Janne Isoaho

LR-jäsennyksen toiminta ja käyttökohteet

Tietotekniikan kandidaatintutkielma

13. toukokuuta 2018

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Janne Isoaho

Yhteystiedot: janne.o.isoaho@student.jyu.fi

Ohjaaja: Antti-Juhani Kaijanaho

Työn nimi: LR-jäsennyksen toiminta ja käyttökohteet

Title in English: Function and uses of LR-parsing

Työ: Kandidaatintutkielma

Sivumäärä: 21+0

Tiivistelmä: LR-jäsennys on eräs kohtuullisen tehokas tapa jäsentää deterministisiä kontekstittomia kieliä. Tämän työn tarkoitus on tutkia, miten LR-jäsennys toimii ja millaisissa tilanteissa sitä voi käyttää. Tutkimus on suoritettu kirjallisuuskatsauksena. LR-jäsennyksen toimintaperiaate on yksinkertainen, mutta jäsentimiä voi luoda useilla eri tekniikoilla. LR-jäsennyksen käyttö on perusteltua silloin, mikäli kielioppi on yhteensopivaa ja tarvitaan tehokasta jäsennintä.

Avainsanat: LR-jäsennys, kääntäjä, jäsennin

Abstract: LR-parsing is one fairly efficient way to parse deterministic context-free languages. The purpose of this thesis is to study how LR-parsing functions and in which cases it is justified to use LR-parsing. The study has been conducted as a literature review. LR-parsing principles are simple, but there are many ways to generate LR-parsers. Usage of LR-parsing is justified if grammar is compatible and efficient parser is needed.

Keywords: LR-parsing, compiler, parser

Kuviot

Taulukot

Sisältö

1 Johdanto

Tässä työssä on tutkittu LR-jäsennyksen toimintaa ja sen käyttökohteita. Isona osana työtä on ensiksi jäsennysongelman ja kontekstittomien kielioppien määritteleminen, toiseksi itse LR-jäsennyksen määritteleminen ja kolmanneksi muiden LR-jäsennykseen liittyvien termien ja algoritmien, kuten esimerkiksi GLR:n, käsitteleminen.

Jäsennysongelmaan, kontekstittomiin kielioppeihin ja eri algoritmeihin sekä muihin jäsennystapoihin pureudutaan luvussa 2. Luvussa 3 käydään läpi LR-jäsennyksen historia, yleinen toiminta ja teoria sekä esimerkki sen toiminnasta. Luvussa 4 puolestaan kuvataan, millaisissa yhteyksissä LR-jäsennystä käytetään ja minkä vuoksi. Työn pääpaino kuitenkin on LR-jäsennyksen toiminnan ympärillä, joten luvun 4 tarkoitus on vain helpottaa hahmottamaan LR-jäsennyksen erilaisia käytännön sovellutuksia.

Työn tarkoituksena on antaa yleiskuva LR-jäsennyksestä ja sen toiminnasta sekä antaa yleiskuvaa sen eri käyttökohteista. LR-jäsennys aiheena on relevantti, sillä se toimii lineaarisessa ajassa ja on siten erittäin tehokas. Jäsennystä yleisesti sekä LR-jäsennystä on tutkittu paljon. Tästä syystä erilaisia algoritmeja, joiden toimintaperiaate on ainakin osittain tai tietyissä tilanteissa sama kuin LR-jäsennyksessä on useampia. Tässä työssä tällaisistä esimerkkinä on Generalized LR -parsing, jatkossa GLR.

2 Jäsennysongelma

Jäsentämiseen on monia erilaisia tapoja ja tyylejä. Eräs tyypillinen tapa jakaa jäsennystapoja on kahtiajako bottom-up- ja top-down-jäsennykseen. Myös järjestys, jossa välikemerkkejä käsitellään, vaihtelee joissain jäsennystavoissa. Näistä käytetään termejä leftmost-derivation ja rightmost-derivation riippuen siitä, käsitelläänkö välikemerkit oikealta vasemmalle vai päinvastoin (Aho, Sethi ja Ullman 2007, s.165-166).

Jäsennyksellä on kuitenkin tärkeä rooli useiden ongelmien ratkaisussa, yhtenä esimerkkinä koodin kääntäminen ajettavaksi ohjelmaksi. Jäsentimen määritelmä Chapmanin (1987, s.19) mukaan on, että se osaa rakentaa lauseen syntaksia vastaavan jäsennyspuun, joka noudattaa kieliopin produktioiden määräämiä sääntöjä.

Tässä luvussa käsitellään sitä, mitä jäsennyksellä tarkoitetaan. Ennen kuin voidaan siirtyä tarkemmin jäsennyksen teoriaan, on tärkeää määritellä kontekstittomat kieliopit. Tämä tehdään alaluvussa 2.1. Samassa luvussa käydään läpi esimerkki selventämään kyseisten kielioppien hyödyllisyyttä. Alaluvussa 2.2 puolestaan käsitellään bottom-up- ja top-down-jäsennyksiä. Luvun loppupuolella siirrytään ensin luvussa 2.3 käymään läpi lyhyesti ensin LR-jäsennyksen monipuolisempi variaatio GLR-jäsennys ja kielten tunnistamiseen hyödyllinen Earleyn algoritmi. Viimeisessä alaluvussa 2.4 esitellään hieman muita jäsennystapoja kuten LL-jäsennys ja Packrat.

2.1 Kontekstittomat kieliopit

Tietojenkäsittelytieteessä käytetään formalismia erilaisten kielten ja kielioppien määrittelyyn. Eräs formaalisti määritelty kielioppijoukko on kontekstittomat kieliopit. Kontekstittomat kieliopit ovat hyvin yleisesti erilaisissa jäsennystavoissa käytetty kielioppityyppi. Nimi kontekstittomuus tulee siitä, että produktioita, joista myöhemmin tässä luvussa annan esimerkin, voi käyttää korvaamiseen välittämättä kunkin merkin sijainnista syötetekstissä, tai siitä onko sen edessä tai takana jokin toinen tietty merkki.

Formaalisti kontekstittomat kieliopit määritellään seuraavasti: on olemassa jokin konteksti-

ton kielioppi $G = (V, \Sigma, R, S)$. Aho, Sethi ja Ullman (2007, s.165-166) mukaillen tässä nelikossa V on kieliopin G päätemerkkien muodostama joukko, Σ välikesymbolien joukko, R produktioiden joukko, joka mainittiin aikaisemmin, ja S on se V:n alkio, joka on myös kieliopin ensimmäinen symboli eli lähtösymboli. Nyt kun on määritelty ja nimetty kaikki osat nelikosta, voidaan tarkemmin näyttää produktioiden, nelikossa R, määritelmä. Koska kontekstittomissa kieliopeissa produktiot ovat muunnoksia välikesymboleista toisiin välikesymboleihin tai päätemerkkeihin, voidaan matemaattisin merkinnöin todeta, että $R \rightarrow (V \cup \Sigma)*$ (jossa *-merkkiä käytetään Kleineen tähtenä).

Seuraavaksi esimerkki eräästä kontekstittomasta kieliopista.

$$
\begin{aligned}
S &\rightarrow P \\
P &\rightarrow A \\
P &\rightarrow B \\
P &\rightarrow APA \\
P &\rightarrow BPB \\
P &\rightarrow \varepsilon \\
A &\rightarrow a \\
B &\rightarrow b
\end{aligned}
$$

Tämä kielioppi määrittelee kielen, josta voi muodostaa erilaisia palindromeja kielen merkistöllä. Tarkastellaan seuraavaksi, mitkä kieliopin osat vastaavat mitäkin aiemmin määritellyn nelikon $G = (V, \Sigma, R, S)$ osia. Tämän kieliopin päätemerkit ovat kielen merkistö $\{a, b, \varepsilon\}$ (ε-merkkiä käytetään tässä ilmaisemaan tyhjää), välikesymbolit $\{S, P, A, B\}$ ja produktioita R ovat kaikki esimerkin rivit. Nelikon viimeinen osa lähtösymboli S on tässä kieliopissa merkitty välikesymbolilla S.

Kielioppien käyttö siis perustuu välikesymbolien korvaamiseen säännön $R \rightarrow (V \cup \Sigma)*$ mukaisesti. Tämän voi paremmin havainnollistaa laatimalla eräs johto, vaikka esimerkki kieliopin sanalle "abba". Johdossa siis näytetään, miten kieliopin avulla luodaan jokin kielen sana.

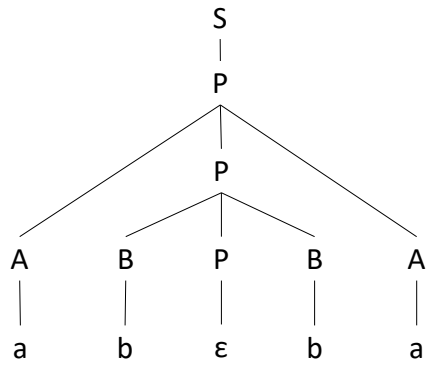
$$
\begin{aligned}
S &\Rightarrow P \\
&\Rightarrow APA \\
&\Rightarrow aBPBa \\
&\Rightarrow ab\varepsilon ba \\
&\Rightarrow abba
\end{aligned}
$$

2.2 Bottom-up- ja top-down -jäsennyksen eroavaisuudet

Jäsentimet luovat saamastaan syötteestä jäsennyspuun, jonka ne sitten antavat kääntäjissä eteenpäin. (Aho, Sethi ja Ullman 2007) Jäsennyksen jälkeen jäsennyspuun lehtiin jää syötteen alkuperäiset merkit ja solmuihin tulee niihin sopivat välikesymbolien ja päätemerkkien joukot. Se, missä järjestyksessä jäsennin luo puurakenteen, vaihtelee riippuen siitä, noudattaako jäsennin bottom-up- vai top-down-periaatteita.

LR-jäsennys toimii bottom-up-tyylisesti, eli alhaalta ylöspäin. Alhaalta ylöspäin eteneminen tarkoittaa jäsennyksessä sitä, että jäsennin aloittaa jäsennyspuun rakentamisen lehdistä ja tunnistaa ensimmäisenä pienimmät rakenteet, ja vasta käytyään kaikki isomman rakenteen pienemmät osaset läpi, muodostaa se niistä kokonaisuuden (Chapman 1987, s. 19). Näin edetään, kunnes ylimmän tasonkin rakenne on tunnistettu.

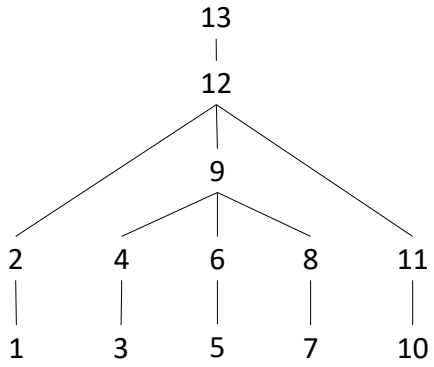
Kuviossa 1 jäsennyspuu luvussa 2.1 esitetyn kieliopin mukaiselle sanalle "abba".



Kuvio 1. Jäsennyspuu sanalle "abba".

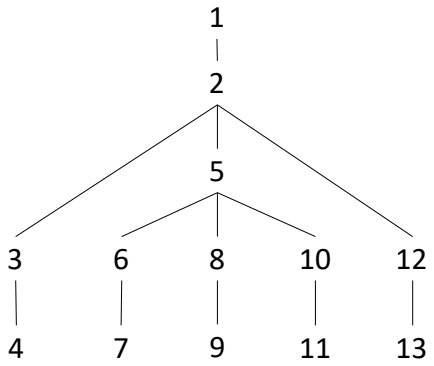
Vaihdetaan seuraavaksi kuviossa 2 kaikkien päätemerkkien ja välikesymbolien tilalle nume-

ro, joka havainnollistaa missä järjestyksessä bottom-up jäsennys tunnistaa ne.



Kuvio 2. Bottom-up jäsennyksen järjestys.

Viimeiseksi vielä top-down jäsennyksen järjestystä havainnollistava kuvio 3.



Kuvio 3. Top-down jäsennyksen järjestys.

Kuvioita 2 ja 3 vertailemalla huomataan, että top-down jäsennyksessä, jäsennyspuu luodaan, nimensä mukaisesti, ylhäältä alaspäin. Top-down jäsennyksestä on esimerkkinä packrat. Packratista enemmän luvussa 2.4.

2.3 GLR-jäsennys ja Earleyn algoritmi

GLR-jäsennys (englanniksi Generalized LR-parsing) on Aycockin ja Horspoolin (1999) mukaan Masaru Tomitan kehittämä variaatio LR-jäsennyksestä. Aycockin ja Horspoolin mukaan GLR-jäsennys toimii niin, että se noudattaa perinteisiä LR-jäsennyksen algoritmeja,

koska ne toimivat lineaarisessa ajassa eli erittäin tehokkaasti ja nopeasti, mahdollisimman pitkään ja turvautuu tehottomampiin menetelmiin vain tarvittaessa. Aycockilla ja Horspoolilla on heidän mukaansa tehokkaampi versio GLR-jäsennyksestä kuin Tomitan alkuperäinen, mutta toisaalta tämän jälkeen on julkaistu väitetysti vielä tehokkaampia algoritmeja kuten esimerkiksi Aycock ym. (2001). Tässä kappaleessa ei kuitenkaan kiinnitetä huomiota eri algoritmien tehokkuuteen vaan tarkastellaan yleisesti GLR-jäsennystä.

Toinen tärkeä algoritmi on Earleyn algoritmi, se tunnistaa toteuttaako merkkijono kielen. Earley (1970) kertoo sen artikkelissaan toimivan ajassa n^3, n:n pituisella merkkijonolla, kaikille kontekstittomille kielille. Kuitenkin Earleyn mukaan kaikille yksiselitteisille kieliopeille algoritmi tunnistaa lauseet ajassa n^2 ja niistäkin useille yleisimmille ja hyödyllisimmille lineaarisessa ajassa n. Juuri tämä monikäyttöisyys tekee Earleyn algoritmista erittäin hyödyllisen. Useat algoritmit eivät joko toimi näin laajalle kirjolle kielioppeja tai sitten niiden aikavaativuus nousee eksponentiaaliseksi c^n, joillain vakiolla c ja lauseen pituudella n.

Merkityksellisen tämän työn kannalta Earleyn algoritmista tekee se, että se toimii myös LR-kieliopeille, sillä nämä ovat kontekstittomien kielioppien alajoukko (Earley 1970).

2.4 Muut jäsennystavat

LR-jäsennys ei suinkaan ole ainut mahdollinen tapa ratkaista niin kutsuttua jäsennysongelmaa. LR-jäsennyksen edustaman bottom-up-jäsennyksen joukkoon kuuluu muitakin tapoja, kuten esimerkksi precedence parsing. Siirrytään nyt kuitenkin tutkailemaan top-down-jäsennystä. Fordin (2002) mukaan top-down-jäsennys jaetaan kahteen luokkaan, ensiksi ennustaviin jäsentimiin (engl. predictive parser) ja toiseksi palaaviin jäsentimiin (engl. backtracking parser). Fordin määritelmän mukaan ennustavassa jäsennyksessä jäsennin voi kurkata rajatun määrän merkkejä voidakseen ennakoida millaisen kielen syöte muodostaa, kun taas palaavassa jäsennyksessä käydään vaihtoehdot yksi kerrallaan läpi ja palataan takaisin valintaan, jos kyseinen vaihtoehto ei johda haluttuun tulokseen.

Fordin (2002) kehittelemässä Packrat jäsennyksessä hyödynnetään top-down-periaatteita, vaikkakin hänen mukaansa Packrat välttelee valintaa ennustavan- ja palaavan jäsennyksen välillä. Ford (2002) kertoo Packrat-jäsennys eliminoi korkean aikavaativuuden säilyttämällä

kaikki jo lasketut jäsennyksen tulokset, jolloin algoritmi varmistaa, ettei yhtäkään tulosta lasketa monikertoihin. Parr ja Fisher (2011) mainitsevat myös, Packrat-jäsennyksestä sen isoimman puutteen, vasenta rekursiota (engl. left recursion) sisältäviä kielioppeja ei voi käyttää. Toisaalta Ford (2002) huomauttaa, että tälläiset kieliopit voi muuntaa muotoon, jossa vasen rekursio korvataan vastaavalla oikealla rekursiolla (engl. right recursion).

Viimeisenä muista esiteltävistä jäsennystavoista on toinen top-down-periaatteita hyödyntävä jäsennystapa, LL-jäsennys. LL-jäsennykseenkin on useita algoritmeja. Parr ja Fisher (2011) esittelemä LL(*) jäsennys on eräs uudemmista tavoista hyödyntää LL-jäsennyksen hyvää virheiden hallintaa sekä muita hyviä puolia.

3 LR-jäsennys

LR-jäsennys tulee sanoista Left-to-right Rightmost derivation. Auki selitettynä se tarkoittaa sitä, että luetaan syöte vasemmalta oikealle ja syödään merkit oikeanpuoleisemmasta alkaen. Chapmanin (1987, s. 3, käännös oma) mukaan ”LR-jäsentimet ovat ovat erittäin hyödyllinen luokka bottom-up -jäsentimiä, niitä [LR-jäsentimiä] voidaan generoida isolle alajoukolle kontekstittomia kielioppia ja ne toimivat lineaarisessa ajassa”. Chapmanin mukaan LR-jäsennyksessä hyödynnettyjen tekniikoiden hyödyllisyys piilee siinä, että niitä pystyy käyttämään suurelle määrälle eri kieliä, mikä taas johtuu siitä, että näiden tekniikoiden avulla voi luoda automaattisesti jäsentimiä eri kielille.

LR-jäsentimet voivat tehdä neljää asiaa tutkiessaan syötettä, joista yleisimmät kaksi ovat seuraavat: lukupään siirto (engl. shift), skannattujen merkkien sievennys (engl. reduce). Muut kaksi ovat syötteen hyväksyminen (engl. accept) ja virheen antaminen (engl. error) (Aho ja Johnson 1974). Shift ja reduce ovat yleisimmät, sillä näiden avulla vuorottelemalla varsinainen jäsennys tapahtuu. Se, miten shift/reduce jäsennys toimii, selitetään yksityiskohtaisemmin alaluvussa 3.2 ja näytetään taulukossa 1.

LR-jäsentimiä merkitään usein LR(k), jossa k tarkoittaa montako merkkiä skannaamatonta syötettä jäsennin voi kurkistaa (engl. lookahead) etukäteen. (Chapman 1987, s. 33). Näin ollen yksinkertaisin versio LR-jäsentimestä on LR(0)-jäsennin, joka ei kurkista yhtäkään merkkiä etukäteen. Chapmanin (1987, s. 33) mukaan LR(0) jäsentimet käyttävät hyödyksi vain sitä tietoa mikä saadaan jo käsitellystä osasta syötettä.

Grunen ym. (2012, s.156) mukaan käytännössä hyödyllisiä tekniikoita LR-jäsentimien generoinnissa ovat LR(1) ja LALR(1) (lookahead LR(1)). Muita LR-jäsentimien generointiin olevia algoritmeja ovat muun muassa LR(0) ja SLR(1). Näistä lisää luvussa 4.2.

LR-jäsentimet generoidaan erilaisilla jäsennin generaattoreilla. Tunnetuimpia bottom-up-jäsennin generaattoreita ovat yacc (Yet Another Compiler Compiler) ja sitä asteittain korvaava GNU:n jakelema bison (Grune ym. 2012, s.191). Yacc generoi LALR(1) algoritmin mukaisesti LR-jäsentimiä.

3.1 LR-jäsennyksen historiasta

Chapmanin (1987, s. 4) mukaan ensimmäiset yritykset syntaksin tulkintaan tietokoneilla olivat aritmeettisten lauseiden tulkintaa. Tälläisten aritmeettisten lauseiden tulkinnassa tietokoneilla oli Chapmanin mukaan ongelmana eri operaattoreiden keskinäinen järjestys eli presedenssi. Chapman (1987, s. 4) kirjoittaa, että isoja edistysaskeleita saatiin kun, peräkkäisalgoritmin (sequential algorithm) keksiminen vuonna 1952 mahdollisti lausekkeiden osittaisen parsimisen koko lausekkeen sijaan.

Vuonna 1957 kontekstittomien kielioppien kuvaksen julkaisu Chomskyn toimesta ja niiden hyödyllisyyden tajuaminen johti Algol 60 -raportin uusittuun painokseen, jossa käytettiin Backus Naur Form -muotoa (BNF), joka on juontaa juurensa kontekstittomiin kielioppeihin. (Chapman 1987, s. 5). Chapmanin mukaan juuri BNF-muodon käyttö Algol 60 -raportissa toi paljon julkisuutta kontekstittomien kielioppien käsitteelle.

Vuonna 1965 Donald E. Knuth julkaisi tutkimuksen ”On the Translation of Languages from Left to Right”, jossa määritteli LR(k) kieliopit yleisimpinä sen tyyppisistä kieliopeista. Paperiinsa Knuth sisällytti myös algoritmit sen tutkimiseen, noudattaako jokin kielioppi ehtoja LR(k) jollain annetulla k:lla sekä metodeja LR(k) kielioppien tunnistimien kehittämiseen. (Knuth 1965)

3.2 LR-jäsennyksen teoria

On olemassa useita erilaisia LR-jäsennysalgoritmeja. Chapmanin (1987, s. 78) mukaan käytännöllisin näistä on LALR(k) (lookahead LR(k)), joka on perinteistä LR(k)-algoritmia huomattavasti kevyempi, mutta SLR(1)-algoritmia (simple LR(1)) ilmaisuvoimaisempi. LR-jäsentimet yleisesti koostuvat syötemerkkijonosta (engl. input), pinosta (engl. stack), driver programista ja jäsennystaulukosta (engl. parsing table), joka sisältää kohdat action ja goto, lisäksi on myös tuloste (engl. output), jonka jäsennin antaa syötteelle (Aho, Sethi ja Ullman 2007, s.216). Yllämainitut algoritmit siis ovat jäsentimien luomiseen. Varsinainen jäsennys toimii samalla lailla riippumatta tavasta, jolla se on luotu, sillä LR-jäsentimet eroavat toisistaan vain jäsennystaulukkojen suhteen (Aho, Sethi ja Ullman 2007, s. 216).

Palataan nyt tämän luvun alussa mainittuihin shift ja reduce toimintoihin. Shift n toimii niin, että siirretään lukupäätä n merkkiä oikealle ja reduce, niin että supistetaan merkkejä produktioiden mukaisesti. (Aho, Sethi ja Ullman 2007, s. 216-217). Lukupäällä tässä yhteydessä tarkoitetaan kohtaa, johon asti olevat syötemerkit on käsitelty. Aho, Sethi ja Ullman (2007, s. 216-220) mukaan LR-jäsentimen pinossa säilytetään senhetkinen tilan ja supistamattomat välikesymbolit, eli kun siirretään (shift) lukupäätä laitetaan pinon päällimäiseksi ne syötemerkit joiden yli lukupää siirretään.

3.3 LR-jäsennyksen toiminta ja esimerkki

Seuraavaksi määritellään uusi kielioppi ja käydään sille ja jollekin sen kielelle läpi, se miten LR-jäsennin käsittelisi sen. Nyt kun kyseessä on jäsennin, aloitetaan varsinaisesta merkkijonosta ja yritetään päästä takaisin kieliopin lähtösymboliin. Mikäli näin voidaan tehdä, tullaan todistaneeksi, että merkkijono kuuluu kieleen.

$$
\begin{aligned}
S &\rightarrow E \\
E &\rightarrow E - T \\
E &\rightarrow T \\
T &\rightarrow T * F \\
T &\rightarrow F \\
F &\rightarrow (E) \\
F &\rightarrow a
\end{aligned}
$$

Taulukossa 1 on riviltä 2 alkaen riveittäin eroteltu jäsennyksen vaiheet. Aluksi pino on tyhjä ja syötteeseen ei ole koskettu, jos jäsennys onnistuu, eli päättyy accept toimintoon, on koko syöte käsitelty. Jos syöte ei ole kieliopin mukainen päättyy jäsennys decline toimintoon, vaikka koko syötettä ei olisi välttämättä vielä käsitelty. Pitää kuitenkin huomata, että käytännön toteutuksissa ei pinossa tarvitsisi olla kieliopin merkkejä vaan siellä olisi tiloja, joissa jäsennin on.

Taulukon 1 kolmannessa sarakkeessa on aina seuraava suoritettava toiminto. Jokainen toiminto eroteltu erikseen niin, että esimerkiksi kaikki lukupään siirrot on eritelty yksittäisiksi

Taulukko 1. LR-jäsennyksen toiminta

Pino	**Syöte**	**Toiminto**
	$a-a*a$	shift
a	$-a*a$	reduce $F \rightarrow a$
F	$-a*a$	reduce $T \rightarrow F$
T	$-a*a$	reduce $E \rightarrow T$
E	$-a*a$	shift
$E-$	$a*a$	shift
$E-a$	$*a$	reduce $F \rightarrow a$
$E-F$	$*a$	reduce $T \rightarrow F$
$E-T$	$*a$	shift
$E-T*$	a	shift
$E-T*a$		reduce $F \rightarrow a$
$E-T*F$		reduce $T \rightarrow T*F$
$E-T$		reduce $E \rightarrow E-T$
E		reduce $S \rightarrow E$
S		accept

toiminnoiksi eikä ole siirretty useamman merkin yli kerralla. Lisäksi on huomattava, että ”reduce $F \rightarrow a$” tarkoittaa a korvaamista F:llä eikä toisinpäin, koska tarkoitus on päästä aloitussymboliin S.

4 LR-jäsennyksen käyttökohteet

Tässä luvussa esitellään pintapuolisesti erilaisia tapoja hyödyntää LR-jäsennystä. Yleisin käyttökohde LR-jäsennykselle on luonnollisesti kääntäjissä, sitä käsitellään alaluvussa 4.1. Alaluvussa 4.2 puolestaan tehdään pieni katsaus LR-jäsennyksen käytännön hyödyllisyyteen. Hyödyllisyyden osalta käydään suurimmaksi osaksi läpi LALR(1) ja GLR tekniikoita, jotka ovat eräitä käytetyimmistä. Näitä aiheita ei käsitellä laajasti, vaan pelkästään sen verran, että annetaan yleiskuva mihin tarkoituksiin tässä työssä esiteltyjä työkaluja voi oikeasti käyttää.

4.1 LR-jäsennys kääntäjissä

LR-jäsennyksen periaatteet ovat käytännöllisiä erilaisissa kääntäjätoteutuksissa. Eräs erityisen hyödyllinen asia LR-jäsennyksessä ja muissa luvussa 2.2 mainitussa bottom-up jäsennystavoissa on se, että vasen rekursio (engl. left recursion) ei ole ongelma, toisin kuin luvussa 2.4 esitetyssä LL-jäsennyksessä. (Grune ym. 2012, s. 158-159). Eräs toinen syy miksi LR-jäsentimiä käytetään kääntäjissä, on Ahon (1974) mukaan, se että LR-jäsennys on tehokas ja hyvin sopiva vaihtoehto toteuttamaan kääntäjissä syntaksin tarkistuksen, eli toteuttaako kirjoitettu ohjelmakoodi kyseisen ohjelmointikielen syntaksin.

4.2 LR-jäsennyksen hyödyllisyys käytännössä

Grune ym. (2012, s. 158) mainitsee LR-jäsennys algoritmeista useita, mutta pitää käyttökelpoisena vain LALR(1) algoritmia. Muita mainittuja algoritmeja ovat LR(0), SLR(1) ja LR(1). LR(0) ja SLR(1) ovat teholtaan liian heikkoja ollakseen hyödyllisiä, vaikka erityisesti LR(0)-algoritmilla onkin teoreettista merkitystä. Grunen ym.(2012, s.158) mielestä LR(1) taas on puolestaan liian muistia kuluttava ollakseen erityisen käytännöllinen, vaikka se onkin tehokas.

Vaikka tavallisen LR-jäsennyksen periaatteet ovatkin hyödyllisiä, ei kaikilla kontekstittomilla kielillä ole LALR(1) kielioppeja (McPeak ja Necula 2004). Tätä varten on GLR-

jäsentimet, jotka McPeakin ja Neculan (2004) mukaan kohdatessaan shift/reduce konfliktin, jakavat pinon ja tutkivat kaikki vaihtoehdot rinnakkain. McPeakin ja Neculan mielestä GLR on teknologiana paljon käytännöllisempi kuin LALR sillä se ei rajoita kielioppeja. Koska kielioppeja ei ole rajattu voidaan GLR-jäsennystä testata erilaisilla protyypeillä ja näin ollen voidaan myös kehittää kielioppia kohti deterministisyyttä ja näin tehostaa huomattavasti jäsentimen toimintaa (McPeak ja Necula 2004). Tärkeä huomio teollisen kehityksen kannalta on juuri tämä McPeakin ja Neculan mainitsema protojen kehittely. On tärkeää, että prosessin aikana voidaan testata työkalua, myös ennen kuin se on täysin valmis johon GLR antaa mahdollisuuden.

5 Yhteenveto

Iso osa työstä koostui erilaisten teorioiden ja käsitteiden määrittelystä, jolla luotiin pohjaa varsinaiselle aiheelle. Kontekstittomiin kielioppeihin esittelyyn esimerkiksi käytettiin iso osa luvusta 2, sillä ne ovat erittäin merkittävä osa paitsi LR-jäsennystä myös jäsennystä yleisesti. Esimerkkejä on käytetty myös runsaasti helpottamaan kulloisenkin aiheen hahmottamista.

LR-jäsennyksen toiminta itsessään on kohtuullisen ymmärrettävää, jos esitiedot ovat kohdallaan. Haastavaa aiheesta tekevät monet erilaiset algoritmit ja tekniikat varsinaisten jäsentimien luontiin. Näitä tekniikoita on sivuttu luvuissa 3 ja 4, mutta pääpaino on pidetty kuitenkin LR-jäsennyksen toiminnan ympärillä. LR-jäsennyksen toiminta ja teoria onkin pyritty esittämään yksityiskohtaisesti.

Kielioppien ja jäsennyksen teorian jälkeen erityisesti luvussa 4 käsitelty LR-jäsennyksen käyttö oli aiheena erityisen mielenkiintoinen, joskin moniselitteinen siltä osin, että ei ole yhtä oikeaa tapaa luoda LR-jäsentimiä tai yhtä ainutta tilannetta jossa niitä voi käyttää. LR-jäsennyksen eri variaatiot ovat hyödyllisiä monissa eri tilanteissa, mutta ei ole yhtä oikeaa ratkaisua kaikkiin tapauksiin. Determinismi kielissä on tärkeä tekijä, joka tulee huomioida valittaessa jäsennysalgoritmia. Kun kyseessä on kieli jolle on esimerkiksi LALR(1) kielioppi on helppo generoida jäsennin esimerkiksi Bisonin avulla, mutta universaalia ratkaisua kaikille kielille ei vielä ole ja siksi jäsennysongelma pysyy edelleen ratkaisemattomana.

Lähteet

Aho, A. V., ja S. C. Johnson. 1974. “LR Parsing”. *ACM Comput. Surv.* (New York, NY, USA) 6, numero 2 (kesäkuu): 99–124. ISSN: 0360-0300. doi:10.1145/356628.356629.

Aho, Alfred V., Ravi Sethi ja Jeffrey D. Ullman. 2007. *Compilers : principles, techniques, & tools* [kielellä eng]. 2nd ed. Boston, MA: Pearson / Addison Wesley.

Aycock, John, ja Nigel Horspool. 1999. “Faster Generalized LR Parsing”. Teoksessa *Compiler Construction,* toimittanut Stefan Jähnichen, 32–46. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 9783-540490517.

Aycock, John, Nigel Horspool, Jan Janoušek ja Bořivoj Melichar. 2001. “Even faster generalized LR parsing”. ID: Aycock2001, *Acta Informatica* 37 (9): 633–651. doi:10.1007/PL00013319.

Chapman, Nigel P. 1987. *LR parsing : theory and practice* [kielellä eng]. Cambridge: Cambridge University Press.

Earley, Jay. 1970. “An Efficient Context-free Parsing Algorithm”. *Commun. ACM* (New York, NY, USA) 13, numero 2 (helmikuu): 94–102. ISSN: 0001-0782. doi:10.1145/362007.362035.

Ford, Bryan. 2002. “Packrat Parsing: Simple, Powerful, Lazy, Linear Time, Functional Pearl”. Teoksessa *Proceedings of the Seventh ACM SIGPLAN International Conference on Functional Programming,* 36–47. doi:10.1145/581478.581483.

Grune, Dick, Kees van Reeuwijk, Henri E. Bal, Ceriel J. H. Jacobs ja Koen Langendoen. 2012. *Modern Compiler Design* [kielellä eng]. 2nd ed. 2012. New York, NY: Springer New York. http://dx.doi.org/10.1007/978-1-4614-4699-6.

Knuth, Donald E. 1965. “On the translation of languages from left to right” [kielellä eng]. *Information and Control* 8 (6): 607–639. doi:10.1016/S0019-9958(65)90426-2.

McPeak, Scott, ja George C. Necula. 2004. “Elkhound: A Fast, Practical GLR Parser Generator”. Teoksessa *Compiler Construction,* toimittanut Evelyn Duesterwald, 73–88. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-24723-4.

Parr, Terence, ja Kathleen Fisher. 2011. “LL(*): The Foundation of the ANTLR Parser Generator”. *SIGPLAN Not.* (New York, NY, USA) 46, numero 6 (kesäkuu): 425–436. ISSN: 0362-1340. doi:10.1145/1993316.1993548. http://doi.acm.org/10.1145/1993316.1993548.