Miko — jezik za mikološke simulacije

Petar Pavlović i Mauro Raguzin

17. lipnja 2023.

Sadržaj

1 Opis jezika

Implementiran je jezik **Miko** koji omogućuje vođenje „pametnije" baze mikoloških uzoraka od recimo SQL baze. Pritom su implementirane specifične mogućnosti simulacije mutacije, selekcije i križanja korištenjem jednostavnih genetskih algoritama. Jezik je dizajniran da omogućuje laku manipulaciju numeričkim podacima te podacima koji predstavljaju uzorke gljiva s punim informacijama o pronalasku, tipu gljive te DNA. Implementacija se sastoji od interpretera ovog jezika napisanog u Pythonu.

Jezik ima sljedeće mogućnosti:

- Rad s listama svih ugrađenih osnovnih tipova: broj (svi su `double` interno), string, bool kao i objekata vezanih za domenu;
- Ugnježđene liste i aritmetiku nad njima;
- Aritmetika s jedinicama tj. dimenzijama; dimenzionalni *mismatch* je greška pri izvođenju ako se nije mogla dokazati statički, ali interpreter uključuje i statički analizator tipova koji pokuša što je više moguće dokazati pri samom parsiranju i tako unaprijed izbjeći nemoguće operacije poput množenja/dijeljenja dimenzionalnih veličina (koje su u ovoj domeni samo mase) ili zbrajanja dimenzijske i nedimenzijske veličine;
- Konkateniranje stringova sa +;
- Korisnički definirane funkcije;
- Ugrađene `read` i `write` funkcije koje (de)serijaliziraju iz/u JSON bilo koji objekt koji se koristi u programu;
- `print` funkcija s očitom namjenom, kao u Pythonu;
- Standardnu `for` petlju na jednoj varijabli;
- Grananje kao u C-u;
- Podrška za sve operatore (osim `compound assignment`) iz C-a.

Jezik je *dynamically typed*, no sve se varijable moraju deklarirati prije uporabe; redeklaracije nisu dozvoljene te su svi objekti koje stvaramo imutabilni, osim `Tree` objekata koji predstavljaju taksonomiju pojedine gljive. Ona se naime konstruira s default konstruktorom i onda se korištenjem dot-operatora (kao u C++-u) može pristupati pojedinim komponentama taksonomije i dodijeliti im stringove. Tijekom parsiranja se obavljaju mnoge statičke provjere tipa koje pokušaju otkriti što više moguće ilegalnih konstrukata prije izvođenja programa poput aritmetike između nekompatibilnih stvari (samo neki primjeri: lista i broj, dimenzioniran broj plus nedimenzioniran broj, string plus nestring...). No naravno problem statičkog određivanja tipa svakog izraza je u ovakvom jeziku općenito neodlučiv pa neće biti pokrivene situacije u kojima se pojavljuju recimo

samo varijable (ne izvodimo daljnju analizu toka kroz funkcije programa kako bismo potencijalno suzili skup mogućih tipova za svaku vidljivu varijablu).

Same gljive tj. uzorci su također objekti tipa `Fungus` koji se konstruiraju s `Fungus` konstruktorom; detalji su dani kroz komentare u izvornom kodu na repozitoriju ovog rada. Glavna funkcionalnost je rad s listama objekata gljiva koristeći tri genetska operatora: **selekcija**, **križanje** i **mutacija**. Križanje je binaran operator, ostali su unarni i rade poštujući neke globalne postavke koje korisnik može postavljati koristeći builtin `setParam` funkciju.

2 Neki primjeri programa

U datoteci `izoliranitest.py` unutar `package` direktorija repozitorija se (na samom dnu) nalazi niz testnih programa jezika koji pokazuju neke funkcionalnosti. Primjerice, za vidjeti kako se radi s listama, jedinicama i genetskim operatorima u ovom jeziku, provjerite zadnja dva primjera u spomenutoj datoteci. Zadnji primjer tamo prikazuje kako bi se u osnovi ovakav jezik mogao koristiti u nekom stvarnom istraživačkom projektu (tj. koristeći te komponente jezika), iako bi naravno još trebalo poraditi na točnom značenju i djelovanju genetskih operatora te još nekih specifičnih funkcionalnosti koje bi konkretnim znanstvenicima olakšale svakodnevni rad i učinile ih produktivnijima.