Automatsko otkrivanje prekoračenja bafera u programskom jeziku C

Milena Vujošević–Janičić
mentor prof. Dušan Tošić
Matematički fakultet, Univerzitet u Beogradu
Studentski trg 16, Beograd, Srbija
www.matf.bg.ac.yu/~milena

Seminar za automatsko rezonovanje
Beograd, 26. Decembar, 2007.

Apstrakt

Prekoračenje bafera je važan problem za kvalitet i bezbednost softvera. Tehnike za automatsko detektovanje prekoračenja bafera dele se na dinamičke i statičke tehnike. Dinamičke tehnike analiziraju program u fazi izvršavanja dok statičke tehnike analiziraju izvorni kôd i imaju za cilj otkrivanje mogućih prekoračenja bafera pre nego što se program pusti u rad. U radu je prikazan nov statički sistem za automatsko otkrivanje prekoračenja bafera i njegova prototip implementacija. Sistem nakon parsiranja transformiše i analizira izvorne komande programa. U okviru analize, koristi biblioteku uslova na osnovu koje generiše uslove ispravnosti komandi koje proverava automatski dokazivač teorema. Arhitektura sistema je modularna i fleksibilna, pa omogućava izmene komponenti sistema i jednostavnu komunikaciju sa različitim spoljašnjim sistemima.

Plan

- Prekoračenje bafera
- Statička i dinamička analiza
- Sistem FADO
- Tekuća pitanja
- Zaključci i dalji rad

Plan

- **Prekoračenje bafera**
- Statička i dinamička analiza
- Sistem FADO
- Tekuća pitanja
- Zaključci i dalji rad

Prekoračenje bafera

- *Prihvatnik* ili *bafer* (eng. buffer) je blok memorije rezervisan za privremeno skladištenje podataka. *Prekoračenje bafera* (eng. buffer overflow ili buffer overrun) nastaje usled greške u programu koja omogućava upisivanje sadržaja van granica rezervisane memorije bafera. Prekoračenje bafera može da se desi u svim delovima memorije pridružene programu.

- Primeri:

```
int niz[10];
niz[5] = 5;
niz[10] = 10;
```

```
char src[200];
char dst[100];
fgets(src,200,stdin);
strcpy(dst,src);
```

Prekoračenje bafera — Uzroci

- Priroda programskog jezika C omogućava greške prekoračenja bafera zato što:
  - Ne postoje automatske provere granica rezervisane memorije dodeljene nizovima ili pridružene pokazivačima.
  - Mnoge funkcije za rad sa niskama iz standardne `C` biblioteke su nebezbedne (kao što su, na primer, funkcije `strcpy`, `strcat`, `sprintf` i `gets`).
  - Programeri često podrazumevaju da su pozivi ovih funkcija bezbedni ili rade pogrešne provere.

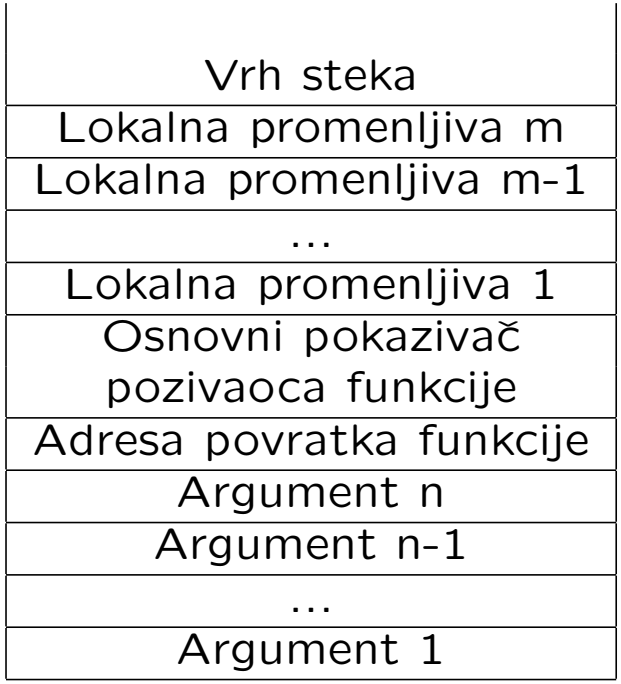
Prekoračenje bafera — Posledice

- Prekoračenje bafera može da se zloupotrebi na veliki broj načina i može da dovede do neočekivanog toka izvršavanja programa ili do kraha programa. Prekoračenja bafera predstavljaju 50% svih ranjivosti softvera.

- Najpoznatije zloupotrebe prekoračenja bafera:
  - *Morris worm* — *Unix sendmail* i *Finger* (1988.)
  - *Code Red worm* — *Microsoft Internet Information Services 5.0* (2001.)
  - *SQL Slammer worm* — *Microsoft SQL Server 2000* (2003.)

Prekoračenje bafera — Posledice

- Najčešća i najlakša eksploatacija se dešava za prekoračenja bafera koji se nalaze na steku.



Plan

- Prekoračenje bafera
- **Statička i dinamička analiza**
- Sistem FADO
- Tekuća pitanja
- Zaključci i dalji rad

Statička i dinamička analiza

- Standardno testiranje nije dovoljno za eliminisanje problema prekoračenja bafera.

- Problem automatskog otkrivanja prekoračenja bafera privlači puno pažnje poslednjih desetak godina.

- Postoje dva osnovna pristupa problemu:

  – Statičke tehnike analiziraju izvorni kôd i imaju za cilj otkrivanje mogućih prekoračenja bafera pre nego što se program pusti u rad.

  – Dinamičke tehnike analiziraju program u fazi izvršavanja.

Statička i dinamička analiza — Statička analiza

- Leksičke tehnike (ITS4 (2000), RATS (2001), Flawfinder (2001))

- Semantičke tehnike
  - BOON (Univ. of California, Berkeley, USA, 2000)
  - Splint (Univ. of Virginia, USA, 2001)
  - CSSV (Univ. of Tel-Aviv, Israel, 2003)
  - ARCHER (Stanford University, USA, 2003)
  - UNO (Bell Laboratories, 2001)
  - PolySpace C Verifier (PolySpace Tehnologies)

Statička i dinamička analiza — Dinamička analiza

- Dinamičko testiranje (Purify, gdb, mini-simulation: PROTOS, ubacivanje grešaka: FIST, Fuzz, Ballista);
- Dinamička prevencija zasnovana na korišćenju specijalizovanih kompajlera (StackGuard, PointGuard, ProPolice; StackShield; CCured, Cyclone);
- Pristup zasnovan na bibliotekama funkcija (Libsafe, Libverify, StrSafe);
- Pristup zasnovan na korišćenju operativnog sistema (OpenBSD, nadgradnja kernela za Linux).

Plan

- Prekoračenje bafera
- Statička i dinamička analiza
- **Sistem FADO**
- Tekuća pitanja
- Zaključci i dalji rad

Sistem FADO

- *FADO* — *F*lexible *A*utomated *D*etection of Buffer *O*verflows

- Sistem pripada grupi statičkih metoda koji se zasnivaju na sematničkoj analizi.

- Arhitektura sistema je modularna i fleksibilna, omogućava izmene komponenti sistema i jednostavnu komunikaciju sa različitim spoljašnjim sistemima.

Parser i generator međukôda
– parsiranje
– generisanje međukôda

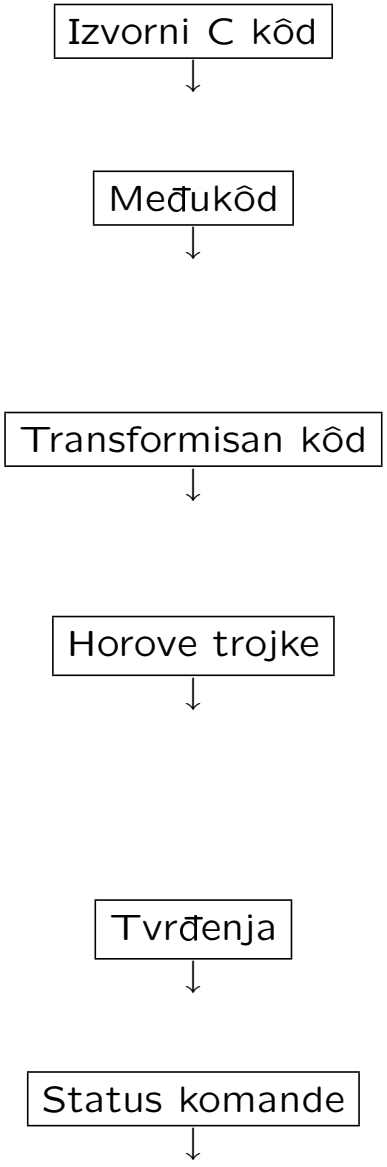
Transformator kôda
– eliminisanje višestrukih deklaracija
– eliminisanje složenih uslova
– eliminisanje sporednih efekata
– ...

Biblioteka uslova i generisanje uslova
– unifikacija sa odgovarajućim slogom biblioteke uslova
– generisanje uslova za svaku komandu pojedinačno
– ažuriranje stanja za nizove komandi

Generator i optimizator tvrđenja ispravnosti i neispravnosti
– razrešavanje preduslova i postuslova funkcija
– eliminisanje irelevantnih konjunkata
– evaluacija
– apstrakcija

Automatski dokazivač teorema
– obrađivanje ulaznih formula u smt-lib formatu
– vraćanje rezultata

Prikaz rezultata
– obezbeđivanje objašnjenja za status komande



Parser i generator međukôda
– parsiranje
– generisanje međukôda

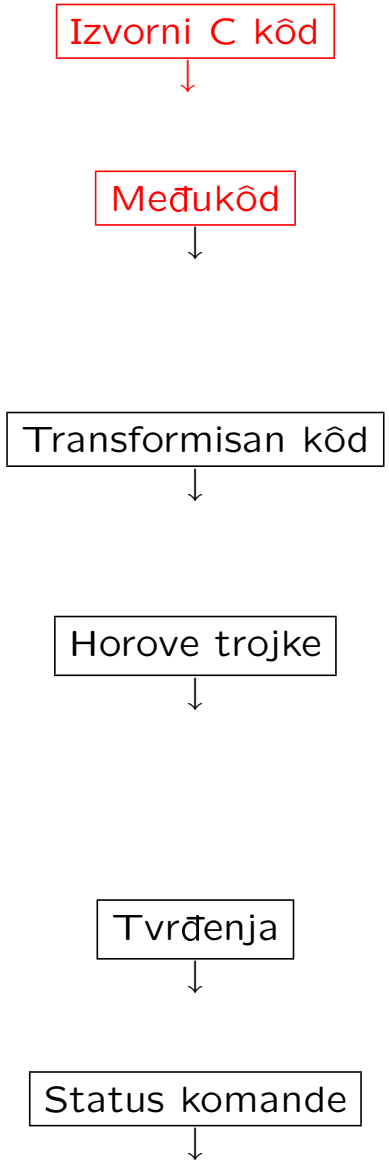
Transformator kôda
– eliminisanje višestrukih deklaracija
– eliminisanje složenih uslova
– eliminisanje sporednih efekata
– ...

Biblioteka uslova i generisanje uslova
– unifikacija sa odgovarajućim slogom biblioteke uslova
– generisanje uslova za svaku komandu pojedinačno
– ažuriranje stanja za nizove komandi

Generator i optimizator tvrđenja ispravnosti i neispravnosti
– razrešavanje preduslova i postuslova funkcija
– eliminisanje irelevantnih konjunkata
– evaluacija
– apstrakcija

Automatski dokazivač teorema
– obrađivanje ulaznih formula u smt-lib formatu
– vraćanje rezultata

Prikaz rezultata
– obezbeđivanje objašnjenja za status komande



Parser i generator međukôda
– parsiranje
– generisanje međukôda

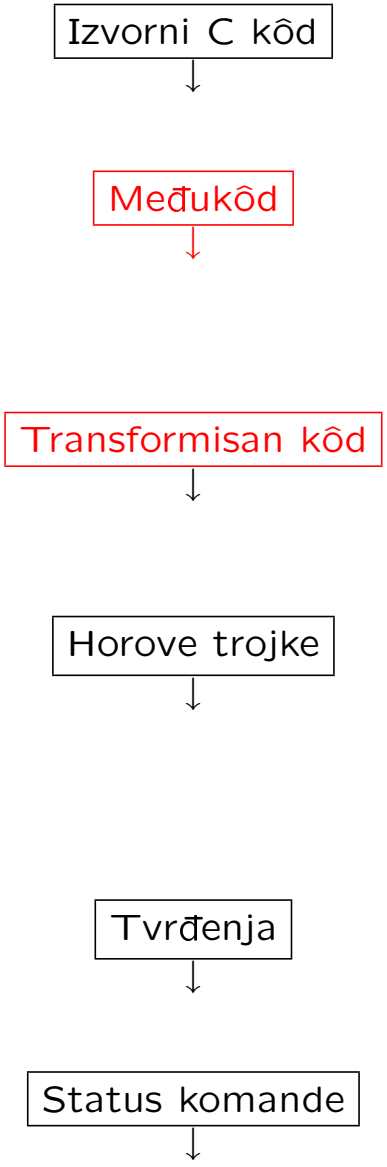
Transformator kôda
– eliminisanje višestrukih deklaracija
– eliminisanje složenih uslova
– eliminisanje sporednih efekata
– ...

Biblioteka uslova i generisanje uslova
– unifikacija sa odgovarajućim slogom biblioteke uslova
– generisanje uslova za svaku komandu pojedinačno
– ažuriranje stanja za nizove komandi

Generator i optimizator tvrđenja ispravnosti i neispravnosti
– razrešavanje preduslova i postuslova funkcija
– eliminisanje irelevantnih konjunkata
– evaluacija
– apstrakcija

Automatski dokazivač teorema
– obrađivanje ulaznih formula u smt-lib formatu
– vraćanje rezultata

Prikaz rezultata
– obezbeđivanje objašnjenja za status komande



Parser i generator međukôda
– parsiranje
– generisanje međukôda

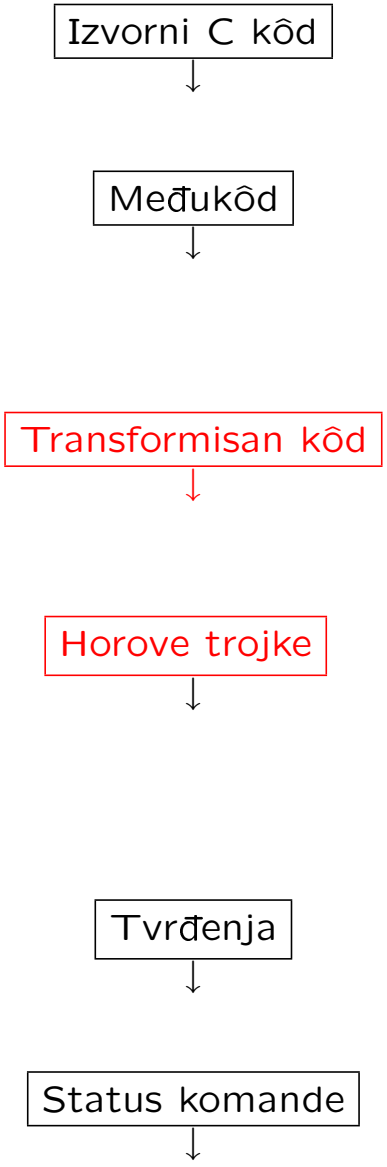
Transformator kôda
– eliminisanje višestrukih deklaracija
– eliminisanje složenih uslova
– eliminisanje sporednih efekata
– ...

Biblioteka uslova i generisanje uslova
– unifikacija sa odgovarajućim slogom biblioteke uslova
– generisanje uslova za svaku komandu pojedinačno
– ažuriranje stanja za nizove komandi

Generator i optimizator tvrđenja ispravnosti i neispravnosti
– razrešavanje preduslova i postuslova funkcija
– eliminisanje irelevantnih konjunkata
– evaluacija
– apstrakcija

Automatski dokazivač teorema
– obrađivanje ulaznih formula u smt-lib formatu
– vraćanje rezultata

Prikaz rezultata
– obezbeđivanje objašnjenja za status komande



Sistem FADO — Biblioteka uslova

- Biblioteka uslova čuva trojke oblika (*preduslov*, *komanda*, *postuslov*). Semantika sloga (ϕ, F, ψ) biblioteke uslova je:
  - da bi F bila bezbedna komanda, uslov ϕ mora da važi;
  - da bi F bila pogrešna komanda, uslov $\neg\phi$ mora da važi;
  - nakon izvršavanja komande F, važi uslov ψ.

- Biblioteka je spoljašnja i korisnik je može menjati. Inicijalno, sadrži informacije o standardnim C operatorima i funkcijama.

Sistem FADO — Modelovanje semantike programa

Za definisanje uslova korektnosti komandi programa koristi se funkcija $value$ i dve funkcije za rad sa pokazivačima $size$ i $used$:

- $value$ vraća tekuću vrednost promenljive
- $size$ vraća tekući broj elemenata alocirane memorije bafera
- $used$ vraća tekući broj iskorišćenih elemenata bafera tipa `char*`

Ove funkcije imaju po dva argumenta: ime promenljive i stanje (timestamp) promenljive koje oslikava dinamičku prirodu promenljivih i memorijskog prostora (na primer, $value(k, 0)$, $used(s, 1)$). Stanja funkcija $value$, $size$, i $used$ se ažuriraju sa ciljem da se uzme u obzir širi kontekst komandi.

Sistem FADO — Biblioteka uslova

Pretpostavimo da biblioteka uslova sadrži sledeće slogove:

preduslov	komanda	postuslov
–	`char x[N]`	$size(x,1) = value(N,0)$
$size(x,0) \geq value(y,0)$	`fgets(x,y,z)`	$used(x,1) \leq value(y,0)$

Na primer, postuslov $used(x,1) \leq value(y,0)$ govori da je prostor koji koristi x *nakon* izvršavanja komande `fgets(x,y,z)` manji ili jednak vrednosti promenljive y *pre* izvršavanja ove komande.

Primer:

```
char src[200];
fgets(src,200,stdin);
```

preduslov	komanda	postuslov
–	`char src[200]`	$size(src,1) = value(200,0)$
$size(src,0) \geq value(200,0)$	`fgets(src,200,stdin)`	$used(src,1) \leq value(200,0)$

Parser i generator međukôda
– parsiranje
– generisanje međukôda

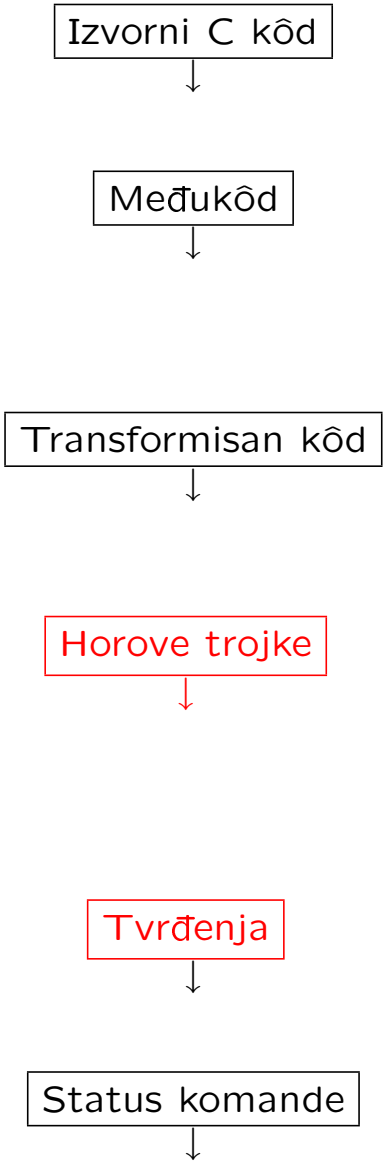
Transformator kôda
– eliminisanje višestrukih deklaracija
– eliminisanje složenih uslova
– eliminisanje sporednih efekata
– ...

Biblioteka uslova i generisanje uslova
– unifikacija sa odgovarajućim slogom biblioteke uslova
– generisanje uslova za svaku komandu pojedinačno
– ažuriranje stanja za nizove komandi

Generator i optimizator tvrđenja ispravnosti i neispravnosti
– razrešavanje preduslova i postuslova funkcija
– eliminisanje irelevantnih konjunkata
– evaluacija
– apstrakcija

Automatski dokazivač teorema
– obrađivanje ulaznih formula u smt-lib formatu
– vraćanje rezultata

Prikaz rezultata
– obezbeđivanje objašnjenja za status komande



Sistem FADO — Tvrđenja ispravnosti

Za komandu C neka je Φ konjunkcija svih postuslova za sve komande koje prethode komandi C. Tvrđenja ispravnosti i neispravnosti komande se generišu na sledeći način: komanda C je

- *bezbedna* ako je valjana formula $\Phi \Rightarrow preduslov(C)$
- *pogrešna* ako je valjana formula $\Phi \Rightarrow \neg preduslov(C)$
- *nebezbedna* ako ne važi ni jedno od prethodnih tvrđenja
- *nedostupna* ako važe oba prethodna tvrđenja.

Sistem FADO — Tvrđenja ispravnosti

Tvrđenja ispravnosti se, nakon pripreme, šalju na proveru dokazivaču teorema. Priprema obuhvata:

- Razrešavanje preduslova i postuslova funkcija
- Eliminaciju irelevantnih konjunkata. Na primer,
  $value(x, 0) = value(5, 0) \wedge value(y, 2) = used(z, 2) \Rightarrow value(x, 0) \geq value(3, 0)$
  se transformiše u
  $$value(x, 0) = value(5, 0) \Rightarrow value(x, 0) \geq value(3, 0)$$
- Evaluaciju:
  $$value(x, 0) = 5 \Rightarrow value(x, 0) \geq 3$$

Sistem FADO — Uslovi ispravnosti

- Apstrakciju:

$$value\_x\_0 = 5 \Rightarrow value\_x\_0 \geq 3$$

- Transformaciju apstrahovanog uslova u SMT format:

```
(: formula
( or
(not (value_x_0 = 5) )
(value_x_0 >= 3)
)
```

Parser i generator međukôda
– parsiranje
– generisanje međukôda

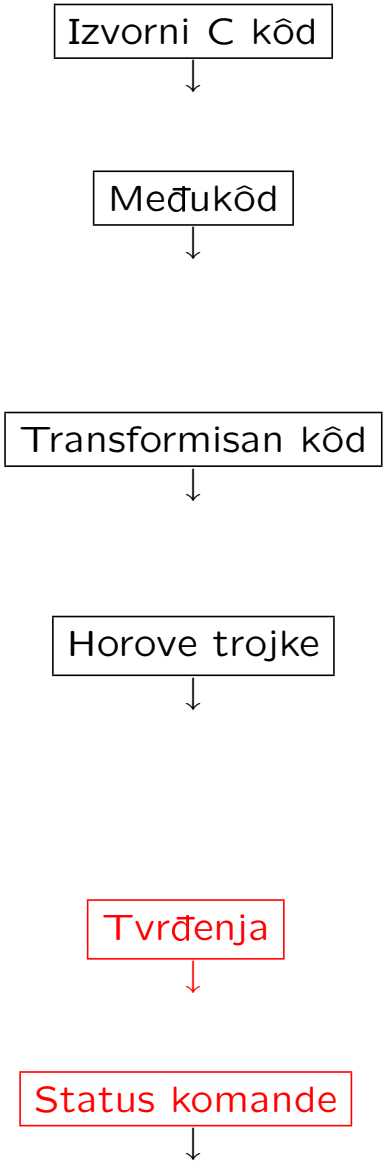
Transformator kôda
– eliminisanje višestrukih deklaracija
– eliminisanje složenih uslova
– eliminisanje sporednih efekata
– ...

Biblioteka uslova i generisanje uslova
– unifikacija sa odgovarajućim slogom biblioteke uslova
– generisanje uslova za svaku komandu pojedinačno
– ažuriranje stanja za nizove komandi

Generator i optimizator tvrđenja ispravnosti i neispravnosti
– razrešavanje preduslova i postuslova funkcija
– eliminisanje irelevantnih konjunkata
– evaluacija
– apstrakcija

Automatski dokazivač teorema
– obrađivanje ulaznih formula u smt-lib formatu
– vraćanje rezultata

Prikaz rezultata
– obezbeđivanje objašnjenja za status komande



Automatsko dokazivanje teorema

- Koristi se automatski dokazivač teorema ArgoLib.
- ArgoLib razvija Filip Marić (Matematički fakultet, Beograd)
- Koristi se modul ArgoLib-a za linearnu aritmetiku zasnovan na metodu simpleks.

Parser i generator međukôda
– parsiranje
– generisanje međukôda

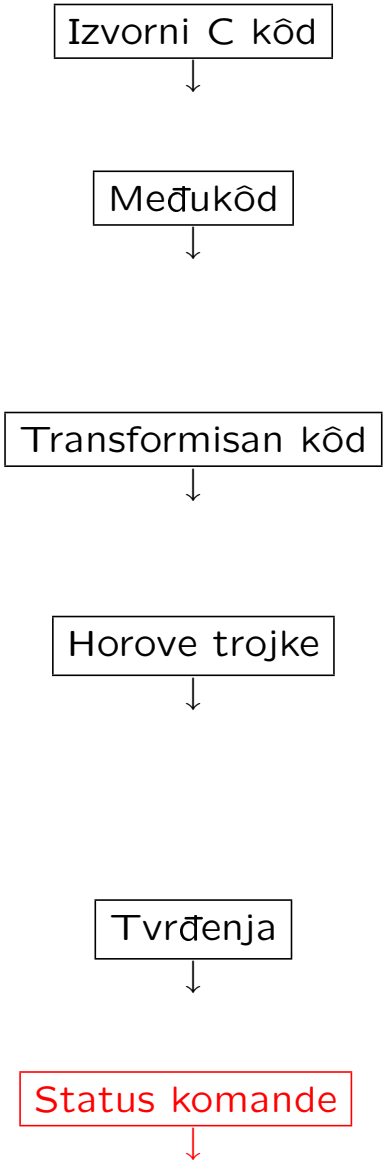
Transformator kôda
– eliminisanje višestrukih deklaracija
– eliminisanje složenih uslova
– eliminisanje sporednih efekata
– ...

Biblioteka uslova i generisanje uslova
– unifikacija sa odgovarajućim slogom biblioteke uslova
– generisanje uslova za svaku komandu pojedinačno
– ažuriranje stanja za nizove komandi

Generator i optimizator tvrđenja ispravnosti i neispravnosti
– razrešavanje preduslova i postuslova funkcija
– eliminisanje irelevantnih konjunkata
– evaluacija
– apstrakcija

Automatski dokazivač teorema
– obrađivanje ulaznih formula u smt-lib formatu
– vraćanje rezultata

Prikaz rezultata
– obezbeđivanje objašnjenja za status komande



Sistem FADO — Primer

Za fragment kôda:

```
char src[200];
fgets(src,200,stdin);
```

na osnovu biblioteke uslova

preduslov	komanda	postuslov
–	`char x[N]`	$size(x,1) = value(N,0)$
$size(x,0) \geq value(y,0)$	`fgets(x,y,z)`	$used(x,1) \leq value(y,0)$

dobijaju se uslovi za konkretne komande

preduslov	komanda	postuslov
–	`char src[200]`	$size(src,1) = value(200,0)$
$size(src,0) \geq value(200,0)$	`fgets(src,200,stdin)`	$used(src,1) \leq value(200,0)$

Sistem FADO — Primer

Na osnovu uslova za konkretne komande generišu se tvrđenja ispravnosti i nesipravnosti komandi. Za komandu `fgets(src,200,stdin)` tvrđenje ispravnosti je:

$$(0 < used(src, 1)) \wedge (0 \leq size(src, 1)) \wedge (size(src, 1) = value(200, 0))$$

$$\Rightarrow (size(src, 1) \geq value(200, 0))$$

Nakon evaluacije, tvrđenje postaje:

$$(0 < used(src, 1)) \wedge (0 \leq size(src, 1)) \wedge (size(src, 1) = 200) \Rightarrow (size(src, 1) \geq 200)$$

Nakon apstrahovanja, tvrđenje postaje:

$$(0 < used\_src\_1) \wedge (0 \leq size\_src\_1) \wedge (size\_src\_1 = 200) \Rightarrow (size\_src\_1 \geq 200)$$

i šalje se na proveru dokazivaču koji utvrđuje da je formula valjana. To znači da je komanda `fgets(src,200,stdin)` bezbedna.

Plan

- Prekoračenje bafera
- Statička i dinamička analiza
- Sistem FADO
- **Tekuća pitanja**
- Zaključci i dalji rad

Tekuća pitanja

- Problemi tehničke prirode: potpuna transformacija kôda, korisnički interfejs, format biblioteke uslova
- Petlje
- Potpuno automatizovanje procesa (razrešavanje preduslova i postuslova funkcija)
- Efikasnost

Plan

- Prekoračenje bafera
- Statička i dinamička analiza
- Sistem FADO
- Tekuća pitanja
- **Zaključci i dalji rad**

Zaključci i dalji rad

- Prekoračenje bafera je važan problem za kvalitet i bezbednost softvera. Postoji prostor da se unaprede postojeći alati.

- Prikazan je sistem za automatsko otkrivanje prekoračenja bafera i njegova prototip implementacija.

- Sistem FADO parsira i obrađuje komande izvornog programa, koristi biblioteku uslova na osnovu koje generiše tvrđenja ispravnosti komandi, koje proverava automatski dokazivač teorema. Sistem je modularan i njegove komponente se mogu lako menjati.

Zaključci i dalji rad

- U daljem radu planiramo
  - proširivanje mehanizama za rad sa petljama kako bi sistem postao saglasan;
  - proširivanje mehanizama za rad sa korisnički definisanim funkcijama kako bi sistem postao u potpunosti automatizovan;
  - povezivanje sistema sa dokazivačima teorema koji imaju širi domen;
  - proširivanje mogućnosti sistema tako da može da detektuje i druge vrste grešaka (na primer, curenje memorije).