Proofreading the Proofreader: The Benefits of Unit Tests for Software Models

Allison Sullivan
The University of Texas at Arlington
Software Modeling

Problem:

ONE DOES NOT SIMPLY

WRITE FLAWLESS CODE
**Problem:**

ONE DOES NOT SIMPLY WRITE FLAWLESS CODE

**Solution:**

```plaintext
one sig List { header: lone Node }
sig Node { link: lone Node }
pred acyclic(){
    no List.header or
    some n : List.header.*link | no n.link
}
run acyclic for 3
```
A Software Model

An Alloy Model:

```alloy
one sig List { header: lone Node }

sig Node { link: lone Node }

pred acyclic()

  { no List.header or some n : List.header.*link | no n.link }

run acyclic for 3
```

*Show me all acyclic lists with up to 3 nodes*

Discovered Scenario:
New Problem:
ONE DOES NOT SIMPLY
WRITE FLAWLESS MODELS
New Solution: Unit Tests for Models

An Alloy Model:

```alloy
one sig List { header: lone Node }
sig Node { link: lone Node }
pred acyclic(){
  no List.header or some n : List.header.*link | no n.link
}
run acyclic for 3
```

**Discovered Scenario: Unit Test**

- List = L0
- Node = N0
- header = L0->N0
- acyclic

Sullivan: Proofreading the Proofreader
New Solution: Unit Tests for Models

An Alloy Model:

```alloy
class List {
    header: lone Node
}
class Node {
    link: lone Node
}

predicate acyclic()
    no List.header or some n : List.header.*link | no n.link

run acyclic for 3
```

Discovering Scenario: Unit Test

- List = L₀
- Node = N₀
- header = L₀→N₀
- link = N₀→N₀
- acyclic
Ex: Mutation Testing

public int min(int x, int y) {
    int v;
    if(x < y)
        v = x;
    else
        v = y;
    return v;
}

Original

Mutant 1.

public int min(int x, int y) {
    int v;
    if(x >= y)
        v = x;
    else
        v = y;
    return v;
}

Mutant 2.

Sullivan: Proofreading the Proofreader
Ex: Mutation Testing

pred acyclic(){
    no List.header or some n : List.header.*link | no n.link
}
pred acyclicMUTATED(){
    no List.header or some n : List.header.*link | some n.link
}
check {acyclic[] <=> acyclicMUTATED} for 3

Ask: Do these two properties differ? (detect equivalent mutant)
If yes, show me a scenario where they differ. (test to kill mutant)

Answer:
What you can walk away with:

- Software models can be intimidating, but enable a really robust automated testing environment
- If working with a non-traditional language, consider investing in unit testing

Thank you! Any questions?

Allison Sullivan
The University of Texas at Arlington
allison.sullivan@uta.edu