UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Ivan Ristović

KREIRANJE ZAJEDNIČKE AST APSTRAKCIJE ZA RAZLIČITE PROGRAMSKE JEZIKE

master rad

Beograd, 2020.

Mentor:

doc. dr Milena Vujošević-Janičić
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Ana Anić
University of Disneyland, Nedođija

dr Laza Lazić
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

TODO zahvalnica

Naslov master rada: Kreiranje zajedničke AST apstrakcije za različite programske jezike

Rezime: Fijuče vetar u šiblju, ledi pasaže i kuće iza njih i gunđa u odžacima. Nidžo, čežnjivo gledaš fotelju, a Đura i Mika hoće poziciju sebi. Ljudi, jazavac Džef trči po šumi glođući neko suho žbunje. Ljubavi, Olga, hajde pođi u Fudži i čut ćeš nježnu muziku srca. Boja vaše haljine, gospođice Džafić, traži da za nju kulučim. Hadži Đera je zaćutao i bacio čežnjiv pogled na šolju s kafom. Džabe se zec po Homolju šunja, čuvar Jožef lako će i tu da ga nađe. Odžačar Filip šalje osmehe tuđoj ženi, a njegova kuća bez dece. Butić Đuro iz Foče ima pun džak ideja o slaganju vaših željica. Džajić odskoči u aut i izbeže đon halfa Pecelja i njegov šamar. Plamte odžaci fabrika a čađave guje se iz njih dižu i šalju noć. Ajšo, lepoto i čežnjo, za ljubav srca moga, dođi u Hadžiće na kafu. Huči šuma, a iza žutog džbuna i panja đak u cveću delje seji frulu. Goci i Jaćimu iz Banje Koviljače, flaša džina i žeđ padahu u istu uru. Džaba što Feđa čupa za kosu Milju, ona juri Živu, ali njega hoće i Daca. Dok je Fehim u džipu žurno ljubio Zagu Čađević, Cile se ušunjao u auto. Fijuče košava nad odžacima a Ilja u gunju žureći uđe u suhu i toplu izbu. Bože, džentlmeni osećaju fizičko gađenje od prljavih šoljica! Dočepaće njega jaka šefica, vođena ljutom srdžbom zlih žena. Pazi, gedžo, brže odnesi šefu taj đavolji ček: njim plaća ceh. Fine džukce ozleđuje bič: odgoj ih pažnjom, strpljivošću. Zamišljao bi kafedžiju vlažnih prstića, crnjeg od čađi. Đače, uštedu plaćaj žaljenjem zbog džinovskih cifara. Džikljaće žalfija između tog busenja i peščanih dvoraca. Zašto gđa Hadžić leči živce: njena ljubav pred fijaskom? Jež hoće peckanjem da vređa ljubičastog džina iz flaše. Džej, ljubičast zec, laže: gađaće odmah pokvašen fenjer. Plašljiv zec hoće jeftinu dinju: grožđe iskamči džabe. Džak je pun žica: ćućeš tad svađu zbog lomljenja harfe. Čuj, džukac Flop bez daha s gađenjem žvaće stršljena. Oh, zadnji šraf na džipu slab: muž gđe Cvijić ljut koči. Šef džabe zvižduće: mlađi hrt jače kljuca njenog psa. Odbaciće kavgadžija plaštom čađ u željezni fenjer. Deblji krojač: zgužvah smeđ filc u tanjušni džepić. Džo, zgužvaćeš tiho smeđ filc najdeblje krpenjače. Štef, bacih slomljen dečji zvrk u džep gđe Žunjić. Debljoj zgužvah smeđ filc — njen škrt džepčić.

Ključne reči: TODO

Sadržaj

Glava 1

Uvod

Fijuče vetar u šiblju, ledi pasaže i kuće iza njih i gunđa u odžacima. Nidžo, čežnjivo gledaš fotelju, a Đura i Mika hoće poziciju sebi. Ljudi, jazavac Džef trči po šumi glođući neko suho žbunje. Ljubavi, Olga, hajde pođi u Fudži i čut ćeš nježnu muziku srca. Boja vaše haljine, gospođice Džafić, traži da za nju kulučim. Hadži Đera je zaćutao i bacio čežnjiv pogled na šolju s kafom. Džabe se zec po Homolju šunja, čuvar Jožef lako će i tu da ga nađe. Odžačar Filip šalje osmehe tuđoj ženi, a njegova kuća bez dece. Butić Đuro iz Foče ima pun džak ideja o slaganju vaših željica. Džajić odskoči u aut i izbeže đon halfa Pecelja i njegov šamar. Plamte odžaci fabrika a čađave guje se iz njih dižu i šalju noć. Ajšo, lepoto i čežnjo, za ljubav srca moga, dođi u Hadžiće na kafu. Huči šuma, a iza žutog džbuna i panja đak u cveću delje seji frulu. Goci i Jaćimu iz Banje Koviljače, flaša džina i žeđ padahu u istu uru. Džaba što Feđa čupa za kosu Milju, ona juri Živu, ali njega hoće i Daca. Dok je Fehim u džipu žurno ljubio Zagu Čađević, Cile se ušunjao u auto. Fijuče košava nad odžacima a Ilja u gunju žureći uđe u suhu i toplu izbu. Bože, džentlmeni osećaju fizičko gađenje od prljavih šoljica! Dočepaće njega jaka šefica, vođena ljutom srdžbom zlih žena. Pazi, gedžo, brže odnesi šefu taj đavolji ček: njim plaća ceh. Fine džukce ozleđuje bič: odgoj ih pažnjom, strpljivošću. Zamišljao bi kafedžiju vlažnih prstića, crnjeg od čađi. Đače, uštedu plaćaj žaljenjem zbog džinovskih cifara. Džikljaće žalfija između tog busenja i peščanih dvoraca. Zašto gđa Hadžić leči živce: njena ljubav pred fijaskom? Jež hoće peckanjem da vređa ljubičastog džina iz flaše. Džej, ljubičast zec, laže: gađaće odmah pokvašen fenjer. Plašljiv zec hoće jeftinu dinju: grožđe iskamči džabe. Džak je pun žica: ćućeš tad svađu zbog lomljenja harfe. Čuj, džukac Flop bez daha s gađenjem žvaće stršljena. Oh, zadnji šraf na džipu slab:

muž gđe Cvijić ljut koči. Šef džabe zvižduće: mlađi hrt jače kljuca njenog psa. Odbaciće kavgadžija plaštom čađ u željezni fenjer. Deblji krojač: zgužvah smeđ filc u tanjušni džepić. Džo, zgužvaćeš tiho smeđ filc najdeblje krpenjače. Štef, bacih slomljen dečji zvrk u džep gđe Žunjić. Debljoj zgužvah smeđ filc — njen škrt džepčić.

Glava 2

Pregled relevantnih pojmova

U ovom poglavlju će biti opisani koncepti i alati čije je razumevanje potrebno kako bi se razumeo opis dalje apstrakcije i implementacije samog programa. Umesto analize samog sadržaja izvornog koda analizira se *apstraktno sintaksno stablo* (eng. *Abstract Syntax Tree*, u daljem tekstu *AST*), opisano u odeljku 2.1. Alat koji je korišćen za generisanje parsera za proizvoljnu gramatiku jezika se zove *Another Tool For Language Recognition* [1], u daljem tekstu *ANTLR*, opisan u odeljku 2.2. Parser generiše AST specifičan za datu gramatiku i nema sličnosti u dobijenim apstrakcijama za različite jezike. Kako bismo poredili stabla različitih jezika, kreiramo reprezentaciju na višem nivou i specifični AST podižemo na taj nivo. Ta reprezentacija će biti opisana u narednim poglavljima, kao i načini kako se ona može analizirati. Takođe, pojmovi specifični za implementaciju će takođe biti opisani u ovom poglavlju.

2.1 Apstraktna sintaksna stabla - AST

Kako bi se kod pisan u nekom programskom jeziku (*izvorni fajl*) preveo u kod koji će se izvršavati na nekoj mašini (*izvršivi fajl*), prevodilac prolazi kroz određene korake. Da bi se izvorni kod preveo, prevodilac mora da zna njegovu formu, ili *sintaksu*, i njegovo značenje, ili *semantiku*. Deo prevodioca koji određuje da li je izvorni kod ispravno formiran u terminima sintakse i semantike se naziva *prednji deo* (engl. *front end*). Ukoliko je izvorni kod ispravan, prednji deo kreira *međureprezentaciju* koda (engl. *intermediate representation*, u daljem tekstu *IR*). Ukoliko to nije slučaj, prevođenje ne uspeva i programeru se daje poruka o detaljima zašto prevođenje nije uspelo. [4]

Postupak rada prednjeg dela će biti opisan kroz konkretan primer. Pretpostavimo da želimo da prevedemo kod pisan u programskom jeziku C prikazan na slici 2.1. Primetimo da postoji greška u datom kodu - simbol `c` koji se koristi u dodeli u liniji 8 će biti prepoznat kao identifikator koji ne odgovara nijednoj deklarisanoj promenljivoj - stoga ne možemo prevesti ovaj kod. Ovo, doduše, nije sintaksna greška - izraz `a + c` je sasvim validan u programskom jeziku C bez analize konteksta u kom se javlja. Problem će postati očigledan tek nakon parsiranja izvornog koda i provere ispunjenosti sintaksih pravila, tačnije u fazi semantičke provere. Stoga se ovakve greške nazivaju *semantičke greške*, dok se greške u sintaksi nazivaju *sintaksne greške*.

```
1   #include<stdio.h>
2
3   #define T int
4
5   int main()
6   {
7       T a, b;
8       a = a + c;          // c nije deklarisano
9       printf("%d", a);
10      return 0;
11  }
```

Slika 2.1: Primer izvornog koda pisanog u programskom jeziku C.

Pre nego što prednji kraj prevodioca uopšte dobije kod koji treba prevesti, vrši se *pretprocesiranje* od strane programa koji se naziva *pretprocesor*. U fazi pretprocesiranja se izvode samo tekstualne operacije kao što su brisanje komentara ili zamena makroa u jezicima kao što je C. Rezultat rada pretprocesora za kod sa slike 2.1 bi izgledao kao na slici 2.2 [1].

Da bi proverio sintaksu izvornog koda, prevodilac mora da uporedi strukturu istog sa unapred definisanom strukturom za određeni programski jezik. Ovo zahteva formalnu definiciju sintakse jezika. Programski jezik možemo posmatrati kao skup *pravila* koji se naziva *gramatika* [3], prikazana na slici 2.3. U prednjem

[1] U nekim implementacijama C standardne biblioteke, moguće je da se poziv funckije `printf` zameni pozivom funkcije `fprintf` sa ispisom na `stdout`. U standardu se propisuje da funkcije kao što je `printf` mogu biti implementirane kao makroi. Izlaz na slici 2.2 je generisan od strane `GCC 7.4.0` po C11 standardu i ovo nije slučaj u datom okruženju.

```
1       # 1 "<stdin>"
2       # 1 "<built-in>"
3       # 1 "<command-line>"
4       # 31 "<command-line>"
5       # 1 "/usr/include/stdc-predef.h" 1 3 4
6
7       ...
8
9       extern char *ctermid (char *__s) __attribute__
           ((__nothrow__ , __leaf__));
10      # 840 "/usr/include/stdio.h" 3 4
11      extern void flockfile (FILE *__stream) __attribute__
           ((__nothrow__ , __leaf__));
12      extern int ftrylockfile (FILE *__stream) __attribute__
           ((__nothrow__ , __leaf__)) ;
13      extern void funlockfile (FILE *__stream) __attribute__
           ((__nothrow__ , __leaf__));
14      # 868 "/usr/include/stdio.h" 3 4
15      # 2 "<stdin>" 2
16      # 2 "<stdin>"
17
18      int main()
19      {
20          int a, b;
21          a = a + c;
22          printf("%d", a);
23          return 0;
24      }
```

Slika 2.2: Prikaz rezultata rada pretprocesora za izvorni kod sa slike 2.1. Pritom, prikazano je samo par linija sa početka i kraja izlaza pretprocesora - kod iznad `main` funkcije je uključen iz `stdio.h` zaglavlja.

delu se izvode dva procesa koji određuju da li ulaz zaista zadovoljava gramatiku određenog programskog jezika. Ova dva procesa se nazivaju *skeniranje* i *parsiranje*, a komponente prednjeg dela koje vrše te procese se nazivaju *skener* (takođe se naziva i *lekser*) i *parser*, redom.

Prilikom faze prevođenja, kako prevodilac ne bi radio nad sirovim karakterima izvornog koda, potrebno je izvršiti pripremu istog - skeniranje. Prevodilac ima u vidu moguće elemente programskog jezika, tzv. *tokene*, koje treba prepoznati u datom fajlu - ključne reči, operatore, promenljive itd. Proces prepoznavanja

```
1     functionDefinition
2         :   declarationSpecifiers? declarator
3             declarationList? compoundStatement
4         ;
5
6     declarationList
7         :   declaration
8         |   declarationList declaration
9         ;
10
11    declaration
12        :   declarationSpecifiers initDeclaratorList ';'
13        |   declarationSpecifiers ';'
14        |   staticAssertDeclaration
15        ;
```

Slika 2.3: Isečak gramatike programskog jezika C po standardu C11.

tokena u izvornom fajlu se naziva *tokenizacija*. Pojednostavljen primer tokena koje lekser pokušava da prepozna se može videti na slici 2.4. Primer izlaza leksera za izlaz pretprocesora sa slike 2.2 se može videti na slici 2.5. [2]

```
1     Identifier : IdentifierNondigit
2                  (IdentifierNondigit | Digit)*
3                ;
4
5     IdentifierNondigit : Nondigit
6                        | UniversalCharacterName
7                        ;
8
9     Nondigit : [a-zA-Z_]
10              ;
11
12    Digit : [0-9]
13          ;
```

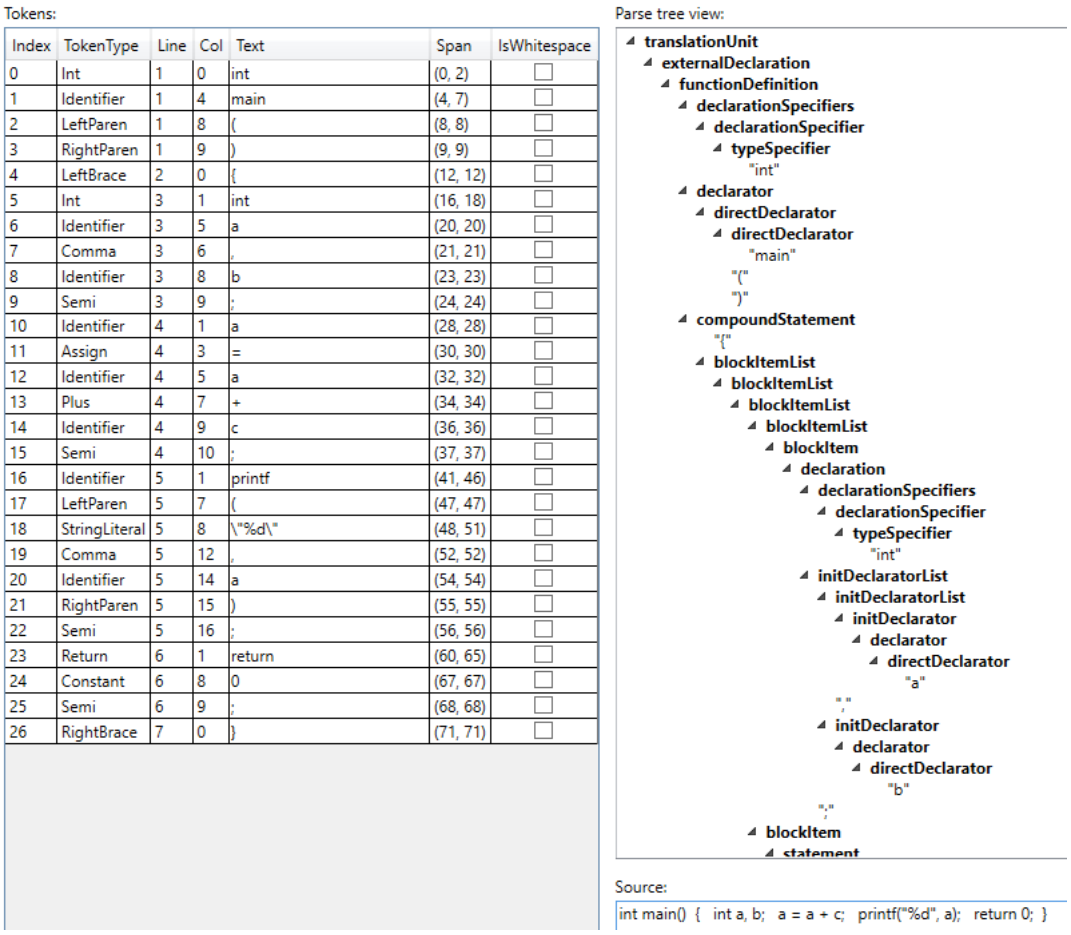
Slika 2.4: Primer delimične definicije tokena za ime promenljive po C11 standardu.

[2]Moderni kompajleri često nemaju odvojene faze u kojima se pozivaju skeniranja i parsiranja, već se skeniranje odvija paralelno sa fazom parsiranja. Međutim, to nas ne sprečava da ispišemo tokene onda kada se oni prepoznaju, i to je demonstrirano na slici 2.5.

```
1     identifier 'main'    [LeadingSpace] Loc=<sample.c:3:5>
2     l_paren '('     Loc=<sample.c:3:9>
3     r_paren ')'     Loc=<sample.c:3:10>
4     l_brace '{'  [StartOfLine]  Loc=<sample.c:4:1>
5     int 'int'    [StartOfLine] [LeadingSpace]
        Loc=<sample.c:5:5>
6     identifier 'a'   [LeadingSpace] Loc=<sample.c:5:9>
7     comma ','       Loc=<sample.c:5:10>
8     identifier 'b'   [LeadingSpace] Loc=<sample.c:5:12>
9     semi ';'        Loc=<sample.c:5:13>
10    identifier 'a'   [StartOfLine] [LeadingSpace]
        Loc=<sample.c:6:5>
11    equal '='    [LeadingSpace] Loc=<sample.c:6:7>
12    identifier 'a'   [LeadingSpace] Loc=<sample.c:6:9>
13    plus '+'     [LeadingSpace] Loc=<sample.c:6:11>
14    identifier 'c'   [LeadingSpace] Loc=<sample.c:6:13>
15    semi ';'        Loc=<sample.c:6:14>
16    identifier 'printf'  [StartOfLine] [LeadingSpace]
        Loc=<sample.c:7:5>
17    l_paren '('     Loc=<sample.c:7:11>
18    string_literal '"%d"'       Loc=<sample.c:7:12>
19    comma ','       Loc=<sample.c:7:16>
20    identifier 'a'   [LeadingSpace] Loc=<sample.c:7:18>
21    r_paren ')'     Loc=<sample.c:7:19>
22    semi ';'        Loc=<sample.c:7:20>
23    return 'return'  [StartOfLine] [LeadingSpace]
        Loc=<sample.c:8:5>
24    numeric_constant '0'     [LeadingSpace]
        Loc=<sample.c:8:12>
25    semi ';'        Loc=<sample.c:8:13>
26    r_brace '}'  [StartOfLine]  Loc=<sample.c:9:1>
27    eof ''      Loc=<sample.c:9:2>
```

Slika 2.5: Proces tokenizacije koda pisanog po C11 gramatici. Generisano uz pomoć `clang` [2] kompajlera.

Nakon faze skeniranja potrebno je parsirati dobijene tokene. Parser, imajući u vidu gramatiku jezika, pokušava da kreira *stablo parsiranja* (eng. *parse tree* ili *derivation tree*). Takvo stablo i dalje sadrži sve relevantne informacije o izvornom kodu. Vizuelni prikaz rada parsera za gramatiku sa slike C11 i izvonog koda sa slike 2.2 je dat na slici 2.6. Stablo parsiranja se koristi u narednim fazama prevođenja.

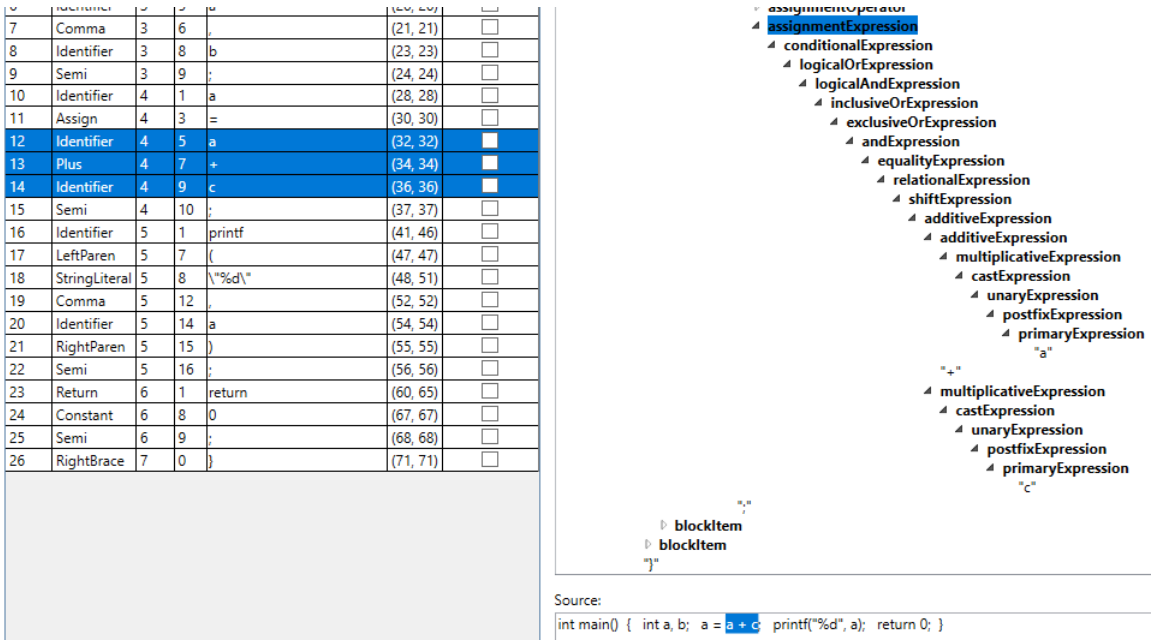


Slika 2.6: Prikaz stabla parsiranja koje generiše parser kreiran od strane alata ANTLR4 za C11 gramatiku. Vizualizacija prikazana pomoću dodatka za Visual Studio: `https://github.com/zspitz/ANTLR4ParseTreeVisualizer`

Za potrebe ovog rada, što se procesa prevođenja tiče, dovoljno je pozvnavanje prednjeg dela, stoga neće biti reči o daljim koracima u fazi prevođenja (semantička provera, optimizacije, generisanje IR). Zainteresovani čitalac može više detalja pronaći u [4] i [8].

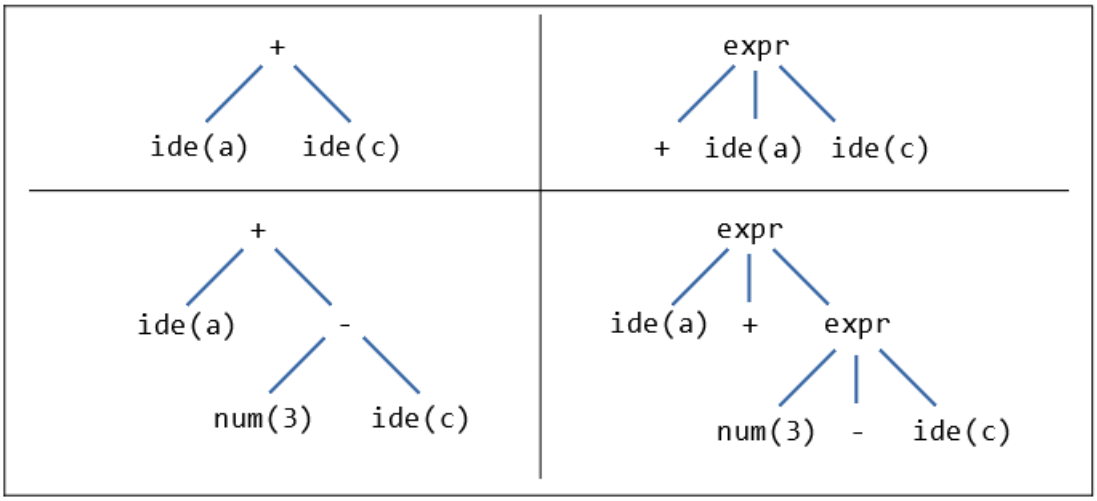
Stablo parsiranja sadrži sve informacije potrebne u fazi parsiranja uključujući detalje korisne samo za parser prilikom provere ispunjenosti gramatičkih pravila. Sa druge strane, *apstraktno sintaksno stablo* sadrži samo sintaksnu strukturu u jednostavnijoj formi. Na slici 2.7 se može videti koliko je stablo parsiranja komplikovano čak i za naizgled jednostavne aritmetičke izraze. Razlog ovolike komplikovanosti dolazi iz rekurzivnih pravila definisanih u C11 gramatici. Parseru su sve ove informacije neophodne ali na apstratknijem nivou nisu potrebne. Jedina

važna semantička odlika izraza `a + c` je ta da je to zbir vrednosti nekih promenljivih. Sve ostale informacije su nepotrebne. Na slici 2.8 možemo videti različita apstraktna sintaksna stabla za pomenuti izraz, ali takođe i za malo komplikovanije izraze. Podrazumeva se, naravno, da je ulaz već tokenizovan.



Slika 2.7: Prikaz kompleksnosti stabla parsiranja za izraz `a + c` u C11 gramatici. Vizualizacija prikazana pomoću dodatka za Visual Studio: `https://github.com/zspitz/ANTLR4ParseTreeVisualizer`

Uloga apstraktnih sintaksnih stabala je da pokažu semantiku strukture koda preko stabala. Kao što se vidi na slici 2.8, postoji određeni nivo slobode u dizajniranju ovih stabala. Generalno, *terminalni simboli*, simboli koji predstavljaju listove stabla parsera, koji odgovaraju operatorima i naredbama se podižu naviše i postaju koreni podstabala, dok se njihovi operandi ostavljaju kao njihovi potomci u stablu. Desna stabla sa slike ne prate u potpunosti ovaj princip, ali se takođe koriste zbog regularnosti izraza - recimo ukoliko binarni izraz posmatramo kao koncept, mnogo je lakše raditi sa ovakvom strukturom. Ovakva struktura će biti korišćena kasnije u implementaciji programa. Primetimo takođe da se u stablima za izraz `a + (3 - c)` (dole) implicitno sačuvala informacija o prioritetu operacije oduzimanja u izrazu. Jasno je, dakle, da se računanje vrednosti aritmetičkih izraza onda vrši kretanjem od listova stabla ka korenu. Takođe, pošto su apstraktna sintaksna stabla apstrakcija stabla parsiranja, više istih izraza jezika može ima-



Slika 2.8: Varijante apstraktnih sintaksnih stabala bez regularnosti (levo) i sa regularnošću (desno) za izraze `a + c` (gore) i `a + (3 - c)` (dole).

ti isto apstraktno sintaksno stablo ali različito stablo parsiranja; na primer, ako razmatramo izraz `(a + 5) - x / 2` i izraz `a + 5 - (x / 2)`.

Apstrakna sintaksna stabla će u daljem tekstu biti referisana skraćenicom *AST*, koja dolazi od engleskog naziva *Abstract Syntax Trees*. Takođe, reči samom dizajnu i tipovima čvorova AST-ova korišćenih u implementaciji će biti u poglavlju **??**. U narednom poglavlju će više biti reči o alatu koji je korišćen za kreiranje parsera za C11 gramatiku (ali i za druge, proizvoljne gramatike), koji daje mogućnost jednostavnog obilaska stabla parsiranja i pruža intuitivan način za izvršavanje logike nad tim stablom, što uključuje i izradu AST-a od stabla parsiranja.

2.2 ANTLR

Pretpostavljajući da imamo gramatiku proizvoljnog programskog jezika, postavlja se pitanje: *Da li je moguće definisati postupak i zatim napraviti program koji će generisati kodove leksera i parsera napisane u određenom programskom jeziku za proizvoljnu gramatiku datu na ulazu?*. Odgovor je potvrdan i postoji veliki broj alata koji se mogu koristiti u ove svrhe, od kojih je navedeno par njih:

- *GNU Bison* [6]
  GNU Bison je generator parsera i deo GNU projekta [7], često referisan samo kao *Bison*. Bison generiše parser na osnovu korisnički definisanih kontekst-

no slobodnih gramatika [3], upozoravajući pritom na dvosmislenosti prilikom parsiranja ili nemogućnost primena gramatičkih pravila. Generisani parser je najčešće C a ređe C++ kod, mada se u vreme pisanja ovog rada eksperimentiše sa Java podrškom. Generisani kodovi su u potpunosti prenosivi i ne zahtevaju specifične kompajlere. Bison može da, osim podrazumevanih *LALR(1)* [?] parsera, generiše i kanoničke *LR* [?], *IELR(1)* [?] i *GLR* [?] parsere.

- *Flex* [5]
  Kreiran kao alternativa *lex*-u [?], Flex generiše samo leksere pa se stoga najčešće koristi u kombinaciji sa drugim alatima koji mogu da generišu parsere, kao što je *BYACC*, opisan u nastavku.

- *BYACC* [?]
  Berkeley YACC, skraćeno *BYACC*, pisan po ANSI C standardu i otvorenog koda, se smatra od strane mnogih kao *najbolja varijanta YACC-a* [?]. BYACC dozvoljava tzv. *reentrant* kod - memorija je deljenja između poziva pa je bezbedno konkurentno izvršavanje koda - na način kompatibilan sa Bison-om i to je delom razlog njegove popularnosti.

- *ANTLR* [1]
  Another Tool for Language Recognition, ili kraće *ANTLR*, je generator *LL(*)* [?] leksera i parsera pisan u programskom jeziku Java sa intuitivnim interfejsom za obilazak stabla parsiranja. Verzija 3 podržava generisanje parsera u jezicima Ada95, ActionScript, C, C#, Java, JavaScript, Objective-C, Perl, Python, Ruby, i Standard ML, dok verzija 4 u vreme pisanja ovog rada samo generiše parsere u narednim jezicima: Java, C#, C++, JavaScript, Python, Swift i Go.

ANTLR, verzije 4, je izabran u ovom radu zbog svoje jednostavnosti, intuitivnosti i podrške za mnoge moderne programske jezike. Verzija 4 je izabrana umesto verzije 3 po preporuci autora, na osnovu eksperimentalne analize brzine i pouzdanosti verzije 4 u odnosu na verziju 3. Lekseri i parseri za ulazne gramatike će u implementaciji biti generisani u C#-u.

Preduslovi za pokretanje ANTLR4

Kako bi ANTLR generisao parser u proizvoljnom programskom jeziku, potrebno je instalirati ANTLR i imati *Java Runtime Environment* (skr. *JRE*) instaliran na sistemu i dostupan globalno pokretanjem putem komande `java`. Instalacija se sastoji od preuzimanja najnovijeg *.jar* fajla [3], sa zvanične stranice [1] ili recimo korišćenjem *curl* alata:

```
1 $ curl -O http://www.antlr.org/download/antlr-4-complete.jar
```

Na UNIX sistemima moguće je kreirati alias `antlr4` ili *shell* skript unutar direktorijuma `/usr/local/bin` sa imenom `antlr4` koji će pokrenuti *.jar* fajl na sledeći način (pretpostavljajući da se *.jar* fajl nalazi u direktorijumu `/usr/local/lib`):

```
1 #!/bin/sh
2 java -cp "/usr/local/lib/antlr4-complete.jar:$CLASSPATH"
     org.antlr.v4.Tool $*
```

Na Windows sistemima moguće je kreirati *batch* skript sa imenom `antlr4.bat` koji će pokrenuti ANTLR4, na sledeći način (pretpostavljajući da se *.jar* fajl nalazi u direktorijumu `C:\lib`):

```
1 java -cp C:\lib\antlr-4-complete.jar;%CLASSPATH%
     org.antlr.v4.Tool %*
```

Ukoliko su aliasi ili skript fajlovi imenovani kao iznad, moguće je iz komandne linije pojednostavljeno pokretati ANTLR4:

```
1 $ antlr4
2 ANTLR Parser Generator Version 4.0
3 -o ___    specify output directory where all output is
     generated
4 -lib ___  specify location of .tokens files
5 ...
```

[3] Takođe je moguće kompajlirati izvorni kod dostupan na servisu GitHub `https://github.com/antlr/antlr4`

Generisanje parsera za proizvoljnu gramatiku koristeći ANTLR

Glava 3

Zaključak

Fijuče vetar u šiblju, ledi pasaže i kuće iza njih i gunđa u odžacima. Nidžo, čežnjivo gledaš fotelju, a Đura i Mika hoće poziciju sebi. Ljudi, jazavac Džef trči po šumi glođući neko suho žbunje. Ljubavi, Olga, hajde pođi u Fudži i čut ćeš nježnu muziku srca. Boja vaše haljine, gospođice Džafić, traži da za nju kulučim. Hadži Đera je zaćutao i bacio čežnjiv pogled na šolju s kafom. Džabe se zec po Homolju šunja, čuvar Jožef lako će i tu da ga nađe. Odžačar Filip šalje osmehe tuđoj ženi, a njegova kuća bez dece. Butić Đuro iz Foče ima pun džak ideja o slaganju vaših željica. Džajić odskoči u aut i izbeže đon halfa Pecelja i njegov šamar. Plamte odžaci fabrika a čađave guje se iz njih dižu i šalju noć. Ajšo, lepoto i čežnjo, za ljubav srca moga, dođi u Hadžiće na kafu. Huči šuma, a iza žutog džbuna i panja đak u cveću delje seji frulu. Goci i Jaćimu iz Banje Koviljače, flaša džina i žeđ padahu u istu uru. Džaba što Feđa čupa za kosu Milju, ona juri Živu, ali njega hoće i Daca. Dok je Fehim u džipu žurno ljubio Zagu Čađević, Cile se ušunjao u auto. Fijuče košava nad odžacima a Ilja u gunju žureći uđe u suhu i toplu izbu. Bože, džentlmeni osećaju fizičko gađenje od prljavih šoljica! Dočepaće njega jaka šefica, vođena ljutom srdžbom zlih žena. Pazi, gedžo, brže odnesi šefu taj đavolji ček: njim plaća ceh. Fine džukce ozleđuje bič: odgoj ih pažnjom, strpljivošću. Zamišljao bi kafedžiju vlažnih prstića, crnjeg od čađi. Đače, uštedu plaćaj žaljenjem zbog džinovskih cifara. Džikljaće žalfija između tog busenja i peščanih dvoraca. Zašto gđa Hadžić leči živce: njena ljubav pred fijaskom? Jež hoće peckanjem da vređa ljubičastog džina iz flaše. Džej, ljubičast zec, laže: gađaće odmah pokvašen fenjer. Plašljiv zec hoće jeftinu dinju: grožđe iskamči džabe. Džak je pun žica: ćućeš tad svađu zbog lomljenja harfe. Čuj, džukac Flop bez daha s gađenjem žvaće stršljena. Oh, zadnji šraf na džipu slab:

muž gđe Cvijić ljut koči. Šef džabe zvižduće: mlađi hrt jače kljuca njenog psa. Odbaciće kavgadžija plaštom čađ u željezni fenjer. Deblji krojač: zgužvah smeđ filc u tanjušni džepić. Džo, zgužvaćeš tiho smeđ filc najdeblje krpenjače. Štef, bacih slomljen dečji zvrk u džep gđe Žunjić. Debljoj zgužvah smeđ filc — njen škrt džepčić.

Fijuče vetar u šiblju, ledi pasaže i kuće iza njih i gunđa u odžacima. Nidžo, čežnjivo gledaš fotelju, a Đura i Mika hoće poziciju sebi. Ljudi, jazavac Džef trči po šumi glođući neko suho žbunje. Ljubavi, Olga, hajde pođi u Fudži i čut ćeš nježnu muziku srca. Boja vaše haljine, gospođice Džafić, traži da za nju kulučim. Hadži Đera je zaćutao i bacio čežnjiv pogled na šolju s kafom. Džabe se zec po Homolju šunja, čuvar Jožef lako će i tu da ga nađe. Odžačar Filip šalje osmehe tuđoj ženi, a njegova kuća bez dece. Butić Đuro iz Foče ima pun džak ideja o slaganju vaših željica. Džajić odskoči u aut i izbeže đon halfa Pecelja i njegov šamar. Plamte odžaci fabrika a čađave guje se iz njih dižu i šalju noć. Ajšo, lepoto i čežnjo, za ljubav srca moga, dođi u Hadžiće na kafu. Huči šuma, a iza žutog džbuna i panja đak u cveću delje seji frulu. Goci i Jaćimu iz Banje Koviljače, flaša džina i žeđ padahu u istu uru. Džaba što Feđa čupa za kosu Milju, ona juri Živu, ali njega hoće i Daca. Dok je Fehim u džipu žurno ljubio Zagu Čađević, Cile se ušunjao u auto. Fijuče košava nad odžacima a Ilja u gunju žureći uđe u suhu i toplu izbu. Bože, džentlmeni osećaju fizičko gađenje od prljavih šoljica! Dočepaće njega jaka šefica, vođena ljutom srdžbom zlih žena. Pazi, gedžo, brže odnesi šefu taj đavolji ček: njim plaća ceh. Fine džukce ozleđuje bič: odgoj ih pažnjom, strpljivošću. Zamišljao bi kafedžiju vlažnih prstića, crnjeg od čađi. Đače, uštedu plaćaj žaljenjem zbog džinovskih cifara. Džikljaće žalfija između tog busenja i peščanih dvoraca. Zašto gđa Hadžić leči živce: njena ljubav pred fijaskom? Jež hoće peckanjem da vređa ljubičastog džina iz flaše. Džej, ljubičast zec, laže: gađaće odmah pokvašen fenjer. Plašljiv zec hoće jeftinu dinju: grožđe iskamči džabe. Džak je pun žica: ćućeš tad svađu zbog lomljenja harfe. Čuj, džukac Flop bez daha s gađenjem žvaće stršljena. Oh, zadnji šraf na džipu slab: muž gđe Cvijić ljut koči. Šef džabe zvižduće: mlađi hrt jače kljuca njenog psa. Odbaciće kavgadžija plaštom čađ u željezni fenjer. Deblji krojač: zgužvah smeđ filc u tanjušni džepić. Džo, zgužvaćeš tiho smeđ filc najdeblje krpenjače. Štef, bacih slomljen dečji zvrk u džep gđe Žunjić. Debljoj zgužvah smeđ filc — njen škrt džepčić.

Literatura

[1] ANTLR4. https://www.antlr.org/.

[2] Clang. https://clang.llvm.org/.

[3] Context-Free Grammars. https://www.cs.rochester.edu/~nelson/courses/csc_173/grammars/cfg.html.

[4] Keith Cooper and Linda Torczon. *Engineering a Compiler, 2nd Edition*. 2013.

[5] Flex. https://www.gnu.org/software/flex/.

[6] GNU Bison. https://www.gnu.org/software/bison/.

[7] GNU Project. https://www.gnu.org/gnu/thegnuproject.en.html.

[8] William M. Waite and Gerhard Goos. *Compiler Construction*. 1995.

Biografija autora

Ivan Ž. Ristović rođen je 17.01.1995. godine u Užicu. Osnovnu školu, kao i prirodno-matematički smer Užičke gimnazije, završio je kao nosilac Vukove diplome. Tokom navedenog perioda školovanja isticao se u oblastima matematike, informatike, fizike, hemije i engleskog jezika, što potvrđuje veći broj nagrada na Državnim takmičenjima.

Smer Informatika na Matematičkom fakultetu Univerziteta u Beogradu upisuje 2014. godine. Na navedenom smeru je diplomirao 2018. godine, posle tri godine studija sa prosečnom ocenom 9,17. Master studije upisuje na istom fakultetu odmah nakon diplomiranja.

U avgustu 2018. biva izabran u zvanje „Saradnik u nastavi“ na Matematičkom fakultetu paralelno sa master studijama. Drži vežbe iz kurseva „Računarske mreže”, „Funkcionalno programiranje”, „Programske paradigme” i „Objektno orijentisano programiranje” na kasnijim godinama osnovnih studija.

Oblasti interesovanja uključuju pre svega razvoj i verifikaciju softvera, mikroservise i računarske mreže.