Drugi kolokvij

Interpretacija programa

28. lipnja 2019.

Ovo je "*open book*" kolokvij. Dozvoljeno je korištenje bilo kakvih materijala — bilješke s vježbi/predavanja, vlastiti USBovi s riješenim zadacima, Python help, tutoriali, postovi na online forumima,... — **nastalih prije** kolokvija (npr. dozvoljeno je na *StackOverflow*u naći rješenje nekog dijela zadatka, ali nije dozvoljeno tamo postavljati pitanja za vrijeme kolokvija). Također, nije dozvoljena komunikacija (razgovor, *chat*, razmjena papira, USBova, ili bilo kakvih materijala) **između** studenata.

U prilogu je zadnja verzija biblioteke `pj.py`. Ako koristite neku drugu, uploadajte je skupa s rješenjem. Rješenje zadatka pišite u datoteku `pj_lambda.py`, te na kraju tu datoteku uploadajte na Moodle. Datoteka `pj_lambda.py` se treba moći izvršiti u Pythonu bez grešaka. Ako imate neki kod koji po Vašem mišljenju pokazuje ideju rješenja, ali iz nekog razloga ne radi, napišite ga u komentar. Korisno je u datoteku uključiti i testove s kojima ste testirali funkcionalnost.

Maksimalno vrijeme rješavanja je 180 minuta. U kolokviju je moguće osvojiti 30 bodova.

Veky

λ-račun je formalni jezik koji služi kao sustav opće izračunljivosti. Nas će ovdje zanimati samo njegova sintaksna struktura. Svaki λ-izraz je jednog od tri oblika:

- *slovo*, pri čemu dopuštamo sva latinična, grčka… i ostala slova, osim malog grčkog slova `λ` (koje ima posebnu ulogu)
- *aplikacija*, koja se piše kao konkatenacija dva λ-izraza MN.
- *apstrakcija*, koja je oblika `λ`v`.`M, gdje je v bilo koje slovo, a M bilo koji λ-izraz. Višestruku apstrakciju istog izraza možemo pisati `λxyz.M`, kao pokratu za `λx.λy.λz.M`.

Moguće je koristiti (oble) zagrade oko bilo kojeg λ-podizraza. Ako je MN aplikacija, a N apstrakcija, zagrade oko N su obavezne (y`λx.M` je sintaksna greška). Aplikacija je lijevo asocirana (`MNK` se shvaća kao `(MN)K`), i većeg prioriteta od apstrakcije (`λx.MN` se shvaća kao `λx.(MN)`).

Za svaku pojavu slova v u apstrakciji λv.M kažemo da je *vezana*. Za λ-izraz kažemo da je *kombinator* ako su sve pojave svih slova u njemu vezane. Recimo, `λxy.xyx`, `λx.(x(λx.x))` i `(λx.xx)(λx.xx)` su kombinatori, dok x, `(λxy.xy)x` i `λx.y` nisu.

[5b] Napišite leksički analizator (*tokenizer*) za λ-račun. [1b] Razmaci su dopušteni svuda osim između λ i vezanog slova. [3b] Potrebne tipove tokena odredite sami. [1b] Znak λ mora se moći unijeti i kao `λ` i kao `^` (radi lakšeg testiranja).

Napišite [5b] beskontekstnu gramatiku i [7b] sintaksni analizator (*parser*) za λ-račun (za [-2b] ne morate podržavati pokrate poput `λxyz.M`; za još [-1b] smijete prihvaćati izraze poput y`λx.M`). [1b] Apstraktna sintaksna stabla i njihove atribute odaberite sami.

[5b] Na apstraktnim sintaksnim stablima napišite metodu koja određuje sve "slobodne varijable": slova koja imaju pojave koje nisu vezane. [2b] Napišite funkciju `kombinator` koja prima λ-izraz, tokenizira ga, parsira, i vraća `bool`: je li zadani λ-izraz kombinator ili nije.