Tik-79.148 **Kevät 2001**
Tietojenkäsittelyteorian perusteet
Laskuharjoitus 10
Ratkaisut

4. Tyypin 0 kielioppi on nelikko $G = (V, \Sigma, R, S)$, missä V, Σ ja S on määritelty samoin kuin yhteydettömien kielioppien tapauksessa. Kieliopin säännöt ovat muotoa

$$R \subseteq V^*(V - \Sigma)V^* \times V^*$$

Säännön vasemmalla puolella saa olla mikä tahansa merkkijono, kunhan se vain sisältää ainakin yhden nonterminaalin.

Tehtävän ratkaisee kielioppi G:

$$\begin{aligned} \Sigma &= \{a\} \\ V &= \Sigma \cup \{S, [,], F, N, A\} \\ R &= \{S \to [N], N \to FNA, N \to e \\ &\quad FA \to AaF, Fa \to aF, F] \to], \\ &\quad [A \to [, [a] \to a[, [] \to e\} \end{aligned}$$

Ideana on johtaa aluksi n kappaletta F-nonterminaalia sanan alkuun ja n-kappaletta A-nonterminaalia sanan loppuun. Tämän jälkeen kuljetetaan F-merkit sanan loppuun siten, että aina kun ylitetään nonterminaali A, lisätään sanaan sille kohtaa terminaalisymboli a. Koska sekä F että A merkkejä on n kappaletta, syntyy a-merkkejä $n \cdot n = n^2$ kappaletta. Lopuksi siivotaan syntynyt merkkijono kuljettamalla alkumerkki [sanan loppuun siten kaikki matkalla tavattavat A-nonterminaalit poistetaan.

Esimerkiksi sana a^{3^2} saadaan johdettua seuraavasti:

$$\begin{aligned} S &\to [N] \to [FNA] \to [FFNAA] \to [FFFNAAA] \to [FFFAAA] \\ &\to [FFAaFAA] \to [FFAaAaFA] \to [FFAaAaAaF] \to [FFAaAaAa] \\ &\to [FAaFaAaAa] \to [FAaaFAaAa] \to [FAaaAaFaAa] \to [FAaaAaaFAa] \\ &\to [FAaaAaaAaFa] \to [FAaaAaaAaaF] \to [FAaaAaaAaa] \\ &\to [AaFaaAaaAaa] \to [AaaFaAaaAaa] \to [AaaaFAaaAaa] \\ &\to [AaaaAaFaaAaa] \to [AaaaAaaFaAaa] \to [AaaaAaaaFAaa] \\ &\to [AaaaAaaaAaFaa] \to [AaaaAaaaAaaFa] \to [AaaaAaaaAaaaF] \\ &\to [AaaaAaaaAaaa] \to [aaaAaaaAaaa] \to a[aaAaaaAaaa] \to aa[aAaaaAaaa] \\ &\to aaa[AaaaAaaa] \to aaa[aaaAaaa] \to aaaa[aaAaaa] \to aaaaa[aAaaa] \\ &\to aaaaaaa[Aaaa] \to aaaaaa[aaa] \to aaaaaaa[aa] \to aaaaaaaa[a] \\ &\to aaaaaaaaa[] \to aaaaaaaaa \end{aligned}$$

5. Jos haluamme todistaa, että jokin ongelma on ratkeamaton, voimme joko

   (i) Lähteä liikkeelle oletuksesta, että ongelma on ratkeava, ja johtaa siitä ristiriita käyttäen valitsemaamme formalismia.

(ii) Jos jo tiedämme jonkin toisen kielen olevan ratkeamaton, voimme yrittää palauttaa (redusoida) oman ongelmamme tähän ongelmaan. Tähänkin on kaksi eri vaihtoehtoa. Ensimmäinen vaihtoehto on se, että osoitamme, että mikäli pystyisimme ratkaisemaan tutkimamme ongelman voisimme myös käyttää sitä kyseisen ratkeamattoman ongelman ratkaisemiseen. Toinen mahdollinen tapa on osoittaa, että pystyäksemme ratkaisemaan ongelmamme meidän täytyisi pystyä ratkaisemaan ratkeamaton ongelma.

Tässä vastauksessa käytetään jälkimmäistä tapaa. Todistuksen ideana on osoittaa, että pystyäksemme ratkaisemaan ongelman “pysähtyykö mielivaltainen Turingin kone M tyhjällä syötteellä“, meidän täytyisi pystyä ratkaisemaan ongelma “pysähtyykö mielivaltainen Turingin kone M' syötteellä x”. Jälkimmäinen ongelma on Turingin koneen yleinen pysähtymisongelma, joka on oppikirjassa todistettu ratkeamattomaksi.

Mikäli ongelma on ratkeava, on olemassa Turingin kone, joka ratkaisee kielen

$$L = \{M \mid M \text{ pysähtyy syötteellä } e\} \ .$$

Toisin sanoen, on olemassa Turingin kone, joka saadessaan syötteekseen minkä tahansa toisen Turingin koneen kuvauksen pystyy selvittämään pysähtyykö tämä kone tyhjällä syötteellä. Todistukseksi kielen ratkeamattomuudesta riittää, että löydämme yhdenkin Turingin koneen, jonka pysähtymistä ei pystytä selvittämään.

Muodostetaan Turingin kone M, joka toimii seuraavasti: saatuaan syötteekseen tyhjän merkkijonon kone kirjoittaa aluksi nauhalle mielivaltaisen merkkijonon x (kirjoitettava merkkinono voidaan valita epädeterministisesti). Tämän jälkeen kone siirtyy simuloimaan mielivaltaista Turingin konetta M', eli se tekee tismalleen samat siirtymät kuin kone M':kin tekisi.

Nyt kone M pysähtyy syötteellä e täsmälleen silloin , kun kone M' pysähtyy syötteellä x. Koska jälkimmäinen ongelma ei ole ratkeava, ei myöskään voi olla olemassa mitään algoritmia sen selvittämiseksi, pysähtyykö M syötteellä e. Näin ollen ongelma on ratkeamaton.

6. Olkoon annettuina kaksi Turing-hyväksyttävää kieltä L_1 ja L_2. Tällöin on määritelmän mukaan olemassa kaksi Turingin konetta M_1 ja M_2, jotka hyväksyvät kyseiset kielet (eli $L(M_1) = L_1$ ja $L(M_2) = L_2$). Muodostetaan näitä käyttäen 2-nauhaiset Turingin koneet $M_\cap$ ja $M_\cup$, jotka hyväksyvät kielten leikkauksen ja unionin.

   (a) Leikkaus:
   Muodostetaan $M_\cap$ siten, että suorituksen aluksi $M_\cap$ kopio syötteen w 2-nauhalle. Tämän jälkeen M_cap simuloi koneen M_1 toimintaa syötteellä w käyttäen ensimmäistä nauhaa työnauhanaan. Mikäli M_1 pysähtyy, tiedetään, että $w \in L_1$. Nyt voidaan simuloida konetta M_2 syötteellä w käyttäen toista nauhaa työnauhana. Mikäli M_2:kin pysähtyy, kuuluu sana molempiin kieliin, ja $M_\cap$ hyväksyy sanan. Mikäli w ei kuulu jompaan kumpaan kieleen, jää kyseisen kielen tunnistava kone ikuiseen silmukkaan, eikä kone $M_\cap$ pysähdy. Näin ollen $M_\cap$ hyväksyy kielen $L_1 \cap L_2$.

(b) Unioni:

Kielten unionin tunnistava kone $M_\cup$ muodostetaan samalla periaatteella. Tässä täytyy kuitenkin huomata, että koneita M_1 ja M_2 ei voida simuloida peräkkäin, koska M_1:n jäädessä ikuiseen silmukkaan (kun $w \notin L_1$) ei päästä koskaan tarkistamaan kuuluisiko sana w kieleen L_2.

Ratkaisuna on simuloida koneita M_1 ja M_2 rinnakkain, yksi askel kerrallaan. Ensiksi suoritetaan koneen M_1 laskennan ensimmäinen siirtymä, sitten M_2:n ensimmäinen siirtymä, sitten M_1:n toinen siirtymä ja niin edespäin. Mikäli sana kuuluu jompaan kumpaan kieleen, joko M_1 tai M_2 pysähtyy joskus, jolloin $M_\cup$ hyväksyy syötteen.