Model Testing – Combining Model Checking and Coverage Testing: Resenha

Gabriela Moreira Mafra[1]

[1]Departamento de Ciência da Computação
Centro de Ciências Tecnológicas
Universidade do Estado de Santa Catarina (UDESC)
Joinville, SC – Brasil

{gabrielamoreiramafra@gmail.com}

1. Tema de pesquisa

A proposta do meu trabalho de dissertação é gerar elementos de um ambiente de desenvolvimento a partir de especificações formais modeladas em TLA$^+$. Um desses elementos, conforme proposto, seria uma suíte de testes automatizados que podem aumentar a confiabilidade do programa executável - também gerado - mesmo que modificado posteriormente.

Uma das abordagens para geração de testes é a baseada em modelo, que pode usar recursos já implementados par TLA$^+$ como o seu model checker, o TLC. Assim, utilizando a estrutura montada pelo TLC para a especificação escrita em TLA$^+$, é possível extrair as informações necessárias para gerar os casos de teste que garantem as propriedades.

2. Proposta da dissertação

O trabalho discutido propõe a aplicação de conceitos de testes orientados a especificação em *model checking*. Em suma, a ideia do autor é, a partir de uma descrição do sistema e outra descrição do seu comportamento, construir especificações formais e gerar propriedades a serem checadas por um *model checker*, que é capaz de apontar se tais propriedades são atendidas pelo sistema e, caso não sejam, indicar qual sequência de ações leva ao erro.

Entende-se que a maior contribuição nessa proposta é a geração de casos de testes a partir de propriedades de comportamento, que traduz algo mais conhecido para o usuário - comportamento - em propriedades mais próximas ao sistema. A transformação das descrições do sistema e do comportamento são traduzidas de forma manual para especificações formais, de onde é possível gerar estruturas intermediárias para serem analisadas automaticamente, produzindo testes. O autor utiliza estruturas de Kripke para essa representação, o que tange bem a linguagem de especificação TLA$^+$, presente na pesquisa a ser comparada, por ser também parte da lógica temporal.

As diversas traduções propostas pelo autor parecem um tanto excessivas, considerando que linguagens de especificação formal mais robustas, como TLA$^+$ e redes de Petri, permitem o uso direto de *model checkers* com uma linguagem de alto nível. Se o trabalho utilizasse uma linguagem como essas, supõe-se que seriam necessárias menos representações intermediárias e traduções.

Um exemplo claro desse excesso é a proposta de geração de propriedades de vivacidade. O autor propõe um algoritmo para descrever todas as propriedades a serem verificadas para que se garanta vivacidade do sistema; porém muitas ferramentas de *model checking* aceitam uma única propriedade que é suficiente para essa garantia.

3. Estrutura

A dissertação mantém a ordem dos elementos apresentados no padrão, iniciando com introdução e trabalhos relacionados. A introdução expõe a importância do trabalho, mas fala de forma bastante superficial sobre o que está sendo feito, dedicando apenas um parágrafo a isso. Os trabalhos relacionados são apresentados de forma breve e organizada.

Na sequência, o autor introduz os conceitos necessários para o entendimento do trabalho, sendo um capítulo para conceitos preeliminares - como modelagem, testes baseados em especificação e *model checking* - e conceitos necessários para entendê-los; e um segundo capítulo para conceitos introduzidos pelo trabalho, que são explicados a partir de um exemplo trivial e clássico do semáforo de trânsito.

O último capítulo apresenta um estudo de caso onde a proposta de teste é aplicada em um programa de gerenciamento de música pessoal. Esse programa é de licensa e código fechado, sendo assim o autor observa as mudanças na tela a partir de cada entrada aplicada para escrever a especificação do sistema; e usa seu manual para escrever a especificação do comportamento esperado. Ambas especificações são traduzidas para as devidas estruturas e levadas ao *model checker* NuSMV, que aponta um conjunto de erros que são então verificados na aplicação real.

A conclusão do trabalho reforça algumas justificativas de escolhas, como a do uso de uma linguagem próxima ao usuário para a descrição do sistema, assim como das vantagens do uso de *model checkers* em comparação com testes tradicionais. São indicadas também algumas melhorias na ferramenta que poderiam ser feitas em trabalhos futuros.

4. Contribuições para o trabalho

O trabalho analisado tem caráter inverso à minha proposta de dissertação, no sentido de que o autor parte de um sistema existente para gerar formalismo e garantias, e o meu trabalho parte de formalismos para gerar um sistema existente com as garantias. Mesmo assim, o trabalho trouxe muitas contribuições, principalemte em quesitos de nomeclatura. O autor propõe termos como cobertura de nós e arestas para se referir a propriedades gerais como alcançabilidade e vivacidade, separando essas de outras propriedades específicas, que chama de “casos de checagem”.

A própria estrutura do trabalho reforça o que eu tinha de planejamento para a apresentação, onde o autor primeiro usa um exemplo simples, do semáforo de trânsito, para explicar os conceitos propostos; e ao final traz um estudo de caso real, para demonstrar a aplicabilidade da ferramenta desenvolvida.

Por fim, a ideia que o autor chama de comportamento caixa preta, onde as propriedades especificadas são somente aquelas que tangem o usuário, me pareceu interessante como mais uma forma em que o trabalho proposto pode ser útil. Esse caso de uso é visto como uma possibilidade de exemplo para mostrar a abrangência de propriedades - e portanto, testes - permitida.

References

GÜLDALI, B. Model testing combining model checking and coverage testing. *Master's thesis, University of Paderborn, Germany*, 2005.