Testarea Sistemelor Software

Abordarea înlănțuită pentru generarea de date de test

**Aspecte generale**

Generarea de date de test pentru un program software este un proces prin care se urmărește găsirea unor valori de intrare care satisfac anumite criterii de testare. Există situații în care, prin aplicarea directă a metodelor structurale sau funcționale, să nu reușim să acoperim toate elementele din program pe care ne propunem să le testăm, mai ales în situațiile în care codul sursă este foarte lung sau este scris de alte persoane.

Un generator de date de test este un instrument care îi ajută pe programatori să găsească date de test pentru un anumit program. Există mai multe tipuri de generatori: generatori de date de test alese în mod aleator, generatori de date pentru parcurgerea anumitor căi în program, generatori de date pentru execuția unei anumite instrucțiuni.

Cercetătorii au constatat că graful asociat instrucțiunilor unui program nu este o metodă suficientă pentru găsirea datelor de test în proiectele de dimensiuni mari în care instrucțiunea testată poate depinde de execuția unei instrucțiuni anterioare.

Abordarea înlănțuită este o metodă derivată din algoritmul de furnizare de date de test folosit de generatorii pentru execuția unei anumite instrucțiuni. Metoda analizează dependența datelor în procesul de identificare a elementelor din program care influențează execuția instrucțiunii de testat.

Vom prezenta abordarea înlănțuită în paralel cu metoda folosită de ceilalți generatori.

Considerând un nod g în program, vrem să găsim setul de date de intrare x pentru care nodul g se execută.

**Generatorul de date de test alese în mod aleator:**

Se bazează pur și simplu pe furnizarea unui set de date alese aleator, motiv pentru care probabilitatea ca intrucțiunile programului să fie acoperite în totalitate este foarte mică.

**Generatori de date pentru parcurgerea anumitor căi din program:**

Selectează o cale a programului care conduce la execuția nodului g. Dacă nu se găsește un set de date de intrare care să determine parcurgerea căii selectate, generatorul alege altă cale pentru verificare. Procesul se repetă până când se obține un set de date de intrare valid care să aibă drept efect parcurgerea unei căi către nodul g. Metoda este destul de ineficientă pentru programele de dimensiuni mari deoarece generatorul nu știe de la început căile pentru care nu se pot găsi date de intrare și se pot consuma foarte multe resurse în testarea acestor căi.

Generatori de date pentru execuția unei anumite instrucțiuni:

Abordarea elimină procesul de selecție a unei căi. Generatorul rulează inițial programul cu date alese în mod aleator și ține evidența instrucțiunilor executate. În timpul rulării, generatorul observă dacă pentru execuția nodului g trebuie aleasă o altă cale decât cea parcursă și încearcă ulterior să caute date care să influențeze modificarea traseului doar pentru acele instrucțiuni care influențează execuția nodului.

**Abordarea înlănțuită:**

Metodele prezentate anterior se bazează pe graful programului, însă acesta nu oferă suficiente informații despre execuția unor noduri din program. De exemplu, există noduri care modifică valorile unor variabile utilizate în clauzele unor instrucțiuni condiționale și care determină, astfel, schimbarea căii în parcurgerea programului, însă execuția lor este independentă de calea aleasă de generator până la întâlnirea lor.

Abordarea înlănțuită analizează interdependența datelor în procesul de generare de date de test. Ideea de bază a acestei metode este identificarea unor secvențe de noduri care să fie executate înaintea execuției nodului g.

Inițial, metoda operează cu un set de date alese aleator. În timpul execuției programului, aceasta parcurge pe rând instrucțiunile cu scopul de a declanșa execuția nodului g. Dacă se observă că programul alege parcurgerea unei căi care nu conține nodul g, se încearcă generarea unui alt set de date de intrare care să schimbe calea urmată de program. În cazul în care nu se găsește un set de date corespunzător, abordarea înlănțuită va încerca să găsească în program nodurile care influențează execuția nodului la care s-a constatat că programul nu urmează calea dorită.

Abordarea înlănțuită generează “secvențe de evenimente”, care sunt folosite de generator pentru a urmări comportamentul programului în procesul de căutare a unui set de date valid. Prin eveniment, ne referim la execuția unui nod din program. Astfel, o ”secvență de evenimente” reprezintă o secvență de noduri care s-au executat la rularea programului.

La definirea unei ”secvențe de evenimente”, se stabilesc constrângerile care determină tranziția de la un eveniment la altul. Constrângerile sunt impuse asupra nodurilor și, prin intermediul lor, se asigură faptul că variabilele nu-și modifică valoarea până la declanșarea evenimentului următor. O ”secvență de evenimente” se numește ”realizabilă” dacă există date de intrare pentru care programul traversează secvența.

Secvențele de evenimente se generează într-o manieră arborescentă, drept consecință a execuției unui nod problemă, adică un nod care nu conduce la execuția nodului g. Secvențele de evenimente au ca punct de start prima instrucțiune din program și se finalizează cu instrucțiunea corespunzătoare nodului țintă. Inițial, setul de secvențe de evenimente cuprinde o singură secvență, formată din evenimentele corespunzătoare nodului de start și nodului țintă. Dacă pe parcursul desfășurării programului se întâlnește un nod problemă, abordarea înlănțuită va căuta în codul sursă toate nodurile în care apar variabilele din nodul problemă și în care valoarea acestora se modifică. Pentru n astfel de noduri, se vor genera n secvențe de evenimente, câte una pentru fiecare nod găsit. Secvențele de evenimente vor cuprinde evenimentele inițiale, la care se adaugă, pentru fiecare în parte, evenimentul asociat nodului corespunzător, dar și evenimentul asociat nodului problemă. Între aceste două noi evenimente se definesc, bineînțeles, constrângeri de tranziție.

Abordarea înlănțuită va încerca să caute valori de intrare pentru execuția fiecărei secvențe generate. Dacă în parcurgerea unei secvențe se întâlnește un alt nod problemă, abordarea înlănțuită va căuta nodurile din program care influențează execuția acestui nod și va genera un alt set de secvențe de evenimente, care vor cuprinde, pe lângă evenimentele secvenței curente, evenimentul asociat unui nod dintre cele găsite, precum și evenimentul corespunzător nodului problemă.

Abordarea înlănțuită continuă să valideze secvențe generate până când găsește un set de date de intrare care să traverseze una dintre secvențe. Dacă nu este găsit un set de date valid pentru una dintre secvențele generate, metoda poate identifica noi noduri problemă care să determine crearea de noi secvențe. Astfel, se poate observa organizarea arborescentă a secvențelor, toate derivând din secvența inițială, alcătuită doar din evenimentul de start și evenimentul țintă. Metoda eșuează dacă nu se găsesc date de intrare pentru niciuna dintre secvențele care se pot genera pe baza codului sursă sau când resursele alocate procesului se epuizează.

În cazul programelor foarte lungi, abordarea înlănțuită poate genera un arbore de secvențe foarte ”stufos”, iar resursele mașinii pe care rulează pot fi limitate. În aceste situații, se poate impune o limită asupra numărului de niveluri care se pot genera pentru arborele secvențelor.

De exemplu, pentru un singur nod problemă, se generează un set de secvențe de evenimente. Arborele secvențelor are, în acest caz, doar un nivel. Dacă execuția unui nod dintr-o secvență din noul set depinde de alte noduri, se generează un alt set de secvențe, derivat dintr-o secvență a setului curent. Nivelul maxim al arborelui devine 2. Se poate observa că algoritmul utilizat de generatorii de date de test pentru execuția unei anumite instrucțiuni este echivalent cu abordarea înlănțuită cu un arbore de nivel maxim 0, deoarece algoritmul respectiv returnează un mesaj de eroare dacă întâlnește un nod problemă și nu poate găsi un set de date astfel încât programul să ajungă la nodul g prin traversarea acelui nod problemă.

**Procesul de căutare folosit în abordarea înlănțuită**

Pe parcursul analizei programului, metoda identifică nodurile care pot influența execuția nodului g. Pentru a descrie comportamentul programului pentru un set inițial de date de test, abordarea înlănțuită realizează o clasificare a ramurilor pe baza impactului asupra nodului țintă. Termenul ramură face referire la o muchie din graful asociat programului.

Ramură critică: este o ramură care influențează valorile variabilelor dintr-un nod problemă. Dacă această ramură nu se execută, programul traversează nodul problemă pentru a declanșa un nou eveniment.

Remură semicritică: este o ramură care poate conduce la traversarea unei ramuri critice.

Restul ramurilor sunt considerate ramuri obișnuite, care nu influențează traseul programului între două evenimente țintă.

Clasificarea ramurilor este referitoare la 2 evenimente succesive, adică este relativă la fiecare nod problemă în parte.

O ramură din graful programului este accesibilă în funcție de rezultatul expresiei matematice asociate instrucțiunii condiționale corespunzătoare. O expresie matematică a unei intrucțiuni condiționale poartă numele de predicat și poate de fi de două feluri: boolean sau aritmetic. Predicatele boolene sunt formate din variabile de tip boolean, în timp ce predicatele aritmetice sunt simple expresii matematice relaționale. Fiecărei ramuri din program i se asociază o funcție după predicatul din instrucțiunea condițională. Funcțiile se formează rescriind expresiile aritmetice prin aducerea tuturor termenilor într-un membru, astfel încât relația să fie exprimată prin raportare la elementul nul. Predicatul este considerat adevărat dacă relația impusă între rezultatul funcției și elementul nul este adevărată. Astfel, execuția ramurii pentru un anumit set de date de intrare este dependentă de rezultatul funcției asociate.

Abordarea înlănțuită apelează funcțiile ramurilor cu datele de intrare și determină dacă predicatele furnizează răspunsul așteptat sau nu. Dacă metoda constată că datele de intrare nu conduc la execuția unui nod țintă, încercă să genereze un alt set de date, care să respecte codomeniul funcțiilor.

**Proiect realizat de**

Mesteacăn Ștefan-Lucian  
Pârjoleanu Andi-Olivian  
Rotariu Maria  
Vasile Clara-Gabriela