XIV EDIÇÃO JIC JORNADA DE INICIAÇÃO CIENTÍFICA E TECNOLÓGICA 14 a 16 de outubro EVENTO ON-LINE UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

RANDOM ELIXIR CODE GENERATION APPLIED TO COMPILER TESTING

BERNARDO BELTRAME FACCHI [1,2], SAMUEL DA SILVA FEITOSA [3]

1 Introdução

Elixir é uma linguagem de programação funcional, orientada a processos, escalável, concorrente e tolerante a falhas. Com o aumento da popularidade da linguagem e a sua adoção em muitos projetos importantes, é importante ter mecanismos para garantir que o compilador funcione corretamente. Garantir que o código da linguagem de máquina seja executado com precisão de acordo com o que foi escrito pelo programador é essencial para o desenvolvimento de software e para o uso da linguagem, uma vez que os programadores não desejam que falhas sejam introduzidas em seu software durante o processo de compilação.

Neste contexto, foi desenvolvida uma ferramenta de geração de código aleatório usando Haskell que gera código Elixir bem tipado aderindo a uma sintaxe especificada e regras de digitação, que serve como entrada para testes baseados em propriedades, buscando contribuir para o desenvolvimento geral. qualidade e confiabilidade de sistemas de software construídos com Elixir.

2 Objetivos

Desenvolver um gerador de código aleatório voltado para a linguagem de programação funcional Elixir, que seja capaz de exaustivamente testar o compilador da linguagem alvo a procura de falhas.

3 Metodologia

Foi desenvolvido um gerador de código aleatório capaz de gerar códigos Elixir válidos usando Haskell. Para garantir que o código gerado fosse válido, para a posterior realização de testes, foi levado em consideração certos formalismos, como a sintaxe e o sistema de tipos da linguagem alvo (VALIM, 2024).

A geração de código é definida, em termos simples, em três passos: (*i*) a geração de um tipo válido da linguagem Elixir, (*ii*) a geração de uma expressão, (*iii*) a conversão da

1 Acadêmico de Ciência da Computação, Universidade Federal da Fronteira Sul, *campus Chapecó,* contato: bernardobf@outlook.com.br
2 Grupo de Pesquisa: Inovação e Desenvolvimento Tecnológico - GIDT
3 Doutor, Universidade Federal da Fronteira Sul, **Orientador**.

expressão gerada para a sintaxe do Elixir.

Primeiramente, a geração de um tipo válido da linguagem é realizada de maneira aleatória, através da atribuição de pesos para cada tipo e sem nenhuma restrição. Posteriormente, é gerado uma expressão que usa como entrada o tipo gerado anteriormente, desta forma, é garantido que apenas expressões válidas serão geradas para um determinado tipo, respeitando as restrições impostas pelo sistema de tipos. Por fim, é feito a tradução da sintaxe abstrata usada no gerador para a sintaxe real do Elixir, resultando em um código Elixir.

A geração aleatória de código usa o método bottom-up (CARDOSO, 2022; FEITOSA, 2020; PALKA, 2011), em que a geração de uma expressão requer a geração de seus pré-requisitos. O objetivo da geração de uma expressão é criar uma expressão bem-tipada aleatoriamente que deve ser avaliada para um tipo específico. É importante mencionar que a geração de código é um processo recursivo no qual expressões podem gerar outras expressões, assim como tipos podem gerar outros tipos.

Para a realização de testes baseados em propriedade, foi utilizado a biblioteca QuickCheck, desenvolvida para simplificar e automatizar a execução de testes em geradores (HUGHES, 2016). Foram estabelecidas duas propriedades a serem testadas, uma de compilação e outra de comparação. A propriedade de compilação busca verificar se os códigos gerados pelo gerador são realmente códigos Elixir válidos, enquanto que a propriedade de comparação busca verificar se o mesmo código gerado resulta no mesmo resultado em diferentes versões do Elixir (foram comparadas as versões 1.15.0, 1.16.3, e 1.17.1).

Para cada propriedade, foram realizadas 10 rodadas de testes, cada um gerando 1000 códigos. Para cada programa gerado, o processo de teste envolve escrever o código em disco e subsequentemente invocar o compilador do Elixir para executar o programa e reportar o status de compilação (bem-sucedida ou não) assim como sua saída.

Além disso, empregamos a ferramenta Haskell Program Coverage (HPC) para avaliar a diversidade dos programas gerados e fornecer insights detalhados sobre os caminhos de execução seguidos por nossa lógica de geração de código. Dado que nossa abordagem para geração de código é aleatória, não temos controle sobre quais ramificações o algoritmo tomará durante a execução, é crucial garantir que essa aleatoriedade explore adequadamente todas as construções sintáticas e não perca inadvertidamente nenhum caminho crítico ou casos extremos.

4 Resultados e Discussão

Durante a execução dos testes foi observado que todos os programas gerados pelo gerador desenvolvido foram compilados e executados pelo Elixir com sucesso. Não foram encontrados nenhuma falha de compilação com as construções do Elixir utilizadas no projeto. Os testes de comparação não revelaram diferenças de saída entre as versões da linguagem testadas.

Ao analisar o relatório estatístico fornecido pelo HPC, observou-se que 100 por cento dos construtores sintáticos fossem cobertos em um lote de 1000 casos de teste. Demostrando que o método de geração de código utilizado explora efetivamente e exaustivamente os padrões de código possíveis dentro das restrições definidas. Uma cobertura completa significa que uma ampla variedade de programas pode ser gerada, garantindo que nenhum caso de teste seja negligenciado, implicando que a nossa abordagem de geração de programas pode ser confiável para testar todo o espectro da sintaxe válida do Elixir.

5 Conclusão

A linguagem de programação Elixir emergiu rapidamente como uma ferramenta poderosa no cenário do desenvolvimento de software moderno. No sentido de contribuir para a confiabilidade do compilador do Elixir, foi desenvolvido um gerador de código aleatório com base em uma formalização da sintaxe e do sistema de tipos do Elixir que gera programas Elixir bem tipados.

Embora os testes não tenham revelado nenhuma falha no compilador, o gerador mostrou-se eficaz na geração de programas. É possível, futuramente, expandir o algoritmo de geração para cobrir mais construtores e expressões sintáticas. Além disso, a mesma metodologia de testes pode ser aplicada com o aparecimento de outros compiladores Elixir.

Referências Bibliográficas

CARDOSO, ENTON; PEREIRA, DANIEL; DE PAULA, REGINA; REIS, LEONARDO; RIBEIRO, RODRIGO. A Type-Directed Algorithm to Generate Well-Formed Parsing Expression Grammars. **The XXVI Brazilian Symposium on Programming Languages**, p. 8, 2022

CASTAGNA, GIUSEPPE; DUBOC, GIULLAUME; VALIM, JOSÉ. The Design Principles of the Elixir Type System. **The Art, Science, and Engineering of Programming**, v. 8, 2023

HUGHES, JOHN. Experiences with QuickCheck: Testing the Hard Stuff and Staying Sane. **A List of Successes That Can Change the World**, v. 9600, p. 169, 2016

FEITOSA, SAMUEL; RIBEIRO, RODRIGO; DU BOIS, ANDRÉ. A type-directed algorithm to generate random well-typed Java 8 programs, **Science of Computer Programming**, v. 196, p. 192, 2020

PALKA, MICHAL; CLAESSEN, KOEN; RUSSO, ALEJANDRO; HUGHES, JOHN. Testing an Optimising Compiler by Generating Random Lambda Terms. **6th International Workshop on Automation of Software Test**, p. 91, 2011

Palavras-chave: Geração de código; Compilador Elixir; Testes baseados em propriedade.

Nº de Registro no sistema Prisma: PES-2023-0183

Financiamento: Projeto desenvolvido com bolsa de pesquisa financiada pela UFFS, Edital nº 73/GR/UFFS/2023; GRUPO 1 (Bolsas IC).