Trabalho 2

Métodos Formais

18 de novembro de 2024

Contents

1 Informações importantes **1**

2 Ajudando o banco **1**
2.1 Artefatos 1
2.2 Procedimento para execução 2
2.3 Relatório 2

1 Informações importantes

- Data de entrega: 02 de dezembro (segunda-feira)
- Trabalho individual ou em dupla
- Formato de entrega: Código + relatório
- Formato de apresentação: Breve explicação + demonstração (somente para a professora)

2 Ajudando o banco

Você recebeu um software em C++ que implementa um banco, e sua missão é encontrar e consertar *bugs* desse código. Você já escreveu uma especificação em Quint com as funcionalidades desse software, verificando propriedades relevantes e ajustanto a especificação até que todas elas fossem satisfeitas (assim como fizemos no trabalho 1). Com isso, você consegue afirmar que o modelo está correto.

Contudo, a sua missão é sobre o código, não sobre o modelo. Então, agora você precisa utilizar o modelo para aumentar a confiança no software em si. Não é possível, com testes, garantir que o modelo e o software correspondem 100%. Mas, quanto mais testes forem rodados, menor a chance de termos diferenças de comportamento.

Conecte o modelo existente ao código existente com testes baseados em modelos. Para cada teste que falhar, ou seja, para cada execução onde o comportamento do modelo for diferente do comportamento do código, ajuste o código para que o teste passe a suceder.

2.1 Artefatos

Você recebe:

1. Um modelo em Quint completo e correto para o sistema do banco (`bank.qnt`)
   - Não precisa ser modificado
   - Se modificá-lo, pode ser necessário ajustar o esqueleto dos testes (3)
2. Uma implementação em C++ do banco com *bugs* (`bank.hpp`)
   - Deve ser modificado
   - Todas as modificações devem ser motivadas por testes falhando, e devidamente documentadas no relatório (i.e. “Adicionei um if na função de depósito devido a esse teste falhando”)

3. Um esqueleto para a implementação dos testes em C++ (`test.cpp`)
    - Deve ser modificado
    - Compilar passando a pasta `lib`: `g++ -I lib test.cpp`

2.2 Procedimento para execução

Atenção: Atualizem a versão do Quint! A mais recente é v0.22.4 Para atualizar: `npm install -g @informalsystems/quint@latest`

1. Obter 10 mil execuções a partir do simulador: `quint run bank.qnt --mbt --out-itf=traces/out.itf.json --n-traces=10000`
    - PS: crie a pasta `traces` para que todos os arquivos JSON sejam escritos nela.
2. Executar o teste (que lerá os JSONs produzidos no passo (1))

Os testes deverão falhar (até que todos os *bugs* tenham sido consertados). Se o teste suceder em alguma iteração, repita o processo algumas vezes para adiquirir mais confiança.

Quando um teste falhar:

1. Observe o que aconteceu. Imprima mais informações se necessário. Entenda o motivo da diferença entre modelo vs código.
2. Ajuste a implementação (Artefato 2) para consertar o *bug* que você encontrou
3. Re-compile e execute novamente (com a mesma execução!). Se seu ajuste foi feito corretamente, o teste deve suceder (ou falhar por outro motivo mais adiante na execução).
4. Documente o que você observou e que ajuste fez no relatório.

Quando um teste suceder:

- Volte ao início, até ter confiança de que o comportamento do modelo e do código são os mesmos. Como já estamos testando 10 mil traces nesse procedimento, é bem provável que uma iteração seja suficiente, mas repita algumas vezes para aumentar a confiança.

2.3 Relatório

Atenção: Deve ser entregue em PDF!

O relatório deve justificar cada alteração feita na implementação do banco (arquivo `bank.hpp`). Cada alteração deve ser mínima: quanto menor, melhor. Isso demonstra que você está utilizando o método corretamente, e que conseguirá aplicá-lo mesmo em uma implementação muito complexa.

Seu relatório deve conter uma seção explicando brevemente como seu teste está funcionando: quais dados estão sendo impressos e quais comparações estão sendo feitas; e uma seção com as modificações que você fez.

Para cada modificação, você deve apresentar:

1. Qual informação do teste levou você a identificar o *bug*. Pode ser uma captura de tela do terminal ou cópia do output, com uma breve descrição.
2. A modificação que você fez. Pode ser uma captura de tela ou cópia do código inserido. Necessário que fique claro o que foi modificado (não coloque uma imagem do código inteiro). Alternativamente, podem informar a *hash* do *commit* do GitHub, de forma que o commit contenha apenas a modificação em questão.

A princípio, seu relatório pode conter apenas a descrição do teste e das modificações, sem necessidade de mais seções como introdução e conclusão. Se houverem mais informações relevantes, podem acrescentar.