UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Uma abordagem relacional para Gramática de Grafos Fuzzy

Alex Bertei

Pelotas, 2017

Alex Bertei

Uma abordagem relacional para Gramática de Grafos Fuzzy

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação

Orientadora: Prof^{a}. Dra. Luciana Foss
Coorientadora: Prof^{a}. Dra. Simone André da Costa Cavalheiro

Pelotas, 2017

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

B537a Bertei, Alex

Uma abordagem relacional para gramática de grafos Fuzzy / Alex Bertei ; Luciana Foss, orientadora ; Simone André da Costa Cavalheiro, coorientadora. — Pelotas, 2017.

97 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2017.

1. Gramática de grafos Fuzzy. 2. Gramática de grafos. 3. Lógica Fuzzy. 4. Verificação de propriedades. I. Foss, Luciana, orient. II. Cavalheiro, Simone André da Costa, coorient. III. Título.

CDD : 005

Elaborada por Aline Herbstrith Batista CRB: 10/1737

UNIVERSIDADE FEDERAL DE PELOTAS
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO DE PÓS-GRADUAÇÃO

PPG Pró-Reitoria de Pesquisa e Pós-Graduação

5

DEFESA DE DISSERTAÇÃO

NOME DO ESTUDANTE	MATRICULA
ALEX BERTEI	15100768

CURSO OU PROGRAMA	NÍVEL
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO	MESTRADO

MEMBROS DA BANCA EXAMINADORA	TÍTULO	ASSINATURA
Luciana Foss (PPGC-UFPel)	DOUTOR	Luciana Foss
André Du Bois (PPGC-UFPel)	DOUTOR	
Renata Hax Sander Reiser (PPGC-UFPel)	DOUTOR	
Rodrigo Geraldo Ribeiro (Universidade Federal de Ouro Preto)	DOUTOR	Rodrigo Ribeiro

APRECIAÇÃO SOBRE A DISSERTAÇÃO

NÃO SIGILOSA

Em 13 de Março de 2017, os membros acima nomeados para a defesa da Dissertação do(a) estudante **ALEX BERTEI**, matriculado no Programa de Pós-Graduação em Computação, consideraram a Dissertação **aprovada**, estabelecendo o título definitivo como sendo

Uma abordagem relacional para Gramática de Grafos Fuzzy

e estabelecem um prazo máximo de 30 dias para as correções e entrega da versão definitiva.

DADOS PESSOAIS DOS MEMBROS DA BANCA EXAMINADORA

NOME COMPLETO	CPF	ANO NASCIMENTO	TITULAÇÃO		
			Área	Local	Ano
Luciana Foss (PPGC-UFPel)	70493430059	1974	Ciência da Computação	UFRGS	2008
André Du Bois (PPGC-UFPel)	93014716049	1977	Ciência da Computação	Heriot-Watt University	2006
Renata Hax Sander Reiser (PPGC-UFPel)	42930995068	1960	Ciência da Computação	UFRGS	2002
Rodrigo Geraldo Ribeiro (Universidade Federal de Ouro Preto)	047.772 106-00	1979	Ciência da Computação	Universidade Federal de Minas Gerais	2013

1ª Via – Coordenador do Curso; 2ª Via – Orientador; 3ª Via – PRPPG.
DISTRIBUIÇÃO A CARGO DA COORDENAÇÃO DO PROGRAMA.

AGRADECIMENTOS

Primeiramente, quero agradecer a Deus, pois com certeza é dele que veio a força maior para que eu conseguisse chegar até aqui, agradeço por ele me proporcionar muita luz e sabedoria para escolher meu caminho.

A minha noiva Jaqueline, pessoa amorosa, amigável, carinhosa e companheira. Obrigado por todo incentivo que me deste, por toda paciência, compreensão e apoio. Obrigado por sempre me transmitir segurança, amor e principalmente confiança em minhas escolhas. Essa conquista é nossa. Te Amo!

Agradeço aos meus pais que incessantemente se esforçaram ao máximo, para me proporcionar toda educação que tenho. A vocês Dirceu e Edilis só posso agradecer por sempre apoiarem e estarem ao meu lado em minhas escolhas, foi com o exemplo de vocês que sempre me mantive forte nesta jornada. Não dedico apenas esse trabalho, mas sim todas as minhas conquistas a vocês.

Ao meu irmão que sempre me apoiou e aguentou meus desabafos. Obrigado pela compreensão nos momentos em que estive ausente. Também gostaria de agradecer de forma geral a todos meus familiares e amigos que de alguma forma contribuíram com esse trabalho, só posso dizer que foi difícil passar todo esse tempo longe de vocês.

Agradeço a minha professora orientadora Luciana Foss, exemplo de profissional. Obrigado por me dar a oportunidade, por permitir que eu apreendesse com a senhora. Obrigado pela dedicação, paciência e grande incentivo na realização deste trabalho, pois tenho certeza que o trabalho desenvolvido não teria sido o mesmo sem a sua participação.

"Alguns homens vêem as coisas como são, e dizem ’Por quê?’ Eu sonho com as coisas que nunca foram e digo ’Por que não?’"— GEROGE BERNARD SHAW

RESUMO

BERTEI, Alex. **Uma abordagem relacional para Gramática de Grafos Fuzzy**. 2017. 97 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2017.

Com o passar dos dias os sistemas de software e hardware evoluem e ficam cada vez mais complexos e sofisticados. Como consequência especificar esses sistemas se torna uma tarefa difícil e trabalhosa. No sentido de assegurar certas propriedades de um sistema, este deve ser especificado por intermédio de uma linguagem que ofereça métodos de análise. Para realizar essas especificações temos diversas técnicas e metodologias, onde algumas técnicas que são disponíveis no mercado podem gerar resultados diferentes do que o especificado e esperado. Para realizar a correção deste problema, tem-se a necessidade de se utilizar métodos formais para gerar a especificação e verificação de sistemas. Sistemas que utilizam métodos formais são especificados formalmente através de um modelo matemático. Gramática de Grafos é uma linguagem formal bastante propícia para especificar sistemas complexos, é interessante o uso delas, pelo fato de possuírem inúmeras técnicas para a verificação e especificação de sistemas que são representadas nesta linguagem, além do que, elas possuem um layout gráfico, que as torna bastante intuitivas, por isso elas são uma linguagem de fácil entendimento. Gramática de Grafos Fuzzy são obtidas através da generalização das gramáticas de grafos, elas são constituídas por vértices e arestas com valores de pertinência associados, dentro do intervalo 0 e 1. Devido a inexistência de técnicas e ferramentas de análise para gramática de grafos fuzzy, o seu uso ainda é bastante restrito. Uma maneira de permitir a análise de sistemas especificados em gramática de grafos fuzzy é através da extensão das abordagens existentes para gramática de grafos, adicionando os conceitos de pertinência aos componentes dos grafos e para isso foi necessário definir gramáticas de grafos fuzzy tipados. Já existe uma abordagem relacional para gramática de grafos que permite o uso dos provadores de teoremas da ferramenta Rodin, utilizando a linguagem Event-B. Desta forma, a proposta deste trabalho é estender essa abordagem de gramática de grafos adicionando os conceitos fuzzy para permitir utilizar a mesma abordagem para analisar gramática de grafos fuzzy, surgindo assim um método de análise para esse tipo de gramática.

Palavras-chave: Gramática de grafos fuzzy, gramática de grafos, lógica fuzzy, verificação de propriedades.

ABSTRACT

BERTEI, Alex. **A Relational Approach for Fuzzy Graph Grammars**. 2017. 97 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2017.

With the passing of days software and hardware systems evolve and become increasingly sophisticated and complex. As a consequence, specifying these systems becomes a difficult and laborious task. In order to ensure certain properties of a system, it must be specified through a language that offers methods of analysis. To realize these specifications we have several techniques and methodologies, where some techniques that are available in the market can generate different results than the specified and expected. To perform the correction of this problem, there is the need of using formal methods for generate the specification and verification systems. Systems that use formal methods are formally specified through a mathematical model. Grammar Graphs is a very suitable to specify complex systems formal language, is interesting to use them because they have innumerable techniques for the verification and specification of systems that are represented in this language, besides that, they have a graphical layout, which makes them very intuitive, so they are an easy-to-understand language. Fuzzy graph grammars are obtained through the generalization of graph grammars, they are constituted by vertices and edges with associated membership values within the interval 0 and 1. Due to the inexistence of techniques and analysis tools for fuzzy graph grammars, its use is still quite restricted. One way to allow the analysis of systems specified in fuzzy graphs grammars is the extension of existing approaches for graphs grammars, adding the concepts of pertinence to the components of graphs and for this it was necessary to define fuzzy graph grammars typed. There is already a relational approach to graph grammar that allows the use of the theorems of the Rodin tool using the language Event-B. In this way, the purpose of this work is to extend this approach of grammars of graphs by adding the fuzzy concepts to allow to use the same approach to analyze fuzzy graphs grammars, thus appearing a method of analysis for this type of grammar.

Keywords: Fuzzy graph grammars, Grammar graphs, Fuzzy Logic, checking properties.

LISTA DE FIGURAS

Figura 1 Grafo 17
Figura 2 Morfismo de grafos 18
Figura 3 Grafo tipado G^T 19
Figura 4 Morfismo de grafos tipados 19
Figura 5 Regra 20
Figura 6 Grafo tipo da gramática de grafos Supermercado 21
Figura 7 Grafo inicial da gramática de grafos Supermercado 21
Figura 8 Regras da gramática de grafos Supermercado 22
Figura 9 Ocorrência 23
Figura 10 Passo de derivação 26

Figura 11 Conjunto Fuzzy 28
Figura 12 Morfismo de conjuntos Fuzzy 29
Figura 13 Grafo Fuzzy G 29
Figura 14 Morfismo de Grafos Fuzzy 30
Figura 15 Grafo fuzzy tipado G^T 31
Figura 16 Morfismo de grafos fuzzy tipados g^T 32
Figura 17 Regra fuzzy 33
Figura 18 grafo tipo fuzzy da gramática de grafos fuzzy ferrovia 34
Figura 19 grafo inicial fuzzy da gramática de grafos fuzzy ferrovia 35
Figura 20 Regras da gramática de grafos fuzzy ferrovia 35
Figura 21 Aplicação da regra p 36

Figura 22 Grafo tipo da gramática Supermercado no modelo Event-B 42
Figura 23 Grafo inicial da gramática Supermercado no modelo Event-B 44
Figura 24 Tradução do lado esquerdo de uma regra para o modelo Event-B . 46
Figura 25 Tradução da aplicação de uma regra para o modelo Event-B 50

Figura 26 Teoria dos Reais 53
Figura 27 Grafo tipo fuzzy da gramática de grafos fuzzy Ferrovia 55
Figura 28 Grafo inicial fuzzy da gramática de grafos fuzzy Ferrovia 58
Figura 29 Tradução do lado esquerdo da regra r1 GGF Ferrovia. 60
Figura 30 Tradução da aplicação da regra r1 63

LISTA DE TABELAS

Tabela 1 Mapeamento da pertinência das arestas no grafo inicial fuzzy 57

LISTA DE ABREVIATURAS E SIGLAS

GGs	Gramática de Grafos
GGF	Gramática de Grafos Fuzzy
LF	Lógica Fuzzy
DPO	Double Pushout
LHS	Lado Esquerdo
RHS	Lado Direito

SUMÁRIO

1 INTRODUÇÃO 12
1.1 Objetivos e benefícios 14
1.2 Organização do trabalho 15

2 GRAMÁTICA DE GRAFOS 16
2.1 Sintaxe de Gramática de grafos 16
2.2 Semântica de Gramática de Grafos 22

3 GRAMÁTICA DE GRAFOS FUZZY 27

4 EVENT-B 37
4.1 Contextos em Event-B 38
4.2 Máquinas em Event-B 38
4.3 Tradução da Gramática de Grafos no Event-B 40

5 TRADUÇÃO DA GRAMÁTICA DE GRAFOS FUZZY PARA O MODELO EVENT-B 51
5.1 Teoria Plug-in 51
5.2 Tradução do grafo tipo fuzzy 54
5.3 Tradução do grafo inicial fuzzy 56
5.4 Tradução das regras fuzzy 57

6 CONCLUSÃO 64

REFERÊNCIAS 66

ANEXO A ESPECIFICAÇÃO DA GRAMÁTICA DE GRAFOS SUPERMERCADO NO EVENT-B 69
A.1 Contexto da Gramática de Grafos Supermercado no modelo Event-B 69
A.2 Máquina da Gramática de Grafos Supermercado no modelo Event-B 72

ANEXO B ESPECIFICAÇÃO DA GRAMÁTICA DE GRAFOS FUZZY FERROVIA NO EVENT-B 80
B.1 Contexto da Gramática de Grafos fuzzy ferrovia no modelo Event-B . 80
B.2 Máquina da Gramática de Grafos fuzzy ferrovia no modelo Event-B . 86

1 INTRODUÇÃO

Em nosso cotidiano os sistemas de software e hardware estão sempre presentes e fazem parte, de alguma forma, dos sistemas operacionais em indústrias e empresas, na organização e informatização de instituições e no desenvolvimento de novas tecnologias. Sistemas cada vez mais sofisticados tornam mais complexa a especificação de tarefas de forma robusta e consistente, promovendo minimização, controle ou mesmo eliminação de erros, ruídos ou redundâncias. Esses sistemas estão sempre evoluindo e crescendo em escala e alcance, muitas vezes interagem com outros ambientes complexos e independentes. Devido ao aumento de complexidade, a chance de erros se intensifica e pode levar a perdas catastróficas, sejam de tempo, dinheiro ou de vidas. Com isso, é necessário, então, que sejam utilizados métodos eficientes para modelagem e especificação de sistemas de software, capazes de representar suas características, além de possibilitar a verificação de propriedades. Uma maneira de alcançar este objetivo é através do uso de métodos formais, que são técnicas baseadas em formalismos matemáticos que podem oferecer formas rigorosas e eficazes para modelar, projetar e analisar sistemas de computador (CRAIGEN; GERHART; RALSTON, 1993).

Inúmeros estudos de caso e aplicações durante os últimos anos confirmaram a importância da utilização de métodos formais para melhorar a qualidade dos sistemas que demandam um alto custo computacional. Um exemplo é o sistema de metrô de Paris (BEHM et al., 1999), o qual é totalmente automático e teve suas partes críticas de segurança formalmente desenvolvida pela Matra Transport International usando B (ABRIAL, 1996).

A descrição de um sistema especificado por uma linguagem formal tem demonstrado fornecer uma base sólida para orientar as atividades de desenvolvimento e obter através da verificação uma elevada confiança de que o sistema satisfaz as suas necessidades. As especificações bem formadas, validadas em relação às propriedades criticas, forneceram uma base para gerar código-fonte correto e eficiente.

Gramática de grafos (GG) é uma linguagem formal que tem sido estudada desde os anos 70 e aplicada em campos da Ciência da Computação que requerem mode-

los de grafos dinâmicos (grafos que podem sofrer transformações ou manipulações em sua estrutura). É uma linguagem visual que permite especificar e verificar formalmente sistemas com características complexas como concorrência, paralelismo e distribuição. Existem diferentes técnicas e ferramentas que permitem o uso de *model checking* (RENSINK, 2008), (FERREIRA; FOSS; RIBEIRO, 2007), (BARESI et al., 2008), (KASTENBERG; RENSINK, 2006), (DOTTI et al., 2003) e provadores de teoremas (CAVALHEIRO, 2010), (RIBEIRO et al., 2010), (COSTA; RIBEIRO, 2012), para a verificação formal de propriedades dos sistemas descritos nesta linguagem. Esta linguagem permite a especificação de linguagens formais, reconhecimento de padrões, reconhecimento e geração de imagens, construção de compiladores e ainda modelagem de sistemas concorrentes e distribuídos, entre tantas outras aplicações (ROZENBERG, 1997).

A lógica fuzzy (LF) (ZADEH, 1965), (ATANASSOV et al., 2003), (ATANASSOV; GARGOV, 1989), (BUSTINCE; BARRENECHEA; MOHEDANO, 2004), (BUSTINCE et al., 2010), (JIANG, 2011) disponibiliza uma modelagem de sistemas de inferência mais flexíveis para a área de tomada de decisões, onde se utilizam conectivos lógicos para atuar sobre variáveis linguísticas.

Gramáticas de Grafos Fuzzy (GGFs) são uma generalização de GGs, que permitem tratar informações imprecisas e vagas para especificação formal de sistemas de forma análoga ao do raciocínio humano. Sendo assim, temos uma modelagem mais flexível para os sistemas que são especificados via GGs, permitindo valores intermediários dentro do intervalo 0 e 1 associados aos vértices e arestas dos grafos que modelam sistemas computacionais. No trabalho de (PARASYUK; YERSHOV, 2006) é introduzida uma abordagem categórica para GGF, onde é desenvolvida uma base conceitual para a teoria de GGFs, além disso, essa abordagem trata de viabilizar a investigação das relações entre vértices e arestas dos grafos fuzzy que são gerados por tais gramáticas. Neste trabalho ainda é mostrado a possibilidade de transformação de uma Gramática de Chomsky em uma correspondente na categoria de grafos fuzzy, ainda é feito uma discussão da indecidibilidade dos problemas do vazio, finitude e equivalência para um grafo fuzzy arbitrário.

Gramáticas de grafos fuzzy distribuídos para modelagem e especificação de sistemas distribuídos dinâmicos são propostas em (PARASYUK; ERSHOV, 2007), que é baseada em uma aplicação de transformações sobre as estruturas de rede de componentes fuzzy que são denominados de FD-grafos. Com esta abordagem, a combinação de grafos fuzzy e as transformações sobre estes grafos podem modelar a interação e a sincronização de FD-grafos. As transformações de FD-grafos são dadas pelo uso da abordagem categórica nas quais os vértices são especificados por grafos fuzzy. Essa abordagem provê a formalização para definir as transformações de FD-grafos como transformações estruturadas de grafos fuzzy. Há ainda uma discus-

são sobre as condições necessárias para não violar a estrutura das transformações na categoria dos FD-grafos. Assim, a construção dos pushouts (e co-limites) para esta categoria é formalizada pela construção dos pushouts dos componentes. As FD-gramáticas são generalizadas através das GGF (PARASYUK; ERSHOV, 2007).

Para as gramáticas de grafos fuzzy não encontram-se muitas aplicações. No trabalho de (WATKINS, 1996) é proposta uma abordagem fuzzy para a interpretação dos ruídos em imagens bidimensionais, usando gramáticas de grafos fuzzy. Nesta abordagem é permitido utilizar funções fuzzy nos predicados de aplicabilidade das regras. Esse novo formalismo tem como intuito fazer a modelagem e a especificação dos ruídos que são testados no contexto do reconhecimento de partituras manuscritas. Essa técnica é aplicada a uma barra de música, onde é apresentada a interpretação correta, e a assinatura de tempo é interpretada como um ruído.

Devido a inexistência de técnicas e ferramentas de análise para GGFs, o seu uso ainda é bastante restrito. Uma maneira de permitir a análise de sistemas especificados em GGFs é através da extensão das abordagens existentes para GGs, adicionando os conceitos de pertinência aos componentes dos grafos. Na tese de Cavalheiro (CAVALHEIRO, 2010) é proposta uma abordagem relacional para GGs que permite o uso dos provadores de teoremas da ferramenta Rodin. Esta abordagem é importante pois é possível conseguir sistemas com espaço de estados muito grandes ou até infinitos. Desta forma, a proposta deste trabalho é estender o trabalho de Cavalheiro adicionando os conceitos fuzzy para permitir utilizar a mesma abordagem para analisar GGFs.

1.1 Objetivos e benefícios

O objetivo principal deste trabalho é estender a abordagem relacional de GG para GGF, com o intuito de especificar esse tipo de gramática na ferramenta Rodin, utilizando a linguagem Event-B, permitindo assim o uso dos provadores desta ferramenta para realizar a análise de propriedades que são especificadas por esse tipo de gramática. Para a realização deste objetivo principal, evidenciam-se alguns objetivos particulares como:

- Estender a abordagem da GGF para ser possível usar grafos fuzzy tipados e assim desenvolver uma gramática de grafos fuzzy tipada.
- Estudar GGF e definir um método para representar os graus de pertinência dos vértices e das arestas no Event-B;
- Analisar a linguagem Event-B , de forma a identificar a melhor maneira de realizar a tradução da GGF;

- Desenvolver um estudo de caso usando a ferramenta Rodin para validar a proposta estabelecida.

Um dos principais benefícios decorrentes do desenvolvimento da abordagem proposta é o uso da ferramenta Rodin para a análise e verificação das propriedades estabelecidas através da técnica de prova de teoremas, com isso os desenvolvedores de software terão uma maneira de especificar e verificar formalmente GGFs.

1.2 Organização do trabalho

O restante do texto está organizado conforme descrito a seguir. Em seguida, no capítulo 2 são apresentados os conceitos básicos e as definições da gramática de grafos, definindo sua sintaxe e semântica, através das definições e assim demonstrando essas com exemplos.

No capítulo 3 são descritos os principais fundamentos de GGs fuzzy, como conjuntos, grafos e morfismos fuzzy, ainda é mostrado um exemplo de uma GGF . O capítulo 4 descreve a linguagem formal Event-B, neste capítulo é descrita informalmente a tradução de uma GG (definida no capítulo 2) para o Event-B.

O capítulo 5 descreve a tradução da GGF (definida no capítulo 3) para a linguagem Event-B, para realizar essa tradução foi necessário utilizar a teoria dos reais, que é uma teoria disponível pela linguagem.

Por fim, no capítulo 6, são apresentadas as principais conclusões alcançadas através deste trabalho.

2 GRAMÁTICA DE GRAFOS

Nesse capítulo, serão abordados os principais conceitos de GGs, de acordo com a abordagem algébrica (EHRIG; PFENDER; SCHNEIDER, 1973), (ROZENBERG, 1997).

Gramáticas de Grafos são utilizadas para especificação de vários tipos de sistemas de software (ROZENBERG, 1997). GGs são a generalização das gramáticas de Chomsky, substituindo strings por grafos. Um grafo consiste de vértices e arestas que conectam esses vértices, os grafos são usados para representar estados dos sistemas e as regras de transformação de grafos para descrever operações ou mudanças de estado destes sistemas. GGs são um método promissor para o desenvolvimento de software confiável, por serem formais e intuitivas ao mesmo tempo, elas permitem tratar com simplicidade aspectos de concorrência e distribuição de sistemas.

Gramática de grafos é uma linguagem formal e visual, o que a torna bastante intuitiva e permite a fácil compreensão do modelo para os não teóricos, por isto ela é uma linguagem formal adequada para especificar vários sistemas computacionais. Existem diferentes técnicas e ferramentas que permitem o uso de *model checking* e prova de teoremas para verificação de propriedades de sistemas descritos nesta linguagem. Nas próximas seções serão definidas a sintaxe e a semântica das GGs.

2.1 Sintaxe de Gramática de grafos

Gramáticas de grafos generalizam gramáticas de Chomsky onde usam grafos ao invés de strings. Diferente das regras nas gramáticas de Chomsky, uma regra de grafos é composta de um grafo do lado esquerdo (L) e um grafo do lado direito (R), e por um (homo)(-)morfismo parcial de grafos p, o qual define um mapeamento dos elementos do grafo L em elementos do grafo R, estabelecendo como uma ocorrência do grafo do lado esquerdo pode ser substituída pela ocorrência do grafo do lado direito em um determinado grafo G. Uma regra só pode ser aplicada no grafo G, se existir uma ocorrência do lado esquerdo da regra no grafo G. Já o lado direito da regra define o grafo resultante da aplicação da regra.

Se as estruturas de dois grafos são compatíveis, então significa que eles estão relacionados. Essa relação é dada por um morfismo de grafos, onde esse morfismo mapeia os vértices e as arestas de um determinado grafo em vértices e arestas de outro grafo, respectivamente, preservando a origem e o destino das arestas.

Definição 1 (Grafo, morfismo de grafos) *Um* ***grafo*** G *é uma tupla* $(VertG, EdgeG, sourceG, targetG)$*, onde* $VertG$ *é um conjunto de vértices,* $EdgeG$ *um conjunto de arestas, e* $sourceG$, $targetG : EdgeG \rightarrow VertG$ *são funções totais, definindo a origem e o destino de cada aresta, respectivamente. Dados dois grafos* $G = (VertG, EdgeG, sourceG, edgeG)$ *e* $H = (VertH, EdgeH, sourceH, edgeH)$*, um* ***morfismo de grafos*** $f : G \rightarrow H$ *é uma tupla* $(f_V : VertG \rightarrow VertH, f_E : EdgeG \rightarrow EdgeH)$*, de funções totais que preservam origem e destino das arestas, ou seja,*

$$f_V \circ sourceG = sourceH \circ f_E \text{ e}$$
$$f_V \circ targetG = targetH \circ f_E$$

Um morfismo de grafos é injetor se seus componentes forem funções injetoras.

Exemplo 1 (Grafo, morfismo de grafos) *Na Figura 1 é apresentado um exemplo de grafo G, onde este grafo é composto por dois vértices* $VertG = \{$👤, 🏧$\}$ *e três arestas* $EdgeG$*={*$Cli\_Caixa, Atend, Ocup$*}. A função de origem associa as arestas aos vértices do seguinte modo:* $sourceG(Cli\_Caixa) =$ 👤*,* $sourceG(Atend) =$ 👤*,* $sourceG(Ocup) =$ 🏧*, já a função de destino determina a seguinte associação:* $targetG(Cli\_Caixa) =$ 👤*,* $targetG(Atend) =$ 🏧*,* $targetG(Ocup) =$ 🏧*.*

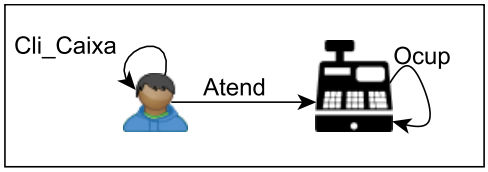


Figura 1: Grafo

A Figura 2 mostra o mapeamento de um morfismo do grafo G *para o grafo* H*.*

Geralmente é usado algum tipo de mecanismo para dar a tipagem nos grafos em vez de usar apenas grafos simples que possuem somente arestas e vértices. Existem três maneiras de realizarmos a tipagem de grafos, ela pode ser feita usando rótulos, atributos ou grafo tipo, neste trabalho a tipagem é dada através do grafo tipo. O grafo tipo é um grafo que contém todos os tipos de vértices e arestas de um sistema, e todos

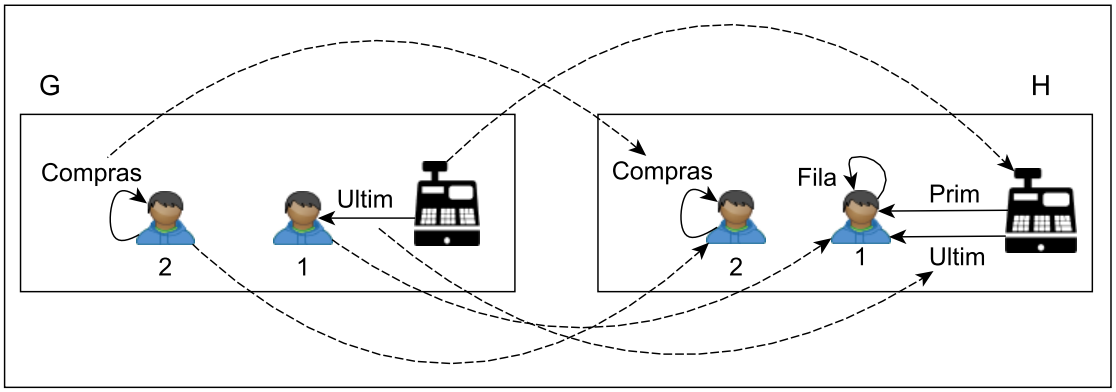


Figura 2: Morfismo de grafos

os grafos que fizerem parte da gramática estão restritos a esses tipos. Essa restrição é dada por um morfismo de grafos, que será responsável de mapear cada grafo no grafo tipo.

Definição 2 (Grafo tipado, morfismo de grafos tipados) *Um **grafo tipado** G^T é definido por uma tupla $G^T = (G, tG, T)$, onde G é um grafo, T o grafo tipo e $tG: G \to T$ é um morfismo de grafos. Quando é possível determinar de maneira exata o tipo de G^T, podemos ocultar o grafo tipo da notação, denotando-o apenas por G. Um **morfismo de grafos tipados** G^T para H^T é definido por um morfismo de grafos $g = (g_V, g_E)$ de G em H de tal modo que a condição de compatibilidade de morfismos tipados seja satisfeita:*

$$tG_V = tH_V \circ g_V \text{ e}$$
$$tG_E = tH_E \circ g_E$$

A categoria dos grafos e morfismos de grafos tipados em T é denotada por T-Graph. Pushouts em T-Graph são construídos a partir do pushout dos componentes dos morfismos em Set (categoria dos conjuntos e funções totais).

Exemplo 2 (Grafo tipado, morfismo de grafos tipados) *Sendo o grafo T da Figura 3 um grafo tipo e o grafo G uma instância deste grafo tipo, um morfismo de grafos é dado pelas setas tracejadas do grafo G para o grafo T, onde os vértices 👤 e 🏪 são mapeados para os tipos 👤, 🏪, respectivamente, e a aresta $Cliente\_livre$ é mapeada para o tipo $Cliente\_livre$, definindo assim o grafo tipado G^T. A Figura 4 demonstra um exemplo de morfismo de grafos tipados, este morfismo deleta as arestas Cli_Caixa, $Atend$ e $Ocup$ e cria as arestas $Cliente\_livre$ e $Livre$.*

Nas GGs, os estados do sistema são representados por grafos e as possíveis alterações de estados ou o comportamento da gramática são descritas por regras,

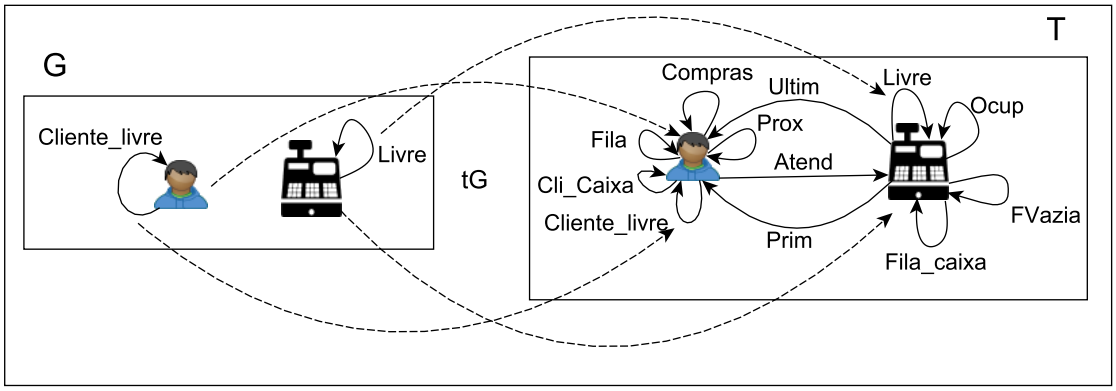


Figura 3: Grafo tipado G^T

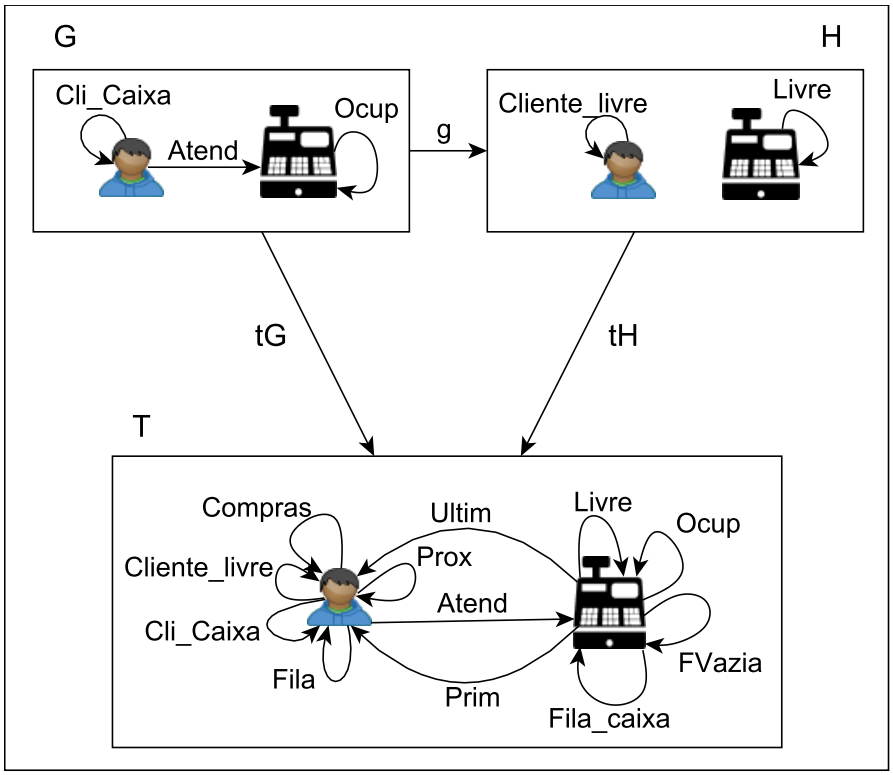


Figura 4: Morfismo de grafos tipados

sendo assim, a regra define quais elementos que devem estar presentes no grafo para que ela possa ser aplicada e as possíveis alterações realizadas por sua aplicação, onde alguns elementos são preservados, outros são criados e outros são eliminados. Na abordagem DPO (Double Pushout) (ROZENBERG, 1997), a regra é composta por três grafos, o lado esquerdo L, o lado direito R e a interface K que representa os elementos comuns nos lados L e R. Isso significa, que se uma ocorrência do grafo L é encontrada no corrente estado do sistema, ela pode ser substituída pelo grafo R, preservando K.

Definição 3 (Regra) *Dado um grafo tipo T, uma **regra** é um morfismo de grafos T-tipados injetivos $(L \overset{l}{\hookleftarrow} K \overset{r}{\hookrightarrow} R)$, onde L é chamado de lado esquerdo (LHS), R*

de lado direito (RHS) e K de interface da regra. Dado uma regra $p : L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$, *definimos os seguintes conjuntos:*

Vértices deletados de L: $RegrasDel_V^p = VertL \backslash rng\ ^1(pl_V)$;
Arestas deletadas de L: $RegrasDel_E^p = EdgeL \backslash rng(pl_E)$;
Vértices preservados de L: $RegrasPreserv_V^p = VertL \backslash RegrasDel_V^p$;
Arestas preservadas de L: $RegrasPreserv_E^p = EdgeL \backslash RegrasDel_E^p$;
Vértices criados de L: $RegrasCriadas_V^p = VertR \backslash rng(pr_V)$;
Arestas criadas de L: $RegrasCriadas_E^p = EdgeR \backslash rng(pr_E)$;

Exemplo 3 (Regra) *Uma regra* p *é demonstrada na Figura 5, onde* L_p *é o lado esquerdo,* R_p *o lado direito e* K_p *a interface. Esta regra pode ser aplicada num estado contendo o grafo* L_p *(que contém os vértices*  *, e as arestas* $compras$, $Livre$ *e* $FVazia$*), resultando em um grafo onde as arestas* $compras$ *e* $Livre$ *são deletadas, os vértices*  *, e a aresta* $FVazia$ *são preservadas (grafo* K_p*), e as arestas* $Atend$, Cli_Caixa *e* $Ocup$ *foram criadas.*

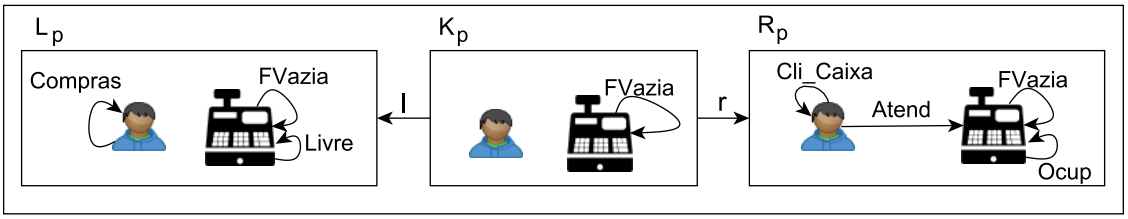


Figura 5: Regra

Gramática de grafos são compostas por um grafo inicial, que representa o estado inicial do sistema e tem como principal função restringir a computação e os estados alcançáveis permitidos pelo sistema, um grafo tipo, que define os tipos dos elementos que podem estar presentes no sistema, e um conjunto de regras, que descrevem o comportamento deste sistema.

Definição 4 (Gramática Grafos) *Uma* ***gramática de grafos*** *tipada é uma tupla* $GG=(T, I, P)$, *onde* T *é o grafo tipo;* I *é o grafo inicial (tipado);* P *é o conjunto de regras tipadas em* T.

Exemplo 4 (Gramática de Grafos) *Este exemplo descreve uma gramática de grafos que especifica um sistema no qual um cliente faz as suas compras em um supermercado e efetua o pagamento. Além disso, o cliente deve ter que entrar na fila, a fim de realizar este pagamento. O grafo tipo* $\mathbf{T}$ *é representado pela Figura 6 e o grafo inicial* $\mathbf{I}$

[1] rng(r) é imagem de uma relação binária r.

*é representados pela Figura 7. Os clientes e os caixas são representados por vértices (veja **T**). Arestas com origem em clientes ou caixas representam informações sobre os mesmos. Um cliente pode estar livre (Cliente_livre), fazendo compras (Compras), esperando em uma fila (Fila), ou ainda em um caixa (Cli_Caixa). Ele pode estar sendo atendido pelo caixa (Atend) ou estar na fila. Neste último caso, o cliente pode ser o primeiro (Prim) ou o último (Ultim) da fila e a aresta Prox indica qual o próximo cliente (quando existir). Um caixa pode estar livre (Livre) ou atendendo um cliente (Ocup) e pode ter fila (Fila_caixa) ou não (FVazia). O grafo inicial descreve um estado onde tem-se dois caixas e dois clientes. O primeiro cliente é o único na fila do caixa 1, que está livre. Já o segundo cliente ainda não iniciou suas compras, enquanto o segundo caixa está livre e sem fila. As regras da gramática são apresentadas na Figura 8, as quais definem o comportamento dos clientes no supermercado. Chegando ao supermercado, um cliente começa a comprar (regra* $\mathbf{Inicio}$*), eventualmente ele pode se dirigir a um caixa. Caso haja fila no caixa, o cliente deve entrar no final dela (regras* $\mathbf{Entrar1}$ *e* $\mathbf{Entrar2}$*), caso contrário, ele é atendido imediatamente e o caixa muda seu status para ocupado (regra* $\mathbf{Pagar1}$*). Quando um caixa está com o status livre e existe um cliente no início da fila, este cliente sai da fila e é atendido, a fila é atualizada e o caixa muda seu status para ocupado (regras* $\mathbf{Pagar2}$ *e* $\mathbf{Pagar3}$*). Depois de pagar suas compras o cliente pode deixar o caixa, atualizando o seu e o status do caixa para livre (regra* $\mathbf{Pago}$*). Por razões de espaço o grafo* K *na Figura 8 é omitido, mas ele pode ser reconstruído pela interseção dos grafos* L *e* R*.*

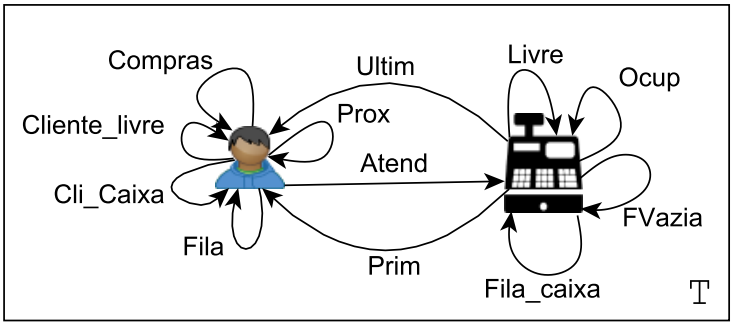


Figura 6: Grafo tipo da gramática de grafos Supermercado

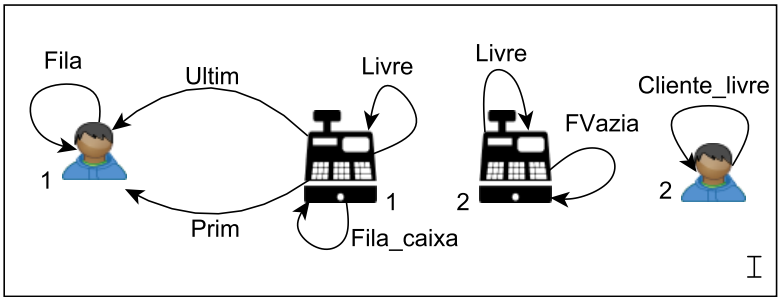


Figura 7: Grafo inicial da gramática de grafos Supermercado

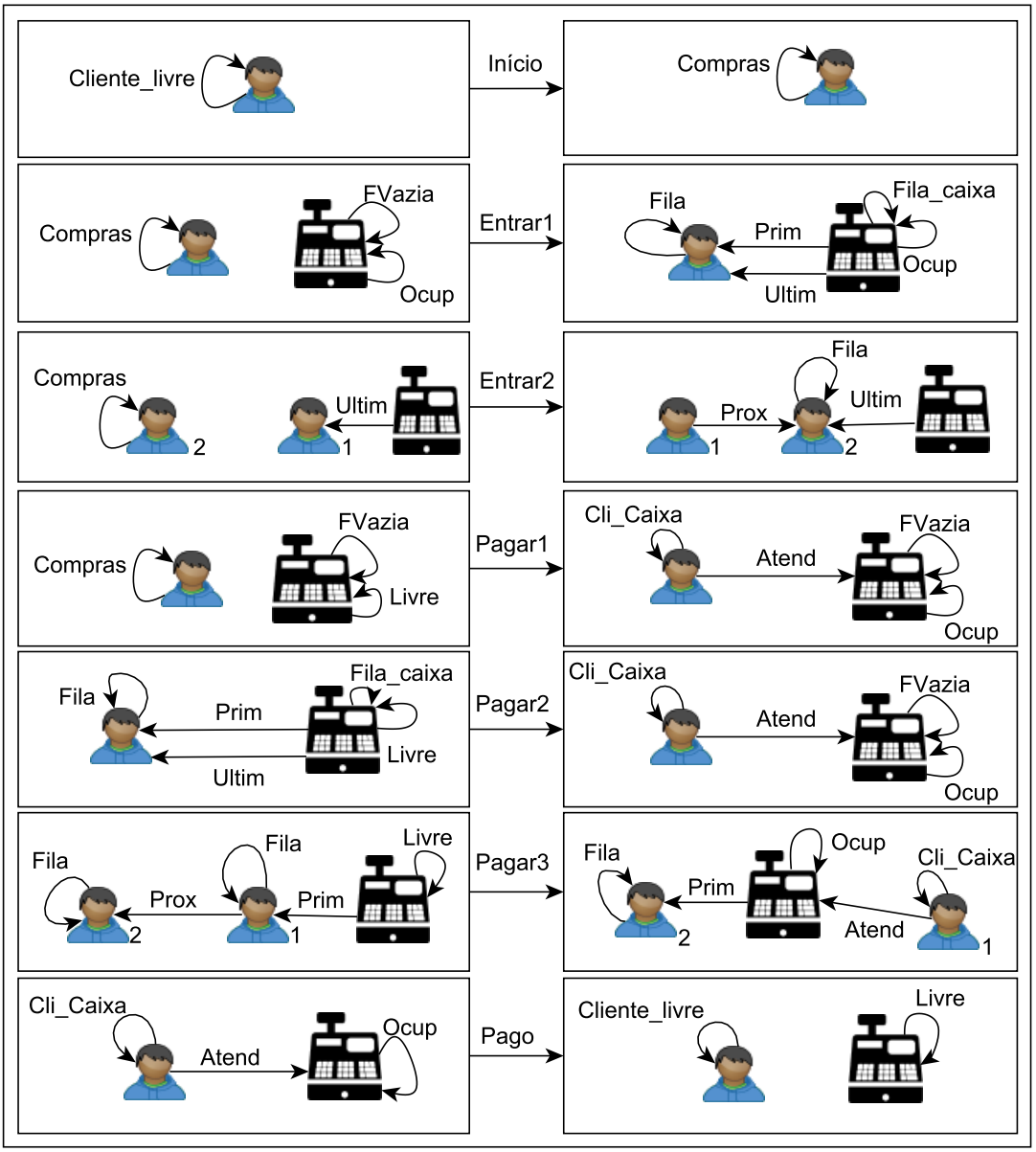


Figura 8: Regras da gramática de grafos Supermercado

2.2 Semântica de Gramática de Grafos

O comportamento de um sistema é descrito pela semântica de uma especificação. Um sistema especificado por uma GG tem o comportamento determinado pelas aplicações das regras nos grafos que representam o estado do sistema. Desta forma, a semântica é dada por derivações a partir do grafo inicial.

Uma regra está habilitada a ser aplicada, caso exista uma ocorrência do lado esquerdo da regra no grafo estado atual, neste caso, existe um morfismo de grafos que mapeia o lado esquerdo da regra no grafo estado.

Definição 5 (Ocorrência) *Dada uma regra* $p : L \leftarrow K \rightarrow R$ *e um grafo* G*. Uma* ***ocorrência*** *de* p *em* G *é um morfismo injetor de grafos tipados* $m : L \rightarrow G$*.*

Exemplo 5 (Ocorrência) *Na Figura 9 é ilustrada uma ocorrência da regra $Pagar2$ no grafo G, a ocorrência é marcada no grafo, por um retângulo de linha tracejadas.*

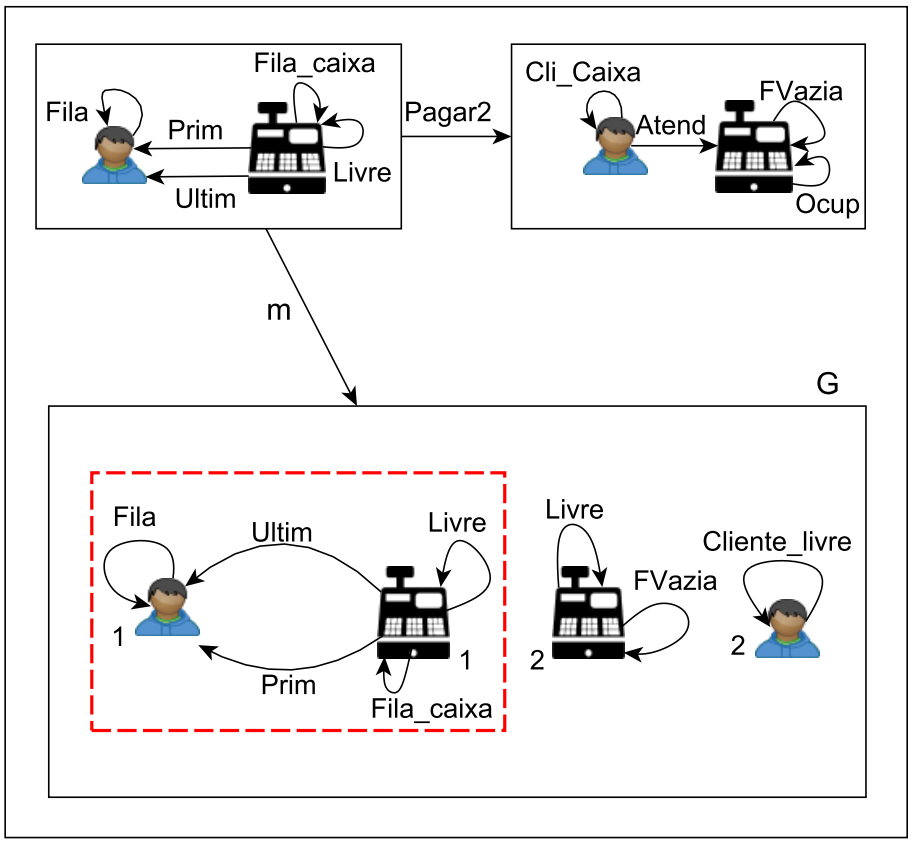


Figura 9: Ocorrência

A seguir são definidos alguns conjuntos que serão úteis para construir o resultado da aplicação de uma regra. Dado um grafo G e uma regra descrevemos quais são os conjuntos de vértices e as arestas que serão excluídos, preservados e criados pela aplicação desta regra. Os conjuntos Del_V e Del_E tem como elementos vértices e arestas que são imagens de itens especificados para exclusão na regra. O conjunto $Dangling$ contém todas as arestas que não estão na imagem do lado esquerdo da regra, mas seus vértices de origem e/ou destino devem ser excluídos. $PreservV$ contém todos os vértices que serão preservados pela aplicação regra, $NewV$ e $NewE$ tem todos os itens que serão criados pela aplicação da regra.[2]

Definição 6 (Itens deletados, preservados e criados) *Seja p uma regra tipada com o lado esquerdo L, uma ocorrência $m = (m_V, m_E) : L \to G$. Em seguida, definimos os seguintes conjuntos:*

Vértices deletados de G: $Del_V^{p,m} = m_V[RegrasDel_V^p]$;

Arestas deletadas de G: $Del_E^{p,m} = m_V[RegrasDel_E^p]$;

[2]Quando a regra é aplicada novas instâncias de elementos de $NewV$ e $NewE$ são criados.

Dangling das arestas de G: $Dangling^{p,m} = dom((sourceG \triangleright Del_V^{p,m})$[3]$\cup(targetG \triangleright Del_V^{p,m}))\backslash Del_E^{p,m}$*;*

Vértices preservados de G: $Preserv_V^{p,m} = VertG\backslash Del_V^{p,m}$*;*

Novos vértices de R: $New_V^{p,m} = RegrasCriadas_V^p$*;*

Novas arestas de R: $New_E^{p,m} = RegrasCriadas_E^p$*;*

Um sistema que é modelado por uma GG tem o seu comportamento descrito pela aplicação das regras nos grafos que representam os estados do sistema. Uma regra p pode ser aplicada em um grafo G se existir uma ocorrência do lado esquerdo da regra p no grafo G. Se essa condição é satisfeita, o próximo passo é excluir do grafo G todos os itens do lado esquerdo L que não fazem parte do grafo interface K, com isso é gerado o grafo X, após se faz a colagem de R com X, criando os itens do lado direito R que não estão em K, e assim é gerado o grafo H, que é o resultado do passo de derivação da regra.

Formalmente, a aplicação de um regra é determinada pela construção de dois pushouts em T-Graph, onde o primeiro pushout apaga todos os itens que devem ser consumidos e o segundo pushout inclui todos os itens que devem ser criados pela regra.

A aplicação de uma regra de transformação de grafo em geral pode levar a efeitos secundários. Isso pode acontecer, por exemplo, quando um vértice é especificado para exclusão e há arestas conectadas a ele no grafo onde a regra é aplicada. O esquema construído acima depende da existência de X, que é chamado de complemento do pushout. Para que seja garantida a existência de X, a ocorrência m deve atender a condição de colagem. A condição de colagem é dividida em duas partes:

1) condição de arestas pendentes: se um vértice é deletado, então não podem existir arestas que partem ou chegam neste vértice;

2) condição de identificação: dois vértices podem ser identificados por m somente se ambos são preservados.

Definição 7 (Condição de Colagem) *Seja p uma regra tipada, com o lado esquerdo L. Seja G um grafo tipado e $m = (m_V, m_E) : L \to G$ uma ocorrência de L em G, dizemos que m satisfaz a condição de colagem se atende as duas condições seguintes:*

Condição de identificação: *Não há conflito entre exclusão e preservação de itens e não há identificação de itens excluídos, ou seja*

$$Del_V^{p,m} \cap m_V[RegrasPreserv_V^p] = \varnothing,$$
$$Del_E^{p,m} \cap m_E[RegrasPreserv_E^p] = \varnothing,$$

[3] $sourceG \triangleright Del_V^{p,m}$ é o conjunto de todos os pares (e, v) em sourceG onde $v \in Del_V^{p,m}$, isto é, em que o vértice de origem foi eliminado.

$$card^{4}(Del_V^p) = card(RegrasDel_V^p) \text{ e}$$
$$card(Del_E^p) = card(RegrasDel_E^p).$$

Condição Dangling: *Não há arestas que serão excluídas que não são especificadas pela regra, ou seja:*

$$Dangling^{p,m} = \varnothing$$

Definição 8 (Passo de derivação) *Dado um grafo G tipado em T, uma regra $p : L_p \leftarrow K_p \rightarrow R_p$ tipada sobre T e uma ocorrência $m = (m_V, m_E) : L_p \rightarrow G$, que satisfaz a condição de colagem, uma derivação direta de G em H usando p (baseado em m) existe, se e somente se o diagrama abaixo pode ser construído, onde ambos os quadros são pushouts em T-Graph.*

$$\begin{array}{ccccc} L & \xleftarrow{l} & K & \xrightarrow{r} & R \\ \downarrow & (1) & \downarrow & (2) & \downarrow \\ G & \longleftarrow & X & \longrightarrow & H \end{array}$$

A derivação direta de $G \overset{p,m}{\Longrightarrow} H$ ou a aplicação de p para G em m gera um grafo tipado (H, tH, T), com $H = (VertH, EdgeH, sourceH, targetH)$ como segue:

$VertH = (VertG \backslash Del_V^{p,m}) \uplus^{5} New_V^{p,m}$

$EdgeH = (EdgeG \backslash Del_E^{p,m}) \uplus New_E^{p,m}$

$$sourceH = (Del_E^{p,m} \lhd^{6} sourceG) \cup \{e \mapsto m_V(sourceR(e)) | e \in New_E^{p,m} \wedge sourceR(e) \in RegrasPreserv_V^p\} \cup \{e \mapsto sourceR(e) | e \in New_E^{p,m} \wedge sourceR(e) \in New_V^{p,m}\}$$

$$targetH = (Del_E^{p,m} \lhd targetG) \cup \{e \mapsto m_V(targetR(e)) | e \in New_E^{p,m} \wedge targetR(e) \in RegrasPreserv_V^p\} \cup \{e \mapsto targetR(e) | e \in New_E^{p,m} \wedge targetR(e) \in New_V^{p,m}\}$$

$$tH_V = (Del_V^{p,m} \lhd tG_V) \cup \{v \mapsto tR_V(v) | v \in New_V^{p,m}\}$$

$$tH_E = (Del_E^{p,m} \lhd tG_E) \cup \{e \mapsto tR_E(e) | e \in New_E^{p,m}\}$$

Exemplo 6 (Passo de derivação) *G é transformado em H através da aplicação da regra $Pago$, isto é ilustrado na Figura 10. Na aplicação desta regra, as arestas*

[4] card(A) número de elementos do conjunto A.
[5] $\uplus$ representa a união disjunta.
[6] $A \lhd r \equiv^{def} \{(a, b) \in r | a \notin A\}$ é a subtração do domínio.

Cli_Caixa, $Ocup$ e $Atend$ são apagadas do grafo G e para o grafo H as arestas $Cliente\_livre$ e $Livre$ são criadas.

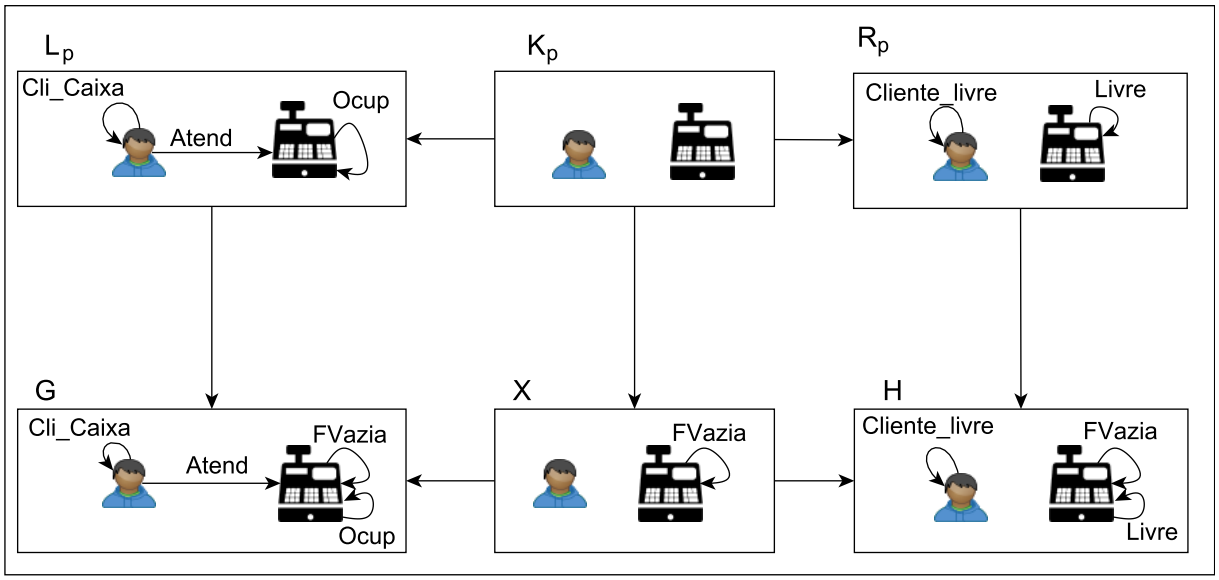


Figura 10: Passo de derivação

3 GRAMÁTICA DE GRAFOS FUZZY

Neste capítulo serão abordados os principais conceitos de gramática de grafos fuzzy (PARASYUK; ERSHOV, 2007) considerando a abordagem categórica de (PARASYUK; YERSHOV, 2006) para formalizar as GGF obtidas através da generalização das GGs. Os conceitos de conjunto fuzzy, morfismo de conjuntos fuzzy, grafo fuzzy, morfismo de grafos fuzzy, regra fuzzy, gramática de grafos fuzzy e a aplicação da regra são apresentados neste capítulo, junto com exemplificações para esses conceitos.

Também é apresentada uma GGF que especifica um sistema de ferrovia simples que consiste de trens e estações. Cada estação tem um ou mais caminhos para outras estações, esses caminhos são os trilhos, onde cada trilho tem um valor de pertinência que significa a velocidade que o trem pode percorrer aquele caminho. Os trens são ligados as estações de origem e destino pelas arestas e essas arestas possuem valor de pertinência que representam o quanto esse trem está longe/perto da sua estação, quanto mais próximo de 1 o valor da pertinência da aresta, mais próximo o trem está da sua estação.

Definição 9 (Conjunto fuzzy) *Um **conjunto fuzzy** é dado por um conjunto parcialmente ordenado e valorado em Q e um par (S, σ) que consiste em S que é um conjunto suporte de (S, σ) e uma função $\sigma : S \to Q$, onde Q é um conjunto verdade. Para cada $s \in S$, o valor $\sigma(s)$ é interpretado como o grau de pertinência de s no conjunto (S, σ) na categoria Fuzzy([0,1]).*

Exemplo 7 (Conjunto fuzzy) *Um conjunto fuzzy é mostrado na Figura 11. A Figura ainda ilustra um conjunto suporte S e um conjunto valorado em Q, que tem como elementos os valores de pertinência t(true), f(false) e $\perp$ (indefinido). O conjunto S é mapeado para Q, onde cada elemento de S tem um valor de pertinência em Q, este mapeamento é dado a seguir: $\sigma(s1) = t$, $\sigma(s2) = v$ e $\sigma(s3) = \perp$.*

Um morfismo de conjuntos fuzzy $f : (S, \sigma) \to (T, \tau)$ entre dois conjuntos fuzzy (S, σ) e (T, τ) valorados em Q, é dado por uma função $f : S \to T$, denominada de função de apoio do morfismo, onde o grau de pertinência de s no conjunto (S, σ) não pode exceder o grau de pertinência de $f(s)$ no conjunto (T, τ).

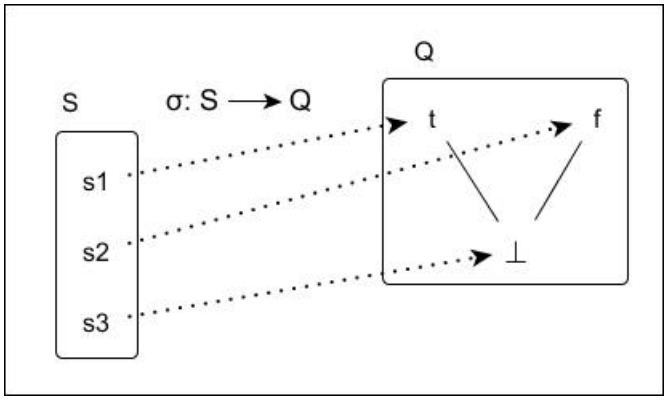


Figura 11: Conjunto Fuzzy

Definição 10 (Morfismo de conjuntos fuzzy) *Sejam* (S, σ) *e* (T, τ) *dois conjuntos fuzzy valorados em* Q*, um* ***morfismo de conjuntos fuzzy*** $f : (S, \sigma) \to (T, \tau)$ *é definido por uma função* $f : S \to T$ *tal que para todo* $s \in S$*,* $\sigma(s) \leq \tau(f(s))$*. A categoria de conjuntos fuzzy e morfismos de conjuntos fuzzy valorados em Q é denotada por* ***Fuzzy(Q)****.*

Seja Q um lattice completo. O pushout de um par de morfismos de conjuntos fuzzy $f : (C, \gamma) \to (A, \sigma)$ e $g : (C, \gamma) \to (B, \tau)$, valorados em Q, é obtido em duas etapas. Primeiro calcula-se, em Set, o pushout das funções suporte $f : C \to A$ e $g : C \to B$ para obter o conjunto P, junto com as funções $j : A \to P$ e $k : B \to P$, como representado no diagrama abaixo.

$$\begin{array}{ccc} C & \xrightarrow{f} & A \\ {\scriptstyle g}\downarrow & PO & \downarrow{\scriptstyle j} \\ B & \xrightarrow[k]{} & P \end{array}$$

Em seguida calcula-se o grau de pertinência ρ, como segue:
$\forall p \in P.\ \rho(p) = supremo(a, b)$ onde $k(b) = p \wedge j(a) = p$.

Exemplo 8 (Morfismo de conjuntos fuzzy) *A Figura 12 ilustra dois conjuntos fuzzy* S *e* T*, com o morfismo de conjutos fuzzy* $f : S \to T$*. O conjunto* Q *é constituído por três elementos, sendo eles* t*,* v *e* $\perp$*. Os elementos* $s1$ *e* $s2$ *do conjunto* S *são mapeados paras os elementos* $t1$ *e* $t2$ *no conjunto* T*, respectivamente,* $s1$ *e* $t1$ *são mapeados para o valor* t *no conjunto valorado* Q*, assim como* $s2$ *e* $t2$ *são mapeados para o valor* f*. No conjunto fuzzy* S*,* $s3$ *é mapeado para* $t2$ *no conjunto fuzzy* T*, isso é possível pois o valor de* $s3$ *no conjunto valorado* Q *é* $\perp$*, e esse valor pode ser* t *ou* f*,* $t2$ *é mapeado para o valor* f*, sendo assim a condição é respeitada.*

Grafos fuzzy são constituídos por conjuntos fuzzy de vértices e arestas valorados no intervalo $[0, 1]$.

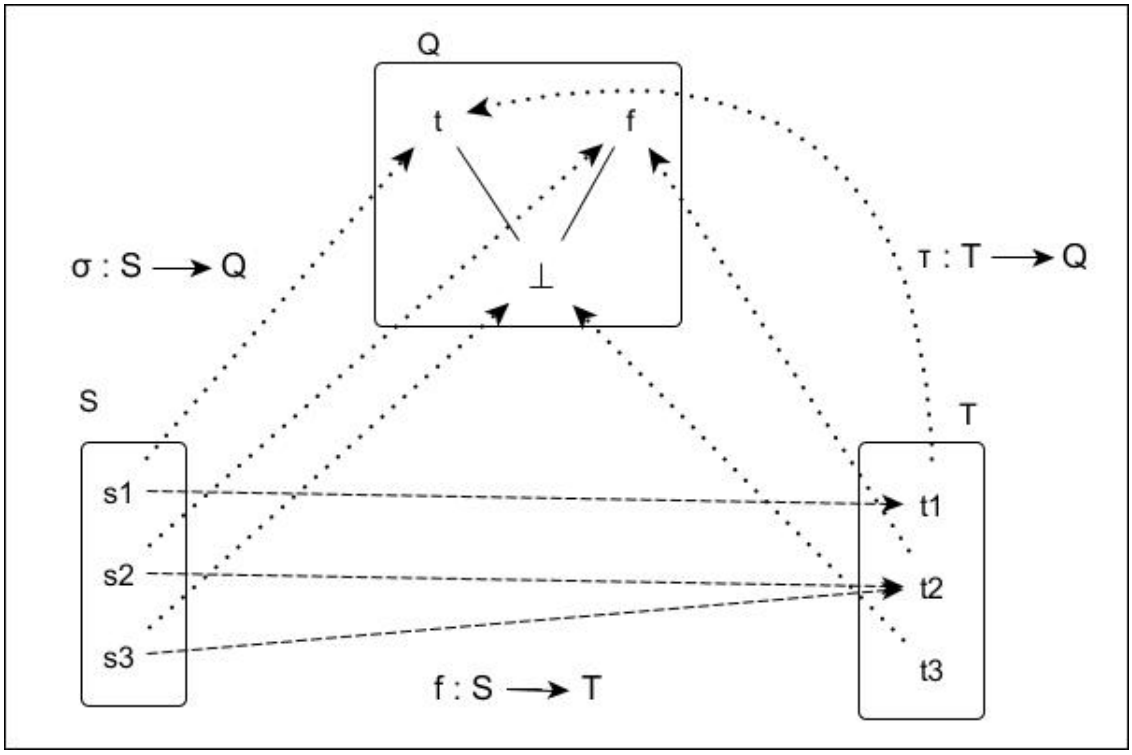


Figura 12: Morfismo de conjuntos Fuzzy

Definição 11 (Grafo fuzzy) *Um **grafo fuzzy** $G = ((V_G, \sigma_{V_G}), (A_G, \sigma_{A_G}), s_G, t_G)$ é uma tupla composta pelos conjuntos fuzzy de vértices e arestas de (V_G, σ_{V_G}) e (A_G, σ_{A_G}), onde o mapeamento verdade para vértices e arestas é $\sigma_{V_G} : V_G \to [0, 1]$ e $\sigma_{A_G} : A_G \to [0, 1]$, respectivamente. O intervalo $[0, 1]$ é o conjunto de valores verdade para vértices e arestas. Os mapeamentos $s_G, t_G : A_G \to V_G$ são chamados de mapeamento de origem e destino das arestas, respectivamente.*

Exemplo 9 (Grafo fuzzy) *Um grafo fuzzy é ilustrado na Figura 13, onde os vértices* 🚂 *e* 🏛 *têm seus valores verdade definidos por $\sigma_{V_G}($🚂$) = 0,5$ e $\sigma_{V_G}($🏛$) = 0,2$. A aresta e tem sua origem e seu destino definidos pelas funções $s_G(e) =$ 🚂 e $t_G(e) =$ 🏛 respectivamente. O valor verdade de e é dado por $\sigma_{A_G} = 0,9$.*

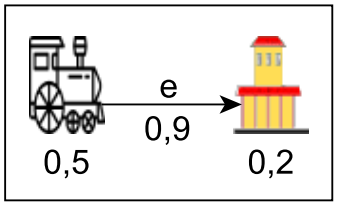


Figura 13: Grafo Fuzzy G

Se as estruturas de dois grafos fuzzy forem compatíveis, então significa que eles podem ser relacionados, por meio de um morfismo de grafos fuzzy, onde os vértices

e as arestas de um grafo fuzzy são mapeados em vértices e arestas de outro grafo fuzzy, respectivamente. Esse mapeamento deve respeitar a origem e o destino das arestas e os valores de pertinência. Deste modo, um morfismo de grafos fuzzy pode ser definido por duas funções de conjuntos fuzzy para mapear arestas e vértices, os quais devem preservar a origem e destino das arestas.

Definição 12 (Morfismo de grafos fuzzy) *Sejam G e H dois grafos fuzzy. Um* ***morfismo de grafos fuzzy*** $f : G \to H$ *é um par de morfismos de conjuntos fuzzy* $(f_A : A_G \to A_H, f_V : V_G \to V_H)$ *tal que,* $f_V \circ s_G = s_H \circ f_A \wedge f_V \circ t_G = t_H \circ f_A$.

Exemplo 10 (Morfismo de grafos fuzzy) *Um morfismo de grafos fuzzy de G em H é ilustrado pela Figura 14, onde os vértices*  *e*  *com valores de pertinência* 1 *são mapeados do grafo G para o grafo H. As arestas com os valores* $0,6$ *,* 0 *e* $0,2$ *são mapeadas para as arestas com os valores de pertinência* $0,8$ *,* $0,2$ *e* $0,2$ *respectivamente e elas respeitam a condição de origem, destino e de pertinência.*

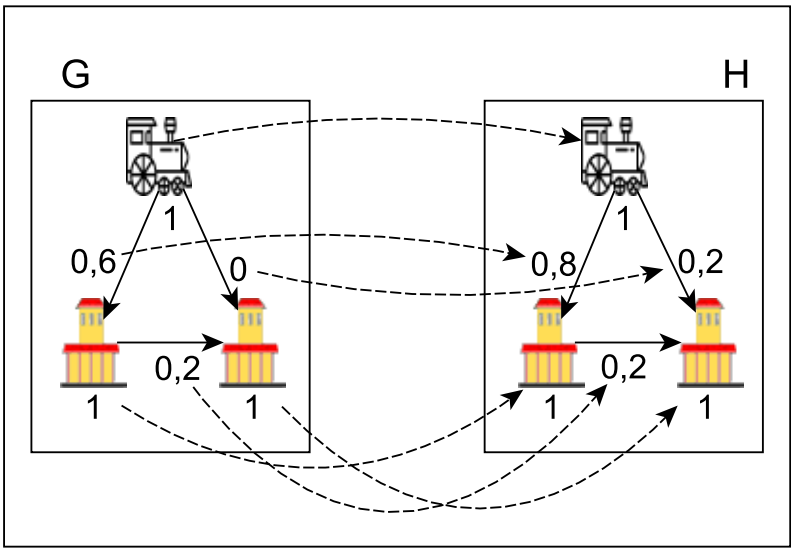
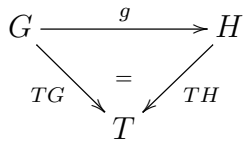


Figura 14: Morfismo de Grafos Fuzzy

Para realizar este trabalho foi necessário estender as definições da gramática de grafos fuzzy para ser possível usar grafos fuzzy tipados, neste trabalho a tipagem de grafos fuzzy é feita através do uso de grafo tipo fuzzy. Um grafo tipo fuzzy serve para caracterizar todos os tipos e limitar os valores de pertinência dos vértices e arestas permitidos no sistema. Todos os grafos fuzzy da gramática estão restritos a esses tipos e valores. Esta restrição é definida através de morfismos de grafos fuzzy mapeando cada grafo fuzzy no grafo tipo fuzzy.

Definição 13 (Grafo fuzzy tipado, morfismo de grafos fuzzy tipados) *Um* ***grafo fuzzy tipado*** G^T *é definido por uma tupla* $G^T = (G, tG, T)$*, onde* G *é um grafo fuzzy,* T

é o grafo tipo fuzzy e $tG : G \to T$, é um morfismo total de grafos fuzzy. Um ***morfismo de grafos fuzzy tipados*** *$g^T : G^T \to H^T$ entre os grafos fuzzy tipados G^T e H^T, consiste em um morfismo de grafos fuzzy $g : G \to H$, onde o diagrama abaixo comuta:*



A categoria dos grafos fuzzy e morfismos de grafos fuzzy tipados em T é denotada por ***T-FGraph****. Pushouts em T-FGraph são construídos a partir do pushout dos componentes dos morfismos em Fuzzy([0,1]) (categoria dos conjuntos fuzzy e morfismos de conjuntos fuzzy valorados no intervalo [0,1]).*

Exemplo 11 (Grafo fuzzy tipado, morfismo de grafos fuzzy tipados) *A Figura 15 ilustra um grafo fuzzy tipado G^T, onde o grafo fuzzy G tem seus tipos restringidos através do morfismo de grafos fuzzy tG no grafo tipo fuzzy T. O tipo no grafo fuzzy tipado G^T não restringe os valores de pertinência e por isso esses valores no grafo tipo fuzzy T são todos 1 (o maior valor do intervalo). A Figura 16 ilustra um exemplo de morfismo de grafos fuzzy tipados, $g : G \to H$ é o morfismo de grafos fuzzy tipados do grafo fuzzy G no grafo fuzzy H, onde f_V e f_A são os mapeamentos para os vértices e arestas, respectivamente. tG e tH são morfismos para tipar os grafos G e H, respectivamente, no grafo tipo fuzzy T.*

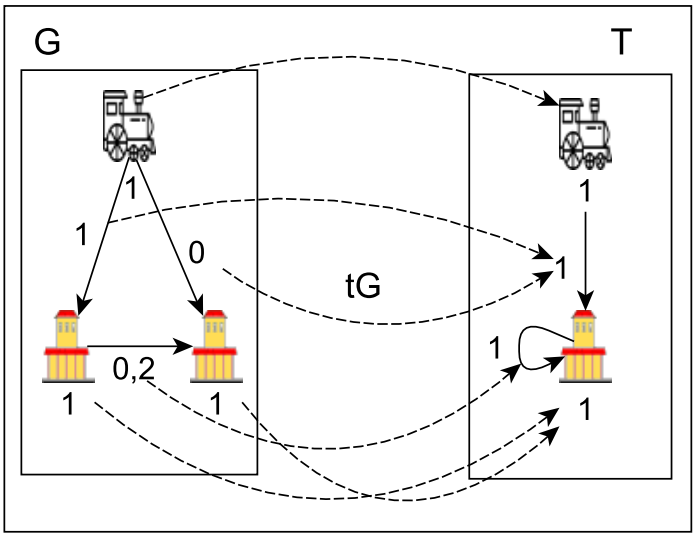


Figura 15: Grafo fuzzy tipado G^T

Em uma GGF, os estados do sistema são representados por um conjunto de grafos fuzzy e as regras descrevem as possíveis mudanças de estado. A regra indica os elementos que devem estar presente no grafo para que ela possa ser aplicada e assim acontecerem algumas alterações, onde alguns elementos são modificados, criados

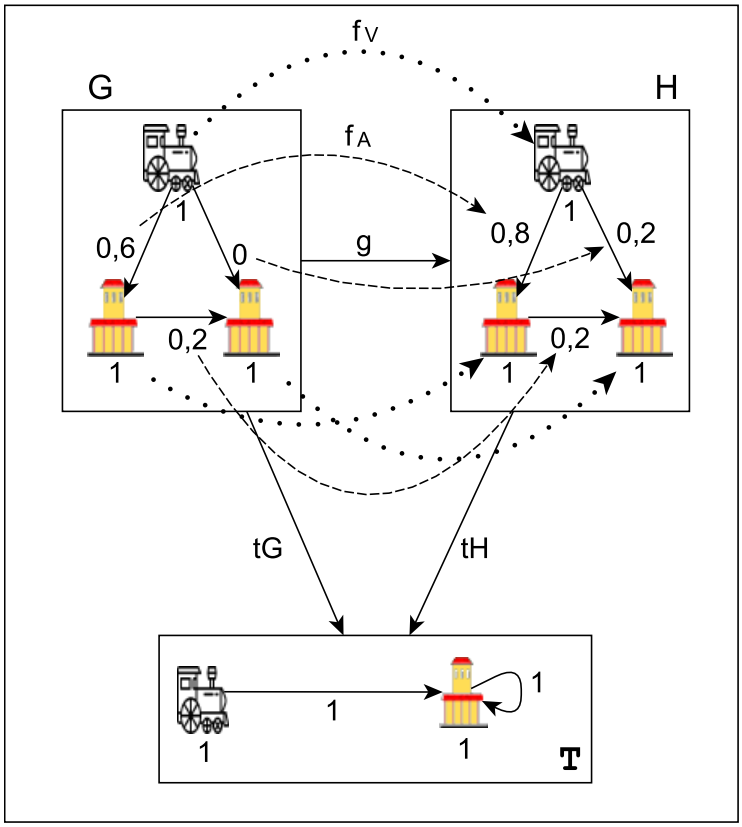


Figura 16: Morfismo de grafos fuzzy tipados g^T

ou até mesmo deletados, em uma GGF a regra é dada por três grafos fuzzy e dois morfismo de grafos fuzzy entre eles.

Definição 14 (Regras fuzzy) *Uma **regra fuzzy** é uma tupla $p : L \xleftarrow{l} K \xrightarrow{r} R$, onde L é chamado de lado esquerdo, R de lado direito, K de interface, l e r são morfismos de grafos fuzzy da regra fuzzy. Os grafos fuzzy L, R, K são grafos tipados fuzzy sobre T.*

Exemplo 12 (Regras fuzzy) *A Figura 17 mostra uma regra fuzzy p, onde a regra é dada pelos grafos fuzzy de lado esquerdo L, de lado direito R e de interface K. A regra pode ser aplicada em um estado contendo o grafo fuzzy L (que contém os vértices*   *e*  *ambos com valores de pertinência 1 e três arestas com valores 0, 1 e $0,2$), resultando em um grafo onde a aresta com valor 1 é deletada e três vértices e duas arestas são preservados (grafo fuzzy K). A aresta com valor 0,2 é mantida com o mesmo valor, já a aresta que tinha o valor 0, com a aplicação da regra passou a ter um novo valor, essa aresta passou a ter o valor 0,2. A aresta com o valor 1 é excluída na aplicação da regra, no seu lugar é criada uma nova, que tem a mesma origem, destino, porém o seu valor de pertinência é 0,8 (grafo fuzzy R). Essa aresta é deletada pois não é permitido diminuir o valor de pertinência para os vértices e arestas, os valores devem ser conservados ou aumentados dentro do intervalo permitido, a única maneira*

de diminuir o valor é deletar a aresta ou vértice, e após criar uma aresta ou vértice com o mesmo tipo e com o valor novo.

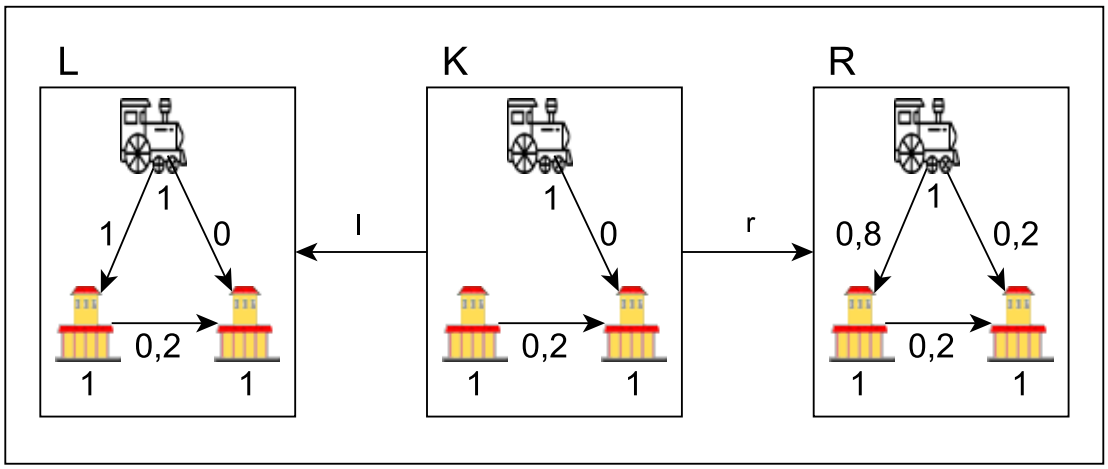


Figura 17: Regra fuzzy

Gramática de Grafos Fuzzy se diferenciam das GGs por suas arestas e vértices possuírem graus de pertinência. As GGFs de forma análoga às GGs são constituídas por um grafo tipo fuzzy, que representa todos os tipos e pesos dos vértices e arestas permitidos no sistema, um grafo inicial fuzzy, que simboliza o estado inicial do sistema, e um conjunto de regras que expõem as possíveis alterações de estado que podem ocorrer no sistema.

Definição 15 (Gramática de Grafos Fuzzy) *Uma **gramática de grafos fuzzy** é definida por uma tupla $GGF = (T, I, P)$, onde T é o grafo tipo fuzzy, I é o grafo inicial fuzzy tipado em T, P é o conjunto de regras fuzzy tipadas em T.*

Exemplo 13 (Gramática de Grafos Fuzzy) *Este exemplo descreve uma GGF que especifica um sistema de ferrovia bem simples. O grafo tipo fuzzy **T** é representado na Figura 18 e o grafo inicial fuzzy **I** é representados na Figura 19. A ferrovia consiste de (trens) e de (estações) e dois tipos de arestas (Distância_estação e trilho) (Veja **T**), cada estação está ligada a uma ou mais estações. Cada trem tem duas arestas que o liga às estações, essas arestas representam o quanto esse trem esta próximo ou não da estação, quanto mais próximo de 1 é o grau de pertinência, mais próximo da estação está o trem. A aresta (trilho) entre as estações representa a velocidade e a direção em que os trens podem percorrer aquele espaço. Devido os graus de pertinência dos vértices possuírem sempre o valor 1 e no exemplo proposto não sofrerem alterações, então decidiu-se omiti-los daqui em diante. O grafo inicial fuzzy descreve um estado onde tem-se dois trens. O primeiro (trem1) está na estação A e quer se dirigir à estação B (aresta com grau de pertinência 1 para o vértice A e com 0 para*

o vértice B). O segundo  *(trem2) esta entre o caminho das estações D e F (aresta com grau de pertinência 0,8 para o vértice D e com 0,2 para o vértice F). A Figura 20 ilustra as regras da gramática ferrovia as quais definem o comportamento dos trens. Um trem começa a se movimentar entre as estações (regras **r1**, **r5** e **r8**) e assim vai diminuindo a pertinência da sua aresta com a estação de origem e aumentando a pertinência da sua aresta com a estação de destino (regra **r2**) até chegar na estação de destino (regras **r3**, **r6** e **r9**), após isso o trem vai para sua próxima estação de destino (regras **r4** e **r7**).*

São necessárias várias regras para especificar esse sistema, visto que para cada grau de pertinência é necessário uma ou mais regras para diminuir/aumentar a distância em que o trem está próximo ou não da estação, logo, quanto maior o grau de pertinência entre as estações (velocidade que o trem pode percorrer) menor é o número de regras para aquela pertinência, ou seja, a pertinência entre as estações é sempre somada a aresta (até chegar ao valor 1) da estação de destino do trem e sempre diminuída da aresta (até chegar ao valor 0) da estação de origem do trem, isto acontece em cada aplicação de regra. Por exemplo, a regra $r1$ tem o valor de pertinência $0,2$ entre as estações, por isso na aplicação da regra a aresta para a estação de destino passou o seu valor de pertinência de 0 para $0,2$ e a aresta com a estação de origem passou o valor de 1 para $0,8$. Porém para diminuir o valor de pertinência desta aresta foi necessário excluir a aresta de valor 1 e criar uma nova aresta de valor $0,8$ com o mesmo tipo da excluída. O restante das regras da GGF é omitida, e sua construção é análoga as regras apresentadas acima, apenas variando o grau de pertinência das arestas, onde os valores vão variar entre o intervalo 0 e 1. O grafo fuzzy K também é omitido, mas pode ser construído pela interseção dos grafos L e R.

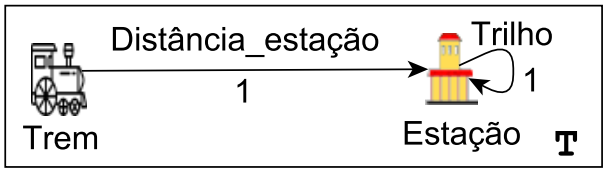


Figura 18: grafo tipo fuzzy da gramática de grafos fuzzy ferrovia

A aplicação de uma regra em um grafo fuzzy G é possível se existe uma ocorrência do lado esquerdo da regra no grafo G. Para caracterizar o resultado da aplicação de uma regra sobre um grafo fuzzy, são usados pushouts na categoria T-FGraph.

Definição 16 (Aplicação da regra) *Um grafo fuzzy G pode ser transformado em um grafo fuzzy H pela **aplicação da regra** $p : L \leftarrow K \rightarrow R$ com a ocorrência m se os pushouts abaixo podem ser construídos em T-FGraph.*

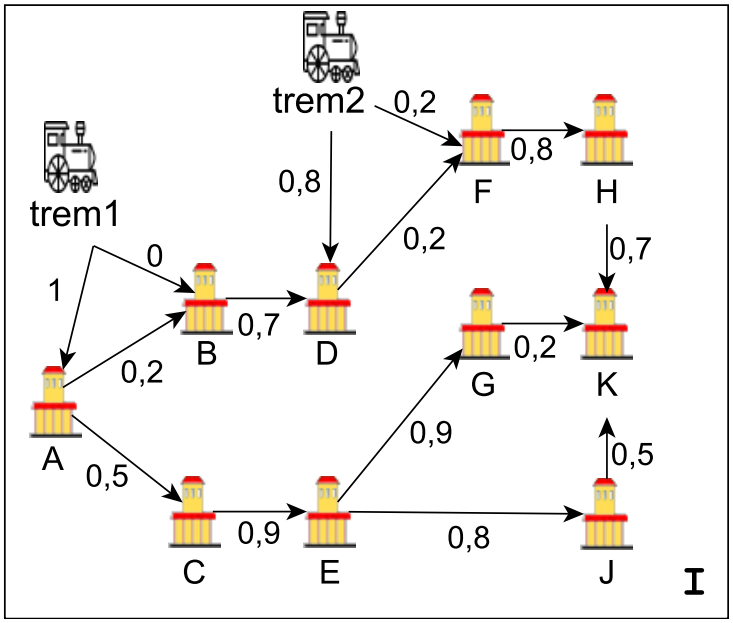


Figura 19: grafo inicial fuzzy da gramática de grafos fuzzy ferrovia

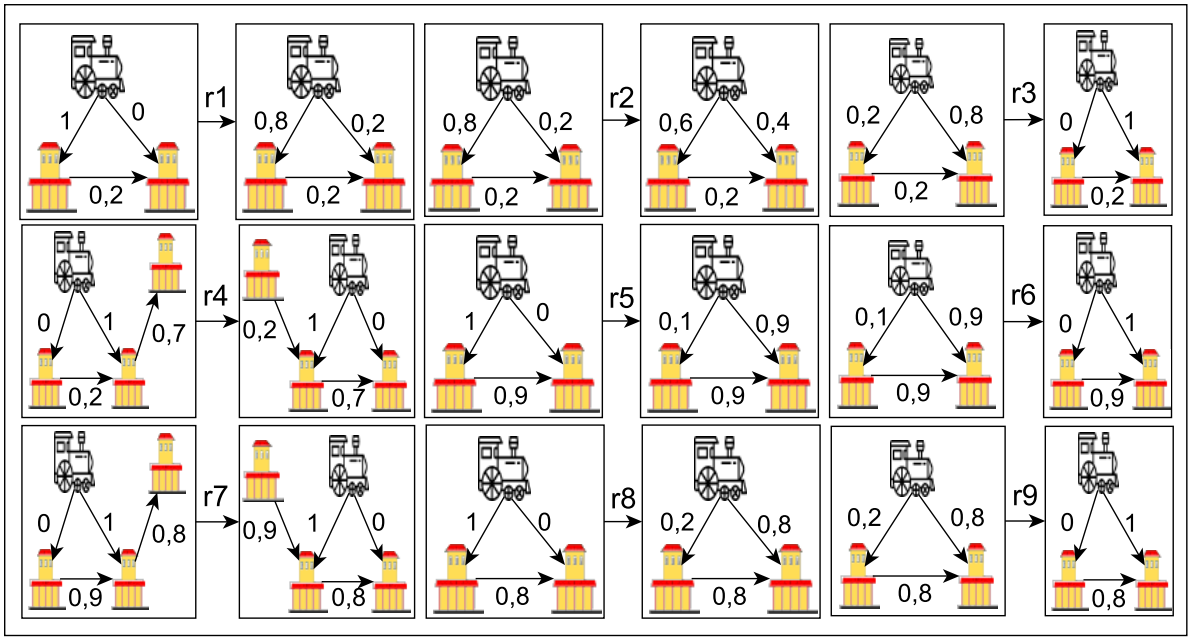


Figura 20: Regras da gramática de grafos fuzzy ferrovia

$$\begin{array}{ccccc} p: L & \longleftarrow & K & \longrightarrow & R \\ m \downarrow & & \downarrow & & \downarrow \\ G & \longleftarrow & C & \longrightarrow & H \end{array}$$

Exemplo 14 (Aplicação da regra) *Na Figura 21, G é transformado em H através da aplicação da regra p. A ocorrência m no grafo fuzzy G é marcado pelo um retângulo com linhas tracejadas.*

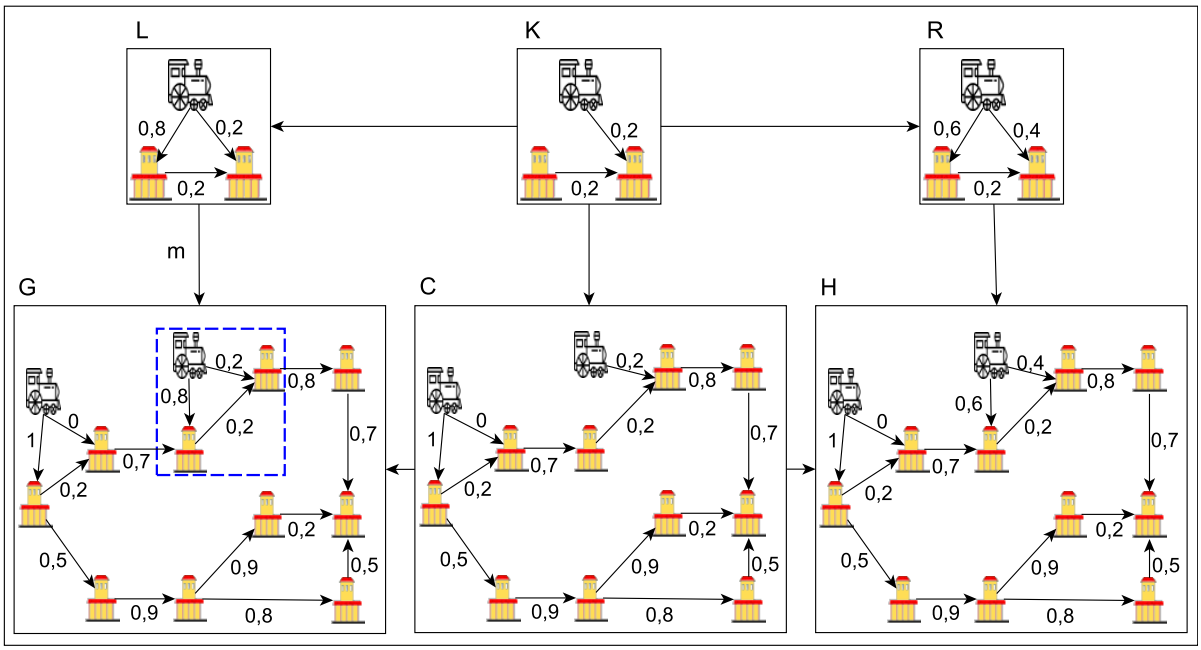


Figura 21: Aplicação da regra p

4 EVENT-B

Event-B (DEPLOY, 2016) foi criado por Jean-Raymond Abrial (ABRIAL, 1996) e consiste em um formalismo baseado em estados próximo ao método B (ABRIAL, 2010). Possibilita o desenvolvimento das principais partes de vida de um software e tem sido aplicada a diversas áreas, como em banco de dados (WANG; WAHLS, 2014), aviação (HONG; LILI, 2013), aplicativos para a medicina (WAHLS, 2016) e computação em nuvem (ZHANG et al., 2015).

A linguagem Event-B é executada na plataforma Rodin, que foi proposta por Michael Butler e Stefan Hallerstede (BUTLER; HALLERSTEDE, 2007) para facilitar a modelagem formal em Event-B. O Rodin foi desenvolvido na plataforma Eclipse e oferece suporte aos modelos no Event-B para aplicar as técnicas de modelagem formal e de verificação de sistemas de software. Um extenso suporte de ferramentas através da plataforma Rodin torna o Event-B especialmente atraente (DEPLOY, 2016).

Um modelo em Event-B consiste em duas partes: uma parte estática chamada contexto e uma parte dinâmica chamada máquina. No contexto são definidas as constantes, os axiomas e os conjuntos. Na máquina são definidas as invariantes, as variáveis e os eventos. Event-B é baseado na lógica de primeira ordem e na teoria de conjuntos.

Definição 17 (Modelo Event-B, evento) *Um **modelo de event-B** é definido por uma tupla EB = (c, s, P, v, I, R_I, E), onde c são Constants e s são Sets conhecidos no modelo, v são as variables do modelo, P(c,s) é um conjunto de axioms que restringem c e s, I(c,s,v) é um modelo invariant limitando os possíveis estados de v, s.t. $\exists$c,s,v. P(c,s) $\bigwedge$ I(c,s,v). P e I caracterizam um conjunto não vazio de estados, R_I(c,s,v') é o evento de inicialização onde são definidos os valores iniciais para as variables do modelo, e E é o conjunto de eventos do modelo. Dados os estados v e v', um **evento** é uma tupla e = (H,S) onde H(c,s,v) é a guarda e S(c,s,v,v') é o predicado antes e depois que define uma relação entre um estado atual e o próximo. Também pode-se denotar uma guarda por H(v), e um predicado antes e depois por S(v,v') e o evento de inicialização por R_I(v').*

4.1 Contextos em Event-B

Um contexto define a forma de como um determinado modelo vai ser parametrizado para logo após ser instanciado. As declarações de conjuntos, constantes, axiomas são feitos aqui. No contexto estão presentes as partes constantes do sistema, compostas pelas cláusulas que serão descritas abaixo:

- **SETS**: Declarações dos nomes dos conjuntos utilizados pelos usuários, estes podem ser adiados (deferred set) ou enumerados. Um conjunto é adiado quando na sua declaração é fornecido apenas seu nome, assumindo que o conjunto é não vazio e finito, onde seus elementos serão fornecidos apenas no módulo de implementação. Por outro lado, quando um conjunto é declarado enumerado são fornecidos explicitamente os seus valores, que necessitam ser distintos entre si. Com um conjunto definido, esse pode ser utilizado na tipagem das variáveis e constantes.
- **CONSTANTS**: Lista os nomes das constantes usadas na máquina. Elementos de conjuntos enumerados são declarados como constantes, os tipos de constantes devem ser definidos como axiomas.
- **AXIOMS**: Axiomas são referentes às constantes e aos conjuntos que foram especificados nas cláusulas anteriores. Cada constante têm seu tipo declarado. Os elementos dos conjuntos enumerados são declarados nesta cláusula. Os axiomas que relacionam as constantes e os conjuntos também são declarados nesta cláusula.

4.2 Máquinas em Event-B

A máquina em Event-B é composta por uma parte dinâmica e uma parte estática, nela são descritas as especificações dos componentes. Na parte estática, definem-se as estruturas que vão formar o estado do sistema que será representado. Na parte dinâmica são descritos os mecanismos que serão responsáveis pelas mudanças de estados. Os principais componentes da máquina são:

- **VARIABLES**: Esta cláusula contém os nomes das variáveis de estado da máquina. Todas as variáveis devem ter um tipo, que é definido por uma invariante.
- **INVARIANT**: É um predicado que serve para especificar os tipos das variáveis e algumas outras restrições. Geralmente, as propriedades dinâmicas do sistema são expressas por este elemento. As invariantes podem ser utilizadas para descrever as propriedades desejadas dos estados alcançáveis de um sistema. Em um sistema modelado em Event-B, as invariantes são as propriedades a serem provadas.

- **EVENTS**: Uma transição é representada por um evento, o qual é composto de guardas e ações. As guardas também são usadas para especificar as condições que devem ser mantidas em um estado para o evento ser habilitado, enquanto as ações representam a maneira como as variáveis da máquina estão sendo modificadas. O evento de inicialização não tem guardas e então é atribuído um valor para cada variável da máquina.

As máquinas de Event-B fazem referência a contextos através de uma cláusula chamada sees. Nesta cláusula pode-se também adicionar e ver qualquer outra máquina da qual os conjuntos e as constantes podem ser lidas, mas não modificadas. Nesta cláusula não é permitido nenhum acesso a variáveis e operações.

Em Event-B, as invariantes podem ser divididas em várias invariantes menores conectadas por conjunções, facilitando assim, a leitura e a validação dessas partes separadas. Os requisitos funcionais do sistema são estabelecidos por um predicado na cláusula INVARIANT juntamente com um conjunto de operação da máquina. É este componente que serve como apoio para a verificação da máquina para os elementos relacionados à transição de estado, o que acontece quando se inicializam as variáveis na cláusula INITIALISATION. Existe, também uma cláusula chamada THEOREMS nas máquinas de Event-B. Nessa cláusula os teoremas são declarados e provados através do uso das invariantes. O uso desta cláusula separa as provas intermediárias das provas de fato desejadas, isso facilita a leitura.

Na parte dinâmica são definidos os eventos que a máquina deve executar. Esses eventos dependem de um conjunto de substituições, ou seja, atribuição de valores a variáveis da máquina, e caso seja essencial, ao retorno de um parâmetro de saída. A máquina possui um evento que é pré-definido para a inicialização das variáveis, visto que estas não dispõem de parâmetros e nem pré-condições. Para uma máquina ser validada, as suas invariantes devem ser verificadas quando um evento é executada, logo, assumimos que a condição do estado no sistema antes da execução do evento respeita a invariante, assim deve-se demonstrar que caso a pre-condição seja satisfeita, a aplicação do evento não vai conduzir ao estado onde a invariante seja invalida. Os eventos em Event-B são compostos por substituições sem pre-condições ou parâmetros de entrada e saída. Um evento é valido, quando suas guardas são satisfeitas, então ele é capaz de ser disparado a partir de qualquer estado do sistema, respeitando a invariante especificada.

A seguir será apresentado uma tradução de GGs para o Event-B, onde o grafo tipo e o grafo do lado esquerdo das regras são determinados no contexto e o grafo inicial e a aplicação das regras são definidos na máquina do Event-B.

4.3 Tradução da Gramática de Grafos no Event-B

Event-B é um método formal que permite a análise e a modelagem de sistemas, utilizando a teoria dos conjuntos e a lógica de primeira ordem como notação de modelagem. Ele tem sido usado com sucesso em várias aplicações, tem suporte de ferramentas disponíveis tanto para especificação quanto análise de modelos. De acordo com Cavalheiro (CAVALHEIRO, 2010) o comportamento de um modelo de Event-B é semelhante a uma GG, onde existe um conceito de estado, que é dado por um conjunto de variáveis do Event-B, e por um grafo da GG, a um passo que é definido por uma operação atômica no estado atual (um evento que atualiza as variáveis em Event-B e uma aplicação da regra em uma GG).

Em cada etapa são mantidas as propriedades do estado, em Event-B, essas propriedades são apresentadas como invariantes. Em uma GGs, as propriedades devem ser garantidas para serem preservadas, e elas estão relacionadas com a estrutura do grafo, onde apenas grafos bem formados podem ser gerados. Neste trabalho são apresentadas as definições da tradução da gramática de grafos para o Event-B, definidas por (CAVALHEIRO, 2010). A seguir são definidas algumas funções auxiliares que são usadas no restante das definições deste trabalho.

- $lista$: dado um conjunto, retorna uma string representando todos os seus elementos separados por espaços em branco, ou seja, $lista(S) = s1\ s2\ ...\ sn$, para todo $sn \in S$.

- $elementos$: dado um conjunto (finito e não vazio), retorna uma sequência que representa todos os elementos deste conjunto, cada um dentro de chaves, ou seja, $elementos(S) = \{s1\}, \{s2\}, ..., \{sn\}$, para todo $sn \in S$.

- $conjunto$: dado um conjunto (finito e não vazio), retorna uma sequência que representa todos os seus elementos dentro de chaves, ou seja, $conjunto(S) = \{s1, s2, ..., sn\}$, para todo $sn \in S$.

- $lista\_prefixada$: dados um conjunto S e uma string s, retorna uma string representando todos os elementos de S prefixado com s separados por espaços. Ou seja, $lista\_prefixada(S, s) = \mathtt{s\_}s1\ \mathtt{s\_}s2\ ...\ \mathtt{s\_}sn$, para todo $sn \in S$.

- $conjunto\_prefixado$: dados um conjunto (finito e não vazio) S e uma string s, retorna uma string representando todos os elementos de S prefixados com s dentro de chaves. Ou seja, $conjunto\_prefixado(S, \mathtt{s}) = \{\mathtt{s\_}s1,...,\mathtt{s\_}sn\}$, para todo $sn \in S$.

Observe que, uma vez que as funções são conjuntos de pares, as construções acima também se aplicam a funções, onde a sintaxe usada para denotar cada par $(a, f(a))$ é $a \mapsto f(a)$.

O grafo tipo e os grafos das regras são determinados no contexto do Event-B, onde temos as cláusulas conjuntos, constantes e axiomas, cada uma representando algum componente do grafo. Os nomes dos conjuntos dos vértices e arestas do grafo tipo e das regras são definidos como conjuntos. Os nomes dos vértices e arestas, os nomes das funções de origem e destino das arestas e ainda os nomes das funções de tipagem são definidos como as constantes. Por fim, os axiomas definem os tipos dos conjuntos e das constantes. Nas próximas definições, tudo que estiver em azul é traduzido para o Event-B literalmente e tudo que estiver em preto deve ser calculado.

Definição 18 (Tradução do grafo tipo) *Dado um grafo tipo* $T = (VertT, EdgeT, sourceT, targetT)$ *definimos sua tradução* $\mathcal{T}^T$ *como os elementos do contexto do event-B a seguir:*

CONTEXT *ctxGG*

SETS

VertT

EdgeT

CONSTANTS

$lista(VertT)$

$lista(EdgeT)$

sourceT

targetT

AXIOMS

axm_VertT : partition(VertT,$elementos(VertT)$)

axm_EdgeT : partition(EdgeT,$elementos(EdgeT)$)

axm_srcTtype : sourceT $\in$ EdgeT $\rightarrow$ VertT

axm_srcTdef : partition(sourceT,$elementos(sourceT)$)

axm_tgtTtype : targetT $\in$ EdgeT $\rightarrow$ VertT

axm_tgtTdef : partition(targetT,$elementos(targetT)$)

END

Exemplo 15 (Tradução do grafo tipo) *A tradução do grafo tipo da Figura 6 para o contexto do Event-B é ilustrada na Figura 22. Todos os vértices e arestas do grafo tipo são representados como constantes, da mesma forma que as funções de origem e destino das arestas. Os conjuntos de vértices e arestas são nomeados nas clausulas Sets e definidos pelos axiomas. Por exemplo o axioma* axm_vertT *determina que o conjunto de vértices* $\{Cliente, Caixa\}$*; o axioma* axm_edgeT *define que as arestas são* $\{Compras, Cliente\_livre, Fila, Cli\_Caixa, Prox, Ultim, Prim, Atend, Livre, Ocup, FVazia, Fila\_caixa\}$*. Já os axiomas* $axm\_srcTdef$ *e* axm_tgTdef *definem a origem*

e destino para cada aresta, respectivamente. Assim pode-se observar que a aresta $Compras$ *tem origem e destino no vértice* $Cliente$*.*

CONTEXT ctxGG
SETS

$vertT$
$edgeT$

CONSTANTS

$Caixa\ \ Cliente\ \ Cliente\ \ Compras\ \ Cliente\_livre\ \ Fila\ \ Cli\_Caixa\ \ Prox\ \ Ultim\ \ Prim$
$Atend\ \ Livre\ \ Ocup\ \ FVazia\ \ Fila\_caixa\ \ sourceT\ \ targetT$

AXIOMS

axm_vertT : $partition(vertT, \{Cliente\}, \{Caixa\})$
axm_edgeT : $partition(edgeT, \{Compras\}, \{Cliente\_livre\}, \{Fila\}, \{Cli\_Caixa\}, \{Prox\}, \{Ultim\}, \{Prim\}, \{Atend\}, \{Livre\}, \{Ocup\}, \{FVazia\}, \{Fila\_caixa\})$
axm_srcTtype : $sourceT \in edgeT \rightarrow vertT$
axm_srcTdef : $partition(sourceT, \{Compras \mapsto Cliente\}, \{Cliente\_livre \mapsto Cliente\}, \{Cli\_Caixa \mapsto Cliente\}, \{Fila \mapsto Cliente\}, \{Prox \mapsto Cliente\}, \{Ultim \mapsto Caixa\}, \{Prim \mapsto Caixa\}, \{Atend \mapsto Cliente\}, \{Livre \mapsto Caixa\}, \{Ocup \mapsto Caixa\}, \{Fila\_caixa \mapsto Caixa\}, \{FVazia \mapsto Caixa\})$
axm_tgtTtype : $targetT \in edgeT \rightarrow vertT$
axm_tgtTdef : $partition(targetT, \{Compras \mapsto Cliente\}, \{Cliente\_livre \mapsto Cliente\}, \{Cli\_Caixa \mapsto Cliente\}, \{Fila \mapsto Cliente\}, \{Prox \mapsto Cliente\}, \{Ultim \mapsto Cliente\}, \{Prim \mapsto Cliente\}, \{Atend \mapsto Caixa\}, \{Livre \mapsto Caixa\}, \{Ocup \mapsto Caixa\}, \{Fila\_caixa \mapsto Caixa\}, \{FVazia \mapsto Caixa\})$

END

Figura 22: Grafo tipo da gramática Supermercado no modelo Event-B

O grafo inicial é especificado na máquina do Event-B. Cada item do grafo inicial é descrito como uma variável e as variáveis são definidas através das invariantes. O evento de inicialização representa o grafo inicial, é neste evento que as variáveis são iniciadas e cada uma recebe um valor de atribuição (número natural).

Os estados representam os grafos e as instâncias de arestas e vértices são descritos por números naturais. Não existe a necessidade de haver números distintos para as arestas e vértices. Um grafo pode ter um vértice e uma aresta a mesma identidade (número), pois o que diferenciam esses elementos são os mapeamentos de tipagem, onde um vértice vai ser mapeado para um tipo de vértice e uma aresta para um tipo de aresta.

Definição 19 (Tradução do grafo Inicial) *Dado um grafo inicial* $G0^T = (G0, tG0, T)$ *com* $VertG0 \subseteq \mathbb{N}$ *e* $EdgeG0 \subseteq \mathbb{N}$*, sua tradução* $\mathcal{T}^{G0}$ *para o event-B é definida pelo evento de inicialização definido abaixo:*

MACHINE *mchGG*

SEES *ctxGG*

VARIABLES

VertG

EdgeG

sourceG

targetG

tG_V

tG_E

INVARIANTS

inv_VertG_type : $\texttt{VertG} \in \mathbb{P}(\mathbb{N})$

inv_EdgeG_type : $\texttt{EdgeG} \in \mathbb{P}(\mathbb{N})$

inv_sourceG_type : $\texttt{sourceG} \in \texttt{EdgeG} \rightarrow \texttt{VertG}$

inv_targetG_type : $\texttt{targetG} \in \texttt{EdgeG} \rightarrow \texttt{VertG}$

inv_tVG_type : $\texttt{tG\_V} \in \texttt{VertG} \rightarrow \texttt{VertT}$

inv_tEG_type : $\texttt{tG\_E} \in \texttt{EdgeG} \rightarrow \texttt{EdgeT}$

EVENTS

Initialisation

begin

act_VertG : $\texttt{VertG} := conjunto(VertG0)$

act_EdgeG : $\texttt{EdgeG} := conjunto(EdgeG0)$

act_srcG : $\texttt{sourceG} := conjunto(sourceG0)$

act_tgtG : $\texttt{targetG} := conjunto(targetG0)$

act_tG_V : $\texttt{tG\_V} := conjunto(tG0\_V)$

act_tG_E : $\texttt{tG\_E} := conjunto(tG0\_E)$

end

END

Exemplo 16 (Tradução do grafo inicial) *A tradução do grafo inicial da Figura 7 para o modelo event-B é ilustrada na Figura 23. As invariantes são usadas para definir o tipo de variáveis. Por exemplo,* $tG\_V \in vertG \rightarrow vertT$ *e* $tG\_E \in edgeG \rightarrow edgeT$ *são funções totais de vertG para vertT e edgeG para edgeT, respectivamente.*

No evento Initialisation os vértices e arestas são definidos por números naturais nas ações act_vertG *e* act_edgeG*. No grafo inicial existem quatro vértices* $\{1, 2, 3, 4\}$*, onde os vértices 1 e 2 são mapeados para o tipo Cliente e os vértices 3 e 4 são mapeados para o tipo Caixa através da ação* act_tG_V*. As arestas são tipadas como segue: aresta 1 é do tipo* $Fila$*, 2 é do tipo* $Ultim$*, 3 é do tipo* $Prim$*, 4 é do tipo* $Fila\_caixa$*, 5 é do tipo* $Livre$*, 6 é do tipo* $Livre$*, 7 é do tipo* $FVazia$ *e 8 é do tipo* $Cliente\_livre$*. As ações* act_srcG *e* act_tgtG *descrevem o mapeamento de origem e destino das arestas, respectivamente.*

MACHINE mchGG
SEES ctxGG
VARIABLES

$vertG \;\; edgeG \;\; sourceG \;\; targetG \;\; tG\_V \;\; tG\_E$

INVARIANTS

inv_vertG_type : $vertG \in \mathbb{P}(\mathbb{N})$
inv_edgeG_type : $edgeG \in \mathbb{P}(\mathbb{N})$
inv_sourceG_type : $sourceG \in edgeG \rightarrow vertG$
inv_targetG_type : $targetG \in edgeG \rightarrow vertG$
inv_tVG_type : $tG\_V \in vertG \rightarrow vertT$
inv_tEG_type : $tG\_E \in edgeG \rightarrow edgeT$

EVENTS
Initialisation
begin

act_vertG : $vertG := \{1, 2, 3, 4\}$
act_edgeG : $edgeG := \{1, 2, 3, 4, 5, 6, 7, 8\}$
act_srcG : $sourceG := \{1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 3, 4 \mapsto 3, 5 \mapsto 3, 6 \mapsto 4, 7 \mapsto 4, 8 \mapsto 2\}$
act_tgtG : $targetG := \{1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 1, 4 \mapsto 3, 5 \mapsto 3, 6 \mapsto 4, 7 \mapsto 4, 8 \mapsto 2\}$
act_tG_V : $tG\_V := \{1 \mapsto Cliente, 2 \mapsto Cliente, 3 \mapsto Caixa, 4 \mapsto Caixa\}$
act_tG_E : $tG\_E := \{1 \mapsto Fila, 2 \mapsto Ultim, 3 \mapsto Prim, 4 \mapsto Fila\_caixa, 5 \mapsto Livre, 6 \mapsto Livre, 7 \mapsto FVazia, 8 \mapsto Cliente\_livre\}$

end

Figura 23: Grafo inicial da gramática Supermercado no modelo Event-B

A tradução do lado esquerdo(L) de uma regra, é análoga a tradução do grafo tipo realizada anteriormente, onde serão especificados o grafo L e seu correspondente morfismo de tipagem. Os vértices e arestas de L, bem como os nomes das funções de origem e destino devem ser declarados como constantes. Os axiomas devem definir explicitamente todos os conjuntos e funções. Todos os elementos que compõem os grafos dos lados esquerdos de todas as regras devem ser diferentes.

Definição 20 (Tradução do lado esquerdo da regra) *Dada uma regra* $p = L^T \leftarrow k^T \rightarrow R^T$*, a tradução do lado esquerdo de* p *para o Event-B, denotado por* $\mathcal{T}^p$ *é definido pelos seguintes elementos no contexto:*

CONTEXT *ctxGG*

SETS

$\texttt{Vert}L$

$\texttt{Edge}L$

CONSTANTS

$lista(VertL)$

$lista(EdgeL)$

$\texttt{source}L$

$\texttt{target}L$

$\texttt{t}L\texttt{\_V}$

$\texttt{t}L\texttt{\_E}$

AXIOMS

$\texttt{axm\_vert}L$: $\texttt{partition}(\texttt{Vert}L, elementos(VertL))$

$\texttt{axm\_edge}L$: $\texttt{partition}(\texttt{Edge}L, elementos(EdgeL))$

$\texttt{axm\_src}L\texttt{type}$: $\texttt{source}L \in \texttt{Edge}L \rightarrow \texttt{Vert}L$

$\texttt{axm\_src}L\texttt{def}$: $\texttt{partition}(\texttt{source}L, elementos(sourceL))$

$\texttt{axm\_tgt}L\texttt{type}$: $\texttt{target}L \in \texttt{Edge}L \rightarrow \texttt{Vert}L$

$\texttt{axm\_tgt}L\texttt{def}$: $\texttt{partition}(\texttt{target}L, elementos(sourceL))$

$\texttt{axm\_t}L\texttt{\_Vtype}$: $\texttt{t}L\texttt{\_V} \in \texttt{Vert}L \rightarrow \texttt{VertT}$

$\texttt{axm\_t}L\texttt{\_Vdef}$: $\texttt{partition}(\texttt{t}L\texttt{\_V}, elementos(tL\_V))$

$\texttt{axm\_t}L\texttt{\_Etype}$: $\texttt{t}L\texttt{\_E} \in \texttt{Edge}L \rightarrow \texttt{EdgeT}$

$\texttt{axm\_t}L\texttt{\_Edef}$: $\texttt{partition}(\texttt{t}L\texttt{\_E}, elementos(tL\_E))$

END

Exemplo 17 (Tradução do lado esquerdo de uma regra) *A tradução do lado esquerdo da regra* $Entrar2$ *(definida na Figura 8) para o Event-B é ilustrada pela Figura 24. Neste exemplo temos os vértices* $Cliente1\_L3$*,* $Cliente2\_L3$ *e* $CaixaL3$*, e as arestas* $ComprasL3$ *e* $UltimL3$*. Os vértices* $Cliente1\_L3$*,* $Cliente2\_L3$ *e* $CaixaL3$ *são mapeados para o tipo* $Cliente$*,* $Cliente$ *e* $Caixa$*, respectivamente. As arestas*

$ComprasL3$ e $UltimL3$ são mapeadas para o tipo $Compras$ e $Ultim$, respectivamente. A aresta $ComprasL3$ tem como origem e destino o vértice $Cliente2\_L3$, já a aresta $UltimL3$ tem o vértice $CaixaL3$ como origem e o vértice $Cliente1\_L3$ como destino.

CONTEXT ctxGG
SETS

...

$vertL3$
$edgeL3$
...

CONSTANTS

...

$Cliente1\_L3$ $Cliente2\_L3$ $CaixaL3$ $ComprasL3$ $UltimL3$
$targetL3$ $sourceL3$ $tL3\_E$ $tL3\_V$
...

AXIOMS

...

axm_vertL3 : $partition(vertL3, \{Cliente1\_L3\}, \{Cliente2\_L3\}, \{CaixaL3\})$
axm_edgeL3 : $partition(edgeL3, \{ComprasL3\}, \{UltimL3\})$
axm_srcL3type : $sourceL3 \in edgeL3 \rightarrow vertL3$
axm_srcL3def : $partition(sourceL3, \{ComprasL3 \mapsto Cliente2\_L3\}, \{UltimL3 \mapsto CaixaL3\})$
axm_tgtL3type : $targetL3 \in edgeL3 \rightarrow vertL3$
axm_tgtL3def : $partition(targetL3, \{ComprasL3 \mapsto Cliente2\_L3\}, \{UltimL3 \mapsto Cliente1\_L3\})$
axm_tVL3type : $tL3\_V \in vertL3 \rightarrow vertT$
axm_tVL3def : $partition(tL3\_V, \{Cliente1\_L3 \mapsto Cliente\}, \{Cliente2\_L3 \mapsto Cliente\}, \{CaixaL3 \mapsto Caixa\})$
axm_tEL3type : $tL3\_E \in edgeL3 \rightarrow edgeT$
axm_tEL3def : $partition(tL3\_E, \{ComprasL3 \mapsto Compras\}, \{UltimL3 \mapsto Ultim\})$
...

END

Figura 24: Tradução do lado esquerdo de uma regra para o modelo Event-B

A aplicação da regra é especificada na máquina do Event-B, cada item na regra é descrito como uma variável. As guardas são predicados construídos sobre os estados das constantes e variáveis do sistema, sendo as condições necessárias para um evento ocorrer, assim é dito que o evento é factível e que ele respeita a invariante especificada, quando a guarda é satisfeita, o evento pode ocorrer e as variáveis são modificadas. Um evento descreve o comportamento da aplicação de uma regra. Quando há valores para as variáveis m_V, m_E e para os novos vértices e arestas criadas onde as guardas do evento são satisfeitas, o evento ocorre. Depois que o evento inicial acontecer, qualquer outro evento habilitado pode vir a realizar-se.

As condições de guarda grd_mV, grd_mE, $grd\_vertices$, grd_edges e grd_srctgt

asseguram que o par (m_V, m_E) é realmente uma ocorrência do lado esquerdo da regra para o grafo estado G. As condições de guardas grd_new_v e grd_new_e asseguram que os vértices e arestas criados são realmente novos elementos no grafo. As guardas $grd\_PreservV$ e grd_DelV são para os vértices preservados e deletados, a guarda grd_DelE é para as arestas deletadas. As guardas $grd\_Ident1V$, $grd\_Ident2V$ e $grd\_Ident1E$, $grd\_Ident2E$ são para a identidade dos vértices e das arestas, respectivamente.

As ações servem para atualizar o grafo de estado de acordo com a regra. As ações act_E e act_V atualizam o grafo G, onde é excluído/adicionado as arestas e os vértices que são deletados/criados, respectivamente. As ações act_src e act_tgt são para descrever o mapeamento da origem e do destino para as arestas novas. Já as ações act_tV e act_tE descrevem o mapeamento para a tipagem dos novos vértices e arestas.

Definição 21 (Tradução de uma aplicação de regra) *Dada uma regra* $p = L^T \leftarrow k^T \rightarrow R^T$*, a tradução* $\mathcal{T}^p$ *da aplicação de uma regra* p *para o event-B é dada pelo evento a seguir:*

Event $p \mathrel{\widehat{=}}$

any

$\texttt{m\_V}$

$\texttt{m\_E}$

$\texttt{DelV}$

$\texttt{PreservV}$

$\texttt{Dangling}$

$\texttt{DelE}$

$lista\_prefixada(NewV, \texttt{new})$

$lista\_prefixada(NewE, \texttt{new})$

where

$\texttt{grd\_mV} : \texttt{m\_V} \in \texttt{Vert}L \rightarrow \texttt{VertG}$

$\texttt{grd\_mE} : \texttt{m\_E} \in \texttt{Edge}L \rightarrow \texttt{EdgeG}$

$\texttt{grd\_PreservV} : \texttt{PreservV} = \texttt{VertG} \setminus \texttt{DelV}$

$\texttt{grd\_DelV} : \texttt{DelV} := \texttt{m\_V}[conjunto(RegrasDel_V)]$

$\texttt{grd\_DelE} : \texttt{DelE} := \texttt{m\_E}[conjunto(RegrasDel_E)]$

$\texttt{grd\_Dang} : \texttt{Dangling} = \texttt{dom}((\texttt{sourceG} \rhd \texttt{DelV}) \cup (\texttt{targetG} \rhd \texttt{DelV})) \setminus \texttt{DelE}$

$\forall v \in NewV$

$\quad \texttt{grd\_new\_}v : \texttt{new\_}v \in \mathbb{N} \setminus \texttt{VertG}$

$\forall e \in NewE$

$\quad \texttt{grd\_new\_}e : \texttt{new\_}e \in \mathbb{N} \setminus \texttt{EdgeG}$

$\forall v_i, v_j \in NewV$ *com* $v_i \neq v_j$

$\quad \texttt{grd\_diff}v_iv_j : \texttt{new\_}v_i \neq \texttt{new\_}v_j$

$\forall e_i, e_j \in NewE$ **com** $e_i \neq e_j$
$\mid$ grd_diff$e_i e_j$: new_$e_i \neq$new_e_j
grd_vertices : $\forall v \cdot v \in \mathtt{Vert}L \Rightarrow \mathtt{t}L\_\mathtt{V}(v) = \mathtt{tG\_V}(\mathtt{m\_V}(v))$
grd_edges : $\forall e \cdot e \in \mathtt{Edge}L \Rightarrow \mathtt{t}L\_\mathtt{E}(e) = \mathtt{tG\_E}(\mathtt{m\_E}(e))$
grd_srctgt : $\forall e \cdot e \in \mathtt{Edge}L \Rightarrow \mathtt{m\_V}(\mathtt{source}L(e)) = \mathtt{sourceG}(\mathtt{m\_E}(e)) \wedge$
$\mathtt{m\_V}(\mathtt{target}L(e)) = \mathtt{targetG}(\mathtt{m\_E}(e))$
grd_Ident1V : $\mathtt{DelV} \cap \mathtt{m\_V}[RegrasPreserv_V] = \varnothing$
grd_Ident2V : $\mathtt{card}(\mathtt{DelV}) = \mathtt{card}(conjunto(RegrasDel_V))$
grd_Ident1E : $\mathtt{DelE} \cap \mathtt{m\_E}[RegraPreserv_E] = \varnothing$
grd_Ident2E : $\mathtt{card}(\mathtt{DelE}) = \mathtt{card}(conjunto(RegrasDel_E))$
grd_DangCondition : $\mathtt{Dangling} = \varnothing$

then

act_V : $\mathtt{VertG} := (\mathtt{VertG} \setminus \mathtt{DelV}) \cup conjunto\_prefixado(NewV, \mathtt{new})$
act_E : $\mathtt{EdgeG} := (\mathtt{EdgeG} \setminus \mathtt{DelE}) \cup conjunto\_prefixado(NewE, \mathtt{new})$
act_src : $\mathtt{sourceG} := (\mathtt{DelE} \lhd \mathtt{sourceG}) \cup$
{
$\forall e \in NewE$ **com** $v = sourceR(e) \in NewV$
$\mid$ new_$e \mapsto v$
$\forall e \in NewE$ **com** $v = sourceR(e) \in RegrasPreserv_V$
$\mid$ new_$e \mapsto$ m_V(v)
}
act_tgt : $\mathtt{targetG} := (\mathtt{DelE} \lhd \mathtt{targetG}) \cup$
{
$\forall e \in NewE$ **com** $v = targetR(e) \in NewV$
$\mid$ new_$e \mapsto v$
$\forall e \in NewE$ **com** $v = targetR(e) \in RegrasPreserv_V$
$\mid$ new_$e \mapsto$ m_V(v)
}
act_tV : $\mathtt{tG\_V} := (\mathtt{DelV} \lhd \mathtt{tG\_V}) \cup$
{
$\forall v \in NewV$ **com** $tv = tR\_V(v).$
$\mid$ new_$v \mapsto tv$
}
act_tE : $\mathtt{tG\_E} := (\mathtt{DelV} \lhd \mathtt{tG\_E}) \cup$
{
$\forall v \in NewE$ **com** $te = tR\_E(e).$
$\mid$ new_$e \mapsto te$
}

end

END

Exemplo 18 (Tradução da aplicação de uma regra) *A tradução da aplicação da regra Entrar2 da GG Supermercado (definida na Figura 8) para o Event-B é ilustrada pela Figura 25. As condições de guarda grd_new_Fila, grd_new_Ultim e grd_new_Prox das arestas são para garantir que as arestas* new_Fila, new_Ultim *e* new_Prox *são realmente novos elementos no grafo. Na aplicação desta regra, as arestas* new_Ultim, new_Fila *e* new_Prox *são criadas pela ação* act_E *essa ação também deleta as arestas* $ComprasL3$ *e* $UltimL3$. *As arestas* new_Ultim, new_Fila *e* new_Prox *tem suas origens e destinos definidos pelas ações* act_src *e* act_tgt, *onde as arestas* new_Ultim, new_Fila *e* new_Prox *tem suas origens nos vértices* $CaixaL3$, $Cliente2\_L3$ *e* $Cliente1\_L3$ *e seus destinos nos vértices* $Cliente2\_L3$, $Cliente2\_L3$ *e* $Cliente2\_L3$, *respectivamente. A ação* act_tE *descreve a tipagem das arestas* new_Ultim, new_Fila *e* new_Prox *onde essas são mapeadas para os tipos* $Ultim$, $Fila$ *e* $Prox$, *respectivamente.*

EVENTS

Event *Entrar2* $\widehat{=}$

any

m_V
m_E
$DelV$
$PreservV$
$DelE$
$Dangling$
new_Ultim
new_Fila
new_Prox

where

grd_mV : $m\_V \in vertL3 \rightarrow vertG$
grd_mE : $m\_E \in edgeL3 \rightarrow edgeG$
grd_DelV : $DelV = m\_V[\varnothing]$
grd_PreV : $PreservV = vertG \setminus DelV$
grd_DelE : $DelE = m\_E[edgeL3 \setminus \{ComprasL3, UltimL3\}]$
grd_Dang : $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$
grd_new_Ultim : $new\_Ultim \in \mathbb{N} \setminus edgeG$
grd_new_Fila : $new\_Fila \in \mathbb{N} \setminus edgeG$
grd_new_Prox : $new\_Prox \in \mathbb{N} \setminus edgeG$
grd_vertices : $\forall v \cdot v \in vertL3 \Rightarrow tL3\_V(v) = tG\_V(m\_V(v))$
grd_edges : $\forall e \cdot e \in edgeL3 \Rightarrow tL3\_E(e) = tG\_E(m\_E(e))$
grd_srctgt : $\forall e \cdot e \in edgeL3 \Rightarrow m\_V(sourceL3(e)) = sourceG(m\_E(e))$
$\wedge\ m\_V(targetL3(e)) = targetG(m\_E(e))$
grd_IdentV : $DelV \cap m\_V[\{Cliente1\_L3, Cliente2\_L3, CaixaL3\}] = \varnothing$
grd_IdentE : $DelE \cap m\_E[\{\}] = \varnothing$
grd_DangCondition : $Dangling = \varnothing$

then

act_V : $vertG := (vertG \setminus DelV) \cup \varnothing$
act_E : $edgeG := (edgeG \setminus DelE) \cup \{new\_Fila, new\_Ultim, new\_Prox\}$
act_src : $sourceG := (DelE \lhd sourceG) \cup \{new\_Fila \mapsto m\_V(Cliente2\_L3),$
$new\_Prox \mapsto m\_V(Cliente1\_L3), new\_Ultim \mapsto m\_V(CaixaL3)\}$
act_tgt : $targetG := (DelE \lhd targetG) \cup \{new\_Ultim \mapsto m\_V(Cliente2\_L3),$
$new\_Fila \mapsto m\_V(Cliente2\_L3), new\_Prox \mapsto m\_V(Cliente2\_L3)\}$
act_tV : $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$
act_tE : $tG\_E := (DelV \lhd tG\_E) \cup \{new\_Ultim \mapsto Ultim, new\_Fila \mapsto Fila,$
$new\_Prox \mapsto Prox\}$

end

Figura 25: Tradução da aplicação de uma regra para o modelo Event-B

5 TRADUÇÃO DA GRAMÁTICA DE GRAFOS FUZZY PARA O MODELO EVENT-B

Neste capítulo é definido a tradução de Gramáticas de Grafos Fuzzy para modelos Event-B. Essa tradução é uma extensão da tradução de Gramática de Grafos apresentadas na seção anterior. Para definir a tradução foi necessário utilizar uma teoria de números reais para especificar os valores de pertinência dos vértices e arestas, visto que a plataforma Rodin tem suporte apenas para os números inteiros e naturais.

As definições da tradução da gramática de grafos para o Event-B definidas na seção anterior foram estendidas para as gramáticas de grafos fuzzy, onde o grafo tipo, inicial, lado esquerdo e aplicação da regra da gramática de grafos são estendidos para o grafo tipo fuzzy, grafo inicial fuzzy, lado esquerdo e aplicação da regra fuzzy, respectivamente. Sendo assim, foi possível fazer a tradução desta gramática pra o Event-B.

Para estender o grafo tipo fuzzy foi adicionado as pertinências nos vértices e nas arestas no grafo tipo fuzzy T. No grafo inicial fuzzy é adicionado as pertinências para os elementos presentes no grafo fuzzy G, ou seja, cada vértice e aresta é mapeado para um valor de pertinência. No lado esquerdo das regras fuzzy é mudado apenas o tratamento das funções de pertinência. Por fim, para a aplicação da regra fuzzy são adicionadas novas guardas para identificar qual o maior valor de pertinência para os vértices e arestas que são preservados. Novas ações são adicionadas, essas são usadas para atualizar o valor de pertinência dos vértices/arestas preservados e ainda para associar um novo valor de pertinência para os novos vértices/arestas.

5.1 Teoria Plug-in

Existem extensões para a plataforma Rodin que são disponibilizadas pela comunidade de pesquisadores e desenvolvedores. Uma destas extensões é a Theory Plug-in (MAAMRIA; FATHABADI, 2014). Esta extensão permite a criação de teorias que facilitam a especificação, validação e desenvolvimento de especificações em Event-B (BUTLER; MAAMRIA, 2013), com isto é permitido definir tipos de dados, axiomas,

operadores, regras de inferência e de reescrita. O novo tipo de dado define uma função construtora e uma destrutora para obter os elementos deste novo tipo. O operador define a propriedade e deve gerar uma obrigação de prova, que quando demonstrada, valida as condições do operador.

Na documentação da theory Plug-in existe uma biblioteca padrão que contém projetos com teorias disponibilizadas por usuários e desenvolvedores da plataforma Rodin, as quais se destacam:

- BasicTheory que inclui teorias para manipulação de árvores binárias (BinaryTree), de operadores booleanos (BoolOps), de listas (List), de números reais (SUMandPRODUCT) e de sequênciad de números (Seq).
- RelationOrderTheory que contém a teoria sobre ponto-fixo(FixPoint).
- RealTheory que contém uma teoria de números reais(Real).

Para a realização deste trabalho foi utilizada a teoria dos reais desenvolvida por Jean-Raymond Abrial e Michael Butler (ABRIAL; BUTLER, 2014). Esta teoria define operações com os números reais como: soma($plus$), subtração(sub), multiplicação($mult$), comparação(smr, gtr, leq), supremo(sup) e ínfimo(inf).

Na Figura 26 é ilustrada uma parte da teoria, com os operadores e axiomas que foram adicionados e utilizados. Originalmente a teoria utilizava apenas duas constantes 0($zero$) e 1(one) para realizar as operações. Para a utilização da teoria neste trabalho foi necessário adicionar outras constantes, como 0,1($zeroone$), 0,2($zerotwo$), 0,3($zerothree$), 0,4($zerofour$), 0,5($zerofive$), 0,6($zerosix$), 0,7($zeroseven$), 0,8($zeroeight$) e 0,9($zeronine$) para representar o intervalo $[0, 1]$, pois foi usado uma abstração deste intervalo. Essa teoria dos reais foi necessária para especificar os valores de pertinência dos vértices e das arestas, pois o Rodin não tem suporte para os números reais.

Para cada constante nova adicionada foi necessário novos axiomas, esses axiomas servem para definir que cada constante é diferente das já existentes ou das que são adicionadas. Também para cada constante nova é definido a ordem entre elas (qual tem maior valor), por exemplo, como é ilustrado na Figura 26, para as duas constantes adicionadas $zerotwo$ e $zerofive$ a operação $axm31 : leq(zerotwo, zerofive)$ define que a constante $zerotwo$ é menor que a constante $zerofive$. Os $axm27$, $axm29$ e $axm31$ definem que as constantes $zero$, $zerotwo$ e $zerotwo$ são menores que as constantes $zerotwo$, one e $zerofive$, respectivamente. A operação é necessária pois as operações supremo(sup) e ínfimo(inf) necessitam saber qual constante é maior/menor entre as comparadas. Os axiomas $axm26$, $axm28$ e $axm30$ definem que essas constantes são diferentes entre elas.

THEORY
Real
AXIOMATIC DEFINITION
Real_def
TYPES
REAL
OPERATORS
...
- **plus:** plus (a : REAL, b : REAL) **EXPRESSION INFIX** REAL
- **zero:** zero ( ) **EXPRESSION INFIX** REAL
- **mult:** mult (a: REAL, b: REAL) **EXPRESSION INFIX** REAL
- **one:** one ( ) **EXPRESSION INFIX** REAL

well-definedness condition
$a \neq zero$
- **leq:** leq (a: REAL, b: REAL) **PREDICATE PREFIX** REAL
- **sup:** sup (A: $\mathbb{P}(REAL)$) **EXPRESSION INFIX** REAL
- **zeroone:** zeroone ( ) **EXPRESSION INFIX** REAL
- **zerotwo:** zerotwo ( ) **EXPRESSION INFIX** REAL
- **zerothree:** zerothree ( ) **EXPRESSION INFIX** REAL
- **zerofour:** zerofour ( ) **EXPRESSION INFIX** REAL
- **zerofive:** zerofive ( ) **EXPRESSION INFIX** REAL
- **zerosix:** zerosix ( ) **EXPRESSION INFIX** REAL
- **zeroseven:** zeroseven ( ) **EXPRESSION INFIX** REAL
- **zeroeight:** zeroeight ( ) **EXPRESSION INFIX** REAL
- **zeronine:** zeronine ( ) **EXPRESSION INFIX** REAL

...
AXIOMS
...
axm9: zero ≠ one //zero different from one
axm11: $\forall x \cdot leq(x,x)$ // order is reflexive
axm12: $\forall x,y \cdot leq(x,y) \land leq(y,x) \implies x = y$ // order is antisymmetric
axm13: $\forall x,y,z \cdot leq(x,y) \land leq(y,z) \implies leq(x,z)$ // order is transitive
axm14: $\forall x,y \cdot leq(x,y) \lor (y,x)$ // order is total
...
axm17: $\forall A \cdot A \subseteq REAL \land A \neq \emptyset \land \big(\exists m \cdot m \in REAL \land (\forall x \cdot x \in A \implies leq(x,m))\big) \implies (\forall x \cdot x \in A \implies leq(x, sup(A)))$ // sup(A) is an upper bound of A
axm18: $\forall A, v \cdot A \subseteq REAL \land A \neq \emptyset \land \big(\exists m \cdot m \in REAL \land (\forall x \cdot x \in A \implies leq(x,m))\big) \implies (\forall x \cdot x \in A \implies leq(x,v)) \implies leq(sup(A), v)$ // sup(A) is the least upper bound of A
...
axm26: zero ≠ zerotwo
axm27: leq (zero, zerotwo)
axm28: zerotwo ≠ one
axm29: leq (zerotwo, one)
axm30: zerotwo ≠ zerofive
axm31: leq (zerotwo, zerofive)
...
PROOF RULES
...
gtr_smr :
Metavariables
- x ∈ REAL
- y ∈ REAL

Rewrite Rules
- rew20 : gtr (x, y) (case- incomplete, interactive) gtr_smr
- rhs1 : ⊤ ▶ $smr(y,x)$

END

Figura 26: Teoria dos Reais

No restante deste capítulo será definida extensão da tradução de GGFs para o Event-B.

5.2 Tradução do grafo tipo fuzzy

Para realizar a tradução do grafo tipo fuzzy da gramática de grafos fuzzy, foi necessário estender a definição do grafo tipo da gramática de grafos definida na seção anterior para poder incluir a pertinência dos vértices e das arestas. As funções de valoração $pert\_vertT$ e $pert\_edgeT$ são adicionadas como constantes e são definidas nos axiomas.

Quatro novos axiomas são adicionados, os quais são responsáveis pela tipagem das funções de pertinência para os vértices e as arestas. Os axiomas $axm\_pertinenciavertT\_type$ e $axm\_pertinenciaedgeT\_type$ definem o tipo da pertinência para os vértices e arestas. A pertinência dos vértices $pert\_vertT$ e a pertinência das arestas $pert\_edgeT$ mapeiam os seus respectivos conjuntos para um valor $REAL$. Os axiomas $axm\_pertinenciavertT\_def$ e $axm\_pertinenciaedgeT\_def$ mapeiam cada vértice e aresta para um valor $REAL$.

Definição 22 (Tradução do grafo tipo fuzzy) *Dado um grafo tipo fuzzy* $T = (VertT, EdgeT, sourceT, targetT)$ *definimos sua tradução* $\mathcal{T}_F^T$ *como os elementos de contexto do event-B:*

CONTEXT *ctxGGF*

SETS

VertT

EdgeT

CONSTANTS

$lista(VertT)$

$lista(EdgeT)$

sourceT

targetT

pert_vertT

pert_edgeT

AXIOMS

Contém todos os axiomas de $\mathcal{T}^T$ *(definição 18) e inclui os axiomas que definem pert_vertT e pert_edgeT:*

axm_pertinenciavertT_type : pert_vertT $\in$ VertT $\rightarrow$ REAL

axm_pertinenciavertT_def : partition(pert_vertT,$elementos(pert\_vertT)$)

axm_pertinenciaedgeT_type : pert_edgeT $\in$ EdgeT $\rightarrow$ REAL

axm_pertinenciaedgeT_def : partition(pert_edgeT,$elementos(pert\_edgeT)$)

END

Exemplo 19 (Tradução do grafo tipo fuzzy) *A Figura 27 mostra a tradução do grafo tipo fuzzy do exemplo da gramática de grafos fuzzy Ferrovia (ver Figura 18),* $vertT$ *e* $edgeT$ *são conjuntos conhecidos no modelo e todos os tipos de vértices e arestas,* $sourceT$, $targeT$, $pert\_vertT$ *e* $pert\_edgeT$ *são especificados como constantes.*

Os conjuntos de vértices $\{trem, estacao\}$ *e arestas* $\{trilho, Distancia\_estacao\}$ *são determinados pelos axiomas* axm_vertT *e* axm_edgeT*, respectivamente. O axioma* $axm\_srcTdef$ *define o mapeamento de origem para as arestas* $trilho$ *e* $Distancia\_estacao$ *que são os vértices* $estacao$ *e* $trem$*, respectivamente. Para o mapeamento de destino destas arestas, o axioma* $axm\_tgtTdef$ *define que os vértices de destino são* $estacao$ *e* $estacao$*, respectivamente.*

O axioma $axm\_pertinenciavertT\_def$ *mapeia os vértices* $trem$ *e* $estacao$ *para o valor* one*. No* $axm\_pertinenciaedgeT\_def$ *é feito o mapeamento das arestas* $trilho$ *e* $Distancia\_estacao$ *para o valor* one*.*

CONTEXT ctxGGF
SETS

$vertT$
$edgeT$

CONSTANTS

$trem\ \ estacao\ \ trilho\ \ Distancia\_estacao\ \ sourceT\ \ targetT\ \ pert\_vertT\ \ pert\_edgeT$

AXIOMS

axm_vertT: $partition(vertT, \{trem\}, \{estacao\})$
axm_edgeT: $partition(edgeT, \{trilho\}, \{Distancia\_estacao\})$
axm_srcTtype: $sourceT \in edgeT \rightarrow vertT$
axm_srcTdef: $partition(sourceT, \{trilho \mapsto estacao\}, \{Distancia\_estacao \mapsto trem\})$
axm_tgtTtype: $targetT \in edgeT \rightarrow vertT$
axm_tgtTdef: $partition(targetT, \{trilho \mapsto estacao\}, \{Distancia\_estacao \mapsto estacao\})$
axm_pertinenciavertT_type: $pert\_vertT \in vertT \rightarrow REAL$
axm_pertinenciavertT_def: $partition(pert\_vertT, \{trem \mapsto one\}, \{estacao \mapsto one\})$
axm_pertinenciaedgeT_type: $pert\_edgeT \in edgeT \rightarrow REAL$
axm_pertinenciaedgeT_def: $partition(pert\_edgeT, \{trilho \mapsto one\},$
$\{Distancia\_estacao \mapsto one\})$

END

Figura 27: Grafo tipo fuzzy da gramática de grafos fuzzy Ferrovia

5.3 Tradução do grafo inicial fuzzy

O grafo inicial fuzzy é especificado no componente máquina em Event-B. O grafo inicial fuzzy também é estendido do grafo inicial da GG (definição 19), onde são adicionados duas constantes, duas invariantes e duas ações, as invariantes são usadas para definir os tipos das variáveis e as ações para definir o mapeamento de cada vértice ou aresta para o seu valor de pertinência.

As constantes adicionadas são $(pert\_vertG)$ e $(pert\_edgeG)$. A pertinência dos vértices $(pert\_vertG)$ e das arestas $(pert\_edgeG)$ são funções totais dos vértices $(VertG)$ e das arestas $(EdgeG)$ para os números reais $(REAL)$, e são dadas pelas invariantes $inv\_pert\_vertG\_type$ e $inv\_pert\_edgeG\_type$. O evento de inicialização $(Initialisation)$ simplesmente atribui os valores às variáveis, sem verificar as condições das guardas. Os vértices e as arestas são definidos (pelas ações act_vertG e act_edgeG). As ações $act\_pert\_vertG\_type$ e $act\_pert\_edgeG\_type$ mapeiam cada vértice e aresta para o seu valor de pertinência.

Definição 23 (Tradução do grafo inicial fuzzy) *Um grafo de estado é definido pelas variáveis e invariantes definidas logo abaixo no event-B, onde um dado grafo inicial fuzzy* $G0^T = (G0, tG0, T)$ *com* $vertG0 \subseteq \mathbb{N}$ *e* $edgeG0 \subseteq \mathbb{N}$*, sua tradução* $\mathcal{T}_F^{G0}$ *para o event-B é definida pelo evento de inicialização definido abaixo:*

MACHINE *mchGGF*

SEES *ctxGGF*

VARIABLES

VertG

EdgeG

sourceG

targetG

tG_V

tG_E

pert_vertG

pert_edgeG

INVARIANTS

Contém todas as invariantes de $\mathcal{T}^{G0}$*(definição 19) e inclui as invariantes pert_vertG e pert_edgeG:*

inv_pert_vertG_type : pert_vertG $\in$ VertG $\rightarrow$ REAL

inv_pert_edgeG_type : pert_edgeG $\in$ EdgeG $\rightarrow$ REAL

EVENTS

Initialisation

begin

Contém todas as ações de $\mathcal{T}^{G0}$(definição 19) e inclui as ações act_pert_vertG_type e act_pert_edgeG_type:

$\texttt{act\_pert\_vertG\_type} : \texttt{pert\_vertG} := conjunto(pert\_vertG0)$

$\texttt{act\_pert\_edgeG\_type} : \texttt{pert\_edgeG} := conjunto(pert\_edgeG0)$

end

END

Exemplo 20 (Tradução do grafo inicial fuzzy) *A Figura 28 ilustra a tradução para o modelo Event-B do grafo inicial fuzzy da GGF Ferrovia (ver Figura 19). Neste exemplo temos 12 vértices e 15 arestas. As ações $act\_pert\_vertG\_type$ e $act\_pert\_edgeG\_type$ mapeiam cada vértice e aresta para o seu valor de pertinência, onde todos os vértices são mapeados para o valor 1(one), e as arestas são mapeadas conforme a Tabela 1.*

Tabela 1: Mapeamento da pertinência das arestas no grafo inicial fuzzy

Aresta	Valor
1	0,2$(zerotwo)$
2	1(one)
3	0$(zero)$
4	0,4$(zeroseven)$
5	0,2$(zerotwo)$
6	0,8$(zeroeight)$
7	0,2$(zerotwo)$
8	0,8$(zeroeight)$
9	0,7$(zeroseven)$
10	0,5$(zerofive)$
11	0,9$(zeronine)$
12	0,9$(zeronine)$
13	0,2$(zerotwo)$
14	0,8$(zeroeight)$
15	0,5$(zerofive)$

5.4 Tradução das regras fuzzy

Assim como é feito na tradução para as GGs, aqui também é traduzido o lado esquerdo das regras para o contexto e a aplicação da regra para os eventos, o que muda agora é o tratamento das funções de pertinência. A tradução do lado esquerdo da regra da gramática de grafos fuzzy estende a tradução do lado esquerdo das regras da GGs.

MACHINE mchGGF
SEES ctxGGF
VARIABLES

$vertG$
$edgeG$
$sourceG$
$targetG$
tG_V
tG_E
$pert\_vertG$
$pert\_edgeG$

INVARIANTS

inv_vertG_type : $vertG \in \mathbb{P}(\mathbb{N})$
inv_edgeG_type : $edgeG \in \mathbb{P}(\mathbb{N})$
inv_sourceG_type : $sourceG \in edgeG \rightarrow vertG$
inv_targetG_type : $targetG \in edgeG \rightarrow vertG$
inv_tVG_type : $tG\_V \in vertG \rightarrow vertT$
inv_tEG_type : $tG\_E \in edgeG \rightarrow edgeT$
inv_pert_vertG_type : $pert\_vertG \in vertG \rightarrow REAL$
inv_pert_edgeG_type : $pert\_edgeG \in edgeG \rightarrow REAL$

EVENTS
Initialisation
begin

act_vertG : $vertG := \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$
act_edgeG : $edgeG := \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$
act_srcG : $sourceG := \{1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 1, 4 \mapsto 3, 5 \mapsto 4, 6 \mapsto 6, 7 \mapsto 6,$
$8 \mapsto 5, 9 \mapsto 7, 10 \mapsto 2, 11 \mapsto 8, 12 \mapsto 9, 13 \mapsto 10, 14 \mapsto 9, 15 \mapsto 12\}$
act_tgtG : $targetG := \{1 \mapsto 3, 2 \mapsto 2, 3 \mapsto 3, 4 \mapsto 4, 5 \mapsto 5, 6 \mapsto 4, 7 \mapsto 5,$
$8 \mapsto 7, 9 \mapsto 11, 10 \mapsto 8, 11 \mapsto 9, 12 \mapsto 10, 13 \mapsto 11, 14 \mapsto 12, 15 \mapsto 15\}$
act_tVG_type : $tG\_V := \{1 \mapsto trem, 2 \mapsto estacao, 3 \mapsto estacao, 4 \mapsto estacao,$
$5 \mapsto estacao, 6 \mapsto trem, 7 \mapsto estacao, 8 \mapsto estacao, 9 \mapsto estacao, 10 \mapsto estacao,$
$11 \mapsto estacao, 12 \mapsto estacao\}$
act_tEG_type: $tG\_E := \{1 \mapsto trilho, 2 \mapsto Distancia\_estacao,$
$3 \mapsto Distancia\_estacao, 4 \mapsto trilho, 5 \mapsto trilho, 6 \mapsto Distancia\_estacao,$
$7 \mapsto Distancia\_estacao, 8 \mapsto trilho, 9 \mapsto trilho, 10 \mapsto trilho, 11 \mapsto trilho,$
$12 \mapsto trilho, 13 \mapsto trilho, 14 \mapsto trilho, 15 \mapsto trilho\}$
act_pert_vertG_type: $pert\_vertG := \{1 \mapsto one, 2 \mapsto one, 3 \mapsto one, 4 \mapsto one,$
$5 \mapsto one, 6 \mapsto one, 7 \mapsto one, 8 \mapsto one, 9 \mapsto one, 10 \mapsto one, 11 \mapsto one, 12 \mapsto one\}$
act_pert_edgeG_type: $pert\_edgeG := \{1 \mapsto zerotwo, 2 \mapsto one, 3 \mapsto zero,$
$4 \mapsto zeroseven, 5 \mapsto zerotwo, 6 \mapsto zeroeight, 7 \mapsto zerotwo, 8 \mapsto zeroeight,$
$9 \mapsto zeroseven, 10 \mapsto zerofive, 11 \mapsto zeronine, 12 \mapsto zeronine, 13 \mapsto zerotwo,$
$14 \mapsto zeroeight, 15 \mapsto zerofive\}$

end
END

Figura 28: Grafo inicial fuzzy da gramática de grafos fuzzy Ferrovia

Definição 24 (Tradução do lado esquerdo de uma regra fuzzy) *Dada uma regra fuzzy* $p = L^T \leftarrow K^T \rightarrow R^T$*, a tradução de* L^T *para o Event-B, denotado por* $\mathcal{T}_F^p$ *é*

definido pelos seguintes elementos:

CONTEXT *ctxGGF*

SETS

$\texttt{Vert}L$

$\texttt{Edge}L$

CONSTANTS

$lista(VertL)$

$lista(EdgeL)$

$\texttt{source}L$

$\texttt{target}L$

$\texttt{t}L\texttt{\_V}$

$\texttt{t}L\texttt{\_E}$

$\texttt{pert\_vert}L$

$\texttt{pert\_edge}L$

AXIOMS

Contém todos os axiomas de $\mathcal{T}^p$ (definição 20) e inclui os axiomas que definem pert_vertL e pert_edgeL :

$\texttt{axm\_pertinenciavert}L\texttt{\_type} : \texttt{pert\_vert}L \in \texttt{Vert}L \rightarrow \texttt{REAL}$

$\texttt{axm\_pertinenciavert}L\texttt{\_def} : \texttt{partition}(\texttt{pert\_vert}L, elementos(pert\_vertL))$

$\texttt{axm\_pertinenciaedge}L\texttt{\_type} : \texttt{pert\_edge}L \in \texttt{Edge}L \rightarrow \texttt{REAL}$

$\texttt{axm\_pertinenciaedge}L\texttt{\_def} : \texttt{partition}(\texttt{pert\_edge}L, elementos(pert\_edgeL))$

END

Exemplo 21 (Tradução do lado esquerdo de uma regra fuzzy) *A Figura 29 ilustra a tradução do lado esquerdo da regra fuzzy $r1$ da GGF Ferrovia descrita na Figura 20. Todos os valores de pertinência dos vértices são mapeadas para o valor real $1(one)$ através do axioma $axm\_pertinenciavertL1\_def$. Já para o valor de pertinência das arestas, é dado pelo axioma $axm\_pertinenciaedgeL1\_def$ como segue: $Distancia\_estacaoAL1$, $Distancia\_estacaoBL1$, $trilhoABL1$ são mapeadas para os valores $1(one)$, $0(zero)$, $0,2(zerotwo)$, respectivamente.*

A definição 21 é estendida para definir a tradução da aplicação de regra fuzzy. As guardas $Pertpreserv\_vert\_v$ são necessárias para identificar qual o maior valor de pertinência para os vértices que são preservados, pois com a aplicação da regra, os vértices preservados podem ter o seu valor de pertinência alterado. Assim, o valor

CONTEXT ctxGGF
SETS

...

$vertL1$
$edgeL1$
...

CONSTANTS

...

$estacaoAL1$ $estacaoBL1$ $tremL1$ $Distancia\_estacaoAL1$ $Distancia\_estacaoBL1$
$trilhoABL1$ $sourceL1$ $targetL1$ $tL1\_V$ $tL1\_E$ $pert\_vertL1$ $pert\_edgeL1$
...

AXIOMS

...

axm_vertL1: $partition(vertL1, \{tremL1\}, \{estacaoAL1\}, \{estacaoBL1\})$
axm_edgeL1: $partition(edgeL1, \{Distancia\_estacaoAL1\}, \{Distancia\_estacaoBL1\}, \{trilhoABL1\})$
axm_srcL1type: $sourceL1 \in edgeL1 \rightarrow vertL1$
axm_srcL1def: $partition(sourceL1, \{trilhoABL1 \mapsto estacaoAL1\}, \{Distancia\_estacaoAL1 \mapsto tremL1\}, \{Distancia\_estacaoBL1 \mapsto tremL1\})$
axm_tgtL1type: $targetL1 \in edgeL1 \rightarrow vertL1$
axm_tgtL1def: $partition(targetL1, \{trilhoABL1 \mapsto estacaoBL1\}, \{Distancia\_estacaoAL1 \mapsto estacaoAL1\}, \{Distancia\_estacaoBL1 \mapsto estacaoBL1\})$
axm_tL1Vtype: $tL1\_V \in vertL1 \rightarrow vertT$
axm_tL1Vdef: $partition(tL1\_V, \{tremL1 \mapsto trem\}, \{estacaoAL1 \mapsto estacao\}, \{estacaoBL1 \mapsto estacao\})$
axm_tL1Etype: $tL1\_E \in edgeL1 \rightarrow edgeT$
axm_tL1Edef: $partition(tL1\_E, \{Distancia\_estacaoAL1 \mapsto Distancia\_estacao\}, \{Distancia\_estacaoBL1 \mapsto Distancia\_estacao\}, \{trilhoABL1 \mapsto trilho\})$
axm_pertinenciavertL1_type: $pert\_vertL1 \in vertL1 \rightarrow REAL$
axm_pertinenciavertL1_def: $partition(pert\_vertL1, \{tremL1 \mapsto one\}, \{estacaoAL1 \mapsto one\}, \{estacaoBL1 \mapsto one\})$
axm_pertinenciaedgeL1_type: $pert\_edgeL1 \in edgeL1 \rightarrow REAL$
axm_pertinenciaedgeL1_def: $partition(pert\_edgeL1, \{Distancia\_estacaoAL1 \mapsto one\}, \{Distancia\_estacaoBL1 \mapsto zero\}, \{trilhoABL1 \mapsto zerotwo\})$
...

END

Figura 29: Tradução do lado esquerdo da regra r1 GGF Ferrovia.

para o vértice é dado pelo maior valor (supremo) entre o valor que ele já possuía (antes da aplicação da regra) e o valor especificado do lado direito da regra. As guardas $Pertpreserv\_edge\_e$ análogas as guardas $Pertpreserv\_vert\_v$, porém referem-se às arestas.

As ações atualizam o grafo de estado de acordo com a especificação da regra. As ações act_V e act_E atualizam o grafo G, excluindo as arestas e os vértices deletados e adicionando as arestas e os vértices criados. A ação act_pert_vertG é usada para atualizar o valor de pertinência dos vértices preservados e para associar o valor de

pertinência aos novos vértices. O mesmo acontece com a ação act_pert_edgeG, mas para as arestas ao invés de vértices.

Definição 25 (Tradução da aplicação de uma regra) *Dada uma regra* $p = L^T \leftarrow K^T \rightarrow R^T$*, a tradução* $\mathcal{T}_F^p$ *de* p *para o event-B é dada pelo evento a seguir:*

Event $r \widehat{=}$

any

Contém todas as variáveis $\mathcal{T}^p$*(definição 21) e inclui as variáveis Pertpreserv_vert e Pertpreserv_edge :*

$prefixed\_list(PreservV, \texttt{Pertpreserv\_vert})$
$prefixed\_list(PreservE, \texttt{Pertpreserv\_edge})$

where

Contém todas as guardas $\mathcal{T}^p$*(definição 21) e inclui as guardas que definem Pertpreserv_vert_v e Pertpreserv_edge_e :*

$\forall v \in PreservV$
$\quad \texttt{grd\_Pertpreserv\_vert\_}v : \texttt{Pertpreserv\_vert\_}v = \texttt{sup}(\{\texttt{pert\_vertG}(\texttt{m\_V}(\texttt{vert}L(v))), \texttt{pert\_vert}R(v)\})$
$\forall e \in PreservE$
$\quad \texttt{grd\_Pertpreserv\_edge\_}e : \texttt{Pertpreserv\_edge\_}e = \texttt{sup}(\{\texttt{pert\_edgeG}(\texttt{m\_E}(\texttt{edge}L(e))), \texttt{pert\_edge}R(e)\})$

then

Contém todas as ações $\mathcal{T}^p$*(definição 21) e inclui as ações que definem pert_vertG e pert_edgeG :*

$\texttt{act\_pert\_vertG} : \texttt{pert\_vertG} := ((\texttt{DelV} \cup \{\texttt{m\_V}(\texttt{vert}L(p))\}) \lhd \texttt{pert\_vertG}) \cup$
$\{$
$\quad \forall p \in PreservV$
$\quad\quad \texttt{m\_V}(\texttt{vert}L(p)) \mapsto \texttt{Pertpreserv\_vert\_}p$
$\quad \forall v \in NewV$
$\quad\quad \texttt{new\_}v \mapsto \texttt{pert\_vert}R(v)$
$\}$
$\texttt{act\_pert\_edgeG} : \texttt{pert\_edgeG} := ((\texttt{DelE} \cup \{\texttt{m\_E}(\texttt{edge}L(p))\}) \lhd \texttt{pert\_edgeG}) \cup$
$\{$
$\quad \forall p \in PreservE$
$\quad\quad \texttt{m\_E}(\texttt{edge}L(p)) \mapsto \texttt{Pertpreserv\_edge\_}p$
$\quad \forall e \in NewE$
$\quad\quad \texttt{new\_}e \mapsto \texttt{pert\_edge}R(e)$
$\}$

end

END

Exemplo 22 (Tradução da aplicação de uma regra fuzzy) *A Figura 30 ilustra a tradução da aplicação da regra $r1$ da GGF Ferrovia (apresentada na Figura 20) para o event-B.*

Nesta regra a aresta $new\_Distancia\_estacaoAL1$ é criada e a aresta $Distancia\_estacaoAL1$ é deletada, isso acontece através da ação act_E. A guarda $grd\_new\_Distancia\_estacaoA$ garante que a aresta é realmente nova no grafo. Essa aresta nova recebe o valor de pertinência $zeroeight$ na ação act_pert_edgeG e tem o mesmo tipo da aresta deletada. Essa ação atualiza ainda os valores das arestas $Distancia\_estacaoBL1$ e $trilhoABL1$, que são preservadas. Nas guardas $grd\_Pertpreserv\_edge\_Distancia\_estacaoB$ e $grd\_Pertpreserv\_edge\_trilhoAB$ são definidos quais valores essas arestas vão receber (maior entre o valor em G e o valor em R). Neste exemplo, a aresta $Distancia\_estacaoBL1$ recebe o valor $zerotwo$ pois esse valor é maior que $zero$ e a aresta $trilhoABL1$ continua com o mesmo valor, devido seu valor não alterar com a aplicação da regra. Devido os vértices manterem sempre o mesmo valor, neste exemplo foram omitidos estas guardas.

EVENTS

Event *r1* $\widehat{=}$

any

m_V
m_E
$DelV$
$PreservV$
$PreservE$
$DelE$
$Dangling$
$new\_Distancia\_estacaoAL1$
$Pertpreserv\_edge\_Distancia\_estacaoB$
$Pertpreserv\_edge\_trilhoAB$

where

grd_mV : $m\_V \in VertL1 \rightarrow VertG$
grd_mE : $m\_E \in EdgeL1 \rightarrow EdgeG$
grd_PreservV : $PreservV = VertG \setminus DelV$
grd_DelV : $DelV = m\_V[VertL1 \setminus \varnothing]$
grd_PreservE: $PreservE = edgeG \setminus DelE$
grd_DelE : $DelE = m\_E[EdgeL1 \setminus \{Distancia\_estacaoAL1\}]$
grd_Dangling : $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$
grd_new_Distancia_estacaoA : $new\_Distancia\_estacaoAL1 \in \mathbb{N} \setminus EdgeG$
grd_vertices : $\forall v \cdot v \in VertL1 \Rightarrow tL1\_V(v) = tG\_V(m\_V(v))$
grd_edges : $\forall e \cdot e \in EdgeL1 \Rightarrow tL1\_E(e) = tG\_E(m\_E(e))$
grd_srctgt : $\forall e \cdot e \in EdgeL1 \Rightarrow m\_V(sourceL1(e)) = sourceG(m\_E(e)) \land$
$m\_V(targetL1(e)) = targetG(m\_E(e))$
grd_IdentV : $DelV \cap m\_V[\{tremL1, estacaoAL1, estacaoBL1\}] = \varnothing$
grd_IdentE : $DelE \cap m\_E[\{trilhoABL1, Distancia\_estacaoBL1\}] = \varnothing$
grd_Pertpreserv_edge_Distancia_estacaoB: $Pertpreserv\_edge\_Distancia\_estacaoB =$
$sup(\{pert\_edgeG(m\_E(Distancia\_estacaoBL1)), zerotwo\})$
grd_Pertpreserv_edge_trilhoAB: $Pertpreserv\_edge\_trilhoAB =$
$sup(\{pert\_edgeG(m\_E(trilhoABL1)), zerotwo\})$
grd_DangCondition : $Dangling = \varnothing$

then

act_V : $VertG := (VertG \setminus DelV) \cup \varnothing$
act_E : $EdgeG := (EdgeG \setminus DelE) \cup \{new\_Distancia\_estacaoAL1\}$
act_src : $sourceG := (DelE \lhd sourceG) \cup \{new\_Distancia\_estacaoAL1 \mapsto m\_V(tremL1)\}$
act_tgt : $targetG := (DelE \lhd targetG) \cup \{new\_Distancia\_estacaoAL1 \mapsto m\_V(estacaoAL1)\}$
act_tV : $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$
act_tE : $tG\_E := (DelE \lhd tG\_E) \cup \{new\_Distancia\_estacaoAL1 \mapsto Distancia\_estacao\}$
act_pert_vertG: $pert\_vertG := (DelV \lhd pert\_vertG) \cup \varnothing$
act_pert_edgeG: $pert\_edgeG := ((DelE \cup \{m\_E(Distancia\_estacaoBL1),$
$m\_E(trilhoABL1)\}) \lhd pert\_edgeG) \cup \{new\_Distancia\_estacaoAL1 \mapsto zeroeight,$
$m\_E(Distancia\_estacaoBL1) \mapsto Pertpreserv\_edge\_Distancia\_estacaoB\},$
$m\_E(trilhoABL1) \mapsto Pertpreserv\_edge\_trilhoAB\}$

end

Figura 30: Tradução da aplicação da regra r1

6 CONCLUSÃO

GGs possibilitam especificar e verificar formalmente sistemas com características complexas, além de ser uma linguagem visual muito intuitiva. Já existem abordagens relacionais para GGs que permitem o uso de provadores de teoremas da ferramenta Rodin usando a linguagem Event-B, como resultado é possível analisar algumas propriedades de sistemas que são especificadas por uma GGs. Neste trabalho foi apresentado um exemplo da GGs Supermercado na plataforma Rodin, utilizando a linguagem Event-B.

Neste trabalho a abordagem relacional de GGs foi estendida para GGFs e para isso foi necessário definir GGFs com tipo, onde a tipagem foi dada através do grafo tipo fuzzy. A tipagem foi necessária para caracterizar todos os tipos e limitar os valores de pertinência dos vértices e das arestas permitidos no sistema. Esses valores são limitados ao valor 1, visto que esse é o maior valor de pertinência permitido pelo intervalo. Para realizar a extensão da GGs para a GGFs surgiu um primeiro problema, que era como diminuir o valor de pertinência de um vértice/aresta pois as restrições da gramática não permitiam fazer isto. Como solução para este problema, foi definido que cada vez que um vértice/aresta precisa ter seu valor de pertinência diminuído então esse vértice/aresta é deletado e ao mesmo tempo um novo vértice/aresta é criado, com os mesmos tipos do vértice/aresta que foi deletado anteriormente, porém com o valor de pertinência diminuído.

Ainda foi definido uma tradução de uma GGF para o Event-B, apresentando o exemplo da GGF ferrovia. Aqui surgiu o segundo problema e a principal dificuldade, que foi especificar esse tipo de gramática no Event-B, devido a ferramenta não possuir a especificação para os números reais, para resolver esse segundo problema foi utilizado uma teoria disponibilizada pelo Rodin, a teoria dos Reais. Esta teoria define operações com os números reais e as operações usadas neste trabalho foram a comparação (leq) e o supremo (sup). A teoria foi modificada, onde foi feito uma extensão dela, pois inicialmente ela tratava apenas dos valores 0($zero$) e 1(one) então foram adicionados novos valores na teoria.

A principal contribuição deste trabalho foi desenvolver uma abordagem relacional

para GGF, com isto será possível especificar e analisar esse tipo de gramática na ferramenta Rodin, utilizando a linguagem Event-B, assim será possível usar o provador de teoremas da ferramenta. Com isso, os desenvolvedores de sistemas podem fazer o uso de uma técnica e ferramenta disponível para gramática de grafos fuzzy a fim de analisar e especificar sistemas.

Para trabalhos futuros podemos investigar um meio de otimizar a representação dos pesos associados a relação de pertinência, visando o aumento da precisão e ainda estender a abordagem da gramática GGF, para que seja possível modelar a incerteza também nas regras de transformação da gramática. Por fim, definir quais os tipos de propriedades da GGF podem ser analisadas com o provador de teoremas da ferramenta Rodin.

REFERÊNCIAS

ABRIAL, J.-R. **The B-book**: Assigning Programs to Meanings. New York, NY, USA: Cambridge University Press, 1996.

ABRIAL, J.-R. **Modeling in Event-B**:system and software engineering. New York, NY, USA: Cambridge University Press, 2010.

ABRIAL, J.-R.; BUTLER, M. **Real Theory.** [Online; accessed 24-January-2017].

ATANASSOV, K.; GARGOV, G. Interval Valued Intuitionistic Fuzzy Sets. **Fuzzy Sets and Systems**, [S.l.], v.31, n.3, p.343–349, 1989.

ATANASSOV, K. T.; PASI, G.; YAGER, R. R.; ATANASSOVA, V. Intuitionistic fuzzy graph interpretations of multi-person multi-criteria decision making. In: EUSFLAT CONF, 2003. **Anais...** University of Applied Sciences at Zittau/Görlitz: Germany, 2003. p.177–182.

BARESI, L.; RAFE, V.; RAHMANI, A. T.; SPOLETINI, P. An Efficient Solution for Model Checking Graph Transformation Systems. **Electron. Notes Theor. Comput. Sci.**, [S.l.], v.213, n.1, p.3–21, 2008.

BEHM, P.; BENOIT, P.; FAIVRE, A.; MEYNADIER, J. M. **Météor**: A Successful Application of B in a Large Project. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999. 369–387p.

BUSTINCE, H.; BARRENECHEA, E.; FERNANDEZ, J.; PAGOLA, M.; MONTERO, J.; GUERRA, C. Contrast of a fuzzy relation. **Information Sciences**, [S.l.], v.180, n.8, p.1326 – 1344, 2010.

BUSTINCE, H.; BARRENECHEA, E.; MOHEDANO, V. INTUITIONISTIC FUZZY IMPLICATION OPERATORS - AN EXPRESSION AND MAIN PROPERTIES. **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems**, [S.l.], v.12, n.03, p.387–406, 2004.

BUTLER, M.; HALLERSTEDE, S. The Rodin formal modelling tool. In: IN PROCEEDINGS OF THE 2007TH INTERNATIOANAL CONFERENCE ON FORMAL METHODS IN INDUSTRY, 2007. **Anais...** [S.l.: s.n.], 2007. p.2–2.

BUTLER, M.; MAAMRIA, I. **Practical Theory Extension in Event-B**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. 67–81p.

CAVALHEIRO, S. A. d. C. **Relational approach of graph grammars**. 2010. Doutorado — Universidade Federal do Rio Grande do Sul - UFRGS.

COSTA, S. A. da; RIBEIRO, L. Verification of graph grammars using a logical approach. **Sci. Comput. Program.**, [S.l.], v.77, n.4, p.480–504, 2012.

CRAIGEN, D.; GERHART, S.; RALSTON, T. **An International Survey of Industrial Applications of Formal Methods**: Volume 1 Purpose, Approach, Analysis, and Conclusions.

DEPLOY. **Event-B and the Rodin Platform.** [Online; accessed 5-January-2017].

DOTTI, F. L.; FOSS, L.; RIBEIRO, L.; SANTOS, O. M. dos. Verification of Distributed Object-Based Systems. In: FMOODS, 2003. **Anais...** Springer, 2003. p.261–275. (Lecture Notes in Computer Science, v.2884).

EHRIG, H.; PFENDER, M.; SCHNEIDER, H. J. Graph-grammars: An Algebraic Approach. In: ANNUAL SYMPOSIUM ON SWITCHING AND AUTOMATA THEORY (SWAT 1973), 14., 1973, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 1973. p.167–180. (SWAT '73).

FERREIRA, A. P. L.; FOSS, L.; RIBEIRO, L. Formal Verification of Object-Oriented Graph Grammars Specifications. **Electron. Notes Theor. Comput. Sci.**, [S.l.], v.175, n.4, p.101–114, 2007.

HONG, Z.; LILI, X. Application of Software Safety Analysis Using Event-B. In: IEEE SEVENTH INTERNATIONAL CONFERENCE ON SOFTWARE SECURITY AND RELIABILITY COMPANION, 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p.137–144.

JIANG, Y. Corrigendum to "Interval-valued intuitionistic fuzzy soft sets and their properties"[Comput. Math. Appl. 60 (2010) 906-918]. **Computers & Mathematics with Applications**, [S.l.], v.61, n.10, p.3179, 2011.

KASTENBERG, H.; RENSINK, A. Model Checking Dynamic States in GROOVE. In: SPIN, 2006. **Anais...** Springer, 2006. p.299–305. (Lecture Notes in Computer Science, v.3925).

MAAMRIA, I.; FATHABADI, A. S. **Theory Plug-in User Manual.** [Online; accessed 24-January-2017].

PARASYUK, I. N.; ERSHOV, S. V. Transformations of fuzzy graphs specified by FD-grammars. **Cybernetics and Systems Analysis**, [S.l.], v.43, n.2, p.266–280, 2007.

PARASYUK, I.; YERSHOV, S. Categorical approach to the construction of fuzzy graph grammars. **Cybernetics and Systems Analysis**, [S.l.], v.42, n.4, p.570–581, 2006.

RENSINK, A. Explicit State Model Checking for Graph Grammars. In: . [S.l.: s.n.], 2008. p.114–132.

RIBEIRO, L.; DOTTI, F. L.; COSTA, S. A. da; DILLENBURG, F. C. Towards Theorem Proving Graph Grammars using Event-B. **ECEASST**, [S.l.], v.30, 2010.

ROZENBERG, G. **Handbook of Graph Grammars and Computing by Graph Transformation**: Volume I. Foundations. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1997.

WAHLS, T. MedicationChecker: Development of a Formally Verified Android Application with EventB2SQL. In: IEEE INTERNATIONAL CONFERENCE ON SOFTWARE QUALITY, RELIABILITY AND SECURITY (QRS), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.307–314.

WANG, Q.; WAHLS, T. **Translating Event-B Machines to Database Applications**. Cham: Springer International Publishing, 2014. 265–270p.

WATKINS, G. The use of fuzzy graph grammars for recognising noisy two-dimensional images. In: NORTH AMERICAN FUZZY INFORMATION PROCESSING, 1996. **Proceedings...** [S.l.: s.n.], 1996. p.415–419.

ZADEH, L. Fuzzy sets. **Information and Control**, [S.l.], v.8, n.3, p.338 – 353, 1965.

ZHANG, G.; ZHAO, L.; ZHANG, W.; YANG, K.; TAN, S. Formalizing Mobile Cloud Service Migration with Event-B. In: IEEE 12TH INTL CONF ON UBIQUITOUS INTELLIGENCE AND COMPUTING AND 2015 IEEE 12TH INTL CONF ON AUTONOMIC AND TRUSTED COMPUTING AND 2015 IEEE 15TH INTL CONF ON SCALABLE COMPUTING AND COMMUNICATIONS AND ITS ASSOCIATED WORKSHOPS (UIC-ATC-SCALCOM), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.991–996.

ANEXO A ESPECIFICAÇÃO DA GRAMÁTICA DE GRAFOS SUPERMERCADO NO EVENT-B

A seguir descrevemos a especificação completa em Event-B da gramática de grafos Supermercado, apresentada nas Figuras 6, 7 e 8.

A.1 Contexto da Gramática de Grafos Supermercado no modelo Event-B

An Event-B Specification of ctxexem
Creation Date: 5Apr2016 @ 10:03:51 AM

CONTEXT ctxexem

SETS

$vertT$ $edgeT$ $vertL1$ $edgeL1$ $vertL2$ $edgeL2$ $vertL3$ $edgeL3$ $vertL4$ $edgeL4$ $vertL5$ $edgeL5$ $vertL6$ $edgeL6$ $vertL7$ $edgeL7$

CONSTANTS

$sourceT$ $targetT$ $Cliente$ $Caixa$ $Compras$ $Cliente\_livre$ Cli_Caixa $Fila$ $Prox$ $Prim$ $Ultim$ $Atend$ $Livre$ $Ocup$ $Fila\_caixa$ $FVazia$ $sourceL1$ $targetL1$ $ClienteL1$ $Cliente\_livreL1$ $tL1\_V$ $tL1\_E$ $sourceL2$ $targetL2$ $tL2\_V$ $tL2\_E$ $ClienteL2$ $CaixaL2$ $ComprasL2$ $FVaziaL2$ $OcupL2$ $targetL3$ $sourceL3$ $tL3\_V$ $tL3\_E$ $CaixaL3$ $Cliente1\_L3$ $Cliente2\_L3$ $ComprasL3$ $UltimL3$ $sourceL4$ $targetL4$ $tL4\_V$ $tL4\_E$ $ClienteL4$ $CaixaL4$ $ComprasL4$ $FVaziaL4$ $LivreL4$ $sourceL5$ $targetL5$ $tL5\_V$ $tL5\_E$ $ClienteL5$ $CaixaL5$ $FilaL5$ $PrimL5$ $UltimL5$ $Fila\_caixaL5$ $LivreL5$ $sourceL6$ $targetL6$ $tL6\_V$ $tL6\_E$ $CaixaL6$ $Cliente\_1L6$ $Cliente\_2L6$ $Fila\_1L6$ $Fila\_2L6$ $ProxL6$ $PrimL6$ $LivreL6$ $sourceL7$ $targetL7$ $tL7\_V$ $tL7\_E$ $CaixaL7$ $ClienteL7$ $Cli\_CaixaL7$ $OcupL7$ $AtendL7$

AXIOMS

axm_vertT : $partition(vertT, \{Cliente\}, \{Caixa\})$

axm_edgeT : $partition(edgeT, \{Compras\}, \{Cliente\_livre\}, \{Cli\_Caixa\}, \{Fila\}, \{Prox\}, \{Prim\}, \{Ultim\}, \{Atend\}, \{Livre\}, \{Ocup\}, \{Fila\_caixa\}, \{FVazia\})$

axm_srcTtype : $sourceT \in edgeT \to vertT$

axm_srcTdef : $partition(sourceT, \{Compras \mapsto Cliente\}, \{Cliente\_livre \mapsto Cliente\}, \{Cli\_Caixa \mapsto Cliente\}, \{Fila \mapsto Cliente\}, \{Prox \mapsto Cliente\}, \{Ultim \mapsto Caixa\}, \{Prim \mapsto Caixa\}, \{Atend \mapsto Cliente\}, \{Livre \mapsto Caixa\}, \{Ocup \mapsto Caixa\}, \{Fila\_caixa \mapsto Caixa\}, \{FVazia \mapsto Caixa\})$

axm_tgtTtype : $targetT \in edgeT \to vertT$

axm_tgtTdef : $partition(targetT, \{Compras \mapsto Cliente\}, \{Cliente\_livre \mapsto Cliente\}, \{Cli\_Caixa \mapsto Cliente\}, \{Fila \mapsto Cliente\}, \{Prox \mapsto Cliente\}, \{Ultim \mapsto Cliente\}, \{Prim \mapsto Cliente\}, \{Atend \mapsto Caixa\}, \{Livre \mapsto Caixa\}, \{Ocup \mapsto Caixa\}, \{Fila\_caixa \mapsto Caixa\}, \{FVazia \mapsto Caixa\})$

axm_vertL1 : $partition(vertL1, \{ClienteL1\})$

axm_edgeL1 : $partition(edgeL1, \{Cliente\_livreL1\})$

axm_srcL1type : $sourceL1 \in edgeL1 \to vertL1$

axm_srcL1def : $partition(sourceL1, \{Cliente\_livreL1 \mapsto ClienteL1\})$

axm_tgtL1type : $targetL1 \in edgeL1 \to vertL1$

axm_tgtL1def : $partition(targetL1, \{Cliente\_livreL1 \mapsto ClienteL1\})$

axm_tVL1type : $tL1\_V \in vertL1 \to vertT$

axm_tVL1def : $partition(tL1\_V, \{ClienteL1 \mapsto Cliente\})$

axm_tEL1type : $tL1\_E \in edgeL1 \to edgeT$

axm_tEL1def : $partition(tL1\_E, \{Cliente\_livreL1 \mapsto Cliente\_livre\})$

axm_vertL2 : $partition(vertL2, \{ClienteL2\}, \{CaixaL2\})$

axm_edgeL2 : $partition(edgeL2, \{ComprasL2\}, \{FVaziaL2\}, \{OcupL2\})$

axm_srcL2type : $sourceL2 \in edgeL2 \to vertL2$

axm_srcL2def : $partition(sourceL2, \{ComprasL2 \mapsto ClienteL2\}, \{FVaziaL2 \mapsto CaixaL2\}, \{OcupL2 \mapsto CaixaL2\})$

axm_tgtL2type : $targetL2 \in edgeL2 \to vertL2$

axm_tgtL2def : $partition(targetL2, \{ComprasL2 \mapsto ClienteL2\}, \{FVaziaL2 \mapsto CaixaL2\}, \{OcupL2 \mapsto CaixaL2\})$

axm_tVL2type : $tL2\_V \in vertL2 \to vertT$

axm_tVL2def : $partition(tL2\_V, \{ClienteL2 \mapsto Cliente\}, \{CaixaL2 \mapsto Caixa\})$

axm_tEL2type : $tL2\_E \in edgeL2 \to edgeT$

axm_tEL2def : $partition(tL2\_E, \{ComprasL2 \mapsto Compras\}, \{FVaziaL2 \mapsto FVazia\}, \{OcupL2 \mapsto Ocup\})$

axm_vertL3 : $partition(vertL3, \{Cliente1\_L3\}, \{Cliente2\_L3\}, \{CaixaL3\})$

axm_edgeL3 : $partition(edgeL3, \{ComprasL3\}, \{UltimL3\})$

axm_srcL3type : $sourceL3 \in edgeL3 \rightarrow vertL3$

axm_srcL3def : $partition(sourceL3, \{ComprasL3 \mapsto Cliente2\_L3\}, \{UltimL3 \mapsto CaixaL3\})$

axm_tgtL3type : $targetL3 \in edgeL3 \rightarrow vertL3$

axm_tgtL3def : $partition(targetL3, \{ComprasL3 \mapsto Cliente2\_L3\}, \{UltimL3 \mapsto Cliente1\_L3\})$

axm_tVL3type : $tL3\_V \in vertL3 \rightarrow vertT$

axm_tVL3def : $partition(tL3\_V, \{CaixaL3 \mapsto Caixa\}, \{Cliente1\_L3 \mapsto Cliente\}, \{Cliente2\_L3 \mapsto Cliente\})$

axm_tEL3type : $tL3\_E \in edgeL3 \rightarrow edgeT$

axm_tEL3def : $partition(tL3\_E, \{ComprasL3 \mapsto Compras\}, \{UltimL3 \mapsto Ultim\})$

axm_vertL4 : $partition(vertL4, \{ClienteL4\}, \{CaixaL4\})$

axm_edgeL4 : $partition(edgeL4, \{ComprasL4\}, \{FVaziaL4\}, \{LivreL4\})$

axm_srcL4type : $sourceL4 \in edgeL4 \rightarrow vertL4$

axm_srcL4def : $partition(sourceL4, \{ComprasL4 \mapsto ClienteL4\}, \{FVaziaL4 \mapsto CaixaL4\}, \{LivreL4 \mapsto CaixaL4\})$

axm_tgtL4type : $targetL4 \in edgeL4 \rightarrow vertL4$

axm_tgtL4def : $partition(targetL4, \{ComprasL4 \mapsto ClienteL4\}, \{FVaziaL4 \mapsto CaixaL4\}, \{LivreL4 \mapsto CaixaL4\})$

axm_tVL4type : $tL4\_V \in vertL4 \rightarrow vertT$

axm_tVL4def : $partition(tL4\_V, \{ClienteL4 \mapsto Cliente\}, \{CaixaL4 \mapsto Caixa\})$

axm_tEL4type : $tL4\_E \in edgeL4 \rightarrow edgeT$

axm_tEL4def : $partition(tL4\_E, \{ComprasL4 \mapsto Compras\}, \{FVaziaL4 \mapsto FVazia\}, \{LivreL4 \mapsto Livre\})$

axm_vertL5 : $partition(vertL5, \{ClienteL5\}, \{CaixaL5\})$

axm_edgeL5 : $partition(edgeL5, \{FilaL5\}, \{PrimL5\}, \{UltimL5\}, \{Fila\_caixaL5\}, \{LivreL5\})$

axm_srcL5type : $sourceL5 \in edgeL5 \rightarrow vertL5$

axm_srcL5def : $partition(sourceL5, \{FilaL5 \mapsto ClienteL5\}, \{PrimL5 \mapsto ClienteL5\}, \{UltimL5 \mapsto ClienteL5\}, \{Fila\_caixaL5 \mapsto CaixaL5\}, \{LivreL5 \mapsto CaixaL5\})$

axm_tgtL5type : $targetL5 \in edgeL5 \rightarrow vertL5$

axm_tgtL5def : $partition(targetL5, \{FilaL5 \mapsto ClienteL5\}, \{PrimL5 \mapsto CaixaL5\}, \{UltimL5 \mapsto CaixaL5\}, \{Fila\_caixaL5 \mapsto CaixaL5\}, \{LivreL5 \mapsto CaixaL5\})$

axm_tVL5type : $tL5\_V \in vertL5 \rightarrow vertT$

axm_tVL5def : $partition(tL5\_V, \{ClienteL5 \mapsto Cliente\}, \{CaixaL5 \mapsto Caixa\})$

axm_tEL5type : $tL5\_E \in edgeL5 \rightarrow edgeT$

axm_tEL5def : $partition(tL5\_E, \{FilaL5 \mapsto Fila\}, \{PrimL5 \mapsto Prim\}, \{UltimL5 \mapsto Ultim\}, \{Fila\_caixaL5 \mapsto Fila\_caixa\}, \{LivreL5 \mapsto Livre\})$

axm_vertL6 : $partition(vertL6, \{Cliente\_1L6\}, \{Cliente\_2L6\}, \{CaixaL6\})$

axm_edgeL6 : $partition(edgeL6, \{Fila\_1L6\}, \{Fila\_2L6\}, \{ProxL6\}, \{PrimL6\}, \{LivreL6\})$

axm_srcL6type : $sourceL6 \in edgeL6 \rightarrow vertL6$

axm_srcL6def : $partition(sourceL6, \{Fila\_1L6 \mapsto Cliente\_1L6\}, \{Fila\_2L6 \mapsto Cliente\_2L6\}, \{ProxL6 \mapsto Cliente\_1L6\}, \{PrimL6 \mapsto CaixaL6\}, \{LivreL6 \mapsto CaixaL6\})$

axm_tgtL6type : $targetL6 \in edgeL6 \rightarrow vertL6$

axm_tgtL6def : $partition(targetL6, \{Fila\_1L6 \mapsto Cliente\_1L6\}, \{Fila\_2L6 \mapsto Cliente\_2L6\}, \{ProxL6 \mapsto Cliente\_2L6\}, \{PrimL6 \mapsto Cliente\_1L6\}, \{LivreL6 \mapsto CaixaL6\})$

axm_tVL6type : $tL6\_V \in vertL6 \rightarrow vertT$

axm_tVL6def : $partition(tL6\_V, \{CaixaL6 \mapsto Caixa\}, \{Cliente\_1L6 \mapsto Cliente\}, \{Cliente\_2L6 \mapsto Cliente\})$

axm_tEL6type : $tL6\_E \in edgeL6 \rightarrow edgeT$

axm_tEL6def : $partition(tL6\_E, \{Fila\_1L6 \mapsto Fila\}, \{Fila\_2L6 \mapsto Fila\}, \{ProxL6 \mapsto Prox\}, \{PrimL6 \mapsto Prim\}, \{LivreL6 \mapsto Livre\})$

axm_vertL7 : $partition(vertL7, \{ClienteL7\}, \{CaixaL7\})$

axm_edgeL7 : $partition(edgeL7, \{Cli\_CaixaL7\}, \{AtendL7\}, \{OcupL7\})$

axm_srcL7type : $sourceL7 \in edgeL7 \rightarrow vertL7$

axm_srcL7def : $partition(sourceL7, \{Cli\_CaixaL7 \mapsto ClienteL7\}, \{AtendL7 \mapsto ClienteL7\}, \{OcupL7 \mapsto CaixaL7\})$

axm_tgtL7type : $targetL7 \in edgeL7 \rightarrow vertL7$

axm_tgtL7def : $partition(targetL7, \{Cli\_CaixaL7 \mapsto ClienteL7\}, \{AtendL7 \mapsto CaixaL7\}, \{OcupL7 \mapsto CaixaL7\})$

axm_tVL7type : $tL7\_V \in vertL7 \rightarrow vertT$

axm_tVL7def : $partition(tL7\_V, \{CaixaL7 \mapsto Caixa\}, \{ClienteL7 \mapsto Cliente\})$

axm_tEL7type : $tL7\_E \in edgeL7 \rightarrow edgeT$

axm_tEL7def : $partition(tL7\_E, \{Cli\_CaixaL7 \mapsto Cli\_Caixa\}, \{AtendL7 \mapsto Atend\}, \{OcupL7 \mapsto Ocup\})$

END

A.2 Máquina da Gramática de Grafos Supermercado no modelo Event-B

An Event-B Specification of mchexem
Creation Date: 5Apr2016 @ 10:03:51 AM

MACHINE mchexem

SEES ctxexem

VARIABLES

$vertG \ \ edgeG \ \ sourceG \ \ targetG \ \ tG\_V \ \ tG\_E$

INVARIANTS

inv_vertG_type : $vertG \in \mathbb{P}(\mathbb{N})$

inv_edgeG_type : $edgeG \in \mathbb{P}(\mathbb{N})$

inv_sourceG_type : $sourceG \in edgeG \to vertG$

inv_targetG_type : $targetG \in edgeG \to vertG$

inv_tVG_type : $tG\_V \in vertG \to vertT$

inv_tEG_type : $tG\_E \in edgeG \to edgeT$

EVENTS

Initialisation

begin

act_vertG : $vertG := \{1, 2, 3, 4\}$

act_edgeG : $edgeG := \{1, 2, 3, 4, 5, 6, 7, 8\}$

act_srcG : $sourceG := \{1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 3, 4 \mapsto 3, 5 \mapsto 3, 6 \mapsto 4, 7 \mapsto 4, 8 \mapsto 2\}$

act_tgtG : $targetG := \{1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 1, 4 \mapsto 3, 5 \mapsto 3, 6 \mapsto 4, 7 \mapsto 4, 8 \mapsto 2\}$

act_tG_V : $tG\_V := \{1 \mapsto Cliente, 2 \mapsto Cliente, 3 \mapsto Caixa, 4 \mapsto Caixa\}$

act_tG_E : $tG\_E := \{1 \mapsto Fila, 2 \mapsto Ultim, 3 \mapsto Prim, 4 \mapsto Fila\_caixa, 5 \mapsto Livre, 6 \mapsto Livre, 7 \mapsto FVazia, 8 \mapsto Cliente\_livre\}$

end

Event *Inicio* $\widehat{=}$

any

$m\_V \ \ m\_E \ \ DelV \ \ PreservV \ \ DelE \ \ Dangling \ \ new\_Compras$

where

grd_mV : $m\_V \in vertL1 \to vertG$

grd_mE : $m\_E \in edgeL1 \to edgeG$

grd_DelV : $DelV = m\_V[\varnothing]$

grd_PreV : $PreservV = vertG \setminus DelV$

grd_DelE : $DelE = m\_E[\{Cliente\_livreL1\}]$

grd_Dang : $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Compras : $new\_Compras \in \mathbb{N} \setminus edgeG$

grd_vertices : $\forall v \cdot v \in vertL1 \Rightarrow tL1\_V(v) = tG\_V(m\_V(v))$

grd_edges : $\forall e \cdot e \in edgeL1 \Rightarrow tL1\_E(e) = tG\_E(m\_E(e))$

grd_srctgt : $\forall e \cdot e \in edgeL1 \Rightarrow m\_V(sourceL1(e)) = sourceG(m\_E(e)) \land m\_V(targetL1(e)) = targetG(m\_E(e))$

grd_IdentV : $DelV \cap m\_V[\{ClienteL1\}] = \varnothing$

grd_IdentE : $DelE \cap m\_E[\varnothing] = \varnothing$

grd_DangCondition : $Dangling = \varnothing$

then

act_V : $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E : $edgeG := (edgeG \setminus DelE) \cup \{new\_Compras\}$

act_src : $sourceG := (DelE \lhd sourceG) \cup \{new\_Compras \mapsto m\_V(ClienteL1)\}$

act_tgt : $targetG := (DelE \lhd targetG) \cup \{new\_Compras \mapsto m\_V(ClienteL1)\}$

act_tV : $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE : $tG\_E := (DelV \lhd tG\_E) \cup \varnothing$

end

Event *Entrar1* $\widehat{=}$

any

$m\_V \;\; m\_E \;\; DelV \;\; PreservV \;\; DelE \;\; Dangling \;\; new\_Fila \;\; new\_Prim \;\; new\_Ultim \;\; new\_Fila\_caixa$

where

grd_mV : $m\_V \in vertL2 \rightarrow vertG$

grd_mE : $m\_E \in edgeL2 \rightarrow edgeG$

grd_DelV : $DelV = m\_V[\varnothing]$

grd_PreV : $PreservV = vertG \setminus DelV$

grd_DelE : $DelE = m\_E[edgeL2 \setminus \{ComprasL2, FVaziaL2\}]$

grd_Dang : $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Fila : $new\_Fila \in \mathbb{N} \setminus edgeG$

grd_new_Prim : $new\_Prim \in \mathbb{N} \setminus edgeG$

grd_new_Ultim : $new\_Ultim \in \mathbb{N} \setminus edgeG$

grd_new_Fila_caixa : $new\_Fila\_caixa \in \mathbb{N} \setminus edgeG$

grd_vertices : $\forall v \cdot v \in vertL2 \Rightarrow tL2\_V(v) = tG\_V(m\_V(v))$

grd_edges : $\forall e \cdot e \in edgeL2 \Rightarrow tL2\_E(e) = tG\_E(m\_E(e))$

grd_srctgt : $\forall e \cdot e \in edgeL2 \Rightarrow m\_V(sourceL2(e)) = sourceG(m\_E(e)) \land m\_V(targetL2(e)) = targetG(m\_E(e))$

grd_IdentV : $DelV \cap m\_V[\{ClienteL2, CaixaL2\}] = \varnothing$

grd_IdentE : $DelE \cap m\_E[\{OcupL2\}] = \varnothing$

grd_DangCondition : $Dangling = \varnothing$

then

act_V : $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E : $edgeG := (edgeG \setminus DelE) \cup \{new\_Fila, new\_Fila\_caixa, new\_Prim, new\_Ultim\}$

act_src : $sourceG := (DelE \lhd sourceG) \cup \{new\_Fila \mapsto m\_V(ClienteL2), new\_Fila\_caixa \mapsto m\_V(CaixaL2), new\_Prim \mapsto m\_V(CaixaL2), new\_Ultim \mapsto m\_V(CaixaL2)\}$

act_tgt : $targetG := (DelE \lhd targetG) \cup \{new\_Fila \mapsto m\_V(ClienteL2), new\_Fila\_caixa \mapsto m\_V(CaixaL2), new\_Prim \mapsto m\_V(ClienteL2), new\_Ultim \mapsto m\_V(ClienteL2)\}$

act_tV : $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE : $tG\_E := (DelV \lhd tG\_E) \cup \{new\_Fila \mapsto Fila, new\_Fila\_caixa \mapsto Fila\_caixa, new\_Prim \mapsto Prim, new\_Ultim \mapsto Ultim\}$

end

Event *Entrar2* $\widehat{=}$

any

$m\_V \;\; m\_E \;\; DelV \;\; PreservV \;\; DelE \;\; Dangling \;\; new\_Prox \;\; new\_Fila \;\; new\_Ultim$

where

grd_mV : $m\_V \in vertL3 \to vertG$

grd_mE : $m\_E \in edgeL3 \to edgeG$

grd_DelV : $DelV = m\_V[\varnothing]$

grd_PreV : $PreservV = vertG \setminus DelV$

grd_DelE : $DelE = m\_E[edgeL3 \setminus \{UltimL3, ComprasL3\}]$

grd_Dang : $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Prox : $new\_Prox \in \mathbb{N} \setminus vertG$

grd_new_Fila : $new\_Fila \in \mathbb{N} \setminus edgeG$

grd_new_Ultim : $new\_Ultim \in \mathbb{N} \setminus edgeG$

grd_vertices : $\forall v \cdot v \in vertL3 \Rightarrow tL3\_V(v) = tG\_V(m\_V(v))$

grd_edges : $\forall e \cdot e \in edgeL3 \Rightarrow tL3\_E(e) = tG\_E(m\_E(e))$

grd_srctgt : $\forall e \cdot e \in edgeL3 \Rightarrow m\_V(sourceL3(e)) = sourceG(m\_E(e)) \land m\_V(targetL3(e)) = targetG(m\_E(e))$

grd_IdentV : $DelV \cap m\_V[\{Cliente1\_L3, Cliente2\_L3, CaixaL3\}] = \varnothing$

grd_IdentE : $DelE \cap m\_E[\varnothing] = \varnothing$

grd_DangCondition : $Dangling = \varnothing$

then

act_V : $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E : $edgeG := (edgeG \setminus DelE) \cup \{new\_Fila, new\_Ultim, new\_Prox\}$

act_src : $sourceG := (DelE \lhd sourceG) \cup \{new\_Fila \mapsto m\_V(Cliente2\_L3), new\_Prox \mapsto m\_V(Cliente1\_L3), new\_Ultim \mapsto m\_V(CaixaL3)\}$

act_tgt : $targetG := (DelE \lhd targetG) \cup \{new\_Fila \mapsto m\_V(Cliente2\_L3), new\_Prox \mapsto m\_V(Cliente2\_L3), new\_Ultim \mapsto m\_V(Cliente2\_L3)\}$

act_tV : $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE : $tG\_E := (DelV \lhd tG\_E) \cup \{new\_Fila \mapsto Fila, new\_Ultim \mapsto Ultim, new\_Prox \mapsto Prox\}$

end

Event *Pagar1* $\widehat{=}$

any

$m\_V \;\; m\_E \;\; DelV \;\; PreservV \;\; DelE \;\; Dangling \;\; new\_Cli\_Caixa \;\; new\_Atend \;\; new\_Ocup$

where

grd_mV : $m\_V \in vertL4 \rightarrow vertG$

grd_mE : $m\_E \in edgeL4 \rightarrow edgeG$

grd_DelV : $DelV = m\_V[\varnothing]$

grd_PreV : $PreservV = vertG \setminus DelV$

grd_DelE : $DelE = m\_E[edgeL4 \setminus \{ComprasL4, LivreL4\}]$

grd_Dang : $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Cli_Caixa : $new\_Cli\_Caixa \in \mathbb{N} \setminus edgeG$

grd_new_Atend : $new\_Atend \in \mathbb{N} \setminus edgeG$

grd_new_Ocup : $new\_Ocup \in \mathbb{N} \setminus edgeG$

grd_vertices : $\forall v \cdot v \in vertL4 \Rightarrow tL4\_V(v) = tG\_V(m\_V(v))$

grd_edges : $\forall e \cdot e \in edgeL4 \Rightarrow tL4\_E(e) = tG\_E(m\_E(e))$

grd_srctgt : $\forall e \cdot e \in edgeL4 \Rightarrow m\_V(sourceL4(e)) = sourceG(m\_E(e)) \wedge m\_V(targetL4(e)) = targetG(m\_E(e))$

grd_IdentV : $DelV \cap m\_V[\{ClienteL4, CaixaL4\}] = \varnothing$

grd_IdentE : $DelE \cap m\_E[\{FVaziaL4\}] = \varnothing$

grd_DangCondition : $Dangling = \varnothing$

then

act_V : $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E : $edgeG := (edgeG \setminus DelE) \cup \{new\_Cli\_Caixa, new\_Atend, new\_Ocup\}$

act_src : $sourceG := (DelE \lhd sourceG) \cup \{new\_Cli\_Caixa \mapsto m\_V(ClienteL4), new\_Atend \mapsto m\_V(ClienteL4), new\_Ocup \mapsto m\_V(CaixaL4)\}$

act_tgt : $targetG := (DelE \lhd targetG) \cup \{new\_Cli\_Caixa \mapsto m\_V(ClienteL4), new\_Atend \mapsto m\_V(CaixaL4), new\_Ocup \mapsto m\_V(CaixaL4)\}$

act_tV : $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE : $tG\_E := (DelV \lhd tG\_E) \cup \{new\_Cli\_Caixa \mapsto Cli\_Caixa, new\_Atend \mapsto Atend, new\_Ocup \mapsto Ocup\}$

end

Event *Pagar2* $\widehat{=}$

any

$m\_V \ m\_E \ DelV \ PreservV \ DelE \ Dangling \ new\_Cli\_Caixa$
$new\_Atend \ new\_Ocup \ new\_FVazia$

where

grd_mV : $m\_V \in vertL5 \rightarrow vertG$

grd_mE : $m\_E \in edgeL5 \rightarrow edgeG$

grd_DelV : $DelV = m\_V[\varnothing]$

grd_PreV : $PreservV = vertG \setminus DelV$

grd_DelE : $DelE = m\_E[edgeL5 \setminus \{FilaL5, PrimL5, UltimL5, Fila\_caixaL5, LivreL5\}]$

grd_Dang : $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Cli_Caixa : $new\_Cli\_Caixa \in \mathbb{N} \setminus edgeG$

grd_new_Atend : $new\_Atend \in \mathbb{N} \setminus edgeG$

grd_new_Ocup : $new\_Ocup \in \mathbb{N} \setminus edgeG$

grd_new_FVazia : $new\_FVazia \in \mathbb{N} \setminus edgeG$

grd_vertices : $\forall v \cdot v \in vertL5 \Rightarrow tL5\_V(v) = tG\_V(m\_V(v))$

grd_edges : $\forall e \cdot e \in edgeL5 \Rightarrow tL5\_E(e) = tG\_E(m\_E(e))$

grd_srctgt : $\forall e \cdot e \in edgeL5 \Rightarrow m\_V(sourceL5(e)) = sourceG(m\_E(e)) \land m\_V(targetL5(e)) = targetG(m\_E(e))$

grd_IdentV : $DelV \cap m\_V[\{ClienteL5, CaixaL5\}] = \varnothing$

grd_IdentE : $DelE \cap m\_E[\varnothing] = \varnothing$

grd_DangCondition : $Dangling = \varnothing$

then

act_V : $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E : $edgeG := (edgeG \setminus DelE) \cup \{new\_Cli\_Caixa, new\_Atend, new\_FVazia, new\_Ocup\}$

act_src : $sourceG := (DelE \lhd sourceG) \cup \{new\_Cli\_Caixa \mapsto m\_V(ClienteL5), new\_Atend \mapsto m\_V(ClienteL5), new\_Ocup \mapsto m\_V(CaixaL5), new\_FVazia \mapsto m\_V(CaixaL5)\}$

act_tgt : $targetG := (DelE \lhd targetG) \cup \{new\_Cli\_Caixa \mapsto m\_V(ClienteL5), new\_Atend \mapsto m\_V(CaixaL5), new\_Ocup \mapsto m\_V(CaixaL5), new\_FVazia \mapsto m\_V(CaixaL5)\}$

act_tV : $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE : $tG\_E := (DelV \lhd tG\_E) \cup \{new\_Cli\_Caixa \mapsto Cli\_Caixa, new\_Atend \mapsto Atend, new\_FVazia \mapsto FVazia, new\_Ocup \mapsto Ocup\}$

end

Event *Pagar3* $\widehat{=}$

any

$m\_V \ m\_E \ DelV \ PreservV \ DelE \ Dangling \ new\_Ocup \ new\_Prim$
$new\_Atend \ new\_Cli\_Caixa$

where

grd_mV : $m\_V \in vertL6 \rightarrow vertG$

grd_mE : $m\_E \in edgeL6 \rightarrow edgeG$

grd_DelV : $DelV = m\_V[\varnothing]$

grd_PreV : $PreservV = vertG \setminus DelV$

grd_DelE : $DelE = m\_E[edgeL6 \setminus \{ProxL6, Fila\_1L6, PrimL6, LivreL6\}]$

grd_Dang : $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Ocup : $new\_Ocup \in \mathbb{N} \setminus edgeG$

grd_new_Prim : $new\_Prim \in \mathbb{N} \setminus edgeG$

grd_new_Atend : $new\_Atend \in \mathbb{N} \setminus edgeG$

grd_new_Cli_Caixa : $new\_Cli\_Caixa \in \mathbb{N} \setminus edgeG$

grd_vertices : $\forall v \cdot v \in vertL6 \Rightarrow tL6\_V(v) = tG\_V(m\_V(v))$

grd_edges : $\forall e \cdot e \in edgeL6 \Rightarrow tL6\_E(e) = tG\_E(m\_E(e))$

grd_srctgt : $\forall e \cdot e \in edgeL6 \Rightarrow m\_V(sourceL6(e)) = sourceG(m\_E(e)) \land m\_V(targetL6(e)) = targetG(m\_E(e))$

grd_IdentV : $DelV \cap m\_V[\{Cliente\_1L6, Cliente\_2L6, CaixaL6\}] = \varnothing$

grd_IdentE : $DelE \cap m\_E[\{Fila\_2L6\}] = \varnothing$

grd_DangCondition : $Dangling = \varnothing$

then

act_V : $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E : $edgeG := (edgeG \setminus DelE) \cup \{new\_Cli\_Caixa, new\_Atend, new\_Prim, new\_Ocup\}$

act_src : $sourceG := (DelE \lhd sourceG) \cup \{new\_Cli\_Caixa \mapsto m\_V(Cliente\_1L6), new\_Atend \mapsto m\_V(Cliente\_1L6), new\_Prim \mapsto m\_V(CaixaL6), new\_Ocup \mapsto m\_V(CaixaL6)\}$

act_tgt : $targetG := (DelE \lhd targetG) \cup \{new\_Cli\_Caixa \mapsto m\_V(Cliente\_1L6), new\_Atend \mapsto m\_V(CaixaL6), new\_Prim \mapsto m\_V(Cliente\_2L6), new\_Ocup \mapsto m\_V(CaixaL6)\}$

act_tV : $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE : $tG\_E := (DelV \lhd tG\_E) \cup \{new\_Cli\_Caixa \mapsto Cli\_Caixa, new\_Atend \mapsto Atend, new\_Prim \mapsto Prim, new\_Ocup \mapsto Ocup\}$

end

Event *Pago* $\widehat{=}$

any

m_V m_E $DelV$ $PreservV$ $DelE$ $Dangling$ $new\_Cliente\_livre$ new_Livre

where

grd_mV : $m\_V \in vertL7 \rightarrow vertG$

grd_mE : $m\_E \in edgeL7 \rightarrow edgeG$

grd_DelV : $DelV = m\_V[\varnothing]$

grd_PreV : $PreservV = vertG \setminus DelV$

grd_DelE : $DelE = m\_E[edgeL7 \setminus \{Cli\_CaixaL7, AtendL7, OcupL7\}]$

grd_Dang : $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Cliente_livre : $new\_Cliente\_livre \in \mathbb{N} \setminus edgeG$

grd_new_Livre : $new\_Livre \in \mathbb{N} \setminus edgeG$

grd_vertices : $\forall v \cdot v \in vertL7 \Rightarrow tL7\_V(v) = tG\_V(m\_V(v))$

grd_edges : $\forall e \cdot e \in edgeL7 \Rightarrow tL7\_E(e) = tG\_E(m\_E(e))$

grd_srctgt : $\forall e \cdot e \in edgeL7 \Rightarrow m\_V(sourceL7(e)) = sourceG(m\_E(e)) \land m\_V(targetL7(e)) = targetG(m\_E(e))$

grd_IdentV : $DelV \cap m\_V[\{ClienteL7, CaixaL7\}] = \varnothing$

grd_IdentE : $DelE \cap m\_E[\varnothing] = \varnothing$

grd_DangCondition : $Dangling = \varnothing$

then

act_V : $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E : $edgeG := (edgeG \setminus DelE) \cup \{new\_Cliente\_livre, new\_Livre\}$

act_src : $sourceG := (DelE \lhd sourceG) \cup \{new\_Cliente\_livre \mapsto m\_V(ClienteL7), new\_Livre \mapsto m\_V(CaixaL7)\}$

act_tgt : $targetG := (DelE \lhd targetG) \cup \{new\_Cliente\_livre \mapsto m\_V(ClienteL7), new\_Livre \mapsto m\_V(CaixaL7)\}$

act_tV : $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE : $tG\_E := (DelV \lhd tG\_E) \cup \{new\_Cliente\_livre \mapsto Cliente\_livre, new\_Livre \mapsto Livre\}$

end

END

ANEXO B ESPECIFICAÇÃO DA GRAMÁTICA DE GRAFOS FUZZY FERROVIA NO EVENT-B

A seguir descrevemos a especificação completa em Event-B da gramática de grafos fuzzy ferrovia, apresentada nas Figuras 18, 19 e 20.

B.1 Contexto da Gramática de Grafos fuzzy ferrovia no modelo Event-B

An Event-B Specification of ctxGGF
Creation Date: 10Jan2017 @ 14:03:51 PM

CONTEXT ctxGGF

SETS

$vertT$ $edgeT$ $vertL1$ $edgeL1$ $vertL2$ $edgeL2$ $vertL3$ $edgeL3$ $vertL4$ $edgeL4$
$vertL5$ $edgeL5$ $vertL6$ $edgeL6$ $vertL7$ $edgeL7$ $vertL8$ $edgeL8$ $vertL9$ $edgeL9$

CONSTANTS

$trem$ $estacao$ $trilho$ $Distancia\_estacao$ $targetT$ $sourceT$ $pert\_edgeT$
$pert\_vertT$ $estacaoAL1$ $tremL1$ $Distancia\_estacaoAL1$ $Distancia\_estacaoBL1$
$trilhoABL1$ $pert\_edgeL1$ $sourceL1$ $targetL1$ $pert\_vertL1$ $estacaoBL1$ $tL1\_V$
$tL1\_E$ $tremL2$ $estacaoAL2$ $estacaoBL2$ $Distancia\_estacaoAL2$ $trilhoABL2$
$Distancia\_estacaoBL2$ $pert\_edgeL2$ $pert\_vertL2$ $sourceL2$ $targetL2$ $tL2\_V$
$tL2\_E$ $estacaoAL3$ $estacaoBL3$ $tremL3$ $Distancia\_estacaoAL3$ $trilhoABL3$
$Distancia\_estacaoBL3$ $pert\_edgeL3$ $pert\_vertL3$ $sourceL3$ $targetL3$
$tL3\_V$ $tL3\_E$ $estacaoAL4$ $estacaoBL4$ $estacaoDL4$ $tremL4$ $trilhoABL4$
$Distancia\_estacaoBL4$ $Distancia\_estacaoAL4$ $trilhoBDL4$ $pert\_edgeL4$
$pert\_vertL4$ $sourceL4$ $targetL4$ $tL4\_V$ $tL4\_E$ $estacaoCL5$ $estacaoEL5$
$tremL5$ $Distancia\_estacaoCL5$ $Distancia\_estacaoEL5$ $trilhoCEL5$
$pert\_vertL5$ $pert\_edgeL5$ $sourceL5$ $targetL5$ $tL5\_V$ $tL5\_E$ $estacaoCL6$

$estacaoEL6$ $tremL6$ $Distancia\_estacaoCL6$ $Distancia\_estacaoEL6$
$trilhoCEL6$ $pert\_vertL6$ $sourceL6$ $targetL6$ $pert\_edgeL6$ $tL6\_V$ $tL6\_E$
$estacaoCL7$ $estacaoEL7$ $estacaoJL7$ $tremL7$ $Distancia\_estacaoCL7$
$Distancia\_estacaoEL7$ $trilhoCEL7$ $trilhoEJL7$ $pert\_vertL7$ $pert\_edgeL7$
$sourceL7$ $targetL7$ $tL7\_V$ $tL7\_E$ $estacaoEL8$ $estacaoJL8$ $tremL8$
$Distancia\_estacaoEL8$ $Distancia\_estacaoJL8$ $trilhoEJL8$ $pert\_vertL8$
$pert\_edgeL8$ $sourceL8$ $targetL8$ $tL8\_V$ $tL8\_E$ $estacaoEL9$ $estacaoJL9$
$tremL9$ $Distancia\_estacaoEL9$ $Distancia\_estacaoJL9$ $trilhoEJL9$
$pert\_vertL9$ $pert\_edgeL9$ $sourceL9$ $targetL9$ $tL9\_V$ $tL9\_E$

AXIOMS

axm_vertT: $partition(vertT, \{trem\}, \{estacao\})$

axm_edgeT: $partition(edgeT, \{trilho\}, \{Distancia\_estacao\})$

axm_srcT_type: $sourceT \in edgeT \to vertT$

axm_srcT_def: $partition(sourceT, \{trilho \mapsto estacao\}, \{Distancia\_estacao \mapsto trem\})$

axm_tgtT_type: $targetT \in EdgeT \to VertT$

axm_tgtT_def: $partition(targetT, \{trilho \mapsto estacao\}, \{Distancia\_estacao \mapsto estacao\})$

axm_pertinenciavertT_type: $pert\_vertT \in vertT \to REAL$

axm_pertinenciavertT_def: $partition(pert\_vertT, \{trilho \mapsto one\},$
$\{Distancia\_estacao \mapsto one\})$

axm_pertinenciaedgeT_type: $pert\_edgeT \in edgeT \to REAL$

axm_pertinenciaedgeT_def: $partition(pert\_edgeT, \{trem \mapsto one\}, \{estacao \mapsto one\})$

axm_vertL1 : $partition(vertL1, \{tremL1\}, \{estacaoAL1\}, \{estacaoBL1\})$

axm_edgeL1 : $partition(edgeL1, \{Distancia\_estacaoAL1\},$
$\{Distancia\_estacaoBL1\}, \{trilhoABL1\})$

axm_srcL1_type : $sourceL1 \in edgeL1 \to vertL1$

axm_srcL1_def : $partition(sourceL1, \{trilhoABL1 \mapsto estacaoAL1\},$
$\{Distancia\_estacaoAL1 \mapsto tremL1\}, \{Distancia\_estacaoBL1 \mapsto tremL1\})$

axm_tgtL1_type : $targetL1 \in edgeL1 \to vertL1$

axm_tgtL1_def : $partition(targetL1, \{trilhoABL1 \mapsto estacaoBL1\},$
$\{Distancia\_estacaoAL1 \mapsto estacaoAL1\}, \{Distancia\_estacaoBL1 \mapsto estacaoBL1\})$

axm_tVL1_type : $tL1\_V \in vertL1 \to vertT$

axm_tVL1_def : $partition(tL1\_V, \{tremL1 \mapsto trem\}, \{estacaoAL1 \mapsto estacao\},$
$\{estacaoBL1 \mapsto estacao\})$

axm_tEL1_type : $tL1\_E \in edgeL1 \to edgeT$

axm_tEL1_def : $partition(tL1\_E, \{Distancia\_estacaoAL1 \mapsto Distancia\_estacao\},$
$\{Distancia\_estacaoBL1 \mapsto Distancia\_estacao\}, \{trilhoABL1 \mapsto trilho\})$

axm_pertinenciavertL1_type: $pert\_vertL1 \in vertL1 \rightarrow REAL$

axm_pertinenciavertL1_def: $partition(pert\_vertL1, \{tremL1 \mapsto one\}, \{estacaoAL1 \mapsto one\}, \{estacaoBL1 \mapsto one\})$

axm_pertinenciaedgeL1_type: $pert\_edgeL1 \in edgeL1 \rightarrow REAL$

axm_pertinenciaedgeL1_def: $partition(pert\_edgeL1, \{Distancia\_estacaoAL1 \mapsto one\}, \{Distancia\_estacaoBL1 \mapsto zero\}, \{trilhoABL1 \mapsto zerotwo\})$

axm_vertL2 : $partition(vertL2, \{tremL2\}, \{estacaoAL2\}, \{estacaoBL2\})$

axm_edgeL2 : $partition(edgeL2, \{Distancia\_estacaoAL2\}, \{Distancia\_estacaoBL2\}, \{trilhoABL2\})$

axm_srcL2_type : $sourceL2 \in edgeL2 \rightarrow vertL2$

axm_srcL2_def : $partition(sourceL2, \{trilhoABL2 \mapsto estacaoAL2\}, \{Distancia\_estacaoAL2 \mapsto tremL2\}, \{Distancia\_estacaoBL2 \mapsto tremL2\})$

axm_tgtL2_type : $targetL2 \in edgeL2 \rightarrow vertL2$

axm_tgtL2_def : $partition(targetL2, \{trilhoABL2 \mapsto estacaoBL2\}, \{Distancia\_estacaoAL2 \mapsto estacaoAL2\}, \{Distancia\_estacaoBL2 \mapsto estacaoBL2\})$

axm_tVL2_type : $tL2\_V \in vertL2 \rightarrow vertT$

axm_tVL2_def : $partition(tL2\_V, \{tremL2 \mapsto trem\}, \{estacaoAL2 \mapsto estacao\}, \{estacaoBL2 \mapsto estacao\})$

axm_tEL2_type : $tL2\_E \in edgeL2 \rightarrow edgeT$

axm_tEL2_def : $partition(tL2\_E, \{Distancia\_estacaoAL2 \mapsto Distancia\_estacao\}, \{Distancia\_estacaoBL2 \mapsto Distancia\_estacao\}, \{trilhoABL2 \mapsto trilho\})$

axm_pertinenciavertL2_type: $pert\_vertL2 \in vertL2 \rightarrow REAL$

axm_pertinenciavertL2_def: $partition(pert\_vertL2, \{tremL2 \mapsto one\}, \{estacaoAL2 \mapsto one\}, \{estacaoBL2 \mapsto one\})$

axm_pertinenciaedgeL2_type: $pert\_edgeL2 \in edgeL2 \rightarrow REAL$

axm_pertinenciaedgeL2_def: $partition(pert\_edgeL2, \{Distancia\_estacaoAL2 \mapsto zeroeight\}, \{Distancia\_estacaoBL2 \mapsto zerotwo\}, \{trilhoABL2 \mapsto zerotwo\})$

axm_vertL3 : $partition(vertL3, \{tremL3\}, \{estacaoAL3\}, \{estacaoBL3\})$

axm_edgeL3 : $partition(edgeL3, \{Distancia\_estacaoAL3\}, \{Distancia\_estacaoBL3\}, \{trilhoABL3\})$

axm_srcL3_type : $sourceL3 \in edgeL3 \rightarrow vertL3$

axm_srcL3_def : $partition(sourceL3, \{trilhoABL3 \mapsto estacaoAL3\}, \{Distancia\_estacaoAL3 \mapsto tremL3\}, \{Distancia\_estacaoBL3 \mapsto tremL3\})$

axm_tgtL3_type : $targetL3 \in edgeL3 \rightarrow vertL3$

axm_tgtL3_def : $partition(targetL3, \{trilhoABL3 \mapsto estacaoBL3\}, \{Distancia\_estacaoAL3 \mapsto estacaoAL3\}, \{Distancia\_estacaoBL3 \mapsto estacaoBL3\})$

axm_tVL3_type : $tL3\_V \in vertL3 \rightarrow vertT$

axm_tVL3_def : $partition(tL3\_V, \{tremL3 \mapsto trem\}, \{estacaoAL3 \mapsto estacao\}, \{estacaoBL3 \mapsto estacao\})$

axm_tEL3_type : $tL3\_E \in edgeL3 \rightarrow edgeT$

axm_tEL3_def : $partition(tL3\_E, \{Distancia\_estacaoAL3 \mapsto Distancia\_estacao\}, \{Distancia\_estacaoBL3 \mapsto Distancia\_estacao\}, \{trilhoABL3 \mapsto trilho\})$

axm_pertinenciavertL3_type: $pert\_vertL3 \in vertL3 \rightarrow REAL$

axm_pertinenciavertL3_def: $partition(pert\_vertL3, \{tremL3 \mapsto one\}, \{estacaoAL3 \mapsto one\}, \{estacaoBL3 \mapsto one\})$

axm_pertinenciaedgeL3_type: $pert\_edgeL3 \in edgeL3 \rightarrow REAL$

axm_pertinenciaedgeL3_def: $partition(pert\_edgeL3, \{Distancia\_estacaoAL3 \mapsto zerotwo\}, \{Distancia\_estacaoBL3 \mapsto zeroeight\}, \{trilhoABL3 \mapsto zerotwo\})$

axm_vertL4 : $partition(vertL4, \{tremL4\}, \{estacaoAL4\}, \{estacaoBL4\}, \{estacaoDL4\})$

axm_edgeL4 : $partition(edgeL4, \{Distancia\_estacaoAL4\}, \{Distancia\_estacaoBL4\}, \{trilhoABL4\}, \{trilhoBDL4\})$

axm_srcL4_type : $sourceL4 \in edgeL4 \rightarrow vertL4$

axm_srcL4_def : $partition(sourceL4, \{trilhoABL4 \mapsto estacaoAL4\}, \{trilhoBDL4 \mapsto estacaoBL4\}, \{Distancia\_estacaoAL4 \mapsto tremL4\}, \{Distancia\_estacaoBL4 \mapsto tremL4\})$

axm_tgtL4_type : $targetL4 \in edgeL4 \rightarrow vertL4$

axm_tgtL4_def : $partition(targetL4, \{trilhoABL4 \mapsto estacaoBL4\}, \{trilhoBDL4 \mapsto estacaoDL4\}, \{Distancia\_estacaoAL4 \mapsto estacaoAL4\}, \{Distancia\_estacaoBL4 \mapsto estacaoBL4\})$

axm_tVL4_type : $tL4\_V \in vertL4 \rightarrow vertT$

axm_tVL4_def : $partition(tL4\_V, \{tremL4 \mapsto trem\}, \{estacaoAL4 \mapsto estacao\}, \{estacaoBL4 \mapsto estacao\}, \{estacaoDL4 \mapsto estacao\})$

axm_tEL4_type : $tL4\_E \in edgeL4 \rightarrow edgeT$

axm_tEL4_def : $partition(tL4\_E, \{Distancia\_estacaoAL4 \mapsto Distancia\_estacao\}, \{Distancia\_estacaoBL4 \mapsto Distancia\_estacao\}, \{trilhoABL4 \mapsto trilho\}, \{trilhoBDL4 \mapsto trilho\})$

axm_pertinenciavertL4_type: $pert\_vertL4 \in vertL4 \rightarrow REAL$

axm_pertinenciavertL4_def: $partition(pert\_vertL4, \{tremL4 \mapsto one\}, \{estacaoAL4 \mapsto one\}, \{estacaoBL4 \mapsto one\}, \{estacaoDL4 \mapsto one\})$

axm_pertinenciaedgeL4_type: $pert\_edgeL4 \in edgeL4 \rightarrow REAL$

axm_pertinenciaedgeL4_def: $partition(pert\_edgeL4, \{Distancia\_estacaoAL4 \mapsto zero\}, \{Distancia\_estacaoBL4 \mapsto one\}, \{trilhoABL4 \mapsto zerotwo\}, \{trilhoBDL4 \mapsto zeroseven\})$

axm_vertL5 : $partition(vertL5, \{tremL5\}, \{estacaoCL5\}, \{estacaoEL5\})$

axm_edgeL5 : $partition(edgeL5, \{Distancia\_estacaoCL5\}, \{Distancia\_estacaoEL5\}, \{trilhoCEL5\})$

axm_srcL5_type: $sourceL5 \in edgeL5 \rightarrow vertL5$

axm_srcL5_def: $partition(sourceL5, \{trilhoCEL5 \mapsto estacaoCL5\}, \{Distancia\_estacaoCL5 \mapsto tremL5\}, \{Distancia\_estacaoEL5 \mapsto tremL5\})$

axm_tgtL5_type : $targetL5 \in edgeL5 \rightarrow vertL5$

axm_tgtL5_def : $partition(targetL5, \{trilhoCEL5 \mapsto estacaoEL5\}, \{Distancia\_estacaoCL5 \mapsto estacaoCL5\}, \{Distancia\_estacaoEL5 \mapsto estacaoEL5\})$

axm_tVL5_type : $tL5\_V \in vertL5 \rightarrow vertT$

axm_tVL5_def : $partition(tL5\_V, \{tremL5 \mapsto trem\}, \{estacaoCL5 \mapsto estacao\}, \{estacaoEL5 \mapsto estacao\})$

axm_tEL5_type : $tL5\_E \in edgeL5 \rightarrow edgeT$

axm_tEL5_def : $partition(tL5\_E, \{Distancia\_estacaoCL5 \mapsto Distancia\_estacao\}, \{Distancia\_estacaoEL5 \mapsto Distancia\_estacao\}, \{trilhoCEL5 \mapsto trilho\})$

axm_pertinenciavertL5_type: $pert\_vertL5 \in vertL5 \rightarrow REAL$

axm_pertinenciavertL5_def: $partition(pert\_vertL5, \{tremL5 \mapsto one\}, \{estacaoCL5 \mapsto one\}, \{estacaoEL5 \mapsto one\})$

axm_pertinenciaedgeL5_type: $pert\_edgeL5 \in edgeL5 \rightarrow REAL$

axm_pertinenciaedgeL5_def: $partition(pert\_edgeL5, \{Distancia\_estacaoEL5 \mapsto zero\}, \{Distancia\_estacaoCL5 \mapsto one\}, \{trilhoCEL5 \mapsto zeronine\})$

axm_vertL6 : $partition(vertL6, \{tremL6\}, \{estacaoCL6\}, \{estacaoEL6\})$

axm_edgeL6 : $partition(edgeL6, \{Distancia\_estacaoCL6\}, \{Distancia\_estacaoEL6\}, \{trilhoCEL6\})$

axm_srcL6_type : $sourceL6 \in edgeL6 \rightarrow vertL6$

axm_srcL6_def : $partition(sourceL6, \{trilhoCEL6 \mapsto estacaoCL6\}, \{Distancia\_estacaoCL6 \mapsto tremL6\}, \{Distancia\_estacaoEL6 \mapsto tremL6\})$

axm_tgtL6_type : $targetL6 \in edgeL6 \rightarrow vertL6$

axm_tgtL6_def : $partition(targetL6, \{trilhoCEL6 \mapsto estacaoEL6\}, \{Distancia\_estacaoCL6 \mapsto estacaoCL6\}, \{Distancia\_estacaoEL6 \mapsto estacaoEL6\})$

axm_tVL6_type : $tL6\_V \in vertL6 \rightarrow vertT$

axm_tVL6_def : $partition(tL6\_V, \{tremL6 \mapsto trem\}, \{estacaoCL6 \mapsto estacao\}, \{estacaoEL6 \mapsto estacao\})$

axm_tEL6_type : $tL6\_E \in edgeL6 \rightarrow edgeT$

axm_tEL6_def : $partition(tL6\_E, \{Distancia\_estacaoCL6 \mapsto Distancia\_estacao\}, \{Distancia\_estacaoEL6 \mapsto Distancia\_estacao\}, \{trilhoCEL6 \mapsto trilho\})$

axm_pertinenciavertL6_type: $pert\_vertL6 \in vertL6 \rightarrow REAL$

axm_pertinenciavertL6_def: $partition(pert\_vertL6, \{tremL6 \mapsto one\}, \{estacaoCL6 \mapsto one\}, \{estacaoEL6 \mapsto one\})$

axm_pertinenciaedgeL6_type: $pert\_edgeL6 \in edgeL6 \rightarrow REAL$

axm_pertinenciaedgeL6_def: $partition(pert\_edgeL6, \{Distancia\_estacaoCL6 \mapsto zeroone\}, \{Distancia\_estacaoEL6 \mapsto zeronine\}, \{trilhoCEL6 \mapsto zeronine\})$

axm_vertL7 : $partition(vertL7, \{tremL7\}, \{estacaoCL7\}, \{estacaoEL7\}, \{estacaoJL7\})$

axm_edgeL7 : $partition(edgeL7, \{Distancia\_estacaoCL7\}, \{Distancia\_estacaoEL7\}, \{trilhoCEL7\}, \{trilhoEJL7\})$

axm_srcL7_type : $sourceL7 \in edgeL7 \rightarrow vertL7$

axm_srcL7_def : $partition(sourceL7, \{trilhoCEL7 \mapsto estacaoCL7\}, \{trilhoEJL7 \mapsto estacaoEL7\}, \{Distancia\_estacaoCL7 \mapsto tremL7\}, \{Distancia\_estacaoEL7 \mapsto tremL7\})$

axm_tgtL7_type : $targetL7 \in edgeL7 \rightarrow vertL7$

axm_tgtL7_def : $partition(targetL7, \{trilhoCEL7 \mapsto estacaoEL7\}, \{trilhoEJL7 \mapsto estacaoJL7\}, \{Distancia\_estacaoCL7 \mapsto estacaoCL7\}, \{Distancia\_estacaoEL7 \mapsto estacaoEL7\})$

axm_tVL7_type : $tL7\_V \in vertL7 \rightarrow vertT$

axm_tVL7_def : $partition(tL7\_V, \{tremL7 \mapsto trem\}, \{estacaoCL7 \mapsto estacao\}, \{estacaoEL7 \mapsto estacao\}, \{estacaoJL7 \mapsto estacao\})$

axm_tEL7_type : $tL7\_E \in edgeL7 \rightarrow edgeT$

axm_tEL7_def : $partition(tL7\_E, \{Distancia\_estacaoCL7 \mapsto Distancia\_estacao\}, \{Distancia\_estacaoEL7 \mapsto Distancia\_estacao\}, \{trilhoCEL7 \mapsto trilho\}, \{trilhoEJL7 \mapsto trilho\})$

axm_pertinenciavertL7_type: $pert\_vertL7 \in vertL7 \rightarrow REAL$

axm_pertinenciavertL7_def: $partition(pert\_vertL7, \{tremL7 \mapsto one\}, \{estacaoCL7 \mapsto one\}, \{estacaoEL7 \mapsto one\}, \{estacaoJL7 \mapsto one\})$

axm_pertinenciaedgeL7_type: $pert\_edgeL7 \in edgeL7 \rightarrow REAL$

axm_pertinenciaedgeL7_def: $partition(pert\_edgeL7, \{Distancia\_estacaoCL7 \mapsto zero\}, \{Distancia\_estacaoEL7 \mapsto one\}, \{trilhoCEL7 \mapsto zeronine\}, \{trilhoEJL7 \mapsto zeroeight\})$

axm_vertL8 : $partition(vertL8, \{tremL8\}, \{estacaoEL8\}, \{estacaoJL8\})$

axm_edgeL8 : $partition(edgeL8, \{Distancia\_estacaoEL8\}, \{Distancia\_estacaoJL8\}, \{trilhoEJL8\})$

axm_srcL8_type : $sourceL8 \in edgeL8 \rightarrow vertL8$

axm_srcL8_def : $partition(sourceL8, \{trilhoEJL8 \mapsto estacaoEL8\}, \{Distancia\_estacaoEL8 \mapsto tremL8\}, \{Distancia\_estacaoJL8 \mapsto tremL8\})$

axm_tgtL8_type : $targetL8 \in edgeL8 \rightarrow vertL8$

axm_tgtL8_def : $partition(targetL8, \{trilhoEJL8 \mapsto estacaoJL8\}, \{Distancia\_estacaoEL8 \mapsto estacaoEL8\}, \{Distancia\_estacaoJL8 \mapsto estacaoJL8\})$

axm_tVL8_type : $tL8\_V \in vertL8 \rightarrow vertT$

axm_tVL8_def : $partition(tL8\_V, \{tremL8 \mapsto trem\}, \{estacaoEL8 \mapsto estacao\}, \{estacaoJL8 \mapsto estacao\})$

axm_tEL8_type : $tL8\_E \in edgeL8 \rightarrow edgeT$

axm_tEL8_def : $partition(tL8\_E, \{Distancia\_estacaoEL8 \mapsto Distancia\_estacao\}, \{Distancia\_estacaoJL8 \mapsto Distancia\_estacao\}, \{trilhoEJL8 \mapsto trilho\})$

axm_pertinenciavertL8_type: $pert\_vertL8 \in vertL8 \rightarrow REAL$

axm_pertinenciavertL8_def: $partition(pert\_vertL8, \{tremL8 \mapsto one\}, \{estacaoEL8 \mapsto one\}, \{estacaoJL8 \mapsto one\})$

axm_pertinenciaedgeL8_type: $pert\_edgeL8 \in edgeL8 \rightarrow REAL$

axm_pertinenciaedgeL8_def: $partition(pert\_edgeL8, \{Distancia\_estacaoEL8 \mapsto one\}, \{Distancia\_estacaoJL8 \mapsto zero\}, \{trilhoEJL8 \mapsto zeroeight\})$

axm_vertL9 : $partition(vertL9, \{tremL9\}, \{estacaoEL9\}, \{estacaoJL9\})$

axm_edgeL9 : $partition(edgeL9, \{Distancia\_estacaoEL9\}, \{Distancia\_estacaoJL9\}, \{trilhoEJL9\})$

axm_srcL9_type : $sourceL9 \in edgeL9 \rightarrow vertL9$

axm_srcL9_def : $partition(sourceL9, \{trilhoEJL9 \mapsto estacaoEL9\}, \{Distancia\_estacaoEL9 \mapsto tremL9\}, \{Distancia\_estacaoJL9 \mapsto tremL9\})$

axm_tgtL9_type : $targetL9 \in edgeL9 \rightarrow vertL9$

axm_tgtL9_def : $partition(targetL9, \{trilhoEJL9 \mapsto estacaoJL9\}, \{Distancia\_estacaoEL9 \mapsto estacaoEL9\}, \{Distancia\_estacaoJL9 \mapsto estacaoJL9\})$

axm_tVL9_type : $tL9\_V \in vertL9 \rightarrow vertT$

axm_tVL9_def : $partition(tL9\_V, \{tremL9 \mapsto trem\}, \{estacaoEL9 \mapsto estacao\}, \{estacaoJL9 \mapsto estacao\})$

axm_tEL9_type : $tL9\_E \in edgeL9 \rightarrow edgeT$

axm_tEL9_def : $partition(tL9\_E, \{Distancia\_estacaoEL9 \mapsto Distancia\_estacao\}, \{Distancia\_estacaoJL9 \mapsto Distancia\_estacao\}, \{trilhoEJL9 \mapsto trilho\})$

axm_pertinenciavertL9_type: $pert\_vertL9 \in vertL9 \rightarrow REAL$

axm_pertinenciavertL9_def: $partition(pert\_vertL9, \{tremL9 \mapsto one\}, \{estacaoEL9 \mapsto one\}, \{estacaoJL9 \mapsto one\})$

axm_pertinenciaedgeL9_type: $pert\_edgeL9 \in edgeL9 \rightarrow REAL$

axm_pertinenciaedgeL9_def: $partition(pert\_edgeL9, \{Distancia\_estacaoEL9 \mapsto zerotwo\}, \{Distancia\_estacaoJL9 \mapsto zeroeight\}, \{trilhoEJL9 \mapsto zeroeight\})$

END

B.2 Máquina da Gramática de Grafos fuzzy ferrovia no modelo Event-B

An Event-B Specification of mchGGF
Creation Date: 12Jan2017 @ 09:03:51 AM

MACHINE mchGGF

SEES ctxGGF

VARIABLES

$vertG \;\; edgeG \;\; sourceG \;\; targetG \;\; tG\_V \;\; tG\_E \;\; pert\_vertG \;\; pert\_edgeG$

INVARIANTS

inv_VertG_type : $vertG \in \mathbb{P}(\mathbb{N})$

inv_EdgeG_type : $edgeG \in \mathbb{P}(\mathbb{N})$

inv_sourceG_type : $sourceG \in edgeG \rightarrow vertG$

inv_targetG_type : $targetG \in edgeG \rightarrow vertG$

inv_tVG_type : $tG\_V \in vertG \rightarrow vertT$

inv_tEG_type : $tG\_E \in edgeG \rightarrow edgeT$

inv_pert_vertG_type : $pert\_vertG \in vertG \rightarrow REAL$

inv_pert_edgeG_type : $pert\_edgeG \in edgeG \rightarrow REAL$

EVENTS

Initialisation

begin

act_vertG : $vertG := \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$

act_edgeG : $edgeG := \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$

act_srcG : $sourceG := \{1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 1, 4 \mapsto 3, 5 \mapsto 4, 6 \mapsto 6, 7 \mapsto 6,$
$8 \mapsto 5, 9 \mapsto 7, 10 \mapsto 2, 11 \mapsto 8, 12 \mapsto 9, 13 \mapsto 10, 14 \mapsto 9, 15 \mapsto 12\}$

act_tgtG : $targetG := \{1 \mapsto 3, 2 \mapsto 2, 3 \mapsto 3, 4 \mapsto 4, 5 \mapsto 5, 6 \mapsto 4, 7 \mapsto 5,$
$8 \mapsto 7, 9 \mapsto 11, 10 \mapsto 8, 11 \mapsto 9, 12 \mapsto 10, 13 \mapsto 11, 14 \mapsto 12, 15 \mapsto 15\}$

act_tVG_type : $tG\_V := \{1 \mapsto trem, 2 \mapsto estacao, 3 \mapsto estacao, 4 \mapsto estacao,$
$5 \mapsto estacao, 6 \mapsto trem, 7 \mapsto estacao, 8 \mapsto estacao, 9 \mapsto estacao, 10 \mapsto estacao,$
$11 \mapsto estacao, 12 \mapsto estacao\}$

act_tEG_type: $tG\_E := \{1 \mapsto trilho, 2 \mapsto Distancia\_estacao,$
$3 \mapsto Distancia\_estacao, 4 \mapsto trilho, 5 \mapsto trilho, 6 \mapsto Distancia\_estacao,$
$7 \mapsto Distancia\_estacao, 8 \mapsto trilho, 9 \mapsto trilho, 10 \mapsto trilho, 11 \mapsto trilho,$
$12 \mapsto trilho, 13 \mapsto trilho, 14 \mapsto trilho, 15 \mapsto trilho\}$

act_pert_vertG_type: $pert\_vertG := \{1 \mapsto one, 2 \mapsto one, 3 \mapsto one, 4 \mapsto one,$
$5 \mapsto one, 6 \mapsto one, 7 \mapsto one, 8 \mapsto one, 9 \mapsto one, 10 \mapsto one, 11 \mapsto one, 12 \mapsto one\}$

act_pert_edgeG_type: $pert\_edgeG := \{1 \mapsto zerotwo, 2 \mapsto one, 3 \mapsto zero,$
$4 \mapsto zeroseven, 5 \mapsto zerotwo, 6 \mapsto zeroeight, 7 \mapsto zerotwo, 8 \mapsto zeroeight,$
$9 \mapsto zeroseven, 10 \mapsto zerofive, 11 \mapsto zeronine, 12 \mapsto zeronine, 13 \mapsto zerotwo,$
$14 \mapsto zeroeight, 15 \mapsto zerofive\}$

end

Event *r1* $\widehat{=}$

any

m_V m_E $DelV$ $PreservE$ $DelE$ $Dangling$ $new\_Distancia\_estacaoAL1$
$PreservV$ $Pertpreserv\_edge\_Distancia\_estacaoB$ $Pertpreserv\_edge\_trilhoAB$

where

grd_mV: $m\_V \in vertL1 \rightarrow vertG$

grd_mE: $m\_E \in edgeL1 \rightarrow edgeG$

grd_PreservV: $PreservV = vertG \setminus DelV$

grd_DelV: $DelV = m\_V[vertL1 \setminus \varnothing]$

grd_PreservE: $PreservE = edgeG \setminus DelE$

grd_DelE: $DelE = m\_E[edgeL1 \setminus \{Distancia\_estacaoAL1\}]$

grd_Dangling: $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Distancia_estacaoA: $new\_Distancia\_estacaoAL1 \in \mathbb{N} \setminus edgeG$

grd_vertices: $\forall v \cdot v \in vertL1 \Rightarrow tL1\_V(v) = tG\_V(m\_V(v))$

grd_edges: $\forall e \cdot e \in edgeL1 \Rightarrow tL1\_E(e) = tG\_E(m\_E(e))$

grd_srctgt: $\forall e \cdot e \in edgeL1 \Rightarrow m\_V(sourceL1(e)) = sourceG(m\_E(e))$
$\wedge\ m\_V(targetL1(e)) = targetG(m\_E(e))$

grd_IdentV: $DelV \cap m\_V[\{tremL1, estacaoAL1, estacaoBL1\}] = \varnothing$

grd_IdentE: $DelE \cap m\_E[\{trilhoABL1, Distancia\_estacaoBL1\}] = \varnothing$

grd_Pertpreserv_edge_Distancia_estacaoB:
$Pertpreserv\_edge\_Distancia\_estacaoB =$
$sup(\{pert\_edgeG(m\_E(Distancia\_estacaoBL1)), zerotwo\})$

grd_Pertpreserv_edge_trilhoAB: $Pertpreserv\_edge\_trilhoAB =$
$sup(\{pert\_edgeG(m\_E(trilhoABL1)), zerotwo\})$

grd_DangCondition: $Dangling = \varnothing$

then

act_V: $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E: $edgeG := (edgeG \setminus DelE) \cup \{new\_Distancia\_estacaoAL1\}$

act_src: $sourceG := (DelE \lhd sourceG) \cup$
$\{new\_Distancia\_estacaoAL1 \mapsto m\_V(tremL1)\}$

act_tgt: $targetG := (DelE \lhd targetG) \cup$
$\{new\_Distancia\_estacaoAL1 \mapsto m\_V(estacaoAL1)\}$

act_tV: $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE: $tG\_E := (DelE \lhd tG\_E) \cup$
$\{new\_Distancia\_estacaoAL1 \mapsto Distancia\_estacao\}$

act_pert_vertG: $pert\_vertG := (DelV \lhd pert\_vertG) \cup \varnothing$

act_pert_edgeG: $pert\_edgeG := ((DelE \cup \{m\_E(Distancia\_estacaoBL1),$
$m\_E(trilhoABL1)\}) \lhd pert\_edgeG) \cup \{new\_Distancia\_estacaoAL1 \mapsto zeroeight,$
$m\_E(Distancia\_estacaoBL1) \mapsto Pertpreserv\_edge\_Distancia\_estacaoB,$
$m\_E(trilhoABL1) \mapsto Pertpreserv\_edge\_trilhoAB\}$

end

Event *r2* $\widehat{=}$

any

$m\_V\ \ m\_E\ \ DelV\ \ PreservV\ \ PreservE\ \ DelE\ \ Dangling$
$Pertpreserv\_edge\_trilhoAB\ \ Pertpreserv\_edge\_Distancia\_estacaoB$

$PreservV$ $new\_Distancia\_estacaoAL2$

where

grd_mV: $m\_V \in vertL2 \rightarrow vertG$

grd_mE: $m\_E \in edgeL2 \rightarrow edgeG$

grd_PreservV: $PreservV = vertG \setminus DelV$

grd_DelV: $DelV = m\_V[vertL2 \setminus \varnothing]$

grd_PreservE: $PreservE = edgeG \setminus DelE$

grd_DelE: $DelE = m\_E[edgeL2 \setminus \{Distancia\_estacaoAL2\}]$

grd_Dangling: $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Distancia_estacaoA: $new\_Distancia\_estacaoAL2 \in \mathbb{N} \setminus edgeG$

grd_vertices: $\forall v \cdot v \in vertL2 \Rightarrow tL2\_V(v) = tG\_V(m\_V(v))$

grd_edges: $\forall e \cdot e \in edgeL2 \Rightarrow tL2\_E(e) = tG\_E(m\_E(e))$

grd_srctgt: $\forall e \cdot e \in edgeL2 \Rightarrow m\_V(sourceL2(e)) = sourceG(m\_E(e))$
$\wedge\ m\_V(targetL2(e)) = targetG(m\_E(e))$

grd_IdentV: $DelV \cap m\_V[\{tremL2, estacaoAL2, estacaoBL2\}] = \varnothing$

grd_IdentE: $DelE \cap m\_E[\{trilhoABL2, Distancia\_estacaoBL2\}] = \varnothing$

grd_Pertpreserv_edge_Distancia_estacaoB:
$Pertpreserv\_edge\_Distancia\_estacaoB =$
$sup(\{pert\_edgeG(m\_E(Distancia\_estacaoBL2)), zerofour\})$

grd_Pertpreserv_edge_trilhoAB: $Pertpreserv\_edge\_trilhoAB =$
$sup(\{pert\_edgeG(m\_E(trilhoABL2)), zerotwo\})$

grd_DangCondition: $Dangling = \varnothing$

then

act_V: $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E: $edgeG := (edgeG \setminus DelE) \cup \{new\_Distancia\_estacaoAL2\}$

act_src: $sourceG := (DelE \lhd sourceG) \cup$
$\{new\_Distancia\_estacaoAL2 \mapsto m\_V(tremL2)\}$

act_tgt: $targetG := (DelE \lhd targetG) \cup$
$\{new\_Distancia\_estacaoAL2 \mapsto m\_V(estacaoAL2)\}$

act_tV: $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE: $tG\_E := (DelE \lhd tG\_E) \cup$
$\{new\_Distancia\_estacaoAL2 \mapsto Distancia\_estacao\}$

act_pert_vertG: $pert\_vertG := (DelV \lhd pert\_vertG) \cup \varnothing$

act_pert_edgeG: $pert\_edgeG := ((DelE \cup \{m\_E(Distancia\_estacaoBL2),$
$m\_E(trilhoABL2)\}) \lhd pert\_edgeG) \cup$
$\{new\_Distancia\_estacaoAL2 \mapsto zerosix, m\_E(Distancia\_estacaoBL2) \mapsto$
$Pertpreserv\_edge\_Distancia\_estacaoB, m\_E(trilhoABL2) \mapsto$
$Pertpreserv\_edge\_trilhoAB\}$

end

Event *r3* $\widehat{=}$

any

m_V m_E $DelV$ $PreservV$ $PreservE$ $DelE$ $Dangling$
$new\_Distancia\_estacaoAL3$ $Pertpreserv\_edge\_Distancia\_estacaoB$
$Pertpreserv\_edge\_trilhoAB$

where

grd_mV: $m\_V \in vertL3 \rightarrow vertG$

grd_mE: $m\_E \in edgeL3 \rightarrow edgeG$

grd_PreservV: $PreservV = vertG \setminus DelV$

grd_DelV: $DelV = m\_V[vertL3 \setminus \varnothing]$

grd_PreservE: $PreservE = edgeG \setminus DelE$

grd_DelE: $DelE = m\_E[edgeL3 \setminus \{Distancia\_estacaoAL3\}]$

grd_Dangling: $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Distancia_estacaoA: $new\_Distancia\_estacaoAL3 \in \mathbb{N} \setminus edgeG$

grd_vertices: $\forall v \cdot v \in vertL3 \Rightarrow tL3\_V(v) = tG\_V(m\_V(v))$

grd_edges: $\forall e \cdot e \in edgeL3 \Rightarrow tL3\_E(e) = tG\_E(m\_E(e))$

grd_srctgt: $\forall e \cdot e \in edgeL3 \Rightarrow m\_V(sourceL3(e)) = sourceG(m\_E(e))$
$\land\ m\_V(targetL3(e)) = targetG(m\_E(e))$

grd_IdentV: $DelV \cap m\_V[\{tremL3, estacaoAL3, estacaoBL3\}] = \varnothing$

grd_IdentE: $DelE \cap m\_E[\{trilhoABL3, Distancia\_estacaoBL3\}] = \varnothing$

grd_Pertpreserv_edge_Distancia_estacaoB:
$Pertpreserv\_edge\_Distancia\_estacaoB =$
$sup(\{pert\_edgeG(m\_E(Distancia\_estacaoBL3)), one\})$

grd_Pertpreserv_edge_trilhoAB: $Pertpreserv\_edge\_trilhoAB =$
$sup(\{pert\_edgeG(m\_E(trilhoABL3)), zerotwo\})$

grd_DangCondition: $Dangling = \varnothing$

then

act_V: $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E: $edgeG := (edgeG \setminus DelE) \cup \{new\_Distancia\_estacaoAL3\}$

act_src: $sourceG := (DelE \lhd sourceG) \cup$
$\{new\_Distancia\_estacaoAL3 \mapsto m\_V(tremL3)\}$

act_tgt: $targetG := (DelE \lhd targetG) \cup$
$\{new\_Distancia\_estacaoAL3 \mapsto m\_V(estacaoAL3)\}$

act_tV: $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE: $tG\_E := (DelE \lhd tG\_E) \cup$
$\{new\_Distancia\_estacaoAL3 \mapsto Distancia\_estacao\}$

act_pert_vertG: $pert\_vertG := (DelV \lhd pert\_vertG) \cup \varnothing$

act_pert_edgeG: $pert\_edgeG := ((DelE \cup \{m\_E(Distancia\_estacaoBL3), m\_E(trilhoABL3)\}) \lhd pert\_edgeG) \cup \{new\_Distancia\_estacaoAL3 \mapsto zero, m\_E(Distancia\_estacaoBL3) \mapsto Pertpreserv\_edge\_Distancia\_estacaoB, m\_E(trilhoABL3) \mapsto Pertpreserv\_edge\_trilhoAB\}$

end

Event *r4* $\widehat{=}$

any

$m\_V \;\; m\_E \;\; DelV \;\; PreservV \;\; PreservE \;\; DelE \;\; Dangling$
$new\_Distancia\_estacaoDL4 \;\; Pertpreserv\_edge\_Distancia\_estacaoB$
$Pertpreserv\_edge\_trilhoBD \;\; Pertpreserv\_edge\_trilhoAB$

where

grd_mV: $m\_V \in vertL4 \rightarrow vertG$
grd_mE: $m\_E \in edgeL4 \rightarrow edgeG$
grd_PreservV: $PreservV = vertG \setminus DelV$
grd_DelV: $DelV = m\_V[vertL4 \setminus \varnothing]$
grd_PreservE: $PreservE = edgeG \setminus DelE$
grd_DelE: $DelE = m\_E[edgeL4 \setminus \{Distancia\_estacaoAL4\}]$
grd_Dangling: $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$
grd_new_Distancia_estacaoD: $new\_Distancia\_estacaoDL4 \in \mathbb{N} \setminus edgeG$
grd_vertices: $\forall v \cdot v \in vertL4 \Rightarrow tL4\_V(v) = tG\_V(m\_V(v))$
grd_edges: $\forall e \cdot e \in edgeL4 \Rightarrow tL4\_E(e) = tG\_E(m\_E(e))$
grd_srctgt: $\forall e \cdot e \in edgeL4 \Rightarrow m\_V(sourceL4(e)) = sourceG(m\_E(e))$
$\land\, m\_V(targetL4(e)) = targetG(m\_E(e))$
grd_IdentV: $DelV \cap m\_V[\{tremL4, estacaoAL4, estacaoBL4, estacaoDL4\}] = \varnothing$
grd_IdentE: $DelE \cap m\_E[\{trilhoABL4, trilhoBDL4, Distancia\_estacaoBL4\}] = \varnothing$
grd_Pertpreserv_edge_Distancia_estacaoB: $Pertpreserv\_edge\_Distancia\_estacaoB = sup(\{pert\_edgeG(m\_E(Distancia\_estacaoBL4)), one\})$
grd_Pertpreserv_edge_trilhoAB: $Pertpreserv\_edge\_trilhoAB = sup(\{pert\_edgeG(m\_E(trilhoABL4)), zerotwo\})$
grd_Pertpreserv_edge_trilhoBD: $Pertpreserv\_edge\_trilhoBD = sup(\{pert\_edgeG(m\_E(trilhoBDL4)), zeroseven\})$
grd_DangCondition: $Dangling = \varnothing$

then

act_V: $vertG := (vertG \setminus DelV) \cup \varnothing$
act_E: $edgeG := (edgeG \setminus DelE) \cup \{new\_Distancia\_estacaoDL4\}$

act_src: $sourceG := (DelE \lhd sourceG) \cup \{new\_Distancia\_estacaoDL4 \mapsto m\_V(tremL4)\}$
act_tgt: $targetG := (DelE \lhd targetG) \cup \{new\_Distancia\_estacaoDL4 \mapsto m\_V(estacaoDL4)\}$
act_tV: $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$
act_tE: $tG\_E := (DelE \lhd tG\_E) \cup \{new\_Distancia\_estacaoDL4 \mapsto Distancia\_estacao\}$
act_pert_vertG: $pert\_vertG := (DelV \lhd pert\_vertG) \cup \varnothing$
act_pert_edgeG: $pert\_edgeG := ((DelE \cup \{m\_E(Distancia\_estacaoBL4), m\_E(trilhoABL4), m\_E(trilhoBDL4)\}) \lhd pert\_edgeG) \cup \{new\_Distancia\_estacaoDL4 \mapsto zero, m\_E(Distancia\_estacaoBL4) \mapsto Pertpreserv\_edge\_Distancia\_estacaoB, m\_E(trilhoABL4) \mapsto Pertpreserv\_edge\_trilhoAB, m\_E(trilhoBDL4) \mapsto Pertpreserv\_edge\_trilhoBD\}$

end

Event *r5* $\widehat{=}$

any

$m\_V \;\; m\_E \;\; DelV \;\; PreservV \;\; PreservE \;\; DelE \;\; Dangling$
$new\_Distancia\_estacaoCL5 \;\; Pertpreserv\_edge\_Distancia\_estacaoE$
$Pertpreserv\_edge\_trilhoCE$

where

grd_mV: $m\_V \in vertL5 \to vertG$
grd_mE: $m\_E \in edgeL5 \to edgeG$
grd_PreservV: $PreservV = vertG \setminus DelV$
grd_DelV: $DelV = m\_V[vertL5 \setminus \varnothing]$
grd_PreservE: $PreservE = edgeG \setminus DelE$
grd_DelE: $DelE = m\_E[edgeL5 \setminus \{Distancia\_estacaoCL5\}]$
grd_Dangling: $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$
grd_new_Distancia_estacaoC: $new\_Distancia\_estacaoCL5 \in \mathbb{N} \setminus edgeG$
grd_vertices: $\forall v \cdot v \in vertL5 \Rightarrow tL5\_V(v) = tG\_V(m\_V(v))$
grd_edges: $\forall e \cdot e \in edgeL5 \Rightarrow tL5\_E(e) = tG\_E(m\_E(e))$
grd_srctgt: $\forall e \cdot e \in edgeL5 \Rightarrow m\_V(sourceL5(e)) = sourceG(m\_E(e)) \land m\_V(targetL5(e)) = targetG(m\_E(e))$
grd_IdentV: $DelV \cap m\_V[\{tremL5, estacaoCL5, estacaoEL5\}] = \varnothing$
grd_IdentE: $DelE \cap m\_E[\{trilhoCEL5, Distancia\_estacaoEL5\}] = \varnothing$
grd_Pertpreserv_edge_Distancia_estacaoE: $Pertpreserv\_edge\_Distancia\_estacaoE = sup(\{pert\_edgeG(m\_E(Distancia\_estacaoEL5)), zeronine\})$
grd_Pertpreserv_edge_trilhoCE: $Pertpreserv\_edge\_trilhoCE = sup(\{pert\_edgeG(m\_E(trilhoCEL5)), zeronine\})$

grd_DangCondition: $Dangling = \varnothing$

then

act_V: $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E: $edgeG := (edgeG \setminus DelE) \cup \{new\_Distancia\_estacaoCL5\}$

act_src: $sourceG := (DelE \lhd sourceG) \cup$

$\{new\_Distancia\_estacaoCL5 \mapsto m\_V(tremL5)\}$

act_tgt: $targetG := (DelE \lhd targetG) \cup$

$\{new\_Distancia\_estacaoCL5 \mapsto m\_V(estacaoCL5)\}$

act_tV: $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE: $tG\_E := (DelE \lhd tG\_E) \cup$

$\{new\_Distancia\_estacaoCL5 \mapsto Distancia\_estacao\}$

act_pert_vertG: $pert\_vertG := (DelV \lhd pert\_vertG) \cup \varnothing$

act_pert_edgeG: $pert\_edgeG := ((DelE \cup \{m\_E(Distancia\_estacaoEL5),$

$m\_E(trilhoCEL5)\}) \lhd pert\_edgeG) \cup \{new\_Distancia\_estacaoCL5 \mapsto zeroone,$

$m\_E(Distancia\_estacaoEL5) \mapsto Pertpreserv\_edge\_Distancia\_estacaoE,$

$m\_E(trilhoCEL5) \mapsto Pertpreserv\_edge\_trilhoCE\}$

end

Event *r6* $\widehat{=}$

any

$m\_V \;\; m\_E \;\; DelV \;\; PreservV \;\; PreservE \;\; DelE \;\; Dangling$

$new\_Distancia\_estacaoCL6 \;\; Pertpreserv\_edge\_Distancia\_estacaoE$

$Pertpreserv\_edge\_trilhoCE$

where

grd_mV: $m\_V \in vertL6 \rightarrow vertG$

grd_mE: $m\_E \in edgeL6 \rightarrow edgeG$

grd_PreservV: $PreservV = vertG \setminus DelV$

grd_DelV: $DelV = m\_V[vertL6 \setminus \varnothing]$

grd_PreservE: $PreservE = edgeG \setminus DelE$

grd_DelE: $DelE = m\_E[edgeL6 \setminus \{Distancia\_estacaoCL6\}]$

grd_Dangling: $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Distancia_estacaoC: $new\_Distancia\_estacaoCL6 \in \mathbb{N} \setminus edgeG$

grd_vertices: $\forall v \cdot v \in vertL6 \Rightarrow tL6\_V(v) = tG\_V(m\_V(v))$

grd_edges: $\forall e \cdot e \in edgeL6 \Rightarrow tL6\_E(e) = tG\_E(m\_E(e))$

grd_srctgt: $\forall e \cdot e \in edgeL6 \Rightarrow m\_V(sourceL6(e)) = sourceG(m\_E(e))$

$\land\, m\_V(targetL6(e)) = targetG(m\_E(e))$

grd_IdentV: $DelV \cap m\_V[\{tremL6, estacaoCL6, estacaoEL6\}] = \varnothing$

grd_IdentE: $DelE \cap m\_E[\{trilhoCEL6, Distancia\_estacaoEL6\}] = \varnothing$

grd_Pertpreserv_edge_Distancia_estacaoE:
$Pertpreserv\_edge\_Distancia\_estacaoE =$
$sup(\{pert\_edgeG(m\_E(Distancia\_estacaoEL6)), one\})$
grd_Pertpreserv_edge_trilhoCE: $Pertpreserv\_edge\_trilhoCE =$
$sup(\{pert\_edgeG(m\_E(trilhoCEL6)), zeronine\})$
grd_DangCondition: $Dangling = \varnothing$

then

act_V: $vertG := (vertG \setminus DelV) \cup \varnothing$
act_E: $edgeG := (edgeG \setminus DelE) \cup \{new\_Distancia\_estacaoCL6\}$
act_src: $sourceG := (DelE \lhd sourceG) \cup$
$\{new\_Distancia\_estacaoCL6 \mapsto m\_V(tremL6)\}$
act_tgt: $targetG := (DelE \lhd targetG) \cup$
$\{new\_Distancia\_estacaoCL6 \mapsto m\_V(estacaoCL6)\}$
act_tV: $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$
act_tE: $tG\_E := (DelE \lhd tG\_E) \cup$
$\{new\_Distancia\_estacaoCL6 \mapsto Distancia\_estacao\}$
act_pert_vertG: $pert\_vertG := (DelV \lhd pert\_vertG) \cup \varnothing$
act_pert_edgeG: $pert\_edgeG := ((DelE \cup \{m\_E(Distancia\_estacaoEL6),$
$m\_E(trilhoCEL6)\}) \lhd pert\_edgeG) \cup \{new\_Distancia\_estacaoCL6 \mapsto zero,$
$m\_E(Distancia\_estacaoEL6) \mapsto Pertpreserv\_edge\_Distancia\_estacaoE,$
$m\_E(trilhoCEL6) \mapsto Pertpreserv\_edge\_trilhoCE\}$

end

Event *r7* $\widehat{=}$

any

$m\_V \;\; m\_E \;\; DelV \;\; PreservV \;\; PreservE \;\; DelE \;\; Dangling$
$new\_Distancia\_estacaoJL7 \;\; Pertpreserv\_edge\_Distancia\_estacaoE$
$Pertpreserv\_edge\_trilhoCE \;\; Pertpreserv\_edge\_trilhoEJ$

where

grd_mV: $m\_V \in vertL7 \rightarrow vertG$
grd_mE: $m\_E \in edgeL7 \rightarrow edgeG$
grd_PreservV: $PreservV = vertG \setminus DelV$
grd_DelV: $DelV = m\_V[vertL7 \setminus \varnothing]$
grd_PreservE: $PreservE = edgeG \setminus DelE$
grd_DelE: $DelE = m\_E[edgeL7 \setminus \{Distancia\_estacaoCL7\}]$
grd_Dangling: $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$
grd_new_Distancia_estacaoJ: $new\_Distancia\_estacaoJL7 \in \mathbb{N} \setminus edgeG$
grd_vertices: $\forall v \cdot v \in vertL7 \Rightarrow tL7\_V(v) = tG\_V(m\_V(v))$
grd_edges: $\forall e \cdot e \in edgeL7 \Rightarrow tL7\_E(e) = tG\_E(m\_E(e))$

grd_srctgt: $\forall e \cdot e \in edgeL7 \Rightarrow m\_V(sourceL7(e)) = sourceG(m\_E(e))$
$\wedge\ m\_V(targetL7(e)) = targetG(m\_E(e))$
grd_IdentV: $DelV \cap m\_V[\{tremL7, estacaoCL7, estacaoEL7,$
$estacaoJL7\}] = \varnothing$
grd_IdentE: $DelE \cap m\_E[\{trilhoCEL7, trilhoEJL7,$
$Distancia\_estacaoEL7\}] = \varnothing$
grd_Pertpreserv_edge_Distancia_estacaoE:
$Pertpreserv\_edge\_Distancia\_estacaoE =$
$sup(\{pert\_edgeG(m\_E(Distancia\_estacaoEL7)), one\})$
grd_Pertpreserv_edge_trilhoCE: $Pertpreserv\_edge\_trilhoCE =$
$sup(\{pert\_edgeG(m\_E(trilhoCEL7)), zeronine\})$
grd_Pertpreserv_edge_trilhoEJ: $Pertpreserv\_edge\_trilhoEJ =$
$sup(\{pert\_edgeG(m\_E(trilhoEJL7)), zeroeight\})$
grd_DangCondition: $Dangling = \varnothing$

then

act_V: $vertG := (vertG \setminus DelV) \cup \varnothing$
act_E: $edgeG := (edgeG \setminus DelE) \cup \{new\_Distancia\_estacaoJL7\}$
act_src: $sourceG := (DelE \lhd sourceG) \cup$
$\{new\_Distancia\_estacaoJL7 \mapsto m\_V(tremL7)\}$
act_tgt: $targetG := (DelE \lhd targetG) \cup$
$\{new\_Distancia\_estacaoJL7 \mapsto m\_V(estacaoJL7)\}$
act_tV: $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$
act_tE: $tG\_E := (DelE \lhd tG\_E) \cup$
$\{new\_Distancia\_estacaoJL7 \mapsto Distancia\_estacao\}$
act_pert_vertG: $pert\_vertG := (DelV \lhd pert\_vertG) \cup \varnothing$
act_pert_edgeG: $pert\_edgeG := ((DelE \cup \{m\_E(Distancia\_estacaoEL7),$
$m\_E(trilhoCEL7), m\_E(trilhoEJL7)\}) \lhd pert\_edgeG) \cup$
$\{new\_Distancia\_estacaoJL7 \mapsto zero, m\_E(Distancia\_estacaoEL7) \mapsto$
$Pertpreserv\_edge\_Distancia\_estacaoE, m\_E(trilhoCEL7) \mapsto$
$Pertpreserv\_edge\_trilhoCE, m\_E(trilhoEJL7) \mapsto$
$Pertpreserv\_edge\_trilhoEJ\}$

end

Event *r8* $\widehat{=}$

any

m_V m_E $DelV$ $PreservV$ $PreservE$ $DelE$ $Dangling$
$new\_Distancia\_estacaoEL8$ $Pertpreserv\_edge\_Distancia\_estacaoJ$
$Pertpreserv\_edge\_trilhoEJ$

where

grd_mV: $m\_V \in vertL8 \rightarrow vertG$
grd_mE: $m\_E \in edgeL8 \rightarrow edgeG$

grd_PreservV: $PreservV = vertG \setminus DelV$

grd_DelV: $DelV = m\_V[vertL8 \setminus \varnothing]$

grd_PreservE: $PreservE = edgeG \setminus DelE$

grd_DelE: $DelE = m\_E[edgeL8 \setminus \{Distancia\_estacaoEL8\}]$

grd_Dangling: $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Distancia_estacaoE: $new\_Distancia\_estacaoEL8 \in \mathbb{N} \setminus edgeG$

grd_vertices: $\forall v \cdot v \in vertL8 \Rightarrow tL8\_V(v) = tG\_V(m\_V(v))$

grd_edges: $\forall e \cdot e \in edgeL8 \Rightarrow tL8\_E(e) = tG\_E(m\_E(e))$

grd_srctgt: $\forall e \cdot e \in edgeL8 \Rightarrow m\_V(sourceL8(e)) = sourceG(m\_E(e))$
$\wedge\ m\_V(targetL8(e)) = targetG(m\_E(e))$

grd_IdentV: $DelV \cap m\_V[\{tremL8, estacaoEL8, estacaoJL8\}] = \varnothing$

grd_IdentE: $DelE \cap m\_E[\{trilhoEJL8, Distancia\_estacaoJL8\}] = \varnothing$

grd_Pertpreserv_edge_Distancia_estacaoJ:
$Pertpreserv\_edge\_Distancia\_estacaoJ =$
$sup(\{pert\_edgeG(m\_E(Distancia\_estacaoJL8)), zeroeight\})$

grd_Pertpreserv_edge_trilhoEJ: $Pertpreserv\_edge\_trilhoEJ =$
$sup(\{pert\_edgeG(m\_E(trilhoEJL8)), zeroeight\})$

grd_DangCondition: $Dangling = \varnothing$

then

act_V: $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E: $edgeG := (edgeG \setminus DelE) \cup \{new\_Distancia\_estacaoEL8\}$

act_src: $sourceG := (DelE \lhd sourceG) \cup$
$\{new\_Distancia\_estacaoEL8 \mapsto m\_V(tremL8)\}$

act_tgt: $targetG := (DelE \lhd targetG) \cup$
$\{new\_Distancia\_estacaoEL8 \mapsto m\_V(estacaoEL8)\}$

act_tV: $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE: $tG\_E := (DelE \lhd tG\_E) \cup$
$\{new\_Distancia\_estacaoEL8 \mapsto Distancia\_estacao\}$

act_pert_vertG: $pert\_vertG := (DelV \lhd pert\_vertG) \cup \varnothing$

act_pert_edgeG: $pert\_edgeG := ((DelE \cup \{m\_E(Distancia\_estacaoJL8)$
$, m\_E(trilhoEJL8)\}) \lhd pert\_edgeG) \cup \{new\_Distancia\_estacaoEL8 \mapsto$
$zerotwo, m\_E(Distancia\_estacaoJL8) \mapsto$
$Pertpreserv\_edge\_Distancia\_estacaoJ, m\_E(trilhoEJL8) \mapsto$
$Pertpreserv\_edge\_trilhoEJ\}$

end

Event *r9* $\widehat{=}$

any

$m\_V\ \ m\_E\ \ DelV\ \ PreservV\ \ PreservE\ \ DelE\ \ Dangling$

$new\_Distancia\_estacaoEL9$ $Pertpreserv\_edge\_Distancia\_estacaoJ$
$Pertpreserv\_edge\_trilhoEJ$

where

grd_mV: $m\_V \in vertL9 \rightarrow vertG$

grd_mE: $m\_E \in edgeL9 \rightarrow edgeG$

grd_PreservV: $PreservV = vertG \setminus DelV$

grd_DelV: $DelV = m\_V[vertL9 \setminus \varnothing]$

grd_PreservE: $PreservE = edgeG \setminus DelE$

grd_DelE: $DelE = m\_E[edgeL9 \setminus \{Distancia\_estacaoEL9\}]$

grd_Dangling: $Dangling = dom((sourceG \rhd DelV) \cup (targetG \rhd DelV)) \setminus DelE$

grd_new_Distancia_estacaoE: $new\_Distancia\_estacaoEL9 \in \mathbb{N} \setminus edgeG$

grd_vertices: $\forall v \cdot v \in vertL9 \Rightarrow tL9\_V(v) = tG\_V(m\_V(v))$

grd_edges: $\forall e \cdot e \in edgeL9 \Rightarrow tL9\_E(e) = tG\_E(m\_E(e))$

grd_srctgt: $\forall e \cdot e \in edgeL9 \Rightarrow m\_V(sourceL9(e)) = sourceG(m\_E(e))$
$\wedge\ m\_V(targetL9(e)) = targetG(m\_E(e))$

grd_IdentV: $DelV \cap m\_V[\{tremL9, estacaoEL9, estacaoJL9\}] = \varnothing$

grd_IdentE: $DelE \cap m\_E[\{trilhoEJL9, Distancia\_estacaoJL9\}] = \varnothing$

grd_Pertpreserv_edge_Distancia_estacaoJ:
$Pertpreserv\_edge\_Distancia\_estacaoJ =$
$sup(\{pert\_edgeG(m\_E(Distancia\_estacaoJL9)), one\})$

grd_Pertpreserv_edge_trilhoEJ: $Pertpreserv\_edge\_trilhoEJ =$
$sup(\{pert\_edgeG(m\_E(trilhoEJL9)), zeroeight\})$

grd_DangCondition: $Dangling = \varnothing$

then

act_V: $vertG := (vertG \setminus DelV) \cup \varnothing$

act_E: $edgeG := (edgeG \setminus DelE) \cup \{new\_Distancia\_estacaoEL9\}$

act_src: $sourceG := (DelE \lhd sourceG) \cup$
$\{new\_Distancia\_estacaoEL9 \mapsto m\_V(tremL9)\}$

act_tgt: $targetG := (DelE \lhd targetG) \cup$
$\{new\_Distancia\_estacaoEL9 \mapsto m\_V(estacaoEL9)\}$

act_tV: $tG\_V := (DelV \lhd tG\_V) \cup \varnothing$

act_tE: $tG\_E := (DelE \lhd tG\_E) \cup$
$\{new\_Distancia\_estacaoEL9 \mapsto Distancia\_estacao\}$

act_pert_vertG: $pert\_vertG := (DelV \lhd pert\_vertG) \cup \varnothing$

act_pert_edgeG: $pert\_edgeG := ((DelE \cup \{m\_E(Distancia\_estacaoJL9),$
$m\_E(trilhoEJL9)\}) \lhd pert\_edgeG) \cup \{new\_Distancia\_estacaoEL9 \mapsto$
$zero, m\_E(Distancia\_estacaoJL9) \mapsto Pertpreserv\_edge\_Distancia\_estacaoJ,$
$m\_E(trilhoEJL9) \mapsto Pertpreserv\_edge\_trilhoEJ\}$

end

END