# Naslov seminarskog rada

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Isidora Đurđević, Ana Stanković, Milica Đurić
kontakt email prvog, drugog (trećeg) autora

5. april 2017.

**Sažetak**

U ovom tekstu je ukratko prikazana osnovna forma seminarskog rada. Obratite pažnju da je pored ove .pdf datoteke, u prilogu i odgovarajuća .tex datoteka, kao i .bib datoteka korišćena za generisanje literature. Na prvoj strani seminarskog rada su naslov, apstrakt i sadržaj, i to sve mora da stane na prvu stranu! Kako bi Vaš seminarski zadovoljio standarde i očekivanja, koristite uputstva i materijale sa predavanja na temu pisanja seminarskih radova. Ovo je samo šablon koji se odnosi na fizički izgled seminarskog rada (šablon koji *morate* da ispoštujete!) kao i par tehničkih pomoćnih uputstava. Molim Vas da kada budete predavali seminarski rad, imenujete datoteke tako da sadrže temu seminarskog rada, kao i imena i prezimena članova grupe (ili samo temu i prezimena, ukoliko je sa imenima predugačko). Predaja seminarskih radova biće isključivo preko web forme, a NE slanjem mejla.

## Sadržaj

# 1 Uvod

Ovde pišem uvod

**Primer 1.1** *Problem zaustavljanja (eng.* halting problem*) je neodlučiv [?].*

# 2 Operaciona semantika

# 3 Denotaciona semantika

Nastala 1960. godina od strane Christopher Strachey-a i njegove istraživačke grupe na Oxford-u [**?**], *denotaciona semantika* predstavlja jednu vrstu reakcije na operacionu semantiku za koju se smatra da sadrži puno informacija. Naziv je dobila po engleskoj reči označiti (eng. *denote*) jer pridružuje značenja sintaksnim definicijama jezika. Alternativno, može se nazivati i *matematička semantika* zbog njene okrenutosti matematičkim formalizmima pri definisanju ove formalne semantike. Jedan način definisanja denotacione semantike je dat u sledećoj definiciji.

**Definicija 3.1** *Pristup formalizaciji semantike konstruisanjem matematičkih objekata koji opisuju značenje jezika naziva se* ***denotaciona semantika*** *[?].*

# 4 Aksiomatska semantika

Za nastanak i razvoj *aksiomatske semantike* su zaslužni pre svega Floyd, Hoar i Dijkstra [**?**]. Aksiomatska semantika razvija metode za proveru korektnosti programa. Zasniva se na matematickoj logici. Za svaku kontrolnu strukturu i komandu se definišu logicki izrazi. Ovi izrazi se nazivaju tvrđenja (eng. *assertions*) i u njima se zadaju ogranicenja za promenljive koja se javljaju u tim kontrolnim strukturama i komandama.
Tvrđenja su data u obliku Horovih trojki: {P}c{Q}

**Definicija 4.1** ***Horova trojka*** $\{P\}C\{Q\}$ *opisuje kako izvršavanje dela koda menja stanje izracunavanja ako je ispunjen preduslov (eng.* precondition*)* $\{P\}$*, izvršavanje komande C vodi do postuslova (eng.* postcondition*)* $\{Q\}$ *[?] .*

Preduslov je logički izraz u kome se definišu ograničenja promenljivih pre izvršavanja komande, a postuslov definiše ograničenja promenljivih posle izvršavanja komande.
Horove trojke se drugačije nazivaju i *delimična ispravnost specifikacije* (eng. *partial correctness specification*) .
Ali one ne mogu da osiguraju da će se program završiti pa se zbog toga i nazivaju "delimičnim".
Pored delimične ispravnosti specifikacije, imamo i *potpunu ispravnost naredbe* (eng. *total correctness statements*) koja osigurava da će se program završiti dok god preduslov važi.
Preduslovi i postuslovi mogu se smatrati interfejsom ili ugovorom između programa i njegovih klijenata. Oni pomažu korisnicima da razumeju šta program treba da proizvede bez potrebe da shvati kako se program

izvršava. Tipično, programeri ih pišu kao komentari za funkcije i funkcionišu kao dokumentacija i olakšavaju održavanje programa. Takve specifikacije su posebno korisne za bibliotečke funkcije za koje izvorni kod često nije dostupan korisnicima [**?**].

Način funkcionisanja ove semantike možemo prikazati u primeru koji sledi (primer je preuzet iz knjige [**?**] ).

**Primer 4.1** { *x=n* }
*y := 1; while neq(x=1) do (y := x*y; x := x-1)*
{ *y=n! and n veće od 0* }

*n je u primeru specijalna promenljiva koja se naziva logička promenljiva i koja, za razliku od programskih promenljivih, ne sme se pojaviti ni u jednoj naredbi koja se izvršava u programu i njena vrednost će uvek biti ista. Njena uloga je da pamte inicijalne vrednosti programskih promenljivih .*
*Zapisali smo preduslov da promenljive n ima jednaku vrednost kao i x u početnom stanju (tj. nego što program sa faktorijalom krene da se izvršava). S obzirom da program neće promeniti vrednost promenljive n, postuslov y=n! će izraziti da konačna vrednost y će biti jednaka faktorijalu početne vrednosti promenljive x, kada se izvršavanje programa završi.*

# 5 Zaključak

Ovde pišem zaključak.