Toteutusdokumentti

Tietorakenteiden harjoitustyö

Joosua Laakso

12. tammikuuta 2014

1 Ohjelman yleisrakenne

Ohjelman suorituksesta voidaan eritellä useita eri välivaiheita. Kaksi päävaihetta ohjelman suorituksessa ovat säännöllisen lausekkeen lataaminen ja merkkijonon tunnistaminen ladatulla säännöllisellä lausekkeella.

1.1 Lausekkeen lataaminen

Lausekkeen lataaminen tapahtuu kahdessa vaiheessa, ensin tapahtuu lausekkeen validointi, jossa lauseke käydään läpi merkki merkiltä ja tarkistetaan, onko se kieliopillisesti validi säännöllinen lauseke. Toisessa vaiheessa, jäsennyksessä, säännöllisen lausekkeen merkkijonoesitys muutetaan jäsennyspuuesitykseksi. Merkkijonoesitys käydään läpi merkki merkiltä ja siitä rakennetaan jäsennyspuu, joka on binääripuu, jonka solmut kuvaavat säännöllisten lausekkeiden peruskieliopillisia rakenteita, jotka ovat katenaatio, unioni ja tähti. Tämän lisäksi jäsennyspuun lehtisolmut voivat olla tavallisia merkkejä, jokerimerkkejä jotka hyväksyvät minkä tahansa merkin, epsilon-alkioita jotka hyväksyvät vain tyhjän merkkijonon tai null-alkioita jotka eivät hyväksy mitään merkkijonoa. Kaikki muut erityisemmät ominaisuudet, jotka ovat tuettuja tässä toteutuksessa, on mahdollista kuvailla näiden kielioppirakenteiden avulla.

1.2 Merkkijonon tunnistaminen

Merkkijonon tunnistaminen tapahtuu käyttämällä Brzozowskin derivaattoja. [1] Säännöllinen lauseke derivoidaan merkkijonon kunkin merkin suhteen, jonka jälkeen tarkistetaan, hyväksyykö jäljelle jäänyt lauseke tyhjän merkkijonon. Jos jäljelle jäänyt lauseke hyväksyy tyhjän merkkijonon, alkuperäinen lauseke hyväksyy merkkijonon, jos taas ei, niin alkuperäinen lauseke ei hyväksy merkkijonoa. Joidenkin säännöllisten lausekkeiden rakenteiden, kuten tähti-operaation, derivointi aiheuttaa jäsennyspuun laajenemisen. Tästä johtuen, pitkien merkkijonojen tunnistaminen saattaa olla hidasta. Tätä ongelmaa lievittää se, että jokaisen derivoinnin jälkeen suoritetaan jäsennyspuun redusointi. Redusoinnissa jäsennyspuusta pyritään poistamaan turhia oksia siten, että redusoitu jäsennyspuu on kuitenkin edelleen ekvivalentti redusoimattoman kanssa. Empiirisen testauksen mukaan hyvin suurien jäsennyspuiden redusointi on kallista, tämän vuoksi redusointi tehdään vain, jos jäsennyspuun korkeus on suurempi kuin merkkijonon pituus kertaa kaksi.

[1] dl.acm.org/citation.cfm?id=321249

2 Tehokkuus

Suoritusaika on riippuvainen kahdesta muuttujasta, säännöllisestä lausekkeesta muodostetun jäsennyspuun solmujen määrästä, sekä syötteeksi annetun merkkijonon pituudesta. Empiirisen testauksen mukaan, kun molempia muuttujia kasvatetaan, ohjelman vaatima suoritusaika näyttää kasvavan eksponentiaalisesti, samoin ohjelman vaatima muisti. Tämä saattaa johtua siitä, että jäsennyspuun redusointi ei toimi täydellisesti, vaan jäsennyspuu saattaa kasvaa joissain tapauksissa suuresti derivoinnin seurauksena. Työ ei kuitenkaan ollut tehokkuusorientoitunut, vaan päätavoite oli tuottaa toiminnallinen toteutus.

3 Parannusehdotukset

Tehokkuus on ilmeinen osa-alue jossa on vielä selvästi parannettavaa. Jäsennyspuun redusointiin voisi ehkä kehittää jonkinlaisia heuristiikkoja, joiden avulla operaatiota voisi tehostaa. Itse lausekkeiden derivointikaan ei välttämättä tällaisenaan ole toteutettu niin tehokkaasti, kun sen voisi toteuttaa. Jo derivoinnin yhteydessä voisi yrittää saada tuloksena olevan puun mahdollisimman redusoituun muotoon. Erilaisia uusia kielioppiominaisuuksia voisi lisätä, kuten operaattorit, jotka merkitsisivät rivin alkua ja rivin loppua, jotka ovat hyödyllisiä sovelluksissa, joissa syötettä käydään läpi rivi riviltä. Ohjelma saattaa vielä sisältää jonkinlaisia virheitä, jotka tietenkin on aina tärkeää korjata.