Plano de Trabalho de Conclusão de Curso
Tradução automática de especificação formal modelada em TLA+ para linguagem de programação

Gabriela Moreira Mafra – gabrielamoreiramafra@gmail.com
Cristiano Damiani Vasconcellos – cristiano.vasconcellos@udesc.br *(orientador)*
Karina Girardi Roggia – karina.roggia@udesc.br *(coorientadora)*

Turma 2019/1 – Joinville/SC

15 de Março de 2019

Abstract

Escrever programas em forma de especificação, como em TLA+, permite um alto nível de abstração e possibilita validações que podem revelar problemas difíceis de perceber em uma linguagem de programação. A partir da especificação, o programador escreve a implementação da ideia modelada, para que de fato seja aplicada. A tradução automática de especificações para programas tem como objetivo mitigar a incoerência entre as duas definições. Para isso, é necessário encontrar um mapeamento dos construtores de TLA+ para estruturas da linguagem de programação. As transições modeladas serão traduzidas para funções da linguagem, e disparadas de forma concorrente conforme a especificação do estado inicial e suas transições. Assim, pretende-se atingir - e, idealmente, garantir - a correspondência do das funções e execuções geradas para a especificação inicial em TLA+.

Palavras-chave: *especificação de software, lógica temporal, geração de código, métodos formais, model checking*

1 Introdução e Justificativa

A linguagem de especificação formal TLA+ (*Temporal Logic of Actions*) foi desenvolvida por Leslie Lamport com o objetivo de escrever provas formais para sistemas concorrentes da maneira mais simples possível [Lamport 2008]. Para isso, o *model checker* TLC foi desenvolvido, e mais recentemente o sistema de provas TLAPS (*TLA Proof System*) que permite checar mecanicamente algumas provas e ainda está incompleto.amazon

Escrever uma especificação nessa linguagem possibilita encontrar potenciais problemas, como a *Amazon Web Services* reportou [Newcombe et al. 2015]. As especificações escritas, contudo, não possuem nenhum vínculo com a implementação se não pelo entendimento do programador que as escreveu. Outras linguagens de especificação formal com objetivos semelhantes ao TLA+, como Z, B-Method e ASM, fornecem formas de gerar código a partir do modelo. Ainda não existem geradores de código a partir de modelos escritos em TLA+, impossibilitando a conversão das especificações em linguagens de programação com garantia de correspondência.

Com o programa especificado, validado e traduzido para linguagem de programação, é possível aplicá-lo diretamente em casos reais com a garantia de correspondência e, portanto, das propriedades verificadas; ou então melhorar a implementação para uma versão mais otimizada, mas que parte da mesma base - nesse caso, a garantia é reduzida, já que as mudanças não estavam representadas no modelo original. Observações sobre os benefícios da geração de código a partir de modelos de especificação formal já foram verificadas em trabalhos como o estudo de caso em [Leonard e Heitmeyer 2008].

2 Objetivos

O propósito é estabelecer um mapeamento entre especificações em TLA+ e uma linguagem de programação, e com isso fazer traduções automáticas dos modelos, buscando maximizar a garantia de correspondência do programa gerado com o especificado, além de viabilizar otimizações e outras alterações.

1. Encontrar correspondências para cada um dos construtores da linguagem TLA+ para estruturas de uma linguagem funcional.
2. Fornecer garantias de que os mapeamentos feitos são tais que garantem as propriedades verificadas
3. Disponibilizar mecanismos para mitigar a perda de garantias diante de alterações no código gerado - como testes unitários.

3 Metodologia

As etapas para o desenvolvimento se dividem em amplos estudos com revisões literárias sobre os recursos de TLA+, e fases com teor mais prático aonde o tradutor será implementado.

1. Etapa 1 - Estudo de TLA+: Inclui leitura de textos e tutoriais, busca de exemplos e outras fontes de conhecimento sobre a linguagem, como o curso em vídeo ensinado pelo criador da linguagem.
2. Etapa 2 - Traduções manuais: Se dá por tentativas de codificação de exemplos de modelos obtidos na Etapa 1. Essa codificação será em uma linguagem de programação funcional com concorrência, maximizando a proximidade com o modelo - possivelmente Elixir.
3. Etapa 3 - Estabelecimento de mapeamentos: Uma observação das traduções manuais ao lado do código, com objetivo de enumerar mapeamentos feitos no processo manual. Dessa etapa, se espera uma lista de possíveis correspondências a serem avaliadas na Etapa 4 e 5.
4. Etapa 4 - Escolha de mapeamentos: Dentre os possíveis mapeamentos, serão escolhidos aqueles que apresentam maior correspondência entre as definições, conforme estudos sobre os significados dos construtores utilizados nas duas linguagens.

5. Etapa 5 - Encontrar garantias para os mapeamentos: O processo de conjecturar, evidenciar ou provar a correspondência entre uma especificação formal qualquer e o código gerado a partir dela com os mapeamentos escolhidos na Etapa 4.

6. Etapa 6 - Implementação do gerador de código: Consiste em implementar um programa capaz de fazer o *parsing* da linguagem TLA+ e escrever um arquivo com o código na linguagem de programação.

7. Etapa 7 - Análise de melhorias do ambiente: É o estudo da capacidade de testes para o código gerado manterem a correspondência perante modificações no mesmo. Será feito atráves da implementação de testes e exploração de quais mudanças teriam potencial de tornar o código incoerente com a especificação e passariam nos testes.

4 Cronograma proposto

Etapas	2018											
	J	F	M	A	M	J	J	A	S	O	N	D
1												
2												
3												
4												
5												
6												
7												

5 Linha de Pesquisa

A base teórica em especificação formal necessária para esse trabalho é um dos fundamentos estudados no Grupo de Pesquisa em Fundamentos da Computação, ao qual o trabalho pertence. O tema é pertinenta à linha de pesquisa em lógica, especialmente lógica temporal - base para o TLA+.

6 Forma de Acompanhamento/Orientação

As reuniões com orientador e aluno serão feitas quinzenalmente, com a possibilidade de agendar encontros extras quando necessário. Para acompanhamento do progresso, serão entregues aos orientadores artefatos com os resultados obtidos em cada etapa (quando aplicável), para que assim exista uma avaliação contínua com mais ciclos de retorno.

References

[Lamport 2008]LAMPORT, L. The specification language tla+. In: HENSON, D. B. e M. C. (Ed.). *Logics of specification languages*. Berlin: Springer, 2008. p. 616–620. ISBN 3540741062. Disponível em: <http://lamport.azurewebsites.net/pubs/commentary-web.pdf>.

[Leonard e Heitmeyer 2008]LEONARD, E. I.; HEITMEYER, C. L. Automatic program generation from formal specifications using apts. In: DANVY, O. et al. (Ed.). *Automatic Program Development: A Tribute to Robert Paige*. Dordrecht: Springer Netherlands, 2008. p. 93–113. ISBN 9781402065859. Disponível em: <https://doi.org/10.1007/978-1-4020-6585-9$_1$0>.

[Newcombe et al. 2015]NEWCOMBE, C. et al. How amazon web services uses formal methods. *Commun. ACM*, ACM, New York, NY, USA, v. 58, n. 4, p. 66–73, mar. 2015. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/2699417>.

Cristiano Damiani Vasconcellos

Gabriela Moreira Mafra

Karina Girardi Roggia

(Líder do Grupo de Pesquisa em Fundamentos da Computação)