Trabalho 2

Métodos Formais

10 de junho de 2024

Contents

1 Informações importantes

- Data de entrega: 01 de julho (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
   - 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*)
   - 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++)
    - Deve ser modificado
    - Compilar passando a pasta `lib`: `g++ -I lib test.cpp`

2.2 Procedimento para execução

1. Obter uma execução a partir do simulador: `quint run bank.qnt --max-samples=1 --mbt --out-itf=out.itf`
2. Executar o teste (que lerá o JSON produzido 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.