Rajoittamattomat kieliopit

Laura Pesola

Helsinki March 1, 2013

Aine

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author

Laura Pesola

Työn nimi — Arbetets titel — Title

Rajoittamattomat kieliopit

Oppiaine — Läroämne — Subject

Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Aine	March 1, 2013	0 sivua

Tiivistelmä — Referat — Abstract

ACM Computing Classification System (CCS):

Avainsanat — Nyckelord — Keywords

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Rajoittamattomat kieliopit

1.1 Formaalit kieliopit

Formaalit kieliopit ovat merkkijonojen uudelleenkirjoitusjärjestelmiä. Niiden säännöt kuvaavat, miten valitusta aakkostosta saadaan muodostettua kaikki ja vain sellaiset merkkijonot, jotka vastaavat sääntöjä. Rajoittamattomat kieliopit ovat formaaleita kielioppeja. Chomskyn hierarkiassa rajoittamattomat kieliopit ovat tyyppiä 0 – siis kaikista suurin luokka. Huomattavaa on, että tyypin 0 kielioppien kuvaamat kielet ovat rekursiivisesti numeroituvia, siis Turing-tunnistettavia. Koska Chomskyn luokittelu ei ole hienojakoisin mahdollinen, voidaan tyypin 0 kieliopit vielä jakaa kahteen luokkaan: rekursiivisiin (Turing-ratkeavat) ja rekursiivisesti numeroituviin (Turing-tunnistettavat) kieliin. Rekursiivisten kielten joukko on rekursiivisesti numeroituvien kielten aito osajoukko.

1.2 Rajoittamattomat kieliopit

Formaali määrittely rajoittamattomille kieliopeille on seuraava:

$G = (N, \Sigma, P, S)$, jossa

- N = muuttujasymbolien joukko
- Σ = päätesymbolien joukko
- P = sääntöjen joukko
- S = lähtösymboli

Huomionarvoista on, että N ja Σ voivat sinällään olla myös sama joukko. Niiden erillisyys johtuu vain siitä, että halutaan tietää, milloin lausemuotojen generoiminen lopetetaan.

1.3 Säännöt

Säännöt ovat muotoa $\alpha \rightarrow \beta$, jossa sekä oikean- että vasemmanpuoleiset säännöt voivat olla mitä tahansa merkkijonoja; niin pääte- kuin muuttujasymboleita. Toisin kuin nimi antaa ymmärtää, rajoittamattomat kieliopit eivät kuitenkaan ole

täysin rajoittamattomia: niiden sääntöjoukkoa koskee yksi rajoite. Tämän rajoitteen mukaan vasemmanpuoleinen symboli ei saa koostua pelkästä tyhjästä merkkijonosta. Mielenkiintoista on se, että sääntö $\alpha \rightarrow \varepsilon$ on sallittu myös silloin kun $\alpha = S$ ja S esiintyy jossain säännössä oikealla puolella. Itse asiassa tämä on merkittävä piirre: jos kontekstiherkissä kieliopeissa sallittaisiin tämänkaltainen sääntö, myös ne tuottaisivat kaikki Turing-tunnistettavat kielet.

2 Turing-tunnistettavuus

Kuten aiemmin mainittiin, Turing-tunnistettavien kielten ja rajoittamattomien kielioppien tuottamien kielten joukko on sama. Siis: jos on olemassa rajoittamaton kielioppi G, joka tuottaa kielen L, on myös olemassa Turingin kone M, joka tunnistaa kielen L. Tämä luonnollisesti pätee myös toiseen suuntaan. Kaikille Turing-tunnistettaville kielille pätee, että on olemassa jokin Turingin kone, joka tunnistaa sen kielen merkkijonot.

2.1 Ekvivalenssitodistus

Seuraavaksi hahmottelen, miten todistus Turingin koneiden ja rajoittamattomien kielioppien ekvivalenssista etenee. Esittelen todistukset molempiin suuntiin, mutta en käy kumpaakaan todistusta formaalisti läpi. Sen sijaan molemmista käyn läpi idean.

2.1.1 Rajoittamattoman kieliopin tuottama kieli Turing-tunnistettava

Jos on olemassa rajoittamaton kielioppi G, joka tuottaa kielen L, niin L on Turing-tunnistettava. Haluamme siis näyttää, että mille tahansa kielelle, jonka tuottaa jokin rajoittamaton kielioppi, voidaan konstruoida Turingin kone, joka tuottaa tai tunnistaa sen kielen merkkijonot. Jos oletamme, että Churchin-Turingin teesi pätee, riittää, että kuvaamme algoritmin, jolla kone toimii; itse konetta ei ole tarpeen konstruoida. Alla esitän algoritmin, jonka avulla voidaan todeta, kuuluuko jokin merkkijono kieleen L.

- Olkoon M kaksinauhainen epädeterministinen Turingin kone ja G mikä tahansa rajoittamaton kielioppi, joka tuottaa kielen L.

- Turingin koneen nauhaa 1 käytetään syötemerkkijonon säilömiseen.
- Nauhalle 2 generoidaan G:n lausemuotoja.
- G:n säännöt on koodattu Turingin koneen tiloihin
- Kone aloittaa vasemmalta ja aina joko pysyy paikallaan tai siirtyy oikealle
    1. Siirretään lukupää vasempaan laitaan, jos se ei ole jo siellä
    2. Valitaan epädeterministisesti jokin G:n sääntö $\beta \to \gamma$
    3. Jos β tulee vastaan nauhalla 2, korvataan se γ:lla
    4. Verrrataan nauhan 2 sisältöä nauhan 1 sisältöön: jos sama, sana kuuluu G:n tuottamaan kieleen.
    5. Palataan kohtaan 1.
- Ajon aikana generoidaan kaikki mahdolliset G:n lausemuodot.

2.1.2 Jokaiselle Turing-tunnistettavalle kielelle L on olemassa rajoittamaton kielioppi, joka tuottaa L:n

Jos L on Turing-tunnistettava kieli, niin jokin rajoittamaton kielioppi tuottaa L:n. Todistus tähän suuntaan on hyvin yksityiskohtainen ja tekninen. Sen voi lukea kokonaisuudessaan lähteestä [1]. Oleellisin ajatus on, että kieliopin ja Turingin koneen suorittaman laskennan välille luodaan jonkinlainen yhteys. Tämä yhteys saadaan aikaan, kun muistetaan, että Turingin koneen tilaa (configuration) voidaan kuvata merkkijonolla. Kun valitaan sopivanlainen kuvaus, voidaan kieliopilla ikään kuin simuloida Turingin koneen toimintaa. Voidaan siis luoda kielioppi, joka generoi Turingin koneen konfiguraatioita.

3 Tyypin 0 kieli

Koska tyypin 0 kielet ovat Chomskyn hierarkian laajin luokka, ja kaikki alempiin luokkiin kuuluvat kielet kuuluvat siis myös tyypin 0 kielten luokkaan, on olemassa paljon kieliä, jotka ovat tyyppiä 0. Mutta minkälainen kieli sitten erottaa tyypin 1 kielet tyypin 0 kielistä? Jos haluamme löytää tällaisen kielen ja uskoa, että se todella edustaa vain tyyppiä nolla, meidän on pystyttävä näyttämään, että kieli on

rekursiivisesti numeroituva ja että sitä ei voida tunnistaa lineaarisesti rajoitetulla automaatilla. Eräs tällainen kieli on seuraava:

$\{M \mid M$ on LBA ja $M \notin L(M)\}$, siis

- M on jokin lineaarisesti rajoitettu automaatti
- M tunnistaa kielen L
- M :n kuvaus ei kuitenkaan kuulu kieleen L

Kieli on Turing-tunnistettava, jos jokin Turingin kone voi tuottaa kaikki kielen merkkijonot. Jos oletetaan, että Churchin-Turingin teesi pätee, niin riittää, että kuvataan algoritmi kielelle. Tässä vaiheessa on hyvä muistaa, että lineaarisesti rajoitettujen automaattien tunnistamat kielet ovat Turing-ratkeavia. Tämä johtuu siitä, että lineaarisesti rajoitetusta automaatista tiedetään aina, milloin se on jäänyt loopiin, eikä ole pysähtymässä.

- Olkoon Turingin kone TM, joka saa syötteekseen merkkijonon x ja lineaarisesti rajoitetun automaatin M
- Syötteellä M, x:
  1. Simuloi M:n toimintaa syötteellä x
  2. Jos M hyväksyy, hylkää; muutoin hyväksy
- Kohta 1 ei voi jäädä loopiiin

Esimerkkikieli ei myöskään ole kontekstiherkkä.

- Olk. L kieli, jonka tunnistaa LBA M
- Oletetaan, että $M \in L$ ja $L = L(M)$
  - Nyt M hyväksyy M:n
    $\rightarrow M \notin L$
  - Ja päinvastoin: jos $M \notin L$ ja $L = L(M)$, niin M ei hyväksy M:ää
    $\rightarrow M \in L$

Kuten huomataan, tästä syntyy ristiriita – siis kielen tunnistaminen lineaarisesti rajoitetulla automaatilla ei ole mahdollista. Näin ollen kieli ei siis voi olla kontekstiherkkä.

4 Muuta

4.1 Thue

Rajoittamattomien kielioppien pohjalta on kehitetty esoteerinen ohjelmointikieli, Thue. Sen avulla voidaan määritellä ja tunnistaa nollatyypin kieliä. Se on Turing-täydellinen, mutta myöskin ns. Turing tarpit – siis kieli jolla voidaan periaatteessa laskea mitä vain laskettavissa olevaa, mutta jolla minkäänlaisen laskennan suorittaminen on yleensä hankalaa. Lisäksi tällaiset kielet pyrkivät toteuttamaan Turing-täydellisyyden mahdollisimman pienillä määrillä komponentteja.

4.2 Semi-Thue-järjestelmä

Tarkemmin sanottuna Thue perustuu semi-Thue-järjestelmälle, joka on nimetty norjalaisen matemaatikon Axel Thuen mukaan. Myös semi-Thue-järjestelmä on uudelleenkirjoitusjärjestelmä. Sen ero rajoittamattomiin kielioppeihin syntyy päätteja muuttujasymbolijoukkojen samuudesta sekä aloitussymbolin puuttumisesta. Chomskyn hierarkia taas perustuu sille, että asetetaan rajoitteita sille, minkälaisissa paikoissa minkäkin joukon jäsenet voivat säännöissä esiintyä. Semi-Thue järjestelmä kehitettiin propositiologiikkaa varten, kun haluttiin hoitaa todistuksia automaattisesti. Tämäkin järjestelmä on isomorfinen Turingin koneiden kanssa.

4.3 Esimerkki Thuesta

Thue-ohjelmointikieli koostuu uudelleenkirjoitussäännöistä. Koodi on siis muotoa merkkijono1::=merkkijono2

- yksinkertainen esimerkki, Hello World
- $a ::= \tilde{}HelloWorld!$
  ::=
  a

4.4 Lähteet

1. Adrian Francalanza: (An Introduction to) Computability and Complexity, Department of Computer Science ICT, University of Malta

http://staff.um.edu.mt/afra1/teaching/coco5.pdf

2. Gul Agha, Mahesh Viswanathan: Theory of Computation
   http://courses.engr.illinois.edu/cs373/fa2010/lectures/notes19.pdf
3. James Hein: Discrete Structures, Logic, and Computability
4. John E. Hopcroft: Introduction to Automata Theory, Languages and Computation
5. http://esolangs.org/wiki/Thue
6. http://en.wikipedia.org/wiki/Semi-Thuesystem
7. http://en.wikipedia.org/wiki/AxelThue