Manutenção e Evolução de Software: Projeto Prático

João Saraiva & José Nuno Macedo

Ano Lectivo 2023/2024

Relembre o projeto de software PicoC que tem vindo a desenvolver ao longo do semestre na Unidade Curricular de Manutenção e Evolução de Software. Nesta fase final do projeto, vai-se implementar um sistema de Unit Testing e Fault Localization sobre esta linguagem.

Data de Entrega/Apresentação: 27 de Maio

1 Tarefas a Desenvolver

Deverá desenvolver cada uma das seguintes tarefas. Guarde todos os resultados que considerar relevantes à avaliação em constantes com nomes relevantes. Por exemplo, quando é pedido a escolha de 3 programas PicoC e eventualmente test suites à volta deles, dever-se-à definir:

```
programa1 = ...
programa2 = ...
programa3 = ...
(...)
testSuitePrograma1 = [(inputs1, 1), (inputs2, 17), (inputs3, -348)]
runTestSuitePrograma1 = runTestSuite programa1 testSuitePrograma1
```

1. Defina uma função `evaluate :: PicoC -> Inputs -> Int` que, dado um programa PicoC e os seus inputs, i.e. os valores atribuídos a cada variável, o execute e produza um resultado.

2. Defina uma função `runTest :: PicoC -> (Inputs, Int) -> Bool` que para um programa PicoC, dado os seus inputs e o seu resultado esperado, irá verificar se o programa com esses inputs de facto produz esse resultado.

3. Defina uma função `runTestSuite :: PicoC -> [(Inputs, Int)] -> Bool` que irá correr vários testes unitários e validar se todos passam.

4. Selecione 3 programas PicoC que considere representativos. Produza testes unitários para cada um, e verifique que estes passam com a sua função `runTestSuite`.

5. Desenvolva código para a inserção de uma mutação **aleatória** num programa PicoC. Para isto, poderá recorrer a uma implementação manual, ou investigar as definições de `once_randomTP` e `mutations` disponíveis na biblioteca de programação estratégica em Haskell.

6. Utilize o código da alínea anterior para inserir uma mutação em cada programa PicoC escolhido anteriormente (ou insira manualmente se não o implementou). Utilize a função `runTestSuite` para correr os testes unitários definidos anteriormente, mas agora a usar o programa mutado (mantendo ainda o resultado esperado do programa original). Certifique-se que cada test suite irá agora falhar por causa da mutação feita.

7. Estenda a sua linguagem PicoC para incluir uma instrução `print`, que poderá imprimir uma string para o terminal. Adapte o seu `evaluate` para que este seja capaz de imprimir texto sempre que encontra uma instrução `print`. (Dica: utilize o módulo **Debug.Trace** para não necessitar de manipular o monad de IO para escrever texto.)

8. Defina uma função `instrumentation :: PicoC -> PicoC` que irá pegar num programa PicoC e instrumentar o programa para auxiliar a localização de falhas.

9. Defina uma função `instrumentedTestSuite :: PicoC -> [(Inputs, Int)] -> Bool` que irá instrumentar cada programa antes de o executar, e de seguida irá correr os testes unitários e validar se todos passam. Em alternativa, implemente uma versão da função `evaluate` que durante a execução do programa e inputs dados como argumento, produz no fim a lista de *instruções* do programa usadas durante a sua execução.

10. Recolha o resultado da execução de `instrumentedTestSuite` nos seus testes unitários que usam o programa mutado e o resultado esperado não-mutado. Coloque estes resultados numa tabela / folha de cálculo e utilize o algoritmo de *Spectrum-Based Fault Localization* para localizar as instruções com mais probabilidade erro em cada um dos 3 programas.