Programiranje 2

Ispravnost programa

Jelena Graovac

www.matf.bg.ac.rs/~jgraovac

Beograd, 27. februar, 2020.

Pregled

Pregled

Ispravnost programa

- Jedno od centralnih pitanja u razvoju programa je pitanje njegove ispravnosti (korektnosti)
- Neki primeri najopasnijih grešaka:
  - Eksplozija rakete Ariane 1996. god (zbog konverzije 64-bitnog realnog u 16-bitni ceo broj)
  - Pad orbitera poslatog na Mars 1999. god (deo softvera je koristio metričke, a deo engleske jedinice)

Ispravnost programa

- Neki primeri najzanimljivijih grešaka:
  - Microsoft Excel 2007 - 77.1 * 850 prikazuje kao 100,000 (greške u algoritmu formatiranja brojeva pre prikazivanja)
  - U Los Andelesu je 14. septembra 2004. godine više od četiristo aviona u blizini aerodroma istovremeno izgubilo vezu sa kontrolom leta (greška prekoračenja u brojaču milisekundi)
  - Pad satelita Kriosat 2005. koštao je Evropsku Uniju oko 135 miliona evra
  - Više od pet procenata penzionera i primalaca socijalne pomoći u Nemačkoj je privremeno ostalo bez svog novca 2005. god (da bi se dobio desetocifreni zapis svih brojeva računa, kod starijih računa koji su imali osam ili devet cifara brojevi su dopunjavani nulama, ali sa desne umesto sa leve strane kako je trebalo)

Osnovni pristupi ispitivanju ispravnosti programa

- Postupak pokazivanja da je program ispravan naziva se verifikovanje programa
- Ispravnost programa počiva na pojmu specifikacije
- Specifikacija je, neformalno, opis željenog ponašanja programa koji treba napisati
- Specifikacija se obično zadaje u terminima preduslova tj. uslova koje ulazni parametri programa moraju da zadovolje, kao i postuslova tj. uslova koje rezultati izračunavanja moraju da zadovolje.
- Verifikovati program znači dokazati da on zadovoljava specifikaciju

Osnovni pristupi ispitivanju ispravnosti programa

- Dva osnovna pristupa verifikaciji su:
  - dinamička verifikacija - koja podrazumeva proveru ispravnosti u fazi izvršavanja programa, najčešće putem testiranja
  - statička verifikacija - koja podrazumeva analizu izvornog kôda programa, često korišćenjem formalnih metoda i matematičkog aparata

Osnovni pristupi ispitivanju ispravnosti programa

- Veoma važno pitanje je pitanje zaustavljanja programa:
  - parcijalna korektnost – podrazumeva da neki program, ukoliko se zaustavi, daje korektan rezultat (tj. rezultat koji zadovoljava specifikaciju)
  - totalna korektnost – podrazumeva da se program za sve (specifikacijom dopuštene) ulaze zaustavlja, kao i da su dobijeni rezultati parcijalno korektni

Testiranje

- Neka tvrdenja o programu je moguće testirati, dok neka nije.
  - Program ima prosečno vreme izvršavanja 0.5 sekundi — proverivo testovima
  - Prosečno vreme izmedu dva pada programa je najmanje 8 sati sa verovatnoćom 95% — proverivo testovima
  - Prosečno vreme izvršavanja programa je dobro — suviše neodređeno da bi moglo da bude testirano
  - Prosečno vreme izmedu dva pada programa je najmanje 8 godina sa verovatnoćom 95% — u principu proverivo testovima ali nije praktično izvodivo

Testiranje

- Iscrpno testiranje
  - Na primer, iscrpno testiranje korektnosti programa koji sabira dva 32-bitna broja, zahtevalo bi ukupno $2^{32} * 2^{32} = 2^{64}$ različitih testova. Ako bi jedan test trajao 1ns, onda bi testiranje trajalo 570 godina.
- Umesto iscrpnog testiranja, koristi se tehnika testiranja tipičnih ulaza programa kao i specijalnih, karakterističnih ulaznih vrednosti
  - Na primer, testiranje korektnosti programa koji sabira dva 32-bitna broja podrazumevalo bi testiranje nekoliko slučajno odabranih parova brojeva, dok bi za specijalne slučajeve mogli biti proglašeni slučajevi kada je neki od sabiraka 0, 1, -1, najmanji negativan broj, najveći pozitivan broj i slično
- Cilj testiranja jeste pronalaženje grešaka u programima

Testiranje

- Neke od metoda testiranja su:
  - Testiranje zasebnih jedinica (engl. unit testing)
    - U ovom metodu testiranja, nezavisno se testovima proverava ispravnost zasebnih jedinica koda.
  - Regresiono testiranje (engl. regression testing)
    - proveravaju se izmene programa kako bi se utvrdilo da se nova verzija ponaša isto kao stara
  - Integraciono testiranje (engl. integration testing)
    - Primenjuje se kada se više programskih modula objedinjuje u jednu celinu i kada je potrebno proveriti kako funkcioniše ta celina i komunikacija između njenih modula.
  - Testiranje valjanosti (engl. validation testing)
    - Testiranje valjanosti treba da utvrdi da sistem ispunjava zadate zahteve i izvršava funkcije za koje je namenjen.

Dinamičko verifikovanje programa

- Debagovanje
  - Pojednostavljeno rečeno, testiranje je proces proveravanja ispravnosti programa, sistematičan pokušaj da se u programu (za koji se pretpostavlja da je ispravan) pronade greška.
  - Debagovanje se primenjuje kada se zna da program ima grešku.
  - kdbg – ilustracija rada
- Otkrivanje curenja memorije
  - program valgrind sa alatkom memcheck – ilustracija rada

Statičko ispitivanje ispravnosti programa

- Proveravanje kritičnih mesta u programima
  - Proveravanje graničnih vrednosti

```
int i;
char s[MAX];
for (i = 0; (s[i] = getchar()) != '\n' && i < MAX; ++i);
s[--i] = '\0';
```

Statičko ispitivanje ispravnosti programa

- Proveravanje kritičnih mesta u programima
  - Proveravanje graničnih vrednosti

```
int i;
char s[MAX];
for (i = 0; (s[i] = getchar()) != '\n' && i < MAX; ++i);
s[--i] = '\0';


for (i=0; i < MAX; i++)
  if ((s[i] = getchar()) == '\n')
    break;
s[i] = '\0';
```

Statičko ispitivanje ispravnosti programa

- Proveravanje kritičnih mesta u programima
  - Proveravanje graničnih vrednosti

```
int i;
char s[MAX];
for (i = 0; (s[i] = getchar()) != '\n' && i < MAX; ++i);
s[--i] = '\0';
```

```
for (i=0; i < MAX; i++)
  if ((s[i] = getchar()) == '\n')
    break;
s[i] = '\0';
```

```
for (i=0; i < MAX-1; i++)
  if ((s[i] = getchar()) == '\n' || s[i]==EOF)
    break;
s[i] = '\0';
```

Statičko ispitivanje ispravnosti programa

- Proveravanje kritičnih mesta u programima
  - Proveravanje pre-uslova
    - Odbrambeno programiranje
  - Proveravanje povratnih vrednosti
- Formalno ispitivanje ispravnosti programa
  - Semantika uobičajenih tipova podataka i operacija u programima razlikuje od uobičajene semantike matematičkih operacija nad celim i realnim brojevima - veliki izazov za verifikaciju
  - Funkcionalni programi

```
int mnozi(int x, int y) {
  if (x == 0)
    return 0;
  else
    return mnozi(x - 1, y) + y;
}
```

Statičko ispitivanje ispravnosti programa

- Formalno ispitivanje ispravnosti programa
  - Funkcionalni programi (nastavak)

```
unsigned faktorijel(unsigned n) {
  if (n == 0)
    return 1;
  else
    return n*faktorijel(n-1);
}
```

Statičko ispitivanje ispravnosti programa

- Formalno ispitivanje ispravnosti programa
  - Funkcionalni programi (nastavak)

```
unsigned faktorijel(unsigned n) {
  if (n == 0)
    return 1;
  else
    return n*faktorijel(n-1);
}


float stepen_brzo(float x, unsigned k) {
  if (k == 0)
    return 1.0f;
  else if (k % 2 == 0)
    return stepen_brzo(x * x, k / 2);
  else
    return x * stepen_brzo(x, k - 1);
}
```

Statičko ispitivanje ispravnosti programa

- Formalno ispitivanje ispravnosti programa
  - Verifikacija imperativnih programa i invarijante petlji

```
z = 0; n = 0;
while(n < x) {
  z = z + y;
  n = n + 1;
}
```

Invarijanta petlje

$(n \leq x) \wedge (z = n * y)$

Statičko ispitivanje ispravnosti programa

- Formalno ispitivanje ispravnosti programa
  - Verifikacija imperativnih programa i invarijante petlji

```
r = x; q = 0;
while (y <= r) {
  r = r - y;
  q = q + 1;
}
```

Invarijanta petlje

$x = q * y + r$

Statičko ispitivanje ispravnosti programa

- Formalni dokazi i Horova logika
- Ispitivanje zaustavljanja programa

Pregled

Obnavljanje

- Šta je cilj verifikacije programa?

Obnavljanje

- Šta je cilj verifikacije programa?
- Testiranjem programa se može:
  (a) dokazati da je program korektan za sve ulaze,

Obnavljanje

- Šta je cilj verifikacije programa?
- Testiranjem programa se može:
  (a) dokazati da je program korektan za sve ulaze,
  (b) opovrgnuti da je program korektan za sve ulaze,

Obnavljanje

- Šta je cilj verifikacije programa?
- Testiranjem programa se može:
  (a) dokazati da je program korektan za sve ulaze,
  (b) opovrgnuti da je program korektan za sve ulaze,
  (c) dokazati da se program zaustavlja za sve ulaze,

Obnavljanje

- Šta je cilj verifikacije programa?
- Testiranjem programa se može:
  (a) dokazati da je program korektan za sve ulaze,
  (b) opovrgnuti da je program korektan za sve ulaze,
  (c) dokazati da se program zaustavlja za sve ulaze,
  (d) dokazati da se program ne zaustavlja za neke ulaze.

Obnavljanje

- Šta je cilj verifikacije programa?
- Testiranjem programa se može:
  (a) dokazati da je program korektan za sve ulaze,
  (b) opovrgnuti da je program korektan za sve ulaze,
  (c) dokazati da se program zaustavlja za sve ulaze,
  (d) dokazati da se program ne zaustavlja za neke ulaze.
- Kako se zove provera korektnosti u fazi izvršavanja programa?

Obnavljanje

- Šta je cilj verifikacije programa?
- Testiranjem programa se može:
  (a) dokazati da je program korektan za sve ulaze,
  (b) opovrgnuti da je program korektan za sve ulaze,
  (c) dokazati da se program zaustavlja za sve ulaze,
  (d) dokazati da se program ne zaustavlja za neke ulaze.
- Kako se zove provera korektnosti u fazi izvršavanja programa?
- Koji je najčešći vid dinamičke verifikacije programa?

Obnavljanje

- Šta je cilj verifikacije programa?
- Testiranjem programa se može:
  (a) dokazati da je program korektan za sve ulaze,
  (b) opovrgnuti da je program korektan za sve ulaze,
  (c) dokazati da se program zaustavlja za sve ulaze,
  (d) dokazati da se program ne zaustavlja za neke ulaze.
- Kako se zove provera korektnosti u fazi izvršavanja programa?
- Koji je najčešći vid dinamičke verifikacije programa?
- Koliko bi, metodom iscrpnog testiranja, bilo potrebno izvršiti testova programa za sabiranje dva neoznačena 32-bitna cela broja?

Obnavljanje

- Šta se može dokazati testiranjem programa?

Obnavljanje

- Šta se može dokazati testiranjem programa?
- Da li uz pomoć debagera može da se:

Obnavljanje

- Šta se može dokazati testiranjem programa?
- Da li uz pomoć debagera može da se:
  (a) efikasnije kompilira program?

Obnavljanje

- Šta se može dokazati testiranjem programa?
- Da li uz pomoć debagera može da se:
  (a) efikasnije kompilira program?
  (b) lakše otkrije greška u programu?

Obnavljanje

- Šta se može dokazati testiranjem programa?
- Da li uz pomoć debagera može da se:
  (a) efikasnije kompilira program?
  (b) lakše otkrije greška u programu?
  (c) izračuna složenost izvršavanja programa?

Obnavljanje

- Šta se može dokazati testiranjem programa?
- Da li uz pomoć debagera može da se:
  (a) efikasnije kompilira program?
  (b) lakše otkrije greška u programu?
  (c) izračuna složenost izvršavanja programa?
  (d) registruje „curenje memorije“ u programu?

Obnavljanje

- Šta se može dokazati testiranjem programa?
- Da li uz pomoć debagera može da se:
  (a) efikasnije kompilira program?
  (b) lakše otkrije greška u programu?
  (c) izračuna složenost izvršavanja programa?
  (d) registruje „curenje memorije“ u programu?
  (e) umanji efekat fragmentisanja

Pregled

Literatura

- Slajdovi su pripremljeni na osnovu knjige
  Predrag Janičić, Filip Marić: Programiranje 2
- Za pripremu ispita, slajdovi nisu dovoljni, neophodno je koristiti knjigu!