Luku 3

Säännölliset kielet ja äärelliset automaatit

- Ohjelmointikielten muuttujat, vakiot jne. ovat tietynmuotoisia, ohjelmointikielen *syntaksin* ("oikeankirjoitusopin") mukaisia. Esim. 0.123 on luku, mutta 0.1.2.3 ei ole. Miten voidaan tunnistaa täsmälleen oikeanmuotoiset merkkijonot?

- Ongelma voidaan ratkaista *säännöllisten kielten* ja *äärellisten automaattien* avulla

- UNIXissa on käsky `grep`, jolla voidaan etsiä tekstistä hahmoja. Esim. `grep [A-Z][a-z]*[0-9] teksti` etsii tiedostosta `teksti` rivit, joilla on määrittelyn muotoisia sanoja: ensin suuri kirjain, sitten mielivaltainen määrä pieniä kirjaimia ja lopuksi numero. `grep` etsii nimenomaan säännöllisiä lausekkei-

ta, joihin tutustumme tässä luvussa, ja onkin lyhenne sanoista "Global search for Regular Expression and Print"

3.1 Säännölliset lausekkeet ja kielet

- Esim. Hyväksy merkkijonot, joissa esiintyy sana "kissa". Ts. merkkijonot ovat muotoa

  [0 tai useampia kirjaimia]*kissa*[0 tai useampia kirjaimia]

- Esim. Etsi tekstitiedostosta osoitteita, jotka ovat muotoa
  [$x$katu tai $x$tie][*numero*] [mahd. rapun kirjain]
  [mahd. asunnonnumero][postinumero][kunnan nimi]

- Kuinka esittää kompaktisti sallitut merkkijonot (ts. kuvata *kieli*, jonka tunnistusohjelma hyväksyy)?

- Määritellään avuksi kolme operaatiota kielten yhdistämiseen: Olkoot A ja B aakkoston Σ kieliä. Tällöin

  – A:n ja B:n *yhdiste* on kieli

  $$A \cup B = \{x \in \Sigma^* \mid x \in A \text{ tai } x \in B\}$$

  – A:n ja B:n *tulo* on kieli

  $$AB = \{xy \in \Sigma^* \mid x \in A,\ y \in B\}$$

  – A:n *potenssit* A^k, $k \geq 0$, määritellään iteratiivisesti:

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

  – A:n *sulkeuma* on kieli

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

- *Määritelmä:* Aakkoston Σ *säännölliset lausekkeet* (engl. regular expression) määritellään induktiivisesti säännöillä:
  (i) $\boldsymbol{\emptyset}$ ja $\boldsymbol{\epsilon}$ ovat Σ:n säännöllisiä lausekkeita;
  (ii) $\boldsymbol{a}$ on Σ:n säännöllinen lauseke kaikilla $a \in \Sigma$;
  (iii) jos r ja s ovat Σ:n säännöllisiä lausekkeita,
  niin $(r \cup s)$, (rs) ja r^* ovat Σ:n säännöllisiä lausekkeita;
  (iv) muita Σ:n säännöllisiä lausekkeita ei ole

- Kukin Σ:n säännöllinen lauseke r *kuvaa* kielen $L(r)$:
  (i) $L(\boldsymbol{\emptyset}) = \emptyset$;
  (ii) $L(\boldsymbol{\epsilon}) = \{\epsilon\}$;
  (iii) $L(\boldsymbol{a}) = \{a\}$ kaikilla $a \in \Sigma$;
  (iv) $L((r \cup s)) = L(r) \cup L(s)$;
  (v) $L((rs)) = L(r)L(s)$;
  (vi) $L(r^*) = (L(r))^*$

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

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

  Lausekkeiden kuvaamat kielet:

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

- Sulkumerkkien vähentämissääntöjä:
  - Operaattoreiden prioriteetti:

$$\boldsymbol{*} \quad \succ \quad \boldsymbol{\cdot} \quad \succ \quad \cup$$

- 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:

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

- *Määritelmä:* Kieli on *säännöllinen*, jos se voidaan kuvata säännöllisellä lausekkeella

- Esim. Olkoon aakkosto $\Sigma = \{a, b, c, ...,\}$. Hyväksytään merkkijonot, jotka ovat muotoa

$$l^* \text{kissa} l^*,$$

  missä l on lyhenne lausekkeelle $l = (\text{a} \cup \text{b} \cup ... \cup \text{ö})$ (ts. $l^* \in \Sigma^*$)

- Esim. C-kielen etumerkittömät liukuluvut (float, double, long double) määritellään seuraavasti:

  - (kokonaisosa).(desimaaliosa) (e tai E) [+ tai −] (eksponetti) [suffiksi]
  - kokonaisosa ja desimaaliosa koostuvat digiteistä
  - joko kokonaisosa tai desimaaliosa voi puuttua (mutta eivät molemmat)
  - joko (i) desimaalipiste tai (ii) (e tai E) ja eksponentti voivat puuttua (mutta eivät molemmat)
  - suffiksi: F tai f: float, L tai l: long double, muuten double

- Etumerkittömät liukuluvut tunnistava kieli voidaan määritellä säännöllisellä lausekkeella (ilman suffikseja):
  number $= (d^+.d^* \cup .d^+)(\epsilon \cup ((e \cup E)(+ \cup - \cup \epsilon)d^+)) \cup$
  $d^+(e \cup E)(+ \cup - \cup \epsilon)d^+$

- Kieleen kuuluvat esim. seuraavat merkkijonot: 12., .12, 1.2, 1.2E3, 1.2e3, 1.2E-3,1E2, 1e23

3.1.1 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 $L(r) = L(s)$
- Lisäksi merkitään $r \subseteq s$ tarkoittamaan $L(r) \subseteq L(s)$
- Lausekkeen sieventäminen = "yksinkertaisimman" ekvivalentin lausekkeen määrittäminen
- Huom: Usein merkitään $r^+ = rr^* = r^*r$

3.1.2 Sievennyssääntöjä

$$\begin{aligned}r \cup r &= r \text{ (mutta } rr \neq r, \text{ kun } r \neq \emptyset, \epsilon) \\ r \cup (s \cup t) &= (r \cup s) \cup t \\ r(st) &= (rs)t \\ r \cup s &= s \cup r \\ r(s \cup t) &= rs \cup rt \\ (r \cup s)t &= rt \cup st \\ \emptyset^* &= \epsilon \\ \emptyset r &= \emptyset \text{ (mutta } \emptyset \cup r = r) \\ \epsilon r &= r \text{ (mutta } \epsilon \cup r \neq r, \text{ kun } r \neq \epsilon) \\ r^* &= r^*r \cup \epsilon = r^+ \cup \epsilon \\ r^* &= (r \cup \epsilon)^* \\ (r^*)^* &= r^*\end{aligned}$$

- Lisäksi pätee päättelysääntö:

  Jos $r = rs \cup t$, niin $r = ts^*$, kun $\epsilon \notin L(s)$
- $L(r) = L(s) \Leftrightarrow L(r) \subseteq L(s) \wedge L(s) \subseteq L(r)$ eli $r = s \Leftrightarrow r \subseteq s \wedge s \subseteq r$

- Esim. yllä:
  1. $(a \cup b) \subseteq (a^*b^*) \Rightarrow (a \cup b)^* \subseteq (a^*b^*)^*$
  2. $((a^*b^*)^*) \subseteq a^*b^* \cup (a \cup b)^*ba(a \cup b)^*$):
     - jos muotoa a^*b^*, niin selvä
     - muuten sisältää osajonon ba
  3. $a^*b^* \cup (a \cup b)^*ba(a \cup b)^*) \subseteq (a \cup b)^*$, sillä $(a \cup b)^*$ kuvaa kaikki Σ:n merkkijonot
- Voidaan todistaa (todistus sivuutetaan): Jos L ja M ovat säännöllisiä kieliä, myös
  1. $L \cap M$
  2. $\overline{L} = \Sigma^* \setminus L$
  3. $L^R = \{w^R | w \in L\}$

  ovat säännöllisiä

3.1.3 Äärelliset automaatit

- Ongelma: Kahviautomaatti, joka ei anna vaihtorahaa, hyväksyy vain 50 sentin ja yhden euron kolikoita ja minimimaksu on 2 euroa. Millaisia syötejonoja kahviautomaatti hyväksyy?

- Kelvollisia syötejonoja ovat esim. seuraavat (yksikkönä snt):
  50 + 50 + 50 + 50
  100 + 100
  50 + 100 +100
  100 + 50 + 50 + 100

- Ts. kahviautomaatti hyväksyy syötejonot, jotka ovat muotoa

  ```
  1 euro +  1 euro +
  [0 tai useampia 50 sentin tai 1 euron kolikoita]
  ```

  tai

  ```
  1 euro +  50 senttiä +
  [1 tai useampia 50 sentin tai 1 euron kolikoita]
  ```

  tai

  ```
  50 senttiä + 1 euro +
  [1 tai useampia 50 sentin tai 1 euron kolikoita]
  ```

  tai

  ```
  50 senttiä +  50 senttiä + 1 euro +
  [0 tai useampia 50 sentin tai 1 euron kolikoita]
  ```

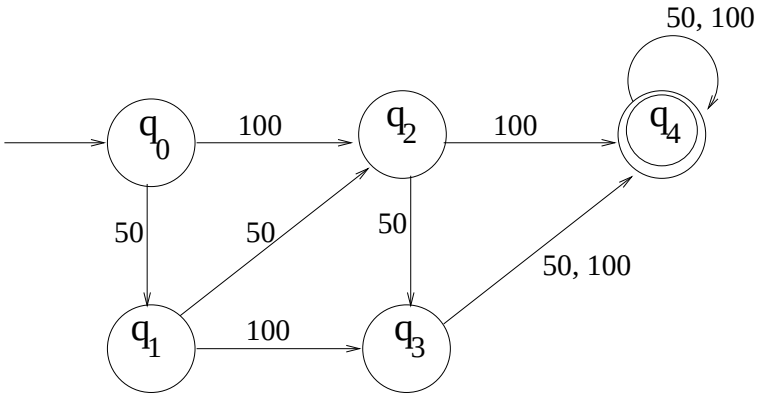
  tai

  ```
  50 senttiä + 50 senttiä + 50 senttiä +
  [1 tai useampia 50 sentin tai 1 euron kolikoita]
  ```

- Kahviautomaatin toiminta voidaan kuvata *äärellisenä automaattina*

- Automaatin syötteitä ovat 50 sentin ja 1 euron kolikot ja automaatti hyväksyy ”syötejonon”, jos siihen sisältyvien rahojen summa on vähintään 2 euro

- Automaatti voidaan esittää *tilasiirtymäkaaviona*



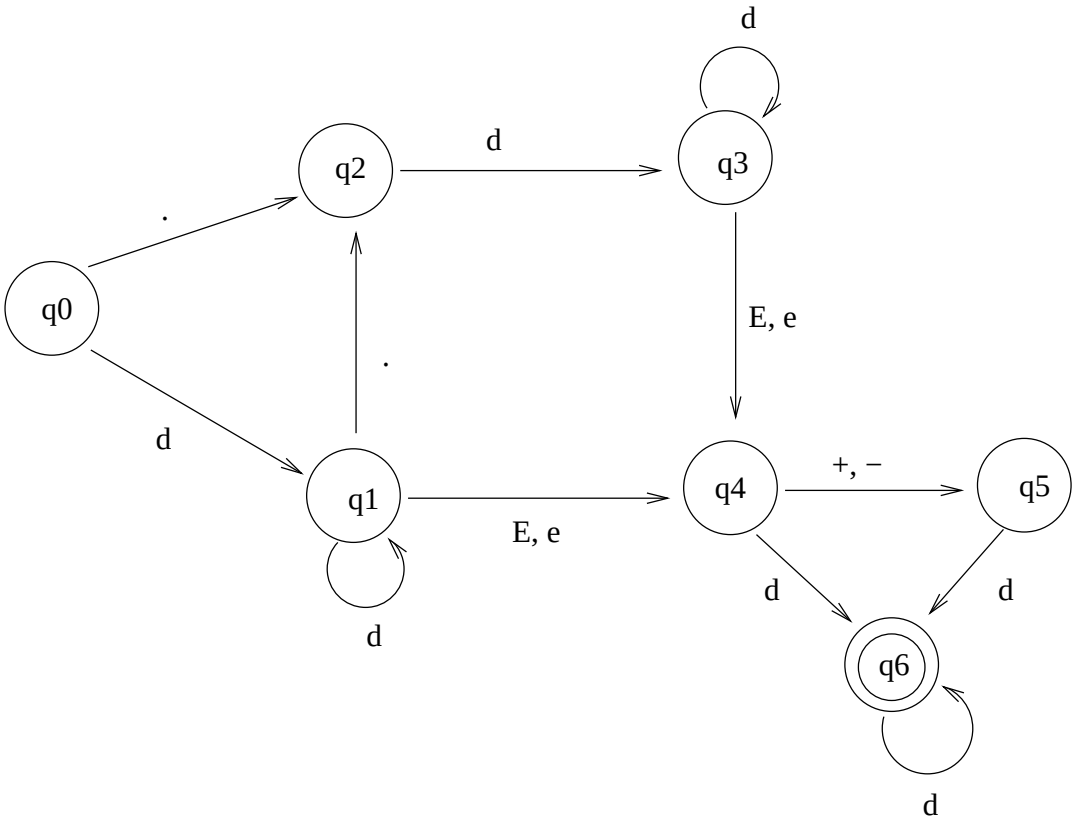
3.1.4 Äärellisen automaatin esitys

- tilasiirtymäkaavio

- tilasiirtymätaulukko

	50 snt	1 euro
$\rightarrow q_0$	q_1	q_2
q_1	q_2	q_3
q_2	q_3	q_4
q_3	q_4	q_4
$\leftarrow q_4$	q_4	q_4

- Esimerkki: C-kielen mukaisen etumerkittömän liukuluvun tunnistava automaatti:



– Tilasiirtymätaulukkona:

		d	.	E, e	+, −
→	q_0	q_1	q_2		
	q_1	q_1	q_3	q_4	
	q_2	q_3			
←	q_3	q_3		q_4	
	q_4	q_6			q_5
	q_5	q_6			
←	q_6	q_6			

Tässä d $= \{0, 1, \ldots, 9\}$. Taulukon puuttuvat kohdet on virhetila "Error", jota käytännössä jätetään merkitsemättä selkeyden takia

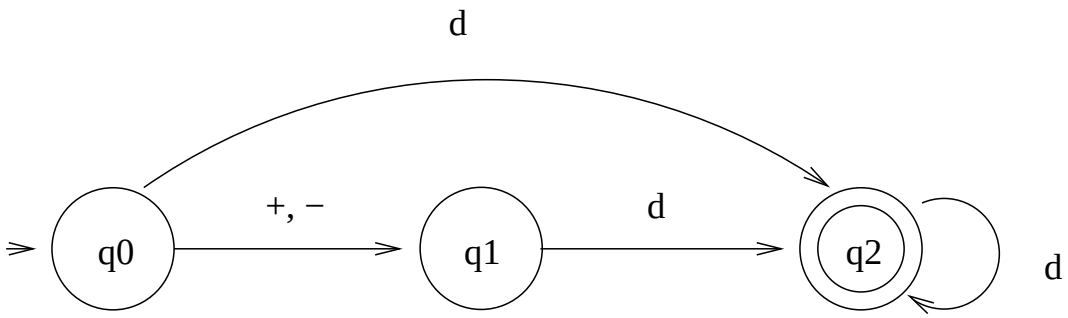
– Ohjelmana:

```
int IsDigit(char c); /* palauttaa 1, jos c on numero, 0 muuten */
```

```
int q = 0
char c while ( (c = fgetc(stdin))!=EOF )
{
   switch ( q )
   {
      case 0: if (IsDigit(c) ) q = 1;
              else if (c=='.') q = 2 else q = 99;
              break;
      case 1: if (IsDigit(c) ) q = 1;
              else if (c=='.') q=3;
              else if (c=='e' || c=='E') q=4; else q=99;
              break;
      case 2: if (IsDigit(c)) q=3; else q=99;
              break;
      case 3: if (IsDigit(c)) q=3;
              else if (c=='e' || c=='E') q = 4 else q = 99;
              break;
      case 4: if (IsDigit(c)) q=6;
              else if (c=='+' || c=='-') q = 5 else q = 99;
              break;
      case 5: if (IsDigit(c)) q=6; else q = 99;
              break;
      case 6: if (IsDigit(c)) q=6; else q = 99;
              break;
      case 99: break;
   }
}
if ( q == 3 || q == 6 ) printf("luku OK!");
else printf("Virheellinen luku");
```

- Äärellisen automaatin pohjalta laadittuun ohjelmaan voidaan liittää myös semanttisia toimintoja

- Esim. Etumerkillisen kokonaisluvun tunnistaminen



- Vastaava ohjelma, joka lisäksi evaluoi luvun arvon:

```
int IsDigit(char c); /* palauttaa 1, jos c on numero, 0 muuten */

int q = 0
char c int sign = 1; int val = 0; while ( (c = fgetc(stdin))!=EOF )
{
   switch ( q )
   {
      case 0: if (c=='+' || c=='-') {
                q=1;
                if (c=='-') sign = -1;
                }
                else if (IsDigit(c) {
                q=2;
                val = c -' 0';
                }
                break;
      case 1: if (IsDigit(c) {
                q=2;
                val = c -' 0';
                }
                else q=99;
                break;
      case 2: if (IsDigit(c)) {
                q=2;
                val = 10 * val + (c -' 0');
                }
                else q=99;
                break;
```

```
        case 99: break;
    }
}
if ( q == 2) printf("luvun arvo on %d", sgn * val);
else printf("Virheellinen luku");
```