Rekursiolause

Laskennan teorian opintopiiri

Sebastian Björkqvist

23. helmikuuta 2014

Tiivistelmä

Työssä käydään läpi itsereplikoituvien ohjelmien toimintaa sekä esitetään ja todistetaan rekursiolause, jonka avulla mikä tahansa Turingin kone voi saada käyttöönsä oman kuvauksensa. Lisäksi työssä todistetaan rekursiolauseen avulla muutamia ratkeamattomuustuloksia.

Johdanto

Rekursiolause on tärkeä matemaattinen tulos tietojenkäsittelytieteessä, ja se liittyy vahvasti mm. itsereplikoitumiseen. Voisi hyvin kuvitella, että kone joka luo toisia koneita on itse monimutkaisempi kuin koneet jotka se luo, koska senhän täytyy sisältää kuvaukset luomistaan koneista. Rekursiolauseen avulla voidaan kuitenkin osoittaa, että mikä tahansa kone pystyy myös replikoimaan itseään, eli se ei välttämättä ole sen monimutkaisempi kuin sen luomat koneet. Yksi konkreettinen rekursiolauseen sovellus on tietokonevirus, ohjelma joka levittää itseään koneesta toiseen ja mahdollisesti tekee tuhoa.

Työ seuraa suurilta osin Sipserin kirjan *Introduction to the Theory of Computation* ([1]) lukua 6.1 (s. 221-228). Itsereplikoituvista ohjelmista, joita englanniksi kutsutaan nimellä quine, löytyy runsaasti esimerkkejä Internetistä ([2], [3]).

Itsereplikoituvan ohjelman olemassaolo

Itsereplikoituvalla ohjelmalla tarkoitetaan ohjelmaa, joka käynnistyessään tulostaa oman kuvauksensa (eli lähdekoodinsa) ja pysähtyy. Esitämme miten itsereplikoituva ohjelma voidaan luoda Turingin koneen avulla; konstruktiota soveltamalla voimme luoda itsereplikoituvan ohjelman millä tahansa Turing-täydellisellä ohjelmointikielellä.

Tavoitteenamme on siis luoda Turingin kone joka tulostaa oman kuvauksensa kun se ajetaan. Kutsutaan tätä konetta nimellä $SELF$. Ideana on luoda kone $SELF$ kahdesta osasta A ja B, jotka ovat itsekin Turingin koneita. Koneen A tehtävänä on tulostaa koneen B kuvaus ja toisinpäin. Tämän toteuttamiseksi tarvitaan seuraavaa määritelmää ja apulausetta:

Määritelmä 1. Olkoon w merkkijono. Turingin kone P_w on kone, joka palauttaa jonon w kaikilla syötteillä. Tässä jonon w palauttamisella tarkoitetaan sitä, että koneen pysähtyessä sen nauhalla on jono w.

Turingin koneesta P_w löytyy siis sisäänohjelmoituna jono w. Seuraava apulause sanoo, että pystymme aina luomaan koneen P_w:

Apulause 2. *Olkoon w mikä tahansa merkkijono. On olemassa Turingin kone Q, joka syötteellä w palauttaa kuvauksen koneesta P_w.*

Todistus. [1, Lemma 6.1] Määritellään Turingin kone Q seuraavasti:

Q = "Syötteellä w:

1. Luo seuraavanlainen Turingin kone P_w:

   P_w = "Kaikilla syötteillä:

   1. Poista syöte.
   2. Tulosta jono w nauhalle ja pysähdy."

2. Palauta jono $\langle P_w \rangle$."

□

Saamme nyt helposti rakennettua koneen $SELF$ ensimmäisen osan: määrittelemme $A = P_{\langle B \rangle}$, eli A on kone, joka käynnistyessään tulostaa kuvauksen koneesta B. Koska $\langle B \rangle$ on merkkijono, pystymme luomaan koneen $P_{\langle B \rangle}$ edeltävän apulauseen avulla. Huomaamme, että koneen A määritelmä riippuu koneesta B, eli koneen A määritelmä ei ole valmis ennen kuin olemme määritelleet koneen B. Täten emme myöskään voi vain määritellä $B = P_{\langle A \rangle}$, sillä tästä seuraisi kehäpäätelmä.

Kone B määritelläänkin hieman eri tavalla: B laskee koneen A kuvauksen saamastaan syötteestä. Formaalisti kone B määritellään seuraavasti:

B = ”Syötteellä $\langle M \rangle$, missä M on jokin Turingin kone:

1. Simuloi apulauseen 2 konetta Q syötteellä $\langle M \rangle$. Palautuu kuvaus $\langle P_{\langle M \rangle} \rangle$.
2. Yhdistä kuvaukset $\langle P_{\langle M \rangle} \rangle$ ja $\langle M \rangle$ yhdeksi Turingin koneeksi T.
3. Tulosta kuvaus koneesta T nauhalle.”

Katsotaan mitä tapahtuu, jos koneelle B annetaan syötteeksi sen oma kuvaus $\langle B \rangle$:

1. B simuloi konetta Q syötteellä $\langle B \rangle$. Palautuu kuvaus $\langle P_{\langle B \rangle} \rangle = \langle A \rangle$!
2. B yhdistää kuvaukset $\langle A \rangle$ ja $\langle B \rangle$ yhdeksi koneeksi AB.
3. B tulostaa kuvauksen $\langle AB \rangle$.

Täten, jos ajamme ensiksi koneen A, tulostaa se jonon $\langle B \rangle$ nauhalle. Kun ajamme koneen B, katsoo se nauhaa ja saa sieltä syötteen $\langle B \rangle$, jolloin B tulostaa kuvauksen $\langle AB \rangle$ nauhalle kuten yllä nähtiin. Koska muodostamme koneen $SELF$ koneista $A = P_{\langle B \rangle}$ ja B, joista A ajetaan ensin, tulee $SELF$ tulostamaan oman kuvauksensa $\langle SELF \rangle = \langle AB \rangle$.

Esimerkkejä itsereplikoituvista ohjelmista

Edellisen kappaleen koneen $SELF$ rakennetta matkimalla voimme luoda itsereplikoituvan ohjelman millä tahansa Turing-täydellisellä ohjelmointikielellä. Voimme myös antaa esimerkkejä itsereplikoituvista ohjelmista suomen kielellä. Tällöin itsereplikoituvalla ohjelmalla tarkoitetaan suomen kielen lausetta, joka ohjeistaa lukijaa kirjoittamaan lukemansa lauseen. Eräs esimerkki on seuraava:

Kirjoita tämä lause.

Tällaista suoraa itseviittausta ei yleensä ole mahdollista toteuttaa suoraan ohjelmointikielissä. On toki mahdollista tehdä ohjelma joka lukee tiedostosta oman lähdekoodinsa, mutta pystymme myös luomaan itsereplikoituvan ohjelman joka ei tarvitse mitään syötettä. Suomen kielellä kone $SELF$ voidaan toteuttaa seuraavasti:

Kirjoita seuraava lause kahdesti, toisen kerran lainausmerkeissä:
"Kirjoita seuraava lause kahdesti, toisen kerran lainausmerkeissä:"

Tässä siis konetta B vastaa lause

Kirjoita seuraava lause kahdesti, toisen kerran lainausmerkeissä:

Kun koneelle B annetaan sen oma kuvaus ajamalla kone A, saadaan aikaiseksi itsereplikoituva kone.

Alla on lähdekoodi itsereplikoituvasta Java-ohjelmasta [2]:

```
1  public class Self {
2      public static void main(String[] args) {
3          char l = 34; // Lainausmerkki
4          String[] k = { // Kooditaulukko
5              "public class Self {",
6              "    public static void main(String[] args) {",
7              "        char l = 34; // Lainausmerkki",
8              "        String[] k = { // Kooditaulukko",
9              "            ",
10             "        };",
11             "        for (int i = 0; i < 4; i++) {",
12             "            System.out.println(k[i]);",
13             "        }",
14             "        for (int i = 0; i < k.length; i++) {",
15             "            System.out.println(k[4] + l + k[i] + l + ',');",
16             "        }",
17             "        for (int i = 5; i < k.length; i++) {",
18             "            System.out.println(k[i]);",
19             "        }",
20             "    }",
21             "}",
22         };
23         for (int i = 0; i < 4; i++) {
24             System.out.println(k[i]);
25         }
26         for (int i = 0; i < k.length; i++) {
27             System.out.println(k[4] + l + k[i] + l + ',');
28         }
29         for (int i = 5; i < k.length; i++) {
30             System.out.println(k[i]);
31         }
32     }
33 }
```

Koko ohjelman lähdekoodi sisältyy rivillä 4 määriteltävään taulukkoon. Taulukon sisällä ei toki neljännen rivin jälkeen sisällytetä taulukkoa uudestaan, sillä koko ohjelman koodihan on jo saatavilla. Riveillä 23-25 tulostetaan ohjelman neljä ensimmäistä riviä. Tämän jälkeen riveillä 26-28 tulostetaan koko lähdekooditaulukko lainausmerkkien sisällä. Lopuksi riveillä 29-31 tulostetaan loput lähdekoodista, eli rivit mitkä sisältävät for-loopit sekä lopun kaarisulut.

Sivulta [3] löytyy runsaasti esimerkkejä itsereplikoituvista ohjelmista eri ohjelmointikielillä.

Rekursiolause

Turingin kone voi rekursiolauseen avulla tehdä omalla kuvauksellaan muutakin kuin tulostaa sen, esimerkiksi käyttää omaa kuvaustaan laskennassa. Formaalisti rekursiolause on seuraava:

Lause 3. *Olkoon T Turingin kone, joka saa syötteenä kaksi aakkoston Σ^* merkkijonoa. Tällöin on olemassa Turingin kone R, joka syötteellä $w \in \Sigma^*$, missä w voi olla mikä tahansa aakkoston merkkijono, tekee samaa kuin kone T syötteellä $\langle R, w\rangle$.*

Jos siis haluamme luoda koneen joka käyttää omaa kuvaustaan ajon aikana, meidän pitää ainoastaan luoda kone T, joka saa ylimääräisenä syötteenä koneen kuvauksen. Rekursiolause antaa meille tällöin koneen R, johon on sisäänrakennettuna sen oma kuvaus $\langle R\rangle$ ja joka toimii kuten T.

Todistus. [1, Thm 6.3] Rekursiolauseen todistus muistuttaa hyvin paljon koneen $SELF$ rakennetta. Rakennamme Turingin koneen R kolmessa osassa A, B ja T, ja määrittelemme ensiksi $A = P_{\langle BT\rangle}$. Tätä tarkoitusta varten muutamme apulauseen 2 koneen Q määritelmää siten, että kone $P_{\langle BT\rangle}$ kirjoittaa jonon $\langle BT\rangle$ saamansa syötteen perään. Kun A on ajettu, nauhalla on siis jono $w\langle BT\rangle$, missä w on koneen R saama syöte.

Kone B määritellään jälleen koneeksi, joka simuloi konetta Q saamallaan syötteellä. Kone Q palauttaa tällöin jonon $\langle P_{\langle BT\rangle}\rangle = \langle A\rangle$. Tämän jälkeen B yhdistää koneet A, B ja T yhdeksi koneeksi $ABT = R$, ja lopuksi B yhdistää koneen R ja jonon w jonoksi $\langle R, w\rangle$ ja pysähtyy. Täten kun kone T käynnistyy, saa se syötteen $\langle R, w\rangle$. □

Voimme nyt luoda koneen $SELF$ rekursiolauseen avulla määrittelemällä seuraavanlainen kone T:

T = ”Syötteellä $\langle M, w\rangle$:

1. Tulosta kuvaus $\langle M\rangle$ ja pysähdy.”

Tällöin rekursiolauseen nojalla löytyy kone R, joka toimii syötteellä w kuten T toimii syötteellä $\langle R, w\rangle$, eli saamme seuraavanlaisen koneen:

R = ”Syötteellä w:

1. Tulosta kuvaus $\langle R\rangle$ ja pysähdy.”

Kone R siis tulostaa oman kuvauksensa.

Rekursiolauseen sovelluksia

Rekursiolausetta käyttämällä voimme missä tahansa kohtaa Turingin koneen kuvauksessa hankkia koneen oman kuvauksen ja laskea sillä, koska pystymme aina muodostamaan rekursiolauseen vaatiman koneen T joka saa kuvauksen syötteenä. Tällä tavalla voimme todistaa tiettyjä ratkeamattomuustuloksia hyvin helposti.

Palautetaan mieleen että hyväksymisongelmassa kysytään, onko olemassa Turingin konetta H, joka pystyy syötteellä $\langle M, w\rangle$ sanomaan hyväksyykö vai hylkääkö Turingin kone M syötteen w. Hyväksymisongelman ratkeamattomuus todistetaan yleensä diagonaalimetodilla, mutta esitämme nyt toisenlaisen todistuksen rekursiolausetta käyttämällä.

Lause 4. *Hyväksymisongelma on ratkeamaton.*

Todistus. [1, Thm 6.5] Tehdään vastaoletus, eli oletetaan, että Turingin kone H ratkaisee hyväksymisongelman. Muodostetaan seuraavanlainen kone B:

B = ”Syötteellä w:

1. Hanki rekursiolauseen avulla oma kuvaus $\langle B\rangle$.
2. Simuloi konetta H syötteellä $\langle B, w\rangle$.
3. Jos H hyväksyy, hylkää. Jos H hylkää, hyväksy.

Nyt kone B tekee millä tahansa syötteellä w päinvastaista kuin mitä H väittää sen tekevän, eli H ei ratkaise hyväksymisongelmaa. □

Voimme siis ikäänkuin huijata konetta H vastaamaan väärin, sillä pääsemme käsiksi koneen B kuvaukseen sen ajon aikana. Esitetään selvennykseksi vielä rekursiolauseen vaatima kone T, johon lauseen 4 todistus pohjautuu:

T = ”Syötteellä $\langle M, w\rangle$:

1. Simuloi konetta H syötteellä $\langle M, w\rangle$.
2. Jos H hyväksyy, hylkää. Jos H hylkää, hyväksy.”

Käsittelin kurssin toisessa työssäni Kolmogorov-kompleksiteettia. Jonon n Kolmogorov-kompleksiteetti määriteltiin jonon lyhimmän kuvauksen pituudeksi, missä jonon n lyhin kuvaus on lyhin merkkijono $\langle M, w\rangle$ jolle pätee, että Turingin kone M palauttaa jonon n syötteellä w. Työssä esitin erään todistuksen sille, että Kolmogorov-kompleksiteettia ei voida yleisesti laskea. Rekursiolauseen avulla saamme tälle lyhyemmän todistuksen:

Lause 5. *Mikään algoritmi ei pysty laskemaan Kolmogorov-kompleksiteettia kaikille merkkijonoille.*

Todistus. Tehdään vastaoletus, eli oletetaan että kone K osaa laskea kaikkien jonojen Kolmogorov-kompleksiteetin. Muodostetaan seuraavanlainen Turingin kone N:

N = "Kaikilla syötteillä:

1. Hanki rekursiolauseen avulla oma kuvaus $\langle N \rangle$.
2. Käy läpi jonoja lyhimmästä alkaen kunnes löytyy jono w jolle pätee että $K(w) > |\langle N \rangle|$.
3. Palauta jono w."

Kaikenpituisia tiivistymättömiä jonoja on olemassa [1, Thm 6.29], joten kone N löytää varmasti jonon w, jolle pätee $K(w) > |\langle N \rangle|$. Jos annamme koneelle N tyhjän syötteen, palauttaa se jonkin jonon w, eli jono $\langle N \rangle$ on eräs jonon w kuvaus. Tämä on ristiriita, sillä jonon w lyhimmän kuvauksen pituudelle $K(w)$ pätee $K(w) > |\langle N \rangle|$. □

Viitteet

[1] Sipser, M. *Introduction to the Theory of Computation*, second edition. Thomson, 2006.

[2] `http://en.wikipedia.org/wiki/Quine_%28computing%29#Examples` Viitattu 22.02.2014

[3] `http://www.nyx.net/~gthompso/quine.htm` Viitattu 23.02.2014