1.6 Aakkostot, merkkijonot ja kielet

Automaattiteoria $\sim$ diskreetin signaalinkäsittelyn perusmallit ja -menetelmät

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

$$\text{“1011”} \xrightarrow{\text{Syöte}} \boxed{\text{Automaatti}} \xrightarrow{\text{Tulos}} \text{“B”}$$

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:
(i) syötteet ovat äärellisiä, diskreettejä *merkkijonoja*
(ii) tulokset ovat muotoa “hyväksy”/”hylkää” ($\sim$ “syöte OK”/”syöte ei kelpaa”)

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

Peruskäsitteitä ja merkintöjä

Aakkosto (engl. alphabet, vocabulary): mikä tahansa 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.

Erikoistapaus: *tyhjä merkkijono* (engl. empty string). Tyhjässä merkkijonossa ei ole yhtään merkkiä, mutta havaittavuuden parantamiseksi sen paikka usein osoitetaan erikoismerkillä ε.

Merkkijonon x *pituutta*, so. siihen sisältyvien merkkien määrää, merkitään $|x|$:llä. Esim.: $|01001| = |\text{XYZZY}| = 5$, $|0000| = |\text{TKTP}| = 4$, $|\varepsilon| = 0$.

Merkkijonojen välinen perusoperaatio on *katenaatio* eli jonojen peräkkäin kirjoittaminen. Katenaation operaatiomerkkinä käytetään joskus selkeyden lisäämiseksi symbolia $\hat{}$. Esim.:
(i) KALA^KUKKO = KALAKUKKO;
(ii) jos $x = 00$ ja $y = 11$, niin $xy = 0011$ ja $yx = 1100$;
(iii) kaikilla x on $x\varepsilon = \varepsilon x = x$;
(iv) kaikilla x, y, z on $(xy)z = x(yz)$;
(v) 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. recognizes) 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 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: $x\hat{}y$ tai vain xy.

Merkkijonon pituus: $|x|$. *Esimerkkejä:*
(i) $|abc| = 3$;
(ii) 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:*
(i) $a^n = \underbrace{aa \ldots a}_{n \text{ kpl}}$;
(ii) $|a^i b^j c^k| = i + j + k$.

Merkkijonon x toisto k kertaa: x^k. *Esimerkkejä:*
(i) $(ab)^2 = abab$;
(ii) $|x^k| = k|x|$.

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:
(i) $\varepsilon^R = \varepsilon$;
(ii) jos $w = ua$, $u \in \Sigma^*$, $a \in \Sigma$, niin $w^R = a\hat{}u^R$.

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

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

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

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

Todistus. Induktio merkkijonon y pituuden suhteen.

(i) Perustapaus $y = \varepsilon$.

$$(x\varepsilon)^R = x^R = \varepsilon^R x^R.$$

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

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

1.7 Numeroituvat ja ylinumeroituvat joukot

Määritelmä 1.10 Joukko X on *numeroituvasti ääretön*, jos on olemassa bijektio $f : \mathbb{N} \to X$. Ääretön joukko, joka ei ole numeroituva on *ylinumeroituva*. Sanotaan yksinkertaisuuden vuoksi, että myös äärelliset joukot ovat numeroituvia.

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

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

On helppo osoittaa (HT), että numeroituvan joukon kaikki osajoukot ovat myös numeroituvia. 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:

(i) ensin luetellaan 0:n mittaiset merkkijonot $(= \varepsilon)$, sitten 1:n mittaiset $(= a_1, a_2, \ldots, a_n)$, sitten 2:n mittaiset jne.;

(ii) kunkin pituusryhmän sisällä merkkijonot luetellaan aakkosjärjestyksessä.

Bijektio f on siis:

$$
\begin{array}{rcl}
0 & \mapsto & \varepsilon \\
1 & \mapsto & a_1 \\
2 & \mapsto & a_2 \\
\vdots & & \vdots \\
n & \mapsto & a_n \\
n+1 & \mapsto & a_1a_1 \\
n+2 & \mapsto & a_1a_2 \\
\vdots & & \vdots \\
2n & \mapsto & a_1a_n \\
2n+1 & \mapsto & a_2a_1 \\
\vdots & & \vdots \\
3n & \mapsto & a_2a_n \\
\vdots & & \vdots \\
n^2+n & \mapsto & a_na_n \\
n^2+n+1 & \mapsto & a_1a_1a_1 \\
n^2+n+2 & \mapsto & a_1a_1a_2 \\
\vdots & & \vdots
\end{array}
\quad \square
$$

Mielenkiintoinen huomio on, että millä tahansa ohjelmointikielellä kirjoitetut ohjelmat ovat oikeastaan vain 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.

Seuraavan lauseen mukaan kuitenkin 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 $\mathcal{P}(\Sigma^*) = \mathcal{A}$. Oletetaan, että todistettavan väitteen vastaisesti olisi olemassa kaikki Σ:n formaalit kielet kattava numerointi:

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

Olkoot Σ^*:n merkkijonot kanonisessa järjestyksessä lueteltuina $x_0, x_1, x_2, \ldots$. 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 olisi $\tilde{A}$:n määritelmän mukaan

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

Saadun ristiriidan takia oletus, että joukko $\mathcal{A}$ on numeroituva, ei voi pitää paikkaansa. $\square$

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":

$$
\begin{array}{c|ccccc}
\tilde{A} \searrow & A_0 & A_1 & A_2 & A_3 & \cdots \\
\hline
x_0 & \overset{1}{\cancel{0}} & 0 & 0 & 1 & \cdots \\
x_1 & 0 & \overset{0}{\cancel{1}} & 0 & 0 & \cdots \\
x_2 & 1 & 1 & \overset{0}{\cancel{1}} & 1 & \cdots \\
x_3 & 0 & 0 & 0 & \overset{1}{\cancel{0}} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

***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`:

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

Merkitään edellä kuvattua funktion `confuse` ohjelmatekstiä `c`:llä ja tarkastellaan funktion `confuse` laskentaa tällä omalla kuvauksellaan. Saadaan ristiriita:

$$\begin{aligned} \mathtt{confuse(c)}\text{ pysähtyy } &\Leftrightarrow \mathtt{halt(c,c) == 1} \\ &\Leftrightarrow \mathtt{confuse(c)}\text{ ei pysähdy.} \end{aligned}$$

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

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