Augmented Dynamic Symbolic Execution

ASE 2012

1Saarland University
2University of Sheffield
3Microsoft Research
public void MethodUnderTestTest1(int x) {
    if (10 <= x && x <= 20)
        ComputeInRange();
    else
        ComputeOutOfRange();
}

Test inputs generated by DSE: x = {0, 10}

By ADSE: x = {0, 9, 10, 11, 19, 20, 21}
To *kill* a mutant, we need a test case that passes on the original program but fails on the mutant.
public void MethodUnderTestTest1(int x)
{
    if (10 <= x && x <= 20)
    {
        Cor< -x
        Cor< -x
    }
    else
    {
        Cor> x+1
        Cor> x+1
    }
    Cor>= x-1:OutOfRange();
    Cor>= x-1:OutOfRange();
    Cor>= x-1:OutOfRange();
    Cor>= x-1:OutOfRange();
    Cor== 1
    Cor== 1
    Cor== 1
    Cor== 1
    Cor!= 0
    Cor!= 0
    Cor!= 0
    Cor!= 0
}

ADSE instances: boundary cases, mutation, logical coverage, exceptions, ...
Tools of the trade

Technology platform: .NET 4.0 / C#

DSE engine: Pex (Program Exploration)

Constraint solver: Z3
Evaluation & minimization

Augmented test suite

Augmented PCs

Path conditions

Pex + extension

Parameterized unit tests

Pex Wizard

DLL (byte-code)

Apex

Apex
Evaluation subjects

Factorial, Power, MaxValue, Fibonacci, GCD

WBS (Wheel brake system)

FindMiddle, WrapRoundCounter

Roops integer examples
Mutation Score

<table>
<thead>
<tr>
<th></th>
<th>DSE</th>
<th>ADSE/B</th>
<th>ADSE/M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>48.44%</td>
<td>80.66%</td>
<td>86.16%</td>
</tr>
</tbody>
</table>
Boundary ADSE and mutation ADSE significantly increase mutation score as compared to plain DSE, thus significantly increasing resulting test suite defect detection ability.

ADSE can create tests exercising code under test with inputs expected by developers, e.g. covering boundary cases.