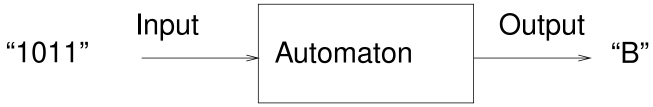
1.6 Aakkostot, merkkijonot ja kielet

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* $\{$A,B, ... ,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 $\frown$. Esim.

- KALA$\frown$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: $x\frown y$ 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 $w^R = a\frown u^R$.

T–79.1001/1002 Tietojenkäsittelyteorian perusteet

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

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

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

T–79.1001/1002 Tietojenkäsittelyteorian perusteet

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:

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

T–79.1001/1002 Tietojenkäsittelyteorian perusteet

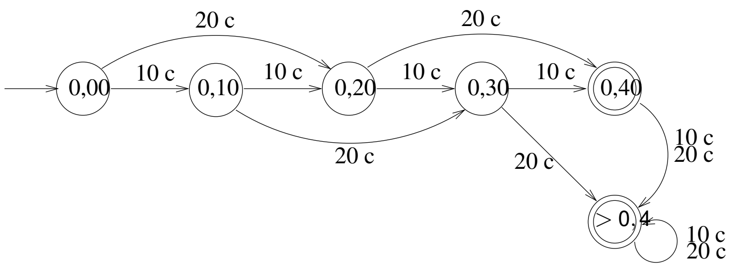
ÄÄRELLISET AUTOMAATIT JA SÄÄNNÖLLISET KIELET

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, ...

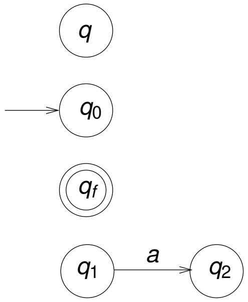
Esimerkki 1: Kahviautomaatti.



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:



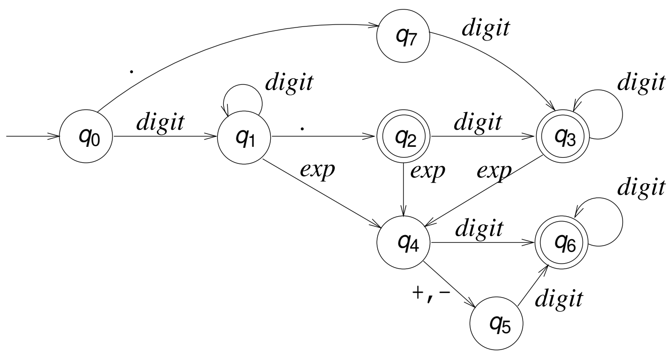
Automaatin tila nimeltä q

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:

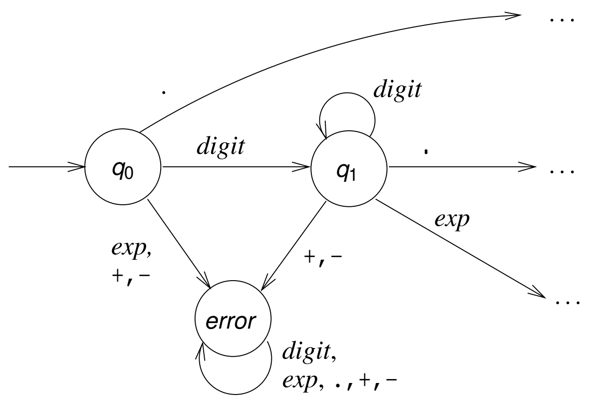
		digit	.	*exp*	+	−
→	q_0	q_1	q_7			
	q_1	q_1	q_2	q_4		
←	q_2	q_3		q_4		
←	q_3	q_3		q_4		
	q_4	q_6			q_5	q_5
	q_5	q_6				
←	q_6	q_6				
	q_7	q_3				

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:

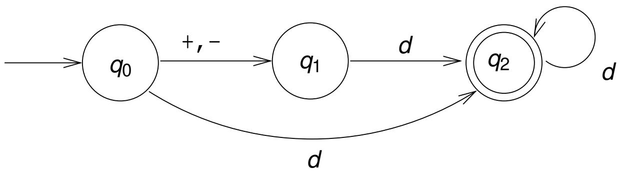
```
#include <stdio.h>
#include <ctype.h>
main() {
   int q, c;
   q = 0;
   while ((c = getchar()) != '\n')
      switch (q) {
      case 0:
         if (isdigit(c)) q = 1;
         else if (c == '.') q = 7;
         else q = 99;
         break;
      case 1:
         if (isdigit(c)) q = 1;
         else if (c == '.') q = 2;
         else if (c == 'E' || c == 'e') q = 4;
         else q = 99;
         break;
```



```
         ...
         case 99:
            break;
         }
      if (q == 2 || q == 3 || q == 6)
         printf("SYÖTE ON REAALILUKU.\n");
      else
         printf("SYÖTE EI OLE REAALILUKU.\n");
}
```


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:

```
#include <stdio.h>
#include <ctype.h>

main()
{
   int q, c;
   q = 0;
   while ((c = getchar()) != '\n') {
      switch (q) {
      case 0:
         if (c == '+' || c == '-') q = 1;
         else if ('0' <= c && c <= '7') q = 2;
         else q = 99;
         break;
```

```
        case 1:
          if ('0' <= c && c <= '7') q = 2;
          else q = 99;
          break;
        case 2:
          if ('0' <= c && c <= '7') q = 2;
          else q = 99;
          break;
        case 99:
          break;
        }
      }
      if (q == 2)
        printf("SYÖTE OK.\n");
      else
        printf("VIRHEELLINEN LUKU.\n");
    }
```


Täydennys syöteluvun arvon laskevilla operaatioilla ("luvun muuttaminen kymmenjärjestelmään"):

```
#include <stdio.h>

int main(void)  {
  int q, c;
  int sgn, val;        /* SEM: sgn = etumerkki  */
  sgn = 1; val = 0;    /* SEM: val = itseisarvo */
  q = 0;
```



```
  while ((c = getchar()) != '\n') {
    switch (q) {
    case 0:
      if (c == '+') q = 1;
      else if (c == '-') {
        sgn = -1;                       /* SEM */
        q = 1;
      }
      else if ('0' <= c && c <= '7') {
        val = c - '0';                  /* SEM */
        q = 2;
      }
      else q = 99;
      break;
```



```
    case 1:
      if ('0' <= c && c <= '7') {
        val = c - '0';                          /* SEM */
        q = 2;
      }
      else q = 99;
      break;
    case 2:
      if ('0' <= c && c <= '7') {
        val = 8 * val + (c - '0');              /* SEM */
        q = 2;
      }
      else q = 99;
      break;
    case 99:
      break;
    }
```

```
  }
  if (q == 2)
  { printf("LUVUN ARVO ON %d.\n", sgn*val);/*SEM*/
    exit(0); }
  else
  { printf("VIRHEELLINEN SYÖTE.\n"); /* SEM */
    exit(1); }
}
```