Razvoj programskog jezika OCaml

Seminarski rad u okviru kursa
Dizajn programskih jezika
Matematički fakultet

Jovana Mirković
mr16218@alas.matf.bg.ac.rs

20. decembar 2019

Sažetak

U ovom radu ćete se upoznati sa programskim jezikom OCaml. Saznaćete pre svega nesto o jezicima koji su uticali na njegov razvoj, ali i o najznačajnijim jezicima na čiji razvoj je uticao OCaml. Ukratko su opisani jezici koji su najviše uticali na njegov razvoj: ML, SML, Caml Light, Caml Special Light, kao i jezici koji su se razvili koristeći ključne koncepte OCaml-a: F# i Reason. Za svaki od ovih jezika ukratko su navedene osnovne karakteristike i povezanost sa jezikom OCaml. Prikazano je razvojno stablo koje uključuje ove jezike.

Sadržaj

1 Uvod

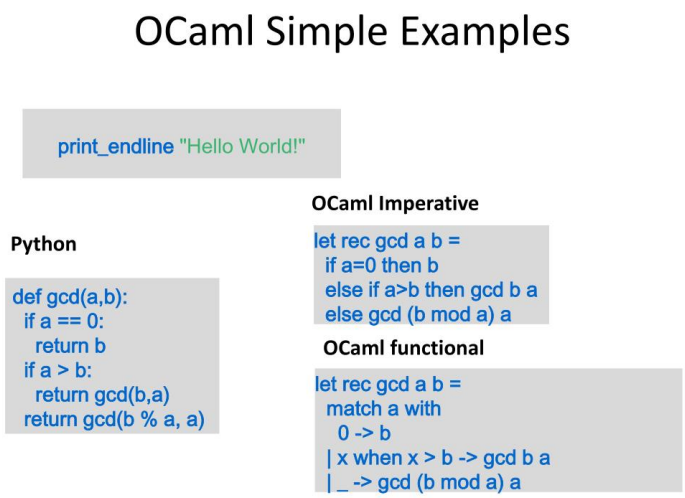
Programski jezici veoma utiču na pouzdanost, sigurnost i efikasnost koda. Programski jezici se veoma brzo razvijaju u skladu sa razvojem nauke i tehnologije. Većina današnjih programskih jezika je inspirisana konceptima starijih programskih jezika. Programski jezici su obično kombinacija više prethodnih značajnih jezika i dizajniraju se tako da nasleđuju njihove korisne karakteristike.
OCaml, jedan od članova ML familije koji je nastao 1996.godine je programski jezik opšte namene. U početku je korišćen za razvoj aplikacija koje koriste simboličko računanje, ali sada se koristi za razvoj softvera u mnogim drugim oblastima primene. Zbog njegove ekspresivnosti, sigurnosti, razvijenih biblioteka, ali i pored svega toga i jednostavnosti neke velike kompanije poput *Microsoft-a*, *IBM-a* i *CEA*(*Commissariat à l'Énergie Atomique*) razvijaju neke od svojih projekata baš u OCaml-u.

Slika 1: Logo programskog jezika OCaml

2 Osnovno o OCamlu

OCaml je industrijski programski jezik koji objedinjuje funkcionalne, imperativne i objektno orijentisane stilove programiranja. Razvila ga je pre vise od 20 godina grupa vodećih istraživača u Inriji(*Institut National de Recherche en Informatique et en Automatique*), francuskom nacionalnom istrazivačkom institutu za digitalne nauke, pod vođstvom Gerard Hueta(engl. *Gérard Huet*). [6]. Ono što karakteriše OCaml je da pruža kombinaciju efikasnosti, ekspresivnosti i praktičnosti kao nijedan drugi jezik do tada. OCaml nudi: efikasne kompajlere izvornog koda i odvojenu kompilaciju samostalnih aplikacija, moćan sistem tipova(koji uključuje statičku proveru tipova, parametrički polimorfizam i zaključavanje tipa), automatsko upravljanje memorijom zahvaljujući efikasnom sakupljaču smeća(end. *garbage collection*), efikasna sredstva za debagovanje. [4, 6] Sve ove pogodnosti čine OCaml odličnim izborom za kompanije poput *Facebook-a*, *Docker-a*,*Bloomberg-a*, *ANSSI-a* i mnogih drugih u kojima su izuzetno bitne efikasnost, što manji broj grešaka i njihovo lako uklanjanje. Koristi se i na univerzitetima jer je odlično sredstvo za podučavanje osnovnih koncepata koji se nalaze iza programskih jezika, teorije tipova i sistema.

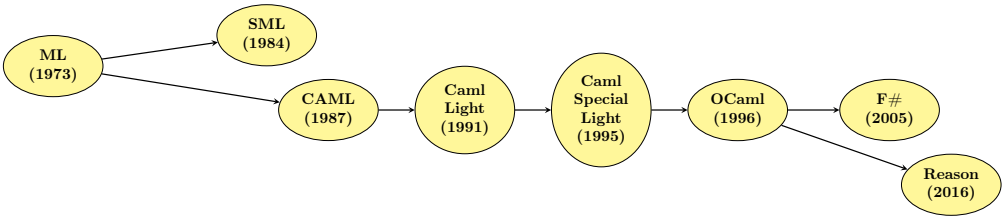
OCaml(*Objective Caml*) je glavna implementacija Caml programskog jezika.



Slika 2: Jednostavni primeri koda u jeziku OCaml upoređeni sa kodom u jeziku Python

3 Razvojno stablo

Na nastanak jezika OCaml je najviše uticao programski jezik **CAML** sa svojim implementacijama **Caml Light** i **Caml Special Light**.Na nastanak i razvoj jezika Caml najveći uticaj je imao jezik **ML**(eng. *Meta-Language*), čijoj porodici jezika i pripada, a zatim i njegov drugi dijalekt **SML**. Neki od mnogobrojnih jezika na koje je OCaml tokom godina uticao, a koji su prikazani na ovom razvojnom stablu su **F#** i **Reason**. Razvojno stablo jezika OCaml i njegovih prethodnika i naslednika može se videti na slici 3.



Slika 3: Razvojno stablo jezika OCaml

3.1 ML

ML(Meta jezik) je prvobitno razvijen na Edinburškom univerzitetu 1970-ih godina. ML je razvio Robin Milner prilikom rada na LCF(*Logic of Computable Functions*) [1] sistemu.[2] ML, jezik sa formalnom semantikom, je omogućio dokazivanje teorema, podržavajuci algebarske tipove podataka,parametarski polimorfizam, apstraktne tipove podataka i izuzetke. Formelov tim se zainteresovao za jezik ML-a 1980–81.Kako bi mogao da koristi LCF na raznim sistemima koji su se tada koristili u Formelu (Multics, Berkelei Unik na Vak, Simbolics), Gerard Huet je učinio ML implementaciju kompatibilnom sa raznim Lisp kompajlerima. Ipak, ML je programski jezik opšte namene, koji je tokom svog razvoja našao primenu u mnogim oblastima. Glavni dijalekti ML-a su SML i CAML.

OCaml deli osnovne karakteristike sa ostalim predstavnicima ML jezika. Pri razvoju ML-a cilj je bio da pravila zaključivanja i metode dokazivanja budu predstavljeni kao funkcije.Iz tog razloga je ML funkcionalni jezik, što znači da se funkcije tretiraju kao prvoklasne vrednosti. Funkcije mogu biti ugnježdene, mogu biti argumenti drugih funkcija i mogu se koristiti u strukturama podataka. ML je strogo tipiziran jezik, što znači da se tip svake promenljive i svaki izraz u programu određuje u vreme kompilacije. Programi koji prođu prevođenje su izuzetno sigurni i retko nastaju greške. ML koristi zaključivanje tipa za automatsko otkrivanje tipa podataka u izrazu. Pravila zaključivanja trebalo je da definišu apstraktni tip-vrstu teorema, pa se u ML jezicima koristi polimorfizam, što znači da je moguce pisati programe koji rade za vrednosti bilo koje vrste. Moguće je definisati generičke strukture podataka kao što su liste, stekovi i stabla koja mogu sadržati elemente bilo koje vrste.[3] U ML-u se koriste izuzeci da bi se na lakši način mogle uočiti greške u pojedinačnim delovima dokaza. Budući da u dokazu teorema ne sme biti nedostataka, ML je dizajniran tako da bude siguran, bez ikakvog oštećenja okoline.[7]

[1] Interaktivni automatizovani istraživač teorema zasnovan na deduktivnom sistemu za računske funkcije koji je predložio Dejn Skot(eng. *Dana Scott*) 1969.godine

3.2 SML

ML koji je razvijen na Edinburškom univerzitetu je bio poprilično spor jer su programi bili prevođeni na programski jezik Lisp[2] i tek onda interpretirani. To je jedan od razloga za nastanak SML-a(*Standard ML*) 1984.godine.Glavna proširenje u okviru SML-a u odnosu na ML je bilo dodavanje obrazaca formalnih parametara iz programskog jezika Hope[3], koji se mogu proširiti deklaracijom novih tipova. Principi na kojima se zasniva ovaj standard su: ograničiti osnove ovog jezika na jednostavne ideje, generalizacija, određivanje načina izračunavanja izraza i uvođenje striktne sintakse, što je omogućilo lak prelazak sa jedne implementacije na drugu.[5]

3.3 Caml

Nastanak Caml-a je inspirisan dugogodišnjim istraživanjima o ML-u počevši još od 1960-ih godina.*CAML* je skraćenica za *Categorical Abstract Machine Language.*Ime Caml ostalo je tokom čitave evolucije jezika, iako trenutna implementacija nema veze sa CAM-om. Caml je prvi osmislio i implementirao Inria-ov Formel-ov tim 1987.godine, na čelu sa Gaj Kazanoum(engl. *Guy Cousineau*) i Askanderom Suarez(engl. *Ascander Suarez*) i razvijana je do 1992. godine. Njegov razvoj su nastavili Pjer Vajs i Mišel Mauni(engl. *Pierre Weis and Michel Mauny*) unutar Cristal tima i njegovog trenutnog naslednika Galliuma. Mnogi osnovni koncepti SML-a su zadržani u ovoj implementaciji. Glavni razlog za razvoj Camla bio je korišćenje za razvoj softvera unutar Formela, pre svega za razvoj sistema Cok, koji je nakon teze Thierrija Cokuanda(engl. *Thierry Coquand*) 1985. godine postao glavni cilj tima. Koncepti na kojima je dizajniran su pre svega bezbednost i pouzdanost programa.I pored propusta poput loše optimizacije pristupa okolini i prenosivosti, Caml je poslužio kao odlična osnova za razvoj novih implementacija: Caml Lighta i OCamla.[1]

3.4 Caml Light

1990. i 1991. godine, Ksavier Leroi(engl. *Xavier Leroy*)je osmislio potpuno novu implementaciju Camla-Caml Light, zasnovanu na interpretatoru bajt kodova napisanom u C-u.Ova implementacija je unapređena u odnosu na prethodnu jer je bila visoko prenosiva i mogla se korisiti na malim desktop mašinama kao što su Macs i PC.Imala je bolji sistem upravljanja memorijom i obezbedila je promociju korišćenja Camla u obrazovanju i u istraživačkim timovima.

[2]Lisp je programski jezik zasnovan na matematičkoj teoriji rekurzivnih funkcija čije je prvo izdanje bilo još 1958.godine. Implementirao ga je Džon Mekarti(eng. *John McCarthy*).Osnova Lisp-a je funkcionalno programiranje, ali se Lisp zbog raznih drugih svojstava smatra multiparadigmatskim programskim jezikom.

[3]Hope je programski jezik koji je razvijen u isto vreme kao i ML na Edinburškom univerzitetu od strane Roda Burstala i John Darlington(engl. *Rod Burstall and John Darlington*) za rad na transformaciji programa. Hope je prvi programski jezik u kom su korišćeni algebarski tipovi podataka.

3.5 Caml Special Light

Caml Light je nastavio da se razvija i 1995.godine Ksavier je implementirao specijalnu, unapređenu verziju: Caml Special Light. Ona je bila poboljšana u odnosu na prethodnu verziju na više načina. Caml Light je uveo sistem modula visokog nivoa koji je omogućio apstraktnost.Pored bajtkod kompajlera, koji omogućava veću prenosivost, dodat je i optimizovani kompajler izvornog koda, koji podržava brojne arhitekture i koji je bio napredniji od postojećih kompajlera za funkcionalne jezike. Time je omogućeno Caml-u da parira mnogim do tada naprednijim jezicima. Caml Light je zastarela verzija, više se ne održava aktivno i na kraju ce biti uklonjen i u potpunosti zamenjena OCaml-om.[1]

3.6 OCaml

OCaml(*Objective Caml*) je implementirao Didier Remi(engl. *Didier Rémy*) uz pomoć Jerome Vouilona(engl. *Jérôme Vouillon*) 1996.godine. OCaml je preuzeo sve bitne karakteristike iz Caml Light-a i još dodao određene pogodnosti. Osnovna prednost OCaml-a je što je to jedina široko dostupna ML implementacija koja uključuje objektni sistem. OCaml uključuje dva prevodioca: kompajler bajt-koda koji proizvodi kod za prenosni tumač bajt-kodova OCaml i kompajler izvornog koda koji proizvodi efikasan kod za mnoge arhitekture. Prednost OCaml-a u odnosu na druge OOP jezike, npr: Java i C++ je što omogućava korišćenje prednosti objektno orijentisane paradigme(korišćenje klasa, binarnih metoda, pravljenja sopstvenih tipova) na siguran način, bez dodatnih provera zbog osobine statičke tipiziranosti.[3] Krajem 90-tih godina popularnost OCaml-a raste pa je zbog toga unapređen brojnim korisničkim bibliotekama i alatima koji su omogućili korišćenje OCaml-a kako u oblasti grafičkog korisničkog interfejsa, tako i baza podataka, Veb programiranja i u mnogim drugim oblastima.

Neki od glavnih predstavnika jezika koji su se razvili iz OCaml-a su: F# i Reason.

3.7 F#

F#(*F Sharp*) je jedan od najpopularnijih jezika koji se razvio iz OCaml-a 2005.godine kao .NET implementacija, pod uticajem i drugih jezika poput Haskell-a, C#-a, Python-a. U središtu dizajna ovog jezika su ključne ideje ML jezika.Programski jezik F# je dizajnirao i implementirao Don Sajm (eng. *Don Syme*) iz Microsoft Research-a.On je prvobitno imao ideju da preusmeri SML na .NET, ali je ipak 2001.godine odlučio da implementira Caml.NET koji je spojio prednosti OCaml-a i .NET platforme.[4] Početkom 2005.godine Don Sajmonova ideja biva prihvaćena i OCalm.NET je preimenovan u F#, što se zadržalo i danas.F# danas pored funkcionalnog programiranja ima primenu i u veb i skript programiranju, finansijskom programiranju, mašinskom učenju, pravljenju igrica.[9]

3.8 Reason

Reason(*Reason ML*) je nova sintaksa i alatni lanac koji je proširenje OCaml-a. Napravio ga je Jordan Valke(eng. *Jordan Walke*) 2016.godine.Reason nudi sintaksu poznatu programerima JavaScript-a[5] sa prednostima preuzetim iz OCaml-a. Reason se može smatrati čvrstim, statički tipiziranim, bržim i jednostavnijim rođakom JavaScript-a.Reason ima identičnu sintaksu Javi Script-u, a ono što dobija iz OCaml-a su: snažan sistem tipova, jednostavnost, kompajler za izvorni kod, sistem koji lako uočava i uklanja greške i druge pozitivne karakteristike. Zbog toga ga koriste Facebook, Viska, Rung, Backtrace i mnogi drugi korisnici.[8]

Još neki jezici koji su usvojili neke od karakteristika OCaml-a su: Meta OCaml,Scala, Haxe, Opa, Elm. A sigurno je da će se sa razvojem programskih jezika pojaviti još mnogo jezika koji će prihvatiti koncepte ML-a nastale još 70-tih godina.

[4]NET je softverska platforma otvorenog koda koju je kreirao Microsoft, za izgradnju različitih vrsta aplikacija.

[5]JavaScript je skriptni programski jezik koji se prvenstveno koristi za definisanje funkcionalnosti web stranica na klijentskoj strani. Dinamičan, slabo tipiziran jezik, sa skromnom podrškom za objektno orijentisano programiranje, Java Script je implementacija standarda ECMAscipt-a.

4 Zaključak

U ovom tekstu ukratko su predstavljene osnove razvoja programskog jezika OCaml, kao i njegovih najuticajnijih prethodnika.Prikazano je elementarno razvojno stablo ovog jezika sa jezicima iz kojih se razvio i sa par njegovih naslednika.

Literatura

[1] Caml. Zvanična stranica programskog jezika Caml. https://caml.inria.fr/.

[2] Jon D. Harrop. *OCaml for Scientists*. Flying Frog Consultancy Ltd, 2005.

[3] Jason Hickey. *Introduction to Objective Caml*. Cambridge University Press, 2008.

[4] Anil Madhavapeddy Jason Hickey and Yaron Minsky. *Real World Ocaml*. O'Reilly Media, June 2013.

[5] Robin Milner. A proposal for standard ml.

[6] OCaml. Zvanična stranica programskog jezika OCaml. https://ocaml.org/.

[7] Lawrence C. Paulson. *ML for the Working Programmer*, chapter Chapter 1:Standard ML.

[8] Reason. Zvanična stranica programskog jezika Reason. https://reasonml.github.io/.

[9] Don Syme. The early history of f# (hopl iv - first draft). https://fsharp.org/history/.