Tik-79.148 **Kevät 1999**
Tietojenkäsittelyteorian perusteet
Laskuharjoitus 5
Ratkaisut

1. (2.3.7a / 2.5.1a)

   Konstruktion perusajatuksena on käydä systemaattisesti läpi kaikki mahdolliset reitit alkutilasta kaikkiin mahdollisiin lopputiloihin. Aluksi kaikki tilat numeroidaan yksikäsitteisesti, ja sen jälkeen muodostetaan reitti käyttäen rekursiivisia kaavoja.

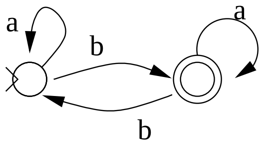
   Rekursiiviset säännöt ovat:

$$R(i,j,1) = \begin{cases} \{\sigma \in \Sigma \,|\, \delta(q_i,\sigma) = q_j\} & i \neq j \\ \{e\} \cup \{\sigma \in \Sigma \,|\, \delta(q_i,\sigma) = q_j\} & i = j \end{cases}$$
$$R(i,j,k+1) = R(i,j,k) \cup R(i,k,k)R(k,k,k)^*R(k,j,k)$$

   Merkintä $R(i,j,k)$ tarkoittaa kaikkia niitä reittejä tilasta q_i tilaan q_j, jotka eivät käy välillä tilassa q_k tai sitä suuremmassa tilassa.

   Ensimmäinen säännöistä sanoo, että mikäli välillä ei saa käydä missään tilassa, ainoat reitit tilojen q_i ja q_j välillä ovat suorat transitiot.

   Säännöistä jälkimmäinen tarkoittaa, että reitit tilasta q_i tilaan q_j käymättä välillä tilassa q_{k+1}, tai sitä suuremmassa tilassa, voidaan jakaan kahteen ryhmään: niihin, joissa ei käydä edes tilassa q_k, ja niihin, joissa käydään tilassa q_k.



$$
\begin{aligned}
L(M) &= R(1,2,3) \\
R(1,2,3) &= R(1,2,2) \cup R(1,2,2)R(2,2,2)^*R(2,2,2) \\
R(1,2,2) &= R(1,2,1) \cup R(1,1,1)R(1,1,1)^*R(1,2,1) \\
R(2,2,2) &= R(2,2,1) \cup R(2,1,1)R(1,1,1)^*R(1,2,1)
\end{aligned}
$$

$$
\begin{aligned}
R(1,2,1) &= R(2,1,1) = b \\
R(1,1,1) &= R(2,2,1) = e \cup a
\end{aligned}
$$

Lopuksi sijoitetetaan saadut lausekkeet aikaisempiin yhtälöihin, ja sievennetään tulokset:

$$
\begin{aligned}
R(1,2,2) &= b \cup (e \cup a)(e \cup a)^*b = a^*b \\
R(2,2,2) &= (e \cup a) \cup b(e \cup a)^*b = e \cup a \cup ba^*b \\
R(1,2,3) &= a^*b \cup a^*b(e \cup a \cup ba^*b)^*(e \cup a \cup ba^*b) \\
&= a^*b(a \cup ba^*b)^*
\end{aligned}
$$

2. (2.4.5 / 2.6.5)

   Tilakoneiden pumppauslemman mukaan jokaiselle äärettömälle säännölliselle kielelle L löytyy jokin $k \geq 0$ siten, että kielen kaikki sitä pidemmät sanat w ($|w| > k$) voidaan jakaa kolmeen osaan x, y ja z ($x, z \in \Sigma^*, y \in \Sigma\Sigma^*$) siten, että $xy^nz \in L, n \geq 0$.

   Toisin sanoen, mikäli säännöllinen kieli on ääretön, täytyy olla olemassa jokin merkkijono, jonka voi lisätä sanan keskelle mielivaltaisen monta kertaa.

   Tehtävässä piti todistaa, että $L = \{ww^R \,|\, w \in \{a,b\}^*\}$ ei ole säännöllinen. Määritellään kieli $L' = L \cap (ab)^*(ba)^*$. Koska säännöllisten kielten joukko on suljettu leikkauksen suhteen, ei L ole säännöllinen, jos L' ei ole säännöllinen.

   Tarkastelemalla kieleen L kuuluvia sanoja huomataan, että $L' = (ab)^n(ba)^n$, $n \geq 0$. Jotta L' voisi olla säännöllinen, pitäisi siis löytää y, joka voidaan lisätä sanaan mielivaltaisen monta kertaa.

   Nyt y voi olla muotoa:

   1° $y = (ab)^r$ tai $y = (ba)^r, r > 0$ Nyt kuitenkin toinen puolisko kasvaisi pidemmäksi kuin toinen, joten syntyneet sanat eivät kuulu enää kieleen.

2° $y = bb, y = aa, y = a(ba)^r$ tai $(ba)^r b, r \geq 0$ eivät myöskään kelpaa, sillä kielen sanoissa on aina yhtä monta a- kuin b-kirjainta.

3° $y = a(ba)^r bb(ab)^r b$ ei käy, koska tällöin sanassa esiintyy ba ennen viimeisiä ab:ta.

Näin huomattiin, että mikään y:n arvo ei ole mahdollinen. L' ei siis ole säännöllinen, joten myöskään L ei voi olla säännöllinen.

3. (2.4.5 / 2.6.5)

Ohjelmointikielen kääntäjää tehtäessä ensimmäinen vaihe on yleensä kielen leksikaalisen analysoijan suunnittelu. Leksikaalinen analysointi on ohjelman jäsennyksen ensimmäinen osa, ja siinä pyritään etsimään ohjelmasta yhteenkuuluvat merkkijonot (lekseemit). Esimerkiksi koodinpätkä `if (value == 10)` voitaisiin jakaa osiin seuraavasti:

(a) Lekseemi: '`if`' Tyyppi: varattu sana

(b) Lekseemi: '`(`' Tyyppi: ryhmitysmerkki

(c) Lekseemi: '`value`' Tyyppi: muuttuja

(d) Lekseemi: '`==`' Tyyppi: operaattori

(e) Lekseemi: '`10`' Tyyppi: numero

(f) Lekseemi: '`)`' Tyyppi: ryhmitysmerkki

Yleensä analysoija tallettaa lukemansa lekseemin jonkin muuttujan arvoksi ja palauttaa jäsentimelle lekseemin tyypin.

Leksikaaliset analysoijat tehdään yleensä käyttäen apuna työkalua (esim lex tai flex), jossa eri lekseemityypit määritellään käyttäen säännöllisiä lausekkeita. Työkalu tuottaa säännöllisistä lausekkeista deterministisen tilakoneen, jonka voi yleensä helposti liittää kielen jäsentimeen. Koska analysoijan täytyy osata vastata monimutkaisempaan kysymykseen kuin tavanomaisen tilakoneen, poikkeaa niiden semantiikka hieman tavallisten koneiden semantiikasta. Leksikaalinen analysoija ei lue automaattisesti koko syötettä kerrallaan, vaan se pysähtyy heti tunnistettuaan yhden lekseemin ja jättää lopun syötteen ennalleen.

Useimmissa työkaluissa määritellään joukko lyhennysmerkintöjä säännöllisten lausekkeiden kirjoittamisen helpottamiseksi. Yleisiä lyhennysmerkintöjä ovat esimerkiksi:

- '.', millä tarkoitetaan mitä merkkiä tahansa.

- '[]', millä kohtaa saa olla mikä tahansa hakasulkujen sisältä löytyvä merkki. Esimerkiksi merkintä [a-cX] on lyhennysmerkintä lausekkeelle $(a \cup b \cup c \cup X)$.
- '?', jota käytetään merkityksessä "korkeintaan yksi". Esimerkiksi lauseketta a? vastaa lauseke $(a \cup \emptyset)$.
- '+', joka tarkoittaa "vähintään yksi". Esimerkiksi a+, jota vastaa aa^*.
- Lisäksi työkaluissa lausekkeiden unionia $\cup$ esitetään yleensä merkillä $'|'$.

Tehtävän esimerkkikielen ellementtejä vastaavat seuraavanlaiset säännölliset lausekkeet:

i) Kokonaisluvut voidaan tunnistaa lausekkeella

$$-?[\mathsf{0}-\mathsf{9}][\mathsf{0}-\mathsf{9}]*$$

ii) Eri operaattorit voidaan tunnistaa lausekkeella

$$('+' \cup' -' \cup' *' \cup' /'\cup' =' \cup' ==')$$

iii) Muuttujat saadaan tunnistettua lauseella

$$[\mathsf{a}-\mathsf{z}]+$$

iv) Varatut sanat voi tunnistaa lausekkeella

$$\mathsf{while}\,|\,\mathsf{if}\,|\,\mathsf{else}$$

v) Ryhmitysmerkkien tunnistamiseen voi käyttää lausetta

$$('\{'\cup'\}' \cup' ('\cup')')$$

Hyvinkin yksinkertaisen kielen leksikaalisessa analysoijassa on useimmiten useita satoja tiloja. Alla esitetty kone tunnistaa varatun sanan while sekä w-kirjaimella alkavat muuttujanimet. Mikäli kone päätyy lopputilaan, jonka numero on 2 on löydetty while, numerolla 1 merkityissä tiloissa on löydetty muuttuja. Mikäli kirjaimen jälkeen luetaan suoraan jokin numero, päädytään virhetilaan.

Koneessa on käytetty lyhennysmerkintöinä ilmaisuja

- "a-z", millä tarkoitetaan mitä tahansa kirjainta, jolle ei löydy omaa kaarta pois tilasta.

- “0-9” tarkoittaa mitä tahansa numeroa.