Etape prevođenja Haskela do mašinskog jezika

Marija Mijailović, Miroslav Mišljenović, Nemanja Antić, Filip Lazić

5. maj 2018

Sadržaj

Glava 1

Uputstva

Prilikom predavanja odgovora na recenziju, obrišite ovo poglavlje.

Neophodno je odgovoriti na sve zamerke koje su navedene u okviru recenzija. Svaki odgovor pišete u okviru okruženja `\odgovor`, kako bi vaši odgovori bili lakše uočljivi.

1. Odgovor treba da sadrži na koji način ste izmenili rad da bi adresirali problem koji je recenzent naveo. Na primer, to može biti neka dodata rečenica ili dodat pasus. Ukoliko je u pitanju kraći tekst onda ga možete navesti direktno u ovom dokumentu, ukoliko je u pitanju duži tekst, onda navedete samo na kojoj strani i gde tačno se taj novi tekst nalazi. Ukoliko je izmenjeno ime nekog poglavlja, navedite na koji način je izmenjeno, i slično, u zavisnosti od izmena koje ste napravili.

2. Ukoliko ništa niste izmenili povodom neke zamerke, detaljno obrazložite zašto zahtev recenzenta nije uvažen.

3. Ukoliko ste napravili i neke izmene koje recenzenti nisu tražili, njih navedite u poslednjem poglavlju tj u poglavlju Dodatne izmene.

Za svakog recenzenta dodajte ocenu od 1 do 5 koja označava koliko vam je recenzija bila korisna, odnosno koliko vam je pomogla da unapredite rad. Ocena 1 označava da vam recenzija nije bila korisna, ocena 5 označava da vam je recenzija bila veoma korisna.

NAPOMENA: Recenzije ce biti ocenjene nezavisno od vaših ocena. Na osnovu recenzije ja znam da li je ona korisna ili ne, pa na taj način vama idu negativni poeni ukoliko kažete da je korisno nešto što nije korisno. Vašim kolegama šteti da kažete da im je recenzija korisna jer će misliti da su je dobro uradili, iako to zapravo nisu. Isto važi i na drugu stranu, tj nemojte reći da nije korisno ono što jeste korisno. Prema tome, trudite se da budete objektivni.

Glava 2

Recenzent — ocena:3

2.1 O čemu rad govori?

Rad govori o prevođenju koda pisanog u Haskel jeziku do mašinskog jezika. Tekst detaljno opisuje faze koje ovakvo prevođenje obuhvata i kroz koje kod prolazi. Govori o bitnosti optimizacije koda kroz faze i pojednostavljanje istog na osnovu precizno definisanih pravila.

2.2 Krupne primedbe i sugestije

Suštinski, rad sadrži sve što se od teme zahteva. Međutim, postoje neke stvari koje bi mogle da se isprave kako bi rad bio zanimljiviji čitaocu. Trebalo bi objasniti neke pojmove, koji se pominju u tekstu, ukratko kako bi se čitalac podsetio ili upoznao sa pojmom.
Smatram da bi bilo korisno u podglavlju **1.1 Haskel** dodati primer kratkog koda napisanog u Haskel programskom jeziku i na taj način približiti Haskel običnom čitaocu. U podglavlju **2 Frontend** postoje četiri podnaslova koja bi mogla biti zanimljivija običnom čitaocu da im se drugačije pristupilo (npr. u podnaslovu **2.1 Parsiranje** kaže se *"Paterni su parsirani kao izraz i transformisani iz HsExpr.HsExp u HsPat.HsPat. Izraz kao sto je [x | x <- xs] koji ne izgleda kao patern je odbijen."* - Šta znači ova transformacija koja se spominje u prvoj rečenici i kakav je to izraz koji izgleda kao patern?). Sličan problem postoji i u podglavlju **4 Backend**, gde se govori *"Kod prevodenja jezika Core u neki imperativni međujezik kao sto je C- - postoji više etapa. Prvo se CoreSyn (GHC's intermediate language) prevodi u StgSyn (GHC's intermediate language), i to u dve faze..."*. Bilo bi dobro približiti svakom čitaocu značenje pojmova *CoreSyn* i *StgSyn*.
Još jedna sugestija je vezana za navođenje literature. Literatura nije dobro napisana, neki linkovi ne otvaraju prave strane ili sadrže samo naslov (pritom nije jasno da li je to naslov knjige, članka ili nečega drugog). Literatura koja je nedovoljno opisana je *[2] Why is haskell diffcult. http://www.scs.stanford.edu/11au-cs240h/notes/ghc.html/*.
Svesni da rad nije lak za čitanje i da se rad nastavlja na rad od prošle godine u uvodu rada smo dodali referencu na link ka radu od prethodne godine, tako da je poželjno da se svaki čitalac upozna sa tim radom pre čitanja načeg.

Što se tiče primedi navedenih za Fronend i Backend naveden je link ka zvaničnoj stranici GHC gde se čitalac takodje može upoznati sa detaljima ovih procesa, jer bi njihovo objašnjavanje prevazišlo obim ovog seminarskog rada.
Literatura - Link je popravljen. Takođe su sređeni i svi naslovi, kako knjiga, tako i članaka.

2.3 Sitne primedbe

- Rečenicu *Radi konkretnosti, fokusirali smo se na Glazgov Haskel Kompajler (eng. Glasgow Haskell Compiler(GHC)) [1], ali ističemo da sve sto navedemo u ovom radu je primenljivo i za bilo koji kompajler za funkcionalni jezik, a možda i na kompilaciju za druge jezike.* zameniti rečenicom *Radi konkretnosti, fokusirali smo se na Glazgov Haskel Kompajler (eng. Glasgow Haskell Compiler(GHC)) [1], ali ističemo da sve sto navedemo u ovom radu se odnosi i na bilo koji kompajler funkcionalnih jezika, a možda i na kompajliranje drugih jezika.*.

  Rečenica je zamenjena.

- Rečenicu *Tekuću verziju kompajlera, razvijalo je 23 istraživača (eng. developers) sa preko 500 priloga (eng. commits).* potrebno je zameniti rečenicom *Tekuću verziju kompajlera, razvijalo je 23 programera (eng. developers) sa preko 500 priloga (eng. commits).*.

  Rečenica je zamenjena.

- Rečenicu *Paterni su parsirani kao izrazi i transformisani iz HsExpr.HsExp u HsPat.HsPat.* zameniti sledećom *Paterni (eng. Pattern) su parsirani kao izrazi i transformisani iz HsExpr.HsExp u HsPat.HsPat.*.

  Rečenica je zamenjena.

- Rečenicu *Greške generisane od strane samog parsera imaju tendenciju samo da kažu da se greška desila na određenoj liniji i ne pružaju nikakve dodatne informacije.* zameniti rečenicom *Greške generisane od strane samog parsera imaju tendenciju samo da ukažu na to da se greška desila na određenoj liniji i ne pružaju nikakve dodatne informacije.*

  Rečenica je zamenjena.

- Prepraviti rečenicu *GHC proverava programe u njihovoj orginalnoj Haskel formi pre nego što ih desugar konvertuje u Core kod* - potrebno je reč *orginalnoj* zameniti sa *originalnoj*, kao i objasniti šta je *desugar* (ili je neka gramatička greška?).

  Greška u kucanju *orginalnoj* je ispravljenja, a što se tiče primedbe za *desugar* , ubačen je link ka objašenjenju samog procesa.

- Rečenicu *Reprezentacija tipova je fiksirana u modulu TypeRep i eksportovana kao podatak tipe Type.* zameniti rečenicom *Reprezentacija tipova je fiksirana u modulu TypeRep i eksportovana kao podatak tipa Type.*.

  Ispravljena greška u kucanju.

- U rečenici *Core jezik se sastoji od nekoliko elemenata: variables, literals, let, case, lambda abstraction, application.* potrebno je prevesti ove elemente i u zagradama navesti reč na engleskom jeziku.

  Namerno su navedeni engleski termini kako bi se skratio postupak upoznavanja sa osnovnim elementima jezika.

- Rečenicu *Želja da se zadrže dobre osobine svođenja i komapiliranja na C, a da se opet prevaziđu problemi, inspirisala je razvoj jezika srednje-niskog nivoa (eng. low-level intermediate languages).* zameniti rečenicom *Želja da se zadrže dobre osobine svođenja i kompajliranja na C, a da se opet prevaziđu problemi, inspirisala je razvoj jezika srednje-niskog nivoa (eng. low-level intermediate languages)..*

  Ispravljena greška u kucanju.

- Deo rečenice *U ovom slučaju, vreme izvreme izvršavanja pomoću LLVM-a je:...* zameniti *U ovom slučaju, vreme izvršavanja pomoću LLVM-a je:...*

  Ispravljeno.

2.4 Provera sadržajnosti i forme seminarskog rada

1. Da li rad dobro odgovara na zadatu temu?
   Rad odgovara na zadatu temu. Fokus je na fazama prevođenja izvornog koda napisanog u funkcionalnom jeziku Haskelu, pri čemu se autori zadržavaju na svakoj od pomenutih faza i govore nešto više o njima.

2. Da li je nešto važno propušteno?
   Smatram da ništa važno nije propušteno u ovom radu. Autori su naveli suštinu teme i zadržali se na bitnim delovima teksta.

3. Da li ima suštinskih grešaka i propusta?
   Suštinskih grešaka u ovom radu nema. Propust na koji bi eventualno trebalo da porade je taj što je tekst pisan za posebnu grupu čitalaca (programeri i matematičari), tako da bi mogli učiniti tekst zanimljivijim i drugim čitaocima.

   Tema rada je takva da je rad namenjen za posebnu grupu čitalaca kojima.

4. Da li je naslov rada dobro izabran?
   Naslov rada predstavlja suštinu onoga o čemu rad govori. S toga smatram da je naslov odgovarajuć i dobro izabran.

5. Da li sažetak sadrži prave podatke o radu?
   Sažetak sadrži prave podatke o radu. U njemu je ukratko opisano koja je svrha funkcionalnih jezika i gde se oni koriste, kao i o čemu će se pisati u daljem tekstu.

6. Da li je rad lak-težak za čitanje?
   Delovi rada nisu laki za čitanje ukoliko čitalac nije upoznat sa Haskel jezikom, jer u tom slučaju čitalac ne može da razume dobro date primere i delove teksta kao što je rečenica "Paterni su parsirani kao izrazi i transformisani iz HsExpr.HsExp u HsPat.HsPat.". Ukoliko je čitalac upoznat

sa Haskel jezikom i načinom rada u njemu, ovaj tekst će mu biti lak za čitanje.

Obrazloženo već u okviru krupnih primedbi.

7. Da li je za razumevanje teksta potrebno predznanje i u kolikoj meri?
   Predznanje jeste potrebno kako bi se tekst u potpunosti razumeo. Kako bi čitalac razumeo svaki deo teksta trebalo bi da je upoznat sa nekim od tipova u Haskelu, paketa koji se koriste u ovom funkcionalnom jeziku i osnovama ovakve vrste programiranja. Naravno, potrebno je i da ima predstavu o tome šta znači prevođenje izvornog koda u mašinski kod.

8. Da li je u radu navedena odgovarajuća literatura?
   Navedena literatura odgovara temi i tekstu autora. Literatura govori o Haskel jeziku, funkcionalnim jezicima uopšteno, specifikacijama C− jezika, pravilima transformacija itd.

9. Da li su u radu reference korektno navedene?
   U tekstu ima više referenci. Međutim, u nekim delovima nije baš jasno na koji deo teksta se odnose navedene reference (da li na jednu rečenicu ili na veći deo teksta).

   Ako se referenca nalazi na kraju pasusa, odnosi se na taj pasus, a ako se nalazi umetnuta u okviru pasusa odnosi se na taj deo rečenice.

10. Da li je struktura rada adekvatna?
    Struktura rada je adekvatna. Rad je podeljen na delove na osnovu faza prevođenja koda koje su objašnjene u tekstu.

11. Da li rad sadrži sve elemente propisane uslovom seminarskog rada (slike, tabele, broj strana...)?
    Rad sadrži sve elemente propisane uslovom seminarskog rada. Navedeni su brojevi strana, tabela, postoje dve slike, u tekstu su navedene reference na obe slike kao i na literaturu u slučaju da je tekst preuzet direktno.

12. Da li su slike i tabele funkcionalne i adekvatne?
    U radu postoje dve slike koje su u skladu sa delovima teksta koji sadrži reference na njih. Prva slika predstavlja dobar prikaz puta koji izvorni kod funkcionalnog jezika pređe prilikom prevođenja do mašinskog jezika.

2.5 Ocenite sebe

Student sam prve godine master studija na Matematičkom fakultetu. Na osnovnim studijama smo imali predmete na kojima smo učil poneštoi o funkcionalnim jezicima i Haskel jeziku konkretno. S toga, smatram da sam srednje upućena u temu jer nemam svakodnevnih dodira sa ovakvim načinom programiranja.

Glava 3

Recenzent — ocena:5

3.1 O čemu rad govori?

Rad detaljno opisuje proces prevođenja Haskell-a do izvršnog koda. Kompilacija transformacijama, ideja koja je najzaslužnija za brzinu izvršavanja programa napisanih na Haskell-u, je objašnjena pomoću velikog broja primera. Opisani su svi delovi kompilatora, sa tim da je kod backenda prikazano više pristupa od kojih korišćenje LLVM kompilatorske infrastrutkure daje najbolje rezultate.

3.2 Krupne primedbe i sugestije

1. Uvod ne bi trebalo da sadrži podsekcije. U njemu bi trebalo opisati problem koji je tema rada, šta će biti izloženo i na koji način, kao i zašto je uopšte bitna tema koju rad obrađuje. Opis haskela i GHC-a bi trebalo odvojiti u posebnu sekciju koja se može zvati "Uvod u haskel i GHC". U ovom slučaju uvod rada bi trebalo da bude opširniji.

   Sugestije šta se može dodati u uvodu:

   - Zašto je problem kompilacije kod haskela (i funkcionalnih jezika uopšte) veoma težak? Zato što podržava funkcije višeg reda, lenju evaluaciju parcijalnu aplikaciju, sintaksu koja krije alokaciju, mehanizam provere tipova.
   - Zašto je problem prevođenja funkcionalnih jezika bitan? Zato što su efikasnost i korektnost koje pametno prevođenje može doneti izuzetno bitni.
   - Zašto GHC koristi "kompilaciju transformacijom"? Odgovor je da automatske transformacije znatno poboljšavaju performanse generisanog koda.
   - Ukratko opisati oblast prevođenja funkcionalnih jezika, i koji su najveći problemi koji se tu mogu sresti.

   Sredili smo uvod u skladu sa primedbama, dodali smo sekciju Uvod u Haskel i GHC, dodata je referenca ka radu od prošle godine

2. Zaključak izgleda nepotpuno. Može se opisati koje su sfere koje će se u narednom periodu najviše razvijati kod kompilatora Haskella. Mogu se navesti najvažniji delovi rada, na primer koje su ključne ideje za efikasnost prevođenja Haskella - transformacija koda i LLVM backend framework.

   Zaključak je upotpunjen

3. Sadržaj rada u sebi ne sadrži podnaslove. Ne vidim razlog zašto sadržaj rada ne bi bio potpun, izgleda kao da bi stao na prvu stranu uz podnaslove.

   Prva verzija rada sa podelom na naslove i podnaslove nije omogućila da sadržaj stane na prvu stranu u celosti sa podnaslovima. Sada je napravljena nova podela glava na podsekcije, pa sadržaj staj eu celosti na prvu stranu sa podnaslovima.

4. Zamerke oko referenci:

   - Sve reference u kojima se nalaze linkovi ka veb stranama bi takođe trebalo da sadrže i datum pristupa veb strani.

     Dodat je datum, tj godina pristupa stranici.
   - Referenca [2]. Link sadrži kosu crtu viška na samom kraju, zbog koje ne pokazuje na željenu stranu.

     Izbrisana kosa crta.
   - Referenca [9]. Link pokazuje na veb stranu na kojoj se nalaze korisni materijali o Windows operativnim sistemima, da li je ovo zaista pravi link?

     U pitanju je bila greška, sada je ispravljeno.

5. Ako se koristi slika sa interneta, možda ne bi škodilo ako bi se u okviru opisa slike dodao i njen autor, u ovom slučaju dovoljno bi bilo dodati referencu [2]-"why is Haskell difficult."uz napomenu da je slika odatle preuzeta.

   Dodata referenca odakle je slika preuzeta.

6. Pošto se u radu koriste termini iz lambda računa, trebalo bi da postoji uvod u lambda račun ili referenca gde se može naći detaljan opis ove teme. Za predlog rešenja pročitajte narednu zamerku.

7. Pošto se ovaj rad nastavlja na rad iz prethodne godine, to bi se moglo naznačiti u uvodu rada, uz referencu ka prošlogodišnjem radu. Takođe bi se moglo navesti da su tamo opisane teorijske osnove lambda računa koje su neophodno predznanje za razumevanje ovog rada.

   Sređeno

8. U uvodu u Haskell bi trebalo dodati da je haskel čist funkcionalni jezik, i potencijalno objasniti šta to znači.

   Zamerku nismo ispravili, jer smatramo da to i nije toliko relevanto za temu rada.

9. Kroz ceo podparagraf 3.1 se prožima primer koji nije adekvatno objašnjen. Iz primera izgleda da su a,b,c konstante, ali tako nešto bi trebalo da eksplicitno bude napomenuto.

   Eksplicitno je navedeno da su to konstante.

10. U 4.1 sekciji piše: "Takođe, C ne podržava repnu rekurziju, pristup steku radi čišćenja memorije(eng. garbage collection) i još mnogo drugih stvari."

    C podržava repnu rekurziju, ali ne podržava eliminaciju repne rekurzije, da li ste na ovo mislili? Ne znam na šta ste mislili u drugom delu rečenice. Jezik C programeru daje potpunu kontrolu nad memorijom. C nema automatski način čišćenja memorije sa heap-a ali ta funkcionalnost se može dodati bilbiotekama kao što je Boehm garbage collector za C i C++.

    Pogrešno je formulisana rechenica. Moguće je implementirati repnu rekurziju u C-u, no C nema efikasan način da je sprovede zbog same konstrukcije ovog jezika upravo zbog načina na koji koristi stek. Rečenica je preformulisana. Naravno, slažem se da u C-u imamo potpunu kontrolu nad memorijom, ali ovaj princip u kome bi GHC svodio program na C je vrlo star, a u to vreme verovatno nije postojala adekvatna podrška koja bi dala efikasan način. Na linku: https://llvm.org/pubs/2010-09-HASKELLSYM-LLVM-GHC.pdf mozete pročitati više o toj diskusiji. Navedena je i referenca u radu.

3.3 Sitne primedbe

1. ima veliki broj anglicizama u radu za koje bi idealno naći prevod, iako razumem da je ovaj problem izuzetno nezahvalan u ovakvim oblastima. Ako se odlučite da ostavite anglicizam zbog nedostatka adekvatnog prevoda, trebalo bi da se konzistentno pišu sve pojave reči - ili svaka pojava da bude pod znacima navoda, ili nijedna. Primeri:

    - typechecker - mehanizam/alat za proveru tipova
    - paterni - šabloni
    - renamer - mehanizam/alat za preimenovanje
    - desugaring - ovo ste preveli na dva načina prečišćavanje i odstranjivanje, odlučiti se za jedno od ta dva. Takođe ste koristili desugar, što bi trebalo prevesti na isti način.
    - inliner - mehanizam/alat za umetanje
    - code generator - generator koda
    - just-in-time kompilacija - ovo treba ili prevesti ili objasniti
    - life-long analiza - ovo treba ili prevesti ili objasniti
    - run-time sistem - sistem izvršavanja

    Anglicizmi su prevedeni i objašnjeni u skladu sa zahtevima.

2. Negde korsitite C–, negde Cmm, konzistentnost bi doprinela čitljivosti rada.

    Ispravljeno.

3. Na kraju uvoda pise sledeće:
"U ovom radu prikazaćemo u primeni transformacione tehnike kroz GHC kompajler za funkcionalni jezik Haskel".

Zar nije suvišno reći GHC(glasgow haskell compiler) kompajler za funkcionalni jezik Haskel, kada se već sve te informacije nalaze u skraćenici GHC?

Jeste suvišno, ispravljeno!

4. Na pocetku 1.2 sekcije:
"Tekuću verziju kompajlera, razvijalo je 23 istraživača (eng. developers) sa preko 500 priloga (eng. commits)".
Da li bi umesto reči "istraživača"bilo prikladnije reći "programera"?

   Promenjeno.

5. Posle tačke nedostaje zarez (sekcija 1.2):
".Sastavni deo svakog kompajlera..."
".Tekuću verziju..."

   Dodat je razmak, recezent je verovatno mislio da fali razmak, a ne zarez.

6. Nepotreban razmak između dve tačke i reči (sekcija 1.2):
"...kompajlera se sastoji iz tri faze :"

   Obrisan razmak.

7. Jedna otvorena, dve zatvorene zagrade (sekcija 1.2):
"(eng. Intermediate language))"

   Obrisan višak zagrada.

8. Neusaglašeno korišćenje padeža u narednom delu teksta (sekcija 2):
========== neispravno ==========
Sastoji se od:
1. Parsiranja (eng. Parser)
2. Promena imena (eng. Rename)
3. Provera tipa (eng. Typecheck)
4. Prečišćavanja (eng. Desugaring)

======= ispravno (izmene velikim slovima) ==========
sastoji se od:
1. parsiranja (eng. parser)
2. PROMENE IMENA (eng. rename)
3. PROVERE TIPA (eng. typecheck)
4. prečišćavanja (eng. desugaring)

   Sređeni padeži.

9. Na samom početku sekcije 2.1 se nalaze tri paragrafa koja su jedva duža od jednog reda. Lepše bi izgledalo ako bi se preformulisalo.

   Da ne bi otišli van obima teme, dodali smo referencu ka objašnjenju ovih delva.

10. Pravopisna greška (sekcija 2.3):
"koristeći strukture koje apstrahtuju"
"koristeći strukture koje APSTRAHUJU"

    Ispravljeno.

11. Da li je zarez potreban pre tri tačke? (sekcija 2.3):
"neiskorišćene biblioteke koje su uključene,..."

    Nije, greškom je tu ubačen.

12. u Sekciji 2.3, podvučeni putevi do fajlova/foldera su vizualno neprijatni. Da li su ove informacije uopšte vredne pomena? Ako jesu da li postoji lepši način da se skrene pažnja na njih, boldovanjem na primer?

    italic je lepši.

13. Provera iskaza pomenuta dva puta (sekcija 2.3):
"tcRnStmt, tcRnStmt"

    Obrisano duplo pojavljivanje.

14. Pravopisne greške (sekcija 2.3):
"kao podatak tipe Type."
kao podatak TIPA type.

    "pre faza povere tipa,"
pre faza PROVERE tipa,

    Ispravljeno.

15. Jezička greška (sekcija 3.1):
"Eliminacija mrtvog koda (eng. Dead code elimination) - odbacuje let vezivanje koje se više ne koriste."
eliminacija mrtvog koda (eng. dead code elimination) - odbacuje let VEZIVANJA KOJA se više ne koriste.

    Ispravjeno

16. U sekciji 3.1.1 dovoljno je jednom prevesti skraćenicu SGNF na engleski. Umesto naknadog ponavljanja prevoda, bolje je koristiti samo skraćenicu na srpskom ili samo skraćenicu na engleskom.

    Prihvaćeno i samo jednom je prevedena skraćenjica

17. Pravopisna greška (sekcija 3.1.2):
"Nažalost, većini linearnih sistema je neadekvatna,"
nažalost, VEĆINA linearnih sistema je neadekvatna,

    Ispravjeno

18. CoreSyn se pojavljuje u drugom poglavlju, gde je opisan kao međujezik GHC-a, nema potrebe da se ponovo to napominje u sekciji 4, takodje zašto je u zagradama opisano šta je, i zašto je na engleskom. Naredni deo teksta u sekciji 4 bi trebalo preformulisati:
    ==========
    Prvo se CoreSyn (GHC's intermediate language)
    prevodi u StgSyn (GHC's intermediate language)
    ==========

    Došlo je do greške u imenima. Ispravljeno je i dodate su reference na fajlove iz dokumentacije gde se može više pročitati o CorePrep i CoreToStg fazama

19. Pravopisna greška (sekcija 4.1):
"osobine svođenja i komapiliranja na C"
osobine svođenja i KOMPILIRANJA/KOMPILACIJE na C

    Ispravjeno

3.4 Provera sadržajnosti i forme seminarskog rada

1. Da li rad dobro odgovara na zadatu temu?
Smatram da rad adekvatno odgovara na zadatu temu sa par zamerki. Zadatak je bio opisati prevođenje do imperativnog jezika, ali zbog fokusa na Haskellu, razumljivo je da bude opisano prevođenje do izvršnog koda.

2. Da li je nešto važno propušteno?
Ništa od presudnog značaja za kvalitet rada, sa tim da su svi propusti detaljno objašnjeni u sekciji krupne primedbe i sugestije.

3. Da li ima suštinskih grešaka i propusta?
Isti odgovor kao i na prethodno pitanje.

4. Da li je naslov rada dobro izabran?
Jeste.

5. Da li sažetak sadrži prave podatke o radu?
Da, sa tim da bi mogao biti dodat razlog zašto je kompilacija funkcionalnih jezika teška i zašto je bitna.

    Dodato je u uvod rada, kao što je predloženo u okviru krupnih zamerki

6. Da li je rad lak-težak za čitanje?
Na skali od 0-10, gde je nula najlakše, radu bih dao ocenu 4.

7. Da li je za razumevanje teksta potrebno predznanje i u kolikoj meri?
Većina primera podrazumeva bar osnovno poznavanje funkcionalnih jezika, kao i lambda računa.

8. Da li je u radu navedena odgovarajuća literatura?
Jedino što nedostaje je rad iz prethodne godine na koji se ovaj rad nastavlja.

    Sredjeno

9. Da li su u radu reference korektno navedene?
Jesu, sa zamerkom da bi uz url veb stranica bilo poželjno da stoji i datum pristupa sajtu.

10. Da li je struktura rada adekvatna?
Jeste, sa malom zamerkom da bi uvod trebalo da bude odvojen od detaljnijeg opisa Haskell-a i GHC-a.

    Odvojeno je

11. Da li rad sadrži sve elemente propisane uslovom seminarskog rada (slike, tabele, broj strana...)?
Da, svi uslovi su ispunjeni.

12. Da li su slike i tabele funkcionalne i adekvatne?
    Jesu, sa malom zamerkom da bi možda tabela bila opisnija, ako bi se elementi sortirali od najsporijeg ka najbržem.

    Sortirano je od najbržeg ka najsporijem, tako je nama lepše

3.5 Ocenite sebe

Srednje sam upućen u datu oblast. Način na koji kompilatori funkcionišu sam imao priliku da naučim na kursu prevođenje programskih jezika u trećoj godini osnovnih studija. Tako da sam bio upoznat sa nekim tehnikama koje se mogu primeniti, ali ni približno ovoliko detaljno i specifično za funkcionalne jezike, sa kojima sam imao priliku da se susretnem u okviru kursa programske paradigme koji je takođe na osnovnim studijama. Na kursu je objašnjena logika prvog reda, i takođe je rađen Haskell.

Glava 4

Recenzent — ocena:4

4.1 O čemu rad govori?

Ovaj rad govori o fazama prevođenja Haskel funkcionalnog jezika. Prevođenje se sastoji iz tri velike faze. Prva faza treba da pretvori izvorni kod u Core jezik sa kojim dalje rade naredne faze. U ovoj fazi se vrši parsiranje, promena imena, provera tipova i prečišćavanje. Druga faza vrši optimizaciju. Umetanje koda je jedna od najefikasnijih tehnika optimizacije. Problem kod ove tehnike je što u nekim slučajevima ovo može značajno da poveća kod, kao i da dovede do toga da se neki delovi izračunavaju više puta. Zato se umetanje koda ne može uvek jednostavno primeniti, već je potrebno izvršiti analiz koda i utvrditi da li će željeno umetanje doneti više štete ili koristi. Treća faza govori o prevođenju Core jezika u neki imperativni međujezik. Glavni problem u ovaj fazi je upotreba lenjih funkcija sa nestandardnom kontrolom toka, repnih rekurzija... Postoje tri pristupa ovom problemu to su NGC, C– i LLVM. NGC pristup generiše asemblerski kod. Drugo rešenje je korišćenjem C– programskog jezika koji je podskup jezika C ali specijalno prilagođen zadacima kompilacije. Treći način je korišćenjem LLVM. LLVM predstavlja skup modularnih ponovnoupotrebljivih kompajlera i različitih alata koji pomažu atomatizaciji procesa.

4.2 Krupne primedbe i sugestije

U uvodu nije opisano zašto je prevođenje funkcionalnih programskih jezika teže od prevođena ostalih jeika i koji su glavni problemi sa kojima se susrećemo i koje pokusavamo da rešimo.

Već kritikovano, dodato je

U poglavlju 3, radi povećanja čitljivosti i razumljivosti rada, treba na početku svake faze ukratko istaći šta je suština te faze, koji je njen glavni cilj ili koji to problem u ovoj fazi pokušavamo da rešimo. Takodje treba napraviti jasniju podelu šta ova faza obuhvata i koje su njen podfaze.

Jasnija podela je napravljenja u okviru *Middle end-a* sada imamo dve podsekcije *Umetanje* i *Transformacija uslova*, a samim tim je dodato i nešto ukratko o samoj fazi pre samih primera.

4.3 Sitne primedbe

- U poglavlju 1.2 pocev od druge rečenice nema razmaka posle tačke.

  Dodati razmaci.

- U tački 3 nije objašnjeno sta je LLVM.

  Dodat pun naziv LLVM-a, i pošto se tu prvi put pojavljuje kao pojam obrisano je značenje u narednim pojavljivanjima

- U poglavlju 2 bolje bi bilo da je podela na Parsiranja, Promene imena, Provere tipova, Prečišćavanja.

  Sređeno.

- U poglavlju 3.1.1 lepo sročiti poslednju rečenicu drugog pasusa. Umesto *Kao i većina kompajlera, koristi se heuristika za odlučivanje kada se radi umetanje funkcija* može da stoji *Kod većine kompaljera koristi se heuristika za odlučivanje kada se radi umetanje funkcija.*

  Prihavćen predlog izmene.

- Prevesti sve strane izraze poput *renamer*, *frontend*, *desugar*, *type checker* u poglavlju 2

  Prevedeno.

- Prevesti strane izraze poput *Middle end Backend Frontend* u podnaslovima.

  Ovi pojmovi su uobičajeni u računarskom svetu, stoga ih nismo prevodili.

4.4 Provera sadržajnosti i forme seminarskog rada

1. Da li rad dobro odgovara na zadatu temu?
   Rad je odgovorio na zadatu temu i opisao proces kompilacije funkcionalnih programskih jezika.

2. Da li je nešto važno propušteno?
   U radu nisu objašnjini osnvni problemi pri prevođenju funkcionalnih programskih jezika i razlike u odnosu na prevđenje imperativnih programskih jezika

   Dodato u uvod.

3. Da li ima suštinskih grešaka i propusta?
   Podela *Middle end* faze na podfaze nije lepo objašnjena u uvodu pa zbunjue čitaoce. U uvodu je rečeno da se primenjuju optimizacije poput umetanja koda, eliminacije zajedničkih podizrara i izbacivanje mrtvog koda. Zatim se objašnjava umetanja i tri transformacije povezane sa umetanjem i jednostavno umetanje za koje nije jasno naznačeno gde pripada. Zatim se prelazi na objašnjavanje transformacije uslova.

   Kao što je i ranije u uvodu pisalo u okviru middle end faze, odlučili smo da opišemo samo umetanje kao najznačajniju optimizaciju, a druge metode optimizacije su pobrojane, ali kako nisu toliko značajne nisu ni opisane.

Što se tiče dela za jednostavno umetanje jeste bio nejasan, jer je došlo do problema u prevođenu stoga je to ispravljeno. Što se tiče transformcije uslova nije napisano šta je tu tačno zbunilo recezenta, tako da to nije bilo nikakvih većih izmena.

4. Da li je naslov rada dobro izabran?
   Jeste. Naslov rada lepo oslikava njegov sadržaj.

5. Da li sažetak sadrži prave podatke o radu?
   Da, sažetak sadrži ispravne podatke o radu.

6. Da li je rad lak-težak za čitanje?
   Rad je dosta težak za čitanje, jer sadrži puno stručnih izraza koji su slabo objašnjeni pa zahteva veliko predznanje čitalaca.

7. Da li je za razumevanje teksta potrebno predznanje i u kolikoj meri?
   Za dobro razumevanje rada neophodno je veliko predznanje iz oblasti funkcionalnih jezika. Potrebno je poznavanje sintakse i osnovnih principa kako bi se razumeli primeri, ali i funkcije nekih modla kompajlera.

8. Da li je u radu navedena odgovarajuća literatura?
   Da, u radu je navedaena neophodna literatura.

9. Da li su u radu reference korektno navedene?
   Da, u sve reference su ispravno navedene

10. Da li je struktura rada adekvatna?
    Da, rad ima adekvatnu struktur.

11. Da li rad sadrži sve elemente propisane uslovom seminarskog rada (slike, tabele, broj strana...)?
    Da, rad ispunjava sve uslove seminarskog.

12. Da li su slike i tabele funkcionalne i adekvatne?
    Da.

4.5 Ocenite sebe

U oblast funkcionalnih programskih jezika sam malo upućena, jer me je ova oblast zaintersovala i malo sam čitala o njoj.

Glava 5

Dodatne izmene