Profajliranje Haskell programa

Jovana Bošković, Ana Jakovljević, Nikola Perić, Mateja Trtica

20. april 2020.

Sadržaj

Glava 1

Recenzent — ocena: 4

1.1 O čemu rad govori?

Rad opisuje profajliranje i njegov značaj za optimizaciju programa. Glavni deo se fokusira na mogućnosti vremenskog i prostornog profajliranja u funkcionalnom programskom jeziku - Haskell. Kroz GHC su predstavljene opcije profajliranja i primene nad konkretnim primerom koda. Objašnjen je način tumačenja statističkih izveštaja i njihovo vizuelno predstavljanje.

1.2 Krupne primedbe i sugestije

Na samom početku rada, u uvodu, nedostajala mi je referenca na neki pouzdan izvor informacija iz dva razloga:

1. da me ubedite da je informacija validna
2. želja da više pročitam o tome

Konkretno mesto o kome pričam je deo sa tekstom: *U procesu razvoja optimizatora profajliranje...*, odnosno kraj prvog pasusa poglavlja **Uvod**.
Ispravljeno je, navedena je referenca. Pri tome, promenjena je formulacija, sadrži više informacija: Pri dizajniranju kompajlera profajliranje ima bitnu ulogu pri proveri optimizacionih tehnika.

Takođe, neka referenca u delu **2 Profajliranje** za tekst: *...a kako bi analiza prikupljenih podataka tokom izvršavanja bila relevantna potrebno je da ciljni program bude izvršen dovoljan broj puta*, bi bila od koristi, jer ne znam šta podrazumevate pod relevantna analiza. Koliko znam uopšteno proces profajliranja za razne jezike može da bude deterministički i statistički, tako da relevantnost može da se dobije i samo jednim izvršavanjem ponekad.
Što se tiče procesa profajliranja, naišla sam da je podela na determinističke i statističke odlika Python profajliranja. Svakao, perspektiva posmatranja rečenice je slična kao kad je u pitanju testiranje: potrebno je izvršiti na različitim ulazima, da bi ono što dobijemo imalo statistički značaj. Neće svaki ulaz isto uticati na izvršavanje delova koda.

Što se tiče sekcije 3 ona me je malo zbunila u početku, ali nakon nekoliko puta čitanja bila mi je jasna ideja u potpunosti. Možda da bi se ta osnovna ideja stekla odmah nakon prvog čitanja, ne bi bilo loše da uvedete celine 3.1, 3.2, 3.3 tako što ćete objasniti iznad zašto ih uvodite i šta njima postižete. Ili da se u 4.1 poglavlju referišete na celinu 3.3, čime je opravdano njeno postojanje u tekstu.
Celina 3 je izmenjena. Promenjen je naslov, kao bi bolje sugerisao na tematiku unutar celine. Dopunjena je i reorganizovana, neki pojmovi su bliže objašnjeni.

Ono što bi još bilo korisno, u cilju potpune reprodukcije Vašeg rada je da ste dali način da se preuzme datoteka *tekst.txt* ili ako je njen sadržaj prevelik da ste to naznačili negde. Takođe, bilo je potrebno da uradim import **Data.Char** i **Data.List** modula, da bi se kod preveo, pa možda to bude od koristi drugim čitaocima kao dodatna informacija.
Kod je ispravljen i u celosti napisan u Dodatku. Nismo smatrali potrebnim da navodimo sadržaj datoteke tekst.txt jer može biti bilo koji tekstualni sadržaj.

1.3 Sitne primedbe

U ovom delu navodim neke sitnije štamparske greške u tekstu ili unutar tabela koje možete ispraviti: jezima, jezka, taćke, funckije, automatkse, nezvisno, prosledeiti, dajle. Takođe, unutar tabela ste negde na kraju teksta stavili tačku, a negde niste.
Ispravljene štamparske greške i izbrisane tačke unutar tabele.

Možda jedan predlog koji možete razmotriti je da sve komande iz Haskell-a stavite u listing (kao što ste kod u poglavlju Dodatak), da bi bilo preglednije.
Komande za prevođenje i pokretanje Haskell programa, kao i izrazi označavanja centara troškova su izmenjeni i postavljeni u listinge.

Informacije u tekstu za ukupno vreme tokom izvršavanja programa za fajl Main.prof se ne poklapa, verovatno ste dodali neku prvu verziju slike i pisali tekst, pa posle dodali drugu verziju slike i zaboravili to da ažurirate u tekstu.
Ispravljeni podaci pročitani sa slike.

1.4 Provera sadržajnosti i forme seminarskog rada

1. Da li rad dobro odgovara na zadatu temu?
   Da. Rad je predstavio opšti koncept profajliranja kao i detaljnije kroz primer kako se profajliranje može izvesti u programskom jeziku Haskell kroz alat koji nudi GHC.

2. Da li je nešto važno propušteno?
   Informacije vezane za datoteku tekst.txt koja se koristi u programu, kao i informacije da li je neophodno uključiti dodatne module za Char i List funkcije, da bi se program preveo.

3. Da li ima suštinskih grešaka i propusta?
   Ne primećujem suštinske greške i propuste.

4. Da li je naslov rada dobro izabran?
   Da. Naslov je u skladu sa temom.

5. Da li sažetak sadrži prave podatke o radu?
   Da. Ne ponavlja informacije iz uvoda, a objašnjava ukratko sve što je dalje u radu predstavljeno.

6. Da li je rad lak-težak za čitanje?
   Tok sekcije 3 je bio težak za razumevanje nakon prvog čitanja. Bilo je potrebno da pročitam ceo rad i da se vratim na sekciju 3, kako bi mi ideja sekcije bila potpuno jasna. Osim toga, čitanje rada je bilo lako za razumevanje.

7. Da li je za razumevanje teksta potrebno predznanje i u kolikoj meri?
   Potrebno je osnovno programersko predznanje o tome zašto je najbitnije da se u procesu optimizacije osvrćemo na vremensko i prostorno zauzeće programa. Zašto se ne osvrćemo na imenovanje promenljivih umesto toga, na primer. Osim toga, tekst uvodi sve pojmove u onoj meri koliko je potrebno da se oni razumeju i primene bez predznanja.

8. Da li je u radu navedena odgovarajuća literatura?
   Da. Navedena literatura je citirana kroz rad i sadrži dosta opširnije informacije o temi.

9. Da li su u radu reference korektno navedene?
   Jesu, ali bi mogle na još nekim bitnim mestima da se navedu.

10. Da li je struktura rada adekvatna?
    Da, jasno su odvojena poglavlja po onome što opisuju i nisu prevelika.

11. Da li rad sadrži sve elemente propisane uslovom seminarskog rada (slike, tabele, broj strana...)?
    Rad sadrži bar jednu tabelu, bar jednu sliku, bar jedan napravljen grafik, adekvatnu literaturu, uklapa se u ograničenje broja stranica i zadrži sve neophodne sekcije.

12. Da li su slike i tabele funkcionalne i adekvatne?
    Jesu, sa odgovarajućim opisima.

1.5 Ocenite sebe

Nisam ekspert u Haskell-u, znam samo njegove osnove sa kursa Programske paradigme, što mi ne dozvoljava da dublje i suštinski razumem kako tačno funkcionišu procesi transformisanja i optimizacije nakon apstraktno napisanog koda. Dakle, ocenjujem sebe kao **c) srednje upućeni**, jer je moj seminarski rad na temu profajliranja i znam neke osnove Haskell-a. Kombinacija razumevanja obe teme u nekoj meri me je navela na ovaj odgovor.

Glava 2

Recenzent — ocena: 3

2.1 O čemu rad govori?

U daljem tekstu ću ukratko prepričati rad, bez detaljnog pisanja komandi i njihovog izvršavanja.

U razvoju softvera profajliranje glavnu ulogu igra u optimizaciji softvera, s obzirom da je skoro nemoguće da programer piše čitljiv i efikasan kod u isto vreme. Profajliranje je metod kojim se efikasno prikupljaju podaci o izvršavanju našeg programa i efikasna analiza istih koja nam kao rezultat daje koje su goruće tačke za optimizaciju. Alat koji vrši ovaj proces se naziva profajler. Profajliranje može koristiti na različite načine, od projektovanja hardvera, preko dizajniranja kompajlera do optimizacije gorućih tačaka u kodu. Deli se na dve osnovne metode: profajliranje vremena i profajliranje prostora. Profajliranje vremena delimo na: uzimanje uzoraka, brojač frevnecija i prolazno vreme procedure. Profajliranje prostora delimo na: profili alokacije, profili curenja i hip profili.

Osnovni problem kod Haskella je što je jezik visokog nivoa, što omogućava programeru da se skoncentriše na problem, ali zadaje problem kada je upitanju optimizacija jer nije moguće na osnovu koda proceniti preformanse. Haskell koristi jezik Core kao međukod. Kod profajliranja Haskell program mora da se pazi da redosled izvršavanja bude isti i da lenja semantika nije narušena. Prvi problem koji se javlja u Haskell profajliranju je ponovno korišćenje funkcije, jer onda istu funkciju mogu koristiti različiti delovi koda na različit način i ne možemo biti sigurni u izvor troška. Kod lenjog izračunavanja moramo da pazimo jer neće uvek svi delovi koda biti aktivni, zato moramo pažljivi biti prilikom pridruživanja troškova. Takođe implementacija funkcionalnih programa uključuje transformaciju i optimizaciju koje nastaju kao posledice visokog nivoa apstrakcije. Kod koji se dobija se značajno razlikuje od početnog.

U okviru Haskell programa postoji i GHC profajliranje koje se sastoji iz tri koraka. Ono što rešava GHC profajliranje je problem povezivanja, to uspeva centrima troškova. Centar troškova je oznaka na koju povezujemo cenu izvršavanja i predstavlja lokaciju u programu za koju želimo da sakpuljamo podatke. Kod centara troškova nam je bitno unošenje troškova i pravilna dodela troškova. Unošenje je na početku automatsko, a potom kada se vide problematični delovi koda, se prelazi na ručno unošenje. Pravilna dodela troškova se vrši korišćenjem strukture stek, svaki put kad se naiđe na centar troškova on se stavlja na stek. U

GHC profajliranju postoji vremensko i alokacijsko profajliranje, ali i prostorno profajliranje. Detaljnije u radu piše koje komande možemo da koristimo da bi izvukli određene podatke.

2.2 Krupne primedbe i sugestije

Prva primedba je na uvođenje pojma GHC, gde niko ko nije upoznat sa Haskellom nema pojma o čemu se radi.
Dodato. Uveden je pojam GHC kroz rečenicu "GHC (The Glasgow Haskell Compiler) je kompajler i interaktivn ookruženje za programski jezik Haskell."

Druga je da se u četvrtoj glavi u jednoj rečenici se nalazi previše tehničkih izraza da mi je teško da razumem značenje i smisao rečenice. Treba uprostiti malo i približiti nekom ko nije tolko upoznat sa funkcionalnim jezikom.
Nije precizirano na koju rečenicu se odnosi. Svakako, potrudili smo se da bolje objasnimo sve što smo naveli i uveli smo neke izmene.

Od sugestija bih naveo da komande ne budu navedene samo u sredini teksta, već da se koristi neka stilizacija u Latexu.
Ispravljeno, komande su navedene u listinzima.

2.3 Sitne primedbe

Ovde jedino imam primedbu na stil pisanja komandi. Umesto pravljenja pasusa opisa za neku komandu, pre bih naveo pa po crtama objasnio šta koja znači. Nešto slično ko man strane. Ovo može da bude i više sitna sugestija nego primedba.
Smatramo da je lepše navođenje svih komandi zajedno, kao malo uputstvo koje se lako može pronaći u radu.

2.4 Provera sadržajnosti i forme seminarskog rada

1. Da li rad dobro odgovara na zadatu temu?
   Rad je odgovorio na temu, primedbe su više na neki stil pisanja nego na to da je rad loš.

2. Da li je nešto važno propušteno?
   Nije, pokrio je sve osnovne pojmove.

3. Da li ima suštinskih grešaka i propusta?
   Nema sem da bi trebalo dodati još par rečenica o GHC.

4. Da li je naslov rada dobro izabran?
   Jeste. Neke promene mogu da se uvedu u vidu igre reči, ali je to više umetnička sloboda.

5. Da li sažetak sadrži prave podatke o radu?
   Sadrži, ceo uvodni deo je dobro napisan.

6. Da li je rad lak-težak za čitanje?
   U većoj meri je lak za čitanje, u primedbama sam skrenuo pažnju na određene propuste.

7. Da li je za razumevanje teksta potrebno predznanje i u kolikoj meri?
   Potrebno je osnovno predznanje iz programiranja (kursevi progrmranje 1 i 2). Ako bi smo želeli da približimo nekome ko nema to predznanje bila bi nam potrebana omanja skripta, 11 strana je malo za to.

8. Da li je u radu navedena odgovarajuća literatura?
   Jeste.

9. Da li su u radu reference korektno navedene?
   Jesu.

10. Da li je struktura rada adekvatna?
    Po mom mišljenju, ima lep tok od početka do kraja.

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

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

Jesu, značile su mi u razumevanju rada.

2.5 Ocenite sebe

Srednje sam upućen. Imam osnovno znanje o profajliranju i osnovno znanje o Haskellu, ali nikad nisam radio profajliranje u Haskellu. Svakako da sam naučio nešto novo čitajući ovaj rad.

Glava 3

Recenzent — ocena: 5

3.1 O čemu rad govori?

Autori ovog rada su nam predstavili osnove profajliranja programa pisanih u Haskell-u, kao i alate za profajliranje koje nam nudi GHC. Na početku rada, ukratko je objašnjeno šta je profajliranje i koji su motivi upotrebe profajlera. Nakon toga, dati su koncepti profajliranja funkcionalnih jezika, sa posebnim osvrtom na profajliranje u Haskell-u i korišćenje GHC alata.

3.2 Krupne primedbe i sugestije

Najveći deo grešaka čine loši prevodi strane literature. Lošim prevodom neka objašnjenja gube smisao i ovakve greške se moraju ispraviti.

3.2.1 Profajliranje

Prilikom opisivanja metode profajliranja vremena - brojač frekvencija, deo u rečenici „Ističe unutrašnje petlje i otkriva neočekivani kod i dinamiku ponašanja algoritama koji se koriste.” gde se spominje neočekivani kod, preuzet je iz literature na engleskom jeziku, ali je loše preveden. Naime, ako pogledamo navedenu literaturu iz koje je rečenica preuzeta, možemo uočiti da autor te literature ne spominje neočekivani kod, već da se otkrivaju neizvršeni delovi koda.
Ispravljeno. “Ističe unutrašnje petlje, otkriva neizvršene delove koda i dinamiku ponašanja korišćenih algoritama.”

U opisu profila alokacije, poslednja rečenica ne objašnjava precizno problem ovog metoda profajliranja. U literaturi korišćenoj za ovaj deo rada, autor navodi da problem može nastati ako profajler neke izvore alokacije ne istakne kao značajne, te nam profil neće otkriti objekte, alocirane na ovom izvoru, koji potencijalno mogu crpiti veliku količinu memorije.
Bilo je rečeno drugim rečima, možda je bilo potrebno bolje se izraziti. Ispravljeno. “Naizgled nešto ne zauzima značajan deo prostora i neće biti istaknuto u profilu alokacije, ali je dugotrajno zastupljeno i na taj način crpi veliku količinu memorije.”

3.2.2 Profajliranje u Haskell-u

Ne bi bilo loše detaljnije pojasniti o kakvoj vrsti optimizacija se radi u delu rečenice iz uvodnog dela ovog pasusa, koji glasi: „program koji profajliramo treba da bude optimiziovan". Preciznije objašnjenje je dato u literaturi koja se koristila za ovaj deo teksta.
Dopunjeno. "Dalje, program koji se profajlira treba da bude *optimizovan* od strane kompajlera jer je to upravo onaj program koji se pokreće i koristi."

3.2.3 GHC profajliranje

U delu teksta „Pravila za dodelu troškova", rečenica „Svaki centar troškova je povezan troškovima izračunavanja koje zahtevaju sve instance izraza određeni određuje taj centar troškova."loše je sastavljena i samim time nije najjasnije o čemu se ovde govori. Kako je ovo uvodna rečenica za kasnije kompleksnije teme, trebalo bi je preformulisati.
Ispravljeno i zamenjeno sa: "Troškovi izračunavanja koje zahtevaju sve instance nekog izraza se povezuju sa centrom koji određuje taj izraz."

U odeljku „Vremensko i alokacijsko profajliranje", navedeno je da kolona „entries"predstavlja koliko je puta čvor u stablu unesen. Pravilinije objašnjenje bi bilo da ova kolona predstavlja broj poseta konkretnom čvoru. Ispravljeno.
Takodje, rečenica gde se objašnjava uloga kolona „individual"i „inherited"bi se mogla preformulisati na sledeći način: „Kolone „individual"i „inherited"pružaju informacije o količini vremenskih i memorijskih resursa koje je centar potrošio nezavisno od ostalih, kao i o količini resursa koje su naslednici tog centra utrošili."
Primedba nije usvojena jer je pogrešna. Kolona *inherited* ne predstavlja vremenske i memorijske resurse koje su naslednici utrošili sami za sebe već resurse koje je utrošio centar zajedno sa svojim naslednicima, što je u radu i napisano. Ostatak predložene rečenice o koloni *individual* je isti kao u radu.
Prilikom opisa generisanja hip profila programa, u trećoj tački treba navesti da je „hp2ps"alat (program), što iz ovakvog zapisa čitaocu verovatno neće biti odmah očigledno.
Ispravljeno. "Pokrenuti alat hp2ps nad prog.hp za Postscript ispis fajla prog.ps."

3.2.4 Zaključak

Ne bi bilo loše da se u zaključku malo detaljnije sumiraju pojmovi koje je ovaj rad predstavio. Trebalo bi spomenuti još neke bitne stavke koje su obuhvaćene ovom temom u razradi.
Zaključak je izmenjen. Promenjen je način sumiranja rada, date su drugačije informacije: "Ovim radom je obuhvaćena osnova i motiv upotrebe profajlera. Uveden je pojam centra troškova, koji predstavlja osnovu za profajliranje. Iznet je specifičan slučaj profajliranja pomoću GHC alata dat kroz primer, kao i dodatne zastavice koje mogu biti korišćene."

3.3 Sitne primedbe

Uočeno je nekoliko stilskih i slovnih grešaka koje se lako mogu ispraviti.

3.3.1 Sažetak

Dva puta je napravljena slovna greška: u trećoj rečenici napisano je „jezima” umesto jezicima, a u četvrtoj „jezka” umesto jezika. S obzirom da je sažetak deo rada na koji se prvo nailazi pri čitanju rada, smatram da je ove greške neophodno ispraviti. Ispravljeno.

3.3.2 Uvod

U pretposlednjoj rečenici, deo „specifičnost i problemi profajliranja na lenjim funkcionalnim jezicima” bih preformulisala kao „specifičnost i problemi profajliranja lenjih funkcionalnih jezika”.

Obe verzije čudne zvuče jer: ne profajliramo na jezicima, i ne profajliramo jezike. Zamenjeno sa: “Zatim su navedeni specifičnost i problemi profajliranja programa pisanih na lenjim funkcionalnim jezicima visokog nivoa kakav je Haskell.”

Takođe, poslednja rečenica je započeta sa veznikom „I”, što bih izbacila i započela rečenicu bez tog veznika na sledeći način: „Na kraju je naveden način...”. Ispravljeno i zamenjeno drugačijom konstrukcijom koja ne sadrži „I” na početku.

3.3.3 Profajliranje

Napravljena je slovna greška, umesto „kritične tačke izvršavanja", napisano je „kritične tačke izvršvanja".
Pri opisu profila alokacije, napravljena je slovna greška u trećoj rečenici, gde je napisano „taćke"umesto tačke. U istoj rečenici, ispred veznika ali fali zarez. Ispravljeno sve navedeno.

3.3.4 Profajliranje u Haskell-u

U naslovu ovog odeljka, smatram da reč „Haskellu"treba zapisati kao „Haskell-u". Naslov je zamenjen, ali je spravljeno u ostatku teksta.
Pri navođenju imena jezika „Core", otvarajući znaci navoda bi trebalo da budu dole, a zatvarajući da ostanu kako jesu, gore.
Obrisani su znaci navoda oko reči Core, ali su napisani drugačiji znaci navoda na drugim mestima. Stavljeni su “ za otvorene navodnike i ” za zatvorene navodnike.
Početak rečenice „Ogleda se u mogućnosti"bi trebalo drugačije zapisati kako bi čitaocu bilo jasnije na šta se rečenica odnosi, na primer „Ovo se ogleda u mogućnosti". Ispravljeno.

Deo teksta gde je objašnjen pojam lenjog izračunavanja ima jednu stilsku i jednu slovnu grešku, obe se nalaze u drugoj po redu rečenici. Prva je „ne moraju nikada biti izvršeni"gde bi se dupla negacija mogla ukloniti i zapisati na sledeći način: „ne moraju ikada biti izvršeni". Druga je „nema potrebe da voditi brigu", rečca „da"je ovde suvišna.
Što se tiče stilske greške, u pitanju je dvostruka negacija koja je u našem jeziku opšteprisutna i ne pobija se u svim slučajevima. Slovne greške su ispravljene.

Na početku dela „Transformacije i optimizacije", reč „funckionalnih"treba prepraviti u „funkcionalnih". Ispravljeno.

3.3.5 GHC profajliranje

U okviru dela teksta „Centar troškova", u rečenici „Princip funkcionisanja: Vrši se povezivanje...", reč vrši bi trebalo zapisati malim slovom. Ispravljeno.
U odeljku „Pravila za dodelu troškova", na kraju četvrtog pasusa fali tačka za kraj rečenice. Ispravljeno.
U delu „Prikupljanje informacija o izvršavanju programa", na kraju druge po redu rečenice, stavljen je zarez ispred ili, što bi trebalo ukloniti. Takođe, na kraju ovog odeljka, treba reč "prosledeiti"prepraviti u "prosleditи". Ispravljeno.
U odeljku „Vremensko i alokacijsko profajliranje", napravljena je slovna greška. Reč „dobile"treba prepraviti u „dobili"u delu rečenice „Da bi se dobile tačni brojevi tih vrednosti". Ispravljeno.
Tabela 1 prekida tekst usred rečenice. Bilo bi dobro da se tabela postavi na kraj pasusa u kome se ista spominje.
Tabela 1 je izmenjena i premeštena u sekciju 4.2 Prikupljanje informacija o izvršavanju programa, postavljena je tako da ne prekida tekst.
Spominje se kolona „entery". Ovo je greška, jer ako pogledamo sliku koju tekst opisuje, možemo uočiti da se kolona zove „entries".Ispravljeno.
Uopšteno, u nekim delovima teksta, imena kolona su navedena iskošenim fontom, dok su u nekim delovima samo stavljena pod navodnike. Trebalo bi odabrati jedan stil i držati ga se tokom celog rada.
Imena kolona su ispravljena i napisana iskošenim fontom.
U delu „Prostorno profajliranje", druga rečenica po redu je suvišna, jer isto je već rečeno u prvoj rečenici. Rečenica je izbačena.
Prilikom opisa generisanja hip profila programa, druga tačka bi se, radi lepšeg stila zapisa, mogla započeti na sledeći način: „Pokrenuti ga sa nekom od opcija za profajliranje hipa...". Ispravljeno.

3.4 Provera sadržajnosti i forme seminarskog rada

1. Da li rad dobro odgovara na zadatu temu?
   Rad detaljno pokriva osnovne i bitnije pojmove ove teme.

2. Da li je nešto važno propušteno?
   Svaka tema spomenuta u radu je dobro obrađena. Možda su se mogle spomenuti još neke zanimljivosti iz oblasti profajliranja u Haskell-u, ali tada bi se prekoračio dozvoljen broj stranica rada ili bi se morala skratiti neka objašnjenja drugih pojmova, što bi verovatno loše uticalo na kvalitet rada.

3. Da li ima suštinskih grešaka i propusta?
   Bilo je par značajnijih grešaka nastalih pri prevođenju teksta iz strane literature, gde su zbog lošeg prevoda neki pojmovi pogrešno objašnjeni. Ukoliko se ovo ispravi, drugih značajnih propusta nema.

4. Da li je naslov rada dobro izabran?
   Naslov rada dobro opisuje temu koju su autori obradili.

5. Da li sažetak sadrži prave podatke o radu?

Sažetak kratko i precizno objašnjava šta će se u daljem radu obraditi. Ispunjava sve uslove kvalitetnog sažetka.

6. Da li je rad lak-težak za čitanje?
   Autori su se potrudili da napišu rad koji je dovoljno lak za čitanje čak i ako čitalac nema mnogo iskustva u ovoj oblasti.

7. Da li je za razumevanje teksta potrebno predznanje i u kolikoj meri?
   Potrebno je osnovno predznanje o samom pojmu profajliranja, šta je to i čemu služi. Predznanje o profajliranju funkcionalnih jezika nije neophodno. Literatura koju su autori naveli kao korišćenu detaljno pokriva ovu oblast, tako da čitalac istraživanjem ove literature se može sam upoznati sa ovom oblašću. Sam tekst je, takođe, dovoljno jasno napisan i lak za čitanje tako da je pogodan za osobe koje se ranije nisu srele sa profajliranjem u funkcionalnim jezicima.

8. Da li je u radu navedena odgovarajuća literatura?
   Sva literatura koja je navedena je odgovarajuća i korišćena u radu.

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

10. Da li je struktura rada adekvatna?
    Tok rada je logički i stilski lepo organizovan.

11. Da li rad sadrži sve elemente propisane uslovom seminarskog rada (slike, tabele, broj strana...)?
    Rad sadrži sve elemente propisane uslovom i poštuje osnovna pravila pisanja seminarskog rada.

12. Da li su slike i tabele funkcionalne i adekvatne?
    Tabele, uz propratni tekst, jasno i precizno daju dodatna objašnjenja nekih pojmova. Slike su detaljno objašnjene delovima teksta gde se spominju. Već na prvi pogled, čitalac može da razume šta je opisano datom tabelom ili slikom.

3.5 Ocenite sebe

U oblast koju ovaj rad obrađuje sam srednje upućena. Upoznata sam sa osnovnim konceptima profajliranja, ali nisam imala prilike do sad da se bavim profajliranjem u ovom programskom jeziku. Zarad recenzije ovog rada, pregledala sam korišćenu literaturu koju su autori naveli i time smatram da sam uspela bolje da se upoznam i sa osnovama profajliranja u Haskell-u.

Glava 4

Dodatne izmene

1. U uvodu je u poslednjem pasusu bliže objašnjeno šta će biti obrađeno u radu.
2. Dodata je slika sa ilustracijom toka profajliranja programa u sekciji 2.
3. Navedene reference koje su izostavljene
4. Bliže su objašnjeni pojmovi: funkcije višeg reda, polimorfizam, centar troškova.
5. Postavljeni naslovi iznad tabela. Promenjen izgled tabele. Poboljašan prevod opcija u tabelama.
6. Promenjen podnaslov trećeg dela u "Specificnost programskog jezika Haskell."
7. Urađene sitne izmene u vidu reorganizovanja teksta