2. Viikkoraportti
Tietorakenteiden harjoitustyö

Joosua Laakso

5. tammikuuta 2014

Jouluviikolla sain työn siihen vaiheeseen, että ohjelma tuki kaikkia säännöllisten lausekkeiden perusoperaatioita, eli katenaatiota, unionia ja tähteä. Tämän jälkeen lisäsin muitakin operaattoreita. Joidenkin operaattorien lisääminen oli helppoa, koska ne pystyi vain määrittelemään valmiiksi tuettujen operaattorien avulla, mutta hakasulkujen, eli vaihtoehtoisten merkkien lisääminen oli hankalaa, samoin aaltosulkujen, eli määrällisen toiston lisääminen vaati hieman enemmän työtä. Opin että jos olisin ollut järkevä, niin olisin paloitellut Parser-luokan useaan eri joista jokin luokka olisi esimerkiksi hoitanut hakasulut, toinen aaltosulut ja niin edes päin. Jos olisin tehnyt näin, niin luokka olisi varmasti paljon selkeämpi, mutta tällä hetkellä se on kaikkea muuta. En kuitenkaan aijo ryhtyä tähän enää koska uskon että siitä olisi tässä vaiheessa enemmän vaivaa kun hyötyä.

Aloitin suorituskykytestauksen ja tein testejä jotka testaavat ohjelman suorituskykyä lausekkeen ollessa vakio, "[A-D]+", satunnaisella syötteellä jonka lauseke hyväksyy ja joka kasvaa. Testeistä voi tehdä kaksi epämielyttävää havaintoa: syötteen kasvaessa suoritusaika kasvaa paljon sekä ohjelmani ei ole ollenkaan yhtä tehokas kuin Javan standardikirjaston säännöllisten lausekkeiden toteutus. Esimerkiksi 10000 pituisella syötteellä Javan standardikirjaston säännölliset lausekkeet toimivat noin 1000 kertaa nopeammin kuin omani.

Aion tehdä vielä lisää testejä jotka testaavat suorituskykyä esimerkiksi syötteen sekä lausekkeen pituuden kasvaessa, sekä myös tapauksissa, joissa lauseke hylkää syötteen. Olen myös kuullut että joillakin säännöllisten lausekkeiden toteutuksilla on olemassa syötteitä, jotka ovat lyhyitä, mutta niiden suorittaminen on suhteettoman hidasta. Aion testata ohjelman toimintaa myös tällaisilla patologisilla syötteillä.

Uskon että pystyn tekemään ohjelmasta vielä paljon tehokkaamman. Tällä hetkellä ohjelma toimiii siten että jos säännöllinen lauseke on vaikka "b*", niin syötteellä "aaaccacaa" ohjelma lukee koko syötteen, vaikka jo ensimmäisen merkin jälkeen tiedetään, että lauseke ei hyväksy merkkijonoa. Jos olisin toteuttanut ohjelman epädeterministisillä äärellisillä tila-automaateilla, tätä ongelmaa ei tietenkään olisi. Voin kuitenkin tehdä niin, että jokaisen lausekkeen derivoinnin jälkeen tarkistan, voiko lauseke enää hyväksyä mitään merkkijonoa, jos ei, niin derivointimetodi palauttaa suoraan null-lausekkeen, jolloin suoritus loppuu. Toinen etu tästä on se, että uskon että se korjaa aiemmin mainitsemani pitkiin syötteisiin liittyvän tehokkuusongelman. Koska jos lauseke sisältää tähtioperaattorin, niin joka kerta kun tämä lauseke derivoidaan, niin lauseketta esittävä jäsennyspuu kasvaa. Uskon kuitenkin että tämän jäsennyspuun kasvun voi estää siten, että puusta karsitaan jatkuvasti oksia, jotka eivät voi enää hyväksyä merkkijonoja, eli jotka voidaan vaihtaa null-lausekkeeseen, muuttamatta lausekkeen merkitystä. Tämän parannuksen pitäisi jopa parantaa toteutuksen aikavaativuusluokkaa, koska jäsennyspuun koko ei enää vaihtelisi syötteen koon mukaan, ainakaan niin rajusti.