkkijonon x pituuden suhteen voidaan osoittaa, että kaikilla q on

$$x \in L(A_q) \text{ joss } (q, x) \underset{M}{\vdash^*} (q_f, \varepsilon) \text{ jollakin } q_f \in F.$$

Erityisesti on siis

$$\begin{aligned} L(G_M) = L(A_{q_0}) &= \{x \in \Sigma^* \mid (q_0, x) \underset{M}{\vdash^*} (q_f, \varepsilon) \\ &\quad \text{jollakin } q_f \in F\} \\ &= L(M) = L. \quad \square \end{aligned}$$

Lause 3.2

Jokainen oikealle lineaarisella kieliopilla tuotettava kieli on säännöllinen.

Todistus. Olkoon $G = (V, \Sigma, P, S)$ oikealle lineaarinen kielioppi. Muodostetaan kielen $L(G)$ tunnistava epädeterministinen äärellinen automaatti $M_G = (Q, \Sigma, \delta, q_S, F)$ seuraavasti:

M_G:n tilat vastaavat G:n välikkeitä:

$$Q = \{q_A \mid A \in V - \Sigma\}.$$

M_G:n alkutila on lähtösymbolia S vastaava tila q_S.

M_G:n syöteaakkosto on G:n pääteaakkosto Σ.

M_G:n siirtymäfunktio δ jäljittelee G:n produktioita siten, että kutakin produktiota $A \to aB$ kohden automaatissa on siirtymä $q_A \xrightarrow{a} q_B$ (so. $q_B \in \delta(q_A, a)$).

M_G:n lopputiloja ovat ne tilat, joita vastaaviin välikkeisiin liittyy G:ssä ε-produktio:

$$F = \{q_A \in Q \mid A \to \varepsilon \in P\}.$$

Konstruktion oikeellisuus voidaan jälleen tarkastaa induktiolla G:n tuottamien ja M_G:n hyväksymien merkkijonojen pituuden suhteen. □

3.3 KIELIOPPIEN JÄSENNYSONGELMA

Ratkaistava tehtävä:

"Annettu yhteydetön kielioppi G ja merkkijono x. Onko $x \in L(G)$?"

Ratkaisumenetelmä = *jäsennysalgoritmi*.

Useita vaihtoehtoisia menetelmiä, erityisesti kun G on jotain rajoitettua (käytännössä esiintyvää) muotoa.

Johdot ja jäsennyspuut

Olkoon $\gamma \in V^*$ kieliopin $G = (V, \Sigma, P, S)$ lausejohdos.

Lähtösymbolista S merkkijonoon γ johtavaa suorien johtojen jonoa

$$S = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_n = \gamma$$

sanotaan γ:n *johdoksi* G:ssä.

Johdon *pituus* on siihen kuuluvien suorien johtojen määrä (edellä n).

Esimerkki: lauseen $a + a$ johtoja kieliopissa G_{expr}:

$$\begin{array}{lllllll}
\text{(i)} & E & \Rightarrow & E+T & \Rightarrow & T+T & \Rightarrow & F+T \\
 & & \Rightarrow & a+T & \Rightarrow & a+F & \Rightarrow & a+a \\
\text{(ii)} & E & \Rightarrow & E+T & \Rightarrow & E+F & \Rightarrow & T+F \\
 & & \Rightarrow & F+F & \Rightarrow & F+a & \Rightarrow & a+a \\
\text{(iii)} & E & \Rightarrow & E+T & \Rightarrow & E+F & \Rightarrow & E+a \\
 & & \Rightarrow & T+a & \Rightarrow & F+a & \Rightarrow & a+a.
\end{array}$$

Johto $\gamma \Rightarrow^* \gamma'$ on *vasen johto*, merkitään

$$\gamma \underset{\text{lm}}{\Rightarrow}^* \gamma',$$

jos kussakin johtoaskelessa on produktiota sovellettu merkkijonon vasemmanpuoleisimpaan välikkeeseen (edellä johto (i)).

Vastaavasti määritellään *oikea johto* (edellä (iii)), jota merkitään

$$\gamma \underset{\text{rm}}{\Rightarrow}^* \gamma'$$

Suoria vasempia ja oikeita johtoaskelia merkitään $\gamma \underset{\text{lm}}{\Rightarrow} \gamma'$ ja $\gamma \underset{\text{rm}}{\Rightarrow} \gamma'$.

Olkoon $G = (V, \Sigma, P, S)$ yhteydetön kielioppi.

Kieliopin G mukainen *jäsennyspuu* on järjestetty puu, jolla on seuraavat ominaisuudet:

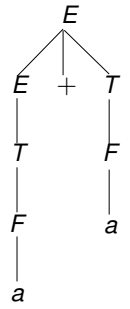
(i) puun solmut on nimetty joukon $V \cup \{\varepsilon\}$ alkioilla siten, että sisäsolmujen nimet ovat välikkeitä (so. joukosta $N = V - \Sigma$) ja juurisolmun nimenä on lähtösymboli S;

(ii) jos A on puun jonkin sisäsolmun nimi, ja $X_1, \ldots, X_k$ ovat sen jälkeläisten nimet järjestyksessä, niin $A \to X_1 \ldots X_k$ on G:n produktio.

Jäsennyspuun τ *tuotos* on merkkijono, joka saadaan liittämällä yhteen sen lehtisolmujen nimet esijärjestyksessä (“vasemmalta oikealle”).

Esimerkki.

Lauseen $a + a$ jäsennyspuu kieliopissa G_{expr}:



Lauseen johto:

$$\begin{aligned} E &\Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \\ &\Rightarrow a+T \Rightarrow a+F \Rightarrow a+a \end{aligned}$$

Johtoa

$$S = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_n = \gamma$$

vastaavan jäsennyspuun muodostaminen:

(i) puun juuren nimeksi tulee S; jos $n = 0$, niin puussa ei ole muita solmuja; muuten

(ii) jos ensimmäisessä johtoaskelessa on sovellettu produktiota $S \to X_1 X_2 \ldots X_k$, niin juurelle tulee k jälkeläissolmua, joiden nimet vasemmalta oikealle ovat $X_1, X_2, \ldots, X_k$;

(iii) jos seuraavassa askelessa on sovellettu produktiota $X_i \to Y_1 Y_2 \ldots Y_l$, niin juuren i:nnelle jälkeläissolmulle tulee l jälkeläistä, joiden nimet vasemmalta oikealle ovat $Y_1, Y_2, \ldots, Y_l$; ja niin edelleen.

Konstruktiosta huomataan, että jos τ on jotakin johtoa $S \Rightarrow^* \gamma$ vastaava jäsennyspuu, niin τ:n tuotos on γ.

Olkoon τ kieliopin G mukainen jäsennyspuu, jonka tuotos on päätemerkkijono x.

Tällöin τ:sta saadaan vasen johto x:lle käymällä puun solmut läpi esijärjestyksessä (“ylhäältä alas, vasemmalta oikealle”) ja laventamalla vastaan tulevat välikkeet järjestyksessä puun osoittamalla tavalla.

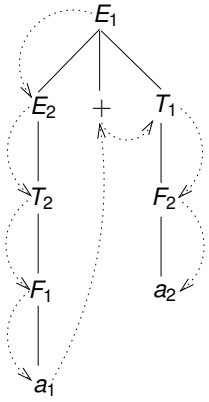
Oikea johto saadaan käymällä puu läpi käänteisessä esijärjestyksessä (“ylhäältä alas, oikealta vasemmalle”).

Muodostamalla annetusta vasemmasta johdosta $S \underset{\text{lm}}{\Rightarrow}^* x$ ensin jäsennyspuu edellä esitetyllä tavalla, ja sitten jäsennyspuusta vasen johto, saadaan takaisin alkuperäinen johto; vastaava tulos pätee myös oikeille johdoille.

Esimerkki.

Lauseen $a + a$ vasemman johdon muodostaminen jäsennyspuusta.

Jäsennyspuu:



Solmut esijärjestyksessä:

$E_1 E_2 T_2 F_1 a_1 + T_1 F_2 a_2$

Vasen johto:

$$E \underset{\text{lm}}{\Rightarrow} E + T \underset{\text{lm}}{\Rightarrow} T + T \underset{\text{lm}}{\Rightarrow} F + T \underset{\text{lm}}{\Rightarrow} a + T \underset{\text{lm}}{\Rightarrow} a + F \underset{\text{lm}}{\Rightarrow} a + a$$

Lause 3.3

Olkoon $G = (V, \Sigma, P, S)$ yhteydetön kielioppi. Tällöin:

(i) jokaisella G:n lausejohdoksella γ on G:n mukainen jäsennyspuu τ, jonka tuotos on γ;

(ii) jokaista G:n mukaista jäsennyspuuta τ, jonka tuotos on päätemerkkijono x, vastaavat yksikäsitteiset vasen ja oikea johto $S \underset{\text{lm}}{\Rightarrow}^* x$ ja $S \underset{\text{rm}}{\Rightarrow}^* x$.

Seuraus 3.4 Jokaisella G:n lauseella on vasen ja oikea johto.

Siis: yhteydettömän kieliopin tuottamien lauseiden jäsennyspuut, vasemmat ja oikeat johdot vastaavat yksikäsitteisesti toisiaan.

Jäsennysongelman ratkaisuun katsotaan usein kuuluvan pelkän päätösongelman “Onko $x \in L(G)$?” ratkaisemisen lisäksi jonkin näistä jäsennysesityksistä tuottaminen.

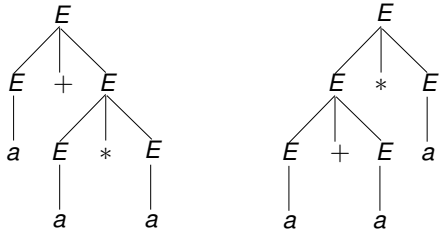
Kieliopin moniselitteisyys

Lauseella voi olla kieliopissa useita jäsennyksiä.

Esimerkki. Tarkastellaan yksinkertaisten aritmeettisten lausekkeiden kielioppia:

$$G'_{\text{expr}} = \{E \to E + E,\ E \to E * E,\ E \to a,\ E \to (E)\}.$$

Lauseella $a + a * a$ on tässä kieliopissa kaksi jäsennystä:



Yhteydetön kielioppi G on *moniselitteinen*, jos jollakin G:n lauseella x on kaksi erilaista G:n mukaista jäsennyspuuta. Muuten kielioppi on *yksiselitteinen*.

Moniselitteisyys on tietojenkäsittelysovelluksissa yleensä ei-toivottu ominaisuus, koska se merkitsee että annetulla lauseella on kaksi vaihtoehtoista “tulkintaa.”

Yhteydetön kieli, jonka tuottavat kieliopit ovat kaikki moniselitteisiä, on *luonnostaan moniselitteinen*.

Esimerkiksi kielioppi G'_{expr} on moniselitteinen, kieliopit G_{expr} ja G_{match} yksiselitteisiä. Kieli $L_{\text{expr}} = L(G'_{\text{expr}})$ ei ole luonnostaan moniselitteinen, koska sillä on myös yksiselitteinen kielioppi G_{expr}. Luonnostaan moniselitteinen on esimerkiksi kieli

$$\{a^i b^j c^k \mid i = j \text{ tai } j = k\}.$$

(Todistus sivuutetaan.)

3.4 Osittava jäsentäminen

Yksi (yleisessä muodossa tehoton!) tapa etsiä vasenta johtoa (jäsennyspuuta) annetun kieliopin G mukaiselle lauseelle x on aloittaa G:n lähtösymbolista ja generoida systemaattisesti kaikki mahdolliset vasemmat johdot (jäsennyspuut), samalla sovittaen muodostetun lausejohdoksen päätemerkkejä (puun lehtiä) x:n merkkeihin. Ei-yhteensopivuuden ilmetessä peruutetaan viimeksi tehty produktiovalinta ja kokeillaan järjestyksessä seuraavaa vaihtoehtoa.

Tällaista lauseenjäsennystapaa sanotaan *osittavaksi*, koska siinä tarkasteltu lause yritetään johtaa kieliopin lähtösymbolista osittamalla se valittujen produktioiden mukaisiin rakenneosiin ja yrittämällä näin, tarvittaessa toistuvasti edelleen osittamalla, sovittaa kieliopin tuottamaa rakennetta yhteen lauseen rakenteen kanssa.

Esim. Tarkastellaan kielioppia G:

$$\begin{aligned} E &\rightarrow T+E \mid T-E \mid T \\ T &\rightarrow a \mid (E). \end{aligned}$$

Lauseen $a-a$ osittava jäsennys G:n suhteen:

$$\begin{aligned} E &\Rightarrow T+E \Rightarrow a+T && \text{[ristiriita; peruutetaan]} \\ &\Rightarrow (E)+T && \text{[ristiriita; peruutetaan]} \\ &\Rightarrow T-E \Rightarrow a-E \Rightarrow a-T+E \Rightarrow a-a+E && \\ & && \text{[ristiriita; peruutetaan]} \\ &\Rightarrow T-E \Rightarrow a-E \Rightarrow a-T+E \Rightarrow a-(E)+E && \\ & && \text{[ristiriita; peruutetaan]} \\ & \Rightarrow a-T-E \Rightarrow a-a-E && \\ & && \text{[ristiriita; peruutetaan]} \\ & \Rightarrow a-T-E \Rightarrow a-(E)-E && \\ & && \text{[ristiriita; peruutetaan]} \\ & \Rightarrow a-T \Rightarrow a-a && \text{[OK!]} \end{aligned}$$

Em. osittava jäsennystekniikka saadaan huomattavasti tehokkaammaksi, jos kieliopilla on sellainen ominaisuus, että jäsennyksen joka vaiheessa määrää tavoitteena olevan lauseen *seuraava merkki* yksikäsitteisesti sen, mikä lavennettavana olevaan välikkeeseen liittyvä produktio on valittava. Kielioppia, jolla on tämä ominaisuus, sanotaan *LL(1)-tyyppiseksi*.

Muokataan G:stä välikkeen E produktiot "tekijöimällä" ekvivalentti kielioppi G':

$$\begin{aligned} E &\rightarrow TE' \\ E' &\rightarrow +E \mid -E \mid \varepsilon \\ T &\rightarrow a \mid (E). \end{aligned}$$

Esimerkiksi lauseen $a-a$ jäsentäminen G':n suhteen (kulloisenkin produktiovalinnan määräävä syötemerkki on tässä merkitty vastaavan johtonuolen päälle):

$$E \underset{\text{lm}}{\Rightarrow} TE' \overset{a}{\underset{\text{lm}}{\Rightarrow}} aE' \overset{-}{\underset{\text{lm}}{\Rightarrow}} a-E \underset{\text{lm}}{\Rightarrow} a-TE' \overset{a}{\underset{\text{lm}}{\Rightarrow}} a-aE' \overset{\varepsilon}{\underset{\text{lm}}{\Rightarrow}} a-a.$$

LL(1)-tyyppiselle kieliopille on helppo kirjoittaa jäsennysohjelma suoraan rekursiivisina proseduureina; esimerkiksi kieliopille G' Pythonilla seuraavasti:

```
from sys import exit,stdin
def error(s): print s; exit(1)
def t():
    global next
    if next=="a":
        print "T -> a"
        next=stdin.read(1)
    elif next=="(":
        print "T -> (E)"
        next=stdin.read(1)
        e()
        if next!=")": error(") expected.")
        next=stdin.read(1)
    else: error("T cannot start with %s"%(next))
```

```
def e():
    print "E -> TE'"
    t(); eprime()
def eprime():
    global next
    if next=="+":
        print "E' -> +E"
        next=stdin.read(1)
        e()
    elif next=="-":
        print "E' -> -E"
        next=stdin.read(1)
        e()
    else: print "E ->"

next=stdin.read(1)
e()
```


Esimerkiksi syötejonoa `a-(a+a)` käsitellessään ohjelma tulostaa seuraavat rivit:

```
E → TE'
T → a
E' → -E
E → TE'
T → (E)
E → TE'
T → a
E' → +E
E → TE'
T → a
E' →
E' →
```

Tulostus vastaa vasenta johtoa:

$$\begin{aligned} E &\Rightarrow TE' \Rightarrow aE' \Rightarrow a-E \Rightarrow a-TE' \\ &\Rightarrow a-(E)E' \Rightarrow a-(TE')E' \\ &\Rightarrow a-(aE')E' \Rightarrow a-(a+E)E' \\ &\Rightarrow a-(a+TE')E' \Rightarrow a-(a+aE')E' \\ &\Rightarrow a-(a+a)E' \Rightarrow a-(a+a). \end{aligned}$$

3.5 Attribuuttikieliopit

Tapa liittää yhteydettömiin kielioppeihin yksinkertaista kielen semantiikan kuvausta.

Kukin kieliopin mukaisen jäsennyspuun solmu, jonka nimenä on symboli X, ajatellaan "tietueeksi", joka on "tyyppiä" X. "Tietuetyyppiin" X kuuluvia "kenttiä" sanotaan X:n *attribuuteiksi* ja merkitään $X.s$, $X.t$ jne. Kussakin X-tyyppisessä jäsennyspuun solmussa ajatellaan olevan X:n attribuuteista eri *ilmentymät*.

Kieliopin produktioihin $A \to X_1 \ldots X_k$ liitetään attribuuttien *evaluointisääntöjä*, jotka ilmaisevat miten annetun jäsennyspuun solmun attribuutti-ilmentymien arvot määräytyvät sen isä- ja jälkeläissolmujen attribuutti-ilmentymien arvoista.

Säännöt voivat olla periaatteessa minkälaisia funktioita tahansa, kunhan niiden argumentteina esiintyy vain paikallisesti saatavissa olevaa tietoa. Tarkemmin sanoen: produktioon $A \to X_1 \ldots X_k$ liitettävissä säännöissä saa mainita vain symbolien $A, X_1, \ldots, X_k$ attribuutteja.

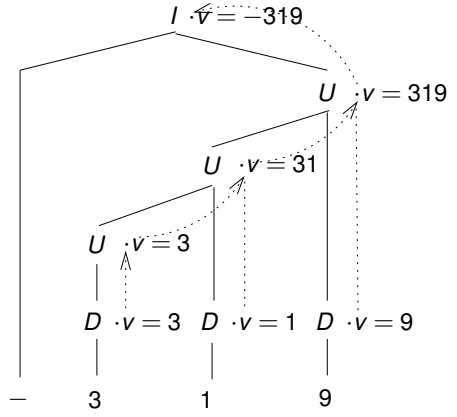
Esimerkki. Etumerkillisten kokonaislukujen arvojen määrittäminen (kielioppi + attribuuttien evaluointisäännöt).

Kuhunkin jäsennyspuun X-tyyppiseen välikesolmuun liitetään attribuutti-ilmentymä $X.v$, jonka arvoksi tulee X:stä tuotetun numerojonon lukuarvo; erityisesti juurisolmun v-ilmentymän arvoksi tulee koko puun tuotoksena olevan numerojonon luku.

Produktiot:			*Evaluointisäännöt:*		
I	$\to$	$+U$	$I.v$	$:=$	$U.v$
I	$\to$	$-U$	$I.v$	$:=$	$-U.v$
I	$\to$	U	$I.v$	$:=$	$U.v$
U	$\to$	D	$U.v$	$:=$	$D.v$
U	$\to$	UD	$U_1.v$	$:=$	$10 * U_2.v + D.v$
D	$\to$	0	$D.v$	$:=$	0
$\ldots$					
D	$\to$	9	$D.v$	$:=$	9

Produktioon $U \to UD$ liittyvässä evaluointisäännössä on välikkeen U eri esiintymät erotettu indekseillä.

Esimerkiksi näitä sääntöjä käyttäen attributoitu lauseen “-319” jäsennyspuu on seuraava:



Attribuuttikieliopin attribuutti t on *synteettinen*, jos sen kuhunkin produktioon $A \to X_1 \ldots X_k$ liittyvä evaluointisääntö on muotoa

$$A.t := f(A, X_1, \ldots, X_k).$$

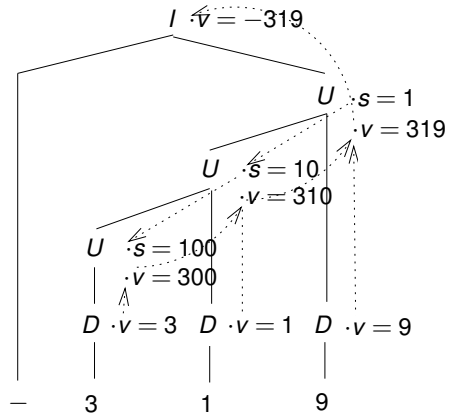
Tällöin jäsennyspuussa kunkin solmun mahdollisen t- ilmentymän arvo riippuu vain solmun omien ja sen jälkeläisten attribuutti-ilmentymien arvoista. Muunlaiset attribuutit ovat *periytyviä*.

Attribuuttisemantiikan kuvauksessa pyritään käyttämään pääasiassa synteettisiä attribuutteja, koska ne voidaan evaluoida helposti yhdellä jäsennyspuun lehdistä juureen suuntautuvalla läpikäynnillä. Mitään periaatteellista estettä myös perittyjen attribuuttien käyttöön ei kuitenkaan ole — kunhan attribuutti-ilmentymien riippuvuusverkkoihin ei tule syklejä.

Esimerkki: kokonaislukujen arvon määrittäminen periytyvää “positiokerroin”-attribuuttia s ja synteettistä “arvo”-attribuuttia v käyttäen:

Produktiot:			*Evaluointisäännöt:*					
I	$\to$	$+U$	$U.s$	$:=$	1,	$I.v$	$:=$	$U.v$
I	$\to$	$-U$	$U.s$	$:=$	1,	$I.v$	$:=$	$-U.v$
I	$\to$	U	$U.s$	$:=$	1,	$I.v$	$:=$	$U.v$
U	$\to$	D				$U.v$	$:=$	$(D.v) * (U.s)$
U	$\to$	UD	$U_2.s$	$:=$	$10 * (U_1.s)$,			
			$U_1.v$	$:=$	$U_2.v + (D.v) * (U_1.s)$			
D	$\to$	0				$D.v$	$:=$	0
$\vdots$								
D	$\to$	9				$D.v$	$:=$	9

Em. positiokerrointekniikkaa käyttäen saadaan lauseelle “-319” seuraava attributoitu jäsennyspuu:



Attribuutti-ilmentymien arvot voidaan usein laskea suoraan jäsennysrutiineissa muodostamatta jäsennyspuuta eksplisiittisesti.

Esimerkki. Ohjelma, joka muuntaa syötteenä annettuja aritmeettisia lausekkeita *postfix*-esitykseen.

Tavanomaiseen, yksinkertaisia aritmeettisia lausekkeita tuottavaan kielioppiin liitetään yksi synteettinen, merkkijonoarvoinen attribuutti pf; kuhunkin välikkeeseen X liittyvän attribuutti-ilmentymän $X.pf$ arvo on X:stä tuotetun alilausekkeen postfix-esitys.

Produktiot:			*Evaluointisäännöt:*		
E	$\rightarrow$	$T+E$	$E_1.pf$	$:=$	$(T.pf)\hat{}(E_2.pf)\hat{}('+')$
E	$\rightarrow$	T	$E.pf$	$:=$	$T.pf$
T	$\rightarrow$	$F*T$	$T_1.pf$	$:=$	$(F.pf)\hat{}(T_2.pf)\hat{}('*')$
T	$\rightarrow$	F	$T.pf$	$:=$	$F.pf$
F	$\rightarrow$	a	$F.pf$	$:=$	$'a'$
F	$\rightarrow$	(E)	$F.pf$	$:=$	$E.pf$

Rekursiivisesti etenevä jäsentäjä, joka evaluoi attribuutti-ilmentymien arvot suoraan jäsennyksen yhteydessä:

```
from sys import stdin
def error(s): print s; exit(1)
def f():          # F -> a | (E)
    global next
    if next=="a":
        next=stdin.read(1)
        return "a"                    # F.pf := a
    elif next=="(":
        next=stdin.read(1)
        pf=e()
        if next!=")": error(") expected.")
        next=stdin.read(1)
        return pf                     # F.pf := E.pf
    else: error("F cannot start with this.")
```



```
def e():          # E -> T + E | T
    global next
    pf1=t()
    if next=="+":
        next=stdin.read(1)
        return pf1+e()+"+" # E1.pf := T.pf E2.pf +
    else: return pf1       # E.pf  := T.pf
def t():          # T -> F * T | F
    global next
    pf1=f()
    if next=="*":
        next=stdin.read(1)
        return pf1+t()+"*" # T1.pf := F.pf T2.pf *
    else: return pf1       # T.pf  := F.pf

next=stdin.read(1)
print e()
```

Tietojenkäsittelyteorian perusmoduuli

Muille kuin tietotekniikan opiskelijoille

Muiden kuin tietotekniikan opiskelijoiden on tietojenkäsittelyteoriaa sivuaineena opiskellakseen luettava tietotekniikan moduuli B1 (jonka esitietovaatimuksena ohjelmoinnin peruskurssi) ja valittava siihen kurssit

- T–79.1001 Tietojenkäsittelyteorian perusteet T (4 op)
- T–79.3001 Logiikka tietotekniikassa, perusteet (4 op)
- T–106.1223 Tietorakenteet ja algoritmit Y (5 op)
- T–106.1243 Ohjelmoinnin jatkokurssi L1 (Java) (6 op)
- ja vielä 1 op moduulin monista valinnaisista kursseista.

Tietotekniikan opiskelijoille

Tietotekniikan opiskelijan on opiskeltava A1-moduuli Tietotekniikka T

- T–79.3001 Logiikka tietotekniikassa: perusteet (4 op)
- T–106.3101 Ohjelmoinnin jatkokurssi T2 (C-kieli) (6 op)
- T–76.3601 Introduction to Software Engineering (5 op)
- S–87.3190 Tietokoneen arkkitehtuuri (5 op)

Tietojenkäsittelyteorian jatkomoduuli A2

Tietojenkäsittelyteorian A2-moduuliin kuuluvat kurssit

- T–79.4001 Tietojenkäsittelyteorian seminaari (3 op)
- T–79.4201 Hakuongelmat ja -algoritmit (4 op)
- T–79.4301 Rinnakkaiset ja hajautetut järjestelmät (4 op)
- T–79.4501 Tiedon salaus ja suojaus (4 op)
- T–106.4100 Algoritmien suunnittelu ja analyysi (5 op)
  - tai T–79.5103 Laskennan vaativuusteoria (5 op), jos T–106.4100 sijoitetaan muuhun moduuliin

Lisätietoja ks. `http://www.tcs.tkk.fi/Studies/` , opettajilta, laboratorion kurssi-ilmoitustaululta (ovesta ulos, portaat ylös) ja pääaineinfossa kevään puolella.

Tietojenkäsittelyteorian syventävä moduuli A3

Tietojenkäsittelyteorian A3-moduuliin voi yhdistellä 20 op kursseja varsin vapaasti laboratorion painopistealueilta:

- Laskennallinen logiikka
  - Laskennallisen logiikan jatkokurssi, Laskennallisen logiikan erikoiskurssi, Laskennan vaativuusteoria
- Verifiointi
  - Reaktiiviset järjestelmät, Symbolinen mallintarkastus, Turvallisuuskriittiset järjestelmät, Formaali konformanssitestaus, Formaalit menetelmät
- Kombinatoriikka
  - Diskreetit rakenteet, Kombinatoriset algoritmit, Graafiteoria, Kombinatoriset mallit ja stokastiset algoritmit
- Kryptologia
  - Kryptologia, Kryptologian jatkokurssi
- Muita
  - Tietojenkäsittelyteorian erikoistyö, Tietojenkäsittelyteorian lisensiaattikurssi, Yksilöllinen opintojakso, Tietojenkäsittelyteorian tutkimuskurssi

2.8 Säännöllisten kielten rajoituksista

Kardinaliteettisyistä on oltava olemassa (paljon) ei-säännöllisiä kieliä: kieliä on ylinumeroituva määrä, säännöllisiä lausekkeita vain numeroituvasti.

Voidaanko löytää konkreettinen, *mielenkiintoinen* esimerkki kielestä, joka ei olisi säännöllinen? Helposti.

Säännöllisten kielten perusrajoitus: äärellisillä automaateilla on vain rajallinen “muisti”. Siten ne eivät pysty ratkaisemaan ongelmia, joissa vaaditaan mielivaltaisen suurten lukujen tarkkaa muistamista.

Esimerkki: sulkulausekekieli

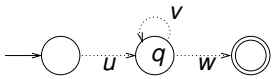
$$L_{\text{match}} = \{(^k)^k \mid k \geq 0\}.$$

Formalisointi: “pumppauslemma”.

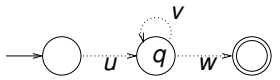
Lemma 2.6 (Pumppauslemma)

Olkoon A säännöllinen kieli. Tällöin on olemassa sellainen $n \geq 1$, että mikä tahansa $x \in A$, $|x| \geq n$, voidaan jakaa osiin $x = uvw$ siten, että $|uv| \leq n$, $|v| \geq 1$, ja $uv^i w \in A$ kaikilla $i = 0, 1, 2, \ldots$

Todistus. Olkoon M jokin A:n tunnistava deterministinen äärellinen automaatti, ja olkoon n M:n tilojen määrä. Tarkastellaan M:n läpikäymiä tiloja syötteellä $x \in A$, $|x| \geq n$. Koska M jokaisella x:n merkillä siirtyy tilasta toiseen, sen täytyy kulkea jonkin tilan kautta (ainakin) kaksi kertaa — itse asiassa jo x:n n:n ensimmäisen merkin aikana. Olkoon q ensimmäinen toistettu tila.



Olkoon u M:n käsittelemä x:n alkuosa sen tullessa ensimmäisen kerran tilaan q, v se osa x:stä jonka M käsittelee ennen ensimmäistä paluutaan q:hun, ja w loput x:stä. Tällöin on $|uv| \leq n$, $|v| \geq 1$, ja $uv^i w \in A$ kaikilla $i = 0, 1, 2, \ldots$ □



Esimerkki. Tarkastellaan em. sulkulausekekieltä (merk. '(' = a, ')' = b):

$$L = L_{\text{match}} = \{a^k b^k \mid k \geq 0\}.$$

Oletetaan, että L olisi säännöllinen. Tällöin pitäisi pumppauslemman mukaan olla jokin $n \geq 1$, jota pitempiä L:n merkkijonoja voidaan pumpata. Valitaan $x = a^n b^n$, jolloin $|x| = 2n > n$. Lemman mukaan x voidaan jakaa pumpattavaksi osiin $x = uvw$, $|uv| \leq n$, $|v| \geq 1$; siis on oltava

$$u = a^i,\ v = a^j,\ w = a^{n-(i+j)} b^n, \quad i \leq n-1,\ j \geq 1.$$

Mutta esimerkiksi "0-kertaisesti" pumpattaessa:

$$uv^0 w = a^i a^{n-(i+j)} b^n = a^{n-j} b^n \notin L.$$

Siten L ei voi olla säännöllinen.

3.6 Cocke-Younger-Kasami -jäsennysalgoritmi

Osittava jäsentäminen on selkeä ja tehokas jäsennysmenetelmä LL(1)-kieliopeille: n merkin mittaisen syötemerkkijonon käsittely sujuu ajassa $O(n)$. LL(1)-kieliopit ovat kuitenkin melko rajoitettu luokka; yleisen jäsennysongelman ratkaisu ei ole yhtä helppoa.

Periaatteessa ongelma voidaan ratkaista esim. soveltamalla yleistä (peruuttavaa) osittavaa jäsennystä, mutta käytännössä vaikeudeksi muodostuu erilaisten kokeiltavien johtovaihtoehtojen suuri määrä. (Tyypillisesti $O(c^n)$ kpl jollakin $c \geq 2$.)

Cocke-Younger-Kasami -algoritmi on yleiseen ns. dynaamisen ohjelmoinnin tekniikkaan (t. osaratkaisujen taulukointiin) perustuva menetelmä mielivaltaisen yhteydettömän kieliopin tuottamien merkkijonojen tunnistamiseen. Menetelmä toimii ajassa $O(n^3)$, missä n on tutkittavan merkkijonon pituus.

Algoritmia varten määritellään ensin joitakin kielioppimuunnoksia.

1. ε-produktioiden poistaminen

Olkoon $G = (V, \Sigma, P, S)$ yhteydetön kielioppi. Välike $A \in V - \Sigma$ on *tyhjentyvä*, jos $A \underset{G}{\Rightarrow}^* \varepsilon$.

Lemma 3.5. Mistä tahansa yhteydettömästä kieliopista G voidaan muodostaa ekvivalentti kielioppi G', jossa enintään lähtösymboli on tyhjentyvä.

Todistus.

Olkoon $G = (V, \Sigma, P, S)$. Selvitetään ensin G:n tyhjentyvät välikkeet seuraavasti:

(i) asetetaan aluksi

$$\text{NULL} := \{A \in V - \Sigma \mid A \to \varepsilon \text{ on } G\text{:n produktio}\};$$

(ii) toistetaan sitten seuraavaa NULL-joukon laajennusoperaatiota, kunnes joukko ei enää kasva:

$$\begin{aligned} \text{NULL} \quad := \quad & \text{NULL} \cup \\ & \{A \in V - \Sigma \mid A \to B_1 \dots B_k \text{ on } G\text{:n prod.}, \\ & B_i \in \text{NULL kaikilla } i = 1, \dots, k\}. \end{aligned}$$

Tämän jälkeen korvataan kukin G:n produktio $A \to X_1 \dots X_k$ kaikkien sellaisten produktioiden joukolla, jotka ovat muotoa

$$A \to \alpha_1 \dots \alpha_k, \quad \text{missä}$$

$$\alpha_i = \begin{cases} X_i, & \text{jos } X_i \notin \text{NULL}; \\ X_i \text{ tai } \varepsilon, & \text{jos } X_i \in \text{NULL}. \end{cases}$$

Lopuksi poistetaan kaikki muotoa $A \to \varepsilon$ olevat produktiot. Jos poistettavana on myös produktio $S \to \varepsilon$, otetaan muodostettavaan kielioppiin G' uusi lähtösymboli S' ja sille produktiot $S' \to S$ ja $S' \to \varepsilon$. □

Esimerkki.

Poistetaan ε-produktiot kieliopista:

$$\begin{aligned} S &\to A \mid B \\ A &\to aBa \mid \varepsilon \qquad \Rightarrow \quad (\text{NULL} = \{A, B, S\}) \\ B &\to bAb \mid \varepsilon \end{aligned}$$

$$\begin{aligned} S &\to A \mid B \mid \varepsilon \\ A &\to aBa \mid aa \mid \varepsilon \qquad \Rightarrow \\ B &\to bAb \mid bb \mid \varepsilon \end{aligned}$$

$$\begin{aligned} S' &\to S \mid \varepsilon \\ S &\to A \mid B \\ A &\to aBa \mid aa \\ B &\to bAb \mid bb. \end{aligned}$$

2. Yksikköproduktioiden poistaminen

Produktio muotoa $A \to B$, missä A ja B ovat välikkeitä, on *yksikköproduktio*.

Lemma 3.6. Mistä tahansa yhteydettömästä kieliopista G voidaan muodostaa ekvivalentti kielioppi G', jossa ei ole yksikköproduktioita.

Todistus.

Olkoon $G = (V, \Sigma, P, S)$. Selvitetään ensin G:n kunkin välikkeen "yksikköseuraajat" seuraavasti:

(i) asetetaan aluksi kullekin $A \in V - \Sigma$:

$$F(A) := \{B \in V - \Sigma \mid A \to B \text{ on } G\text{:n produktio}\};$$

(ii) toistetaan sitten seuraavia F-joukkojen laajennusoperaatioita, kunnes joukot eivät enää kasva:

$$F(A) := F(A) \cup \bigcup \{F(B) \mid A \to B \text{ on } G\text{:n produktio}\}.$$

Tämän jälkeen poistetaan G:stä kaikki yksikköproduktiot ja lisätään niiden sijaan kaikki mahdolliset produktiot muotoa $A \to \omega$, missä $B \to \omega$ on G:n ei-yksikköproduktio jollakin $B \in F(A)$. □

Esimerkki.

Poistetaan yksikköproduktiot aiemmin muodostetusta kieliopista:

$$\begin{array}{lcl} S' & \to & S \mid \varepsilon \\ S & \to & A \mid B \\ A & \to & aBa \mid aa \\ B & \to & bAb \mid bb. \end{array}$$

Välikkeiden yksikköseuraajat ovat: $F(S') = \{S, A, B\}$, $F(S) = \{A, B\}$, $F(A) = F(B) = \emptyset$. Korvaamalla yksikköproduktiot edellä esitetyllä tavalla saadaan kielioppi:

$$\begin{array}{lcl} S' & \to & aBa \mid aa \mid bAb \mid bb \mid \varepsilon \\ S & \to & aBa \mid aa \mid bAb \mid bb \\ A & \to & aBa \mid aa \\ B & \to & bAb \mid bb. \end{array}$$

(Huomataan, että välike S on nyt itse asiassa "turha", so. se ei voi esiintyä minkään kieliopin lauseen johdossa. Myös turhat välikkeet voidaan haluttaessa poistaa kieliopista samantapaisella algoritmilla (HT).)

Chomskyn normaalimuoto

Yhteydetön kielioppi $G = (V, \Sigma, P, S)$ on *Chomskyn normaalimuodossa*, jos sen välikkeistä enintään S on tyhjentyvä, ja mahdollista produktiota $S \to \varepsilon$ lukuunottamatta muut produktiot ovat muotoa

$$A \to BC \qquad \text{tai} \qquad A \to a,$$

missä A, B ja C ovat välikkeitä ja a on päätemerkki.

Lisäksi vaaditaan yksinkertaisuuden vuoksi, että lähtösymboli S ei esiinny minkään produktion oikealla puolella.

Lause 3.7.

Mistä tahansa yhteydettömästä kieliopista G voidaan muodostaa ekvivalentti Chomskyn normaalimuotoinen kielioppi G'.

Todistus. Olkoon $G = (V, \Sigma, P, S)$. Mikäli lähtösymboli S esiintyy G:ssä jonkin produktion oikealla puolella, otetaan käyttöön uusi lähtösymboli S' ja lisätään G:hen produktio $S' \to S$. Poistetaan sitten G:stä ε-produktiot ja yksikköproduktiot lemmojen 3.5 ja 3.6 konstruktioilla. Tämän jälkeen kaikki G:n produktiot ovat muotoa $A \to a$ tai $A \to X_1 \dots X_k$, $k \geq 2$ (tai $S \to \varepsilon / S' \to \varepsilon$).

Lisätään nyt kielioppiin kutakin päätemerkkiä a varten uusi välike C_a ja sille produktio $C_a \to a$. Korvataan kussakin muotoa $A \to X_1 \dots X_k$, $k \geq 2$, olevassa produktiossa ensin kaikki päätemerkit em. uusilla välikkeillä, ja sitten koko produktio produktiojoukolla

$$\begin{aligned} A &\to X_1 A_1 \\ A_1 &\to X_2 A_2 \\ &\vdots \\ A_{k-2} &\to X_{k-1} X_k, \end{aligned}$$

missä $A_1, \dots, A_{k-2}$ ovat jälleen uusia välikkeitä. □

Esimerkki. Kielioppi:

$$\begin{aligned} S &\to aBCd \mid bbb \\ B &\to b \\ C &\to c \end{aligned}$$

Em. konstruktiolla saatu Chomskyn normaalimuoto:

$$\begin{aligned} S &\to C_a S_1^1 \\ S_1^1 &\to B S_2^1 \\ S_2^1 &\to C C_d \\ S &\to C_b S_1^2 \\ S_1^2 &\to C_b C_b \\ B &\to b \\ C &\to c \\ C_a &\to a \\ C_b &\to b \\ C_c &\to c \\ C_d &\to d. \end{aligned}$$

CYK-algoritmi

Olkoon $G = (V, \Sigma, P, S)$ yhteydetön kielioppi. Lauseen 3.7 nojalla voidaan olettaa, että G on Chomskyn normaalimuodossa. Kysymys, kuuluuko annettu merkkijono x kieleen $L(G)$ voidaan tällöin ratkaista seuraavasti:

Jos $x = \varepsilon$, niin $x \in L(G)$ joss $S \to \varepsilon$ on G:n produktio.

Muussa tapauksessa merkitään $x = a_1 \dots a_n$ ja tarkastellaan x:n eri osajonojen tuottamista.

Merkitään N_{ik}:lla niiden välikkeiden A joukkoa, joista voidaan tuottaa x:n positiosta i alkava, k merkin mittainen osajono:

$$\begin{aligned} N_{ik} &= \{A \in V - \Sigma \mid A \underset{G}{\Rightarrow}^* a_i \dots a_{i+k-1}\}, \\ &\quad 1 \leq i \leq i + k - 1 \leq n. \end{aligned}$$

Joukot N_{ik} voidaan laskea taulukoimalla lyhyistä osajonoista pitempiin seuraavassa esitettävällä tavalla. Selvästi on $x \in L(G)$ joss $S \in N_{1n}$.

Joukkojen N_{ik} laskeminen:

(i) asetetaan aluksi kaikilla $i = 1, \ldots, n$:

$$N_{i1} := \{A \in V - \Sigma \mid A \to a_i \text{ on } G\text{:n produktio}\};$$

(ii) lasketaan sitten kaikilla $k = 2, \ldots, n$ ja kullakin k kaikilla $i = 1, \ldots, n-k+1$:

$$N_{ik} := \bigcup_{j=1}^{k-1} \{A \in V - \Sigma \mid A \to BC \text{ on } G\text{:n produktio, missä } B \in N_{ij} \text{ ja } C \in N_{i+j,k-j}\}. \quad \square$$

Esimerkki.

Chomskyn normaalimuotoinen kielioppi G:

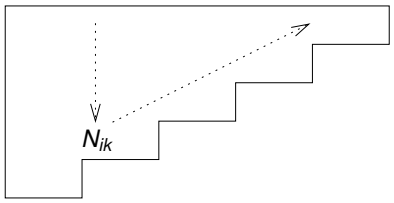
$$\begin{array}{lcl|l} S & \to & AB & BC \\ A & \to & BA & a \\ B & \to & CC & b \\ C & \to & AB & a \end{array}$$

CYK-algoritmin laskenta kieliopilla G ja syötteellä $x = baaba$:

		$i \to$				
	N_{ik}	$1:b$	$2:a$	$3:a$	$4:b$	$5:a$
	1	B	A, C	A, C	B	A, C
	2	S, A	B	S, C	S, A	–
$k \downarrow$	3	$\emptyset$	B	B	–	
	4	$\emptyset$	S, A, C	–		
	5	S, A, C	–			

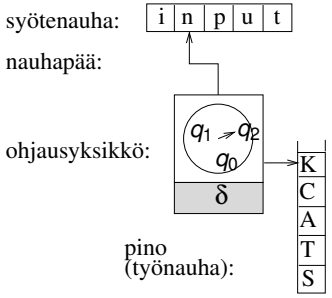
Koska lähtösymboli S kuuluu joukkoon N_{15}, päätellään että x kuuluu kieleen $L(G)$.

Yleisesti ottaen CYK-algoritmissa jotakin joukkoa N_{ik} määritettäessä edetään samanaikaisesti sarakkeessa N_{ij} joukkoa N_{ik} "kohti" ja diagonaalia $N_{i+j,k-j}$ pitkin siitä "poispäin":



3.7 Pinoautomaatit

Yhteydettömille kielille saadaan automaattikarakterisointi ns. *pinoautomaattien* avulla:



Pinoautomaatti on kuin äärellinen automaatti, johon on lisätty rajoittamattoman suuri *pino*. Pinon käyttö on varsin rajallista: automaatti voi lukea, kirjoittaa, poistaa tai lisätä merkkejä vain pinon päälle.

Määritelmä 3.2

Pinoautomaatti on kuusikko

$$M = (Q, \Sigma, \Gamma, \delta, q_0, F),$$

missä

- Q on *tilojen* äärellinen joukko;
- Σ on *syöteaakkosto*;
- Γ on *pinoaakkosto*;
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$ on (joukkoarvoinen) *siirtymäfunktio*;
- $q_0 \in Q$ on *alkutila*;
- $F \subseteq Q$ on *(hyväksyvien) lopputilojen* joukko.

Siirtymäfunktion arvon

$$\delta(q, \sigma, \gamma) = \{(q_1, \gamma_1), \ldots, (q_k, \gamma_k)\}$$

tulkinta on, että ollessaan tilassa q ja lukiessaan syötemerkin σ ja pinomerkin γ automaatti voi siirtyä johonkin tiloista $q_1, \ldots, q_k$ ja korvata vastaavasti pinon päällimmäisen merkin jollakin merkeistä $\gamma_1, \ldots, \gamma_k$. Pinoautomaatit ovat siis yleisessä tapauksessa *epädeterministisiä*.

Jos $\sigma = \varepsilon$, automaatti tekee siirtymän syötemerkkiä lukematta. Jos $\gamma = \varepsilon$, automaatti ei lue pinomerkkiä ja uusi kirjoitettu merkki tulee pinon päälle vanhaa päällimmäistä merkkiä poistamatta ("push"-operaatio). Jos pinosta luettu merkki on $\gamma \neq \varepsilon$ ja kirjoitettavana on $\gamma_i = \varepsilon$, pinosta poistetaan sen päällimmäinen merkki ("pop"-operaatio).

Automaatin *tilanne* on kolmikko $(q, w, \alpha) \in Q \times \Sigma^* \times \Gamma^*$; erityisesti automaatin *alkutilanne syötteellä* x on kolmikko (q_0, x, ε).

Intuitio: tilanteessa (q, w, α) automaatti on tilassa q, syötemerkkijonon käsittelemätön osa on w ja pinossa on ylhäältä alas lukien merkkijono α.

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

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

jos voidaan kirjoittaa $w = \sigma w'$, $\alpha = \gamma\beta$, $\alpha' = \gamma'\beta$ ($|\sigma|, |\gamma|, |\gamma'| \leq 1$), siten että

$$(q', \gamma') \in \delta(q, \sigma, \gamma).$$

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

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

jos on olemassa tilannejono (q_0, w_0, α_0), (q_1, w_1, α_1), …, (q_n, w_n, α_n), $n \geq 0$, siten että

$$(q, w, \alpha) = (q_0, w_0, \alpha_0) \underset{M}{\vdash} (q_1, w_1, \alpha_1) \underset{M}{\vdash} \cdots \underset{M}{\vdash} (q_n, w_n, \alpha_n) = (q', w', \alpha').$$

Pinoautomaatti M *hyväksyy* merkkijonon $x \in \Sigma^*$, jos

$$(q_0, x, \varepsilon) \underset{M}{\vdash}^* (q_f, \varepsilon, \alpha) \quad \text{joillakin } q_f \in F \text{ ja } \alpha \in \Gamma^*,$$

siis jos se syötteen loppuessa on jossakin hyväksyvässä lopputilassa; muuten M *hylkää* x:n.

Automaatin M *tunnistama kieli* on:

$$L(M) = \{x \in \Sigma^* \mid (q_0, x, \varepsilon) \underset{M}{\vdash}^* (q_f, \varepsilon, \alpha) \text{ joillakin } q_f \in F \text{ ja } \alpha \in \Gamma^*\}.$$

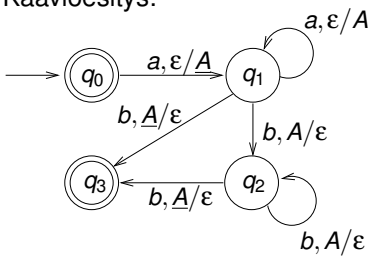
Esimerkki. Pinoautomaatti kielelle $\{a^k b^k \mid k \geq 0\}$:

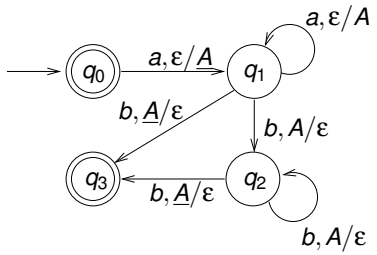
$M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{A, \underline{A}\}, \delta, q_0, \{q_0, q_3\})$,

missä

$$\begin{aligned}
\delta(q_0, a, \varepsilon) &= \{(q_1, \underline{A})\},\\
\delta(q_1, a, \varepsilon) &= \{(q_1, A)\},\\
\delta(q_1, b, A) &= \{(q_2, \varepsilon)\},\\
\delta(q_1, b, \underline{A}) &= \{(q_3, \varepsilon)\},\\
\delta(q_2, b, A) &= \{(q_2, \varepsilon)\},\\
\delta(q_2, b, \underline{A}) &= \{(q_3, \varepsilon)\},\\
\delta(q, \sigma, \gamma) &= \emptyset \quad \text{muilla } (q, \sigma, \gamma).
\end{aligned}$$

Kaavioesitys:





Automaatin toiminta syötteellä *aabb*:

$$\begin{aligned}
(q_0, aabb, \varepsilon) &\vdash (q_1, abb, \underline{A}) \vdash (q_1, bb, A\underline{A})\\
&\vdash (q_2, b, \underline{A}) \vdash (q_3, \varepsilon, \varepsilon).
\end{aligned}$$

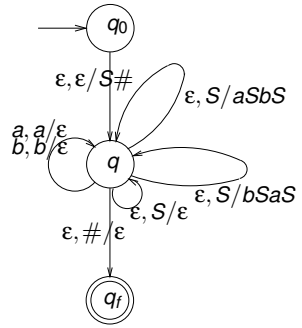
Koska $q_3 \in F = \{q_0, q_3\}$, on siis $aabb \in L(M)$.

Pinoautomaatit ja yhteydettömät kielet

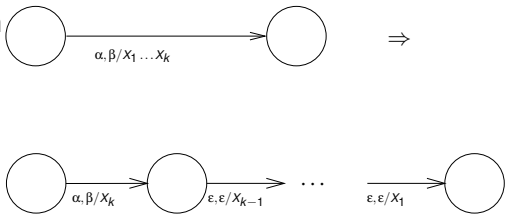
Lause 3.8 Kieli on yhteydetön, jos ja vain jos se voidaan tunnistaa jollakin (epädeterministisellä) pinoautomaatilla. □

Em. lauseen todistus sivuutetaan tässä, mutta periaatteena esim. annettua kielioppia G vastaavan pinoautomaatin M_G toiminnassa on, että M_G:n pinon käyttäytyminen syötteellä x noudattelee G:n mukaisen vasemman lausejohdon $S \underset{\text{lm}}{\Rightarrow}^* x$ etenemistä: jos pinon päällimmäisenä on välikemerkki, sovelletaan jotain G:n produktiota ja lisätään pinon pinnalle vastaavat merkit; jos pinon päällimmäisenä on päätemerkki, se sovitetaan yhteen seuraavan syötemerkin kanssa.

Esimerkki. Kielioppia $\{S \to aSbS \mid bSaS \mid \varepsilon\}$ vastaava pinoautomaatti:



Automaatin kuvauksessa on käytetty seuraavaa luonnollista lyhennemerkintää:



Esimerkiksi syötteellä *abab* on em. automaatilla seuraava hyväksyvä laskenta:

$$\begin{array}{lll} (q_0, abab, \varepsilon) & \vdash & (q, abab, S\#) \\ & \vdash & (q, bab, SbS\#) \\ & \vdash & (q, ab, SaSbS\#) \\ & \vdash & (q, b, SbS\#) \\ & \vdash & (q, \varepsilon, S\#) \\ & \vdash & (q_f, \varepsilon, \varepsilon). \end{array} \qquad \begin{array}{ll} \vdash^* & (q, abab, aSbS\#) \\ \vdash^* & (q, bab, bSaSbS\#) \\ \vdash & (q, ab, aSbS\#) \\ \vdash & (q, b, bS\#) \\ \vdash & (q, \varepsilon, \#) \end{array}$$

Tämä vastaa annetun kieliopin mukaista lauseen *abab* vasenta johtoa:

$$\begin{array}{lllllll} \underline{S} & \Rightarrow & a\underline{S}bS & \Rightarrow & ab\underline{S}aSbS & \Rightarrow & aba\underline{S}bS \\ & \Rightarrow & abab\underline{S} & \Rightarrow & abab. & & \end{array}$$

Pinoautomaatti M on *deterministinen*, jos jokaisella tilanteella (q, w, α) on enintään yksi mahdollinen seuraaja (q', w', α'), jolla

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

Toisin kuin äärellisten automaattien tapauksessa, *epädeterministiset pinoautomaatit ovat aidosti vahvempia kuin deterministiset*. Esimerkiksi kieli $\{ww^R \mid w \in \{a,b\}^*\}$ voidaan tunnistaa epädeterministisellä, mutta ei deterministisellä pinoautomaatilla. (Tod. siv.)

Yhteydetön kieli on *deterministinen*, jos se voidaan tunnistaa jollakin deterministisellä pinoautomaatilla. Deterministiset kielet voidaan jäsentää ajassa $O(n)$; yleiset yhteydettömät kielet vaativat tunnetuilla menetelmillä lähes ajan $O(n^3)$.

3.8 Yhteydettömien kielten rajoituksista

Yhteydettömille kielille on voimassa säännöllisten kielten pumppauslemman vastine. Nyt kuitenkin merkkijonoa on pumpattava samanaikaisesti kahdesta paikasta.

Lemma 3.9 (“uvwxy-lemma”) Olkoon L yhteydetön kieli. Tällöin on olemassa sellainen $n \geq 1$, että mikä tahansa $z \in L$, $|z| \geq n$, voidaan jakaa osiin $z = uvwxy$ siten, että

(i) $|vx| \geq 1$,

(ii) $|vwx| \leq n$,

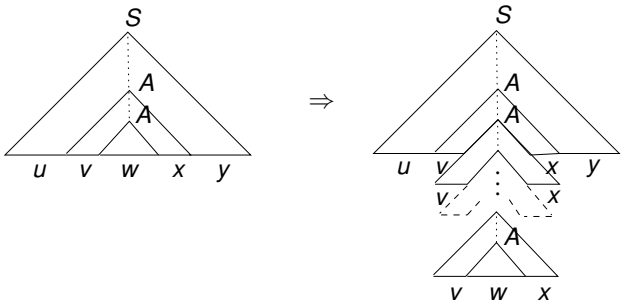
(iii) $uv^iwx^iy \in L$ kaikilla $i = 0, 1, 2, \ldots$.

Todistus.

Olkoon $G = (V, \Sigma, P, S)$ Chomskyn normaalimuotoinen kielioppi L:lle. Tällöin missä tahansa G:n jäsennyspuussa, jonka korkeus on h, on enintään 2^h lehteä. Toisin sanoen, minkä tahansa $z \in L$ jokaisessa jäsennyspuussa on polku, jonka pituus on vähintään $\log_2 |z|$.

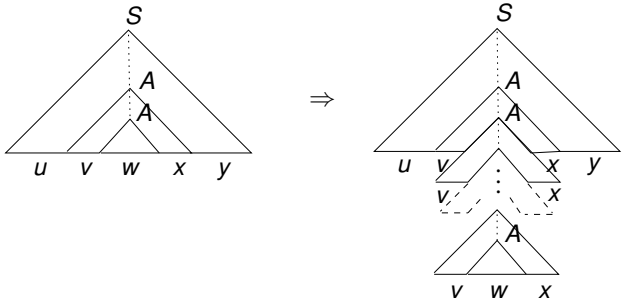
Olkoon $k = |V - \Sigma|$ kieliopin G välikkeiden määrä. Asetetaan $n = 2^{k+1}$. Tarkastellaan jotakin $z \in L$, $|z| \geq n$, ja sen jotakin jäsennyspuuta.

Edellisen nojalla puussa on polku, jonka pituus on $\geq k+1$; tällä polulla on siis jonkin välikkeen toistuttava — itse asiassa jo polun $k+2$ alimman solmun joukossa. Olkoon A jokin tällainen välike.



Merkkijono z voidaan nyt osittaa $z = uvwxy$, missä w on A:n alimmasta ilmentymästä tuotettu osajono ja vwx seuraavaksi ylemmästä A:n ilmentymästä tuotettu osajono; osajonot saadaan johdosta

$$S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uvwxy.$$



Koska siis $S \Rightarrow^* uAy$, $A \Rightarrow^* vAx$ ja $A \Rightarrow^* w$, osajonoja v ja x voidaan "pumpata" w:n ympärillä:

$$S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uv^2Ax^2y \Rightarrow^* \ldots \Rightarrow^* uv^iAx^iy \Rightarrow^* uv^iwx^iy.$$

Siten $uv^iwx^iy \in L$ kaikilla $i = 0, 1, 2, \ldots$.

Koska kielioppi G on Chomskyn normaalimuodossa ja $A \Rightarrow^* vAx$, on oltava $|vx| \geq 1$.

Koska edelleen välikkeen A valinnan perusteella sen toiseksi ylin ilmentymä on enintään korkeudella $k+1$ jäsennyspuun lehdistä, on tähän ilmentymään juurtuvan alipuun tuotokselle voimassa pituusraja $|vwx| \leq 2^{k+1} = n$. □

Esimerkki.

Tarkastellaan kieltä

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

Oletetaan, että L olisi yhteydetön; valitaan parametri n lemman mukaisesti ja tarkastellaan merkkijonoa $z = a^n b^n c^n \in L$.

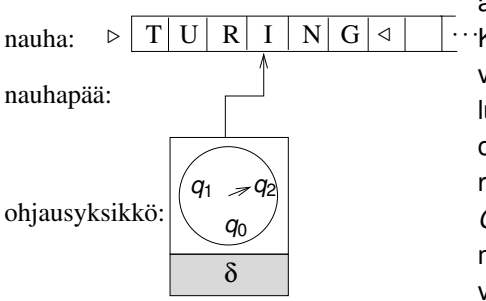
Lemman 3.9 mukaan z voidaan jakaa pumpattavaksi osiin

$$z = uvwxy, \qquad |vx| \geq 1, \qquad |vwx| \leq n.$$

Viimeisen ehdon takia merkkijono vx ei voi sisältää sekä a:ta, b:tä että c:tä. Merkkijonossa $uv^0wx^0y = uwy$ on siten ylijäämä jotakin merkkiä muihin merkkeihin nähden, eikä se voi olla kielen L määritelmässä vaadittua muotoa, vaikka lemman mukaan pitäisi olla $uwy \in L$.

4. TURINGIN KONEET

Alan Turing 1935–36.



Turingin kone on kuin äärellinen automaatti, jolla on käytössään nauha. Kone voi siirtää nauhapäätä vasemmalle tai oikealle; se voi myös lukea tai kirjoittaa nauhapään kohdalla olevan merkin. Nauha on oikealle rajaton.

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

Turingin koneen kanssa ekvivalentteja laskentamalleja:

- Gödelin–Kleenen rekursiivisesti määritellyt funktiot (1936),
- Churchin λ-kalkyyli (1936),
- Postin (1936) ja Markovin (1951) merkkijonomuunnossysteemit, kaikki nykyiset ohjelmointikielet.

Turingin koneet ≡ ohjelmointikieli.

Määritelmä 4.1 *Turingin kone* on seitsikko

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}),$$

missä:

- Q on koneen *tilojen* äärellinen joukko;
- Σ on koneen *syöteaakkosto*;
- $\Gamma \supseteq \Sigma$ on koneen *nauha-aakkosto* (ol. että $\triangleright, \triangleleft \notin \Gamma$);
- $\delta : (Q - \{q_{\text{acc}}, q_{\text{rej}}\}) \times (\Gamma \cup \{\triangleright, \triangleleft\}) \to Q \times (\Gamma \cup \{\triangleright, \triangleleft\}) \times \{L, R\}$ on koneen *siirtymäfunktio*;
- $q_0 \in Q$ on koneen *alkutila*;
- $q_{\text{acc}} \in Q$ on koneen *hyväksyvä* ja
- $q_{\text{rej}} \in Q$ sen *hylkäävä lopputila*.

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 $a =$ ‘$\triangleright$’ tai ‘$\triangleleft$’, ja siirtymäfunktion arvo on aina määrittelemätön, kun $q = q_{\text{acc}}$ tai $q = q_{\text{rej}}$. Joutuessaan jompaan kumpaan näistä tiloista kone pysähtyy heti.

Siis: siirtymäfunktion arvoilta

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

vaaditaan:

(i) jos $b = \triangleright$, niin $a = \triangleright$;

(ii) jos $a = \triangleright$, niin $b = \triangleright$ ja $\Delta = R$;

(iii) jos $b = \triangleleft$, niin $a = \triangleleft$ ja $\Delta = L$.

Koneen *tilanne* on nelikko

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

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

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 $a = \varepsilon$, jos nauhapää sijaitsee aivan nauhan alussa tai sen käytetyn osan lopussa. Ensimmäisessä tapauksessa ajatellaan, että kone "havaitsee" merkin '$\triangleright$' ja toisessa tapauksessa merkin '$\triangleleft$'.

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

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

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 $c \in \Gamma \cup \{\varepsilon\}$:

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

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

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

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

Tilanteet, jotka ovat muotoa (q_{acc}, w) tai (q_{rej}, 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

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

muuten M *hylkää* x:n.

Koneen M *tunnistama kieli* on:

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

Esimerkki 1.

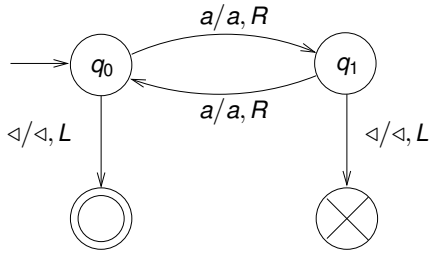
Kieli $\{a^{2k} \mid k \geq 0\}$ voidaan tunnistaa Turingin koneella

$$M = (\{q_0, q_1, q_{\text{acc}}, q_{\text{rej}}\}, \{a\}, \{a\}, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}),$$

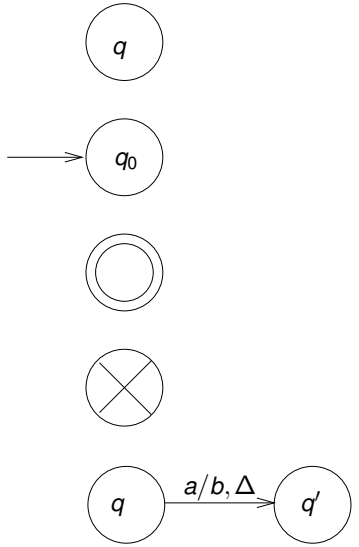
missä

$$\begin{aligned}
\delta(q_0, a) &= (q_1, a, R), \\
\delta(q_1, a) &= (q_0, a, R), \\
\delta(q_0, \triangleleft) &= (q_{\text{acc}}, \triangleleft, L), \\
\delta(q_1, \triangleleft) &= (q_{\text{rej}}, \triangleleft, L).
\end{aligned}$$

Kaavioesitys:



Kaavioesityksessä käytetyt merkinnät:



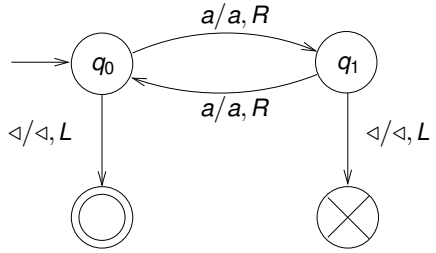
Tila q

Alkutila

Hyväksyvä lopputila (q_{acc})

Hylkäävä lopputila (q_{rej})

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



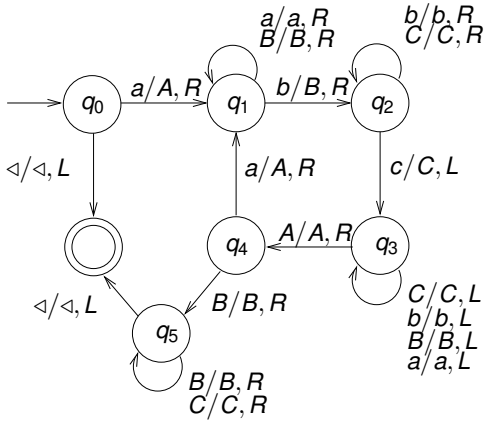
Koneen M laskenta esimerkiksi syötteellä aaa etenee seuraavasti:

$$\begin{aligned}
(q_0, \underline{a}aa) &\underset{M}{\vdash} (q_1, a\underline{a}a) \underset{M}{\vdash} (q_0, aa\underline{a}) \\
&\underset{M}{\vdash} (q_1, aaa\underline{\varepsilon}) \underset{M}{\vdash} (q_{\text{rej}}, aa\underline{a}).
\end{aligned}$$

Kone pysähtyy tilassa q_{rej}, joten $aaa \notin L(M)$.

Esimerkki 2.

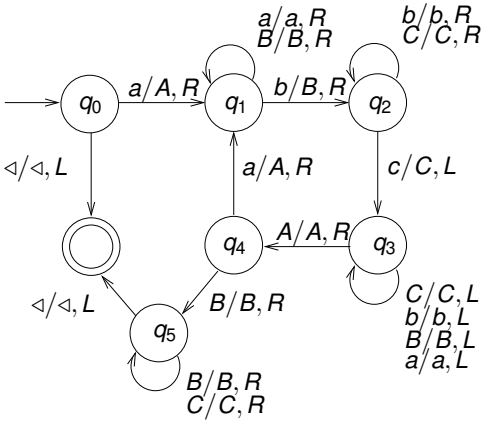
Kielen $\{a^k b^k c^k \mid k \geq 0\}$ tunnistava Turingin kone:



Selkeyden vuoksi ei koneen hylkäävää lopputilaa q_{rej} ole tässä esitetty eksplisiittisesti. Tulkinta on tällöin, että kaikki kaaviosta "puuttuvat" kaaret johtavat tähän tilaan.

Koneen laskenta syötteellä *aabbcc*:

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



4.2 Turingin koneiden laajennuksia

1. Moniuraiset koneet
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:

A	L	A	N	#	#	#	#		
M	A	T	H	I	S	O	N		⋯
T	U	R	I	N	G	#	#		

nauhapää:

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

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 #.

Formaalisti *k-urainen Turingin kone* on seitsikko

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}),$$

missä muut komponentit ovat kuten standardimallissa, paitsi siirtymäfunktio:

$$\delta : (Q - \{q_{\text{acc}}, q_{\text{rej}}\}) \times (\Gamma^k \cup \{\triangleright, \triangleleft\}) \to Q \times (\Gamma^k \cup \{\triangleright, \triangleleft\}) \times \{L, R\}.$$

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

Lause 4.1.

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

Todistus. Olkoon $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ k-urainen Turingin kone, joka tunnistaa kielen L. Vastaava standardimallinen kone $\widehat{M}$ muodostetaan seuraavasti:

$$\widehat{M} = (\hat{Q}, \Sigma, \hat{\Gamma}, \hat{\delta}, \hat{q}_0, q_{\text{acc}}, q_{\text{rej}}),$$

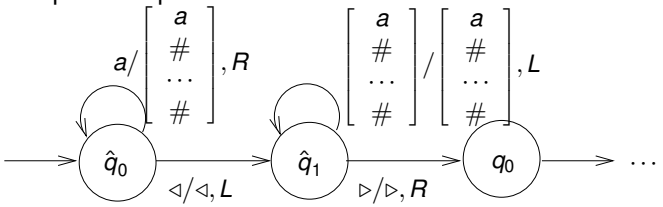
missä $\hat{Q} = Q \cup \{\hat{q}_0, \hat{q}_1, \hat{q}_2\}$, $\hat{\Gamma} = \Sigma \cup \Gamma^k$ ja kaikilla $q \in Q$ on

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

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

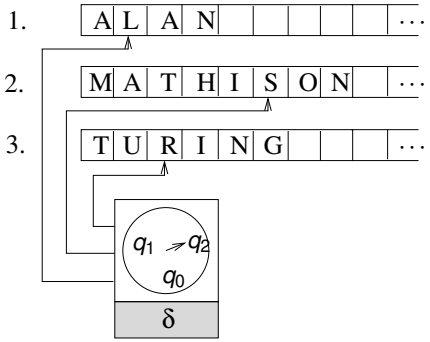
$$\begin{bmatrix} a_1 \\ \# \\ \vdots \\ \# \end{bmatrix} \begin{bmatrix} a_2 \\ \# \\ \vdots \\ \# \end{bmatrix} \cdots \begin{bmatrix} a_n \\ \# \\ \vdots \\ \# \end{bmatrix}.$$

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



□

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.

Formaalisti *k-nauhainen Turingin kone* on seitsikko

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}),$$

missä muut komponentit ovat kuten standardimallissa, paitsi siirtymäfunktio:

$$\delta : (Q - \{q_{\text{acc}}, q_{\text{rej}}\}) \times (\Gamma \cup \{\triangleright, \triangleleft\})^k \to Q \times ((\Gamma \cup \{\triangleright, \triangleleft\}) \times \{L, R\})^k.$$

Seuraajatilannerelaatio ym. peruskäsitteet määritellään pienin muutoksin entiseen tapaan.

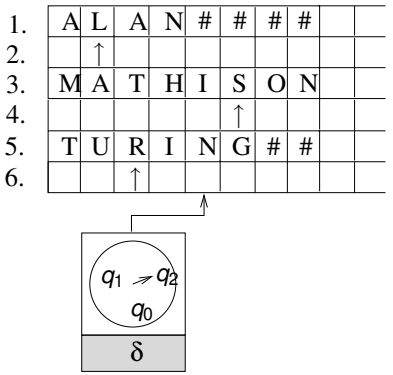
Lause 4.2.

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

Todistus (idea). Olkoon

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

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 ↑ 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, ja takaisin oikealta vasemmalle suuntautuvalla pyyhkäisyllä kirjoittaa $\uparrow$-osoittimien kohdalle asianmukaiset uudet merkit ja siirtää osoittimia. □

3. Epädeterministiset koneet

Formaalisti *epädeterministinen Turingin kone* on seitsikko

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}),$$

missä muut komponentit ovat kuten deterministisessä standardimallissa, paitsi siirtymäfunktio:

$$\delta : (Q - \{q_{\text{acc}}, q_{\text{rej}}\}) \times (\Gamma \cup \{\triangleright, \triangleleft\}) \to \mathcal{P}(Q \times (\Gamma \cup \{\triangleright, \triangleleft\}) \times \{L, R\}).$$

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 $(q', b, \Delta) \in \delta(q, a)$.

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:

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

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.

Yhdistettyjen lukujen "tunnistaminen" epädeterministisillä Turingin koneilla.

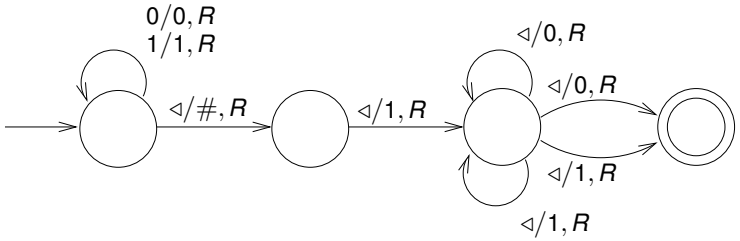
Ei-negatiivinen kokonaisluku n on *yhdistetty*, jos sillä on kokonaislukutekijät $p, q \geq 2$, joilla $pq = n$. Luku, joka ei ole yhdistetty, on *alkuluku*.

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.

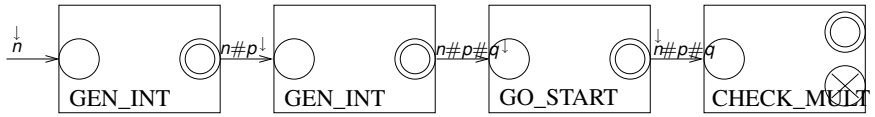
Olkoon edelleen GEN_INT seuraava mielivaltaisen ykköstä suuremman binääriluvun nauhan loppuun tuottava epädeterministinen Turingin kone:



Epädeterministinen Turingin kone TEST_COMPOSITE, joka tunnistaa kielen

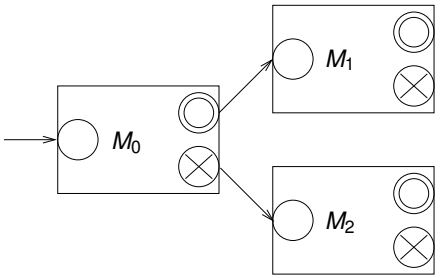
$$L(\text{TEST\_COMPOSITE}) = \{n \mid n \text{ on binäärimuotoinen yhdistetty luku}\}$$

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.

Huom. Yleinen kaaviomerkintä Turingin koneiden yhdistämiselle:



Lause 4.3

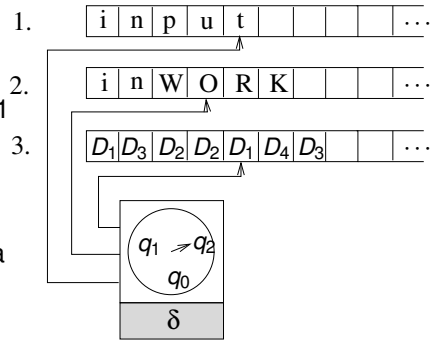
Jos formaali kieli L voidaan tunnistaa epädeterministisellä Turingin koneella, se voidaan tunnistaa myös standardimallisella deterministisellä Turingin koneella.

Todistus (idea). Olkoon

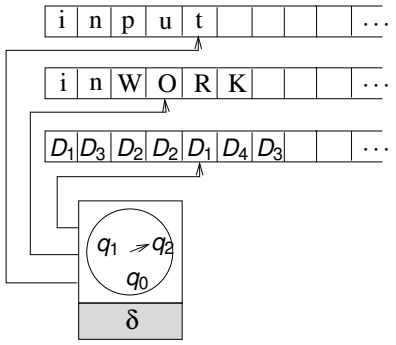
$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

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.

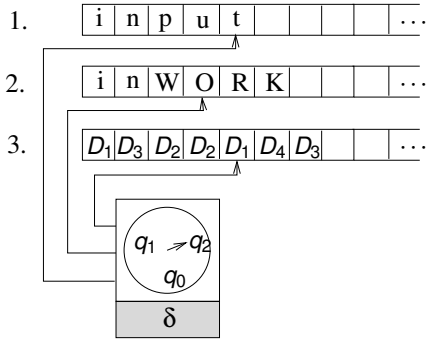
Yksityiskohtaisemmin:
Nauhalla 1 $\widehat{M}$ säilyttää kopiota syötejonosta ja nauhalla 2 se simuloi koneen M työnauhaa. Kunkin simuloitavan laskennan aluksi $\widehat{M}$ kopioi syötteen nauhalta 1 nauhalle 2 ja pyyhkii pois nauhalle 2 edellisen laskennan jäljiltä mahdollisesti jääneet merkit. Nauhalla 3 $\widehat{M}$ pitää kirjaa vuorossa olevan laskennan "järjestysnumerosta".

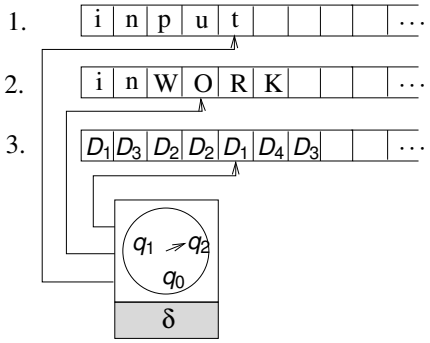


Tarkemmin sanoen, olkoon r suurin M:n siirtymäfunktion arvojoukon koko. Tällöin $\widehat{M}$:lla on erityiset nauhamerkit $D_1, \ldots, D_r$, joista koostuvia jonoja se generoi nauhalle 3 kanonisessa järjestyksessä: ε, D_1, D_2, $\ldots$, D_r, D_1D_1, D_1D_2, $\ldots$, D_1D_r, D_2D_1, $\ldots$. Kutakin 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. □

1.7 Numeroituvat ja ylinumeroituvat joukot

Määritelmä 1.10 Joukko X on *numeroituvasti ääretön*, jos on olemassa bijektio $f : \mathbb{N} \to X$. Joukko on *numeroituva*, jos se on äärellinen tai numeroituvasti ääretön. Joukko, joka ei ole numeroituva on *ylinumeroituva*.

Intuitiivisesti sanoen joukko X on numeroituva, jos sen alkiot voidaan järjestää ja indeksoida luonnollisilla luvuilla:

$$X = \{x_0, x_1, x_2, \ldots, x_{n-1}\},$$

jos X on n-alkioinen äärellinen joukko ja

$$X = \{x_0, x_1, x_2, \ldots\},$$

jos X on numeroituvasti ääretön.

Numeroituvan joukon kaikki osajoukot ovat myös numeroituvia (HT), mutta ylinumeroituvilla joukoilla on sekä numeroituvia että ylinumeroituvia osajoukkoja. Siten ylinumeroituvat joukot ovat jossain mielessä “isompia” kuin numeroituvat.

Lause 1.11 Minkä tahansa aakkoston Σ merkkijonojen joukko Σ^* on numeroituvasti ääretön.

Todistus. Muodostetaan bijektio $f : \mathbb{N} \to \Sigma^*$ seuraavasti. Olkoon $\Sigma = \{a_1, a_2, \ldots, a_n\}$. Kiinnitetään Σ:n merkeille jokin “aakkosjärjestys”; olkoon se $a_1 < a_2 < \cdots < a_n$.

Joukon Σ^* merkkijonot voidaan nyt luetella valitun aakkosjärjestyksen suhteen *kanonisessa* t. *leksikografisessa järjestyksessä* (engl. canonical t. lexicographic order) seuraavasti:

- ensin luetellaan 0:n mittaiset merkkijonot (= ε), sitten 1:n mittaiset (= $a_1, a_2, \ldots, a_n$), sitten 2:n mittaiset jne.;
- kunkin pituusryhmän sisällä merkkijonot luetellaan aakkosjärjestyksessä.

Bijektio f on siis:

$$\begin{array}{rcl}
0 & \mapsto & \varepsilon \\
1 & \mapsto & a_1 \\
2 & \mapsto & a_2 \\
\vdots & & \vdots \\
n & \mapsto & a_n \\
n+1 & \mapsto & a_1 a_1 \\
n+2 & \mapsto & a_1 a_2 \\
\vdots & & \vdots \\
2n & \mapsto & a_1 a_n
\end{array}
\qquad
\begin{array}{rcl}
2n+1 & \mapsto & a_2 a_1 \\
\vdots & & \vdots \\
3n & \mapsto & a_2 a_n \\
\vdots & & \vdots \\
n^2+n & \mapsto & a_n a_n \\
n^2+n+1 & \mapsto & a_1 a_1 a_1 \\
n^2+n+2 & \mapsto & a_1 a_1 a_2 \\
\vdots & & \vdots
\end{array}
\qquad \square$$

Itse asiassa millä tahansa ohjelmointikielellä kirjoitetut ohjelmat ovat kielen perusaakkoston (esim. C-kielessä ASCII-merkistön) merkkijonoja. Lauseen 1.11 mukaan minkä tahansa aakkoston merkkijonojen joukko on numeroituvasti ääretön, joten myös millä tahansa ohjelmointikielellä mahdollisten ohjelmien joukko on numeroituva.

Seuraavaksi todistetaan, että kaikkien formaalien kielten joukko on ylinumeroituva. Formaaleja kieliä on siis “enemmän” kuin mahdollisia tietokoneohjelmia, ja siksi *millään ohjelmointikielellä ei voida laatia tunnistusautomaatteja kaikille formaaleille kielille*. (Tai toisin sanoen: on olemassa “periaatteessa mahdollisia” I/O-kuvauksia, joita ei voida toteuttaa tietokoneella.)

Lause 1.12 Minkä tahansa aakkoston Σ kaikkien formaalien kielten perhe on ylinumeroituva.

Todistus (ns. Cantorin diagonaaliargumentti). Merkitään aakkoston Σ kaikkien formaalien kielten perhettä $\mathcal{P}(\Sigma^*) = \mathcal{A}$. Tehdään vastaoletus: oletetaan, että olisi olemassa kaikki Σ:n formaalit kielet kattava numerointi:

$$\mathcal{A} = \{A_0, A_1, A_2, \ldots\}.$$

Olkoot Σ^*:n merkkijonot kanonisessa järjestyksessä $x_0, x_1, x_2, \ldots$. Määritellään em. numerointeja käyttäen formaali kieli $\tilde{A}$:

$$\tilde{A} = \{x_i \in \Sigma^* \mid x_i \notin A_i\}.$$

Koska $\tilde{A} \in \mathcal{A}$ ja $\mathcal{A}$:n numerointi oletettiin kattavaksi, pitäisi olla $\tilde{A} = A_k$ jollakin $k \in \mathbb{N}$. Mutta tällöin $\tilde{A}$:n määritelmän mukaan

$$x_k \in \tilde{A} \Leftrightarrow x_k \notin A_k = \tilde{A}.$$

Saatu ristiriita osoittaa, että vastaoletus on väärä. □

$\tilde{A}$

$\searrow$	A_0	A_1	A_2	A_3	$\cdots$
x_0	1 $\not\emptyset$	0	0	1	$\cdots$
x_1	0	0 $\not 1$	0	0	$\cdots$
x_2	1	1	0 $\not 1$	1	$\cdots$
x_3	0	0	0	1 $\not\emptyset$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Kuvallisesti todistuksen idea voidaan esittää seuraavasti. Muodostetaan kielten $A_0, A_1, A_2, \ldots$ ja merkkijonojen $x_0, x_1, x_2, \ldots$ "insidenssimatriisi", jonka rivin i sarakkeessa j on arvo 1 jos $x_i \in A_j$ ja muuten 0. Tällöin kieli $\tilde{A}$ poikkeaa kustakin kielestä A_k matriisin "diagonaalilla":

1.8 *Ekskursio: Turingin pysähtymisongelma

Lauseiden 1.11 ja 1.12 mukaan on siis olemassa formaaleja kieliä (I/O-kuvauksia), joita ei voida toteuttaa esim. C-ohjelmilla. Entä jokin *konkreettinen esimerkki* tällaisesta?

Tunnetuin esimerkki on ns. *Turingin pysähtymisongelma*. (Alan Turing, 1936). C-ohjelmia käyttäen tulos voidaan muotoilla seuraavasti:

Väite. Ei ole olemassa C-funktiota `halt(p,x)`, joka saa syötteenään mielivaltaisen C-funktion tekstin `p` ja tälle tarkoitetun syötteen `x` ja tuottaa tuloksen 1, jos `p`:n suoritus pysähtyy syötteellä `x`, ja 0 jos `p`:n suoritus `x`:llä jää ikuiseen silmukkaan.

Todistus. Oletetaan väitteen vastaisesti, että tällainen funktio `halt` voitaisiin laatia. Muodostetaan tätä käyttäen toinen funktio `confuse` (ks. alla).

Merkitään funktion `confuse` ohjelmatekstiä `c`:llä ja tarkastellaan funktion `confuse` laskentaa tällä omalla kuvauksellaan.

```
void confuse(char *p){
   int halt(char *p, char *x){
   ... /* Funktion halt runko. */
   }
   if (halt(p,p) == 1) while (1);
}
```

Saadaan ristiriita: `confuse(c)`
pysähtyy
⇔
`halt(c,c) == 1`
⇔
`confuse(c)` ei pysähdy.

Ristiriidasta seuraa, että oletettua pysähtymistestausfunktiota `halt` ei voi olla olemassa. □

Samansukuisia ns. *ratkeamattomia ongelmia* on itse asiassa *paljon*. Asiaan palataan kurssin loppupuolella.

6. LASKETTAVUUSTEORIAA

Churchin–Turingin teesi: Mielivaltainen (riittävän vahva) laskulaite $\equiv$ Turingin kone.

Laskettavuusteoria: Tarkastellaan mitä Turingin koneilla voi ja erityisesti mitä *ei* voi laskea.

Tärkeä erottelu: Pysähtyvät ja ei-pysähtyvät Turingin koneet.

Määritelmä 6.1 Turingin kone

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

on *totaalinen*, jos se pysähtyy kaikilla syötteillä. Formaali kieli A on *rekursiivisesti numeroituva*, jos se voidaan tunnistaa jollakin Turingin koneella, ja *rekursiivinen*, jos se voidaan tunnistaa jollakin totaalisella Turingin koneella.

Vaihtoehtoinen termistö: Palautetaan mieliin päätösongelmien (binäärivasteisten I/O-kuvausten) ja formaalien kielten vastaavuus: päätösongelmaa Π vastaava formaali kieli A_Π koostuu niistä syötteistä x, joille ongelman Π vastaus on "kyllä" (so. toivottu vaste = 1).

Päätösongelma Π on *ratkeava*, jos sitä vastaava formaali kieli A_Π on rekursiivinen, ja *osittain ratkeava*, jos A_Π on rekursiivisesti numeroituva. Ongelma, joka ei ole ratkeava, on *ratkeamaton*. (*Huom.*: ratkeamaton ongelma voi siis olla osittain ratkeava.)

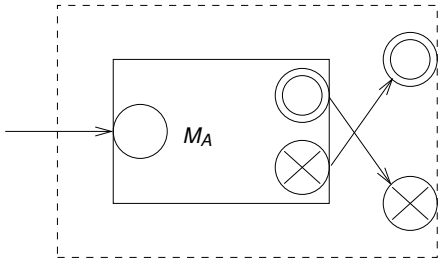
Toisin sanoen: päätösongelma on ratkeava, jos sillä on totaalinen, kaikilla syötteillä pysähtyvä ratkaisualgoritmi, ja osittain ratkeava, jos sillä on ratkaisualgoritmi joka "kyllä"-tapauksissa vastaa aina oikein, mutta "ei"-tapauksissa voi jäädä pysähtymättä.

6.2 Rekursiivisten ja rek. num. kielten perusominaisuuksia

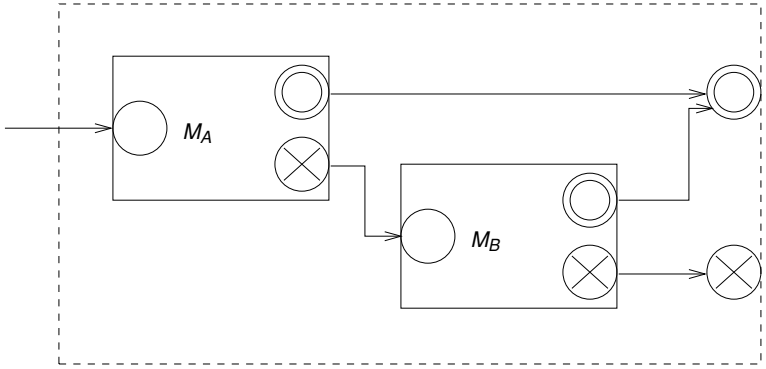
Lause 6.1 Olkoot $A, B \subseteq \Sigma^*$ rekursiivisia. Tällöin myös $\bar{A} = \Sigma^* - A$, $A \cup B$ ja $A \cap B$ ovat rekursiivisia.

Todistus.

(i) Olkoon M_A totaalinen Turingin kone, jolla $L(M_A) = A$. Kielen $\bar{A}$ tunnistava totaalinen Turingin kone saadaan vaihtamalla M_A:n hyväksyvä ja hylkäävä lopputila keskenään.



(ii) Olkoot M_A ja M_B totaaliset Turingin koneet, joilla $L(M_A) = A$, $L(M_B) = B$. Kielen $A \cup B$ tunnistava totaalinen Turingin kone M saadaan yhdistämällä M_A ja M_B toimimaan peräkkäin: jos M_A hyväksyy syötteen, myös M hyväksyy; jos M_A päätyy hylkäämiseen, M simuloi vielä M_B:tä.



(iii) $A \cap B = \overline{\bar{A} \cup \bar{B}}$.

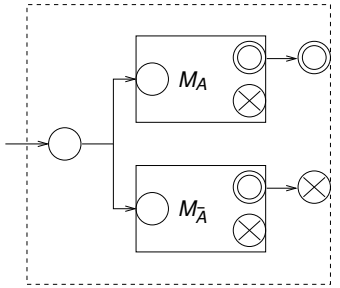
Lause 6.2 Olkoot $A, B \subseteq \Sigma^*$ rekursiivisesti numeroituvia. Tällöin myös $A \cup B$ ja $A \cap B$ ovat rekursiivisesti numeroituvia.

Todistus. $A \cap B$ kuten Lause 6.1 ja $A \cup B$ kuten Lause 6.3. (HT) □

Lause 6.3 Kieli $A \subseteq \Sigma^*$ on rekursiivinen, jos ja vain jos kielet A ja $\bar{A}$ ovat rekursiivisesti numeroituvia.

Todistus. Lauseen 6.1(i) nojalla jos A on rekursiivinen, myös $\bar{A}$ on rekursiivinen ja siis sekä A että $\bar{A}$ ovat rekursiivisesti numeroituvia. Osoitetaan, että jos A ja $\bar{A}$ ovat rekursiivisesti numeroituvia, A on rekursiivinen:

Olkoot M_A ja $M_{\bar{A}}$ Turingin koneet kielten A ja $\bar{A}$ tunnistamiseen. Kaikilla $x \in \Sigma^*$ *joko* M_A *tai* $M_{\bar{A}}$ *pysähtyy* ja hyväksyy x:n. Muodostetaan M_A ja $M_{\bar{A}}$ "rinnakkain" yhdistämällä totaalinen kaksinauhainen tunnistajakone M: M simuloi ykkösnauhallaan konetta M_A ja kakkosnauhallaan konetta $M_{\bar{A}}$.



Jos ykkössimulaatio pysähtyy hyväksyvään lopputilaan, M hyväksyy syötteen; jos taas kakkossimulaatio hyväksyy, M hylkää syötteen. □

Seuraus 6.4 Olkoon $A \subseteq \Sigma^*$ rekursiivisesti numeroituva kieli, joka ei ole rekursiivinen. Tällöin kieli $\bar{A}$ ei ole rekursiivisesti numeroituva. □

6.3 Turingin koneiden koodaus

Tarkastellaan standardimallisia Turingin koneita, joiden syöteaakkosto on $\Sigma = \{0,1\}$. Jokainen tällainen kone

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

voidaan esittää binäärijonona:

Oletetaan, että $Q = \{q_0, q_1, \ldots, q_n\}$, missä $q_{\text{acc}} = q_{n-1}$ ja $q_{\text{rej}} = q_n$; ja että $\Gamma \cup \{\triangleright, \triangleleft\} = \{a_0, a_1, \ldots, a_m\}$, missä $a_0 = 0$, $a_1 = 1$, $a_2 = \triangleright$ ja $a_3 = \triangleleft$. Merkitään lisäksi $\Delta_0 = L$ ja $\Delta_1 = R$.

Siirtymäfunktion δ arvojen koodaus: 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 on

$$\begin{aligned} c_M \quad = \quad & 111c_{00}11c_{01}11\ldots11c_{0m}11c_{10}11\ldots11c_{1m}11 \\ & \ldots11c_{n-2,0}11\ldots11c_{n-2,m}111. \end{aligned}$$

Kääntäen voidaan jokaiseen binäärijonoon c liittää jokin Turingin kone M_c. Binäärijonoihin, jotka eivät ole edellisen koodauksen mukaisia Turingin koneiden koodeja, liitetään jokin triviaali, kaikki syötteet hylkäävä kone M_{triv}.

Määritellään siis:

$$M_c = \begin{cases} \text{kone } M, \text{ jolla } c_M = c, \text{ jos } c \text{ on kelvollinen konekoodi} \\ \text{kone } M_{\text{triv}}, \text{ muuten.} \end{cases}$$

Saadaan luettelo kaikista aakkoston $\{0,1\}$ Turingin koneista, ja epäsuorasti myös kaikista aakkoston $\{0,1\}$ rekursiivisesti numeroituvista kielistä. Koneet ovat

$$M_\varepsilon, M_0, M_1, M_{00}, M_{01}, \ldots,$$

kielet vastaavasti

$$L(M_\varepsilon), L(M_0), L(M_1), L(M_{00}), L(M_{01}), \ldots$$

(indeksit kanonisessa järjestyksessä). Kukin kieli voi esiintyä luettelossa monta kertaa.

Eräs ei rekursiivisesti numeroituva kieli

Lemma 6.5 Kieli

$$D = \{c \in \{0,1\}^* \mid c \notin L(M_c)\}$$

ei ole rekursiivisesti numeroituva.

Todistus. Oletetaan, että olisi $D = L(M)$ jollakin standardimallisella Turingin koneella M. Olkoon d koneen M binäärikoodi, so. $D = L(M_d)$. Tällöin on

$$d \in D \quad \Leftrightarrow \quad d \notin L(M_d) = D.$$

Ristiriidasta seuraa, että kieli D ei voi olla rekursiivisesti numeroituva. □

Kieltä D vastaava päätösongelma: “Onko niin, ettei annetun koodin c esittämä Turingin kone hyväksy syötettä c?” Luontevampia esimerkkejä seuraa jatkossa.

Kielen D muodostaminen kuvallisesti: jos kielten $L(M_\varepsilon)$, $L(M_0)$, $L(M_1)$, ... karakteristiset funktiot esitetään taulukkona, niin kieli D poikkeaa kustakin kielestä taulukon diagonaalilla:

D ↘	$L(M_\varepsilon)$	$L(M_0)$	$L(M_1)$	$L(M_{00})$	$\cdots$
ε	$\not{0}^1$	0	0	0	$\cdots$
0	0	$\not{1}^0$	1	0	$\cdots$
1	0	0	$\not{1}^0$	1	$\cdots$
00	0	0	0	$\not{0}^1$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

6.4 Universaalit Turingin koneet

Aakkoston $\{0,1\}$ *universaalikieli* U määritellään:

$$U = \{c_M w \mid w \in L(M)\}.$$

Olkoon A jokin aakkoston $\{0,1\}$ rekursiivisesti numeroituva kieli, ja olkoon M kielen A tunnistava standardimallinen Turingin kone. Tällöin on

$$A = \{w \in \{0,1\}^* \mid c_M w \in U\}.$$

Myös kieli U on rekursiivisesti numeroituva. Kielen U tunnistavia Turingin koneita sanotaan *universaaleiksi Turingin koneiksi*.

Lause 6.6

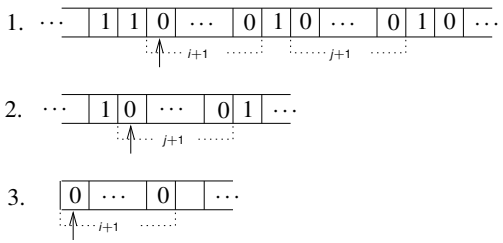
Kieli U on rekursiivisesti numeroituva.

Todistus. Kielen U tunnistava universaalikone M_U on helpointa kuvata kolmenauhaisena mallina. (Standardointi tavalliseen tapaan.) Laskennan aluksi tarkastettava syöte sijoitetaan koneen M_U ykkösnauhan alkuun.

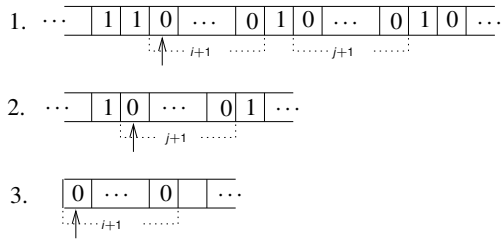
Tämän jälkeen kone toimii seuraavasti:

1. Aluksi M_U tarkastaa, että syöte on muotoa cw, missä c on kelvollinen Turingin koneen koodi. Jos syöte ei ole kelvollista muotoa, M_U hylkää sen; muuten se kopioi merkkijonon $w = a_1 a_2 \ldots a_k \in \{0,1\}^*$ kakkosnauhalle muodossa

$$00010^{a_1+1}10^{a_2+1}1 \ldots 10^{a_k+1}10000.$$



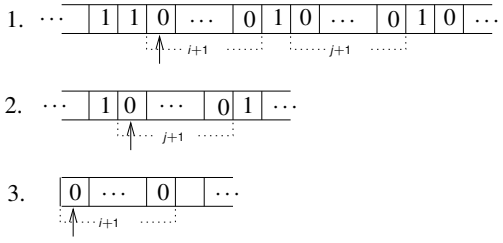
2. Jos syöte on muotoa cw, missä $c = c_M$ jollakin koneella M, M_U:n on selvitettävä, hyväksyisikö kone M syötteen w. Tässä tarkoituksessa M_U säilyttää ykkösnauhalla M:n kuvausta c, kakkosnauhalla simuloi M:n nauhaa, ja kolmosnauhalla säilyttää tietoa M:n simuloidusta tilasta muodossa $q_i \sim 0^{i+1}$ (aluksi siis M_U kirjoittaa kolmosnauhalle tilan q_0 koodin 0).



3. Alkutoimien jälkeen M_U toimii vaiheittain, simuloiden kussakin vaiheessa yhden koneen M siirtymän. Vaiheen aluksi M_U etsii ykkösnauhalta M:n kuvauksesta kohdan, joka vastaa M:n simuloitua tilaa q_i ja merkkiä a_j.

Olkoon ykkösnauhalla koodinkohta $0^{i+1}10^{j+1}10^{r+1}10^{s+1}10^{t+1}$.

Tällöin M_U korvaa kolmosnauhalla merkkijonon 0^{i+1} merkkijonolla 0^{r+1}, kakkosnauhalla merkkijonon 0^{j+1} merkkijonolla 0^{s+1}, ja siirtää kakkosnauhan nauhapäätä yhden simuloidun merkin vasemmalle, jos $t = 0$, ja oikealle, jos $t = 1$.



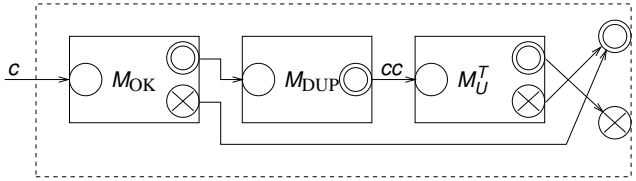
Jos ykkösnauhalla ei ole yhtään simuloituun tilaan q_i liittyvää koodia, simuloitu kone M on tullut hyväksyvään tai hylkäävään lopputilaan; tällöin $i = k+1$ tai $i = k+2$, missä q_k on viimeinen ykkösnauhalla kuvattu tila. Kone M_U siirtyy vastaavasti lopputilaan q_{acc} tai q_{rej}. □

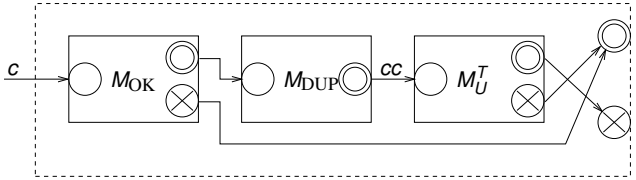
Lause 6.7

Kieli U ei ole rekursiivinen.

Todistus. Oletetaan, että kielellä U olisi totaalinen tunnistajakone M_U^T. Tällöin voitaisiin Lemman 6.5 kielelle D muodostaa totaalinen tunnistajakone M_D seuraavasti.

Olkoon M_{OK} totaalinen Turingin kone, joka testaa, onko syötteenä annettu merkkijono kelvollinen Turingin koneen koodi, ja olkoon M_{DUP} totaalinen Turingin kone, joka muuntaa syötejonon c muotoon cc. Kone M_D muodostetaan koneista M_U^T, M_{OK} ja M_{DUP} yhdistämällä seuraavan kaavion esittämällä tavalla:





Selvästi kone M_D on totaalinen, jos kone M_U^T on, ja

$$\begin{aligned} c \in L(M_D) &\Leftrightarrow c \notin L(M_{\mathrm{OK}}) \text{ tai } cc \notin L(M_U^T) \\ &\Leftrightarrow c \notin L(M_c) \\ &\Leftrightarrow c \in D. \end{aligned}$$

Mutta lemman 6.5 mukaan kieli D ei ole rekursiivinen; ristiriita. □.

Seuraus 6.8

Kieli

$$\tilde{U} = \{c_M w \mid w \notin L(M)\}$$

ei ole rekursiivisesti numeroituva.

Todistus. Kieli $\tilde{U}$ on oleellisesti sama kuin universaalikielen U komplementti $\bar{U}$; tarkasti ottaen on $\bar{U} = \tilde{U} \cup \mathrm{ERR}$, missä ERR on helposti tunnistettava rekursiivinen kieli

$$\begin{aligned} \mathrm{ERR} \quad = \quad & \{x \in \{0,1\}^* \mid x \text{ ei sisällä alkuosanaan} \\ & \text{kelvollista Turingin koneen koodia}\}. \end{aligned}$$

Jos siis kieli $\tilde{U}$ olisi rekursiivisesti numeroituva, olisi samoin myös kieli $\bar{U}$. Koska kieli U on rekursiivisesti numeroituva, seuraisi tästä, että U on peräti rekursiivinen. Mutta tämä on vastoin edellisen lauseen tulosta, mistä päätellään, että kieli $\tilde{U}$ ei voi olla rekursiivisesti numeroituva. □

6.5 Turingin koneiden pysähtymisongelma

Lause 6.9 Kieli

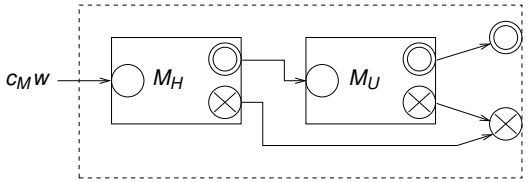
$$H = \{c_M w \mid M \text{ pysähtyy syötteellä } w\}$$

on rekursiivisesti numeroituva, mutta ei rekursiivinen.

Todistus. Todetaan ensin, että kieli H on rekursiivisesti numeroituva. Lauseen 6.6 todistuksessa esitetystä universaalikoneesta M_U on helppo muokata kone, joka syötteellä $c_M w$ simuloi koneen M laskentaa syötteellä w ja pysähtyy hyväksyvään lopputilaan, jos ja vain jos simuloitu laskenta ylipäätään pysähtyy.

Osoitetaan sitten, että kieli H ei ole rekursiivinen. Oletetaan nimittäin, että olisi $H = L(M_H)$ jollakin totaalisella Turingin koneella M_H. Oletetaan lisäksi, että kone M_H pysähtyessään jättää nauhalle alkuperäisen syötteensä, mahdollisesti tyhjämerkeillä jatkettuna. Olkoon M_U lauseen 6.6 todistuksessa konstruoitu universaalikone.

Kielelle U voitaisiin nyt muodostaa totaalinen tunnistaja yhdistämällä koneet M_H ja M_U seuraavasti:




```
def U(m,w):
  if not M_H(m,w):
    reject
  return M_U(m,w)
```

Lauseen 6.7 mukaan tällaista kielen U tunnistajakonetta ei kuitenkaan voi olla olemassa. Saatu ristiriita osoittaa, että H ei voi olla rekursiivinen. □

Seuraus 6.10 Kieli

$$\tilde{H} = \{c_M w \mid M \text{ ei pysähdy syötteellä } x\}$$

ei ole rekursiivisesti numeroituva. □

6.7 Ricen lause

Ricen lauseen mukaan *kaikki* Turingin koneiden tunnistamia kieliä, t. niiden laskemia I/O-kuvauksia koskevat epätriviaalit kysymykset ovat ratkeamattomia.

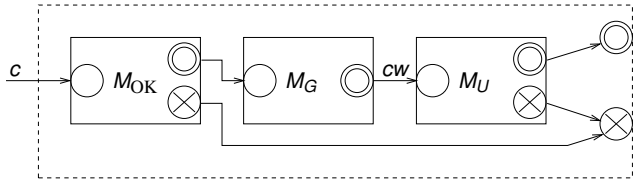
Johdantona lauseen todistukseen tarkastellaan ensin yhtä sen erikoistapausta, Turingin koneiden tunnistamien kielten *epätyhjyysongelmaa*: “Hyväksyykö annettu Turingin kone yhtään syötemerkkijonoa?” Ongelman esitys formaalina kielenä on

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

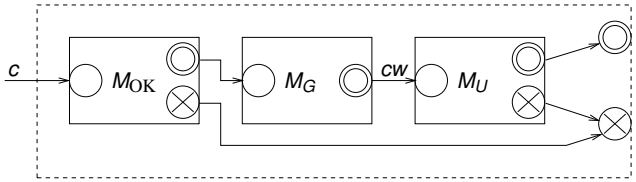
Lause 6.11 Kieli NE on rekursiivisesti numeroituva, mutta ei rekursiivinen.

Todistus. Todetaan ensin, että kieli NE on rekursiivisesti numeroituva muodostamalla sille tunnistajakone M_{NE}. Kone M_{NE} on helpointa suunnitella epädeterministisenä.

M_{OK} testaa, onko annettu syöte kelvollinen Turingin koneen koodi, ja M_G kirjoittaa epädeterministisesti nauhalla jo olevan merkkijonon perään mielivaltaisen binäärijonon w. Muodostetaan M_{NE} yhdistämällä koneet M_{OK}, M_G ja universaalikone M_U seuraavasti:




```
def NE(m):
  if not M_ok(m):
    reject
  w = choose_string_nondeterministically()
  return M_U(m,w)
```

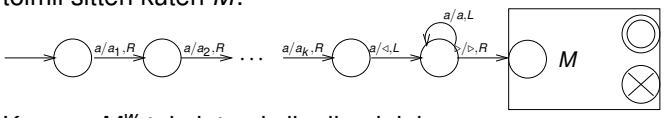



Selvästi on:

$$\begin{aligned}
& c \in L(M_{\text{NE}}) \\
\Leftrightarrow\ & c \text{ on kelvollinen Tk-koodi ja } \exists\ w \text{ s.e. } cw \in U \\
\Leftrightarrow\ & c \text{ on kelvollinen Tk-koodi ja } \exists\ w \text{ s.e. } w \in L(M_c) \\
\Leftrightarrow\ & L(M_c) \neq \emptyset.
\end{aligned}$$

Osoitetaan, ettei kieli NE ole rekursiivinen. Oletetaan, että kielellä NE olisi totaalinen tunnistajakone M^T_{NE}, ja muodostetaan sitä käyttäen totaalinen tunnistajakone M^T_U kielelle U. (Ristiriita.)

Konstruktio perustuu syötteiden koodaamiseen Turingin koneiden "ohjelmavakioiksi". Olkoon M mielivaltainen Turingin kone, jonka toimintaa syötteellä $w = a_1 a_2 \ldots a_k$ halutaan tutkia. Merkitään M^w:llä konetta, joka aina korvaa "todellisen" syötteensä merkkijonolla w ja toimii sitten kuten M:

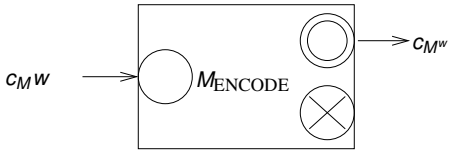


Koneen M^w toiminta ei siis riipu lainkaan sen todellisesta syötteestä, vaan se joko hyväksyy tai hylkää kaikki merkkijonot, sen mukaan miten M suhtautuu w:hen:

```
def Mw(x):
  return M_U(M,w)
```

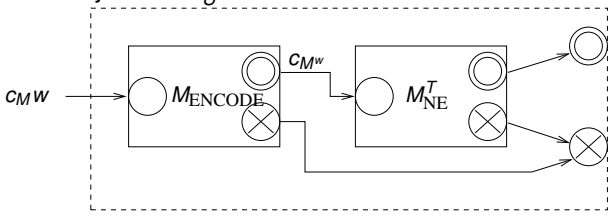
$$L(M^w) = \begin{cases} \{0,1\}^*, & \text{jos } w \in L(M); \\ \emptyset, & \text{jos } w \notin L(M). \end{cases}$$

Olkoon sitten M_{ENCODE} Turingin kone, joka saa syötteenään mielivaltaisen Turingin koneen M koodista c_M ja binäärijonosta w muodostuvan jonon $c_M w$ ja jättää tuloksenaan nauhalle edellä kuvatun koneen M^w koodin c_{M^w}:



(Jos syöte ei ole muotoa cw, missä c on kelvollinen Turingin koneen koodi, kone M_{ENCODE} päätyy hylkäävään lopputilaan.) Kone M_{ENCODE} operoi siis Turingin koneiden *koodeilla*. Annetun koneen M koodiin se lisää siirtymäviisikoita ("konekäskyjä") ja muuttaa tilojen numerointia siten, että koodi tulee koneen M sijaan esittämään konetta M^w.

Universaalikielelle U voitaisiin nyt koneet M_{ENCODE} ja hypoteettinen M^T_{NE} seuraavalla tavalla yhdistämällä muodostaa totaalinen tunnistajakone M^T_U:




```
def MTU(m,w):
  def Mw(x):
    return M_U(m,w)
  return MTNE(Mw)
```

Kone M^T_U on totaalinen, jos M^T_{NE} on, ja $L(M^T_U) = U$, koska:

$$c_M w \in L(M^T_U) \Leftrightarrow c_{M^w} \in L(M^T_{\mathrm{NE}}) = \mathrm{NE} \Leftrightarrow L(M^w) \neq \emptyset \Leftrightarrow w \in L(M).$$

Mutta kieli U ei ole rekursiivinen, joten tällainen totaalinen tunnistajakone M^T_U ei ole mahdollinen. Saadusta ristiriidasta päätellään, että myöskään kielellä NE ei voi olla totaalista tunnistajaa M^T_{NE}. □

Ricen lause

Turingin koneiden *semanttinen ominaisuus* $\mathcal{S}$ on mikä tahansa kokoelma rekursiivisesti numeroituvia aakkoston $\{0,1\}$ kieliä; koneella M *on ominaisuus* $\mathcal{S}$, jos $L(M) \in \mathcal{S}$. *Triviaalit ominaisuudet* ovat $\mathcal{S} = \emptyset$ (ominaisuus, jota ei ole millään koneella) ja $\mathcal{S} = RE$ (ominaisuus, joka on kaikilla koneilla).

Ominaisuus $\mathcal{S}$ on *ratkeava*, jos joukko

$$\mathrm{codes}(\mathcal{S}) = \{c \mid L(M_c) \in \mathcal{S}\}$$

on rekursiivinen. Toisin sanoen: ominaisuus on ratkeava, jos annetusta Turingin koneen koodista voidaan algoritmisesti päätellä, onko koneella kysytty semanttinen ominaisuus.

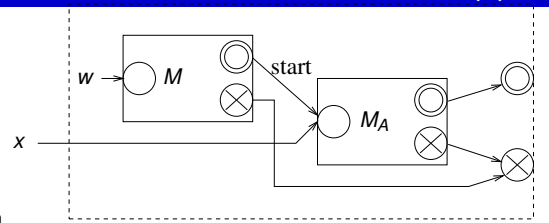
Lause 6.12 [Rice 1953] Kaikki Turingin koneiden epätriviaalit semanttiset ominaisuudet ovat ratkeamattomia.

Todistus.

Olkoon $\mathcal{S}$ mielivaltainen epätriviaali semanttinen ominaisuus. Voidaan olettaa, että $\emptyset \notin \mathcal{S}$: toisin sanoen, että tyhjän joukon tunnistavilla Turingin koneilla ei ole tarkasteltavaa ominaisuutta. Jos nimittäin $\emptyset \in \mathcal{S}$, voidaan ensin osoittaa, että ominaisuus $\bar{\mathcal{S}} = \mathrm{RE} - \mathcal{S}$ on ratkeamaton, ja päätellä edelleen tästä että myös ominaisuus $\mathcal{S}$ on ratkeamaton. (Koska $\mathrm{codes}(\bar{\mathcal{S}}) = \overline{\mathrm{codes}(\mathcal{S})}$.)

Koska $\mathcal{S}$ on epätriviaali, on olemassa jokin Turingin kone M_A, jolla on ominaisuus $\mathcal{S}$ — jolla siis $L(M_A) \neq \emptyset \in \mathcal{S}$.

Olkoon nyt M_{ENCODE} Turingin kone, joka muodostaa syötteenä annetusta merkkijonosta $c_M w$ seuraavanlaisen Turingin koneen M^w koodin.



Jos syöte on väärää muotoa, M_{ENCODE} hylkää sen.

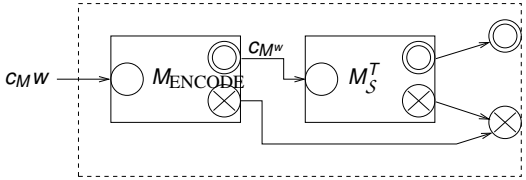
Syötteellä x kone M^w toimii ensin kuten M syötteellä w. Jos M hyväksyy w:n, M^w toimii kuten kone M_A syötteellä x. Jos M hylkää w:n, myös M^w hylkää x:n. Kone M^w tunnistaa siis kielen

```
def Mw(x):
  if M(w):
    return M_A(x)
  else:
    reject
```

$$L(M^w) = \begin{cases} L(M_A), & \text{jos } w \in L(M); \\ \emptyset, & \text{jos } w \notin L(M). \end{cases}$$

Koska oletuksen mukaan $L(M_A) \in \mathcal{S}$ ja $\emptyset \notin \mathcal{S}$, on koneella M^w ominaisuus $\mathcal{S}$, jos ja vain jos $w \in L(M)$.

Oletetaan sitten, että ominaisuus $\mathcal{S}$ olisi ratkeava, so. että kielellä $\mathrm{codes}(\mathcal{S})$ olisi totaalinen tunnistajakone $M_{\mathcal{S}}^T$. Tällöin saataisiin edellisen todistuksen tapaan totaalinen tunnistajakone kielelle U yhdistämällä koneet M_{ENCODE} ja $M_{\mathcal{S}}^T$ seuraavasti:



Selvästi kone M_U^T on totaalinen, jos $M_{\mathcal{S}}^T$ on, ja

```
def MTU(M,w):
  def Mw(x):
    if M(w):
      return M_A(x)
    else:
      reject
  return MTS(Mw)
```

$$c_M w \in L(M_U^T) \Leftrightarrow c_{M^w} \in L(M_{\mathcal{S}}^T) = \mathrm{codes}(\mathcal{S}) \Leftrightarrow L(M^w) \in \mathcal{S} \Leftrightarrow w \in L(M).$$

Koska kieli U ei ole rekursiivinen, tämä on mahdotonta, mistä päätellään, ettei ominaisuus $\mathcal{S}$ voi olla ratkeava. □

6.8 Muita ratkeamattomuustuloksia

Lause 6.13 (Predikaattikalkyylin ratkeamattomuus; Church/Turing 1936)

Ei ole olemassa algoritmia, joka ratkaisisi, onko annettu ensimmäisen kertaluvun predikaattikalkyylin kaava ϕ validi (“loogisesti tosi”, todistuva predikaattikalkyylin aksioomista). □

Lause 6.14 (“Hilbertin 10. ongelma”; Matijasevitsh/Davis/Robinson/Putnam 1953–70)

Ei ole olemassa algoritmia, joka ratkaisisi, onko annetulla kokonaislukukertoimisella polynomilla $P(x_1, \ldots, x_n)$ kokonaislukunollakohtia (so. jonoja $(m_1, \ldots, m_n) \in \mathbf{Z}^n$, joilla $P(m_1, \ldots, m_n) = 0$). Ongelma on ratkematon jo, kun $n = 15$ tai $\deg(P) = 4$. □

Eräiden kielioppiongelmien ratkeavuus, kun annettuna on kieliopit G ja G' Chomskyn hierarkian tietyltä tasolta i ja merkkijono w. Taulukossa $R \sim$ "ratkeava", $E \sim$ "ei ratkeava", $T \sim$ "aina totta".

	Taso i:			
Ongelma: onko	3	2	1	0
$w \in L(G)$?	R	R	R	E
$L(G) = \emptyset$?	R	R	E	E
$L(G) = \Sigma^*$?	R	E	E	E
$L(G) = L(G')$?	R	E	E	E
$L(G) \subseteq L(G')$?	R	E	E	E
$L(G) \cap L(G') = \emptyset$?	R	E	E	E
$L(G)$ säännollinen?	T	E	E	E
$L(G) \cap L(G')$ tyyppiä i?	T	E	T	T
$\overline{L(G)}$ tyyppiä i?	T	E	T	E

6.9 Rekursiiviset funktiot

Turingin koneen $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ *laskema osittaiskuvaus* (t. *-funktio*)

$$f_M : \Sigma^* \to \Gamma^*$$

määritellään:

$$f_M(x) = \begin{cases} u, \text{ jos } (q_0, \underline{x}) \vdash_M^* (q, u\underline{a}v) \text{ jollakin } q \in \{q_{\text{acc}}, q_{\text{rej}}\}, av \in \Gamma^*; \\ \text{määrittelemätön, muuten.} \end{cases}$$

Osittaisfunktio $f : \Sigma^* \to A$ on *osittaisrekursiivinen* jos se voidaan laskea jollakin Turingin koneella ja *(kokonais-)rekursiivinen*, jos se voidaan laskea jollakin totaalisella Turingin koneella. Ekvivalentisti voitaisiin määritellä, että osittaisrekursiivifunktio f on rekursiivinen, jos sen arvo $f(x)$ on määritelty kaikilla x.

Lause 6.15

(i) Kieli $A \subseteq \Sigma^*$ on rekursiivinen, jos ja vain jos sen karakteristinen funktio

$$\chi_A : \Sigma^* \to \{0,1\}, \qquad \chi_A(x) = \begin{cases} 1, & \text{jos } x \in A; \\ 0, & \text{jos } x \notin A \end{cases}$$

on rekursiivinen funktio.

(ii) Kieli $A \subseteq \Sigma^*$ on rekursiivisesti numeroituva, jos ja vain jos on $A = \emptyset$ tai on olemassa rekursiivinen funktio $g : \{0,1\}^* \to \Sigma^*$, jolla

$$A = \{g(x) \mid x \in \{0,1\}^*\}.$$

Todistus. HT. □

5. RAJOITTAMATTOMAT KIELIOPIT

Määritelmä 5.1 *Rajoittamaton kielioppi* t. *yleinen merkkijonomuunnossysteemi* 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 ($V^+ = V^* - \{\varepsilon\}$);
- $S \in N$ 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 $\gamma \Rightarrow \gamma'$.

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 $\gamma \Rightarrow^* \gamma'$.

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-yhteydettömälle kielelle $\{a^k b^k c^k \mid k \geq 0\}$.

S	$\to$	$LT \mid \varepsilon$			
T	$\to$	$ABCT \mid ABC$	aA	$\to$	aa
BA	$\to$	AB	aB	$\to$	ab
CB	$\to$	BC	bB	$\to$	bb
CA	$\to$	AC	bC	$\to$	bc
LA	$\to$	a	cC	$\to$	cc.

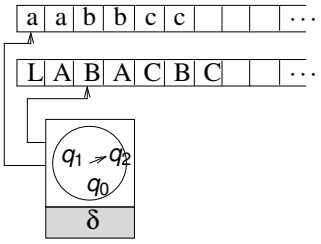
Esimerkiksi lauseen *aabbcc* johto:

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

Lause 5.1

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 epädeterminstinen 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 5.2

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

Todistus. Olkoon $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ 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

$$[q_0 x] \underset{G_M}{\Rightarrow}^* [uq_{\text{acc}}v]$$

joillakin $u, v \in \Sigma^*$.

Kaikkiaan kielioppiin G_M tulee kolme ryhmää produktioita:

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 $[uq_{\text{acc}}v]$, jos ja vain jos M hyväksyy x:n.

3. Produktiot, joilla muotoa $[uq_{\text{acc}}v]$ oleva merkkijono muutetaan tyhjäksi merkkijonoksi.

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

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

Määritellään siis $G = (V, \Sigma, P, S)$, 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:

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

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

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

3. Lopputilanteen siivous:

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

□

Yhteysherkät kieliopit

Rajoittamaton kielioppi on *yhteysherkkä*, jos sen kaikki produktiot ovat muotoa $\omega \to \omega'$, missä $|\omega'| \geq |\omega|$, tai mahdollisesti $S \to \varepsilon$, missä S on lähtösymboli.

Lisäksi vaaditaan, että jos kieliopissa on produktio $S \to \varepsilon$, niin lähtösymboli S ei esiinny minkään produktion oikealla puolella.

Formaali kieli L on *yhteysherkkä*, jos se voidaan tuottaa jollakin yhteysherkällä kieliopilla.

Normaalimuoto: Jokainen yhteysherkkä kieli voidaan tuottaa kieliopilla, jonka produktiot ovat muotoa $S \to \varepsilon$ ja $\alpha A \beta \to \alpha\omega\beta$, missä A on välike ja $\omega \neq \varepsilon$. (Säännön $A \to \omega$ sovellus "kontekstissa" α_β.)

Lause 5.3

Formaali kieli L on yhteysherkkä, 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 $\delta(q,\triangleleft) = (q',b,\Delta)$ olevia siirtymiä, missä $b \neq$ '$\triangleleft$'. □

Lauseen 5.3 koneita sanotaan *lineaarisesti rajoitetuiksi automaateiksi*.

Avoin ongelma ("LBA ?= DLBA"): onko epädeterminismi lauseessa 5.3 välttämätöntä?

Chomskyn hierarkia

Kielioppien, niillä tuotettavien kielten ja vastaavien tunnistusautomaattien ryhmittely:
Luokka 0: rajoittamattomat kieliopit / rekursiivisesti numeroituvat kielet / Turingin koneet.
Luokka 1: yhteysherkät kieliopit / yhteysherkät kielet / lineaarisesti rajoitetut automaatit.
Luokka 2: yhteydettömät kieliopit / yhteydettömät kielet / pinoautomaatit.
Luokka 3: oikealle ja vasemmalle lineaariset (säännölliset) kieliopit / säännölliset kielet / äärelliset automaatit.