

## Sinteza programa

Anja Ivanišević  
Ivan Ristović  
Milana Kovačević  
Vesna Katanić

maj 2018.

# Šta je sinteza programa?

- ▶ Oblast koja se bavi automatskim generisanjem programa
- ▶ Vizija:

*Nema više programiranja, sve se generiše automatski!*

- ▶ Pričaćemo o:
  - ▶ primenama
  - ▶ izazovima
  - ▶ tehnikama

# Primene - Programiranje vodjeno primerima

	A	B
1	Email	Column 2
2	Nancy.FreeHafer@fourthcoffee.com	nancy freehafer
3	Andrew.Cencici@northwindtraders.com	andrew cencici
4	Jan.Kotas@litwareinc.com	jan kotas
5	Mariya.Sergienko@gradicdesigninstitute.com	mariya sergienko
6	Steven.Thorpe@northwindtraders.com	steven thorpe
7	Michael.Neipper@northwindtraders.com	michael neipper
8	Robert.Zare@northwindtraders.com	robert zare
9	Laura.Giussani@adventure-works.com	laura giussani
10	Anne.HL@northwindtraders.com	anne hl
11	Alexander.David@contoso.com	alexander david
12	Kim.Shane@northwindtraders.com	kim shane
13	Manish.Chopra@northwindtraders.com	manish chopra
14	Gerwald.Oberleitner@northwindtraders.com	gerwald oberleitner
15	Amr.Zaki@northwindtraders.com	amr zaki
16	Yvonne.McKay@northwindtraders.com	yvonne mckay
17	Amanda.Pinto@northwindtraders.com	amanda pinto

**Figure:** Automatske transformacije alata FlashFill

# Neke od oblasti primene sinteze programa

- ▶ Priprema podataka
- ▶ Grafika
- ▶ Sugestije prilikom kodiranja
- ▶ Superoptimizacija
- ▶ Konkurentno programiranje
- ▶ Popravka koda

# Primene - Popravka koda - Primer

inb	Ulaz		Izlaz	
	usep	dsep	expected	actual
1	0	100	0	0
1	11	110	1	0
0	100	50	1	1
1	-20	60	1	0
0	0	10	0	0

```
int buggy(int inb, int usep, int dsep)
{
    int bias;
    if (inb)
        bias = dsep; //fix: bias = usep+100
    else
        bias = usep;
    if (bias > dsep)
        return 1;
    else
        return 0;
}
```

**Figure:** Primer koda sinteziranog od strane programa *SemFix* koristeći skup ulaznih i izlaznih test primera.

# Izazovi

- ▶ Potproblemi:
  - ▶ Definisanje specifikacija željenog programa
  - ▶ Pretraživanje prostora mogućih programa u potrazi za onim koji zadovoljava definisane specifikacije
    - ▶ Enumerativna pretraga
    - ▶ Deduktivna pretraga
    - ▶ Tehnika sa ograničenjima
    - ▶ Induktivna i statistička tehnika pretrage
- ▶ Veličina željenog programa eksponencijalno utiče na veličinu prostora programa

# Izazovi - Pretraživanje prostora programa - Enumerativna pretraga

Koraci:

- ▶ Opisati prostor pretrage kome se nalazi željeni program
- ▶ Numerisati (sortirati) programe po osobinama
- ▶ Izvršiti *čišćenje*: Odbaciti sve programe koje ne zadovoljavaju specifikacije
- ▶ Pretražiti preostali prostor programa i naći rešenje

Mana: Poluodlučivost

# Izazovi - Pretraživanje prostora programa - Deduktivna pretraga

- ▶ Formalna specifikacija željenog programa
- ▶ Rešenje se sintetiše postupkom dokazivanja teorema, logičkim zaključivanjem i razrešavanjem ograničenja
- ▶ Tehnika odozgo nadole: podeli-pa-vladaj
- ▶ Deljenje problema na potprobleme nije moguće u opštem slučaju
- ▶ Kombinovanje sa enumerativnom pretragom



# Izazovi - Pretraživanje prostora programa - Tehnika sa ograničenjima

- ▶ Prilagođavanje zadatim ograničenjima
- ▶ Koraci:
  - ▶ Generisanje ograničenja
  - ▶ Razrešavanje ograničenja

# Izazovi - Pretraživanje prostora programa - Statistička pretraga

Upotreba statističkih metoda za usmeravanje pretrage

- ▶ *Mašinsko učenje*
- ▶ *Genetsko programiranje*
- ▶ *Probabilističko zaključivanje*

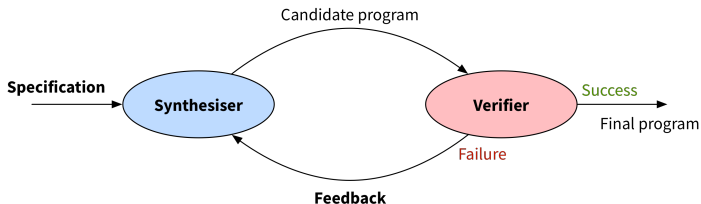
# Izazovi - Pretraživanje prostora programa - Induktivna pretraga

- ▶ Nadogradnja tehnike pretrage sa ograničenjima
- ▶ Iterativni postupak: svakom iteracijom se generišu ograničenja
- ▶ Rešavačem se dođe do mogućeg rešenja a zatim se ispita da li je ono zadovoljavajuće kao opšte rešenje
- ▶ Može da koristi tehnike mašinskog učenja - *Aktivno učenje*
- ▶ *CEGIS*

- ▶ Ideja:
  - ▶ Definiše se specifikacija programa u vidu formule
  - ▶ SMT rešavač pronalazi program koji zadovoljava specifikaciju
- ▶ Problem: previše ulaza
- ▶ *Koji je najmanji podskup ulaza koji je potrebno razmatrati da bi se sintetisao program koji zadovoljava date specifikacije?*
- ▶ CEGIS pokušava da reši ovaj problem
- ▶ Iterativno se povećava prostor pretrage i pronalazi program kandidat za rešenje
- ▶ Drugi SMT rešavač pronalazi kontraprimer za nađeni kandidat
- ▶ Ako ne postoji kontraprimer, kandidat je traženi program

# CEGIS - Arhitektura

- ▶ Pretraga vođena kontraprimerima  
(eng. *Counterexample-Guided Inductive Synthesis*)
- ▶ Dve faze:
  - ▶ *Faza sinteze* - pronalazi program kandidat
  - ▶ *Faza verifikacije* - proverava da li kandidat zadovoljava specifikaciju



**Figure:** CEGIS petlja

- ▶ Da bismo u potpunosti definisali CEGIS sintezu programa, potrebno je odgovoriti na sledeća pitanja:
  - ▶ Kako treba da izgleda specifikacija traženog programa?
  - ▶ Kako ćemo vršiti sintezu programa kandidata?
  - ▶ Kako da proverimo da li program kandidat zadovoljava specifikacije?
  - ▶ Kako da prosledimo povratne informacije za buduće kandidate?

# CEGIS - Sinteza vodjena uzorom

- ▶ eng. *Oracle-guided synthesis*
- ▶ Pretpostavlja postojanje uzora (npr. imlementacija programa)
- ▶ Biblioteka komponenti za kreiranje programa
- ▶ Primer:  
program(x,y):  
  o1 = add(x, y)  
  o2 = add(o1, y)  
  o3 = sqrt(o1)  
  return o3

## CEGIS - Sinteza vodjena uzorom

- ▶ Faza verifikacije: *Da li postoji program  $P'$ , različit od kandidata za rešenje  $P$ , koji takođe zadovoljava sve test primere, ali se na nekom ulazu  $z$  razlikuje od  $P$ ?*
- ▶ Povratni korak: kreira novi test primer na osnovu ulaza  $z$
- ▶ Faza validacije: potvrđuje da program zadovoljava sve ulaze



# CEGIS - Stohastička superoptimizacija

- ▶ Traži se brži ili efikasniji ekvivalent polaznog programa
- ▶ Faza sinteze:
  - ▶ novi program sintetišemo na osnovu tekućeg programa
  - ▶ novi program prihvatamo sa određenom verovatnoćom
  - ▶ verovatnoća je veća što su polazni i ciljni program sličniji
- ▶ Faza verifikacij: proverava da li su ciljni i program kandidat isti
- ▶ Povratni korak:
  - ▶ poredi prethodno prihvaćeni i novodobijeni program
  - ▶ određuje koji od njih dalje razmatra

# CEGIS - Enumerativna pretraga

- ▶ Specifikacija - konačan skup test primera
- ▶ Gramatika opisuje ciljni jezik (`add(x, sub(x,y))`)
- ▶ Faza sinteze: pretražuje sve moguće programe
- ▶ Faza verifikacije: proverava da li program zadovoljava sve test primere
- ▶ Povratni korak:
  - ▶ razmatra samo različite programe
  - ▶ različiti programi daju različite rezultate na istom test primeru
  - ▶ Na dubini  $k$ , ispituju se svi programe koji imaju oblik `operacija(a,b)`, gde su  $a$  i  $b$  bilo koji izrazi dubine  $k - 1$

# Zaključak

- ▶ Da li će programeri moći da prestanu da govore računarima *kako* da rade, već da se fokusiraju na to da im kažu *šta* treba da urade?
- ▶ Najveći potencijal: induktivna sinteza programa
- ▶

# Pitanja

???

Hvala na pažnji!