T-79.148 **Syksy 2002**
Tietojenkäsittelyteorian perusteet
Harjoitus 12
Demonstraatiotehtävien ratkaisut

4. Tehtävässä halutaan todistaa seuraava ongelma ratkeamattomaksi:

   Hyväksyykö annettu Turingin kone M syötteen ε?

   Määritellään aluksi kieli $L = \{M \mid M \text{ pysähtyy syötteellä } \varepsilon\}$. Nyt L on rekursiivinen jos ja vain jos tehtävänannon ongelma on ratkeava. Seuraavaksi osoitetaan, että kieli $H = \{Mw \mid M \text{ pysähtyy syötteellä } w\}$ voidaan palauttaa rekursiivisesti (*recursively reduce*) kieleen L (merkitään $H \leq_m L$), joten L on vähintään yhtä vaikea kuin H. Koska H ei ole rekursiivinen (monisteen lause 6.9), ei L voi myöskään olla rekursiivinen.

   Rekursiivinen palautus määritellään seuraavasti: Olkoon $A \subseteq \Sigma^*$ ja $B \subseteq \Gamma^*$ kieliä. Nyt $A \leq_m B$ jos ja vain jos on olemassa rekursiivinen funktio $f : \Sigma^* \to \Gamma^*$ siten, että

   $$\forall w \in \Sigma^* : w \in A \Leftrightarrow f(w) \in B \ .$$

   Tässä tapauksessa halutaan löytää funktio f siten, että $f(Mw) \in L$ jos ja vain jos $Mw \in H$. Käytännössä tämä tarkoittaa sitä, että halutaan löytää systemaattinen tapa rakentaa Turingin kone M', joka pysähtyy tyhjällä syöttellä täsmälleen silloin, kun kone M pysähtyy syötteellä $w = w_1 w_2 \cdots w_n$.

   Onneksi tämä on helppo tehtävä. Kone M' kirjoittaa ensin nauhalle sanan w, siirtyy takaisin nauhan alkuun, ja alkaa simuloida konetta M tekemällä samat siirtymät kuin M:kin tekisi. Nyt M' pysähtyy jos ja vain jos M pysähtyy.

   Formaalisti f voidaan määritellä seuraavasti:

   $$f(\langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}\rangle, w_1 w_2 \cdots w_n) = \langle Q', \Sigma, \Gamma, \delta', q_0', q_{\text{acc}}, q_{\text{rej}}\rangle,$$

   missä

   $$\begin{aligned} Q' &= Q \cup \{q_i' \mid 0 \leq i \leq n\} \\ \delta' &= \delta \cup \{\langle q_i', \varepsilon, q_{i+1}', w_{i+1}, R\rangle \mid 0 \leq i < n\} \\ &\quad \cup \{\langle q_n', x, q_n', x, L\rangle \mid x \in \Gamma \cup \{<\}\} \\ &\quad \cup \{\langle q_n', >, q_0, >, R\rangle\} \end{aligned}$$

   Koska koneeseen M lisätään vain äärellinen määrä uusia tiloja ja siirtymiä (n ei voi olla ääretön), on f selvästikin rekursiivinen funktio.

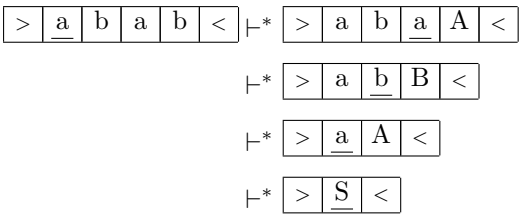
5. Yhteysherkän kieliopin määritelmän mukaan produktiot ovat sellaisia, että yksittäisessä säännössä oikea puoli on aina vähintään yhtä pitkä kuin vasen.

   Yhteysherkkä kieli voidaan tunnistaa lineaarisesti rajoitetulla automaatilla, joka epädeterminisesti siirtyy johonkin kohtaan syötettä ja soveltaa jotain sääntöä oikealta vasemmalle. Koska merkkijono tällöin lyhenee, lisätilaa ei tarvita. Toisaalta, jos väliin jäisi tyhjiä merkkejä, voidaan implementoida suoraviivainen tiivistys. Jos nauhalle jää pelkkä kieliopin aloitussymboli, siirrytään hyväksyvään lopputilaan.

   Tarkastellaan esimerkiksi yhteysherkkää kielioppia:

   $$\begin{aligned} S &\to aA \mid bB \\ aA &\to abB \mid ab \\ bB &\to baA \mid ba \end{aligned}$$

Tässä kielen tunnistava lineaarisesti rajoitettu automaatti muokkaisi syötenauhaansa seuraavaan tapaan (syötteellä *abab*):



Yllä kuvattu kone ei kuitenkaan ole totaalinen, sillä on mahdollista, että sen laskenta ei koskaan pysähdy. Esimerkiksi, jos käsiteltävä kielioppi on:

$$S \to \varepsilon$$
$$ab \to ba$$
$$ba \to ab,$$

niin kaikki laskennat ei-tyhjällä syötteellä jäävät ikuiseen silmukkaan.

Asian voi korjata huomaamalla, että koska koneen nauhan pituus ei voi kasvaa, on koneella on vain rajallinen määrä mahdollisia tilanteita. Näiden tilanteiden lukumäärä on suuruusluokkaa $q \times n \times |\Gamma|^n$, missä q on tilojen lukumäärä, n syötteen pituus ja $|\Gamma|$ aakkoston koko.

Edellä mainitulla tavalla rakennettu kone saadaan totaaliseksi liittämällä siihen laskuri, joka laskee suoritettujen askelten määrän. Helpoiten tämä käy siten, että tehdään koneesta kaksiurainen, jonka toiselle uralle kirjoitetaan askelten määrä binäärilukuna, jota kasvatetaan aina kunkin alkuperäisen koneen laskenta-askeleen yhteydessä yhdellä. Kun laskuri ylittää raja-arvon, sana hylätään, sillä tällöin kone on joutunut silmukkaan.

Lopuksi täytyy tarkistaa, että laskuri voidaan toteuttaa rikkomatta vaatimusta lineaarisesta tilankäytöstä. Luvun $q \times n \times |\Gamma|^n$ esittämiseen tarvitaan $k = \log_2(q) + \log_2(n) + n\log(|\Gamma|)$ bittiä, joka kasvaa lineaarisesti n:n kasvaessa. Vaikka tässä $k > n$, niin laskuri saadaan puristettua käytössä olevaan tilaan esittämällä se sopivassa kannassa.