GRAMÁTICA DE GRAFOS EM EVENT-B: ESTRATÉGIAS PARA DEMONSTRAR PROPRIEDADES UTILIZANDO PROVA DE TEOREMAS

LUIZ CARLOS LEMOS JUNIOR [1]; LUCIANA FOSS[1]; SIMONE ANDRÉ DA COSTA CAVALHEIRO[1]

[1]Universidade Federal de Pelotas – {lclemos,lfoss,simone.costa }@inf.ufpel.edu.br

1. INTRODUÇÃO

Uma forma de aumentar a confiabilidade em sistemas de computador está na utilização de métodos formais, os quais envolvem a especificação formal (utiliza um modelo matemático) e a verificação formal (permite garantir propriedades do sistema).

Existem diversas linguagens de especificação, entre elas se destaca gramática de grafos (GG) (EHRING et al., 1997), a qual é visual, baseada em um mecanismo simples de reescrita de regras e ao mesmo tempo capaz de descrever comportamentos complexos. De uma forma geral GG é composta por um grafo tipo (define os vértices e arcos permitidos); um grafo inicial (modela o estado inicial); e um conjunto de regras (descrevem o comportamento do sistema).

Como técnica de verificação se destaca prova de teoremas (ROBINSON; VORONKOV, 2001). Nesta técnica tanto o sistema como suas propriedades são descritas em alguma lógica. O processo de prova consiste em encontrar uma prova a partir de axiomas e lemas intermediários do sistema. É ideal para sistemas com espaços de estado grande e até mesmo infinito (DE MELLO et al., 2011).

Existe uma abordagem proposta por DA COSTA (2010) que permite a prova de teoremas de gramática de grafos. O trabalho apresenta uma abordagem lógica e relacional para GG, utilizando lógica de primeira ordem e teoria dos conjuntos. Condições lógicas impostas sobre relações garantem que elas codificam grafos e morfismos de grafos. Esta abordagem foi traduzida para estruturas do Event-B (DEPLOY, 2013), o que permitiu o uso dos provadores disponíveis para esta linguagem (através da plataforma Rodin (ABRIAL et al., 2010)) para o desenvolvimento das provas. No momento da verificação do sistema, a ferramenta gera obrigações de prova, e algumas necessitam da interação do usuário para sua demonstração. A interação, em alguns casos exige conhecimentos específicos do usuário, o que acaba restringindo o uso da técnica.

Este trabalho tem por objetivo apresentar estratégias de prova de propriedades atômicas, as quais auxiliam o desenvolvedor no momento da verificação. Tais estratégias são apresentadas através dos passos necessários para cada demonstração.

2. METODOLOGIA

Um modelo Event-B tem funcionamento semelhante a GG. O conceito de estado é representado por variáveis (um grafo em GG), uma transição de estado em Event-B é um evento que atualiza as variáveis (em GG uma aplicação de regra). Cada transição deve preservar as propriedades do sistema, em Event-B tal propriedade foi declarada como invariante (em GG a propriedade tem relação com a estrutura do grafo).

Em sistemas de GG em Event-B, um grafo G é definido por dois conjuntos: conjuntos de vértices $vertG$ e conjuntos de arcos $edgeG$, e por quatro funções: funções totais $sourceG$ e $targetG$, que definem a origem e destino, respectivamente de cada arco em um vértice; funções totais tG_V e tG_E, que mapeiam cada vértice e arco, respectivamente a um tipo. Na especificação do sistema todos os grafos da gramática seguem os padrões acima.

Em Event-B, no componente contexto se define o grafo tipo e regras. Estes grafos são representados através dos conjuntos, das constantes e axiomas. Os conjuntos representam os nomes dos conjuntos de arcos e vértices do grafo tipo e regras. As constantes definem os nomes dos vértices e arcos, os nomes das funções de origem e destino, e ainda os nomes das funções de tipagem. A última especificação feita no componente contexto são os axiomas, que definem explicitamente os tipos dos conjuntos e as constantes.

No componente máquina se especifica o grafo inicial e aplicações de regras. Cada item do grafo inicial e regras utilizam nomes que foram descritos como variáveis, as quais são definidas pelas invariantes. O grafo inicial é representado pelo evento de inicialização (as variáveis são iniciadas). As aplicações de regras são especificadas nos demais eventos. Quando ocorrer uma condição pré-estabelecida dentro da regra (chamada condição de guarda) o evento pode ocorrer (as variáveis são modificadas). Um exemplo de especificação de GG em Event-B pode ser analisado em LEMOS et al. (2013).

Após a especificação é possível verificar o sistema dentro do ambiente de prova da ferramenta. Para a verificação se utiliza táticas de prova e ou provadores, tanto disponíveis na plataforma quanto externos. O provador da ferramenta é chamado novo provador de predicados NewPP (possui três forças NewPP R., *restricted configuration*, NewPP (lasso), operação de lasso e NewPP, *unrestricted configuration*) como provador externo aparecem o provador de predicados PP (possui três forças P0, P1 e PP) e o mono-lemma (ML). Cada provador tem sua particularidade e a utilização depende do problema a demonstrar. Mais detalhes sobre os provadores podem ser obtidos em DEPLOY (2013). No ambiente de prova aparecem as obrigações de prova (OP), as quais devem ser demonstradas para garantir que toda mudança na variável, continue preservando seu tipo. De forma geral são geradas OP para o evento de inicialização e para cada uma das regras (eventos que atualizam uma variável).

Algumas OP são demonstradas de forma automática pela ferramenta, já outras necessitam da interação do usuário. Esta interação se resume em rodar um provador, adicionar uma hipótese, aplicar uma tática da ferramenta, executar uma regra de prova, entre outros. Em trabalhos anteriores foram identificadas as OP genéricas, geradas na especificação de uma GG em Event-B, bem como quais os provadores e táticas que devem ser executadas para sua demonstração (veja (LEMOS JR et al., 2013)).

Como invariante é possível definir uma propriedade específica, a qual deve ser válida para todo o sistema. Estas propriedades também geram obrigações de prova, as quais necessitam de interação. Esta interação exige um usuário conhecedor tanto do sistema quanto das regras utilizadas para sua demonstração. A Figura 2 apresenta algumas propriedades possíveis de serem especificadas como invariantes.

INVARIANTS

propFin : $finite(tG\_E \rhd \{t\})$
propCard : $card(tG\_E \rhd \{t\}) = 1$
propExEdge : $\exists x \cdot x \in tG\_E \rhd \{t\}$
propExVert : $\exists x \cdot x \in tG\_V \rhd \{t\}$

Figura 2: Propriedades Especificadas Como Invariantes

É possível observar quatro propriedades PropFin, PropCard, PropExEdge e PropExVert. A propriedade PropFin, garante que o conjunto de arcos de um determinado tipo t é finito. PropCard garante que a cardinalidade do conjunto de arcos de tipo t é exatamente 1. PropExEdge e PropExVert garantem a existência de um arco e vértice, respectivamente do tipo t. Tais propriedades declaradas como invariantes devem ser válidas para todos os estados alcançáveis do sistema. A demonstração destas propriedades pode ser considerada difícil, assim são apresentadas algumas estratégias.

Considerando a propriedade PropFin $finite(tG\_E \rhd \{t\})$, sua demonstração relativa ao evento de inicialização é dada de acordo com a execução dos seguintes passos: adicionar como hipótese $tG\_E \rhd \{t\} = \{x\}$, onde x representa o resultado de arcos do tipo t; executar o provador PP na força P1; executar o provador ML. Em geral, para regras que excluem e criam arcos do tipo t, é preciso aplicar a regra *rewrites range distribution left in goal* (regra disponível na ferramenta); adicionar como hipótese que os itens deletados pela aplicação da regra, restrito ao tipo t, seja subconjunto de $tG\_E \rhd \{t\}$. De mesma forma é necessário adicionar como hipótese que o conjunto dos novos arcos, restritos ao tipo t seja subconjunto dos arcos criados pela aplicação da regra. Após estes procedimentos é preciso executar o provador ML nos próximos dois objetivos, concluindo assim a prova da propriedade.

Uma estratégia de prova da PropCard $card(tG\_E \rhd \{t\}) = 1$ relativa ao evento de inicialização está na execução das seguintes etapas: se adiciona com hipótese $tG\_E \rhd \{t\} = \{e \mapsto t\}$, onde $e \mapsto t$ é o par resultante de tG_E restrito ao tipo t; após se executa o provador PP na força P1 nos dois próximos objetivos. Considerando regras que excluem e criam arcos de tipo t, em geral se adiciona como hipótese que a cardinalidade dos arcos deletados seja igual a 0. Após esta ação, se executa o provador ML nos próximos dois objetivos. Prosseguindo a demonstração, se adiciona como hipótese que a cardinalidade do conjunto de arcos deletados, restrito ao tipo t, é igual a 0, após se executa o provador NewPP (lasso) nos próximos dois objetivos. Para concluir a prova se instancia o novo arco de tipo t no objetivo (dentro do controle de prova) e se executa o provador NewPP (lasso).

Visando demonstrar a propriedade PropExEdge $\exists x \cdot x \in tG\_E \rhd \{t\}$ relativa ao evento de inicialização, é necessário executar o provador NewPP (lasso). Para os demais eventos, em geral é necessário instanciar o arco criado pela aplicação da regra, definido na propriedade. Após a instanciação é preciso executar o provador NewPP (lasso). Já a propriedade PropExVert $\exists x \cdot x \in tG\_V \rhd \{t\}$ é demonstrada aplicando o provador NewPP (lasso), está regra é válida tanto para o evento de inicialização quanto para os demais eventos do sistema (aplicações de regra).

3. RESULTADOS E DISCUSSÃO

Este trabalho apresenta táticas de prova para demonstrar propriedades atômicas em sistemas de GG especificados em Event-B. Estas propriedades

podem ser utilizadas em sistemas descritos neste formalismo, mais precisamente no momento da verificação. Foram descritos os passos para demonstrar cada uma das propriedades, em forma de estratégias de prova. Estes passos servem para auxiliar os desenvolvedores no momento da verificação. Cada sistema tem sua particularidade, assim se tem a necessidade de aumentar os tipos de propriedades, juntamente com as táticas para sua demonstração. Com isso é possível elevar o índice de uso das estratégias de prova.

Novas propriedades já estão sendo propostas, como a propriedade `PropELoop`, onde é possível demonstrar que em todos estados do sistema existirá um determinado arco com origem e destino em um mesmo vértice.

4. CONCLUSÕES

A principal contribuição do trabalho está nas táticas propostas para propriedades específicas. Tais táticas podem ser utilizadas pelos desenvolvedores no momento da verificação de sistemas de GG descritos em Event-B, utilizando a prova de teoremas. Em trabalhos futuros é pretendido mostrar novas propriedade e suas táticas, bem como criar uma configuração na ferramenta Rodin para sua aplicação de forma automática. Com esta ação é possível reduzir o tempo de verificação em sistemas que utilizem esta abordagem.

5. REFERÊNCIAS BIBLIOGRÁFICAS

ABRIAL, J. R.; BUTLER, M.; et al. Rodin: An Open Toolset for Modelling and Reasoning in Event-B. **International Journal on Software Tools for Technology Transfer**, Springer, 2010.

DA COSTA, S. A. **Relational Approach of Graph Grammars**. 2010. Tese (Doutorado em Ciência da Computação) – Curso de Pós-Graduação em Computação, UFRGS.

De MELLO, A. M.; LEMOS Jr., L. C.; FOSS, L.; CAVALHEIRO, S. A. da C. Graph grammars: A comparison between verification methods. In: **WEIT**, Pelotas, 2011, p. 88-94.

DEPLOY. **Event-b and the Rodin Platform**. Rodin Development is Supported by European Unicon ICT Projects DEPLOY (2008 to 2012) and RODIN (2004 to 2007), Acessado em outubro 2013. Online. Disponível em: http://www.event-b.org/

EHRIG, H.; HECKEL, R.; KORFF, M.;LOWE, M.;RIBEIRO, L.; WAGNER, A.; CORRADINI, A. **Handbook of graph grammars and computing by graph transformation**, River Edge, NJ, USA, v. I, p. 247 – 312, 1997.

LEMOS Jr., L. C.; FOSS, L.; CAVALHEIRO, S. A. da C. Theorem Proving Graph Grammars: Strategies for Discharging Proof Obligations. **Lecture Notes in Computer Science**, v.7316, p. 147-162, 2013.

LEMOS Jr., L. C.; FOSS, L.; CAVALHEIRO, S. A. da C. Towards the use of Proof Tactics for Theorem Proving Graph Grammars through Rodin. In: **WEIT**, Rio Grande, 2013, p. 1-8. To appear.

ROBINSON, J. A.; VORONKOV, A. **Handbook of Automated Reasoning**. Elsevier and MIT Press, 2001.