Tradutor Automático de Gramática de Grafos para Event-B

NÍCOLAS OREQUES DE ARAUJO[1]; SIMONE ANDRÉ DA COSTA CAVALHEIRO[2]; LUCIANA FOSS[3]

[1]Universidade Federal de Pelotas – nodaraujo@inf.ufpel.edu.br
[2]Universidade Federal de Pelotas– simone.costa@inf.ufpel.edu.br
[3]Universidade Federal de Pelotas – lfoss@inf.ufpel.edu.br

1. INTRODUÇÃO

Sistemas computacionais estão presentes nas mais diversas áreas, incluindo aquelas em que qualquer falha pode ser catastrófica, como sistemas bancários ou de controle aéreo (TEIXEIRA 2014). Assim, é de fundamental importância garantir que se comportem da forma correta e esperada.

Dessa forma, a utilização de diagramas na construção de especificações de sistemas, como forma de prevenir e garantir seu comportamento, é uma característica inerente ao desenvolvimento de software atual. Porém, devido a constante evolução dos sistemas, as linguagens utilizadas para modelagem devem permitir a definição de transformações de modelos que, por sua vez, devem possuir embasamento formal, a fim de garantir a preservação das características essenciais do sistema.

Gramática de Grafos é um modelo formal utilizado para descrever e analisar sistemas (EHRIGH 1997). A ideia básica consiste em modelar estados de um sistema como grafos e descrever suas possíveis mudanças de estado como regras de transformação de grafos. Uma gramática de grafos é composta por um grafo inicial, um grafo tipo e um conjunto de regras (RIBEIRO 2000). Este grafo inicial define o estado inicial do sistema e sofre as alterações provenientes das regras. Além disso, são oferecidas diferentes técnicas e/ou ferramentas para análise dessas transformações, como testes de terminação, confluência e independência. Portanto, a utilização de uma gramática de grafos como linguagem para especificação de sistemas é adequada para o paradigma de desenvolvimento de software atual, pois garante uma descrição visual e intuitiva, além de embasamento formal para suas transformações (EHRIG 1999).

Existem diferentes abordagens de verificação que permitem garantir que um sistema descrito em gramática de grafos possua certas propriedades, dentre eles a utilização de verificadores automáticos de modelos (BALDAN 2004; KASTENBERG 2006; KONIG 2010). Um exemplo são os verificadores automáticos, que realizam suas análises expandindo todos os estados possíveis de uma gramática. Porém, para gramáticas com alta gama de variações, esta expansão é uma operação custosa e, para gramáticas com número de estados infinitos, é impossível de ser realizada. Nestes casos, a automatização do processo de análise é impossibilitada, tornando o processo mais lento e sujeito a erros. Como consequência destes fatos, o tempo de desenvolvimento de sistemas ao utilizar-se da abordagem de gramática de grafos é consideravelmente estendido, levantando dúvidas sobre as vantagens oferecidas pelo uso do método pela comunidade de engenharia de software (BOWEN 1995; BOWEN 1997).

Por sua vez, Event-B é uma linguagem de modelagem e análise formal de sistemas baseado em um formalismo de estados. Esta linguagem destaca-se pelo uso da teoria de conjuntos para modelagem, utilização de conceitos de refinamento para construção de sistemas com diferentes níveis de abstração, além da utilização de provas matemáticas para verificação de consistência entre estes níveis (ABRIAL 2010). Um sistema em notação Event-B é composto por

diversos componentes de dois tipos: contextos e máquinas. Contextos representam partes estáticas do sistema, enquanto máquinas, por sua vez, representam a parte dinâmica do sistema modelado.

Em (CAVALHEIRO 2010; CAVALHEIRO 2017) foi proposta uma tradução de uma gramática de grafos para a notação matemática Event-B, de forma a possibilitar a utilização das mesmas em provadores de teoremas semi-automáticos, que, por sua vez, utilizam-se de propriedades oriundas da matemática de conjuntos da notação para realização das provas, tornando desnecessária a expansão de todos os estados da gramática para realização de provas. Este fato combate o problema de gramáticas com muitos estados ou com número de estados infinitos. Entretanto, o processo de tradução de uma gramática para notação Event-B continua sendo realizada de forma manual, o que mantém o processo lento em relação a outras técnicas e torna-o sujeito a falha humana.

O objetivo deste trabalho é o desenvolvimento de uma ferramenta em Java que permita realizar automaticamente a tradução de uma gramática de grafos, utilizando a abordagem relacional, para uma especificação na linguagem Event-B. Essa tradução facilitará o uso da técnica de prova de teoremas semi-automática para verificação de propriedades de sistemas utilizando a tradução definida em (CAVALHEIRO 2017).

2. METODOLOGIA

O primeiro passo para a realização deste trabalho foi a escolha de uma ferramenta apropriada para modelagem das gramáticas. A ferramenta escolhida foi o AGG (*The Attributed Graph Grammar System*), que permite a representação de grafos e sistemas de transformação de grafos de forma visual e intuitiva, utilizando como arquivo de saída o *GGX*, um formato derivado do *XML* (*Extensible Markup Language*) que é padrão para representação de gramáticas de grafos (TAENTZER, 2003).

Porém, a ausência de alguns recursos no AGG, como a definição de tipos de usuário, representação de dependência e a representação de elementos não estáveis de um grafo tipo, demonstram que outras ferramentas para modelagem podem ser usadas para outros sistemas específicos. Assim, visando o desenvolvimento de uma ferramenta que futuramente pudesse ser adaptada para outro programa de modelagem com facilidade, o projeto foi divido em três etapas: (1) desenvolvimento de um parser para arquivos *GGX*, convertendo-os para uma estrutura de dados representativa de uma gramática; (2) tradução desta estrutura de gramática de grafos para outra estrutura, representativa na notação Event-B; e (3) desenvolvimento de um gerador de arquivos para utilização no provador de teoremas semi-automático Rodin, a partir da estrutura em Event-B.

A divisão do trabalho em três etapas permite uma maior versatilidade e flexibilidade do tradutor, pois a etapa central de tradução fica encapsulada pelas duas estruturas de dados. Assim, para utilização do mesmo com outras ferramentas ou arquivos, deve-se apenas modificar o parser da primeira etapa, com o restante do projeto permanecendo intacto. O mesmo pode ser dito referente a utilização com outros provadores, neste caso, apenas a terceira etapa deve ser modificada.

Destarte, a partir dos padrões de modelagem utilizados em gramáticas de grafos, pelas quais vértices são destacados para representar objetos e arestas para representar as relações entre esses vértices, optou-se pela não utilização de atributos nas arestas, o que, por conseguinte, torna esses atributos não suportados pelo parser. Além disso, seguindo as restrições do método de

tradução definido por (CAVALHEIRO 2010), são consideradas apenas gramáticas que contém um grafo tipo.

3. RESULTADOS E DISCUSSÃO

A tradução do modelo matemático para o computacional foi a base para o trabalho. Todas as estruturas que foram implementadas seguiram as definições utilizadas em (CAVALHEIRO 2017), acrescentando métodos que se mostraram necessários no decorrer da implementação do algoritmo para obtemção dos dados sobre os grafos. Um enfoque especial foi dado na modelagem das estruturas de dados para representação de uma gramática de grafos em sua notação tradicional e em Event-B, de modo que tais estruturas fossem o mais genéricas possíves, maximizando assim sua flexibilidade e interoperabilidade.

Durante a primeira etapa, foi desenvolvido um parser, capaz de capturar os dados necessários do arquivo de saída padrão para definição de gramáticas de grafos tipadas, utilizando-se a linguagem Java. A ferramenta criada captura os dados necessários do arquivo fonte e os armazena em uma estrutura de dados desenvolvida para representar o sistema modelado na notação de uma gramática de grafos.

A seguir, durante a segunda etapa, é realizada a tradução da gramática encapsulada na estrutura desenvolvida para a notação matemática Event-B, armazenando-a em uma estrutura desenvolvida para esta finalidade.

A terceira e última etapa, gera os arquivos necessários para utilização da gramática traduzida no provador de teoremas semi-automático Rodin.

Como resultado, tem-se um tradutor automatizado de uma gramática de grafos para notação Event-B, gerando os arquivos necessários para a utilização no provador de teoremas Rodin.

4. CONCLUSÕES

O projeto visou agilizar e otimizar a utilização de gramáticas de grafos para modelagem de sistemas, através da exploração de bibliotecas e interfaces de programação, buscando definir modelos que maximizassem a possibilidade de reutilização em novas aplicações e que também atendessem todas as necessidades do algoritmo implementado.

Como resultado, temos um tradutor automatizado capaz de realizar a tradução de gramáticas que normalmente levariam horas ou dias em poucos segundos, além de minimizar o erro removendo a variável humana. Assim, todo o processo de utilização de gramáticas de grafos é otimizado, tornando-se mais eficiente como um todo.

Como trabalhos futuros, destaca-se a ampliação da compatibilidade do tradutor com outras ferramentas, principalmente de entrada, destaca-se a compatibilidade com a ferramenta Verigraph, desenvolvida especificamente para suprir as lacunas presentes na ferramenta de modelagem AGG (COSTA 2016).

5. REFERÊNCIAS BIBLIOGRÁFICAS

TEIXEIRA, Marcelo et al. A formal method applied to the automated software engineering with quality guarantees. In: **Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on**. IEEE, 2014. p. 108-111.

EHRIG, Hartmut; ROZENBERG, Grzegorz; RG KREOWSKI, Hans-J. **Handbook of graph grammars and computing by graph transformation**. world Scientific, 1999.

BALDAN, Paolo; CORRADINI, Andrea; KÖNIG, Barbara. Verifying finite-state graph grammars: an unfolding-based approach. In: **International Conference on Concurrency Theory**. Springer Berlin Heidelberg, 2004. p. 83-98.

KASTENBERG, Harmen; RENSINK, Arend. Model checking dynamic states in GROOVE. In: **International SPIN Workshop on Model Checking of Software**. Springer, Berlin, Heidelberg, 2006. p. 299-305.

KÖNIG, Barbara; ESPARZA, Javier. Verification of Graph Transformation Systems with Context-Free Specifications. In: **ICGT**. 2010. p. 107-122.

BOWEN, Jonathan P.; HINCHEY, Michael G. Ten commandments of formal methods. **Computer**, v. 28, n. 4, p. 56-63, 1995.

BOWEN, Jonathan P.; HINCHEY, Michael G. The use of industrial-strength formal methods. In: **Computer Software and Applications Conference, 1997. COMPSAC'97. Proceedings., The Twenty-First Annual International**. IEEE, 1997. p. 332-337.

CAVALHEIRO, Simone André da Costa. Relational approach of graph grammars. 2010.

CAVALHEIRO, Simone André da Costa; FOSS, Luciana; RIBEIRO, Leila. Theorem proving graph grammars with attributes and negative application conditions. **Theoretical Computer Science**, v. 686, p. 25-77, 2017.

EHRIG, Hartmut et al. Algebraic approaches to graph transformation: part ii: single pushout approach and comparison with double pushout approach. In: **Handbook of Graph Grammars**. 1997. p. 247-312.

RIBEIRO, Leila. Métodos formais de especificação: gramáticas de grafos. **VIII Escola de Informática da SBC-Sul. Porto Alegre: Editora da UFRGS**, p. 1-33, 2000.

ABRIAL, Jean-Raymond. **Modeling in Event-B: system and software engineering**. Cambridge University Press, 2010.

TAENTZER, Gabriele. AGG: A graph transformation environment for modeling and validation of software. In: **International Workshop on Applications of Graph Transformations with Industrial Relevance**. Springer, Berlin, Heidelberg, 2003. p. 446-453.

COSTA, Andrei et al. Verigraph: A System for Specification and Analysis of Graph Grammars. In: **Formal Methods: Foundations and Applications: 19th Brazilian Symposium, SBMF 2016, Natal, Brazil, November 23-25, 2016, Proceedings 19**. Springer International Publishing, 2016. p. 78-94.