Tik-79.148 **Kevät 2000**
Tietojenkäsittelyteorian perusteet
Laskuharjoitus 9
Ratkaisut

1. (4.1.8 / 4.3.1)

   Monimutkaisen Turingin koneen suunnittelu voi olla hyvinkin työlästä. Siksi onkin kehitetty konekaaviot, joiden avulla voidaan yhdistää yksinkertaisia Turingin koneita monimutkaisten toimintojen mahdollistamiseksi.

   Yksinkertaiset peruskoneet ovat:

   - Jokaista Σ:n symbolia varten on kone, joka kirjoittaa kyseisen symbolin nauhan kohdalle ja pysähtyy sitten siirtämättä lukupäätä.
   - Koneet R ja L, jotka siirtävät lukupäätä yhden askeleen verran oikealle tai vasemmalle ja pysähtyvät sitten.

   Tehtävän koneet toimivat seuraavasti:

   (a) Kone siirtää ikuisessa silmukassa lukupäätään oikealle.

   (b) Kone siirtää ensin lukupäätään yhden askeleen oikealle. Mikäli lukupään alla on nyt a, sen päälle kirjoitetaan b. Mikäli siinä on b, kirjoitetaan a.

   (c) Kone siirtää lukupäätään kaksi askelta vasemmalle. Tässä kannattaa huomata, että mikäli lukupää on jo valmiiksi nauhan vasemmassa reunassa, kone jää jumiin.

   (d) Kone ensin siirtää lukupäätään askeleen vasemmalle, ja mikäli se on nyt ei-tyhjän merkin kohdalla, siirretään lukupää takaisin oikealle.

2. (4.2.3 / 4.2.8)

   Turingin kone hyväksyy kielen L, mikäli $(s, \#w\underline{\#}) \vdash_M^* (h, \alpha), \alpha \in \Sigma^*$, aina kun $w \in L$. Kielen hyväksyttävyys on siten heikompi vaatimus kuin kielen ratkeavuus.

   Kielen a^*ba^*b hyväksyvä Turingin kone on:

$$
\begin{aligned}
M &= (K, \Sigma, \delta, s) \\
K &= \{q_0, q_1, q_2, q_3\} \\
\Sigma &= \{a, b, \#\} \\
s &= q_0
\end{aligned}
$$

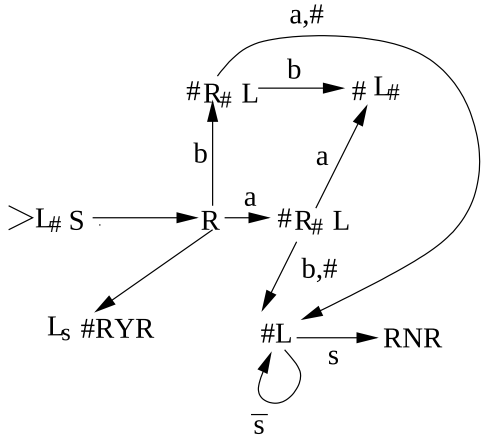
q	σ	$\delta(q,\sigma)$
q_0	$\#$	(q_1, L)
q_1	a	(q_1, a)
q_1	b	(q_2, L)
q_1	$\#$	$(q_1, \#)$
q_2	a	(q_2, L)
q_2	b	(q_3, L)
q_2	$\#$	$(q_2, \#)$
q_3	a	(q_3, L)
q_3	b	(q_3, b)
q_3	$\#$	$(h, \#)$

Kone on muodostettu lähtemällä liikkeelle vastaavasta tilakoneesta. Koska kone saa pysähtyä vain silloin, kun tutkittava sana kuuluu kieleen, jätetään kone ikuiseen silmukkaan aina, kun virhe löytyy. Tehtävän kieli on itseasiassa ratkeava, eli sille olisi mahdolista konstruoida myös tehtävän 3. tapainen Turingin kone.

3. (-)

Tehtävässä täytyy siis muodostaa kone, joka saa syötteensä muodossa $(\#w\underline{\#})$, ja tuloksena nauhalla on $(\#Y\underline{\#})$, mikäli $w \in L$ tai $(\#N\underline{\#})$, mikäli $w \notin L$.

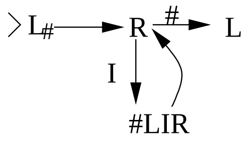
Koneen toteutuksessa käydään ensiksi merkitsemässä nauhan alkupiste symbolilla S, minkä jälkeen käydään sanaa läpi vertailemalla aina sanan alussa olevaa kirjainta sanan viimeiseen kirjaimeen.



Muotoa L_σ olevat koneet siirtävät lukupäätä vasemmalle kunnes ensimmäinen σ tulee vastaan.

4. (- / 4.4.7 a)

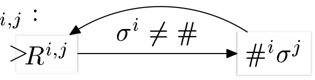
Suunnitellaan kone, joka laskee funktion $f(m,n) = m + n$. Kone saa syötteensä muodossa $(\#I^m\#I^n\underline{\#})$, ja sen pitää jättää nauha muotoon $(\#I^{m+n}\underline{\#})$. Helpoiten asia hoituu siirtämällä merkkijonoa I^n yhden askeleen verran vasemmalle:



5. (4.3.5 / -)

Ensiksi määritellään kone $C_{i,j}$, joka kopioi nauhalla i lukupään oikealla puolella olevat merkit nauhalle j kunnes vastaan tulee ensimmäinen tyhjä merkki:

$C_{i,j}$:



Esimerkiksi konfiguraatio:

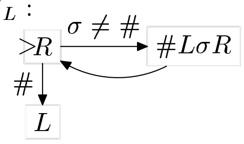
$\underline{\#}$	#	#	#	#	#
#	1	$\underline{0}$	1	0	#

muutetaan muotoon:

#	1	0	$\underline{\#}$	#	#
#	1	0	1	0	$\underline{\#}$

Seuraavaksi määritellään toinen apukone, M_L, joka siirtää lukupään oikealla puolella olevan merkkijonon yhden askeleen vasemmalle, ja jättää lukupään siirretyn alueen oikeaan reunaan.

M_L :



Esimerkiksi:

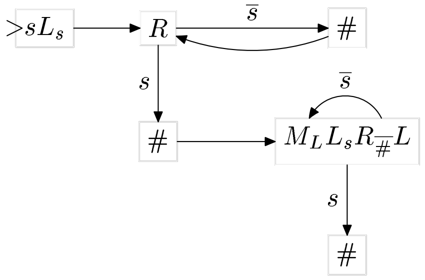
$\underline{\#}$	0	1	0

muutetaan muotoon:

0	1	0	$\underline{\#}$

Apukone E siirtää lukupään oikealla puolella olevan merkkijonon nauhan alkuun.

E :



Konfiguraatio:

$$\begin{array}{|c|c|c|c|c|c|c|}\hline \# & 0 & 1 & \underline{\#} & 1 & 0 & \# \\ \hline \end{array}$$

muutetaan muotoon:

$$\begin{array}{|c|c|c|c|c|c|c|}\hline \underline{\#} & 1 & 0 & \# & \# & \# & \# \\ \hline \end{array}$$

Muodostetaan ylläolevia apukoneita käyttäen 2-nauhainen kone $\sum_{1,2}$, joka laskee yhteen kaksi binäärilukua. Kone perustuu oppikirjan esimerkkiin 4.3.2 ja se on suhteellisen monimutkainen[1]. Kirjan esimerkkikone laskee yhteenlaskun oikein vain, jos molemmissa yhteenlaskettavissa on yhtä monta bittiä, ja sen lisäksi se jättää lopuksi nauhan virheelliseen tilaan. Näiden toiminnallisuuksien korjaaminen vaatii aika tavalla työtä.

Lähdettäessä liikkeelle konfiguraatiosta:

$$\left\{\begin{array}{l} \begin{array}{|c|c|c|c|c|c|c|c|}\hline \underline{\#} & \# & \# & \# & \# & \# & \# & \# \\ \hline \end{array} \\ \begin{array}{|c|c|c|c|c|c|c|c|}\hline \# & 1 & 0 & \# & 1 & 0 & 1 & \underline{\#} \\ \hline \end{array} \end{array}\right.$$

Kone $\sum_{1,2}$ merkitsee nauhojen alkukohdat symbolilla s, kopioi syötteen ensimmäisen sanan 2-nauhalle ja siirtää lukupäät nauhojen loppuun:

$$\left\{\begin{array}{l} \begin{array}{|c|c|c|c|c|c|c|c|}\hline \mathrm{s} & 1 & 0 & \underline{\#} & \# & \# & \# & \# \\ \hline \end{array} \\ \begin{array}{|c|c|c|c|c|c|c|c|}\hline \mathrm{s} & 1 & 0 & \# & 1 & 0 & 1 & \underline{\#} \\ \hline \end{array} \end{array}\right.$$

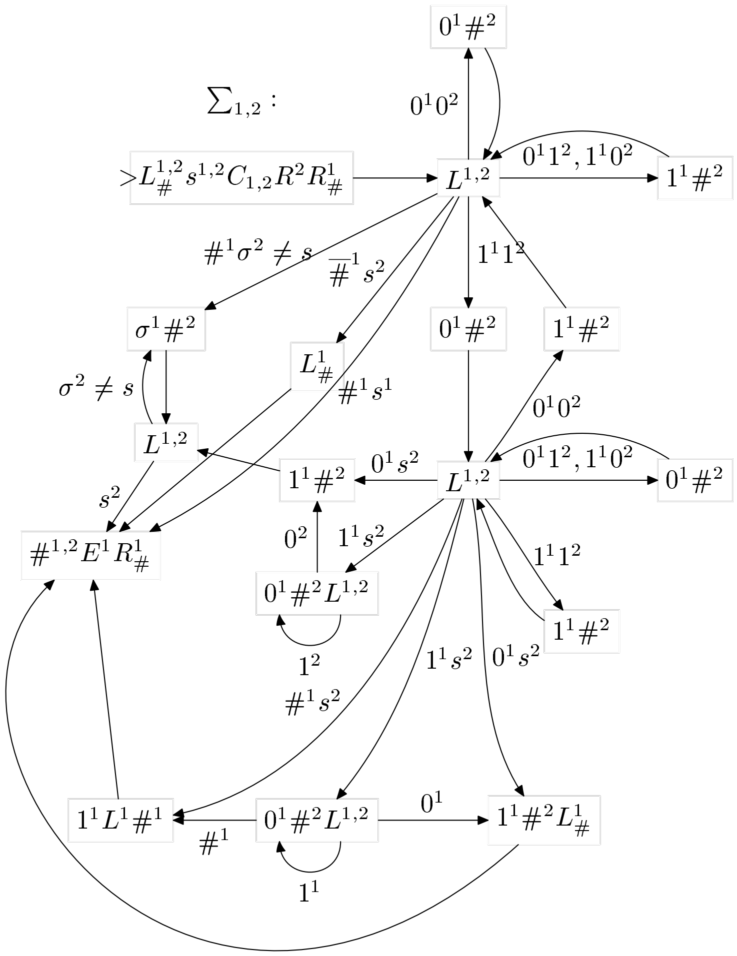
Tämän jälkeen käydään molempia lukuja lopusta alkuunpäin ja lasketaan niiden summa biteittäin. Samalla siivotaan 2-nauha.

$$\left\{\begin{array}{l} \begin{array}{|c|c|c|c|c|c|c|c|}\hline \underline{\mathrm{s}} & \# & \# & \# & \# & \# & \# & \# \\ \hline \end{array} \\ \begin{array}{|c|c|c|c|c|c|c|c|}\hline \mathrm{s} & 1 & 0 & \# & \underline{1} & 1 & 1 & \# \\ \hline \end{array} \end{array}\right.$$

Näin jatketaan, kunnes saavutaan jomman kumman luvun alkuun. Tässä esimerkissä ylempi luku loppui ensin. Nyt siirretään vastaukseksi saatu luku 111 ensimmäisen nauhan alkuun, ja poistetaan s-merkit.

$$\left\{\begin{array}{l} \begin{array}{|c|c|c|c|c|}\hline \underline{\#} & \# & \# & \# & \# \\ \hline \end{array} \\ \begin{array}{|c|c|c|c|c|}\hline \# & 1 & 1 & 0 & \underline{\#} \\ \hline \end{array} \end{array}\right.$$

[1]Mikäli joku lukijoista keksii yksinkertaisemman toteutuksen, niin kurssin assistentti on erittäin kiitollinen, jos sitä ei kerrota hänelle.

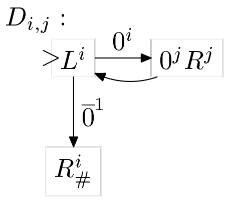


Ennen kuin päästään toteuttamaan varsinaista kertolaskua, tarvitaan vielä kaksi apukonetta: $D_{i,j}$ ja W_L. Kone $D_{i,j}$ laskee, kuinka monta nollaa on 1-nauhalla olevan luvun lopussa, ja kertoo 2-nauhalla olevan luvun kahdella yhtä monta kertaa. Esimerkiksi:

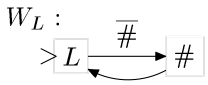
#	1	0	$\underline{\#}$	#	#	#	#
#	1	0	#	1	0	0	$\underline{\#}$

muutetaan muotoon:

#	1	0	0	0	$\underline{\#}$	#	#
#	1	0	#	$\underline{1}$	0	0	#



Kone W_L siivoaa nauhaa poistamalla lukupään vasemmalla puolella olevan sanan.

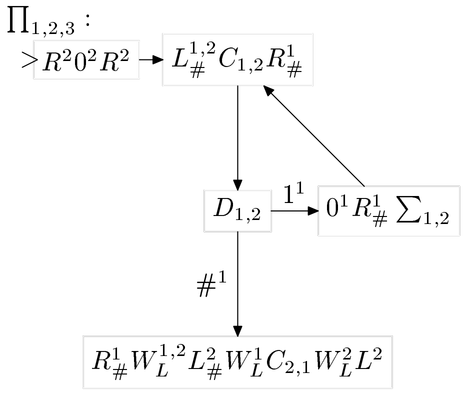


Binäärilukujen kertolasku voidaan palauttaa sarjaksi yhteenlaskuja ja kahdella kertomisia. Esimerkiksi:

$$101_2 \cdot 110_2 = 101_2 \cdot 2^2 + 101_2 \cdot 2^1$$

Näin ollen kertolasku voidaan toteuttaa seuraavanlaisella algoritmilla 3-nauhaisella Turingin koneella:

1. Kirjoitetaan 2-nauhan alkuun 0.
2. Kopioidaan ensimmäinen tekijä 2-nauhalle.
3. Käydään toista tekijää läpi lopusta alkuun. Jokaista luvun lopussa olevaa nollaa kohden kerrotaan 2-nauhalla oleva tekijä kahdella.
4. Mikäli 3-vaiheen lopuksi on löydetty toisesta tekijästä yksi 1-bitti, korvataan se 0:lla. Mikäli luku on muodostunut pelkistä nollista, on kertolasku valmis ja tulos on 2-nauhan alussa.
5. Lasketaan 2-nauhalla olevien lukujen summa käyttäen apuna 3-nauhaa ja palataan kohtaan 2.



Vaikka kone $\prod_{1,2,3}$ onkin 3-nauhainen, se käyttää ainoastaan kahta nauhaa eksplisiittisesti kolmannen toimiessa yhteenlaskukoneen apunauhana.

Tarkastellaan vielä, miten kone laskee kertolaskun $2 \cdot 3$ (eli $10_2 \cdot 11_2$).

Alussa kahden ensimmäisen nauhan sisältönä on:

$$\left\{\begin{array}{l}\begin{array}{|c|c|c|c|c|c|c|}\hline \underline{\#} & \# & \# & \# & \# & \# & \# \\ \hline\end{array}\\ \begin{array}{|c|c|c|c|c|c|c|}\hline \# & 1 & 0 & \# & 1 & 1 & \underline{\#} \\ \hline\end{array}\end{array}\right.$$

Alustetaan kertolaskun tulos nollaksi ja kopioidaan ensimmäinen tekijöistä 2-nauhalle.

$$\left\{\begin{array}{l}\begin{array}{|c|c|c|c|c|c|c|}\hline \# & 0 & \# & 1 & 0 & \underline{\#} & \# \\ \hline\end{array}\\ \begin{array}{|c|c|c|c|c|c|c|}\hline \# & 1 & 0 & \# & 1 & 1 & \underline{\#} \\ \hline\end{array}\end{array}\right.$$

Koska toisen tekijän viimeinen bitti ei ole nolla, ei 2-nauhan lukua tarvitse kertoa kahdella vaan yhteenlasku voidaan tehdä suoraan.

$$\left\{\begin{array}{l}\begin{array}{|c|c|c|c|c|c|c|}\hline \# & 1 & 0 & \underline{\#} & \# & \# & \# \\ \hline\end{array}\\ \begin{array}{|c|c|c|c|c|c|c|}\hline \# & 1 & 0 & \# & 1 & 0 & \underline{\#} \\ \hline\end{array}\end{array}\right.$$

Kopioidaan ensimmäinen tekijä uudestaan 2-nauhalle.

$$\left\{\begin{array}{l}\begin{array}{|c|c|c|c|c|c|c|}\hline \# & 1 & 0 & \# & 1 & 0 & \underline{\#} \\ \hline\end{array}\\ \begin{array}{|c|c|c|c|c|c|c|}\hline \# & 1 & 0 & \# & 1 & 0 & \underline{\#} \\ \hline\end{array}\end{array}\right.$$

Tällä kertaa ensimmäisen tekijän lopussa on nollabitti, joten jälkimmäinen yhteenlaskettavista kerrotaan kahdella.

$$\left\{\begin{array}{l}\begin{array}{|c|c|c|c|c|c|c|c|}\hline \# & 1 & 0 & \# & 1 & 0 & 0 & \underline{\#} \\ \hline\end{array}\\ \begin{array}{|c|c|c|c|c|c|c|c|}\hline \# & 1 & 0 & \# & \underline{1} & 0 & \# & \# \\ \hline\end{array}\end{array}\right.$$

Seuraavaksi ensimmäisen nauhan viimeinen ykkösbitti nollataan, ja toisen nauhan luvut lasketaan yhteen.

$$\left\{\begin{array}{l}\begin{array}{|c|c|c|c|c|c|c|}\hline \# & 1 & 1 & 0 & \underline{\#} & \# & \# \\ \hline\end{array}\\ \begin{array}{|c|c|c|c|c|c|c|}\hline \# & 1 & 0 & \# & 0 & 0 & \underline{\#} \\ \hline\end{array}\end{array}\right.$$

Seuraavalla iteraatiokierroksella huomataan, että tulon arvo on valmis ja se kopioidaan 1-nauhan alkuun.

$$\left\{\begin{array}{l}\begin{array}{|c|c|c|c|c|}\hline \underline{\#} & \# & \# & \# & \# \\ \hline\end{array}\\ \begin{array}{|c|c|c|c|c|}\hline \# & 1 & 1 & 0 & \underline{\#} \\ \hline\end{array}\end{array}\right.$$