

# Neki elementi kompiliranja funkcionalnih programskih jezika

Ajzenhamer Nikola  
Bukurov Anja  
Stanković Vojislav  
Stanković Una

25. maj 2017

# Uvod

Funkcionalna paradigma, Džon Bakus, 1977.

# Osnovni pojmovi

- Lambda račun
  - svojstva: jednostavnost i izražajnost
  - primena:  $(f \ a\_1 \ a\_2 \ \dots \ a\_n)$
  - redukcija:  $(+ \ 1 \ 2) \rightarrow 3$
  - apstrakcija:  $(\lambda x.E)$
- Polimorfizam
  - Polimorfni programski jezici, polimorfne funkcije, polimorfni tipovi
  - Parametarski polimorfizam

# Transformacije lambda računa

- Efikasnost izvršavanja je jedan od najvažnijih problema
- Lambda račun se koristi zbog jednostavnosti i izražajnosti
- 2 grupe transformacija:
  - jednostavnije (lokalne): umetanje, simplifikacija
  - složenije (globalne): analiza strogosti

# Umetanje

- Dobar metod za unapređivanje performansi programa
- Osnovni princip: funkcijski poziv zameniti telom funkcije
- Tri transformacije
  - 1 samo umetanje (engl. inlining itself)
  - 2 eliminacija mrtvog koda (engl. dead code elimination)
  - 3  $\beta$ -odsecanje (engl.  $\beta$ -reduction)

## Primer

$$\begin{aligned}
 &\text{let } \{f = \lambda x.x*4\} \text{ in } (f (a*b - c)) + a*d \xrightarrow{\text{inline } f} \\
 &\text{let } \{f = \lambda x.x*4\} \text{ in } ((\lambda x.x*4) (a*b - c)) + a*d \\
 &\xrightarrow{\text{dead } f} ((\lambda x.x*4) (a*b - c)) + a*d \\
 &\xrightarrow{\beta} (\text{let } x = a*b - c \text{ in } x*4) + a*d
 \end{aligned}$$

# Uparivanje šablona

- Funkcije nad običnim tipovima mogu se definisati preko različitih slučajeva
- Slučajevi su dati šablonima
- Promenljive šablona vezuju se za odgovarajuće promenljive komponente vrednosti kojoj šablon odgovara

## Primer

```
fibonaci n  
| n==0 = 1  
| n==1 = 1  
| otherwise = (fibonaci (n-1))+(fibonaci (n-2))
```

## Zaključivač tipova

- Moderni jezici imaju svojstvo koje omogućava programeru da ne navodi tipove objekata
- Od velike koristi programeru jer mu ukazuje na greške
- Pri izvršavanju se neće javiti greške poput upotrebe promenljive tipa `bool` kao da je tipa `int`
- Proces zaključivanja tipova
  - uparivanje tipova operatora
  - instanciranje tipova promenljivih

# Motivacija

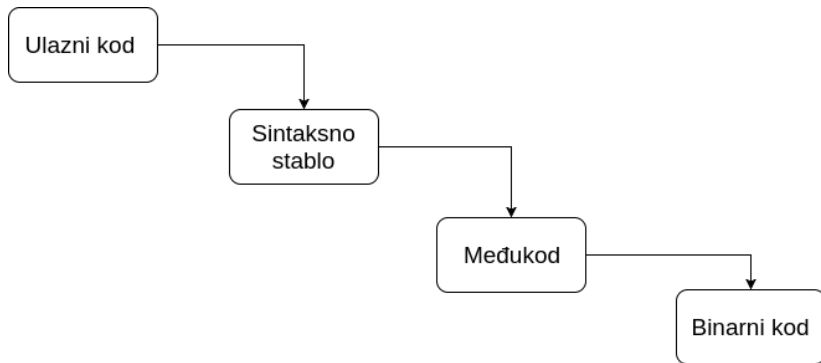
- Otpaci su delovi memorije koji nisu dostupni za alociranje
- Sakupljač otpadaka eliminiše otpatke
- Ukoliko se otpatci ne eliminišu dolazi do curenja memorije
- Sakupljač otpadaka je potreban funkcionalnim programskim jezicima



## Tipovi sakupljača otpadaka

- Markirajući sakupljač otpadaka
- Sakupljač otpadaka sa brojanjem referenci
- Prepisujući sakupljač otpadaka
- Generacijski sakupljač otpadaka

# Motivacija



Slika 1 : Vizuelni prikaz prevođenja kôda.

# Vrste apstraktnih mašina

- SECD mašina
- STG mašina
- G mašina

# SECD mašina

- Jedna od prvih mašina za izvršavanje funkcionalnih programskih jezika
- Osnovna uloga je izvršavanje kompiliranog koda
- Formalno, SECD mašina je torka četiri liste sa precizno definisanim skupom operacija nad njima
- Komponente torke su četiri steka:
  - S (engl. stack)
  - E (engl. environment)
  - C (engl. control)
  - D (engl. dump)

# G mašina

- Osnovna uloga G mašine je redukovanje grafa izračunavanja
- Osnovna ideja je da:
  - program predstavimo grafom
  - evaluacijom deljenog podgrafa automatski razrešimo sve izraze koji pokazuju na njega
  - graf "prepišemo" evaluacijom
- Važni koncepti:
  - ne postoje promenljive, već imenovani izrazi
  - vrednosti funkcije ne zavise ni od čega, osim od argumenata funkcije

## Zaključak

- Fokusirali smo se na određeni podskup tehnika i procesa koji omogućavaju efikasno kompiliranje koda
- Lambda račun kao međujezik za kompiliranje funkcionalnih programskih jezika
- Efikasan izvršni kôd je veoma važan za sve programske jezike
- Polimorfna provera tipova je korisno svojstvo programskih jezika koje veoma olakšava posao programerima
- Programiranje u funkcionalnim programskim jezicima se ne može zamisliti bez podrške koju pružaju sakupljači otpadaka
- Apstraktne mašine predstavljaju prelaz između jezika visokog nivoa i arhitekture niskog nivoa

# Literatura



Simon L. Peyton Jones

The Implementation of Functional Programming Languages  
Prentice Hall 1987.



R. Wilhelm and H. Seidl

Compiler Design  
Springer 2010.



Robert W. Sebesta

Concepts of programming languages, 10th ed.  
Pearson 2009.

Hvala na pažnji!