Razvoj programskog jezika Mercury

Seminarski rad u okviru kursa
Dizajn programskih jezika
Matematički fakultet

Andrijana Peković
pekovic.andrijana@gmail.com

20. decembar 2019

Sažetak

U ovom radu prikazuju se osnove razvoja programskog jezika Mercury. Ukratko su opisani jezici koji su najviše uticali na njegov razvoj: Prolog, Haskell i Smalltalk. Za svaki jezik date su osnovne informacije kao i način na koji je taj jezik uticao na osobine i razvoj jezika Mercury. Prikazano je razvojno stablo koje uključuje ove jezike.

Sadržaj

1 Uvod

Uprkos znanju da su logički programski jezici teorijski superiorniji od imperativnih programskih jezika, nisu bili korišćeniji prilikom izrade aplikativnih programa. Razlozi za ovo su u tome da, prilikom prevođenja programa napisanih u logičkom jeziku, prevodioci su detektovali daleko manje grešaka u odnosu na kompilatore za imperativne jezike. To je činilo debagovanje programa logičke paradigme težim, a produktivnost manjom. Takođe je implementacija u logičkim jezicima bila sporija od implementacija u imperativnim. Samim tim, čim su produktivnost i brzina bili bitni, imperativni jezici su pobeđivali u izboru. Iz tih razloga je rađeno na razvijanju novog programskog jezika, *Mercury*, koji je rešavao ove probleme i prvi put se pojavio 1995. godine. [8]. Logo programskog jezika Mercury može se videti na sledećoj slici 1 .

Slika 1: Logo programskog jezika Mercury

2 Osnovno o Mercury-ju

Mercury je čist[1] deklarativni programski jezik, sa primesama funkcionalnih i objektno orijentisanih jezika, namenjen kreaciji velikih, brzih i pouzdanih programa. Njegova kreacija je započeta 1993. godine od strane **Zoltan Somogyu**-a, **Fergus Henderson**-a i **Thomas Conway**-a na **Uverzitetu u Melburnu** [11], a prva verzija je objavljena 1995. godine. Nedostatak bočnih efekata omogućava da programi napisani u Mercury-ju nemaju širok spektar grešaka koje se pojavljuju u programima napisanim u imperativnim jezicima. Takođe, omogućava nam da transformišemo i optimizujemo Mercury programe na načine koji nisu mogući u programima napisanim u imperativnim jezicima ili u Prologu. Neke od implementiranih podrška u ovom programskom jeziku su za: deklarativno debagovanje, automatsku paralelizaciju, bektreking, fazu kompilacije, sakupljanje otpadaka u toku kompilacije (eng. *Compile Time Garbage Collection*)[2], otkrivanje grešaka u toku kompilacije, programiranje visokog

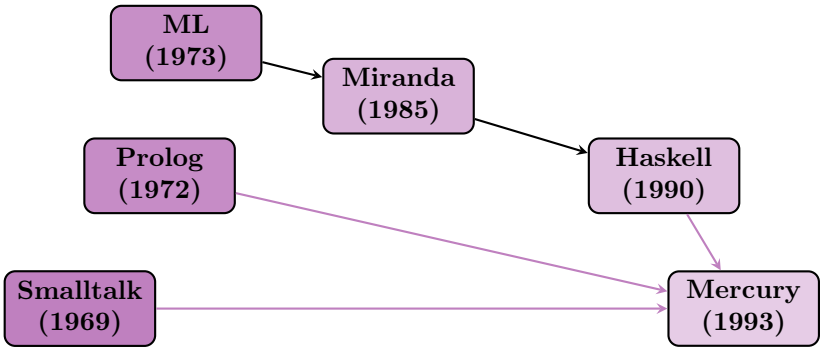
[1] Čista deklarativnost: predikati i funkcije u Mercury programskom jeziku nemaju nelogičke bočne efekte

[2] Nancy Mazur je u svojoj doktorskoj disertaciji[7] 2004. godine, izjavila da ona i njen tim veruju da je CTGC sistem Mercury programskog jezika prvi i jedini potpuni sistem sakupljanja otpadaka koji je ikada bio napravljen za neki programski jezik.

nivoa i parametarski polimorfizam. Uprkos sjajnim performansama i osobinama, više se koristi u akademske svrhe i za naučna istraživanja nego u komercijalnoj upotrebi. Firme koje koriste Mercury za razvoj svojih proizvoda su softverske kompanije *Mission Critical IT* za razvoj ODASE[6] platforme (eng. Ontology-Driven software development platform) i *YesLogic* za razvoj Prince XML[10] formatera. Sam Mercury-jev kompilator je napisan u Mercury-ju, a bio je pokrenut (eng. *bootstrapped*) korišćenjem NU-Prologa i SICStus Prologa. Taj kompilator prevodi Mercury program u C program, što mu omogućava da biva pokrenut na svim hardverskim i softverskim platformama. Poslednja stabilna verzija Mercury-ja je objavljena u septembru 2014. godine pod nazivom *Mercury 14.01.1*.

3 Razvojno stablo

Mercury jezik je zanimljiva kombinacija funkcionalnih, objektno orijentisanih i logičkih jezika. I ako je sintaksu nasledio iz **Prologa**, najveći uticaj na njega imao je **Haskell**, a iz **Smalltalk** jezika je povukao samo neke koncepte. Razvojno stablo jezika Mercury može se videti na sledećoj slici 2.



Slika 2: Razvojno stablo jezika Mercury

3.1 Prolog

Prolog je logički programski jezik nastao sedamdesetih godina kao rezultat projekta koji nije ni imao za cilj da stvori novi programski jezik, već radi na obradi prirodnih jezika. Njegovi idejni tvorci su **Alain Colmerauer** koji je prvo razvio 1969. **Q-Systems**, za obradu prirodnih jezika i **Philippe Roussel** na Univerzitetu u Marselju, a kasnije im se pridružio i **Robert Kowalski** koji je bio član grupe za razvoj veštačke inteligencije na Univerzitetu u Edinburgu. Kombinovanjem Q-Systems sa automatskim dokazivačem teorema, 1971. godine napravljena je preliminarna verzija programskog jezika, koji je iz Q-Systems preimenovan u **PROLOG** (franc. **PRO**grammation en **LOG**ique)[2]. Zvanična prva verzija PROLOGA je objavljena 1973. godine. Koristi veoma moćnu tehniku dokazivanja teorema poznatu kao **Metod rezolucije**, koji je stvoren 1963. godine od strane britanskog logičara **Alana Robinsona** [3]. Veliku pažnju je privukao kao prvi logički programski jezik i i dalje drži značajni uticaj i popularnost u okviru logičke paradigme, pa se koristi i kao sinonim za logičko programiranje - **Prolog**. Osnovne primene su mu dokazivanje

```
:- pred write_total(int::in, io::di, io::uo) is det.

write_total(Total, IO0, IO) :-
  print("The total is ", IO0, IO1),
  print(Total, IO1, IO2),
  print('.', IO2, IO3),
  nl(IO3, IO).
```

Listing 1: Mercury, verzija 1

```
write_total(Total) :-
  write("The total is "),
  write(Total),
  write('.'),
  nl.
```

Listing 2: Prolog

```
write_total(Total, !IO) :-
  print("The total is ", !IO),
  print(Total, !IO),
  print('.', !IO),
  nl(!IO).
```

Listing 3: Mercury, verzija 2

Slika 3: Kôd za ispisivanje teksta

teorema, obrada prirodnih jezika, istraživačka analiza (eng. *Exploatory analysis*), i izvozni sistemi (eng. *Export systems*). Takođe se koristi za rad u okviru razvoja veštačke inteligencije. Sintaksa Mercury jezika je bazirana na sintaksi Prologa, ali semantički su ta dva programska jezika suštinski različita usled čistoće Mercury programskog jezika, njegovih sistema tipova, režima, determinizma i modula[8]. Jako je bitno shvatiti da su ciljevi stvaranja ta dva programska jezika potpuno različiti. Mercury za razliku od Prologa, jeste čist deklarativni jezik, ima fazu kompilacije, CTGC sistem[7], jak determinizam i još mnogo prednosti. Mehanizam, radi očuvanja čistoće, za ulaz i izlaz koji koristi Mercury je "nanizavanje"(eng. *threading*) objekta koji predstavlja stanje sveta kroz rad programa. Tip ovog objekta je **io.state**, tj skraćeno **io**. Svaka operacija koja utiče na taj objekat mora imati dva argumenta ovog tipa, predstavljajući redom stanje sveta pre primene operacije i stanje sveta nakon primene operacije. Režimi ova dva argumenta poziva su **di** (eng. *destructive input*) i **uo** (eng. *unique output*). Prvo znači da ulazna promenljiva mora da bude poslednja referenca na originalno stanje sveta, a drugo znači da je izlazna promenljiva garantovano jedina referenca na stanje sveta koje je izlaz ovog predikata[9]. Primer[3] ispisa u Mercury-ju i Prologu se može videti na slici 3. Na slici 3.1 vidimo da se prvo definiše deterministički predikat sa tri argumenta, gde sa **is det** deklarišemo da je izlaz tog predikata jedinstven. Referenca na originalno stanje IO0, a referenca na izlazno IO, tj. moramo početi u IO0, a završiti u IO, a tokom nanizavanja vršimo preimenovanja bez ponavljanja. Sintaksna skraćenica koja je uvedena da olakša pisanje programa je **!**. Ona omogućava da kompajler imenuje različite verzije naslednih stanja tako što drži nasledna stanja objekta kroz klauze u novoj varijabli stanja (eng. *state variable*). Uvođenjem te oznake možemo još bolje videti na slici 3 skoro identičnu sintaksu ova dva jezika.

[3]Primeri su u potpunosti preuzeti iz uputstva kako preći iz Prologa u Mercury jezik [9].

3.2 Haskell

Haskell je čist funkcionalni programski jezik koji je statički tipiziran, podržava lenjo izračunavanje, ima podršku za parametarski polimorfizam, paralelno i distribuirano programiranje[5]. Nazvan je po **Haskell Brooks Curry**-ju, čiji rad u oblasti matematičke logike služi kao osnova za sve funkcionalne jezike. Zasniva se na lambda računu što mu omogućava da hardverski bude nezavisan jezik, a pritom to objašnjava zašto mu je lambda logo. Omogućava visoku efikasnost, kraći, čistiji i lako održivi kod, manje grešaka i visoku pouzdanost. Sistem tipova je Mercury nasledio od Haskella 98[4]. Neka od mesta na kojima se dosta poklapaju Haskell 98 i Mercury su: tipski sinonimi, tipovi funkcija visokog nivoa i lambda izrazi, tuple tipovi, parametarski polimorfizam, funkcionalne zavisnosti i sličan skup osnovnih tipova. Bez obzira na sve te sličnosti, postoji dosta odudaranja poput sintakse, predikatski račun Mercury-ja u odnosu na lambda račun Haskella, semantika izvršavanja operacija, tretiranje ulaza i izlaza, itd [8].

3.3 Smalltalk

Smalltalk je dinamički, objektno orijentisani programski jezik, nastao u Xerox PARC-u 1969. godine od strane **Alan Kay**-a, **Dan Ingalls**-a i **Adele Goldberg**.[1, 4] Kreiran je da bi istražio učenje dece programiranju. Jednostavan je i mali programski jezik, čak i najjednostavniji među najpopularnijim jezicima. Ima jako bogatu istoriju. Zbog njega je svet upoznat sa jezikom virtuelne mašine na kome se zasnivaju Java i Ruby. Jedan je od prvih koji je razvio JIT (eng. *Just In Time*) kompilaciju. Iz njega je nastao prvi moderni IDE (eng. *Integrated Development Environment*), koji je podrazumevao i tekst editor, pretraživač klasa, inspektor objekata i debager. Njegova poslednja poznata stabilna verzija je Smalltalk-80 koji se pojavio 1980. godine. Od pojave te verzije, Smalltalk je imao podršku za funkcionalno programiranje. Stiv Džobs (eng. **Steve Jobs**) je bio inspirisan Xerox PARC-ovim grafičkio-korisničkim interfejsom (eng. *Graphic User Interface*) i WIMP-om (eng. *Windows, Icons, Menus, Pointer*), tako da je na grafičko-korisnički interfejs Appleovih prozvoda direktan uticaj imao Smalltalk. Devedesetih godina je bio najpopularniji objektno orijentisani programski jezik odmah iza C++ programskog jezika. Danas je i dalje široko rasprostranjen i koriste ga firme poput JP Morgan, Desjardin, UBS, Florida Power & Light, Texas Instruments, Telecom Argentina, Orient Overseas Container Lines, BMW i Siemens AG. Smalltalk je uticao na razvoj debagera Mercury programskog jezika.

4 Zaključak

U ovom tekstu ukratko su predstavljene osnove razvoja programskog jezika Mercury. Prikazani su jezici koji su najviše uticali na njegov nastanak i razvoj, i prikazano je njegovo elementarno razvojno stablo. Uprkos manjku njegove popularnosti, čini se kao interesantan jezik za poznavati. Za više informacija možete posetiti njegovu zvaničnu stranicu[8].

[4]Haskell 98 je verovatno najbolje dokumentovana i najpoznatija verzija Haskella

Literatura

[1] David Robson Adele Goldberg. *Smalltalk-80: The language and its implementation*. Addison-Wesley Publishing Company, 1983.

[2] Philippe Roussel Alain Colmerauer. The birth of prolog, 1992.

[3] Encyclopedia Britanica. Evolutionary computing. https://www.britannica.com/technology/artificial-intelligence/Evolutionary-computing.

[4] Richard Eng. Introduction to the smalltalk programming language. https://www.codeproject.com/Articles/1241904/Introduction-to-the-Smalltalk-Programming-Language.

[5] Haskell. Zvanična stranica programskog jezika Haskell. https://www.haskell.org/.

[6] Mission Critical IT. Zvanična stranica ODASE projekta. https://www.odaseontologies.com/.

[7] Nancy Mazur. *Compile-Time Garbage Collectionfor the Declarative Language Mercury*. PhD thesis, Katolički Univerzitet u Luvenu, 2004.

[8] Mercury. Zvanična stranica programskog jezika Mercury. http://www.mercurylang.org/.

[9] Thomas Conway, Zoltan Somogyi, Fergus Henderson. *The Prolog to Mercury transition guide*.

[10] YesLogic. Zvanična stranica firme YesLogic. https://yeslogic.com/.

[11] Zoltan Somogyu, Fergus Henderson, Thomas Conway. Mercury: an efficient, purely declarative programming language, 1995.