TEHNIKA I INFORMATIKA U OBRAZOVANJU
3. Internacionalna Konferencija, Tehnički fakultet Čačak, 7–9. maj 2010.
TECHNICS AND INFORMATICS IN EDUCATION
3rd International Conference, Technical Faculty Čačak, 7–9th May 2010.

UDK: 371.26 Stručni rad

KOREKTNOST PROGRAMA

Branko Markoski[1], Sofija Hotomski[2], Natasa Đuričić[3], Sanja Stanisavljev[4], Zdravko Ivanković[5]

Rezime: U radu je opisan opšti okvir za istraživanje problema korektnosti programa metodom rezolucijskog pobijanja. Dokazivanje korektnosti programa svedeno je na nalaženje dokaza za tvrdjenje da dati niz formula predstavlja izvodjenje u formalnoj teoriji tipa specijalnog predikatskog računa. Dokazivanje korektnosti programa je teorijski problem koji ima veliki praktični značaj i rešava se u okviru analize programa. Njemu srodan teorijski problem je projektovanje korektnih programa koji se rešava na drugi način - u okviru sinteze programa. Evidentno je da se ovi procesi uzajamno prepliću jer su analiza i sinteza programa usko povezani. Navedeni su ilustrativni primeri koji su realizovani na prologolikom LP-jeziku (bez ogranicenja na Hornove klauzule i bez konačnog neuspeha). Date su i osnovne informacije o LP-jeziku. Pokazano je kako se Pascal-program izvršava u dokazivaču LP-sistema.

Ključne reči: *testiranje, test, rezolucija, korektnost programa*

CORRECTNESS PROGRAM

Summary*: This paper describes a general framework for the problem of correctness of the method resolution avoidance. Proving correctness of programs is reduced to finding evidence to claim that a given formula is a performance series in the formal theory of a special type predicate account. Proving the correctness of the theoretical problem that has great practical significance and resolves within the analysis program. To him the problem is related theoretically correct design programs that are solved in another way - in the synthesis program. It is evident that these processes are mutually intertwined because the analysis and synthesis of closely related. Listed illustrative examples that were implemented in LP-language (no restrictions on Horne clauses without a finite failure). Are given basic information about the LP-language. It is shown how the Pascal program is running in LP systems indicator.*

Key words: *testing, test, resolution, program correctness*

[1] Prof. dr Branko Markoski, Tehnički fakultet „Mihajlo Pupin“ Zrenjanin, E-mail: markoni@uns.ac.rs
[2] Sofija Hotomski, Fakultet Tehničkih Nauka, Novi Sad, E-mail: sofka.hrcak@gmail.com
[3] Nataša Đuričić, stručni saradnik, TF „Mihajlo Pupin“ Zrenjanin, E-mail: natasa@tfzr.uns.ac.rs
[4] Sanja Stanisavljev, stručni saradnik, TF „Mihajlo Pupin“ Zrenjanin, E-mail: sanja@tfzr.uns.ac.rs
[5] Msc Zdravko Ivanković, TF „Mihajlo Pupin“, Zrenjanin, E-mail: zdravko@tfzr.uns.ac.rs

1. UVOD

Poslednjih godina porasla je aktivnost na polju verifikacije programa. Cilj ovih napora je da se konstruišu kompjuterski sistemi za određivanje je li dati program ispravan, u smislu zadovoljavanja datih specifikacija. Ovi pokušaji doživaljavaju sve veći uspeh. Automatska korektura ispravnosti velikih programa još je daleko, ali izgleda očito da su se razvile tehnike koje će biti korisne u praksi, za nalaženje bagova u programima i za potvrđivanje njihove ispravnosti. Verifikacija programa se može osloniti na tehnike automatskog dokazivanja teoreme. Ove tehnike otelotvoruju principe deduktivnog zaključivanja, iste one koje koristi i programeri prilikom samog konstruisanja programa. Zašto ne bi koristili iste principe u sistemu za automatsku sintezu, koji može da konstruiše program umesto da samo dokazuje njegovu ispravnost? Dijkstra je otišao toliko daleko da je rekao kako ne bi trebalo da automatizujemo programiranje čak i kad bi mogli, jer bi to oduzelo tom poslu svo uživanje. Korektnost i pouzdanost (neki autori umesto termina pouzdanost koriste termin ‘’robusnost’’).

2. MAN-OV, HOARE-OV I DIJKSTRIN PRISTUP REŠAVANJU KOREKTNOSTI PROGRAMA

Problematiku analize i sinteze programa (koja se rešava putem rezolucijske procedure dokazivanja i dedukovanja odgovora), objavio je u svojim radovima Z. Manna [Mann, 1969-1974]. Drugi prilaz rešavanju problema korektnosti programa je primena aksiomatskog definisanja semantike programskog jezika Pascal i to u obliku posebnih pravila izvođenja programske logike [Floyd, 1967], [Hoare, 1969], [Hoare, Wirth, 1972-1973]. Poređenjem gore navedenih pristupa rešavanju problema korektnosti programa zaključuje se da su koncepcijski bitno različiti, ali sa jednom zajedničkom osobinom a to je deduktivni sistem na predikatskom jeziku. Najpre će biti prikaz Mann-ov pristup rešavanju problema korektnosti programa.

Ukoliko se posmatra jedan program, postavlja se pitanje terminiranja (okončanja) i korektnosti, a ukoliko se posmatraju dva programa postavlja se pitanje ekvivaletnosti datih programa. Ukoliko se izvrši interpretacija slobodnih promenljivih u parcijalnom interpretiranom programu nastaje realizovan program. Rad ovakvog programa prati se pomoću izvršene sekvence. Realizovan program, koji se smatra determinističkim ima jednu izvršnu sekvencu, a u slučaju da uopšte ne postoji nema ni jednu izvršnu sekvencu. Suprotno tome, kada je u pitanju parcijalno interpretirani program, uočava se više različitih izvršnih sekvenci. U malo pre navedenoj vrsti programa za svaki interpretirani predikat se zna kada je tačan, a kada nije tačan, što bi značilo da su u zavisnosti od ulaznih promenljivih mogući različiti putevi izvršavanja. Razmatrajući apstraktni program, dolazi se do zaključka da on ima samo jednu izvršnu sekvencu, gde se ne zna da li je tačan predikat P ili njegova negacija.

Sada će biti prikazan Hoare - ov pristup rešavanju problema korektnosti programa. Osnovna relacija {P} S {Q} predstavlja specifikaciju programa S sa sledećim značenjem: ako je predikat P na ulazu zadovoljen (tačan) pre izvršavanja programa S, onda je predikat Q na izlazu zadovoljen (tačan) posle izvršenja programa S. Da bi se dokazala korektnost programa S, neophodno je dokazati relaciju {P}S{Q}, pri čemu ulazne vrednosti promenljivih treba da zadovolje predikat P i izlazne vrednosti promenljivih treba da zadovolje predikat Q. Pošto nije dokazano da S terminira, već je to samo pretpostavka onda

se može reći da je definisana parcijalna korektnost programa. Ukoliko se dokaže da S terminira i da je relacija {P}S{Q} zadovoljena kaže se da je S u potpunosti korektan. Za projektovanje programa koristi se ovako određen pojam korektnosti. Druga tehnika dokazivanja je zasnovana na pojmu simboličnog izvršenja. Programske linije se obrađuju istim redom kao što će biti izvršavane, za razliku od redosleda "unatraške" kod prvog metoda.

Sada će biti prikaz Dijkstrin pristup pristup rešavanju problema korektnosti programa. Sistem koji se smatra interesantnim je onaj koji će kada se pokrene iz početnog stanja "okončati" u konačnom stanju (koji po pravilu zavisi od izbora početnog stanja). Pretpostavi se da se vrednost ulaza odražava u izboru početnog stanja i da se vrednost izlaza odražava u konačnom stanju. "Uslov koji karakteriše skup svih početnih stanja iz kojih aktiviranje sigurno vodi ispravnom okončanju događanja tako da ostavlja sistem u konačnom stanju koje zadovoljava dati zaključak zove se najširi preduslov u odnosu na taj zaključak" - [Dijkstra, 1988]. Ukoliko se mehanizam ili mašina kao sistem označi sa S, a željeni zaključak sa R, onda se najširi preduslov može prikazati u sledećoj formi:

wp(S,R),

gde je *wp* funkcija dva argumenta od *S* i predikata *R*. Semantiku nekog mehanizma poznajemo dovoljno dobro ako poznajemo njegov predikatni transformator, koji nam kaže kako za bilo koji zaključak *R* možemo izvesti najširi preduslov (koga smo označili sa *wp(S,R)*).

Može se reći da wp predstavlja skup svih stanja takvih da izvršenje počinje u jednom od njih. Ako S počinje u stanju koje zadovoljava R i ako izvršenje terminira onda će konačno stanje zadovoljiti R. Može se postaviti oštriji uslov da predikat P implicira R za sva stanja tj. $P \Rightarrow wp(S,R)$. Ako početno stanje zadovoljava predikat P tada:

1. S obavezno terminira
2. R postaje tačno

Pošto $P \Rightarrow wp(S,R)$ važi za sva stanja, iz ovoga bi se moglo izvesti da wp(S,R) je istinito kad god je P istinito.

Sada će biti dat jedan mali primer koji je realizovan u BASELOG sistemu koji ima kakarkeristike da ne postoji ograničenje na Hornove klauzule i prisustvo CWA-kontrolera. Linearna rezolucija sa markiranim literalima predstavlja deduktivnu osnovu sistema, dok CWA-kontroler omogućuje izbor režima rada u zatvorenom, otvorenom ili delimično zatvorenom/otvorenom svetu. U komentarima se umesto termina BASELOG –sistem upotrebljava kraći termin LP. Analiza korektnosti programa vršena je za manje programe, ali se može koristiti i za složenije programe. Korišćeni su programi sa prostim tipovima podataka, a od strukturnih tipova rađeno je sa nizovima. Primer:

```
begin
  max:=x[1]; i:=0;
      while i<=n do
          begin
              i:=i+1;
              if x[i]>max then max:=x[i];
          end;
end.
s(s(d(max,x[1[),d(i,0)),w(i<=n,s(d(i,i+1),if(x[i]>max,d(max,x[i])))))
```

Označimo*:*

- konstantom b - predikat i<=n,
- sa b1- predikat x[i]>max*,*
- konstantom t1 - term x[i],
- konstantom t2 - term x[i]

Dobija se zapis*:*

s(s(d(max,t1),d(i,0)),w(b,s(d(i,i+1,if(b1,d(max,t2)))))
1
/IM1(X0,Y0)/IM(X2,Y2)O(X1,V1)~K(X1,s(h,g),Y1)~K(Y1,w(b,if(b1,e)),V1)&
11
~K(Y1,d(max,t1),V1)K(Y1,h,V1)& / rezerva za skraćivanje dužine zapisa
~K(Y1,d(i,0),V1)K(Y1,g,V1)& / rezerva za skraćivanje dužine zapisa
~K(Y1,d(max,t2),V1)K(Y1,e,V1)& / rezerva za skraćivanje dužine zapisa
~K(X1,Y1,U1)~K(U1,Y2,V1)K(X1,s(Y1,Y2),V1)& /aksioma za sekvencu
~K(k(X1,Y1),Z1,U1)~IM(k(X1,n(Y1)),U1)K(X1,if(Y1,Z1),U1)& /aksioma za if
~K(k(X1,V2),U0,X1)K(X1,w(V2,U0),k(X1,ng(V2)))& /aksioma za while
~K(t(X1,Z1,Y1),d(Z1,Y1),X1)& /aksioma za dodelu
~IM(X2,Y1)~K(Y1,U0,V1)K(X2,U0,V1)& /aksioma za kosekvencu
~IM(Y1,V1)~K(X1,U0,Y1)K(X1,U0,V1)& /aksioma za kosekvencu
~IM1(X0,Y0)IM(X0,Y0)&
~O(X1,Y1)&
0
0

LP sistem generiše sledeće pobijanje:

broj generisanih rezolventi = 14
maksimalno dostignuti nivo = 15
STAMPA DOKAZA
nivo na kojem je generisan prazan sastavak = 15
NIVO= 15; rezolventa:
&
DOKAZ JE ODSTAMPAN

Sada još treba dokazati saglasnost, tj. potvrditi da važi:

$(X_\theta \Rightarrow Y_\theta^{Z\theta}{}_{T\theta}) \wedge (U \Rightarrow P_\theta, \ \wedge \ Q_\theta \Rightarrow V)$ što se nalazi na NIVO= 14; rezolventa:
/IM1(k(k(X1,b),b1),t(X1,max,t2))/IM(k(k(X1,b),n(b1)),X1)O(t(t(X1,i,1),max,t1),k(X1,ng(b
)))&

Vraćanjem oznaka na domenski nivo dobijamo:

$(X1 \wedge (i<=n) \Rightarrow X1^{i}_{i+1}{}^{p}{}_{p/i})i \ (U \Rightarrow X1^{i}{}_{0}{}^{p}{}_{x}) \wedge (X1 \wedge \neg (i<=n) \Rightarrow V)7$

Stavljajući za X1:

$\max = x(i) \ \underset{j=1}{\overset{i}{\wedge}}(x(j) \geq x(i))$ dobijamo sledeće tačne implikacije:
$i \in [1..n]$

$$\max = x(i+1) \bigwedge_{j=1}^{i+1} (x(j) \geq x(i))$$

$$=> \max = x(i+1) \bigwedge_{j=1}^{i} (x(j) \geq x(i)) \wedge (x(i+1) \geq x(i)$$

Za i=1 dobija se:

$$\max = x(1) \bigwedge_{j=1}^{1} (x(j) \geq x(i)) => \max = x(1)$$

Time je saglasnost dokazana, što je dovoljno za zaključak da je dati program (parcijalno) korektan (do na terminiranje).

3. ZAKLJUČAK

Testiranjem se utvrđuje prvo u kojoj meri program obavlja posao za koji je namenjen, a zatim i kako se ponaša u različitim eksploatacionim uslovima. Kod testiranja softvera najbitnije je odrediti niz test stavki za ono što se testira. Pre toga moramo razjasniti koje informacije se moraju nalaziti u test stavki. Najočitija informacija su ulazi kojih ima dve vrste: preduslovi (okolnosti koje postoje pre isvršenja test stavke) i stvarni ulazi, koje identifikujemo nekim metodom testiranja. Nema smisla testirati greške koje verovatno ne postoje. Mnogo je efikasnije dobro razmisliti o vrstama grešaka koje su najverovatnije (ili najštetnije) i tada odabrati metode testiranja koji će verovatno moći da otkriju takve greške. U radu je opisan opšti okvir za istraživanje problema korektnosti programa metodom rezolucijskog pobijanja Osnovu za razvoj osnovnih deduktivnih metoda i njihovih modifikacija koje se koriste za utvrđivanje korektnosti programa .Jedan od načina rešavanja problema korektnosti programa je korišćenjem pravila programske logike. On omogućuje dedukciju na ovoj osnovi ali i proveru saglasnosti konkretnih ulazno-izlaznih predikata sa dedukovanim vrednostima.

4. LITERATURA

[1] Perry, William E., "Year 2000 Software Testing" New York: John Wiley & Sons,1999.

[2] Hotomski P., Malbaski D., "Matematička logika i principi programiranja", Tehnički fakultet "M. Pupin", Zrenjanin, 2000.

[3] Pressman R.S.,"Software Engineering"A Practitioner's Approach, McGraw-Hill, New York, 1992.

[4] Jones, Capers, "Critical Problems in Software Measurement", Carlsbad, CA: Infosystems Management, 1993.

[5] Zeller A., "Yesterday, my program worked. Today, it does not. Why?", 2000., Passau, Germany.

[6] Berkovic I., Markoski B., Setrajcic J., Brtka Vladimir, Dobrilovic D. "Testing of program correctnes in formal theory", Ubiquitous Computing and Communication Journal Octobar 2009, ISSN Online 1992-8424 , Special Issue on ICIT 2009 conference - Bioinformatics and Image , Bioinformatics and Image , 7/30/2009.

[7] Маркоски Б., Хотомски П., Малбашки Д. "Verifying program corectness resolution metod", ETAI 2000, V National conference with international participation, Ohrid, FR Macedonia.

[8] Markoski B., Hotomski P., Malbaski D. "Testing the complex software", International ZEMAK symposium, Ohrid, FR Macedonia, 2004.
[9] Markoski B., Markoski S., Babic Dj,"Algorithm linear systems", ETAI 2007. Ohrid, FR Macedonia, I6 R54.
[10] Markoski B. Jevremovic D.,Malbaski D., Hotomski P."Odabir test stavki", DQM - 2006.
[11] Markoski B., Ivankovic Z., Pelemis S., Setrajcic J., Mirjanic D. "Tehnike testiranja programa", Yuinfo Kopaonik, Serbia 2009.