Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Testiranje z uporabo logičnih izrazov

Študijsko gradivo za interno uporabo pri predmetu Testiranje in kakovost (TiK)

Ljubljana, 2017/18

Kazalo

Pokritja

Štirje abstraktni modeli pokritij zajemajo vse možne testne tehnike:

1. **pokritje z grafi (ang. Graph Coverage)**
2. **pokritje z logični izrazi (ang. Logic Coverage)**
3. **delitev prostora vhodnih podatkov (ang. Input Space Partitioning)**
4. **sintaksno testiranje (ang. Syntax-Based Testing)**

Pokritje logičnih izrazov

- Logični izrazi so pogosti del PO
- Znan pristop: zahteva po pokritju logičnih izrazov izhaja iz letalske industrije USFAA
- Viri logičnih izrazov:
  - odločitve v programih
  - končni avtomati
  - zahteve
- Namen testov je izbira ustrezne podmnožice pravilnih trditev izmed logičnih izrazov

Pokritje logičnih izrazov

Predikat (logični izraz) je izraz, katerega rezultat je vrednost boolean (da/ne)

Vsebuje: - logične spremenljivke

- primerjalne izraze (==,!=, <,>,<=,>=)
- klice logičnih funkcij

Predikat (lahko) vsebuje logične operatorje:

- $\neg$ – negacijo
- $\wedge$ – konjunkcijo
- $\vee$ – disjunkcijo
- $\rightarrow$ – implikacijo
- $\oplus$ – ekskluzivni ali
- $\leftrightarrow$ – ekvivalenco

Predikat brez logičnih operatorjev je **trditev** (clause)

Pokritje logičnih izrazov

Težave pri določanju predikatov iz naravnega jezika

Pristop: razvijemo model PO, ki je zapisan z enim ali več predikati. Testi morajo pokriti neko kombinacijo trditev

Kratice:

- P – množica predikatov
- p – en predikat iz te množice
- C – množica trditev v P
- C_p – množica trditev v predikatu p
- c – posamezna trditev v C

Pokritje logičnih izrazov

1.Kriterij pokritja predikatov (PC): Za vsak p iz P, TR vsebuje dve zahtevi: p ovrednoti kot TRUE in p ovrednoti kot FALSE.

- Če so predikati na povezavah, je to ekvivalentno EC (?)
- Kaj pa povezave brez predikatov?

2. Kriterij pokritja trditev (CC): Za vsako c iz C, TR vsebuje dve zahtevi: c ovrednoti kot TRUE in c ovrednoti kot FALSE

Problemi:

- PC ne pokrije vseh trditev, še zlasi ne pri kratkostični vezavi
- CC ne zagotavlja tudi PC – lahko CC brez PC

Pokritje logičnih izrazov

3.Kriterij pokritja kombinacij (CoC): Za vsak p iz P, TR vsebuje testne zahteve za trditve iz Cp za ovrednotenje vseh možnih kombinacij vrednosti TRUE in FALSE.

- Enostavno, razumljivo, a zelo drago
- 2^N možnosti, če N trditev
- Ni praktično, če več kot 3 trditve
- Veliko (nejasnih) rešitev

Rešitev: testiraj vsako trditev neodvisno od drugih trditev

- Zahteven problem: kaj je neodvisnost?
- Uporabljamo pristop aktivnih trditev

Pokritje logičnih izrazov

8

CC težava: različne vrednosti včasih ne vplivajo na vrednost predikata

Samo nekatere trditve so odločilni faktor pri določanju vrednosti predikata

Trditev c iz predikata p je pomembna trditev (major clause) za določanje p v primeru, ko so ostale nepomembne trditve (minor clause) takšne, da spremenjena vrednost c spremeni vrednost predikata p – to pomeni določiti aktivno trditev.

Cilj: določiti test za vsako trditev, ki določa vrednost predikata

Več (malenkostno spremenjenih) možnosti z (zelo) različnimi učinki

Pokritje logičnih izrazov

9

4. Kriterij pokritja aktivnih trditev (ACC): Za vsak p iz P in vsako pomembno trditev c iz Cp, izberi nepomembne trditve tako, da c določa p. TR vsebuje dve zahtevi: c naj se ovrednoti TRUE in c naj se ovrednoti FALSE.

- Nejasnost: kakšna naj bo vrednost nepomembnih trditev
- Tri rešitve:
  - lahko so poljubne vrednosti
  - biti morajo vedno enake vrednosti
  - biti morajo take, da prisilijo predkat, da je TRUE in FALSE

Pokritje logičnih izrazov

5. Kriterij splošnega pokritja aktivnih trditev – General Active Clause Coverage (GACC): Za vsak p iz P in vsako pomembno trditev c iz Cp, izberi nepomembne trditve tako, da c določa p. TR vsebuje dve zahtevi: c ovrednoti TRUE in c ovrednoti FALSE. Pri tem ni treba, da so vrednosti nepomembnih trditev enake takrat, ko je c enak TRUE in c enak FALSE.

- Zahtevno!
- Ni nujno, da to pomeni tudi PC (a ↔ b)
- Pravzaprav želimo preveriti, ko je vrednost p TRUE in FALSE

Pokritje logičnih izrazov

6. Kriterij omejenega pokritja aktivnih trditev – Restricted Active Clause Coverage (RACC): Za vsak p iz P in vsako pomembno trditev c iz Cp, izberi nepomembne trditve tako, da c določa p. TR vsebuje dve zahtevi: c ovrednoti TRUE in c ovrednoti FALSE. Vrednosti nepomembnih trditev morajo biti enake takrat, ko je c enak TRUE in c enak FALSE.

- Običajna zahteva pri letalstvu!
- Veliko neizvedljivih testnih zahtev
- Ni logičnega razloga za tovrstno omejitev

Pokritje logičnih izrazov

7. Kriterij povezanega pokritja aktivnih trditev – Correlated Active Clause Coverage (CACC): Za vsak p iz P in vsako pomembno trditev c iz Cp, izberi nepomembne trditve tako, da c določa p. TR vsebuje dve zahtevi: c ovrednoti TRUE in c ovrednoti FALSE. Vrednosti nepomembnih trditev morajo biti za eno vrednost c take, da je vrednost p enaka TRUE, v drugem primeru pa mora biti p enaka FALSE.

- Sodobnejša opredelitev!
- Dovoljuje različne vrednosti nepomembnih trditev
- Vsebuje PC

Pokritje logičnih izrazov

8. Kriterij pokritja neaktivnih trditev – Inactive Clause Coverage (ICC): Za vsak p iz P in vsako pomembno trditev c iz Cp, izberi nepomembne trditve tako, da c NE določa p. TR vsebuje štiri zahteve za vse c:

- **c in p ovrednoti TRUE,**
- **c ovrednoti TRUE in p ovrednoti FALSE,**
- **c ovrednoti FALSE in p ovrednoti TRUE ter**
- **c in p ovrednoti FALSE.**
- Korelacija ne vpliva, ker c ne vpliva na vrednost p!
- PC vedno velja.

Pokritje logičnih izrazov

9. Kriterij splošnega pokritja neaktivnih trditev – General Inactive Clause Coverage (GICC): Za vsak p iz P in vsako pomembno trditev c iz Cp, izberi nepomembne trditve tako, da c NE določa p. Ni treba, da so vrednosti izbranih nepomembnih trditev enake takrat, ko je c enak TRUE in c enak FALSE.

10. Kriterij omejenega pokritja neaktivnih trditev – Restricted Inactive Clause Coverage (RICC): Za vsak p iz P in vsako pomembno trditev c iz Cp, izberi nepomembne trditve tako, da c NE določa p. Vrednosti izbranih nepomembnih trditev morajo biti enake takrat, ko je c enak TRUE in c enak FALSE.

Pokritje logičnih izrazov

Običajno predikati enostavni:

- če vsebujejo eno trditev – PC
- če vsebujejo dve ali tri – CoC
- če veliki – ACC ali ICC

ODVISNOSTI:

CoC

RACC **RICC**

CACC **GICC**

GACC PC **CC PC**

CC

Pokritje logičnih izrazov

Določanje vrednosti nepomembnih trditev je za enostavne predikate preprosto

Sicer uporabimo pristop:

- $p_{c=true}$ – predikat p, ko je vrednost c enaka TRUE
- $p_{c=false}$ – predikat p, ko je vrednost c enaka FALSE
- Za določanje vrednosti nepomembnih trditev, določimo

$$p_c = p_{c=true} \oplus p_{c=false}$$

Pri zapletenih predikatih uporabimo vedno poenostavljen izraz

Če so predikati neizvršljivi, jih prepoznamo in ignoriramo (težak inženirski problem - nedoločenost)

Pokritje logičnih izrazov programske kode

Predikati = odločitve v programih

Običajno predikati manj kot 4 trditve (večinoma eno)

Če eno, vsa pokritja sovpadejo v PC

Težak kriterij testiranja, saj je treba upoštevati:

- Dosegljivost – pred uporabo PC je treba doseči mesto predikata
- Nadzor – določanje vrednosti, ki neposredno določajo predikat

Težak problem: določanje vrednosti notranjih spremenljivk

Tudi prenešene vrednosti upoštevamo kot notranje spremenljivke

Previdnost s transformacijami, ki skrivajo strukturo predikatov

Pokritje logičnih izrazov programske kode - Primer 18

```
public class Trikotnik {
private static String[] triTipi = { "", "raznostranicen
trikotnik", "enakokrak trikotnik", "enakostranicen
trikotnik", "trikotnik ne obstaja"};

  public static void main (String[] argv) {
    Scanner sc = new Scanner(System.in);
    int a, b, c, t;
    System.out.print("Stranica A: "); a = sc.nextInt();
    System.out.print("Stranica B: "); b = sc.nextInt();
    System.out.print("Stranica C: "); c = sc.nextInt();
    t = UganiTrikotnik(a,b,c);
    System.out.println ("Rezultat: " + triTipi[t]);
  }
```


Pokritje logičnih izrazov programske kode - Primer 19

```
private static int UganiTrikotnik (int s1, int s2, int s3){
    int rez;
1   if (s1 <= 0 || s2 <= 0 || s3 <= 0)  {
      rez = 4;
      return(rez); }
    rez = 0;
2   if (s1 == s2)       rez = rez + 1;
3   if (s1 == s3)       rez = rez + 2;
4   if (s2 == s3)       rez = rez + 3;
5   if (rez == 0) {
6     if (s1+s2 <= s3 || s2+s3 <= s1 ||s1+s3 <= s2)
        rez = 4;
6´    else
        rez = 1;
      return (rez);}
```

Pokritje logičnih izrazov programske kode - Primer


```
7    if (rez > 3)
       rez = 3;
8    else if (rez == 1 && s1+s2 > s3)
       rez = 2;
9    else if (rez == 2 && s1+s3 > s2)
       rez = 2;
10   else if (rez == 3 && s2+s3 > s1)
       rez = 2;
10´  else
       rez = 4;

     return (rez);
   }
}
```


Pokritje logičnih izrazov specifikacij

Lahko formalne (matematični prikaz) ali neformalne (opisne)

Običajno vsebujejo jasne logične izraze

Tipičen primer : predpogoji za izvedbo metod

Lahko se nahajajo na različnih nivojih

Pogosto jih izražamo v:

- konjuktivni normalni obliki (CNF) ali
- disjunktivni normalni obliki (DNF)

Vsaka trditev postane aktivna, če naredimo diagonalo:

false (CNF) oziroma

true (DNF)

Pokritje logičnih izrazov končnih grafov

22

Končni grafi (FSM) na vseh ravneh abstrakcije

Različne oblike predstavitve

Običajno na prehodih med vozlišči - predpogoji in sprožilci

Poišči logični izraz in ga pokrij z ustreznim pokritjem

Težave:

- Istočasno pritisnjeni gumbi (časovna komponenta)
- Dosegljivost (začetno stanje)
- Pričakovano stanje ob izhodu

Testni primeri – prirejanje vrednosti spremenljivkam

Problem preslikave med FSM in programom (preslikave imen)

Pokritje logičnih izrazov - DNF

23

Običajno logične funkvcije predstavimo v DNF

Malo drugačna oblika zapisa:

- Element (literal) – trditev ali negirana trditev
- Izraz (term, implicant) – konjunkcija elementov
- DNF predikat – disjunkcija izrazov

Efekt – če je en izraz TRUE, je cel izraz TRUE

- Lahko določimo dodatna pokritja za izraze:
- Testira le TRUE vrednosti
- Dodamo DNF predikat za negacijo

Pokritje logičnih izrazov - DNF

11. Kriterij pokritja izrazov DNF – Implicant Coverage (IC): Za podano DNF predstavitev predikata f in negacije predikata f, za vsak izraz iz f in negacije f, TR vsebuje zahtevo, da ta izraz ovrednoti kot TRUE.

- Šibak kriterij

Natančneje opredelitve:

- **Pravi podizraz** (proper subterm): izraz brez ene ali več trditev
- **Primitivni izraz** (prime implicant): ko noben pravi podizraz ni izraz
- **Odvečni izraz** (redundant implicant): lahko odstranimo brez spremembe vrednosti predikata
- **Minimalni DNF zapis**: vsebuje le primitivne neodvečne izraze
- **Enolična true točka** (unique true point): za določen izraz je določanje TRUE vrednosti, da je izraz TRUE in vsi ostali FALSE.

Pokritje logičnih izrazov - DNF

12. Kriterij pokritja enoličnih TRUE točk – Unique True Points Coverage (UTPC): Za podano DNF predstavitev predikata f in negacije predikata f, TR vsebuje enolično TRUE točko za vsak izraz iz f in negacije f.

- Precej močnejši kriterij
- Po eni strani zahteva več testov kot ACC
- Vendar ne vsebuje niti GACC
- Običajno toliko testov kolikor izrazov

Dodatna opredelitev:

- **Bližnja false točka** (near false point): za določeno trditev v izrazu je določanje TRUE vrednosti, da je izraz FALSE, a če je trditev negirana ima izraz vrednost TRUE

Pokritje logičnih izrazov - DNF

13. Kriterij pokritja parov enolične TRUE točke in bližnjih FALSE točk – Unique True Point and Near False Point Pairs Coverage (CUTPNFPC): Za podano DNF predstavitev predikata f , za vsak izraz in vsako trditev, TR vsebuje enolično TRUE točko za izraz in bližnjo FALSE točko tako, da se točki razlikujeta le v vrednosti trditve.

- Le za predikat, brez negacije
- Vsebuje RACC
- Približno število testov: število izrazov * število elementov

Razredi sintaktičnih napak na DNF

Množice testov po UTPC, CUTPNFPC zagotavljajo, da najdemo napake določene vrste – iste najdejo tudi napake druge vrste.

Pokritje logičnih izrazov - DNF

DNF Razredi napak:

ENF: Expression Negation Fault	$f = ab+c$	$f' = (ab+c)'$
TNF: Term Negation Fault	$f = ab+c$	$f' = (ab)'+c$
TOF: Term Omission Fault	$f = ab+c$	$f' = ab$
LNF: Literal Negation Fault	$f = ab+c$	$f' = ab+c'$
LRF: Literal Reference Fault	$f = ab + bcd$	$f' = ad + bcd$
LOF: Literal Omission Fault	$f = ab + c$	$f' = a + c$
LIF: Literal Insertion Fault	$f = ab + c$	$f' = ab + bc$
ORF+: Operator Reference Fault	$f = ab + c$	$f' = abc$
ORF*: Operator Reference Fault	$f = ab + c$	$f' = a + b + c$

Pokritje logičnih izrazov - DNF

Diagram razmerij:

LIF

LRF

TOF LOF

LNF

ORF*

ORF+ TNF

ENF

Pokritje logičnih izrazov - DNF

Primer: TOF

- UTPC – enolične točke za vse izraze
- Najde vse TOF napake
- Glede na diagram najde tudi vse : LNF, TNF, ORF+,ENF napake

CUTNFP najde LOF in naslednike

Čeprav CUTPNFP ne vsebuje UTPC, vsebuje enake razrede v skladu z diagramom

Vsebovanost : odkrivanje napak

Obstaja še veliko (bolj zahtevnih) DNF kriterijev

Pokritje logičnih izrazov - DNF

Karnough-ov diagram (K-map):

Grafična predstavitev predikatov – z združevanjem trditev

Do 5 spremenljivk

Omogočajo enostavno določanje:

- Kdaj trditev določa predikat?
- Negacijo predikata
- Primitivnih in odvečnih izrazov
- Enoličnih TRUE točk - UTPC
- Parov enoličnih TRUE točk in bližnjih FALSE točk - CUTFNPFC

Literatura

1. **Paul Ammann, Jeff Offutt: Introduction to software testing, Cambridge University Press, 2008.**
2. **Wikipedia**