Generazione Automatica di Casi di Test

Fabrizio Pastore

www.fabriziopastore.com

Feedback Directed Random Testing con Randoop

Pacheco, C.; Lahiri, S. K.; Ernst, M. D. & Ball, T. Feedback-directed random test generation ICSE '07: Proceedings of the 29th International Conference on Software Engineering, IEEE Computer Society, 2007

Randoop: generazione dei test (1/3)

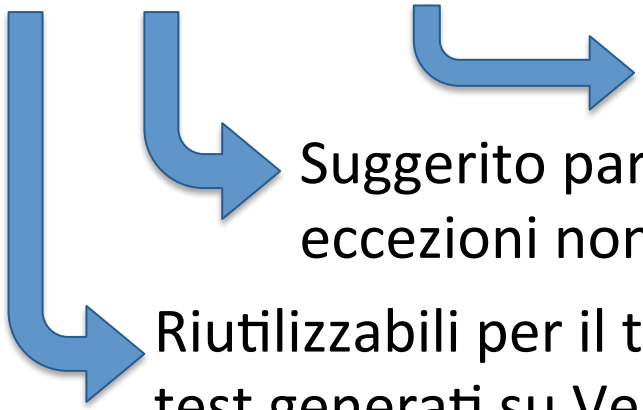
INPUTS: timeLimit, classes, contracts, filter
errorSeqs ← {} //Sequenze di metodi che portano a violazioni di contratti
nonErrorSeqs ← {} // Sequenze di metodi che non portano a violazioni

Randoop: generazione dei test (2/3)

INPUTS: timeLimit, classes, contracts, filter
while timeLimit not reached **do**
    ...
    newSeq ← extend(m, seqs, vals)
    //Scarta duplicati
    if newSeq ∈ nonErrorSeqs ∪ errorSeqs **then**
        continue
    end if

Randoop: generazione dei test (3/3)

INPUTS: timeLimit, classes, contracts, filter
while timeLimit not reached **do**
    . . .
    if *violated* = *true* **then**
        errorSeqs ← *errorSeqs* ∪ {newSeq }
    else
        nonErrorSeqs ← *nonErrorSeqs* ∪ {newSeq }
        setNonExtensible(newSeq)
    end if
end while
return <nonErrorSeqs,errorSeqs>



Hanno trovato difetti nel software

Suggerito parsing de codice, potrebbero esserci eccezioni non previste, e.g. ArrayIndexOutOfBounds

Riutilizzabili per il test di non regressione:
test generati su Versione_0 da eseguire su Versione_1

Come definire le classi da usare nei test

- solo le classi/metodi indicati attraverso i seguenti parametri sono usate/i nei test (tutte le altre classi non sono usate nei test)
  - --testclass <ClasseDaUsareNeiTest>
  - --classlist <fileContenenteElencoDiClassi>
  - --methodlist <fileContenenteElencoDiMetodi>
- è possibile specificare dei metodi da escludere
  - --omitmethods <regexMetodiDaEscludere>

Valori di Input

- Di default usa i seguenti valori
  - byte: -1, 0 1, 10, 100
  - short: -1, 0 1, 10, 100
  - int: -1, 0 1, 10, 100
  - long: -1, 0 1, 10, 100
  - float: -1, 0 1, 10, 100
  - double: -1, 0 1, 10, 100
  - char: '#', ' ', '4', 'a'
  - java.lang.String: "", "hi!”
- più qualsiasi valore restituito a runtime dai metodi
- valori scelti da utente
  - opzione --literals-file <file>
  - classe ad hoc

Esempio 1: Generazione Tests

- Generiamo Casi di Test per la Casse QueueArray
  randoop gentests --testclass dataStructures.QueueArray

Normal method executions: 1203150
Exceptional method executions: 15

Esempio 2: Test di Regressione

- A questo punto i nostri test di randoop possiamo usarli per verificare la non regressione
  - modifichiamo StackArray:95 gli facciamo ritornare null
  - eseguiamo la test suite

Cercare solo i test che falliscono

- Opzione
  - --output-tests=fail
- Esempio
  - ripristiniamo il difetto in QueueArray
    - bash restore.sh
  - eseguiamo randoop

```
randoop gentests --testclass dataStructures.QueueArray --output-tests=fail
```

  - gli oracoli automatici di Randoop non trovano nulla

Esercizio

- Generare casi di test con Randoop per StackArray e verificare quali difetti trova
- Esercizio 2
  - Definire una representation invariant per StackArray e verificare che randoop riesca ad idnetificare automaticamente dei casi che falliscono.

Oracoli di Randoop:
cioè come Randoop identifica i fallimenti

- *Equals to null:*
  - o.equals(null) should return false
- *Reflexivity of equality:*
  - o.equals(o) should return true
- *Symmetry of equality:*
  - o1.equals(o2) implies o2.equals(o1)
- *Equals-hashcode:*
  - If o1.equals(o2)==true, then o1.hashCode() == o2.hashCode()
- *No null pointer exceptions:*
  - No NullPointerException is thrown if no null inputs are used in a test.
- Class Invariant, detta Representation Invariant
  - definita nella classe con @CheckRep

Representation Invariant

- un metodo annotato con il tag @CheckRep
  - verifica la consistenza dello stato dell'oggetto
  - visibilità: pubblico
  - può restituire:
    - boolean (true stato ok, false stato non ok)
    - void (lancia eccezione se lo stato non è ok)

Representation Invariant Stack Array

```
@CheckRep
    public boolean checkState(){
        if ( topOfStack < -1 || topOfStack >= theArray.length ){
            return false;
        }

        for ( int i = topOfStack+1; i < theArray.length ; i++ ){
            if ( theArray[i] != null ){
                return false;
            }
        }
        return true;
    }
```

randoop gentests --testclass dataStructures.StackArray
--output-tests=fail --simplify-failed-tests=true

Representation Invariant QueueArray

- Provare a definire una Representation Invariant per QueueArray

Esempio per QueueArray

```
public boolean checkState(){
    if ( back == -1 && front == 0 ){
        return true;
    }
    return ( front + currentSize -1 ) % theArray.length == back;
}
```

Riassumendo

- Approccio (semi) randomico di randoop **utile** per
  - trovare difetti che portano ad eccezioni non gestite
  - generare test di non-regressione

Attenzione

- Di solito il test di non-regressione prevede l'utilizzo di tecniche specifiche per identificare un **sottoinsieme** di test da riutilizzare per soddisfare i requisiti di costo
  - questo perchè di solito i test di non-regressione impongono dei costi
  - randoop genera test per le classi
    - esecuzione a costozero

Search Based Software Testing: EvoSuite

Idea di Base

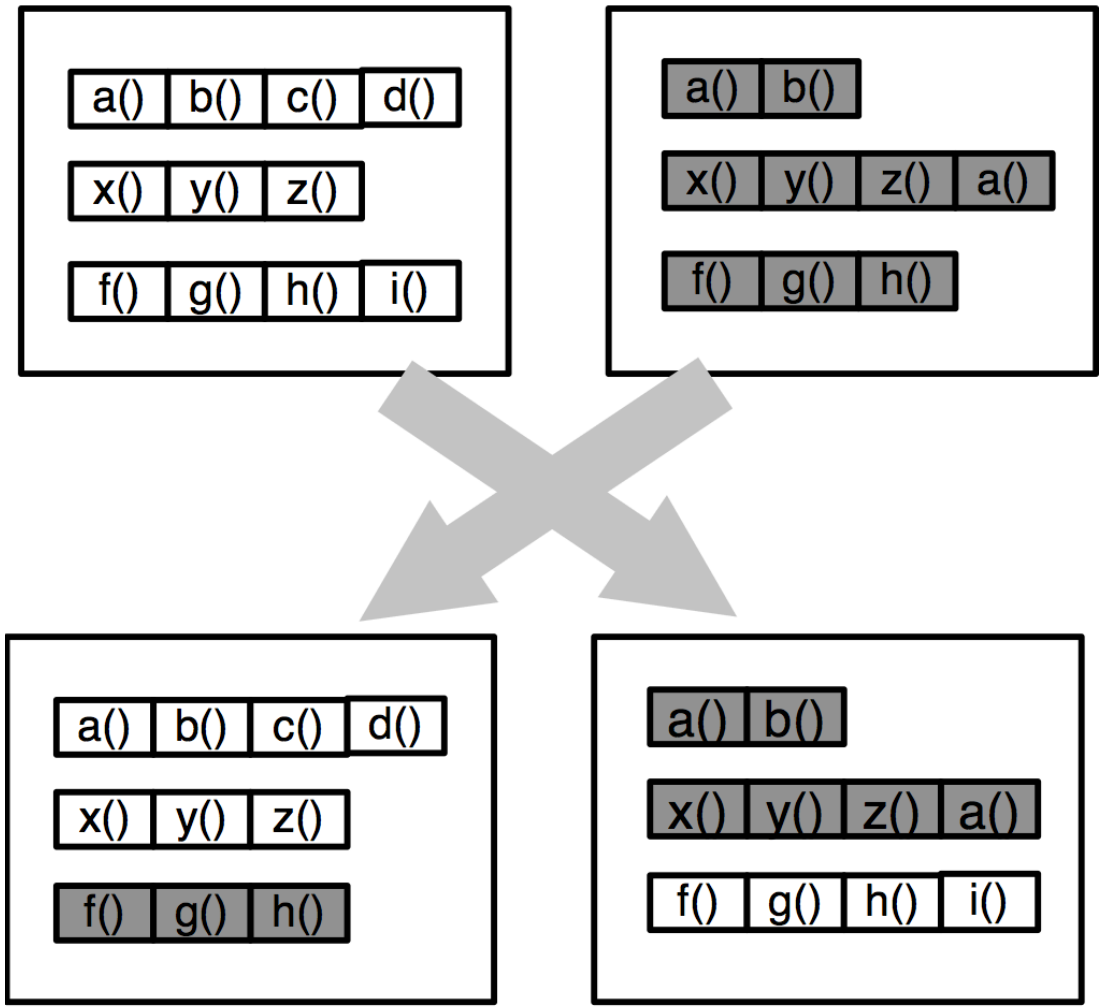
- Casi di Test Generati in Maniera (semi) Randomica
  - numerosi
- Guidiamo la Generazione dei Casi di Test
  - Numero di casi di test fisso
  - Popolazione iniziale di test randomica
  - Applichiamo un algoritmo genetico per far evolvere la popolazione
  - Teniamo le test

EvoSuite Algorithm 1/4

current population ← generate random population
repeat
    Z ← elite of current population
    while |Z| ≠ |current population| **do**
        P1,P2 ← select two parents with rank selection
        if crossover probability **then**
            O1,O2 ← crossover P1,P2
        else
            O1,O2 ← P1,P2
        mutate O1 and O2

        . . .

Crossover

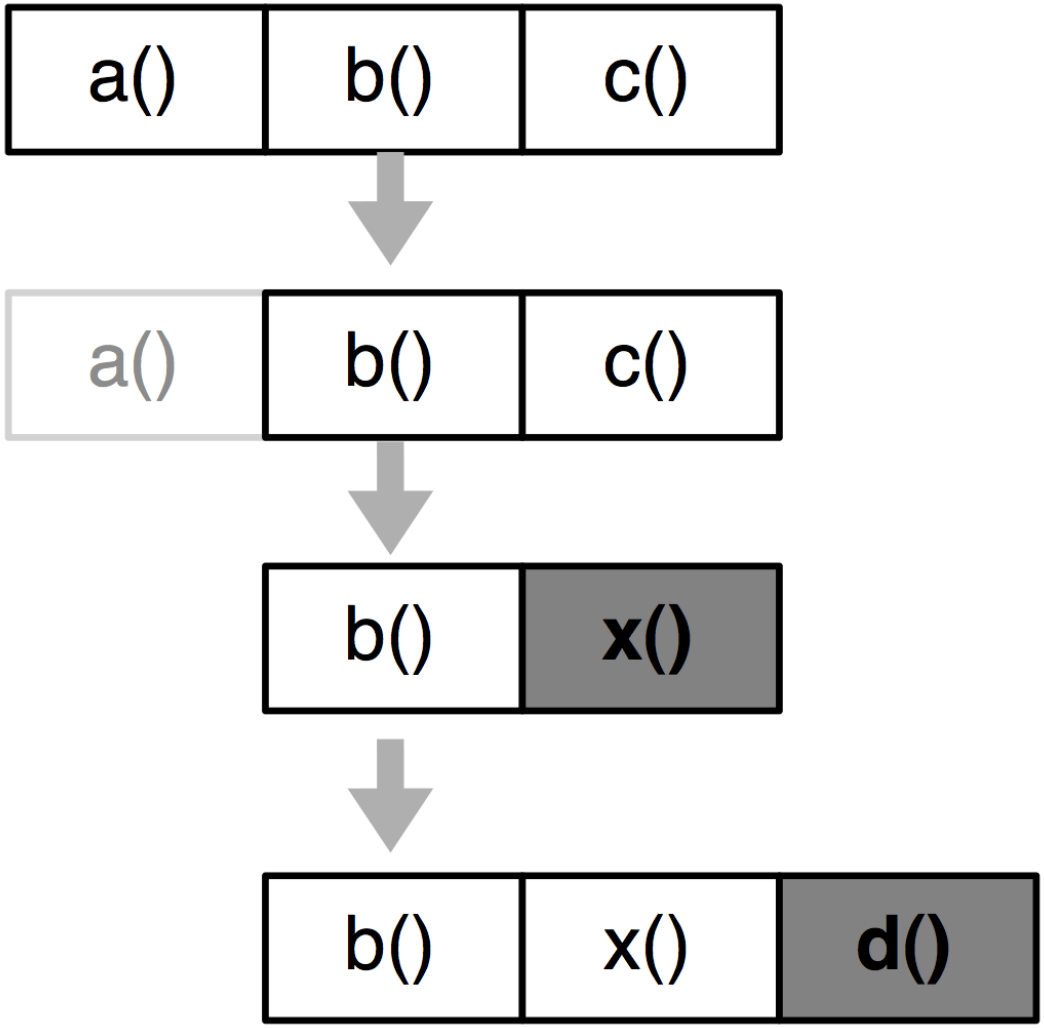


EvoSuite Algorithm 1/4

current population ← generate random population
repeat
    Z ← elite of current population
    while |Z| ≠ |current population| **do**
        P1,P2 ← select two parents with rank selection
        if crossover probability **then**
            O1,O2 ← crossover P1,P2
        else
            O1,O2 ← P1,P2
        mutate O1 and O2

        . . .

Mutation



EvoSuite Algorithm 2/4

current population ← generate random population

repeat

Z ← elite of current population

while |Z| ≠ |current population| **do**

. . .

fP = min(fitness(P1),fitness(P2))

fO = min(fitness(O1),fitness(O2))

LP =length(P1)+length(P2)

LO = length(O1) + length(O2)

TB = best individual of current population

. . .

EvoSuite Algorithm 3/4

. . .

fP = min(fitness(P1),fitness(P2))
fO = min(fitness(O1),fitness(O2))
LP =length(P1)+length(P2)
LO = length(O1) + length(O2)
TB = best individual of current population

if fO <fP $\vee$ (fO =fP $\wedge$ LO $\leq$ LP) **then**

  for O in {O1,O2} **do**

    if length(O) $\leq 2 \times$ length(TB) **then**

      $Z \leftarrow Z \cup \{O\}$

    else

      $Z \leftarrow Z \cup \{P1 \text{ or } P2\}$

  end for

else

  $Z \leftarrow Z \cup \{P1,P2\}$

EvoSuite Algorithm 4/4

current population ← generate random population

repeat

    Z ← elite of current population

    while |Z| ≠ |current population| **do**

    . . .

    end while

    current population ← Z

until solution found or maximum resources spent

Applichiamo Evosuite a TransformersMap

- Ripristiniamo la versione con il difetto
  - bash restore.sh
- Eseguiamo EvoSuite
  - evosuite -projectCP bin/ -class org.apache.commons.math.util.TransformerMap
- Correggiamo l'implementazione
- Estendiamo la test suite esistente

Generazione Automatica con Evosuite

- Esempio: TransformersMap
  - controlliamo le asserzioni
    - correggiamo gli eventuali difetti
  - misuriamo copertura
  - estendiamo la TS esistente
- Esercizio: Runner
  - identificare difetto: numero di run non decrementato
  - aggiungere manualmente casi di test per incrementare la copertura

Utilizzo di Dynamic Symbolic Execution per Aumentare la Copertura

```
Runner r0= new Runner();
Runner r1= new Runner();
r1.addRun(1372L, 1372L);
r0.addRun(1372L, 1372L);
r1.addRun(1372L, 1372L);
r0.addRun(1372L, 1372L);
r1.distanceInBetween(r0, r1);
```

```
public boolean distanceInBetween( Runner d1, Runr
    if ( runs < MIN_RUNS_N || d1.runs < MIN_RUNS_
        return false;
    }
    if ( d1.getOverallDistance() >= overallDistar
        return false;
    }
    if ( overallDistance <= d2.getOverallDistance
        return true;
    }
    return false;
}
```

Utilizzo di Dynamic Symbolic Execution per Aumentare la Copertura

```
Runner r0= new Runner();
Runner r1= new Runner();
r1.addRun(1372L, 1372L);
r0.addRun(1372L, 1372L);
r1.addRun(1372L, 1372L);
r0.addRun(1372L, 1372L);
r1.distanceInBetween(r0, r1);
```

```
public boolean distanceInBetween( Runner d1, Runr
    if ( runs < MIN_RUNS_N || d1.runs < MIN_RUNS_
        return false;
    }
    if ( d1.getOverallDistance() >= overallDistar
        return false;
    }
    if ( overallDistance <= d2.getOverallDistance
        return true;
    }
    return false;
}
```

Utilizzo di Dynamic Symbolic Execution per Aumentare la Copertura

```
Runner r0= new Runner();
Runner r1= new Runner();
r1.addRun(1372L, 1372L);
r0.addRun(1372L, 1372L);
r1.addRun(1372L, 1372L);
r0.addRun(1372L, 1372L);
r1.distanceInBetween(r0, r1);
```

```
public boolean distanceInBetween( Runner d1, Runr
    if ( runs < MIN_RUNS_N || d1.runs < MIN_RUNS_
        return false;
    }
    if ( d1.getOverallDistance() >= overallDistar
        return false;
    }
    if ( overallDistance <= d2.getOverallDistance
        return true;
    }
    return false;
}
```

```
Runner s_r0 = new Runner();
Runner s_r1 = new Runner();
s_r1.addRun(s1, s2);
s_r0.addRun(s3, s4);
s_r1.addRun(s5, s6);

s_r0.addRun(s7, s8);

s  r1.distanceInBetween(s  r0, s  r1);
```

```
s_r0.runs=0
s_r1.runs=0
! (s1<=0 ) && !(s2<=0) && !(s_r1.runs==5) && s_r1.oD=s1
! (s3<=0 ) && !(s4<=0) && !(s_r0.runs==5) && s_r0.oD=s3
! (s5<=0 ) && !(s6<=0) && !(s_r0.runs==5)
      && s_r1.oD.1=s_r1.oD+s5
! (s7<=0 ) && !(s8<=0) && !(s_r0.runs==5)
      && s_r1.oD.1=s_r1.oD+s5
s_r1.oD.1 >= s_r0.oD.1
```

```
evosuite -projectCP bin/ -class Runner
    -Dlocal_search_dse=TEST -Dlocal_search_rate=90
    -Dassertion_strategy=ALL
```