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

Dissertação

Uma Tradução de Redes de Petri Fuzzy Generalizadas para Gramáticas de Grafos com Atributos

Júlia Krüger Vieira

Pelotas, 2021

Júlia Krüger Vieira

Uma Tradução de Redes de Petri Fuzzy Generalizadas para Gramáticas de Grafos com Atributos

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

Orientador: Prof. Dr. Luciana Foss
Coorientador: Prof. Dr. Simone A. C. Cavalheiro

Pelotas, 2021

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

V657t Vieira, Júlia Krüger

Uma tradução de redes de Petri Fuzzy generalizadas para gramáticas de grafos com atributos / Júlia Krüger Vieira ; Luciana Foss, orientadora ; Simone A. C. Cavalheiro, coorientadora. — Pelotas, 2021.

78 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2021.

1. Redes de Petri Fuzzy. 2. Gramática de grafos com atributos. 3. Transformação entre modelos. I. Foss, Luciana, orient. II. Cavalheiro, Simone A. C., coorient. III. Título.

CDD : 005

Elaborada por Aline Herbstrith Batista CRB: 10/1737

Júlia Krüger Vieira

Uma Tradução de Redes de Petri Fuzzy Generalizadas para Gramáticas de Grafos com Atributos

Dissertação aprovada, como requisito parcial, para obtenção do grau de Mestre em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 19 de fevereiro, 2021.

Banca examinadora:

Prof. Dr. Luciana Foss (Orientador)
Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Simone André da Costa Cavalheiro (Coorientador)
Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Renata Hax Sander Reiser
Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Hélida Salles Santos
Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Norte.

Prof. Dr. André Du Bois
Doutor em Ciência da Computação pela Universidade Heriot-Watt University.

RESUMO

VIEIRA, Júlia Krüger. **Uma Tradução de Redes de Petri Fuzzy Generalizadas para Gramáticas de Grafos com Atributos**. 2021. 78 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2021.

Redes de Petri (RP) e Gramáticas de Grafos (GG) são formalismos adequados para especificar sistemas concorrentes e distribuídos. Existem diversos trabalhos que relacionam RP e GG, onde GG é vista como uma generalização de RP por permitir expressar mais informações.

Existem diferentes tipos de RP, porém, neste trabalho, o foco é dado às Redes de Petri Fuzzy (RPF), mais especificamente às Redes de Petri Fuzzy Generalizadas (RPFG).

O objetivo deste trabalho é propor uma tradução de RPFG para Gramática de Grafos com Atributos (GGA) que permita descrever o comportamento de um sistema de inferência fuzzy através de gramática de grafos. Uma GGA é uma extensão da GG convencional que incorpora atributos com tipos de dados abstratos nas definições dos grafos.

A tradução, consiste em transformar os componentes da RPFG em componentes de uma GGA. As operações fuzzy da rede são definidas por meio de uma especificação algébrica associada à GGA e os componentes da gramática, isto é, o grafo tipo, o grafo inicial e o conjunto de regras, são construídos a partir da estrutura da rede. De acordo com a semântica da rede, dois modos de disparo das transições são considerados. No primeiro modo, os tokens são consumidos das pré-condições da transição disparada. Já no segundo, os tokens são preservados. Esses dois comportamentos, geram dois tipos de regras para a gramática obtida da tradução, as quais simulam os modos de disparo da rede.

Para garantir que a GGA obtida da tradução mantém o mesmo comportamento da RPFG traduzida, prova-se que a tradução preserva a semântica das RPFG, isto é, todo comportamento observado nas redes deve ser observado na gramática e vice-versa. Mais especificamente, se ocorre uma mudança de estado na RPFG por meio do disparo de uma transição e um novo estado é alcançado, então esse novo estado, deve ser também alcançado na GGA se a regra, obtida na tradução da transição disparada na rede, for aplicada e vice-versa.

Com a tradução espera-se prover um novo modelo para a abordagem fuzzy derivado da tradução aqui proposta. De forma direta a GGA obtida pode ser usada na verificação de propriedades de sistemas fuzzy através de ferramentas como o provador de teoremas do Rodin. Além disso, espera-se que o modelo obtido pela tradução

sirva como inspiração para a definição de Gramáticas de Grafos Fuzzy mais flexíveis, nas quais pode-se escolher o tipo de conjuntos e operações fuzzy desejados.

Palavras-Chave: redes de petri fuzzy; gramática de grafos com atributos; transformação entre modelos

ABSTRACT

VIEIRA, Júlia Krüger. **A Translation from Generalized Fuzzy Petri Nets into Attributed Graph Grammars**. 2021. 78 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2021.

Petri Nets (PN) and Graph Grammars (GG) are suitable formalism's to specify concurrent and distributed systems. There are several studies that relate PN and GG, where GG is seen as a generalization of PN since it allows expressing more information.

There are different types of PN, however, in this work, the focus is given to Fuzzy Petri Nets (FPN), more specifically to Generalized Fuzzy Petri Nets (GFPN).

The objective of this work is to propose a translation from GFPN to Attributed Graph Grammars, that allows to describe the behavior of a fuzzy inference system using graph grammars. An Attributed Graph Grammar (AGG) is an extension of conventional GG that incorporates attributes with abstract data types in graph definitions.

The translation consists of transforming the components of the GFPN into components of an AGG. The fuzzy operations of the net are defined by means of an algebraic specification associated with the AGG and the components of the grammar, that is, the type graph, the initial graph and the set of rules, are built from the structure of the net. According to the semantics of GFPN, two modes of transition triggering are considered. In the first mode, tokens are consumed from the preconditions of the triggered transition. In the second, tokens are preserved. These two behaviors generate two types of rules for the grammar obtained from the translation, which simulate the net's trigger modes.

In order to ensure that the AGG obtained from the translation maintains the same behavior as the translated GFPN, it is proven that the translation preserves the semantics of the GFPN, that is, all behavior observed in the nets must be observed in the grammars and vice versa. More specifically, if a state change occurs in the GFPN by triggering a transition and a new state is achieved, then this new state must also be achieved in the AGG if the rule, obtained in the translation from the transition triggered in the net, is applied and vice versa.

The translation is expected to provide a new model for the fuzzy approach derived from the translation proposed here. The obtained AGG can be directly used to verify properties of fuzzy systems using tools such as Rodin's theorem prover. In addition, it is expected that the model obtained by the translation will serve as an inspiration for the definition of a more flexible Fuzzy Graph Grammars, in which one can choose the desired type of fuzzy sets and operations.

Keywords: fuzzy petri nets; attributed graphs grammars; transformation between models

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

RP	Rede de Petri
RPF	Rede de Petri Fuzzy
RPFG	Rede de Petri Fuzzy Generalizada
LF	Lógica Fuzzy
CF	Conjunto Fuzzy
SBRF	Sistemas Baseados em Regras Fuzzy
GG	Gramática de Grafos
GGA	Gramática de Grafos com Atributos
SPO	Single Pushout
DPO	Double Pushout
LHS	Lado Esquerdo da Regra
RHS	Lado Direito da Regra

LISTA DE SÍMBOLOS

$f \circ g$	Composição de funções e morfismos f e g
$\rightarrow$	Funções totais ou morfismos
$\rightarrowtail$	Funções totais e injetivas ou morfismos
$\nrightarrow$	Funções parciais e injetivas ou morfismos
$rng(r)$	Imagem de uma relação binária r
$\setminus$ ou $-$	Diferença de conjuntos
$\|A\|$	Cardinalidade do conjunto A
$\mathbb{R}$	Conjunto dos números reais
$\mathbb{N}$	Conjunto dos números naturais
$\int$	Operação de integração algébrica
$\bar{A}$	Complemento do conjunto A

SUMÁRIO

1 INTRODUÇÃO

A especificação e verificação de sistemas concorrentes, distribuídos e/ou reativos pode ser realizada através de diferentes métodos formais os quais utilizam linguagens matemáticas como ferramenta. O uso de métodos formais na especificação dos sistemas computacionais viabiliza a descrição e a otimização durante todo o desenvolvimento dispondo de soluções simplificadas com maior grau de eficiência (RIBEIRO, 2006). Dentre as linguagens formais adequadas para descrever sistemas concorrentes e complexos, tem-se as Redes de Petri (RP) e a Gramática de Grafos (GG) para as quais existem diversas técnicas e ferramentas para a especificação e verificação formal de sistemas (ROZENBERG, 1997; DESEL; REISIG, 1996).

Em uma RP um modelo é descrito por lugares, transições e recursos disponíveis na rede (tokens), os quais representam as condições mínimas necessárias para ocorrerem mudanças de um estado para outro. Um estado é definido pela marcação corrente da rede, isto é, todos os recursos disponíveis em um determinado momento. O comportamento do sistema é baseado na criação e consumo de recursos durante os disparos das transições (DESEL; REISIG, 1996; SILVA, 1993). As Redes de Petri Fuzzy (RPF) são generalizações das Redes de Petri que incorporam elementos fuzzy permitindo modelar bases de regras de acordo com os sistemas fuzzy. Assim, pode-se representar formalmente a etapa de inferência de um sistema fuzzy através de uma RPF e verificar se o comportamento desejado será satisfeito (CHEN; KE; CHANG, 1990; VIRTANEN, 1995). Diferentes tipos de RPF foram criadas agregando fatores como pesos, temporização, números fuzzy considerando diferentes algoritmos para dar o sentido semântico de cada rede (ZHOU; ZAIN, 2016; YEUNG; YSANG, 1998; PEDRYCZ; CAMARGO, 2003; AZIZ et al., 2010; CHEN, 2002).

GG constitui uma forma de modelagem baseada em grafos. Para especificar um sistema utilizando GG, é necessário descrever o estado inicial do sistema e fornecer um conjunto de regras para expressar as ações que o sistema poderá realizar. Os elementos sintáticos das GGs são descritos por grafos e a semântica é dada pela aplicação das regras em grafos estados que representam estágios do sistema. GGs podem incluir especificações algébricas para agregar o tratamento de atributos com

tipos de dados abstratos. Essas gramáticas são denominadas Gramáticas de Grafos com Atributos (GGA) (ROZENBERG, 1997; CAVALHEIRO; FOSS; RIBEIRO, 2017).

Existem diversos trabalhos na área de métodos formais que tratam de GGs e RPs. Alguns deles apresentam traduções de um modelo para o outro. A principal motivação destas traduções encontra-se no fato de uma GG possuir um maior grau de expressividade que uma RP. Como as RPs são mais abstratas não se consegue detalhar diversas relações entre os componentes que compõem os estados de um sistema. Dessa maneira, caso exista essa necessidade ou se a sequência das computações não podem ser reduzidas a causa e conflito, então, utilizar uma gramática torna-se mais eficiente (BALDAN; CORRADINI; KÖNIG, 2001; CORRADINI, 1999; BALDAN et al., 2010; KORFF; RIBEIRO, 1994).

Neste contexto, será desenvolvida uma tradução entre modelos partindo de um tipo de RPF chamado de Generalizada (RPFG) para se obter uma GGA. Nesta tradução, visa-se uma GGA que descreva o mesmo sistema de produções fuzzy que a RPFG modela. As operações fuzzy da rede são especificadas por meio de uma especificação algébrica que define todas as operações utilizadas na rede. Os componentes da gramática, isto é, o grafo tipo, o grafo inicial e o conjunto de regras são construídos a partir da estrutura da rede. De acordo com a semântica da rede, dois modos de disparo das transições são considerados. No primeiro modo, os tokens serão consumidos nas pré-condições de cada transição, já no segundo, os tokens serão preservados. Esses dois comportamentos, geram dois tipos de regras para a gramática obtida os quais simulam os modos de disparo da rede.

A GGA derivada da tradução representa genericamente um tipo de sistema de inferência fuzzy e pode ser utilizada como base para formalizar outras extensões, onde, pode-se utilizar diferentes instâncias de lógica fuzzy, como por exemplo, a valorada intervalarmente (BUSTINCE SOLA et al., 2016).

Como motivação, exploram-se as transformações entre os modelos para alcançar uma Gramática de Grafos com Atributos Fuzzy com o fim de permitir a modelagem e a verificação formal em fase de especificação de tais sistemas. Visto que existem diferentes técnicas e ferramentas que permitem a verificação de propriedades e a prova de teoremas para GGs

A PACE (GMBH, 2008) e a PNeS (SURAJ; GROCHOWALSKI, 2017; SURAJ, 2018) são exemplos de ferramentas disponíveis para RPF. A ferramenta PACE permite a simulação de sistemas fuzzy modelados por RPF, porém não permite a verificação de propriedades. Já a ferramenta PNeS é um ambiente integrado utilizado na simulação e análise de diversos tipos de RPs além de permitir a simulação de diversos tipos de RPFGs. Com essa ferramenta, é possível detectar deadlocks, verificar as propriedades estruturais das redes, calcular invariantes, realizar reduções na rede e construir grafos de análise de alcançabilidade.

A vantagem em utilizar as ferramentas para GG na abordagem fuzzy reside na possibilidade de provar propriedades que dizem respeito a computação e não apenas aos estados alcançáveis dos sistemas fuzzy. Além disso, pode-se lidar com sistemas que possuem espaço de estados infinito por meio do provador de teoremas do Rodin (CAVALHEIRO; FOSS; RIBEIRO, 2017; CAVALHEIRO, 2010). Com relação ao modelo em si, essa tradução pode servir como base para uma Gramática de Grafos fuzzy mais genérica, onde se consiga definir sistemas fuzzy mais complexos (onde os tokens possam se relacionar entre si) com grafos não discretos.

O ferramental utilizado como base para realizar a transformação tem como aparato diversos trabalhos relacionados, o primeiro trabalho foi realizado por (KREOWSKI, 1981) que compara ambos os formalismos e relaciona suas similaridades expondo uma simulação de uma RP através de uma GG. Outros trabalhos, como (CORRADINI, 1999; EHRIG; PADBERG, 2003; SANTOS, 1999; BALDAN et al., 2010) ampliam a estratégia do mapeamento que consiste em transformar cada transição da RP em uma regra da gramática.

Para mostrar que a tradução está correta, deve-se garantir que a semântica foi preservada pela codificação, isto é, que a RPFG e a GGA possuem o mesmo comportamento. Para tanto, apresenta-se a prova de que o comportamento de ambos os formalismos possuem execuções equivalentes (CORRADINI, 1999; EHRIG; PADBERG, 2003; SANTOS, 1999).

1.1 Objetivos e Resultados esperados

Este trabalho, tem como principal objetivo realizar a tradução de Rede de Petri Fuzzy Generalizada para Gramática de Grafos com Atributos, considerando nesta tradução a preservação semântica das RPF.

Como objetivos específicos destacam-se:

1. realizar um levantamento técnico sobre os tipos de RPFs, GGAs e transformações entre RPs e GGs;
2. estudar a sintaxe e a semântica dos modelos a fim de relacionar a forma de especificação dos elementos sintáticos e semânticos das RPF e das GGAs;
3. estudar os métodos de transformação entre RPs e GGs;
4. aplicar uma estratégia com base nos métodos de transformação entre RPs e GGs para traduzir uma RPF para GGA;
5. demonstrar que a tradução proposta preserva a semântica das RPF.

Como resultados espera-se prover um novo método de especificação e análise de sistemas fuzzy, para o qual se possa fazer uso das ferramentas disponíveis para GG, principalmente o provador de teoremas, e também introduzir uma base para a definição de gramática de grafos fuzzy generalizada.

1.2 Organização do Trabalho

No Capítulo 2 serão enumerados os trabalhos relacionados de maior relevância de acordo com o tema aqui abordado. No Capítulo 3 serão apresentados os conceitos e definições relacionados com a Lógica Fuzzy. No Capítulo 4 enumeram-se as definições de Gramática de Grafos com Atributos, assim como, diversos exemplos. No Capítulo 5 os principais conceitos sobre Redes de Petri Fuzzy e Rede de Petri Fuzzy Generalizada serão apresentados. No Capítulo 6 apresenta-se a contribuição científica do projeto, mostrando, a tradução de RPFG para GGA e a prova da preservação da semântica. No Capítulo 7 encontram-se as conclusões alcançadas a partir da realização deste projeto.

2 TRABALHOS RELACIONADOS

Neste capítulo, serão apresentados os trabalhos de maior relevância que se relacionam com os tópicos abordados nesta dissertação.

Estrutura do Capítulo:

Na Seção 2.1 estão enumerados os principais trabalhos que se referem à tradução de Redes de Petri para Gramática de Grafos. Na Seção 2.2 estão enumerados os principais trabalhos que introduzem o conceito de Gramáticas de Grafos Fuzzy.

2.1 Traduções de Redes de Petri para Gramática de Grafos

Nas traduções aqui apresentadas a estratégia básica adotada para realizar o mapeamento entre ambos os modelos é obter o grafo inicial da gramática a partir da marcação inicial da rede. Cada transição gera uma regra da gramática cujo grafo do lado esquerdo da regra (LHS) é formado pelas pré-condições da rede. O grafo de interface geralmente é vazio visto que as regras não conservam itens. O grafo do lado direito da regra (RHS) é formado pelas pós-condições da rede. No caso de gramáticas tipadas, o grafo tipo é composto pela representação de todos os lugares existentes na rede, pois, os lugares descrevem quais tipos de recursos estão disponíveis. Essas codificações comumente geram gramáticas discretas, isto é, o conjunto de arestas dos grafos é vazio (grafos discretos).

Nas traduções, O disparo das transições são executados através da aplicação das regras da gramática podendo-se escolher qualquer tipo de abordagem algébrica para efetuar uma codificação. A abordagem algébrica para GG define como se dá a aplicação das regras nos grafos estados. Existem diferentes tipos de abordagens algébricas cujas mais usadas são a double pushout (DPO) e a single pushout (SPO) as quais são consideradas como clássicas. A abordagem DPO utiliza dois pushouts e um grafo de interface (que define os elementos a serem preservados) para aplicar uma regra. Já a abordagem SPO utiliza um único pushout para aplicar uma regra, onde, a abordagem SPO é definida como uma generalização da DPO (ROZENBERG, 1997).

O estudo das relações existentes entre RPs e GGs começou por (KREOWSKI,

1981) que compara ambas abordagens e propõem uma simulação de uma RP por meio de uma GG. Esta proposta, segue a abordagem DPO e durante a simulação a estrutura da rede é totalmente preservada, pois, o grafo do LHS é formado pela transição (pré-condições e pós-condições) antes do disparo e o grafo do RHS é formado pela transição (pré-condições e pós-condições) após o efeito do disparo. Nos grafos da gramática os tokens são modelados como vértices adicionais ligados aos seus respectivos lugares através de arestas.

Uma tradução da Rede de Petri de Alto Nível para uma Gramática de Grafos Rotulada com Atributos foi proposta por (KORFF; RIBEIRO, 1994). Uma Rede de Petri de Alto Nível adiciona às RPs convencionais estruturas internas de tipos de dados abstratos, isto é, uma especificação algébrica. A relação entre Rede de Petri de Alto Nível e Gramática de Grafos com Atributos é equivalente a relação existente entre RP e GG. Nesta tradução, o conjunto de vértices dos grafos da gramática é obtido a partir dos tokens da rede. Os lugares aos quais os tokens pertencem rotulam tais vértices. O conjunto de atributos dos grafos da gramática é obtido a partir dos valores vinculados aos tokens. A álgebra de termos das regras é construída a partir dos elementos algébricos presentes nas transições da rede tais como variáveis e equações. A abordagem algébrica utilizada é a SPO. Neste trabalho, a prova da conservação da semântica é apresentada demonstrando a compatibilidade do comportamento entre ambos modelos.

Em (CORRADINI; MONTANARI, 1995) uma RP Lugar/Transição é traduzida para uma GG a qual utiliza grafos coloridos, onde, as cores dos grafos são definidas como rótulos pertencentes a alfabetos. Nesta tradução, as marcações da rede são consideradas como grafos discretos. No mapeamento entre os modelos o conjunto de vértices dos grafos é constituído pelos tokens da rede e os rótulos que representam as cores são obtidos a partir dos lugares. A abordagem algébrica utilizada é a DPO.

Um estudo do mapeamento entre GG Tipada e RP Lugar/Transição é apresentado por (SANTOS, 1999). Neste trabalho, é demonstrado que GGs podem ser codificadas em RPs levando em consideração a perda de informação e RPs podem ser codificadas em GGs. Somente GGs discretas podem ser traduzidas para RPs com preservação semântica. Os tipos de tradução apresentados são: de uma GG Discreta para uma RP Lugar/Transição e vice-versa e de uma GG de Ocorrência para uma RP de Ocorrência e vice-versa. Nas codificações apresentadas os vértices dos grafos das gramáticas são obtidos a partir dos tokens e os lugares são considerados como os rótulos de tais vértices. A abordagem algébrica adotada foi a SPO.

Em (BALDAN et al., 2010) é feita uma discussão sobre traduções de diferentes classes de RPs para GG. As classes de RPs consideradas são as Redes de Sistemas Elementares (RSE), as Redes de Consumo-Produção-Leitura (CPL) e as Redes Lugar/Transição. Todas as classes de redes podem ser equipadas com arcos especiais

de leitura, inibição ou reset. Em (BALDAN et al., 2010) diferentes abordagens algébricas são discutidas. As clássicas DPO e SPO podem ser empregadas para realizar a tradução de qualquer classe de RP para GG, mas existem classes menos conhecidas como a Sistema de Transformação de Sub-objetos (STS) e suas derivadas as quais são baseadas na reescrita de estruturas de sub-grafos. Neste artigo, também é exposta uma relação de qual abordagem algébrica é mais apropriada para realizar a codificação de cada classe de RP com a presença de arcos de leitura, inibição ou reset para GG. Para a codificação das redes Lugar/Transição com a presença dos arcos mencionados é recomendada a utilização das abordagens algébricas clássicas, pois, estas facilitam a preservação da semântica durante a tradução. Para codificar as classes RSE e CPL como gramáticas com a presença dos arcos especiais, é recomendado utilizar a abordagem STS e suas sub-classes por terem mais flexibilidade e expressarem melhor a semântica dessas redes. A tradução exemplificada neste trabalho, tem como base a representação exposta por Kreowski e codifica uma classe de RP chamada de Enriquecida em uma GG tipada. A característica da rede Enriquecida é ser uma extensão da rede Lugar/Transição a qual possui arcos de contexto e inibição. Nesta tradução, os lugares são modelados por vértices, tokens são representados como arestas unárias (arestas conectadas a um único vértice) conectadas aos seus respectivos vértices. No grafo tipo a multiplicidade dos tokens presentes nos lugares é modelada por uma aresta unária conectada a cada vértice. Neste trabalho, o grafo de interface não é vazio e contém os arcos de contexto vinculados a rede, pois, estes serão preservados. A abordagem algébrica utilizada é a DPO a qual simula o efeito de inibição.

Como exemplos de traduções com aplicação prática pode-se referenciar (BARDOHL; ERMEL; PADBERG, 2003) e (BIERMANN et al., 2008). Em (BARDOHL; ERMEL; PADBERG, 2003), a tradução de Rede de Petri de Alto Nível para Gramática de Grafos Rotulada com Atributos é utilizada para executar a rede através da gramática em uma ferramenta visual chamada GenGED. Já em (BARDOHL; ERMEL; PADBERG, 2003) é realizada uma simulação de RPs através de Gramática de Grafos com Atributos a qual permite a reconfiguração da rede. Esta simulação é utilizada para implementar um módulo de uma ferramenta chamada RON (Reconfigurable Object Nets) o qual converte partes da rede em grafos e as transições da rede em regras da gramática possibilitando a simulação da reconfiguração da rede.

2.2 Gramática de Grafos Fuzzy

Alguns trabalhos referentes à Gramática de Grafos Fuzzy foram revisados com o intuito de verificar os métodos empregados nas abordagens existentes.

Em (WATKINS, 1996) e em (WATKINS, 1997) é utilizada uma gramática conhe-

cida como fuzzy string para realizar o reconhecimento de ruídos em imagens bidimensionais de partituras musicais. Estes trabalhos, apresentam duas versões de gramáticas fuzzy strings para realizar a análise de imagens uma sem atributos e outra com atributos nos vértices e nas arestas. Tais gramáticas, modelam a aplicação de predicados e não utilizam uma abordagem algébrica para realizar a aplicação das regras nos grafos estados. Cada regra está associada a um fator de certeza que expressa a pertinência da ação. O efeito da aplicação da regra atua sobre a pertinência da ação e depende da operação fuzzy escolhida que pode ser o máximo (operador OU fuzzy) ou mínimo (operador E fuzzy). Nestes trabalhos, considera-se ainda um controlador sobre as gramáticas que realiza o controle da explosão do espaço de estados na árvore de derivação.

No trabalho de (PARASYUK; YERSHOV, 2006), foi realizada uma abordagem categórica para gramática de grafos fuzzy. Este conceito, é apoiado sobre a categoria de objetos fuzzy e descreve as transformações entre grafos fuzzy os quais são gerados a partir de conjuntos fuzzy. Nesta pesquisa, os vértices e as arestas que fazem parte dos grafos fuzzy são averiguados expondo as relações fuzzy existentes, além disso, dois tipos de morfismos são introduzidos na categoria de grafos fuzzy. A aplicação de uma regra representa a operação de mínimo utilizando a construção de um pushout o qual leva em consideração os elementos fuzzy presentes nos grafos. Este trabalho, expõem a relação direta entre gramática de Chomsky e gramática de grafos fuzzy e relata que os problemas de indecidibilidade relacionados com as gramáticas de chomsky são mantidos pelas gramáticas de grafos fuzzy.

Seguindo a mesma linha do trabalho anterior, em (PARASYUK; ERSHOV, 2007), uma abordagem categórica é utilizada para desenvolver o conceito de transformações entre FD-grafos. Neste trabalho, os FD-grafos (Grafos Fuzzy Distribuídos) são utilizados para descrever os estados de um sistema cujos componentes são distribuídos. Os vértices dos FD-grafos são formados por grafos fuzzy que representam os componentes do sistema. As transformações entre os FD-grafos (mudanças entre os estados) utiliza o conceito de gramática de grafos fuzzy porque considera as transformações entre os grafos fuzzy isolados (vértices). A estrutura distribuída do sistema tem que ser preservada durante as mudanças entre os estados, para isso, as operações de junção e disjunção (as quais são modeladas por morfismos especiais) são utilizadas para sincronizar os componentes considerando a sua distribuição na rede. Transformações paralelas sobre FD-grafos também foram analisadas e são realizadas sobre os componentes não distribuídos (grafos fuzzy). A abordagem algébrica usada neste artigo é a double pushout por não alterar a estrutura distribuída do sistema.

Para representar transformações entre modelos arquiteturais, (PARASYUK; YERSHOV, 2008) desenvolve uma representação para a arquitetura de software fuzzy baseada em regras as quais modelam a transformação do módulo independente de

plataforma (PIM) para o modelo plataforma específica (PSM). Grafos fuzzy são adaptados gerando os grafos (L, U)-fuzzy, onde, um conjunto de rótulos vinculados aos vértices expressam as pertinências fuzzy e as arestas representam estruturas arbitrárias de um reticulado não restringindo-se somente ao uso do intervalo $[0, 1]$. Os grafos (L, U)-fuzzy permitem inserir atributos fuzzy elementares os quais são pertinentes ao sistema que os módulos representam. Os grafos fuzzy são utilizados no sistema de transformação de grafos fuzzy (derivado das GGs convencionais) cujas regas equivalem a uma produção do tipo *Se grafo fuzzy do antecedente Então grafo fuzzy de resultado*. Os métodos de transformação de grafos fuzzy possuem uma biblioteca implementada na linguagem java. Esta biblioteca, foi usada para desenvolver o módulo que transforma o modelo PIM no modelo PSM. O módulo transforma o modelo PIM dado como entrada aplicando as transformações entre os grafos fuzzy através das regras gerando um tipo de modelo PSM o qual possui uma aproximação da implementação em termos de programação.

A GGA fuzzy derivada da tradução elaborada nesta dissertação difere-se das Gramáticas de Grafos Fuzzy revisadas por fornecer um modelo mais flexível que permiti dispor de diferentes tipos de operações fuzzy. Além disso a lógica fuzzy presente nas gramáticas revisadas é fixa dificultando a extensão da mesma. Outro ponto importante é que a GGA obtida da tradução permite a inclusão de situações não fuzzy na modelagem. Isto pode ser útil para descrever situações nas quais a tomada de decisão não depende exclusivamente da avaliação fuzzy, mas sim, de um conjunto de fatores fuzzy e não fuzzy que definem os estados do sistema a ser especificado.

3 LÓGICA FUZZY

A Lógica Fuzzy (LF) aumenta o teor da expressividade da informação nos sistemas computacionais através da extensão dos conjuntos e operadores clássicos associando um grau de representabilidade aos elementos. Os Conjuntos Fuzzy (CF) são conjuntos cujas fronteiras não são bem definidas (difusas). Em relação aos conjuntos clássicos os conjuntos fuzzy exprimem relações fracas através da sua função característica utilizando valores definidos no intervalo $[0, 1]$ (ZADEH, 1965).

Os Sistemas fuzzy inserem um grau de incerteza o qual é inerente ao mundo real por possuírem uma natureza maleável. Esse tipo de sistema modela problemas categóricos cujos limites não são bem definidos, um exemplo, é a modelagem da variação da temperatura. Dessa maneira, a função sinal pertencente à lógica clássica não é adequada já que não considera o fator de atenuação inerente a essas classes de problemas. Na modelagem da informação fuzzy são considerados termos linguísticos os quais são derivados da linguagem natural e utilizados para categorizar os conjuntos fuzzy, por exemplo, alguém é alto ou baixo (BENINI; JR MENEGUETTE, 2009).

Todos os conceitos e definições abordados neste capítulo, assim como, outros exemplos, podem ser encontrados em: (ROSS et al., 2004; DUBOIS; PRADE, 2000; NETO; HENRIQUE; COELHO, 2019; REDDY, 2017; GOMIDE; GUDWIN, 1994; SANCHEZ, 2009; BENINI; JR MENEGUETTE, 2009; MOTTA JAFELICE; BARROS; BASSANEZI, 2005; SILVA, 2011; LIU et al., 2017; CHEN; KE; CHANG, 1990; KLEMENT; MESIAR; PAP, 2004; ZADEH, 1965; BASSANEZI; BARROS, 2010).

Estrutura do Capítulo:

Na Seção 3.1 são enumerados os principais conceitos referentes aos conjuntos fuzzy. Na Seção 3.2 são apresentadas as definições que envolvem o conceito de relações fuzzy. Na Seção 3.3 uma breve introdução do método de operação de sistemas fuzzy é descrita.

3.1 Conjuntos Fuzzy

Na lógica clássica as relações de pertinência existentes são modeladas de forma inflexível, isto é, um elemento pertence ou não a um dado conjunto. Já na LF essas relações não são definidas de forma rígida, um grau de atenuação é considerado. Assim, um elemento pode pertencer a diversos conjuntos fuzzy com graus de pertinência variáveis descrevendo informações adicionais. Os limites entre os conjuntos fuzzy são expressados de forma tênue através da extensão da função de pertinência da lógica clássica inserindo um valor definido no intervalo $[0, 1]$ que diz o quanto um elemento pertence a um dado conjunto fuzzy. A definição formal da função de pertinência fuzzy e de conjuntos fuzzy é apresentada na Definição 1.

Definição 1 ***(Função de Pertinência Fuzzy e Conjunto Fuzzy):*** *Dado o conjunto $U \neq \varnothing$ e o subconjunto A, a função de pertinência $u_A : U \rightarrow [0, 1]$, relaciona o elemento $x \in U$, de forma a definir o quão ele pertence ao subconjunto A, através do grau de pertinência associado, representado por $u_A(x)$, tal que $0 \leq u_A(x) \leq 1$. Assim, um* ***conjunto fuzzy A*** *em relação a U é dado pela expressão:*

$$A = \{(x, u_A(x))|x \in U\}$$

Dessa forma, a LF constrói conectivos mais flexíveis que consideram o nível de atenuação determinado pela função de pertinência fuzzy. O universo de discurso para conjuntos fuzzy é o espaço fuzzy completo de variação de uma variável do modelo.

Na modelagem da informação utilizando-se CF, o conceito das **variáveis linguísticas** é de total relevância. As variáveis linguísticas tem seus valores dados de forma qualitativa e quantitativa. Um termo linguístico é derivado da linguagem natural e é vinculado a variável para definir o conceito que está sendo modelado de forma qualitativa. Os termos linguísticos representam um fator categórico e geralmente levam em consideração a imprecisão inerente à linguagem natural, por exemplo, o dia está quente ou frio. Já a faixa de valores vinculada a variável (pertinências) define quantitativamente a informação representando o grau de associabilidade da mesma ao conceito.

Uma variável linguística é caracterizada pela tupla $(\mathbf{Z}, \mathbf{T}(\mathbf{Z}), \mathbf{U}, \mathbf{G}, \mathbf{M})$, onde, Z é o nome da variável (por exemplo: altura, pressão, umidade, etc.), $T(Z)$ é o conjunto de termos linguísticos de Z (por exemplo: alto, médio, baixo, etc.), U é o universo de discurso considerado, G é a regra sintática que gera termos de $T(Z)$ e M é a regra semântica utilizada para associar cada termo gerado por G a um CF definido em U.

Exemplificação 1 ***(Conjuntos Fuzzy):*** *A Figura 1 ilustra a modelagem fuzzy da variável temperatura. Nela pode-se ver, quatro conjuntos fuzzy representados pelos termos linguísticos (BAIXA, NORMAL, ALTA e ALTÍSSIMA) que modelam a variabilidade dos graus da temperatura de acordo com o grau de pertinência que está entre 0% e*

100%. *Dessa forma, se tomarmos o elemento cuja a temperatura possui o valor de* $30°C$ *como exemplo, de acordo com a modelagem, ele pertence a três dos conjuntos mas com uma variação na pertinência que indica o quanto ele pertence a cada um deles, isto é,* $u_{BAIXA}(30) = 0.4$, $u_{NORMAL}(30) = 0.4$, $u_{ALTÍSSIMA}(30) = 0$. *Nesta modelagem, quando a temperatura atinge* $30°C$ *tem o mesmo valor de pertinência em ambos conjuntos fuzzy Baixa e Normal. Este fato, representa o ponto onde a temperatura começa a deixar de ser considerada baixa passando gradualmente a ser considerada normal.*

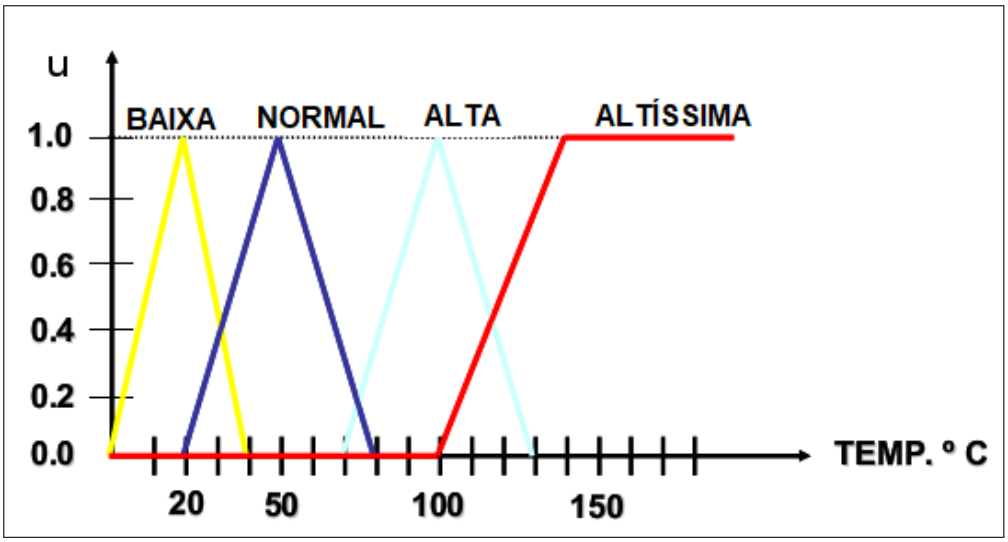

Figura 1 – Modelagem fuzzy da variável temperatura (NETO; HENRIQUE; COELHO, 2019)

A fim de realizar a avaliação lógica de expressões as quais incluem uma pertinência utiliza-se os conectivos fuzzy, estes, são funções construídas sobre um conjunto de propriedades que as caracterizam. Os principais conectivos fuzzy são a negação, as T-normas e as T-conormas.

Definição 2 ***(Negação Fuzzy):*** *Uma negação fuzzy é uma função definida em* $N : [0, 1] \to [0, 1]$ *que satisfaz as seguintes propriedades:*

1. *N1:* $N(0) = 1$ *e* $N(1) = 0$.
2. *N2: É monotônica decrescente. Se* $x \leqslant y$ *então* $N(x) \geqslant N(y)$, $\forall x, y \in [0, 1]$.

Se uma negação fuzzy satisfaz a propriedade de involutividade, N3: $N(N(x)) = x, \forall x \in [0, 1]$, então ela é dita negação forte. Um exemplo de negação forte é a negação fuzzy $N_S(x) = 1 - x$.

Definição 3 ***(T-norma):*** *Uma* ***T-norma*** *é uma operação definida em* $T : [0, 1]^2 \to [0, 1]$, *onde, para* $\forall x, y, z \in U$ *o seguinte conjunto de propriedades é satisfeito:*

1. *Comutatividade:* $T(x, y) = T(y, x)$.
2. *Associatividade:* $T(T(x, y), z) = T(x, T(y, z))$.
3. *Monotonicidade: Se* $x \leq z$ *então* $T(x, y) \leq T(y, z)$.
4. *Elemento absorvente:* $T(x, 0) = 0$.
5. *Elemento neutro:* $T(x, 1) = x$.

São exemplos de T-normas:

- Mínimo: $T_M(x, y) = min(x, y)$.
- Lukasiewicz: $T_{LK}(x, y) = max(x + y - 1, 0)$.
- Produto algébrico: $T_P(x, y) = x.y$.
- Fraca: $T_W(x, y) = 1$ se $max(x, y) = 1$ e $T_W(x, y) = 0$, caso contrário.
- Produto drástico: $T_D(x, y) = \begin{cases} 0 & se \quad x, y \in [0, 1) \\ min(x, y) & c.c. \end{cases}$

Definição 4 ***(T-conorma):*** *Uma* ***T-conorma*** *é uma operação definida em* $S : [0, 1]^2 \rightarrow [0, 1]$*, onde, para* $\forall x, y, z \in U$ *o seguinte conjunto de propriedades é satisfeito:*

1. *Comutatividade:* $S(x, y) = S(y, x)$.
2. *Associatividade:* $S(S(x, y), z) = S(x, S(y, z))$.
3. *Monotonicidade: Se* $x \leq z$ *então* $S(x, y) \leq S(y, z)$.
4. *Elemento absorvente:* $S(x, 1) = 1$.
5. *Elemento neutro:* $S(x, 0) = x$.

São exemplos de T-conormas:

- Máximo: $S_M(x, y) = max(x, y)$.
- Soma probabilística: $S_P(x, y) = x + y - x.y$.
- Lukasiewicz: $S_{LK}(x, y) = min(x + y, 1)$.
- Soma drástica: $S_D(x, y) = \begin{cases} 1 & se \quad x, y \in (0, 1] \\ max(x, y) & c.c. \end{cases}$

3.2 Relações Fuzzy

Relações Fuzzy também mapeiam elementos entre dois universos. Uma relação será fuzzy quando utiliza-se a teoria dos conjuntos fuzzy na modelagem da informação e será clássica quando utiliza-se a teoria dos conjuntos clássica.

Uma relação fuzzy R entre duas variáveis, $x \in X$ e $y \in Y$, é expressa através de uma função que mapeia o par ordenado (x, y) definido no espaço $X \times Y$ para um determinado grau o qual diz o teor desta relação, isto é, $R : X \times Y \rightarrow [0, 1]$.

Definição 5 ***(Relação Fuzzy):*** *Uma relação fuzzy R sobre $U_1 \times U_2 \times ... \times U_n$, é qualquer subconjunto fuzzy do produto cartesiano $U_1 \times U_2 \times ... \times U_n$. Se o produto cartesiano for formado por apenas dois conjuntos, $U_1 \times U_2$, a relação é chamada de fuzzy binária sobre $U_1 \times U_2$. Assim, uma relação fuzzy é definida por uma função de pertinência $u_R : U_1 \times U_2 \times ... \times U_n \rightarrow [0, 1]$.*

Uma relação fuzzy além de indicar se existe ou não uma relação entre dois elementos como ocorre nas relações clássicas fornece também o grau desta relação, isto é, indica o quanto dois elementos estão relacionados entre si. Dessa forma, uma relação fuzzy R é um mapeamento do produto cartesiano realizado entre conjuntos fuzzy definido no intervalo $[0, 1]$. Tal mapeamento, é expressado pela função característica u_R a qual é responsável por efetivar a relação dos pares ordenados cujos elementos pertencem a dois universos devolvendo a pertinência desta relação.

A Definição 6 enumera o conceito de produto cartesiano fuzzy, pois, uma relação fuzzy é formalizada a partir do produto cartesiano clássico estendendo a função característica da relação clássica para uma função de pertinência fuzzy.

Definição 6 ***(Produto Cartesiano Fuzzy):*** *O produto cartesiano fuzzy $A_1 \times A_2 \times ... \times A_n$ dos subconjuntos fuzzy $A_1, A_2, ..., A_n$ de $U_1, U_2, ..., U_n$, é caracterizado pela função de pertinência associada a relação fuzzy R sobre $A_1 \times A_2 \times ... \times A_n \subset U_1 \times U_2 \times ... \times U_n$:*

$$u_R(x_1, x_2, ..., x_n) = u_{A_1}(x_1) \wedge u_{A_2}(x_2) \wedge ... \wedge u_{A_n}(x_n),$$ *onde $\wedge$ é a T-norma min.*

3.3 Sistemas Baseados em Regras Fuzzy

Um Sistema Baseado em Regras Fuzzy (SBRF) realiza o processamento da informação de forma mais flexível, isto, aproxima as computações desse tipo de sistema do método realizado pela linha do raciocínio humano. As etapas de um sistema de inferência fuzzy estão ilustradas na Figura 2. Nela pode-se ver que, a primeira ação a ser realizada é a fuzzificação dos dados de entrada (crisp) transformando-os em dados fuzzy. Em seguida, os dados fuzzy são processados pelo motor de inferência fuzzy. Este motor, é formado por um agrupamento de regras fuzzy definidas pelo conhecimento de um especialista chamado base de regras. O processo final é executado

na etapa de defuzzificação cuja saída do sistema é dada através de um número real (crisp) obtido através da interpretação do conjunto fuzzy de saída.

Considerando a base de regras, um SBRF é definido como um mapeamento entre a entrada e a saída da forma $y = f(x)$, tal que, $x \in \mathbb{R}^n$ e $y \in \mathbb{R}^m$, onde, n representa a aridade das variáveis de entradas e m representa a aridade das possíveis saídas.

Os SBRF são amplamente utilizados no desenvolvimento de aplicações industriais e biomédicas, como por exemplo, nas aplicações de diagnóstico médico.

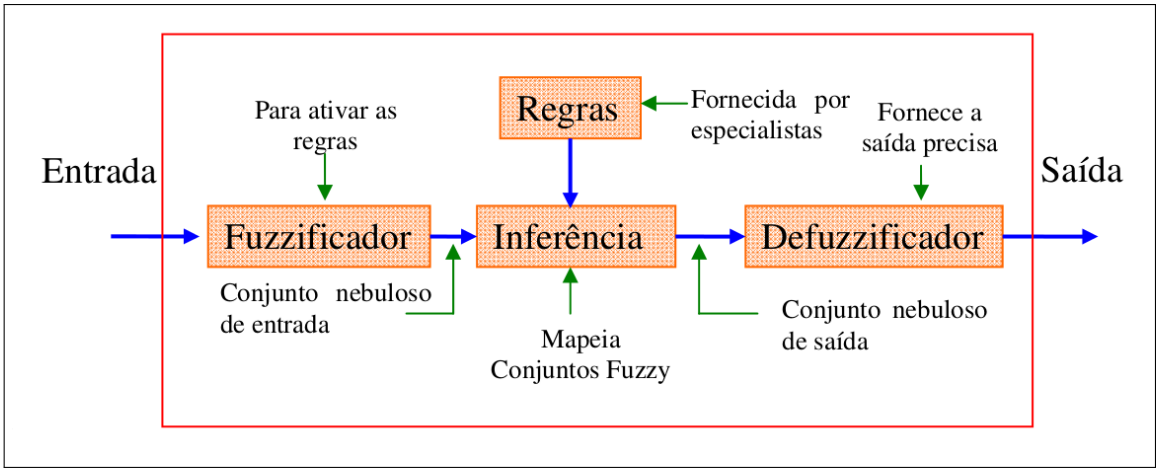

Figura 2 – Sistema de Inferência Fuzzy (SANCHEZ, 2009)

3.3.1 Fuzzificação

Fuzzificação é a transformação dos dados de entrada modelados por números reais (crisp) para valores fuzzy. O fuzzificador categoriza a informação de entrada mapeando cada valor real obtido para uma dada pertinência utilizando as funções fuzzy previamente definidas. Esta etapa, realiza o pré-processamento da informação transformando-a na informação fuzzy a qual será operada por outras etapas de um SBRF.

3.3.2 Regras e Inferência Fuzzy

As regras fuzzy são representadas como implicações lógicas que relacionam conjuntos fuzzy. Elas são produzidas de acordo com os conhecimentos de um especialista e enumeradas na forma de sentenças linguísticas.

Cada proposição fuzzy é trabalhada pelo motor de inferência por meio das técnicas de raciocínio aproximado. Os operadores lógicos fuzzy são utilizados para operar as proposições fuzzy definindo a relação existente na base de regras.

Uma regra fuzzy é uma sentença da forma **Se** X **é** A **então** Y **é** B, onde, A e B são conjuntos fuzzy em X e Y, respectivamente. Tal regra é uma relação fuzzy R entre A e B cuja função de pertinência $u_R(x, y)$ depende de $u_A(x)$ e $u_B(y)$ para cada

$(x, y) \in X \times Y$. Isto é, $u_{A \times B}(x, y) = u_A(x) \wedge u_B(y)$, onde, $\wedge$ denota uma instância de uma T-norma.

Considere a regra fuzzy **Se** a febre é “baixa” **então** a gripe é “leve”. Os termos “baixa“ e ”leve“ representam conjuntos fuzzy. Os fatos ”a febre é baixa“ e a ”a gripe é leve“ são as proposições utilizadas na composição da implicação. Este exemplo, descreve como o conhecimento pode ser organizado utilizando a técnica do raciocínio aproximado.

Geralmente um sistema fuzzy é formado por múltiplas regras fuzzy, para tratar isso, um operador da classe T-conorma é utilizado para realizar a operação de disjunção entre elas visando alcançar uma única resposta.

As regras fuzzy podem ser classificadas em tipos dependendo da composição dos operadores fuzzy (E/OU) e de acordo com o local aonde essa composição está presente, se é no antecedente ou no consequente de cada regra. A baixo estão representados os tipos de produções que podem ser usados para construir um motor de inferência (LIU et al., 2017; CHEN; KE; CHANG, 1990).

- Tipo 1: $\mathbf{R_i}$: **Se** a **então** b
- Tipo 2: $\mathbf{R_i}$: **Se** a_1 E a_2 E... E a_m **então** b
- Tipo 3: $\mathbf{R_i}$: **Se** a **então** b_1 E b_2 E... E b_m
- Tipo 4: $\mathbf{R_i}$: **Se** a_1 OU a_2 OU... OU a_m **então** b
- Tipo 5: $\mathbf{R_i}$: **Se** a **então** b_1 OU b_2 OU... OU b_m

Regras do Tipo 5 geralmente não são utilizadas na etapa de inferência fuzzy por não constituírem implicações específicas.

Os tipos de regras fuzzy listados podem ser combinados, por exemplo, pode-se compor uma regra combinando o Tipo 3 com o Tipo 4. Dessa maneira, um maior grau de refinamento é obtido enriquecendo a base de regras aumentando a aproximação ao raciocínio humano.

Algumas modelagens incluem um fator de certeza (CF) (μ) associado às regras para indicar o grau de crença depositado na regra a qual está associado e é determinado pelo conhecimento do especialista. Nesta abordagem, o fator de certeza é considerado na computação da regra pela operação de implicação. A obtenção do fator de certeza é dado através das medidas de crença e descrença na evidência que está sendo modelada pelo sistema (REDDY, 2017).

Exemplificação 2 ***(Produções Fuzzy):*** *Neste exemplo, está enumerada a seguinte base de produções fuzzy:*

1. *Se* d_1 *E* d_2 *então* d_5 *E* d_6*, com* $\mu_1 = 0.4$*.*

2. *Se* d_3 *então* d_6 *E* d_7*, com* $\mu_2 = 0.5$*.*

3. *Se* d_4 *OU* d_1 *então* d_5*, com* $\mu_3 = 0.2$*.*

4. *Se* d_7 *então* d_2*, com* $\mu_4 = 0.8$*.*

As variáveis d_i *representam as proposições fuzzy. Os fatores de certeza vinculados a cada regra são modelados por* μ_i*. A regra* 1 *exemplifica uma composição do Tipo 2 com o Tipo 3, veja que, o operador fuzzy* E *é utilizado no antecedente e no consequente da regra. As regras* $2, 3$ *e* 4 *são exemplos de produções fuzzy do Tipo 3, do Tipo 4 e do Tipo 1 respectivamente. Esse exemplo de base de regras, pode ser descrito formalmente através de uma Rede de Petri Fuzzy o que permite obter uma construção gráfica e simulável.*

3.3.3 Defuzzificação

A defuzzificação é considerada como o processo inverso da fuzzificação. Nesta etapa, é realizada a transformação de um conjunto fuzzy de saída previamente processado pelo motor de inferência em um valor real. Dessa forma, a defuzzificação é a conversão de valores numéricos fuzzy para valores precisos (crisps). Esta conversão é realizada pelos sistemas computacionais práticos porque eles devem retornar saídas satisfatórias e interpretáveis.

4 GRAMÁTICA DE GRAFOS COM ATRIBUTOS

Gramáticas de Grafos (GGs) são uma generalização das gramáticas de Chomsky, substituindo strings por grafos. Dessa forma, constituem um formalismo eficaz para especificar sistemas complexos, com características como concorrências, distribuição e reatividade. Grafos são usados para descrever os estados dos sistemas e regras de transformação de grafos modelam as mudanças entre os estados (ROZENBERG, 1997).

Gramática de Grafos com Atributos (GGA) é uma extensão da gramática de grafos convencional na qual pode-se associar valores de tipos abstratos aos elementos do grafo, permitindo também fazer uso de variáveis e operações em suas definições. Especificações algébricas podem ser usadas para definir tipos de dados e álgebras para descrever os valores que podem ser usados como atributos. Uma GGA é composta por um grafo tipo, um grafo inicial e um conjunto de regras. A base das definições apresentadas é a abordagem **DPO** que restringe a aplicação das regras utilizando uma condição chamada de Condição de Colagem. Nesta abordagem, as regras são representas por um morfismo total entre grafos e as derivações diretas são definidas por meio de duas construções algébricas. Logo, os grafos são considerados uma álgebra e a aplicação das regras é definida como uma construção algébrica chamada *pushout* (EHRIG; KORFF; LÖWE, 1991; CORRADINI; MONTANARI et al., 1996; ROZENBERG, 1997).

Todos os conceitos e definições enumerados neste capítulo são encontrados em (CAVALHEIRO; FOSS; RIBEIRO, 2017; EHRIG et al., 2006; FOSS, 2008; EHRIG; MAHR, 2012; ROZENBERG, 1997).

Estrutura do Capítulo:

Na Seção 4.1 estão enumeradas as definições, assim como, os exemplos referentes aos elementos sintáticos que fazem parte de uma GGA. A Seção 4.2 apresenta as definições e os exemplos referentes ao comportamento da GGA.

4.1 Sintaxe

A sintaxe descreve os elementos utilizados na modelagem a ser construída, nas GGs os elementos fundamentais são os grafos. Neste trabalho, são utilizados dígrafos para modelar sistemas, isto é, grafos orientados cujas relações de adjacência não são simétricas. Isto significa que, em um par do tipo (v, w), v pode estar conectado a w, mas não necessariamente w estará conectado a v, modelando uma conexão orientada. Para definir essas relações são usadas as funções de origem e destino de cada aresta.

Definição 7 ***(Grafo e Morfismo de Grafos):*** *Um* ***grafo*** *é uma tupla* $G = (Vert_G, Edge_G, source_G, target_G)$*, onde,* $Vert_G$ *é um conjunto não vazio que representa os vértices ou nodos do grafo,* $Edge_G$ *é um conjunto de arestas do grafo,* $source_G : Edge_G \rightarrow Vert_G$ ***e*** $target_G : Edge_G \rightarrow Vert_G$ *são função totais que mapeiam origem e destino das arestas, respectivamente. Dados dois grafos* $G = (Vert_G, Edge_G, source_G, target_G)$ ***e*** $H = (Vert_H, Edge_H, source_H, target_H)$ *um* ***morfismo de grafos*** $f : G \rightarrow H$ *é um par de funções totais* $g_V : Vert_G \rightarrow Vert_H, g_E : Edge_G \rightarrow Edge_H$*, tal que* f *comuta com as funções de origem e destino, isto é:*

$$g_V \circ source_G = source_H \circ g_E \text{ e } g_V \circ target_G = target_H \circ g_E.$$

Um morfismo de grafos pode ser injetivo ou sobrejetivo se ambos os componentes são funções injetivas ou sobrejetivas, respectivamente. A categoria de grafos e morfismo de grafos é denotada por **Graph**.

Uma assinatura $SIG = (S, OP)$ consiste em um conjunto S de *sorts* (tipos) e um conjunto OP de constantes e símbolos de operações. Dado um conjunto de variáveis X de tipos definidos em S, o conjunto de termos sobre SIG é denotado por $T_{OP}(X)$, isto é definido indutivamente afirmando que todas as variáveis e constantes são termos e, em seguida, todas as aplicações possíveis de símbolos e de operações definidos em OP para termos existentes também são termos. Uma equação é um par de termos $(t1, t2)$ e é usualmente denotado por $t1 = t2$. Uma especificação é um par $SPEC = (SIG, Eqns)$ consistindo de uma assinatura e um conjunto de equações sobre esta assinatura. Uma álgebra para especificação $SPEC$, ou $SPEC$-álgebra, consiste em um conjunto carregador para cada *sort* em S e uma função para cada operação em OP, tal que todas as equações em $Eqns$ sejam satisfeitas. A satisfação de uma equação é verificada substituindo-se todas as variáveis existentes na equação por valores correspondentes nos conjuntos carregadores e, se a igualdade for válida para todas as substituições possíveis então a equação é satisfeita. Dadas duas $SPEC$-álgebra, um homomorfismo entre elas é um conjunto de funções que mapeiam os correspondentes conjuntos carregadores e que são compatíveis com todas as funções das duas álgebras. O conjunto obtido pela união disjunta de todos os conjuntos

carregadores da álgebra A é denotado por $\mathcal{U}(A)$ (CAVALHEIRO; FOSS; RIBEIRO, 2017; ROZENBERG, 1997).

Neste trabalho são associados atributos somente para vértices.

Um grafo com atributos definido em uma especificação $SPEC$ é composto pela adição de uma $SPEC$-álgebra A, um conjunto de atributos e um par de funções que associam os atributos aos vértices e a seus valores. As funções val_G e $attrv_G$ são responsáveis por definir essa relação. A função val_G associa cada atributo a um valor de algum conjunto carregador da $SPEC$-álgebra A e a função $attrv_G$ associa cada atributo a um vértice do grafo, indicando a qual vértice esse atributo pertence. Cada tipo de grafo de uma gramática é associado a um diferente tipo de $SPEC$-álgebra. Para o grafo tipo, tem-se uma álgebra final, a qual permite vincular cada atributo a um tipo de dados abstrato. Para o grafo inicial tem-se uma álgebra inicial, onde, os valores associados aos atributos são constantes. Já os grafos das regras fazem uso da álgebra de termos, a qual permite utilizar constantes, variáveis e expressões como valores para os atributos.

Definição 8 ***(Grafo com Atributos e Morfismo de Grafos com Atributos):*** *Dada uma especificação $SPEC$, um **grafo com atributos** é uma tupla $G = (Vert_G, Edge_G, source_G, target_G, A, Attr_G, val_G, attrv_G)$, onde, $Vert_G, Edge_G, source_G$ e $target_G$ são os elementos que compõem um grafo, A é uma $SPEC$-álgebra, $Attr_G$ é um conjunto chamado de conjunto de atributos, e $val_G : Attr_G \to \mathcal{U}(A)$, $attrv_G : Attr_G \to Vert_G$ são funções totais. Elementos de $Attr_G$ são chamados de arestas atributo, ou somente atributos. Um **morfismo entre os grafos com atributos** G e H **é uma tripla** $g = (g_{Graph}, g_{Alg}, g_A)$ consistindo de um morfismo de grafos $g_{Graph} = (g_V, g_E)$, um homomorfismo de álgebras g_{Alg} e uma função total $g_A : Attr_G \to Attr_H$ entre os correspondentes componentes que são compatíveis com a atribuição:*

$$g_{Alg} \circ val_G = val_H \circ g_A, \; g_V \circ attrv_G = attrv_H \circ g_A.$$

*Um morfismo entre grafos com atributos é chamado injetivo se todos os seus componentes são injetivos. A categoria de grafos com atributos e morfismo de grafos com atributos é denominada **AGraph**.*

O grafo tipo define restrições para a estrutura gráfica e para os tipos de valores dos atributos. Dessa forma, elementos gráficos que não fazem parte do grafo tipo não podem ser usados nos grafos instâncias que modelam o sistema. Com relação aos atributos, o grafo tipo estabelece o tipo de dados para cada aresta atributo, por isso, a álgebra vinculada ao grafo tipo é considerada final. Ou seja, uma álgebra na qual todos os conjuntos carregadores são unitários. Na construção do grafo tipo, usa-se o nome do *sort* (tipo) do atributo como o único elemento no conjunto carregador que o representa.

Definição 9 ***(Grafo Tipo, Grafo Tipado e Morfismo de Grafos Tipados):*** *Dada uma especificação* $SPEC$*, um* ***grafo tipo*** *é um grafo com atributos* $T = (Vert_T, Edge_T, source_T, target_T, A, Attr_T, val_T, attrv_T)$ *no qual todos os conjuntos carregadores de* A *são unitários. Um* ***grafo (com atributos) tipado*** *é uma tupla* $G^T = (G, tG, T)$*, onde,* G *é um grafo com atributos,* T *é um grafo tipo e* $t_G : G \to T$ *é um morfismo de grafos com atributos, chamado de morfismo de tipagem. Um* ***morfismo de grafos (com atributos) tipados*** $f^T : G^T \to H^T$ *é um morfismo de grafos* $f : G \to H$ *tal que,* $tG_V = tH_V \circ g_V$*,* $tG_E = tH_E \circ g_E$ *e* $tG_A = tH_A \circ g_A$*.*

Um morfismo de grafos tipados é dito injetor se todos os seus componentes forem injetores. A categoria de grafos (com atributos) e morfismos de grafos (com atributos) tipados sobre T *é denominada* ***TAGraph(T)****.*

Uma vez que, somente grafos tipados são utilizados, se omitirá a palavra “tipado” por questões de simplificação.

Exemplificação 3 ***(Grafo Tipado com Atributos e Morfismo de Grafos Tipados com Atributos):*** *Dado o grafo tipo com atributos* T*, ilustrado na Figura3, o grafo* L *é uma instância de* T*, onde, as setas tracejadas simbolizam o morfismo de tipagem de* L*. Cada elemento de* L *é mapeado para* T*, onde o atributo cont é mapeado para o seu tipo (Nat). Na Figura4 é apresentado um exemplo de morfismo entre grafos tipados com atributos cujos elementos presentes no grafo* K *estão relacionados com os elementos do grafo* L *através do morfismo de grafos* g*. Dessa forma, os elementos pertencentes ao grafo* K *são mapeados no grafo* L *de acordo com o seu respectivo tipo.*

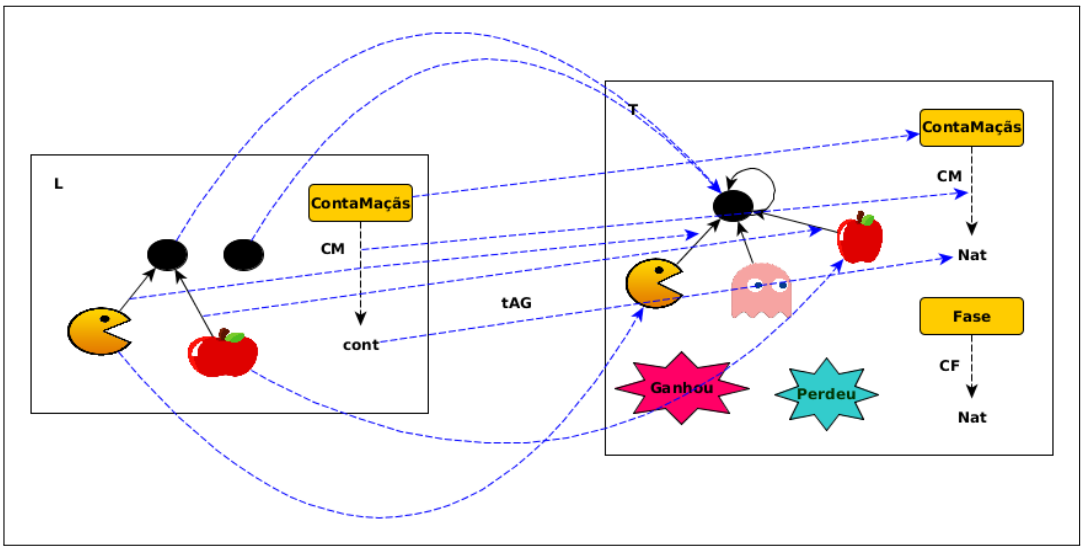

Figura 3 – Grafo Tipado com atributos L^T

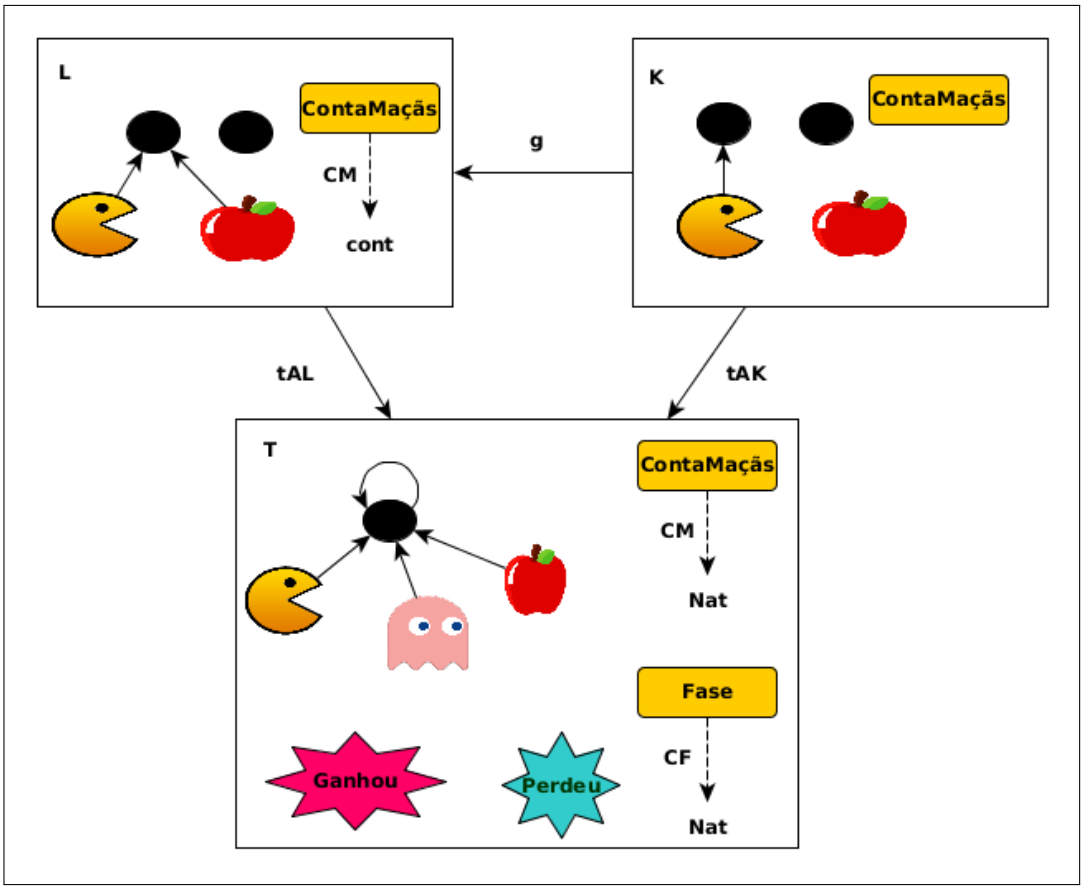

Figura 4 – Morfismo de grafos tipados com Atributos

As regras especificam o padrão de comportamento do sistema, isto é, as ações que podem ser realizadas. Uma regra é aplicada sobre um grafo estado do sistema causando a deleção, preservação e criação de itens, levando em conta os atributos e respeitando o homomorfismo de álgebras.

Na abordagem DPO, as regras de uma GGA são definidas por um span de morfismos, isto é, um par de morfismos totais que mapeiam um grafo de interface nos grafos dos lados esquerdo (LHS) e direito (RHS) da regra. O grafo do LHS determina o contexto no qual uma regra pode ser aplicada. Já o grafo do RHS determina o efeito da aplicação desta regra, caso haja uma imagem do LHS em um grafo estado. Esse efeito pode ser apagar, preservar ou criar elementos no grafo estado. Os elementos que devem ser preservados, são aqueles que correspondem ao grafo de interface. Os elementos que são apagados (ou criados), são aqueles correspondentes aos elementos que estão no grafo do LHS (ou RHS) mas não são imagem do morfismo que mapeia a interface no LHS (ou RHS).

A álgebra dos grafos das regras é a de termos com relação a um dado conjunto de variáveis. Sem perda de generalidade, restringe-se os valores dos atributos destes grafos à variáveis. Equações sobre essas variáveis podem estar associadas a cada regra para restringir sua aplicação e relacionar os valores dos atributos (variáveis) dos

grafos LHS e RHS a termos que podem ter operações.

Definição 10 ***(Regras):*** *Dada uma especificação* $SPEC = (SIG, Eqns)$ *e um conjunto* X *de variáveis sobre os sorts (tipos) de SPEC. Uma regra sobre* $SPEC$ *com tipo* T *é uma tupla* $r = (\alpha, X, rEqns)$*, onde,*

- $\alpha = L \overset{\alpha L}{\hookleftarrow} K \overset{\alpha R}{\rightarrowtail} R$ *é um span de morfismos entre grafos com atributos sobre a especificação* $(SIG, \varnothing)$*, com*
  $L = (Vert_L, Edge_L, source_L, target_L, T_{OP}(X), Attr_L, val_L, attrv_L)$,
  $K = (Vert_K, Edge_K, source_K, target_K, T_{OP}(X), Attr_K, val_K, attrv_K)$ *e*
  $R = (Vert_R, Edge_R, source_R, target_R, T_{OP}(X), Attr_R, val_R, attrv_R)$*, no qual a álgebra é a identidade dos termos da álgebra* $T_{OP}(X)$*,* $rng(val_L) \subseteq X$*,* $rng(val_K) \subseteq X$ *e* $rng(val_R) \subseteq X$*.*
- $rEqns$ *é um conjunto de equações (condicionais) usando termos de* $T_{OP}(X)$*.*

Como a componente da álgebra nos morfismos das regras é a identidade, a aplicação das regras não alteram a álgebra do grafo estado.

Exemplificação 4 ***(Regras):*** *Na Figura5 é ilustrada a regra que modela a ação que o Pacman realiza quando se alimenta. O grafo* $L4$ *é o LHS da regra, o grafo de interface é o* $K4$ *e o grafo* $R4$ *é o RHS da regra. O grafo* $L4$ *estabelece os pré-requisitos para a aplicação da regra: dois vértices de lugar (círculos pretos), uma maçã e o pacman conectados no mesmo vértice de lugar e o vértice ContaMaçãs com o atributo CM. Em caso destes elementos estarem presentes em um grafo estado, o efeito da aplicação da regra neste grafo será: (i) apagar a aresta que conecta a maçã ao seu lugar e criar uma nova aresta conectando-a ao outro lugar e (ii) associar o valor inicial do atributo CM à variável cont, apagar o atributo CM e criá-lo novamente associando-o ao valor da variável cont1. O valor da variável cont1 é definido pela equação cont1=add(cont, 1).*

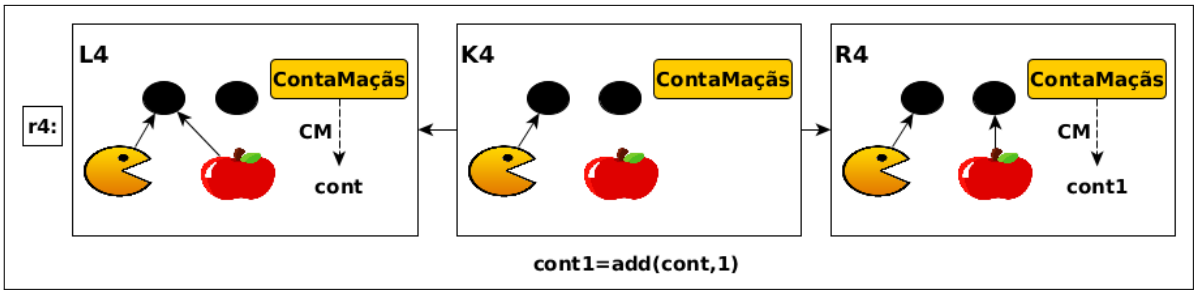

Figura 5 – Regras

Uma GGA, com relação a uma especificação algébrica $SPEC$, é composta por um grafo tipo, um grafo inicial e um conjunto de regras, onde os últimos são tipados sobre o grafo tipo da gramática e todos com atributos em $SPEC$.

Definição 11 ***(Gramática de Grafos com Atributos):*** *Dada uma especificação $SPEC$ e uma $SPEC$-álgebra A, uma* ***gramática de grafos (tipada) com atributos*** *é uma tupla $GGA = (Type, G0, Rules)$, tal que $Type$ (o tipo da gramática) é um grafo tipo com atributos sobre $SPEC$, $G0$ (grafo inicial da gramática) é um grafo com atributos tipado sobre $Type$ usando a álgebra A, e $Rules$ é um conjunto de regras sobre $SPEC$ com tipo $Type$.*

Exemplificação 5 ***(Gramática de Grafos com Atributos):*** *Este exemplo, descreve a especificação do jogo Pacman. A especificação algébrica associada a esta gramática é mostrada na Figura 6. Ela especifica o tipo de dado Nat (conjunto dos números Naturais) e a operação que pode ser realizada sobre este tipo é a add, a qual formaliza a operação de soma. Na Figura7 estão ilustrados o grafo tipo (T) e o grafo inicial (G0). O grafo T determina os tipos dos elementos presentes na gramática. O vértice representado pelo círculo preto e a aresta conectada a ele definem o tipo dos vértices dos lugares do tabuleiro e a forma como eles estão conectados. A maçã, o fantasma e o Pacman podem estar conectados apenas a vértices de lugar. Os vértices ContaMaçãs e Fase possuem os atributos CM e CF, respectivamente, que são usados como contadores que acumulam o número de maçãs que o Pacman comeu e a fase atual do jogo, ambos com valores do tipo Nat. Os vértices Ganhou e Perdeu indicam os tipos dos vértices que são utilizados para descrever se o jogador chegou a uma situação de sucesso ou falha, respectivamente. O grafo G0 é formado por um tabuleiro de doze lugares conectados por setas duplas, as quais representam duas arestas, uma em um sentido e a outra em outro. No tabuleiro, tem-se um Pacman, uma maçã, e um fantasma. O número de maçãs comidas inicialmente é zero e o jogo começa na Fase 1. As regras da gramática estão ilustradas na Figura8, elas definem as possíveis ações do jogo (mudanças de estado). Por questões de simplificação omitimos a representação do grafo K no morfismo $L \overset{\alpha L}{\hookleftarrow} K \overset{\alpha R}{\rightarrowtail} R$ utilizando a notação $L \overset{a_i}{\rightarrow} R$, onde $i = \{1, 2, 3, 4, 5, 6\}$. As regras $r1$ e $r2$ descrevem como o Pacman e o fantasma se locomovem, respectivamente, no tabuleiro. A regra $r3$ descreve a ação de perder o jogo: se o Pacman e o fantasma estiverem conectados ao mesmo local o Pacman é destruído e o vértice Perdeu será criado. A regra $r4$ especifica a ação que o Pacman realiza ao se alimentar: quando o Pacman e a maçã estão conectados ao mesmo local, o valor do atributo CM é atualizado, contabilizando uma maçã, e a maçã é conectada em outro local aleatório do tabuleiro. A regra $r5$ modela a mudança de fase: quando o Pacman come quatro maçãs o jogo pode trocar de fase atualizando o valor do atributo CF, zerando o número de maçãs e adicionado um fantasma no jogo. A regra $r6$ especifica a ação de ganhar o jogo: se o Pacman resistiu as mudanças de fase e contabilizou o total de doze maçãs então o jogo termina criando o vértice Ganhou.*

TPacman: sorts Nat
Opns
0: → Nat
succ: → Nat
add: Nat x Nat → Nat
eqns
∀x, y, z ∈ Nat
add(x, 0)= x
add(x, succ(y))= succ(add(x, y))
add(add(x,y), z)= add(x, add(y, z))

Figura 6 – Especificação Algébrica $SPEC_{TPacman}$

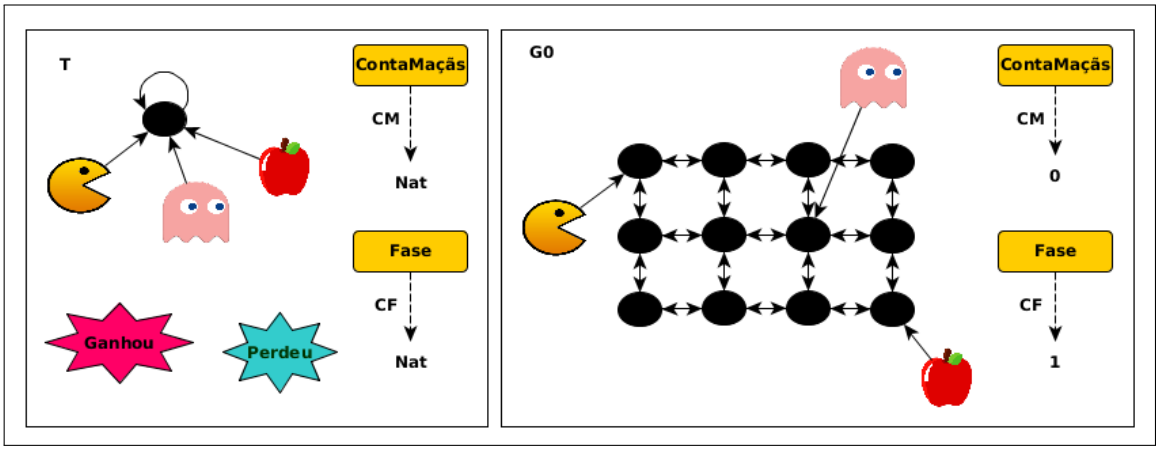

Figura 7 – Grafo tipo (T) e Grafo inicial (G0)

4.2 Semântica

A semântica operacional de uma gramática de grafos é definida em termos de derivações, que são as aplicações das regras a grafos que representam os estados do sistema, ou seja, grafos estados. Para que uma regra possa ser aplicada em um grafo estado, deve-se encontrar uma imagem (ocorrência) do LHS desta regra no grafo estado. Essa ocorrência é definida por um morfismo de grafos. Quando são considerados grafos com atributos, as variáveis da regra devem ser associadas a valores reais da álgebra do grafo estado. Além disso, a ocorrência deve garantir que todas as equações da especificação e da regra sejam satisfeitas escolhendo uma atribuição de valores para as variáveis. Para isso, transforma-se a regra em uma correspondente que possui como álgebra uma álgebra de termos quociente (EHRIG; MAHR, 2012) e a ocorrência passa a relacionar o LHS desta nova regra com o grafo estado, relacionando assim a álgebra de termos quociente com a álgebra do grafo estado.

Definição 12 ***(Ocorrência):*** *Dados uma especificação* $SPEC = (SIG, Eqns)$ *e uma*

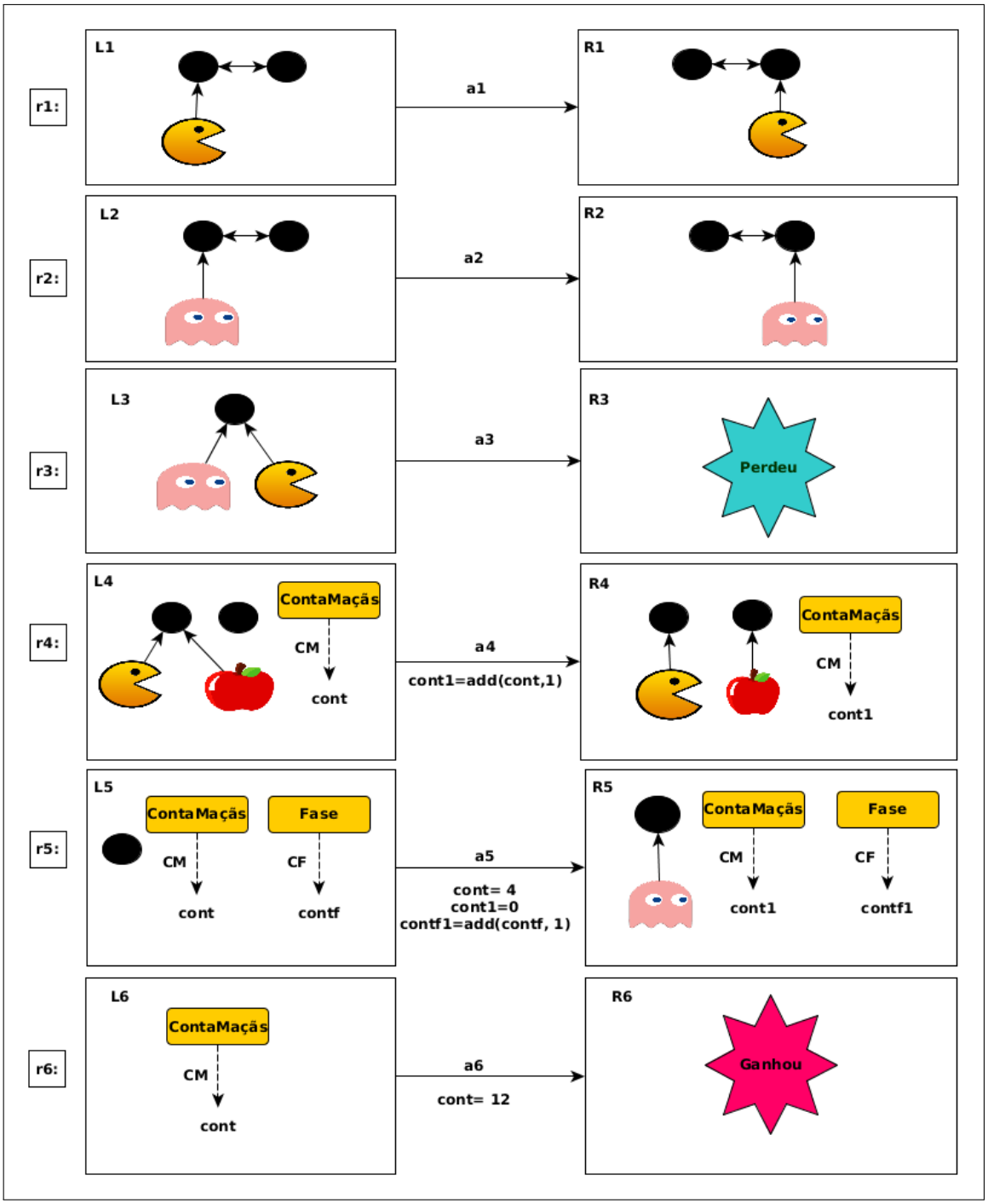

Figura 8 – Conjunto de Regras

regra $r = (\alpha, X, rEqns)$ *sobre* $SPEC$*, com LHS* $L = (Vert_L, Edge_L, source_L, target_L, T_{OP}(X), Attr_L, val_L, attrv_L)$*, e um grafo* G^T *sobre* $SPEC$*. Uma* ***ocorrência*** $m : \overline{L^T} \rightarrow G^T$ *é um morfismo de grafos injetor* $m = (m_V, m_E, m_{Alg}, m_A)$ *tal que* $\overline{L^T} = (Vert_L, Edge_L, source_L, target_L, T_{eq}(X), Attr_L, \overline{val_L}, attrv_L)$*, onde* $T_{eq}(X)$ *é uma álgebra obtida pela construção da álgebra termos quociente da especificação* $(SIG, Eqns \cup rEqns)$ *usando o conjunto de variáveis* X*, e, para todos os elementos* $a \in Attr_L$*,* $\overline{val_L}(a) = [val_L(a)]$*.*

De forma intuitiva, dado um conjunto de variáveis X e uma álgebra A, se for definida uma função de avaliação $eval : X \to \mathcal{U}(A)$, há uma forma única para construir o homomorfismo de álgebras $m_{Alg} : T_{eq}(X) \to \mathcal{U}(A)$ (caso ele exista para esta atribuição de valores para variáveis). Verifica-se se todas as equações em $Eqns \cup rEqns$ são satisfeitas pela atribuição. Caso não sejam, $eval$ não induz o homomorfismo de álgebras e, consequentemente, não existe uma ocorrência usando a função $eval$. Caso contrário, constrói-se a extensão de $eval$ para (a classe de equivalência de) termos, denotada por $\overline{eval} : Teq(X) \to \mathcal{U}(A)$, onde $\overline{eval} = m_{Alg}$.

Exemplificação 6 ***(Ocorrência):*** *Na Figura9 está exemplificada uma ocorrência do lado esquerdo da regra $r4$ no grafo G, a ocorrência é marcada pelo retângulo com linhas tracejadas. Nela identifica-se o casamento da variável cont com o valor zero.*

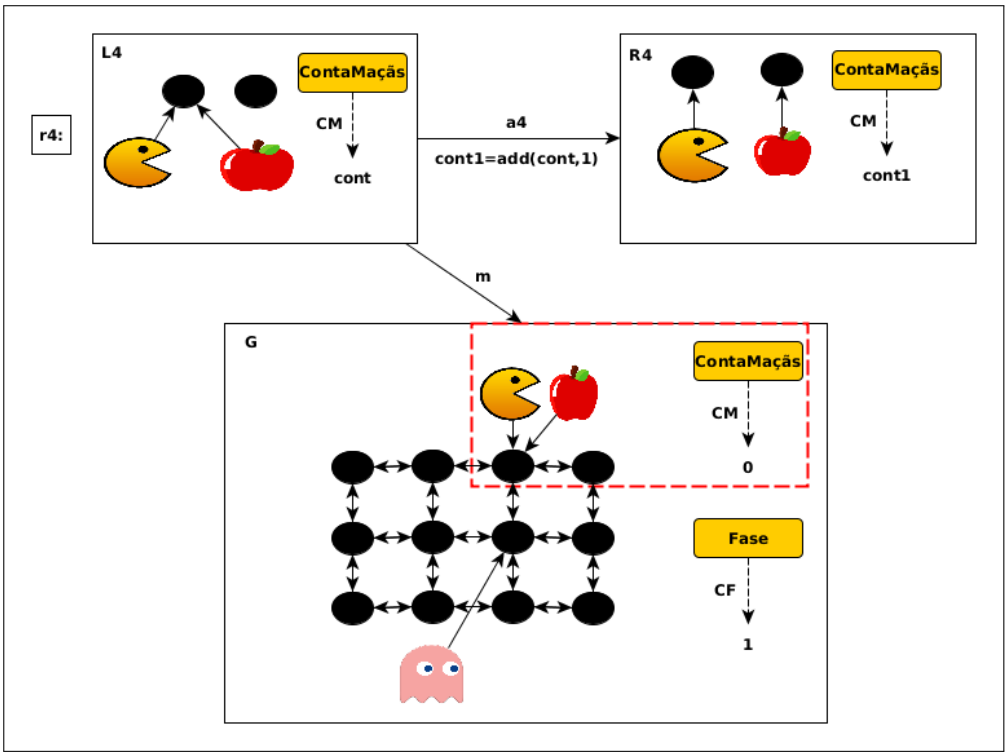

Figura 9 – Ocorrência

Existem duas situações que devem ser tratadas com atenção ao aplicar uma regra em um grafo estado G:

1. quando um vértice é especificado para exclusão e há arestas ou atributos conectados a ele em G que não estão na imagem do LHS da regra (condição de arestas pendentes);

2. no caso em que dois elementos da regra, um a ser excluído e o outro a ser preservado, são mapeados para o mesmo elemento de G, ou dois elementos excluídos são mapeados para o mesmo em G (condição de identificação).

Para garantir que tais situações não ocorram quando uma regra é aplicada tem-se uma condição chamada de **Condição de Colagem**. Na abordagem DPO, esta condição é obrigatória porque de outra forma não seria tecnicamente possível obter o resultado da aplicação. Visto que neste trabalho somente ocorrências injetoras estão sendo consideradas, isto é, a condição de colagem restringe-se à condição de arestas pendentes.

Definição 13 ***(Condição de Arestas Pendentes):*** *Dados uma regra* $r = (\alpha, X, rEqns)$*, com* $\alpha = L \overset{\alpha L}{\hookleftarrow} K \overset{\alpha R}{\rightarrowtail} R$ *e* $\alpha L = (\alpha L_{Graph}, \alpha L_{Alg}, \alpha L_A) : K^T \to L^T$*, um grafo (com atributos) tipado* G^T *e uma ocorrência* $m = (m_V, m_E, m_{Alg}, m_A) : L^T \to G^T$*. A ocorrência* m *satisfaz a* ***condição de arestas pendentes*** *se satisfizer as seguintes condições:*

- *Nenhuma aresta* $e \in Edge_G - m_E(Edge_L)$ *é incidente para qualquer vértice em* $m_V(Vert_L - \alpha L_V(Vert_K))$*.*
- *Nenhum atributo* $a \in Attr_G - m_A(Attr_L)$ *é incidente para qualquer vértice em* $m_V(Vert_L - \alpha L_V(Vert_K))$*.*

A aplicação de uma regra r a um grafo estado G, com relação à ocorrência m do lado esquerdo de r em G, descreve uma derivação de G em um novo grafo H ($G \overset{r,m}{\Rightarrow} H$), o qual é obtido a partir de G eliminando-se a imagem dos elementos deletados por r e adicionando-se instâncias dos elementos criados por r. Na abordagem algébrica DPO para GGA, essa derivação é formalizada pela construção de dois pushouts na categoria **TAGraph(T)**. No primeiro pushout é construído um grafo intermediário D, o complemento do pushout, o qual difere-se de G, pela eliminação dos elementos identificados pela regra para serem deletados. No segundo, constrói-se o grafo H, o objeto do pushout, no qual os elementos criados são adicionados a D. Para garantir a existência do complemento do pushout D, a ocorrência m deve satisfazer a **condição de arestas pendentes**.

Definição 14 ***(Aplicação de Regra):*** *Dadas uma especificação* $SPEC$*, uma regra* $r = (\alpha, X, rEqns)$ *sobre* $SPEC$*, com tipo* T *e* $\alpha = L \overset{\alpha L}{\hookleftarrow} K \overset{\alpha R}{\rightarrowtail} R$*, e uma ocorrência* $m = (m_V, m_E, m_{Alg}, m_A)$ *de* L *em um grafo* G^T*. A* ***aplicação da regra*** r *em* G *com relação a* m*, denotada por* $G \overset{r,m}{\Rightarrow} H$*, é definida pelos pushouts* (1) *e* (2) *(do diagrama abaixo) na categoria* ***TAGraph(T)****, caso* m *satisfaça a condição de arestas pendentes, resultando no grafo* H^T*.*

$$\begin{array}{ccccc} L & \xleftarrow{\alpha L} & K & \xrightarrow{\alpha R} & R \\ {\scriptstyle m}\downarrow & (1) & \downarrow & (2) & \downarrow \\ G & \longleftarrow & D & \longrightarrow & H \end{array}$$

Exemplificação 7 ***(Aplicação de Regra):*** *Na Figura10 é ilustrada a aplicação da regra $r4$ em um grafo estado G transformando-o no grafo estado H. Considerando a ocorrência m, definida na Figura 9, o complemento do pushout D é obtido eliminando os elementos identificados pela ocorrência m no grafo G os quais pertencem ao grafo $L4$ mas não pertencem ao grafo $K4$ (isto é definido pelo morfismo que os relaciona). Por isso, o atributo CM e a aresta que conecta a maçã ao vértice de lugar não fazem parte do grafo D. O objeto pushout H é obtido adicionando os elementos presentes no grafo $R4$ os quais não fazem parte do grafo $K4$ (isto também é definido pelo morfismo que os relaciona) no grafo D. Nesta etapa, são adicionados os seguintes elementos no grafo D obtendo-se o grafo H: (i) uma nova aresta que conecta a maçã a um vértice de lugar aleatório no tabuleiro e (ii) o atributo CM é atualizado com um novo valor definido pela equação cont1 = add(cont, 1).*

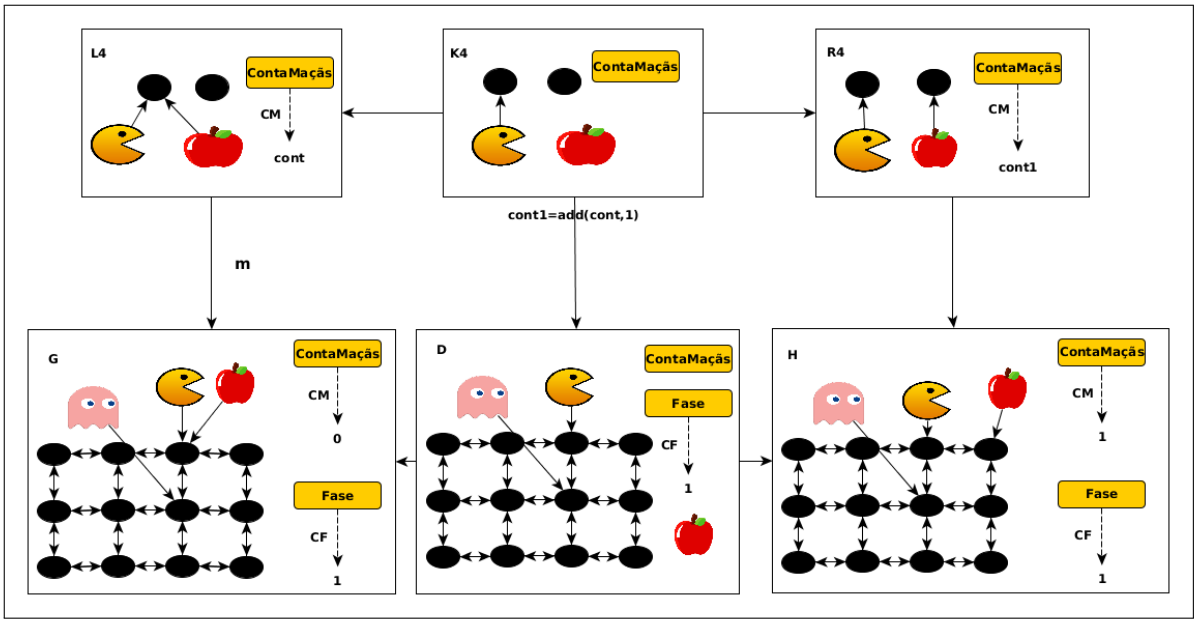

Figura 10 – Aplicação da regra r4 no grafo G

5 REDES DE PETRI FUZZY

Redes de Petri Fuzzy (RPFs) constituem uma linguagem de especificação formal que permite simular e avaliar sistemas fuzzy. São utilizadas na etapa de inferência provendo uma disposição gráfica para as produções fuzzy. As RPFs são utilizadas para especificar diversas aplicações industriais considerando a inclusão da incerteza presente no raciocínio humano (ZHOU; ZAIN, 2016; VIRTANEN, 1995).

Todas as definições aqui presentes podem ser encontras em (VIRTANEN, 1995; LIU et al., 2017; PAVLISKA, 2006; SURAJ, 2012; ZHOU; ZAIN, 2016; CARDOSO; VALETTE; DUBOIS, 1996; GENRICH, 1991; VEUNG et al., 1996; CARDOSO; VALETTE; DUBOIS, 1999; CHEN; KE; CHANG, 1990; KIM; YANG, 2018; CHEN, 2000; ZURAWSKI; ZHOU, 1994).

Estrutura do Capítulo:

Na Seção 5.1 são apresentados os principais conceitos correspondentes às RPFs. Na Seção 5.2 enumeram-se os principais conceitos referentes às Redes de Petri Fuzzy Generalizadas, assim como, suas definições e seu comportamento.

5.1 Redes de Petri Fuzzy Convencionais

Uma RPF representa graficamente produções fuzzy utilizando um fator de certeza e um limiar. São redes consideradas livres de conflito porque em uma RPF não existem recursos disponíveis, mas sim, proposições que podem ser compartilhadas por diferentes regras fuzzy ao mesmo tempo. Tokens podem ser removidos ou não das pré-condições das transições dependendo do modo do disparo escolhido, pois, o que acontece durante a dinâmica das marcações é a propagação da verificação da verdade das proposições (LIU et al., 2017).

Uma RPF é composta de lugares, proposições, tokens, transições e marcação inicial. Cada lugar da rede é vinculado a uma proposição fuzzy. Os lugares podem conter somente um único token que possui valores determinados no intervalo fuzzy $[0, 1]$, os quais definem o valor verdade das proposições associadas. Cada transição da RPF representa uma regra fuzzy cujos tipos estão enumerados na Seção 3.3. O

antecedente da regra dá origem as pré-condições da transição e o consequente dá origem as pós-condições da transição. Além disso, são associados a cada transição um fator de certeza $(CF = \mu)$ e um limiar (λ) que determina o grau mínimo de verdade requerido para as proposições do antecedente para que a transição seja disparada. Os limiares da rede correspondem ao conceito fuzzy α-Cut o qual é responsável por seccionar a função de pertinência horizontalmente em um determinado ponto α que demarca um limite onde as pertinências consideradas são maiores ou iguais a α. Dessa maneira, são designados os possíveis valores verdade para as proposições do antecedente de acordo com a modelagem fuzzy (MOCKOR, 2012; KIM; YANG, 2018). As marcações de uma RPF definem os valores verdade das proposições que estão vinculadas aos lugares. Lugares que não possuem tokens contém proposições com grau de verdade igual a zero. Os valores associados aos tokens são alterados com os disparos das transições gerando novas marcações. A primeira atribuição de valores aos tokens na RPF é denominada marcação inicial.

Exemplificação 8 ***(Rede de Petri Fuzzy):*** *Na Figura 11 é ilustrada uma RPF que representa a base de produções fuzzy mostrada no Exemplo 2.*

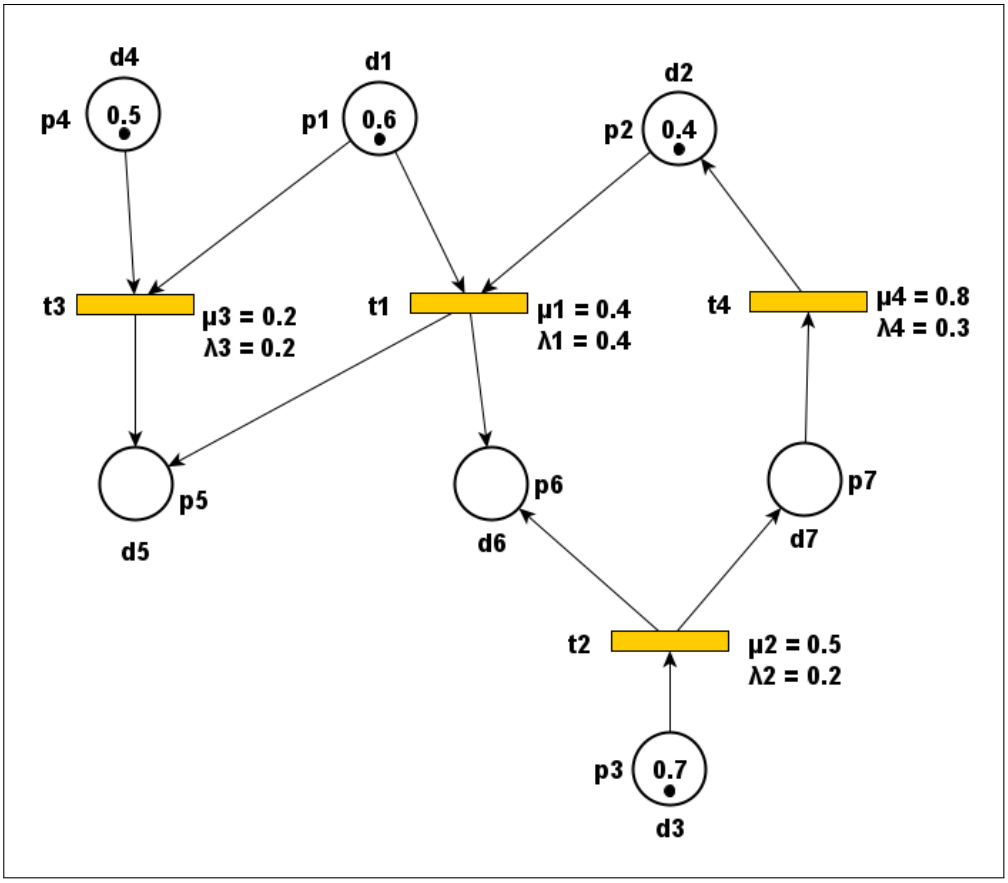

Figura 11 – Exemplo de uma RPF

Nesta RPF, existem sete lugares representados por círculos cujos rótulos p_1, ..., p_7 definem os nomes destes lugares. Para cada lugar existe uma proposição associada,

estas são determinadas pelos rótulos d_1, ..., d_7. As transições $t1$, ..., t_4 são modeladas por retângulos cujas arestas dão o sentido das pré(pós)-condições. Os rótulos μ_i e λ_i representam, respectivamente, o fator de certeza e o limiar para cada transição. Sendo assim, para a transição t_1 o fator de certeza é o $\mu_1 = 0.4$ e o limiar é o $\lambda_1 = 0.4$. Os lugares p_1 e p_2 são os lugares de entrada e constituem as pré-condições e os lugares p_5 e p_6 são os lugares de saída e constituem as pós-condições. Na transição t_2 o fator de certeza é o $\mu_2 = 0.5$ e o limiar é o $\lambda_2 = 0.2$ cujo lugar de entrada p_3 define a pré-condição e os lugares de saída p_6 e p_7 definem as pós-condições. Para a transição t_3 o fator de certeza é o $\mu_3 = 0.2$ e o limiar é o $\lambda_3 = 0.2$ cujas pré-condições são definidas pelos lugares p_1 e p_4 e a pós-condição é definida pelo lugar p_5. A transição t_4 possui como fator de certeza o $\mu_4 = 0.8$ e como limiar o $\lambda_4 = 0.3$, onde o lugar p_7 descreve a pré-condição e o lugar p_2 descreve a pós-condição. A distribuição dos tokens é definida pelas marcações da rede e é representada pelas pertinências não nulas. Para este exemplo, a primeira atribuição de valores aos tokens é dada pela marcação inicial $M_0(0.6, 0.4, 0.7, 0.5, 0, 0, 0)$.

A configuração da RPF é utilizada para representar os tipos de regras fuzzy listados na Seção 3.3. Os principais formatos para estes tipos são apresentados por (CHEN; KE; CHANG, 1990). A Figura 12 ilustra as configurações possíveis de acordo com os referentes tipos de regras, onde:

- **Regra Tipo 1**: Modela o tipo de regra mais simples $(d_j \rightarrow d_k)$. Quando uma transição é disparada o token da pré-condição é consumido e um novo token y_k é criado na pós-condição definindo o valor verdade da proposição d_k o qual é obtido multiplicando o valor verdade da proposição d_j pelo fator de certeza da transição t_i, isto é, $y_k = x_j * \mu_i$.

- **Regra Tipo 2**: Modela a associação da conjunção no antecedente da regra $(d_{j1} \wedge ... \wedge d_{jn} \rightarrow d_k)$. Quando este tipo de transição é disparada os tokens dos lugares de entrada são consumidos e um novo token y_k é criado na pós-condição definindo o valor verdade da proposição d_k. Para isso, os valores verdade das proposições d_{j1}, ..., d_{jn} presentes nas pré-condições são agregados através da operação min cuja saída é multiplicada com o fator de certeza da transição t_i. Esta operação é determinada pela equação $y_k = Min(x_{j1}, ..., x_{jn}) * \mu_i$.

- **Regra Tipo 3**: Modela a associação da conjunção no consequente da regra $(d_j \rightarrow d_{k1} \wedge ... \wedge d_{kn})$. Quando a transição é disparada o token do lugar de entrada é consumido e em cada lugar de saída p_{k1}, ..., p_{kn} um token y_k é criado determinando os valores verdades das proposições do consequente $d_{k1}, ..., d_{kn}$. Assim, o valor verdade de cada proposição d_{kn} é obtido multiplicando o valor verdade da proposição d_j pelo fator de certeza da transição t_i, ou seja, $y_{k1} = ... = y_{kn} = x_j * \mu_i$.

- **Regra Tipo 4**: Modela a associação da disjunção no antecedente da regra $(d_{j1} \vee ... \vee d_{jn} \rightarrow d_k)$. Neste tipo de regra, cada proposição do antecedente gera uma transição cujo o único lugar de saída é o associado a proposição d_k. Quando uma ou uma sequência de transições dispara é necessário processar a associação de OUs, para isso, utiliza-se o operador max. O efeito do disparo consome os tokens dos lugares de entrada de cada transição envolvida e cria um novo token y_k o qual define o valor verdade da proposição d_k. Este valor verdade é obtido utilizando o operador max cujos argumentos são a multiplicação de cada valor verdade pertencente as proposições d_{j1}, ..., d_{jn} pelo fator de certeza de cada transição t_{i1}, ..., t_{in}, respectivamente. Esta operação é definida pela equação $y_k = Max(x_{j1} * \mu_{i1}, ..., x_{jn} * \mu_{in})$.
- **Regra Tipo 4 + Tipo3**: Modela a combinação da regra do Tipo 4 com a do Tipo 3 $(d_{j1} \vee ... \vee d_{jn} \rightarrow d_{k1} \wedge ... \wedge d_{kn})$. Note que, serão criados tokens em cada lugar de saída $p_{k1}, ..., p_{kn}$ definindo os valores verdade das proposições d_{k1}, ..., d_{kn} como resultado da operação de disjunção realizada pelo operador max aplicado sobre os valores verdade das proposições dos lugares de entrada multiplicados pelo fator de certeza de cada transição associada.

Nesta versão de RPF, os operadores fuzzy de máximo, mínimo e produto algébrico são utilizados de forma padrão não existindo a possibilidade de substituí-los por outras extensões de T-normas e T-conormas. Como os tipos de regras são determinados por uma configuração específica da rede, as redes acabam se tornando mais complexas. Além disso, uma proposta inicial de tradução para este tipo de rede foi realizada e apresentada no artigo (KRÜGER. V. et al., 2019). Esta tradução, mostrou-se complexa devido a necessidade de combinar diversas transições em uma única regra da gramática resultando em regras complexas e difíceis de serem analisadas.

5.2 Redes de Petri Fuzzy Generalizadas

Dado que existem diversas classes de RPF, neste trabalho, o foco é dado na definição de um tipo de rede chamado de **Rede de Petri Fuzzy Generalizada** (RPFG) que utiliza conjuntos de operadores fuzzy associados as transições para representar as operações que estão sendo utilizadas. Dessa forma, as agregações podem ser expressadas na rede em forma de classes de operadores pertencentes as T-normas e as T-conormas (SURAJ, 2012). Nesta versão de RPF, além de se ter um modelo genérico, visto que se pode escolher diferentes tipos de agregadores, a forma de representar os diferentes tipos de regras fuzzy é bem mais simples. Por esta razão, a RPFG foi escolhida para realizar a tradução para a GGA, pois, permite obter regras com menor grau de complexidade.

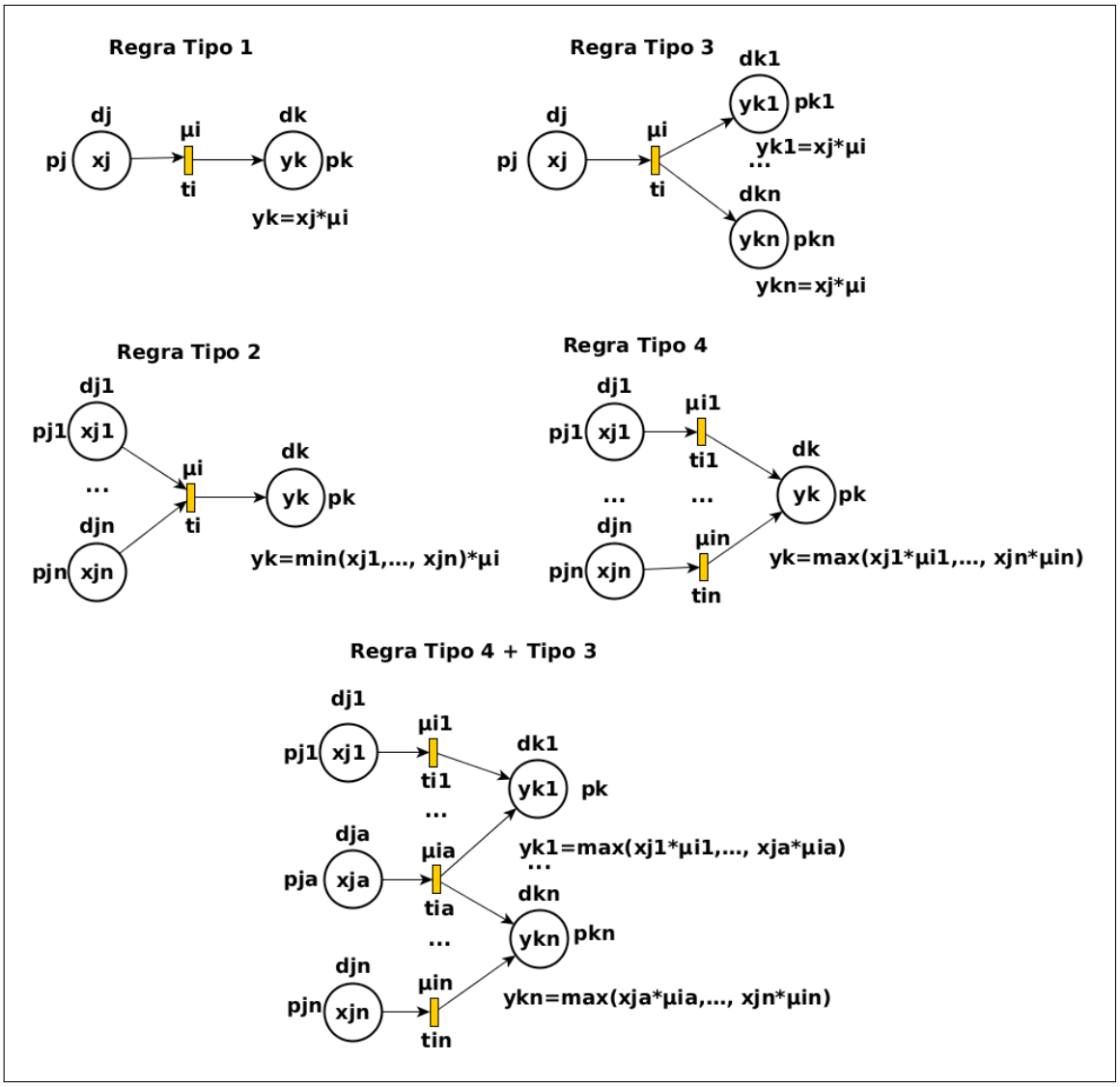

Figura 12 – Tipos de regras fuzzy modeladas por uma RPF (CHEN; KE; CHANG, 1990)

As RPFGs possuem os mesmos componentes das RPFs convencionais, isto é, são formadas de lugares, proposições, tokens, transições e marcação inicial. Porém, ela insere na definição formal das RPFs um conjunto de operadores fuzzy e uma função a qual permite vincular a cada transição uma tripla destes operadores.

Definição 15 ***(Rede de Petri Fuzzy Generalizada):*** *Uma* ***RPFG*** *é uma tupla* $N = (P, Tr, D, I, O, \mu, \beta, \lambda, Op, \delta, M_0)$*, onde:*

- $P = \{p_1, ..., p_n\}$ *é o conjunto finito de lugares,* $n > 0$*;*
- $Tr = \{t_1, ..., t_m\}$ *é o conjunto finito de transições,* $m > 0$*;*
- $D = \{d_1, ..., d_n\}$ *é o conjunto finito de proposições com* $P \cap Tr \cap D = \varnothing$*, com cardinalidade* $|P| = |D|$*;*

- $I : Tr \rightarrow 2^P$ *denota a função de entrada que mapeia cada transição em um conjunto de lugares;*
- $O : Tr \rightarrow 2^P$ *denota a função de saída que mapeia cada transição em um conjunto de lugares;*
- $\mu : Tr \rightarrow [0, 1]$ *é a função que mapeia cada transições para um valor real entre 0 e 1 estabelecendo o seu fator de certeza;*
- $\beta : P \rightarrow D$*, é um mapeamento bijetivo entre lugares e proposições;*
- $\lambda : Tr \rightarrow [0, 1]$ *é a função que associa cada transição a um limiar com valor real entre 0 e 1;*
- Op *é um conjunto finito de T-normas e T-conormas denominado de conjunto de operadores;*
- $\delta : Tr \rightarrow Op_{In} \times Op_{Out1} \times Op_{Out2}$ *é a função que associa uma tripla de operadores com cada transição;*
- $M_0 : P \rightarrow [0, 1]$ *é uma função total que determina a marcação inicial da rede.*

Cada lugar $p \in P$ da rede está vinculado a uma proposição fuzzy $d \in D$ de acordo com um mapeamento bijetivo dado pela função β, logo, o conjunto de lugares P e o conjunto de proposições D devem ter a mesma cardinalidade. A função I modela as pré-condições para cada transição e todo lugar $p \in I(t)$ é chamado de lugar de entrada. A função O modela as pós-condições para cada transição e todo lugar $p \in O(t)$ é chamado de lugar de saída. A função $\delta(t)$ vincula cada transição $t \in Tr$ a uma tripla de operadores fuzzy representada por $(Op_{In}, Op_{Out1}, Op_{Out2})$. Estes operadores, devem ser definidos previamente no conjunto de operadores Op. O operador Op_{In} é o operador de entrada e define a agregação das pertinências dos tokens vinculados aos lugares de entrada. Como os operadores lógicos comumente utilizados no antecedente das regras são o E e o OU fuzzy, então utilizam-se as classes T-norma e T-conorma para especificá-los. O operador Op_{Out1} e o operador Op_{Out2} são chamados de operadores de saída e definem a atualização dos valores da próxima marcação depois do disparo de uma transição. O operador Op_{Out1} computa a saída do operador de entrada com o fator de certeza μ, onde, $\mu(t) \in (0, 1]$. Esta operação é geralmente realizada através de um operador da classe T-norma. O operador Op_{Out2} representa a operação de disjunção realizada entre as regras, geralmente utiliza-se um operador da classe T-conorma. A primeira disposição de tokens na rede é denotada por M_0 e pode ser representada por um vetor de números reais (de tamanho n) contidos no intervalo fuzzy $[0, 1]$. Lugares que não possuem tokens tem valor verdade igual a 0. Dessa forma, a pertinência de um token vinculado a um lugar $p \in P$, é denotada

por $M_0(p) \in [0,1]$. A partir da marcação inicial a rede evolui para novas marcações determinadas pelo disparo das transições.

Definição 16 ***(Marcação e Transição Habilitada):*** *Dada uma RPFG N, uma marcação M da rede N é determinada pela função total $M : P \rightarrow [0,1]$. Se $M(p) = y_i$, sendo $y_i \in [0,1]$ e $\beta(p) = d_i$, então o grau de verdade da proposição d_i é y_i em um determinado estado do sistema. Logo, uma transição $t \in Tr$ está* ***habilitada*** *na marcação M se respeitar a seguinte condição:*

$$Op_{In}(M(p_1), ..., M(p_i)) \geq \lambda(t) > 0, \forall p_i \in I(t). \tag{1}$$

A semântica da RPFG é definida pela atualização das marcações através do disparo das transições habilitadas. Uma transição está habilitada se o resultado do operador de entrada Op_{In} é positivo e maior ou igual ao valor limiar definido pela função λ. Os disparos das transições podem consumir ou não os tokens nas pré-condições, de acordo com o modo de disparo que está sendo considerado. A ação do disparo cria ou recria tokens nas pós-condições da transição envolvida. Os disparos também atualizam os valores verdades das proposições, isto é, atualizam as pertinências dos tokens nas pós-condições. Dessa forma, nenhuma alteração é realizada nos demais lugares da rede e transições que não compartilham lugares podem disparar paralelamente.

Como já mencionado, é possível escolher entre dois modos de disparo das transições, onde pode-se ou não consumir os tokens dos lugares de entrada. Desta forma, existem duas opções para a semântica operacional de uma RPFG, as quais são definidas de acordo com dois diferentes modos de disparo de transições. No primeiro modo ao executar o disparo da transição t todos os tokens dos lugares de entrada são consumidos. No segundo modo os tokens dos lugares de entrada são mantidos.

Definição 17 ***(Disparo de transição- Modo 1):*** *Dada uma marcação M de uma RPFG N que habilita a transição t, a marcação M' é obtida, a partir do disparo da transição t em M, como segue:*

$$M'(p) = \begin{cases} 0 & \textit{Se} \quad p \in I(t) \\ Op_{Out2}(Op_{Out1}(Op_{In}(M(p_1), ..., M(p_i)), \mu(t)), M(p)) & \textit{Se} \quad p \in O(t) \\ M(p) & \textit{c.c} \end{cases} \tag{2}$$

onde $p \in P$ é um lugar de N.

Definição 18 ***(Disparo de transição- Modo 2):*** *Dada uma marcação M de uma RPFG N que habilita a transição t, a marcação M' é obtida, a partir do disparo da*

transição t em M, como segue:

$$M'(p) = \begin{cases} Op_{Out2}(Op_{Out1}(Op_{In}(M(p_1), ..., M(p_i)), \mu(t)), M(p)) & \textit{Se} \quad p \in O(t) \\ M(p) & \textit{c.c} \end{cases} \tag{3}$$

onde $p \in P$ é um lugar de N.

De acordo com o Modo 1 a marcação M' é obtida através das condições definidas na Equação 2. A primeira remove todos os tokens das pré-condições. A segunda atualiza as pós-condições da seguinte forma: (i) o operador Op_{In} agrega as pertinências dos tokens das pré-condições, (ii) o operador Op_{Out1} aplica o fator de certeza ao resultado do operador Op_{In}, (iii) o operador Op_{Out2} agrega a saída do operador Op_{Out1} com os valores da marcação corrente dos lugares de saída, dados por $M(p)$. A terceira mantém inalterado os valores dos tokens dos lugares não envolvidos na transição. No Modo 2 a obtenção da marcação M', definida na Equação 3, é quase análoga ao Modo 1. A única diferença é que os tokens são preservados nas pré-condições após o disparo de uma transição disseminando a verificação da verdade das proposições na rede.

Exemplificação 9 ***(Rede de Petri Fuzzy Generalizada):*** *A Figura 13 ilustra a RPFG que modela as seguintes regras fuzzy:*

1. *Se d_1 OU d_2 então d_4 E d_5, com $\mu_1 = 0.7$ e $\lambda_1 = 0.4$*
2. *Se d_2 E d_3 então d_6, com $\mu_2 = 0.8$ e $\lambda_2 = 0.3$*

A rede deste exemplo é composta pelos conjuntos: $P = \{p_1, ..., p_6\}$ de lugares; $Tr = \{t_1, t_2\}$ de transições; $D = \{d_1, ..., d_6\}$ de proposições; $Op = \{min, *, max\}$ de operadores, com min e $*$ representando as T-normas mínimo e produto algébrico e max representando a T-conorma máximo; pela marcação inicial $M_0 = (0.6, 0.4, 0.7, 0, 0, 0)$; e pelas funções: $\beta(p_1) = d_1$, $\beta(p_2) = d_2$, $\beta(p_3) = d_3$, $\beta(p_4) = d_4$, $\beta(p_5) = d_5$, $\beta(p_6) = d_6$, associando cada lugar p_i à proposição d_i; $\mu(t_1) = \mu_1 = 0.7$ e $\mu(t_2) = \mu_2 = 0.8$, determinando o fator de certeza de cada transição; $\lambda(t_1) = \lambda_1 = 0.4$ e $\lambda(t_2) = \lambda_2 = 0.3$, definindo o limiar de cada transição; $\delta(t_1) = \{max, *, max\}$ e $\delta(t_2) = \{min, *, max\}$, associando cada transição a uma tripla de operadores; $I(t_1) = \{p_1, p_2\}$ e $I(t_2) = \{p_2, p_3\}$, determinando os lugares de entrada para cada transição; e, $O(t_1) = \{p_4, p_5\}$ e $O(t_2) = \{p_6\}$, definindo os lugares de saída de cada transição. A Figura 14(a) ilustra o comportamento da rede de acordo com o Modo 1. As transições t_1 e t_2 estão habilitadas na marcação M_0 (Figura 14), pois, tanto para t_1 como para t_2 a condição necessária para ocorrer um disparo está satisfeita, isto é, $max(0.6, 0.4) \geq 0.4 > 0$ e $min(0.4, 0.7) \geq 0.3 > 0$, respectivamente. Se t_1 dispara então uma nova marcação é alcançada, a M_1. Esta é computada consumindo os tokens de p_1 e p_2 e atualizando os tokens nos lugares de saída com

$max(*(max(0.6, 0.4), 0.7), M_0(p_4))$ *para* p_4 *e* $max(*(max(0.6, 0.4), 0.7), M_0(p_5))$ *para* p_5*, o que resulta em* $M_1 = (0, 0, 0.7, 0.42, 0.42, 0)$*. Neste modo, a transição* t_2 *fica desabilitada, já que os tokens presentes em* p_1 *e* p_2 *são removidos. Na Figura 14(b) tem-se exemplificado o comportamento da rede de acordo com o Modo 2. Inicialmente, ambas as transições estão habilitadas em* M_0 *da mesma forma que no Modo 1. Se a transição* t_2 *dispara então a marcação* M_2 *é alcançada atualizando o valor de pertinência do token em* p_6 *com o resultado de* $max(*(min(0.4, 0.7), 0.8), M_0(p_6))$*, o que resulta em* $M_2 = (0.6, 0.4, 0.7, 0, 0, 0.32)$*. Neste modo, os tokens não são removidos das pré-condições, permitindo que o disparo da transição* t_1 *possa ocorrer após o disparo da transição* t_2*.*

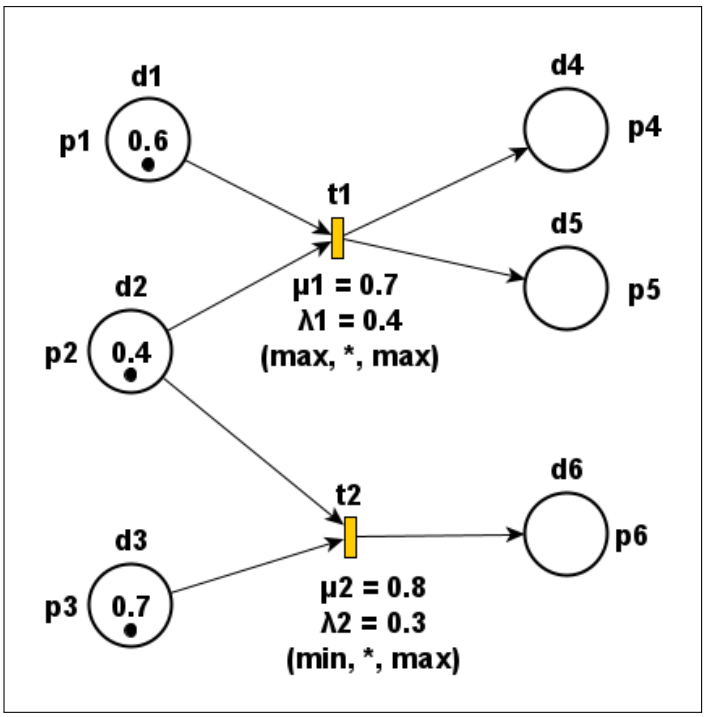

Figura 13 – Exemplo de RPFG (SURAJ, 2012)

As operações fixas definidas na rede desenvolvida por (CHEN; KE; CHANG, 1990) cujos formatos gráficos estão exemplificados na Figura 12, podem ser substituídas sem perda semântica pelos operadores de entrada e saída, Op_{In} e Op_{Out1}, presentes na RPFG. A operação definida pelo operador de saída Op_{Out2} é compatível com a forma que a disjunção é tratada no algoritmo de Raciocínio Fuzzy descrito em (CHEN; KE; CHANG, 1990), o que torna o comportamento da RPFG análogo ao da RPF. A utilização dos operadores na RPFG simplifica a modelagem gráfica modificando o formato básico das regras do Tipo 2 e do Tipo 4 além de prover a escolha do operador fuzzy a ser utilizado.

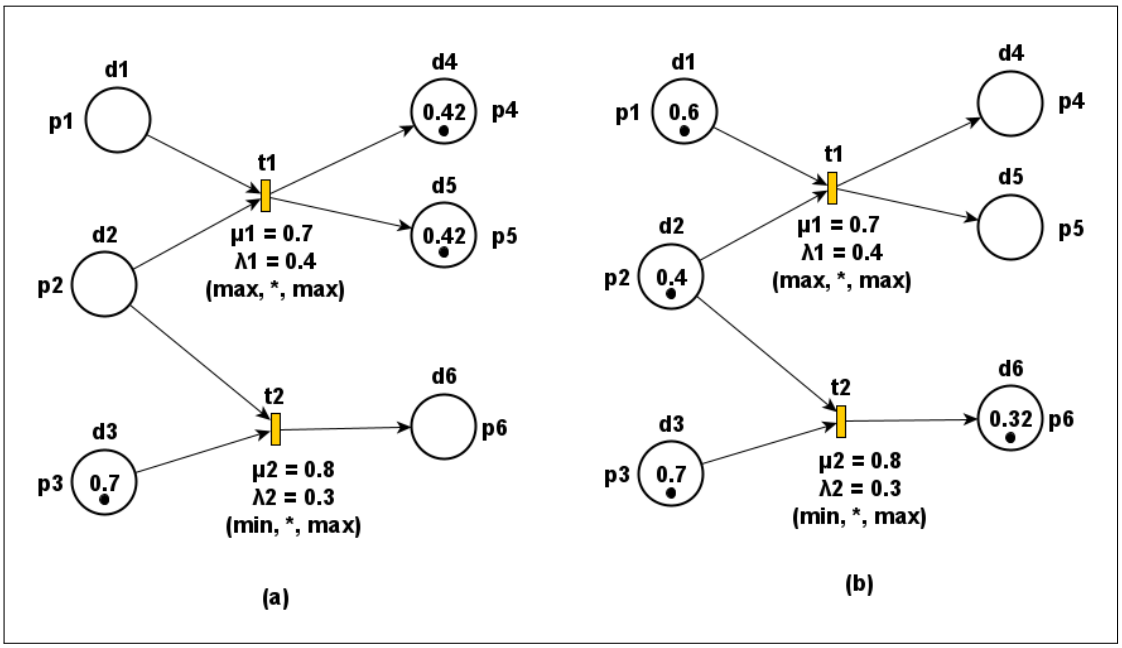

Figura 14 – Exemplo do disparo das transições: (a) marcação depois do disparo de t_1 (conforme Modo 1), (b) marcação depois do disparo de t_2 (conforme Modo 2) (SURAJ, 2012)

6 TRADUÇÃO DA REDE DE PETRI FUZZY GENERALIZADA PARA GRAMÁTICA DE GRAFOS COM ATRIBUTOS

Neste capítulo é apresentado um método para realizar a tradução de uma RPFG para uma GGA. Essa tradução difere-se das já realizadas pelo fato da RPFG possuir elementos e funções da lógica fuzzy. Logo, para obter-se uma GGA a partir de uma RPFG deve-se levar em consideração a tradução dos elementos fuzzy para os elementos que compõem a GGA. A preservação da semântica durante a tradução deve ser considerada, pois, a GGA derivada da tradução deve manter o mesmo padrão de comportamento da RPFG.

Alguns dos conceitos utilizados, assim como, outros tipos de mapeamento podem ser encontrados em (KORFF; RIBEIRO, 1994; CORRADINI; MONTANARI, 1995; CORRADINI, 1996, 1999; SANTOS, 1999; EHRIG; PADBERG, 2003; BALDAN et al., 2010).

Estrutura do Capítulo:

Na Seção 6.1 o método de tradução de uma RPFG para uma GGA é descrito. Na Seção 6.2 será apresentada a prova da preservação do comportamento do modelo traduzido.

6.1 Tradução

A tradução dá-se por meio da relação entre os elementos existentes em ambas as linguagens. Para obter uma GGA a partir de uma RPFG marcada é necessário construir os componentes da GGA_{RPFG} através da estrutura que define a RPFG. Para realizar tal mapeamento, constrói-se a especificação algébrica, o grafo tipo, o grafo inicial e o conjunto de regras a partir dos lugares, dos tokens, das operações fuzzy e das transições que formulam a RPFG.

A especificação algébrica associada à GGA_{RPFG} é a $SPEC_{TFuzzy}$ ilustrada na Figura 15. Nela considera-se a discretização do conjunto dos valores fuzzy referentes ao intervalo unitário $[0, 1]$ de reais, pois apenas racionais deste intervalo (ou os correspondentes em um sistema de ponto flutuante) são representados pelo tipo definido

em $SPEC_{TFuzzy}$. Essa restrição é imposta porque não é possível representar todos os valores contínuos deste intervalo em um sistema computacional. Ao discretizar os valores pertencentes ao intervalo $[0, 1]$ torna-se viável a simulação computacional do modelo e a utilização dos provadores de teoremas. A especificação é definida usando a especificação algébrica $SPEC_{NatBool}$ a qual especifica a álgebra dos números naturais e dos booleanos, esta, também está ilustrada na Figura 15.

Na especificação $SPEC_{TFuzzy}$ constrói-se o tipo de dados abstrato fuzzy, denotado pelo *sort* $[0, 1]$, onde as constantes deste conjunto serão representadas como a divisão de dois números naturais, isto é, pelos números racionais maiores ou iguais a 0 e menores ou iguais a 1. As constantes e operações que podem ser realizadas sobre o tipo abstrato de dados fuzzy possuem as suas assinaturas definidas em **opns** e as suas propriedades definidas em **eqns**. As constantes são o zero e o um os quais indicam os limites do intervalo. A constante $undef$ especifica um valor indefinido. As operações com R subscrito definem a forma como a comparação entre valores pode ser realizada. A operação que possibilita realizar a aproximação é a $./.$ a qual modela a divisão de dois números naturais. O operador $. + .$ define a operação de soma entre valores do intervalo.

Além das operações básicas sobre o *sort* $[0, 1]$, as operações associadas às transições de uma RPFG também devem ser incluídas na especificação algébrica $TFuzzy$. Assim, a especificação algébrica associada à GGA_{RPFG} é obtida adicionando-se, à especificação $SPEC_{TFuzzy}$, a definição das operações fuzzy utilizadas na rede. Dessa forma, o conjunto **opns** deve ser estendido com a assinatura dos operadores definidos no conjunto Op da rede e o conjunto **eqns** deve ser estendido com as propriedades destes operadores.

O grafo tipo estabelece as restrições para os elementos gráficos e para os tipos de valores dos atributos utilizados nos grafos instancias. Para construí-lo, tem-se que levar em consideração que todos os grafos da gramática terão apenas vértices que representem os tokens da rede. Além disso, cada tipo de token é definido pelo lugar da rede o qual se encontra, pois, cada lugar pode conter apenas um único token. Deste modo, o conjunto de vértices do grafo tipo será exatamente o conjunto de lugares da rede, definindo todos os tipos de tokens possíveis na rede. Na gramática GGA_{RPFG} tem-se apenas grafos discretos, ou seja, grafos que não possuem arestas. Logo, para o grafo tipo os conjuntos de arestas, origem e destino são definidos pelo conjunto vazio. O conjunto de atributos é formado pelo conjunto de proposições. Cada vértice está associado com seu respectivo atributo representando o mapeamento bijetivo entre lugares e proposições. A álgebra associada ao grafo tipo é uma álgebra final da especificação algébrica $SPEC_{TFuzzy}$, onde, o tipo abstrato de dados dos atributos é o intervalo $[0, 1]$.

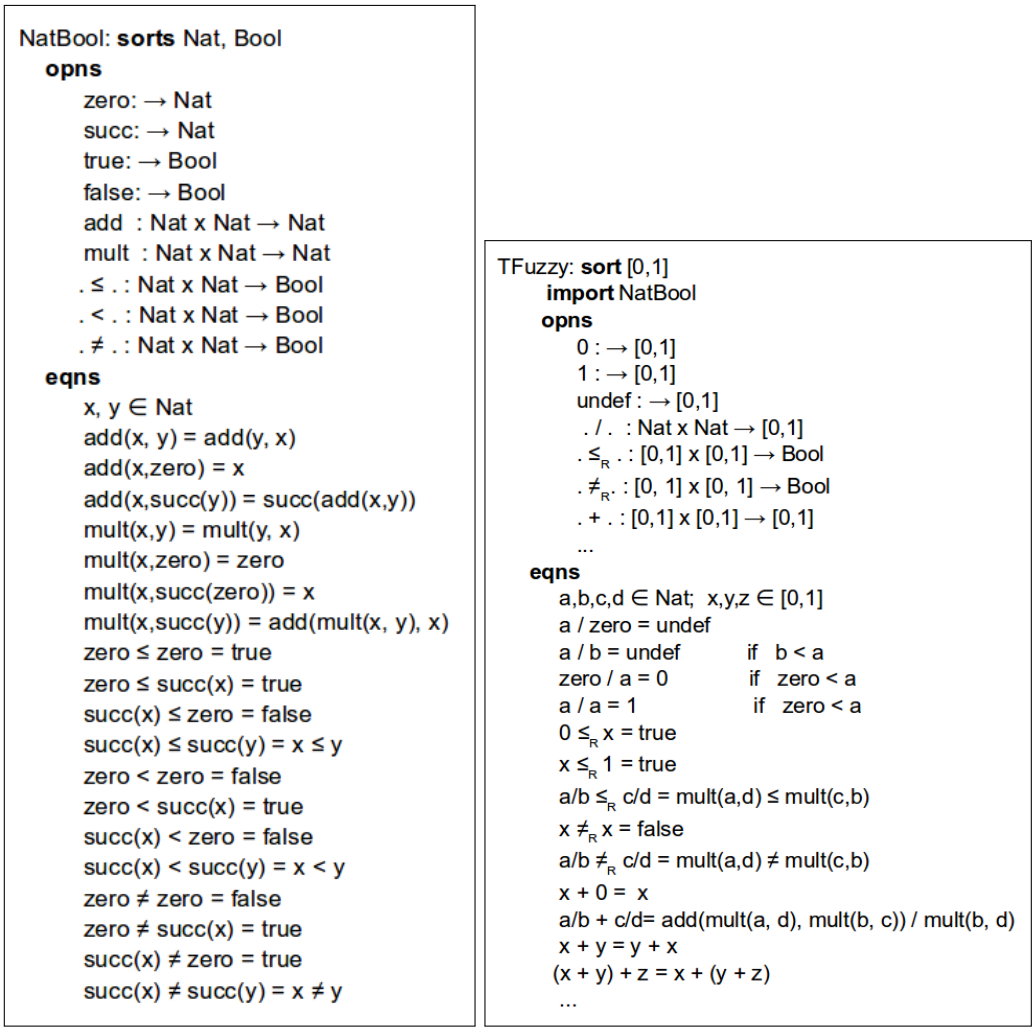

Figura 15 – Especificação algébrica naturais e booleanos $SPEC_{NatBool}$ e especificação algébrica fuzzy $SPEC_{TFuzzy}$

Definição 19 ***(Tradução de uma RPFG em um Grafo Tipo):*** *Dada uma* ***RPFG*** $N = (P, Tr, D, I, O, \mu, \beta, \lambda, Op, \delta, M_0)$*, a tradução de N para um grafo tipo com atributos, denotada por* $Trad_T(N)$*, é definida pelo grafo* $T = (V_T, \varnothing, \varnothing, \varnothing, A_T, Attr_T, val_T, attrv_T)$ *o qual é obtido como segue:*

- $V_T = P$
- A_T *é a álgebra final da especificação algébrica TFuzzy estendida com as operações de* Op
- $Attr_T = D$
- $\forall d \in Attr_T \cdot val_T(d) = [0, 1]$
- $\forall d \in Attr_T \cdot attrv_T(d) = p$*, com* $\beta(p) = d$

Um grafo com atributos é obtido a partir de uma RPFG através da função de tradução $Trad_G$ cujos argumentos são a marcação atual da rede e a própria rede. Esta

função, constrói o grafo com atributos da seguinte forma: o conjunto de vértices do grafo é formado pelos lugares da rede; o conjunto de arestas é vazio, dado que, lugares e tokens não se relacionam entre si; os atributos vinculados aos vértices são obtidos através das proposições; a associação entre vértices e atributos dá-se através do mapeamento bijetivo que ocorre entre lugares e proposições; os valores vinculados aos atributos são exatamente os valores de pertinência associados aos tokens através da marcação; a álgebra associada ao grafo é a álgebra inicial da especificação algébrica $SPEC_{TFuzzy}$, incluindo as operações associadas às transições da rede dada.

Definição 20 ***(Tradução de uma marcação da RPFG para um Grafo Tipado com Atributos):*** *Dada uma* ***RPFG*** $N = (P, Tr, D, I, O, \mu, \beta, \lambda, Op, \delta, M)$ *e uma marcação* M *de* N*, a tradução de* M *em um* ***grafo tipado com atributos*** G*, denotada por* $Trad_G(M, N)$*, é definida pelo grafo* $G = ((V_G, \varnothing, \varnothing, \varnothing, A_G, Attr_G, val_G, attrv_G), t_G, T)$ *o qual é obtido como segue:*

- $V_G = P$
- A_G *é a álgebra inicial da especificação algébrica TFuzzy estendida com as operações de* Op
- $Attr_G = D$
- $\forall d \in Attr_G \cdot val_G(d) = M(p)$*, com* $\beta(p) = d$
- $\forall d \in Attr_G \cdot attrv_G(d) = p$*, com* $\beta(p) = d$
- $t_G = (t_{V_G}, \varnothing, t_{Alg_G}, t_{A_G})$*, onde:*
  - $\forall p \in V_G \cdot t_{V_G}(p) = p$
  - $\forall d \in Attr_G \cdot t_{A_G}(d) = d$
  - $\forall v \in \mathcal{U}(A_G) \cdot t_{Alg_G}(v) = [0, 1]$
- $T = Trad_T(N)$

Para construir o conjunto de regras, cada transição da rede é transformada em uma regra da gramática. A abordagem para gramática de grafos utilizada neste trabalho é a DPO. Assim, de forma geral, o grafo do LHS, o grafo de interface e o grafo do RHS de uma regra ρ_t são obtidos a partir de uma transição t, considerando seus lugares de entrada e saída e suas proposições associadas. Já as equações que devem ser satisfeitas pela ocorrência da regra em um grafo estado são obtidas a partir das funções do limiar, fator de certeza e operações fuzzy vinculadas à transição.

Como no Capítulo 5 foram apresentados dois modos que descrevem a dinâmica das marcações de uma RPFG, a definição da construção do conjunto de regras levará em conta esses dois modos. Dessa forma, são obtidos dois tipos distintos de conjuntos de regras para a gramática gerada de acordo com os modos de disparo.

A Definição 21 apresenta a tradução de uma RPFG em um conjunto de regras de transformação de grafos conforme o modo de disparo escolhido. A ideia aqui é que o LHS das regras contenham um vértice para cada pré e pós-condições da transição associada. Todos os vértices do LHS são deletados e criados novamente pela aplicação da regra. Para o conjunto de regras que descreve o Modo 1 (onde os tokens de entrada são consumidos), os valores fuzzy associados aos atributos dos vértices da pré-condição são zerados no RHS. Já para o conjunto de regras que descreve o Modo 2 (onde os tokens de entrada são mantidos), os valores fuzzy associados aos atributos das pré-condições são mantidos inalterados no RHS.

Definição 21 ***(Tradução de uma RPFG em um Conjunto de Regras):*** *Dada uma* ***RPFG*** $N = (P, Tr, D, I, O, \mu, \beta, \lambda, Op, \delta, M_0)$ *com (ou sem) preservação dos tokens, a tradução de* N *em um* ***conjunto de regras*** *de transformação de grafos, denotada por* $Trad_R(N)$*, é definido pelo conjunto de regras obtido como segue:*

$\forall t \in Tr \cdot \rho_t \in Regras$*, onde,* $\rho_t = (\alpha, X, rEqns)$*, com:*

- $\alpha = L \overset{\alpha L}{\hookleftarrow} K \overset{\alpha R}{\rightarrowtail} R$
  - A *é a álgebra de termos para a especificação algébrica TFuzzy, estruturada com as operações em* $\delta(t)$ *com variáveis em* X
  - $T = Trad_T(N)$
  - $L = ((V_L, \varnothing, \varnothing, \varnothing, A, Attr_L, val_L, attrv_L), t_L, T)$
    - $V_L = I(t) \cup O(t)$
    - $Attr_L = \{\beta(p) | p \in I(t) \vee p \in O(t)\}$
    - $\forall p \in I(t) \wedge \beta(p) = d \cdot val_L(d) = x_d$
    - $\forall p \in O(t) \wedge \beta(p) = d \cdot val_L(d) = y_d$
    - $\forall d \in Attr_L \wedge \beta(p) = d \cdot attrv_L(d) = p$
    - $t_L = (t_{V_L}, \varnothing, t_{Alg_L}, t_{A_L})$
      - $\forall p \in V_L \cdot t_{V_L}(p) = p$
      - $\forall d \in Attr_L \cdot t_{A_L}(d) = d$
      - $\forall v \in \mathcal{U}(A) \cdot t_{Alg_L}(v) = [0, 1]$
  - $K = ((\varnothing, \varnothing, \varnothing, \varnothing, A, \varnothing, \varnothing, \varnothing), t_K, T)$
    - $t_K = (\varnothing, \varnothing, t_{Alg_K}, \varnothing)$
      - $\forall v \in \mathcal{U}(A) \cdot t_{Alg_K}(v) = [0, 1]$

- $R = ((V_R, \varnothing, \varnothing, \varnothing, A, Attr_R, val_R, attrv_R), t_R, T)$
  - $V_R = I(t) \cup O(t)$
  - $Attr_R = \{\beta(p) | p \in I(t) \vee p \in O(t)\}$
  - $\forall p \in I(t) \wedge \beta(p) = d \cdot val_R(d) = x'_d$
  - $\forall p \in O(t) \wedge \beta(p) = d \cdot val_R(d) = y'_d$
  - $\forall d \in Attr_R \wedge \beta(p) = d \cdot attrv_R(d) = p$
  - $t_R = (t_{V_R}, \varnothing, t_{Alg_R}, t_{A_R})$
    - $\forall p \in V_R \cdot t_{V_R}(p) = p$
    - $\forall d \in Attr_R \cdot t_{A_R}(d) = d$
    - $\forall v \in \mathcal{U}(A) \cdot t_{Alg_R}(v) = [0, 1]$
- $\alpha L : K \rightarrow L = (\varnothing, \varnothing, id_A, \varnothing)$, *onde* $id_A : A \rightarrow A$ *é a função identidade*
- $\alpha R : K \rightarrow R = (\varnothing, \varnothing, id_A, \varnothing)$, *onde* $id_A : A \rightarrow A$ *é a função identidade*

- $X = X_L^I \cup X_L^O \cup X_R^I \cup X_R^O$, *onde,*
  $X_L^I = \{x_d | p \in I(t) \wedge \beta(p) = d\}$
  $X_L^O = \{y_d | p \in O(t) \wedge \beta(p) = d\}$
  $X_R^I = \{x'_d | p \in I(t) \wedge \beta(p) = d\}$
  $X_R^O = \{y'_d | p \in O(t) \wedge \beta(p) = d\}$

- *Para RPFG sem preservação de tokens (Modo 1):*
  Se $\delta(t) = (Op_{In}, Op_{Out1}, Op_{Out2})$ *Então*
  $rEqns = \{\lambda(t) \leq Op_{In}(x_{d_1}, ..., x_{d_n}) = true | x_{d_i} \in X_L^I\} \cup$
  $\{y'_d = Op_{Out2}(Op_{Out1}(Op_{In}(x_{d_1}, ..., x_{d_n}), \mu(t)), y_d) | y'_d \in X_R^O, y_d \in X_L^O, x_{d_i} \in X_L^I\} \cup$
  $\{x'_d = 0 | x'_d \in X_R^I\}$

- *Para RPFG com preservação de tokens (Modo 2):*
  Se $\delta(t) = (Op_{In}, Op_{Out1}, Op_{Out2})$ *Então*
  $rEqns = \{\lambda(t) \leq Op_{In}(x_{d_1}, ..., x_{d_n}) = true | x_{d_i} \in X_L^I\} \cup$
  $\{y'_d = Op_{Out2}(Op_{Out1}(Op_{In}(x_{d_1}, ..., x_{d_n}), \mu(t)), y_d) | y'_d \in X_R^O, y_d \in X_L^O, x_{d_i} \in X_L^I\} \cup$
  $\{x'_d = x_d | x'_d \in X_R^I, x_d \in X_L^I\}$

A tradução do conjunto de transições da rede para o conjunto de regras da gramática pode ser descrita brevemente como segue.

O grafo do LHS é constituído por um conjunto de vértices correspondente aos lugares de entrada e aos lugares de saída da transição, onde, os atributos desses vértices são as proposições associadas a tais lugares. Como valores para os atributos, serão utilizadas apenas variáveis. As variáveis presentes no LHS estão divididas em dois grupos a fim de diferenciar os elementos obtidos das pré-condições daqueles das pós-condições da transição que está sendo mapeada. As variáveis nomeadas como x_d são

usadas como valores dos atributos obtidos a partir das pré-condições, e as nomeadas como y_d, são usadas como valores dos atributos obtidos a partir das pós-condições. O grafo de interface K não possui vértice algum, visto que nenhum elemento deve ser preservado pela aplicação da regra. A álgebra associada é a mesma do LHS. O grafo do RHS também será composto por um conjunto de vértices que correspondem aos lugares de entrada e saída da transição, visto que os elementos são todos deletados e recriados na aplicação da regra. Os atributos também correspondem às proposições associadas aos lugares e todos terão uma variável como valor. Os valores que deverão ser atualizados na aplicação da regra são todos dados pelas suas equações. As variáveis x'_d e y'_d definem os novos valores para os atributos que tinham os valores x_d e y_d, respectivamente, no LHS. Os componentes dos vértices, arestas e atributos dos morfismos do span da regra são funções vazias, pois, nada será preservado pela sua aplicação. O conjunto de variáveis X possui todas as variáveis utilizadas nos grafos para cada regra. Já o conjunto de variáveis da regra conterá duas variáveis (x_d e x'_d) para cada lugar de entrada da transição e duas variáveis (y_d e y'_d) para cada lugar de saída da transição. O conjunto de equações será composto de equações que restringem a aplicação da regra e que definem os valores dos atributos a serem atualizados. As equações que restringem a aplicação correspondem à condição necessária para o disparo da transição na rede, isto é, a satisfação do limiar da transição. As demais equações atualizam os valores dos atributos. Os atributos que correspondem às pré-condições da transição são atualizados conforme o modo de disparo que está sendo considerado. Desta forma, se for o Modo 1, estes atributos serão atualizados com o valor 0 (pois os tokens na rede serão consumidos), caso contrário, estes atributos receberão o mesmo valor do LHS conforme o Modo 2. Por sua vez, os atributos que correspondem às pós-condições da transição terão seus valores atualizados de acordo com as operações definidas pela rede para a correspondente transição.

A marcação inicial da rede é transformada no grafo inicial da gramática utilizando a função $Trad_G$. Como a distribuição dos tokens é representada por um vetor de números reais cujo valor zero indica a ausência de um token, então para construir o grafo inicial $G0$, todos os lugares são transformados em vértices e todas as proposições são transformadas em atributos os quais estão associados aos valores definidos pela função M_0. Dessa forma, representa-se a existência de um token através do valor de pertinência o qual deve ser diferente de zero.

Com base na definição das traduções anteriores, pode-se definir a gramática de grafos com atributos obtida a partir de uma RPFG. Nesta gramática, o grafo tipo é obtido a partir da tradução apresentada na Definição 19; o garfo inicial é obtido usando a tradução caracterizada na Definição 20; e o conjunto de regras é obtido usando a tradução de transições para regras mostrada na Definição 21.

Definição 22 ***(Tradução de RPFG para GGA):*** *Dada uma* ***RPFG*** $N =$

$(P, Tr, D, I, O, f, \beta, \lambda, Op, \delta, M_0)$ *a* ***GGA*** *equivalente a* N *é definida pela gramática* $GGA_{RPFG} = (Trad_T(N), Trad_G(M_0, N), Trad_R(N))$.

Exemplificação 10 ***(Tradução de RPFG para GGA):*** *Neste exemplo, apresenta-se a tradução da RPFG, ilustrada na Figura 16. Essa rede modela a base de produções fuzzy mostrada no Exemplo 2, onde cada transição* $t_i \in Tr$ *descreve uma produção fuzzy* i*. Aqui, será vinculado um limiar a cada produção fuzzy* i*. Para* R_1 *tem-se o*

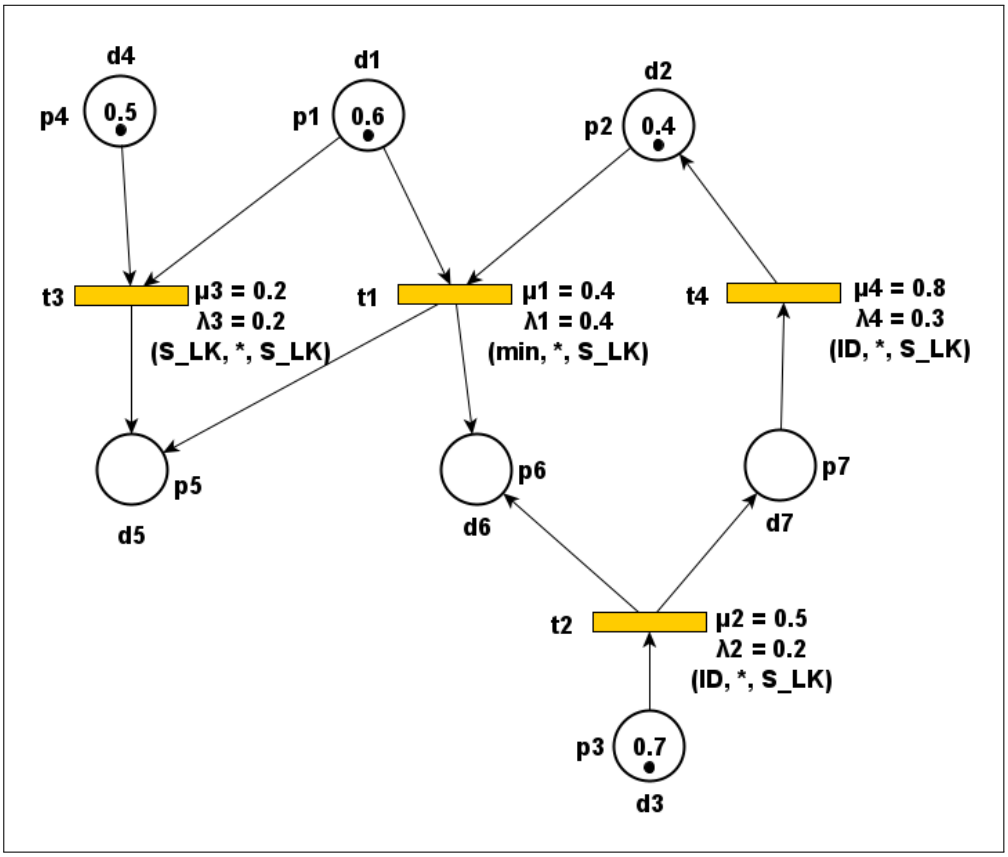

Figura 16 – Exemplo de RPFG a ser traduzida

$\lambda_1 = 0.4$*, para* R_2 *tem-se o* $\lambda_2 = 0.2$*, para* R_3 *tem-se o* $\lambda_3 = 0.2$ *e para* R_4 *tem-se o* $\lambda_4 = 0.3$*. A RPFG foi traduzida para as GGAs ilustradas nas Figuras 18 e 19, considerando os modos 1 e 2 de disparo das transições, respectivamente. Note que, o grafo tipo, a especificação algébrica e o grafo inicial são os mesmos diferindo apenas na obtenção do conjunto de regras das gramáticas. A especificação algébrica que define os tipos abstratos de dados destas gramáticas é mostrada na Figura 17. Esta especificação é obtida estendendo-se* $SPEC_{TFuzzy}$*, adicionando a especificação dos operadores fuzzy de* Op*, isto é,* ID*,* min*,* $*$ *e* S_LK*. O grafo tipo* T *das gramáticas contêm sete vértices, onde cada vértice representa um lugar da rede cujo atributo corresponde à uma proposição associada ao respectivo lugar. Como a álgebra do grafo tipo é a final, os valores dos atributos definem o tipo dos atributos da gramática. Desta forma, os atributos que representam os valores fuzzy terão o tipo definido pelo sort*

$[0, 1]$. *O grafo inicial* $G0$*, também possui sete vértices correspondendo aos lugares da rede, porém, agora os valores associados aos atributos são constantes do sort* $[0, 1]$ *obtidos a partir da marcação inicial* M_0*. As regras foram ilustradas omitindo-se o grafo* K *por questões de simplificação. Em ambos os modos eles são vazios, visto que os elementos referenciados nas regras são todos deletados e recriados por elas. Cada regra* r_i *modela uma transição* $t_i \in Tr$ *cujos grafos* L_i *e* R_i *contém vértices que representam as pré e pós-condições da transição* t_i $(I(t_i) \cup O(t_i))$*. O sub-escrito das operações sobre o sort* $[0, 1]$ *foram omitidos nas equações a fim de simplificar a notação. Na Figura 18 é ilustrada a tradução do conjunto de regras considerando que os tokens das pré-condições não são preservados com o disparo da transição. As equações de cada regra descrevem as condições de disparo e a atualização dos tokens nos lugares de entrada e saída da transição. Por exemplo, a regra* r_1 *que corresponde a transição* t_1 *tem cinco equações. A primeira equação modela a restrição do limiar no disparo da transição. Deste modo, a regra só poderá ser aplicada quando o mínimo entre os valores de pertinência das proposições* d_1 *e* d_2 *for maior ou igual a 0.4. A segunda equação e a terceira atualizam os valores das proposições* d_5 *e* d_6 *após a aplicação da regra definindo os valores para as variáveis* y'_5 *e* y'_6*. Estas atualizações são dadas pelas mesmas operações realizadas na rede. Já as duas últimas equações atualizam os valores das proposições* d_1 *e* d_2*, zerando seus valores e modelando assim o consumo dos tokens das pé-condições da transição. Na Figura 19 é ilustrada a tradução do conjunto de regras considerando que os tokens das pré-condições são preservados. O que difere estas regras daquelas apresentadas na Figura 18 são as equações que atualizam os valores das pré-condições da transição. Por exemplo, agora os valores das proposições* d_1 *e* d_2 *permanecem iguais após a aplicação da regra* r_1*, como definido pelas últimas duas equações da regra.*

6.2 Análise da Preservação Semântica

A semântica determina a dinâmica de execução de cada modelo, isto é, a forma em que ocorrem as mudanças entre os estados alcançáveis de um sistema (ROZENBERG, 1997).

Em uma RPFG os estados possíveis de um sistema são modelados pelas marcações da rede e as mudanças entre estes estados são descritas pelas transições. Uma mudança de estado só pode ocorrer quando o disparo de alguma transição estiver habilitado. Assim, dada a marcação corrente da rede, se uma transição é disparada, uma nova marcação é alcançada (SURAJ, 2012).

Em uma GGA os estados de um sistema são modelados por grafos e as mudanças entre estes estados ocorre por meio da aplicação de regras de transformação de grafos. Quando tem-se uma ocorrência do LHS em um grafo estado que satisfaz todas as

```
TFuzzy: sort [0,1]
    import NatBool
    opns
        0 : → [0,1]
        1 : → [0,1]
        undef : → [0,1]
        . / . : Nat x Nat → [0,1]
        . ≤R . : [0,1] x [0,1] → Bool
        . ≠R . : [0, 1] x [0, 1] → Bool
        . + . : [0,1] x [0,1] → [0,1]
        min: [0,1] x [0,1] → [0,1]
        . * . : [0,1] x [0,1] → [0,1]
        S_LK: [0,1] x [0,1] → [0,1]
        ID : [0,1] → [0,1]
    eqns
        a,b,c,d ∈ Nat; x,y,z ∈ [0,1]
        a / zero = undef
        a / b = undef          if  b < a
        zero / a = 0           if  zero < a
        a / a = 1              if  zero < a
        0 ≤R x = true
        x ≤R 1 = true
        a/b ≤R c/d = mult(a,d) ≤ mult(c,b)
        x ≠R x = false
        a/b ≠R c/d = mult(a,d) ≠ mult(c,b)
        x + 0 = x
        a/b + c/d= add(mult(a, d), mult(b, c)) / mult(b, d)
        x + y = y + x
        (x + y) + z = x + (y + z)
        min(x, 0) = 0
        min(x, 1) = x
        min(x, y)= x                if  x ≤R y
        min(x, y)= y                if  y ≤R x
        min(x, y) = min(y, x)
        min(min(x, y), z) = min(x, min(y, z))
        min(x, y) ≤R min(y, z) = true     if x ≤R z
        x * 0 = 0
        x * 1 = x
        a/b * c/d= mult(a,c)/ mult(b,d)
        x * y = y * x
        (x * y) * z = x *(y * z)
        x* y ≤R  y * z = true     if x ≤R z
        S_LK(x, 0) = x
        S_LK(x, 1) = 1
        S_LK(x, y) =min( x+y, 1)
        S_LK(x, y) = S_LK(y, x)
        S_LK(S_LK(x, y), z) = S_LK(x, S_LK(y, z))
        S_LK(x, y) ≤R S_LK(y, z) = true     if x ≤R z
        ID(x)= x
```

Figura 17 – Especificação Algébrica $SPEC_{TFuzzy}$ para a tradução da RPFG da Figura 16

equações da regra, então esta regra pode ser aplicada resultando em um novo estado do sistema (CAVALHEIRO; FOSS; RIBEIRO, 2017).

Para que a tradução de um modelo para o outro não altere o comportamento do

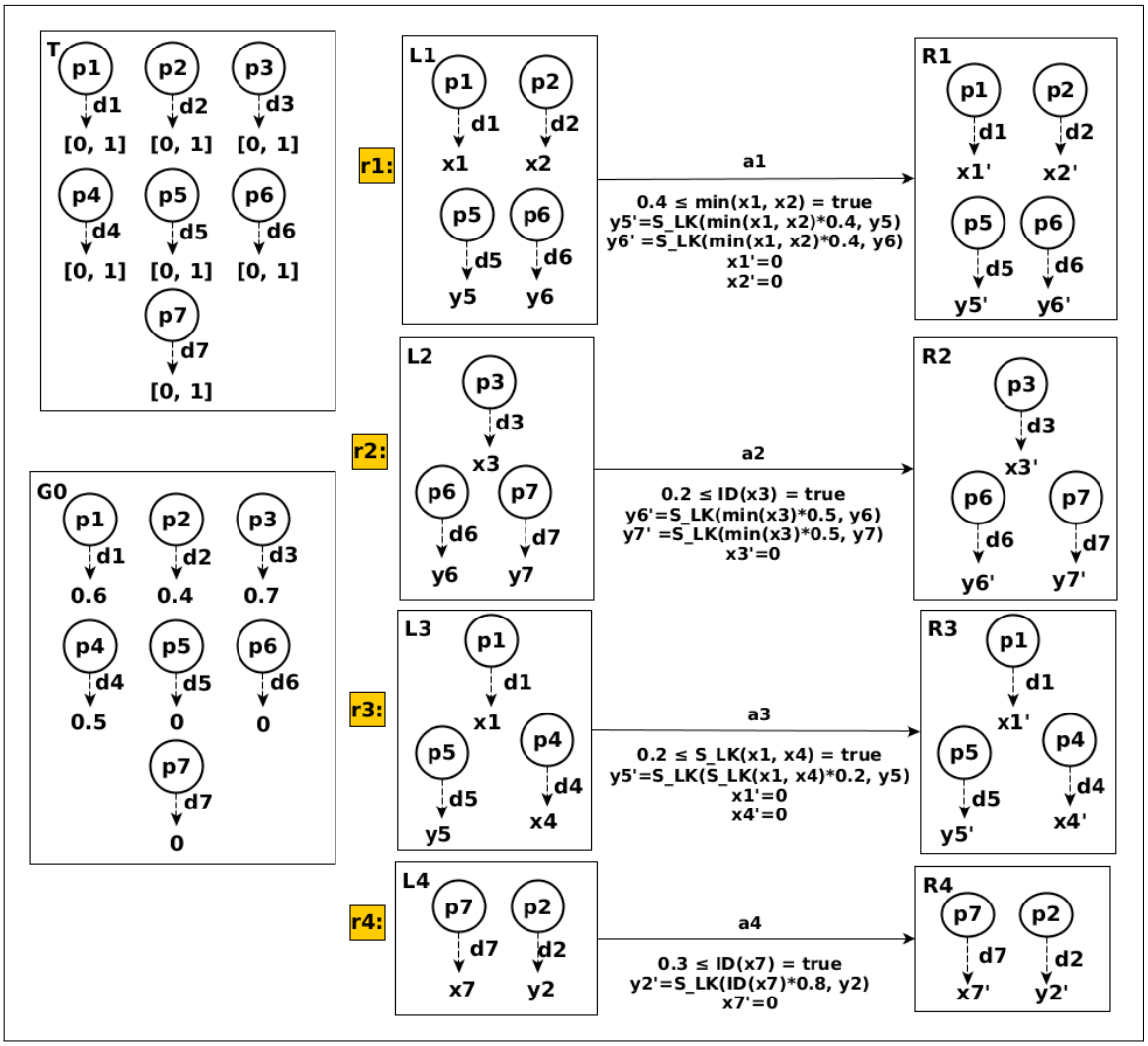

Figura 18 – GGA_{RPFG} obtida através da tradução da RPFG da Figura 16, considerando o Modo 1 de disparo das transições.

sistema modelado por uma RPFG, deve-se garantir que a semântica das RPFGs seja preservada pela tradução. Ou seja, dada uma RPFG que modela um sistema fuzzy e a GGA obtida pela tradução, considerando um mesmo estado em ambos os modelos, se na RPFG há uma transição habilitada neste estado cujo disparo gera uma nova marcação, então a regra da GGA correspondente à essa transição, também está habilitada no mesmo estado, e sua aplicação gera um grafo que corresponde à marcação obtida pelo disparo da transição na RPFG. Isso significa que todo o comportamento da RPFG é reproduzido pela GGA. A relação inversa também deve ser válida, garantindo que a tradução não gera nenhum comportamento que não exista na RPFG. Isto se deve, ao fato que a tradução proposta deve preservar a relação entre os disparos da rede e a alcançabilidade entre os estados (BALDAN et al., 2010).

Para demonstrar a preservação da semântica, é necessário comparar uma marcação e um grafo estado para verificar se correspondem ao mesmo estado. Para isso, define-se a função $Trad_M$, que dado um grafo estado retorna uma marcação corres-

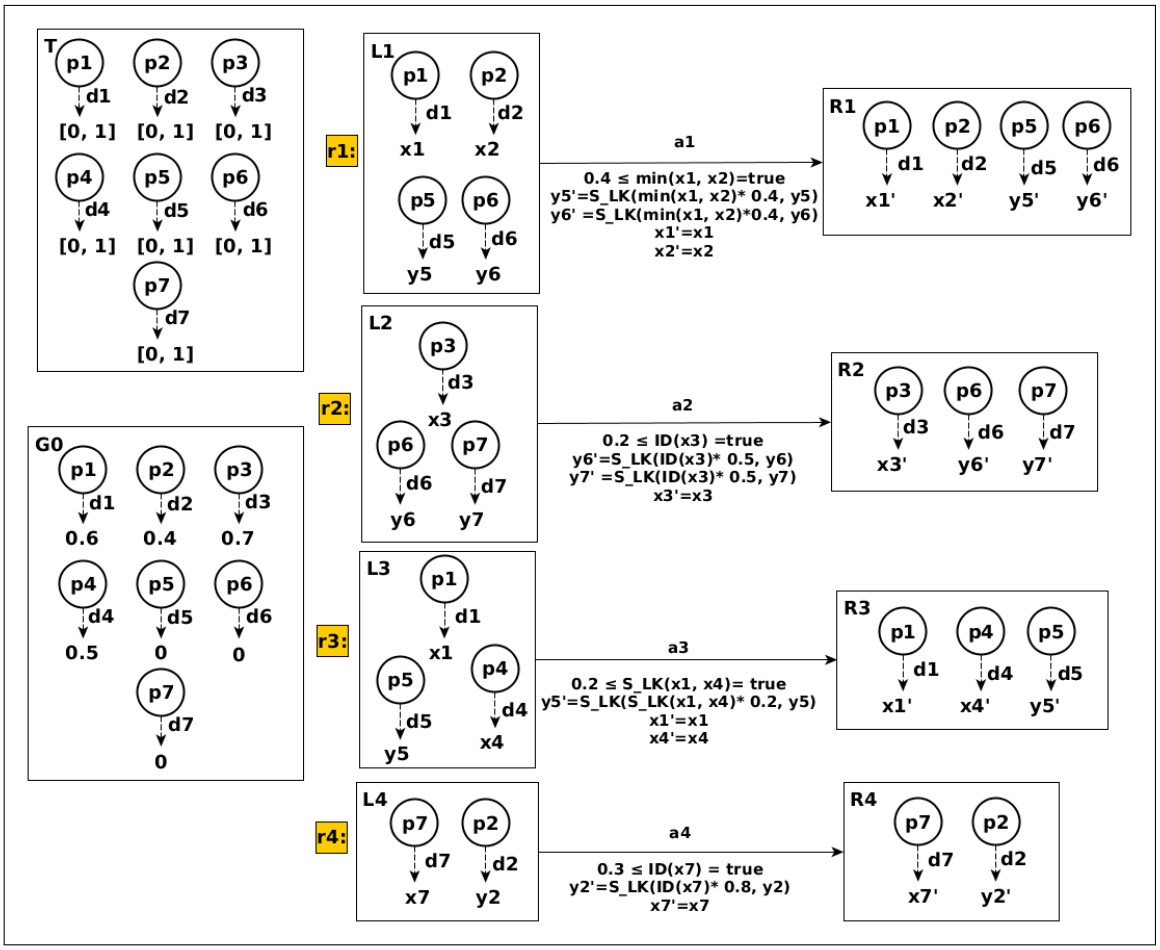

Figura 19 – GGA_{RPFG} obtida através da tradução da RPFG da Figura 16, considerando o Modo 2 de disparo das transições.

pondente. A função $Trad_M$ recebe como argumento um grafo estado e retorna uma marcação da rede. Porém, esta função não está definida para qualquer grafo tipado com atributos. Para que exista uma marcação correspondente na rede, este grafo deve ser tipado sobre o grafo tipo obtido a partir da tradução $Trad_T$. Assim, garante-se que o grafo estado é composto apenas por vértices que correspondam aos lugares da rede. Além disso, o grafo não pode ter duas instâncias do mesmo lugar e deve conter todos os lugares da rede. Para isso, exige-se que o morfismo de tipagem seja um isomorfismo. Já a função $attrv$ deve ser bijetora para garantir que exista exatamente uma proposição associada a cada lugar (lembrando que os atributos dos vértices correspondem às proposições). Com relação à álgebra associada ao grafo, ela deve ser a álgebra inicial (por ser um grafo estado) da especificação algébrica $SPEC_{TFuzzy}$ estendida com as operações da rede.

Definição 23 ***(Tradução de um Grafo Estado para uma Marcação):*** *Dada uma* ***RPFG*** N *e um grafo tipado com atributos* $G = ((V_G, \varnothing, \varnothing, \varnothing, A_G, Attr_G, val_G, attrv_G), t_G, T)$*, onde* $T = Trad_T(N)$*,* A_G *é uma álgebra inicial para* $SPEC_{TFuzzy}$ *e,* t_G *e* $attrv_G$ *são injetores e sobrejetores, pode-se obter uma marcação* M *para* N*, denotada por*

$Trad_M(G)$, *da seguinte forma:*

$$\forall v \in V_G \cdot M(t_{V_G}(v)) = val_G(a), \textit{ com } atttrv_G(a) = v.$$

Para garantir que as traduções $Trad_M$ e $Trad_G$ preservam as marcações, deve-se mostrar que $Trad_M(Trad_G(M, N)) = M$, isto é, traduzir uma marcação para um grafo e este grafo para uma marcação novamente, resultando na marcação original.

Proposição 1 *Dada uma marcação M de N, $Trad_M(Trad_G(M, N)) = M$.*

Prova 1 *Considerando $M(p_i) = v_i$. Pela definição de $Trad_G$, tem-se que, $Trad_G(M, N) = ((V_G, \varnothing, \varnothing, \varnothing, A_G, Attr_G, val_G, attrv_G), t_G, T)$, com:*

- $T = Trad_T(N)$
- $\forall d_i \in Attr_G \cdot val_G(d_i) = v_i$, *com* $\beta(p_i) = d_i$
- $\forall d_i \in Attr_G \cdot attrv_G(d_i) = p_i$, *com* $\beta(p_i) = d_i$
- $t_G = (t_{V_G}, \varnothing, t_{Alg_G}, t_{A_G})$
  $\forall p_i \in V_G \cdot t_{V_G}(p_i) = p_i$

Pela definição de $Trad_M$ (Definição 23), tem-se que, $Trad_M(((V_G, \varnothing, \varnothing, \varnothing, A_G, Attr_G, val_G, attrv_G), t_G, T)) = M$, onde, $\forall p_i \in V_G \cdot M(t_{V_G}(p_i)) = val_G(d_i)$, com $atttrv_G(d_i) = p_i$. Resultando em $M(p_i) = v_i$, como queria-se provar.

Independente da tradução realizada conservar itens ou não, para cada transição habilitada por uma marcação da rede existe uma produção aplicável correspondente em um grafo estado da gramática, isto é, a mesma sequência de disparos da rede deve ser reproduzida pelas derivações da gramática. Com isso, a proposição a seguir demonstra que todo o comportamento da rede também está presente na gramática.

Proposição 2 *Dada uma **RPFG** $N = (P, Tr, D, I, O, \mu, \beta, \lambda, Op, \delta, M_0)$. Se $t \in Tr$ está habilitada em uma marcação M de N, que deriva M' considerando os modos de disparo das transições (Modo 1 e Modo 2), então existe $G \overset{r,m}{\Rightarrow} H$, onde $G = Trad_G(M, N)$, $r = p_t \in Trad_R(N)$ e $Trad_M(H) = M'$.*

Prova 2 *Esta demonstração é dada como segue:*

1. *Existe $m : \overline{L_r} \rightarrow G = (m_V, \varnothing, m_{Alg}, m_A)$ que satisfaz todas as equações em $rEqns$.*

   Pelas definições de $Trad_G$ e $Trad_R$ (Definição 20 e Definição 21, respectivamente), sabe-se que G contém um vértice para cada lugar da rede e L_r contém um vértice para cada lugar de entrada e saída da transição t. Assim, é possível definir m mapeando cada vértice de L_r no vértice de G que corresponde ao mesmo lugar, ou seja, pode-se definir m como segue:

- $m_V : V_{L_r} \to V_G$

  $v \in V_{L_r} \cdot m_V(v) = v'$, *onde* $t_{L_r}(v) = t_G(v')$ *(como* $t_{V_{L_r}}$ *e* t_{V_G} *são injetoras,* m_V *é univocamente determinada pelos tipos de vértices de* L_r *e* G*).*

- $m_A : Attr_{L_r} \to Attr_G$

  $a \in Attr_{L_r} \cdot m_A(a) = a'$, *onde* $t_{A_{L_r}}(a) = t_{A_G}(a')$ *(como* $t_{A_{L_r}}$ *e* t_{A_G} *são injetoras,* m_A *é univocamente determinada pelos tipos dos atributos de* L_r *e* G*).*

- $m_{Alg} : T_{eq}(X) \to \mathcal{U}(A) = \overline{eval}$*, tal que* $\overline{eval}$ *é uma extensão de* $eval : X \to \mathcal{U}(A_G)$ *para a classe de equivalência de termos com variáveis em* X*, onde* $eval$ *é definida como segue:*

  ⋆ *Considerando a não preservação de tokens (Modo 1):*

  $$\forall var \in X \cdot eval(var) = \begin{cases} val_G(m_A(a)) & \text{Se } var \in X_L^I \cup X_L^O, \text{ onde } var = val_{L_r}(a) \\ & \text{(pela injetividade de } val_{L_r}, a \text{ é o único atributo} \\ & \text{com valor igual a } var) \\ 0 & \text{Se } var \in X_R^I \\ v & \text{Se } var = y'_d \in X_R^O, \text{ onde} \\ & v = Op_{Out2}(Op_{Out1}(Op_{In}(eval(x_{d_1}), ..., eval(x_{d_n})), \mu(t)), \\ & eval(y_d)), \text{ com } \delta(t) = (Op_{In}, Op_{Out1}, Op_{Out2}), \\ & x_{d_i} \in X_L^I \text{ e } y_d \in X_L^O \end{cases}$$

  ⋆ *Considerando a preservação de tokens (Modo 2):*

  $$\forall var \in X \cdot eval(var) = \begin{cases} val_G(m_A(a)) & \text{Se } var \in X_L^I \cup X_L^O, \text{ onde } var = val_{L_r}(a) \\ & \text{(pela injetividade de } val_{L_r}, a \text{ é o único atributo} \\ & \text{com valor igual a } var) \\ u & \text{Se } var = x'_d \in X_R^I, \text{ onde } u = eval(x_d), x_d \in X_L^I \\ v & \text{Se } var = y'_d \in X_R^O, \text{ onde} \\ & v = Op_{Out2}(Op_{Out1}(Op_{In}(eval(x_{d_1}), ..., eval(x_{d_n})), \mu(t)), \\ & eval(y_d)), \text{ com } \delta(t) = (Op_{In}, Op_{Out1}, Op_{Out2}), \\ & x_{d_i} \in X_L^I \text{ e } y_d \in X_L^O \end{cases}$$

Além disso, m *satisfaz todas as equações em* $rEqns$*:*

- *Pela definição de* m_{Alg}*, as equações que definem os valores das variáveis são satisfeitos para ambos os modos.*

- *A equação* $lim \leq Op_{In}(x_{d_1}, ..., x_{d_n}) = true$ *é também satisfeita pois:*

  * $lim = \lambda(t)$ *(pela definição de* $Trad_R$*)*

  * $Op_{In}(eval(x_{d_1}), ..., eval(x_{d_n}))$ *(pela definição de* m_{Alg}*)*

  $= Op_{In}(val_G(m_A(a_1)), ..., val_G(m_A(a_n)))$ *(pela definição de* $eval$*)*

$= Op_{In}(M(p_1), ..., M(p_n)))$ *(pela definição de* $Trad_G$*), com* $val_{L_r}(a_i) = x_{d_i}$ *e* $attrv_G(m_A(a_i)) = p_i$

Como t *está habilitada em* M*, então tem-se que*
$\lambda(t) \leq Op_{In}(M(p_1), ..., M(p_n)))$ *(Definição 16), como queria-se provar.*

2. $Trad_M(H) = M'$

Pela definição de $Trad_G$ *tem-se que* $G = ((V_G, \varnothing, \varnothing, \varnothing, A_G, Attr_G, val_G, attrv_G), t_G, T)$*, onde,*

- $V_G = P$
- A_G *é a álgebra inicial da especificação algébrica TFuzzy*
- $Attr_G = D$
- $\forall d \in D \cdot val_G(d) = M(p) \wedge attrv_G(d) = p$*, com* $\beta(p) = d$
- $t_G = (t_{V_G}, \varnothing, t_{Alg_G}, t_{A_G})$*, onde:*
  - $\forall p \in V_G \cdot t_{V_G}(p) = p$
  - $\forall d \in Attr_G \cdot t_{A_G}(d) = d$
  - $\forall v \in \mathcal{U}(A_G) \cdot t_{Alg_G}(v) = [0, 1]$
- $T = Trad_T(N)$

Pela construção do pushout tem-se que:

$H = ((V_H, \varnothing, \varnothing, \varnothing, A_H, Attr_H, val_H, attrv_H), t_H, T)$

- $V_H = (V_G - m_v(V_L)) \uplus V_R$
  V_H *é isomórfico a* V_G*, pois existe um isomorfismo de* V_G *em* V_H *induzido pelas funções de tipagem, visto que* $V_L = V_R$ *(pela definição de* $Trad_R$*).*
- $A_H = A_G$ *pois a derivação preserva a álgebra.*
- $Attr_H = (Attr_G - m_A(Attr_L)) \uplus Attr_R$
  De forma análoga aos vértices, $Attr_H$ *é isomórfico à* $Attr_G$*, pois* $Attr_L = Attr_R$ *pela definição de* $Trad_R$ *(Definição 21).*
- $\forall \in Attr_H.attrv_H(a) = \begin{cases} attrv_G(a) & \textit{Se } a \in Attr_G \\ attrv_R(a) & \textit{Se } a \in Attr_R \end{cases}$
  Como $attrv_L$ *e* $attrv_R$ *estabelecem as mesmas relações entre os atributos e os vértices,* $attrv_G$ *e* $attrv_H$ *também definirão as mesmas relações.*
- *Considerando a não preservação de tokens (Modo 1):*
  $\forall a \in Attr_H \cdot val_H(a) =$
  $$\begin{cases} val_G(a) & \textit{Se } a \in Attr_G \\ 0 & \textit{Se } a \in Attr_R \wedge p \in I(t) \textit{ com } t_{V_R}(attrv_R(a)) = p \\ \overline{eval}(val_R(a)) & \textit{Se } a \in Attr_R \wedge p \in O(t) \textit{ com } t_{V_R}(attrv_R(a)) = p \end{cases}$$

- *Considerando a preservação de tokens (Modo 2):*

  $\forall a \in Attr_H \cdot val_H(a) =$

  $$\begin{cases} val_G(a) & \textbf{Se } a \in Attr_G \\ \overline{eval}(val_R(a)) & \textbf{Se } a \in Attr_R \wedge p \in I(t) \cup O(t) \textbf{ com } t_{V_R}(attrv_R(a)) = p \end{cases}$$

- $t_H = (t_{V_H}, \varnothing, t_{Alg_H}, t_{A_H})$

  $$\forall v \in V_H \cdot t_{V_H}(v) = \begin{cases} t_{V_G}(v) & \textbf{Se } v \in V_G \\ t_{V_R}(v) & \textbf{Se } v \in V_R \end{cases}$$

- $\forall v \in \mathcal{U}(A_H) \cdot t_{Alg}(v) = [0, 1]$

- $$\forall a \in Attr_H \cdot t_{A_H}(a) = \begin{cases} t_{A_G}(a) & \textbf{Se } a \in Attr_G \\ t_{A_R}(a) & \textbf{Se } a \in Attr_R \end{cases}$$

Pode-se observar que H difere de G apenas pelos valores dos atributos associados aos lugares de entrada e saída da transição vinculada à regra aplicada.

Pela definição de $Trad_M$, tem-se que $Trad_M(H) = M''$, onde $\forall p \in V_H \cdot M''(t_{V_H}(p)) = val_H(d)$ com $attrv_H(d) = p$.

Pela definição de val_H tem-se que o valor de $M''(t_{V_H}(p))$ pode ser obtido em três casos:

1. Se $d \in Attr_G$, isto é, $t_{V_H}(v) \notin I(t) \cup O(t)$ então $M''(t_{V_H}(p)) = val_G(d) = M(t_{V_G}(p)) = M(p)$ (pela definição de G).

2. Considerando a não preservação de tokens (Modo 1):
Se $d \in Attr_R \wedge t_{V_H}(p) \in I(t)$ então $M''(t_{V_H}(p)) = 0$.

2. Considerando a preservação de tokens (Modo 2):
Se $d \in Attr_R \wedge t_{V_H}(p) \in I(t)$ então $M''(t_{V_H}(p)) = \overline{eval}(val_R(d)) = val_G(m_A(a))$, com $m_A(a) = d$. Pela definição de G, $val_G(d) = M(p)$ portanto $M''(t_{V_H}(p)) = M(p)$.

3. Se $d \in Attr_R \wedge t_{V_H}(p) \in O(t)$ então $M''(t_{V_H})(p) = \overline{eval}(val_R(d)) = Op_{Out2}(Op_{Out1}(Op_{In}(val_G(m_A(a_1)), ..., val_G(m_A(a_n))), \mu(t)), val_G(m_A(a')))$ com $m_A(a_i) = d_i$, $m_A(a') = d$, $attrv_G(d_i) = p_i$, $t_{V_G}(p_i) \in I(t)$ (pela definição de $Trad_R$ e $eval$). Pela definição de G, $val_G(d_i) = M(p_i)$ e $val_G(d) = M(p)$, portanto, $M''(t_{V_H}(p)) = Op_{Out2}(Op_{Out1}(Op_{In}(M(p_1), ..., M(p_n)), \mu(t)), M(p))$.

Pela definição de disparo de transição Modo 1 (Definição 17) tem-se que:

$$\forall p \in P \cdot M'(p) = \begin{cases} M(p) & \textbf{Se } p \notin I(t) \cup O(t) \\ v & \textbf{Se } p \in O(t) \\ 0 & \textbf{Se } p \in I(t) \end{cases}$$

com $v = Op_{Out2}(Op_{Out1}(Op_{In}(M(p_1), ..., M(p_n)), \mu(t)), M(p))$.

Pela definição de disparo de transição Modo 2 (Definição 18) tem-se que:

$$\forall p \in P \cdot M'(p) = \begin{cases} M(p) & \textit{Se } p \notin I(t) \cup O(t) \\ v & \textit{Se } p \in O(t) \\ M(p) & \textit{Se } p \in I(t) \end{cases}$$

com $v = Op_{Out2}(Op_{Out1}(Op_{In}(M(p_1), ..., M(p_n)), \mu(t)), M(p))$.

Sendo assim, pode-se observar que $M'' = M'$ *e consequentemente* $Trad_M(H) = M'' = M'$*, para os dois modos de disparo, como queria-se demonstrar.*

De forma análoga, se uma regra pode ser aplicada na gramática obtida então a transição que derivou tal regra pode disparar na rede. Assim, a proposição a seguir demonstra que a gramática não apresenta comportamentos que não estão presentes na rede.

Proposição 3 *Dada uma* ***GGA*** $GG = (T, G0, Rules)$ *obtida pela tradução de uma* ***RPFG*** $N = (P, Tr, D, I, O, \mu, \beta, \lambda, Op, \delta, M_0)$*, considerando os modos de disparo das transições (Modo 1 e Modo 2). Se existe uma ocorrência* m *para uma regra* $p_t \in Rules$ *em um grafo estado* G *(obtido a partir de G0 pela aplicação de uma ou mais regras de* GG*), tal que* $G \overset{p_t,m}{\Rightarrow} H$*, então a transição* t *está habilitada em* $Trad_M(G)$ *e o disparo de* t *resulta em uma marcação* M'*, tal que* $Trad_G(M', N) = H$*.*

Prova 3 *Esta demonstração é dividida em duas etapas:*

1. *A transição* t *está habilitada em* $Trad_M(G)$ *considerando que* $Trad_M(G) = M$*, pela definição de* $Trad_M$*, tem-se que* $\forall v \in V_G \cdot M(t_V(v)) = val_G(a)$*, com* $attrv_G(a) = v$*.*

   Pela definição de G*, tem-se que* $\forall p \in P \cdot M(p) = val_G(a)$*, onde* $V_G = P$*,* $t_{V_G}(p) = p$*,* $t_{A_G}(a) = d$ *e* $\beta(p) = d$*.*

   Se p_t *está habilitada em* G *pela ocorrência* $m : \overline{L_{p_t}} \to G$*, então existe um mapeamento* $eval : X \to \mathcal{U}(A_G)$*, tal que as equações em* $rEqns$ *são satisfeitas. Por meio deste mapeamento, para cada variável* $x_{d_i} \in X_L^I (y_{d_i} \in X_L^O)$*, tem-se o valor* $M(p_i)$ *associado, onde* $M(p_i)$ *corresponde ao valor da proposição* d_i *vinculada ao lugar* p_i *de entrada (saída) de* t*.*

   Pela definição de $Trad_R$*, existe uma equação* $eq \in rEqns$ *que é satisfeita por* m*, onde* $eq = \lambda(t) \leq Op_{In}(x_{d_1}, ..., x_{d_n}) = true$*, esta equação estabelece que a seguinte condição é verdadeira* $\lambda(t) \leq Op_{In}(M(p_1), ..., M(p_n))$ *visto que* m *associa cada variável* x_{d_i} *ao valor* $M(p_i)$*.*

   Esta condição garante que t *está habilitada em* M *(pela Definição 16).*

2. $Trad_G(M', N) = H$

Pela Definição 17 (Modo 1 de disparo), tem-se que para todo $p \in P$

$$M'(p) = \begin{cases} M(p) & \textit{Se } p \notin I(t) \cup O(t) \\ v & \textit{Se } p \in O(t) \\ 0 & \textit{Se } p \in I(t) \end{cases}$$

com $v = Op_{Out2}(Op_{Out1}(Op_{In}(M(p_1), ..., M(p_n)), \mu(t)), M(p))$.

Pela Definição 18 (Modo 2 de disparo), tem-se que para todo $p \in P$

$$M'(p) = \begin{cases} M(p) & \textit{Se } p \notin I(t) \cup O(t) \\ v & \textit{Se } p \in O(t) \\ M(p) & \textit{Se } p \in I(t) \end{cases}$$

com $v = Op_{Out2}(Op_{Out1}(Op_{In}(M(p_1), ..., M(p_n)), \mu(t)), M(p))$.

Com $Trad_M(G) = M$, H *é obtido pela aplicação de* p_t *em* G *com a ocorrência* m, *o grafo* H *é isomórfico ao grafo* G *na parte dos vértices e atributos, diferenciando-se de* G *apenas nos valores associados aos atributos, isto é,* H *pode ser definido como o grafo* $((V_G, \varnothing, \varnothing, \varnothing, A_G, Attr_G, val_H, attrv_G), t_G, T)$, *onde:*

- *Considerando a não preservação de tokens (Modo 1):*

$\forall a \in Attr_G \cdot val_H(a) =$

$$\begin{cases} val_G(a) & \textit{Se } a \notin rng(m_A) \\ 0 & \textit{Se } m_A(a') = a \wedge val_{L_{p_t}}(a') = x_{d_i} \wedge x_{d_i} \in X_L^I \\ v & \textit{Se } m_A(a') = a \wedge val_{L_{p_t}}(a') = y_{d_i} \wedge y_{d_i} \in X_L^O \end{cases}$$

com $v = Op_{Out2}(Op_{Out1}(Op_{In}(val_G(m_A(a_1)), ..., val_G(m_A(a_n))), \mu(t)), val_G(m_A(a)))$.

- *Considerando a preservação de tokens (Modo 2):*

$\forall a \in Attr_G \cdot val_H(a) =$

$$\begin{cases} val_G(a) & \textit{Se } a \notin rng(m_A) \\ val_G(m_A(a)) & \textit{Se } m_A(a') = a \wedge val_{L_{p_t}}(a') = x_{d_i} \wedge x_{d_i} \in X_L^I \\ v & \textit{Se } m_A(a') = a \wedge val_{L_{p_t}}(a') = y_{d_i} \wedge y_{d_i} \in X_L^O \end{cases}$$

com $v = Op_{Out2}(Op_{Out1}(Op_{In}(val_G(m_A(a_1)), ..., val_G(m_A(a_n))), \mu(t)), val_G(m_A(a)))$.

Pela definição de $Trad_G$, *tem-se que*
$Trad_G(M', N) = ((V, \varnothing, \varnothing, \varnothing, A, Attr, val, attrv), t, T)$, *onde:*

- $V = P$
- A *é a álgebra inicial para* $TFuzzy$
- $Attr = D$
- $\forall d \in Attr \cdot attrv(d) = p$ *e* $val(d) = M'(p)$, *com* $\beta(p) = d$
- $\forall p \in V \cdot t_V(p) = p$
- $\forall d \in Attr \cdot t_A(d) = d$

- $\forall u \in \mathcal{U}(A) \cdot t_{Alg}(v) = [0,1]$
- $T = Trad_T(N)$

Pela definição de H pode-se observar que:

- $V = P = V_G = V_H$
- $A = A_G = A_H$
- $Attr = Attr_G = Attr_H$
- $\forall d \in Attr \cdot attrv(d) = p = attrv_G(d) = attrv_H(d)$, *com* $\beta(p) = d$
- $\forall p \in V \cdot t_V(p) = p = t_{V_G}(p) = t_{V_H}(p)$
- $\forall d \in Attr \cdot t_A(d) = d = t_{A_G}(d) = t_{A_H}(d)$
- $\forall u \in \mathcal{U}(A) \cdot t_{Alg}(u) = [0,1] = t_{Alg_G}(u) = t_{Alg_H}(u)$
- *Considerando a não preservação de tokens (Modo 1):*

  $\forall d \in Attr \cdot val(d) = M'(p) = val_H(d)$, *pois,* $val(d) =$

  - 0 *Se* $p \in I(t)$ *(pois, se* $p \in I(t)$*, então*
    $= val_H(d)$ $m_A(d') = d$ *e* $val_{L_{p_t}}(d') \in X_L^I$*).*
  - $M(p)$ *Se* $p \notin I(t) \cup O(t)$ *(pois, se* $p \notin I(t) \cup O(t)$*, então*
    $= val_G(d)$ $d \in rng(m_A)$*).*
    $= val_H(d)$
  - $Op_{Out2}(Op_{Out1}(Op_{In}(M(p_1), ..., M(p_n)), \mu(t)), M(p))$ *Se* $p \in O(t)$
    $= Op_{Out2}(Op_{Out1}(Op_{In}(val_G(m_A(d_1)), ..., val_G(m_A(d_n))), \mu(t)), val_G(m_A(d)))$
    $= val_H(d)$ *(pois, se* $p \in O(t)$*, então* $m_A(d') = d$ *e* $val_{L_{p_t}}(d') \in X_L^O$*).*
- *Considerando a preservação de tokens (Modo 2):*

  $\forall d \in Attr \cdot val(d) = M'(p) = val_H(d)$, *pois,* $val(d) =$

  - $M(p)$ *Se* $p \in I(t)$ *(pois, se* $p \in I(t)$*, então*
    $= val_G(d)$ $m_A(d') = d$ *e* $val_{L_{p_t}}(d') \in X_L^I$*).*
    $= val_H(d)$
  - $M(p)$ *Se* $p \notin I(t) \cup O(t)$ *(pois, se* $p \notin I(t) \cup O(t)$*, então*
    $= val_G(d)$ $d \in rng(m_A)$*).*
    $= val_H(d)$
  - $Op_{Out2}(Op_{Out1}(Op_{In}(M(p_1), ..., M(p_n)), \mu(t)), M(p))$ *Se* $p \in O(t)$
    $= Op_{Out2}(Op_{Out1}(Op_{In}(val_G(m_A(d_1)), ..., val_G(m_A(d_n))), \mu(t)), val_G(m_A(d)))$
    $= val_H(d)$ *(pois, se* $p \in O(t)$*, então* $m_A(d') = d$ *e* $val_{L_{p_t}}(d') \in X_L^O$*).*

Com isso conclui-se que $Trad_G(M', N) = H$*, para ambos os modos de disparo.*

As proposições 2 e 3 permitem concluir que a tradução proposta preserva a semântica da RPFG. Pois, a GGA obtida não apresenta comportamentos fora do contexto da semântica da rede generalizada.

Teorema 1 ***(Preservação da Semântica):*** *Dada uma* ***RPFG*** $N = (P, Tr, D, I, O, \mu, \beta, \lambda, Op, \delta, M_0)$ *e sua tradução para* ***GGA*** *(considerando o Modo 1 ou 2 de disparo)* $GG = (T, G0, Rules)$*, uma transição* $t \in Tr$ *está habilitada em uma marcação* M *de* N *e seu disparo deriva* M' *(considerando o Modo 1 ou 2 de disparo) se e somente se existe uma ocorrência* m *para a regra* $p_t \in Rules$ *em* G *(um grafo estado da GGA) resultando em um grafo* H*, de modo que* $Trad_G(M, N) = G$ *e* $Trad_G(M', N) = H$*.*

Prova 4 *Segue das proposições 2 e 3.*

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foi proposta uma tradução de RPFG para GGA, onde, os elementos fuzzy da rede foram incorporados aos grafos através da construção de uma especificação algébrica fuzzy para a GGA. A estrutura da rede deu origem aos componentes da gramática, ou seja, o grafo tipo, o grafo inicial e o conjunto de regras. Na tradução, foram modelados dois modos de disparo das transições (com e sem a preservação dos tokens de entrada) isto derivou dois tipos de regras distintos para a GGA os quais simulam tais modos.

A prova de preservação da semântica pela codificação também foi realizada. Esta prova, permitiu comprovar que o comportamento da RPFG foi mantido pela GGA durante a tradução, isto é, ambos modelos possuem equivalência semântica e preservam a alcançabilidade entre os estados de uma determinada base de regras fuzzy modelada por tais abordagens.

Para alcançar este objetivo, foi realizada uma revisão sistemática sobre o referencial teórico que envolve a codificação entre RPs e GGs. Dentre diversos tipos de RPF, optou-se pela RPFG por agregar triplas de operadores fuzzy às transições o que possibilita uma maior flexibilidade na modelagem permitindo usar diferentes tipos de operações de acordo com o sistema que está sendo modelado. A sintaxe e a semântica dos modelos foram analisados a fim de relacionar as similaridades existentes entre ambos. Métodos de tradução de RP para GG desenvolvidos anteriormente foram estudados para realizar a codificação que envolve elementos fuzzy. E por fim, elaborou-se a prova de preservação do comportamento de ambos os modelos.

A tradução aqui apresentada, difere-se das já realizadas por levar em conta valores e operações fuzzy pertencentes à rede. Para manter o comportamento da RPFG os grafos das regras são formados pelas pré(pós)-condições das transições e os valores dos atributos, os quais representam a pertinência dos tokens, são definidos pelas equações que compõem cada regra.

Em comparação com as GGs Fuzzy revisadas, o modelo de GGs baseado nesta tradução pode dar origem à GGs fuzzy mais flexíveis por permitir a escolha das operações fuzzy utilizadas nas regras de transformação. Além disso, utilizar GGAs, que

é um tipo de GG já estabelecido, permite o uso de várias técnicas e ferramentas de análise disponíveis as quais não dão suporte para as GGs Fuzzy já existentes.

A maior dificuldade encontrada no decorrer da tradução foi definir a especificação algébrica associada à GGA_{RPFG}. Devido a não representabilidade continua do intervalo de números reais $[0, 1]$ nos sistemas computacionais, uma aproximação do mesmo foi definida na especificação algébrica. Essa aproximação foi necessária para que se possa discretizar os valores contidos nesse intervalo para serem operados pelas ferramentas disponíveis para GG, tais como, o Rodin.

No desenvolvimento do projeto, alguns detalhes da tradução construída se mostraram incorretos durante a realização das provas das proposições e do teorema que atestam a equivalência semântica entre ambos os modelos. Isso se deve ao fato de que alguns detalhes, os quais à primeira vista pareciam ser óbvios, foram identificados como errôneos no decorrer das provas permitindo a correção da tradução.

A principal contribuição científica reside na construção de um novo modelo disponível para a abordagem fuzzy derivado da tradução o qual pode ser estendido com outros elementos, como arestas e outros tipos de vértices. Com isso, obtêm-se um modelo mais genérico para a modelagem de sistemas que possuem atributos fuzzy, além de dispor das técnicas e ferramentas existentes para Gramática de Grafos, como os provadores de teoremas, a fim de realizar a verificação de propriedades desejáveis para sistemas fuzzy em fase de especificação.

Como trabalhos futuros propõem-se construir um novo modelo de GGAs Fuzzy, usando como base a tradução definida neste trabalho e utilizar a ferramenta Rodin, mais especificamente o provador de teoremas, para realizar a verificação de propriedades dos sistemas especificados com este novo modelo. Outro trabalho futuro que pode ser considerado é a alteração da especificação algébrica $SPEC_{TFuzzy}$ para considerar outros tipos de Lógica Fuzzy.

REFERÊNCIAS

AZIZ, M.; BOHEZ, E. L.; PARNICHKUN, M.; SAHA, C. Classification of fuzzy petri nets, and their applications. **World Academy of Science, Engineering and Technology**, [S.l.], v.72, n.2010, p.394–401, 2010.

BALDAN, P.; CORRADINI, A.; GADDUCCI, F.; MONTANARI, U. From Petri Nets to Graph Transformation Systems. **ECEASST**, [S.l.], v.26, jan 2010.

BALDAN, P.; CORRADINI, A.; KÖNIG, B. A static analysis technique for graph transformation systems. In: INTERNATIONAL CONFERENCE ON CONCURRENCY THEORY, 2001. **Anais...** [S.l.: s.n.], 2001. p.381–395.

BARDOHL, R.; ERMEL, C.; PADBERG, J. **Formal Relationship between Petri Nets And Graph Grammars as Basis for Animation View in GenGED**.

BASSANEZI, R.; BARROS, L. **Tópicos de Lógica Fuzzy e Biomatemática**. [S.l.]: Coleção IMECC – Textos didáticos, 2010. 392p. v.5.

BENINI, L. C.; JR MENEGUETTE, M. **UMA ABORDAGEM PARA MODELAGEM DE DADOS COM O USO DE SISTEMAS NEURO-FUZZY**: APLICAÇÕES GEOESPACIAIS. São Carlos/SP: [s.n.], 2009. Sociedade Brasileira de Matemática Aplicada e Computacional.

BIERMANN, E.; ERMEL, C.; MODICA, T.; SYLOPP, P. Implementing Petri Net Transformations using Graph Transformation Tools. **ECEASST**, [S.l.], v.14, jan 2008.

BUSTINCE SOLA, H. et al. A historical account of types of fuzzy sets and their relationships. **IEEE Transactions on Fuzzy Systems**, [S.l.], v.24, 2016.

CARDOSO, J.; VALETTE, R.; DUBOIS, D. Fuzzy Petri nets: an overview. **IFAC Proceedings Volumes**, [S.l.], v.29, n.1, p.4866–4871, 1996.

CARDOSO, J.; VALETTE, R.; DUBOIS, D. Possibilistic Petri nets. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, [S.l.], v.29, n.5, p.573–582, oct 1999.

CAVALHEIRO, S. A. C. **Relational approach of graph grammars**. 2010. Trabalho de Conclusão (Doutorado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

CAVALHEIRO, S. A. d. C.; FOSS, L.; RIBEIRO, L. Theorem proving graph grammars with attributes and negative application conditions. **Theoretical Computer Science**, [S.l.], v.686, p.25–77, 2017.

CHEN, S.-M. Fuzzy backward reasoning using fuzzy Petri nets. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, [S.l.], v.30, n.6, p.846–856, 2000.

CHEN, S.-M. Weighted fuzzy reasoning using weighted fuzzy Petri nets. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], v.14, n.2, p.386–397, 2002.

CHEN, S.-M.; KE, J.-S.; CHANG, J.-F. Knowledge representation using fuzzy Petri nets. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], v.2, n.3, p.311–319, set 1990.

CORRADINI, A. Concurrent graph and term graph rewriting. In: INTERNATIONAL CONFERENCE ON CONCURRENCY THEORY, 1996. **Anais...** [S.l.: s.n.], 1996. p.438–464.

CORRADINI, A. Concurrent Computing: from Petri Nets to Graph Grammars. **Electronic Notes in Theoretical Computer Science**, [S.l.], v.2, abr 1999.

CORRADINI, A.; MONTANARI, U. Specification of Concurrent Systems: from Petri Nets to Graph Grammars. In: QUALITY OF COMMUNICATION-BASED SYSTEMS, 1995, Dordrecht. **Anais...** Springer Netherlands, 1995. p.35–52.

CORRADINI, A.; MONTANARI, U. et al. **Algebraic Approaches to Graph Transformation, Part I**: Basic Concepts and Double Pushout Approach. [S.l.]: Universidade de Pisa, 1996.

DESEL, J.; REISIG, W. Place/transition Petri nets. In: ADVANCED COURSE ON PETRI NETS, 1996. **Anais...** [S.l.: s.n.], 1996. p.122–173.

DUBOIS, D.; PRADE, H. **Fundamentals of Fuzzy Sets**. Boston: Kluwer Pubs., 2000.

EHRIG, H.; EHRIG, K.; PRANGE, U.; TAENTZER, G. Fundamental Theory for Typed Attributed Graphs and Graph Transformation Based on Adhesive HLR Categories. **Fundam. Inf.**, Amsterdam, The Netherlands, The Netherlands, v.74, n.1, p.31–61, Jan. 2006.

EHRIG, H.; KORFF, M.; LÖWE, M. Tutorial introduction to the algebraic approach of graph grammars based on double and single pushouts. In: GRAPH GRAMMARS AND THEIR APPLICATION TO COMPUTER SCIENCE, 1991, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 1991. p.24–37.

EHRIG, H.; MAHR, B. **Fundamentals of algebraic specification 1**: Equations and initial semantics. [S.l.]: Springer Science & Business Media, 2012. v.6.

EHRIG, H.; PADBERG, J. Graph grammars and Petri net transformations. In: ADVANCED COURSE ON PETRI NETS, 2003. **Anais...** [S.l.: s.n.], 2003. p.496–536.

FOSS, L. **Transactional Graph Transformation Systems**. 2008. 123p. Trabalho de Conclusão (Doutorado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

GENRICH, H. Equivalence transformations of prt-nets. In: **High-level Petri Nets**. [S.l.]: Springer, 1991. p.426–455.

GMBH, I. **Cookbook PACE 2008**. Acessado em 10/01/2020. Disponível em: <https://manualzilla.com/doc/5772577/3-introduction-into-pace—ibe-simulation-engineering-gmbh?page=1>.

GOMIDE, F. A. C.; GUDWIN, R. R. Modelagem, controle, sistemas e lógica fuzzy. In: 1994. **Anais...** [S.l.: s.n.], 1994.

KIM, S.-y.; YANG, Y. A self-navigating robot using Fuzzy Petri nets. **Robotics and Autonomous Systems**, [S.l.], v.101, p.153–165, 2018.

KLEMENT, E. P.; MESIAR, R.; PAP, E. Triangular norms. Position paper I: basic analytical and algebraic properties. **Fuzzy Sets and Systems**, [S.l.], v.143, n.1, p.5–26, 2004. Advances in Fuzzy Logic.

KORFF, M.; RIBEIRO, L. Formal relationship between graph grammars and Petri nets. In: INTERNATIONAL WORKSHOP ON GRAPH GRAMMARS AND THEIR APPLICATION TO COMPUTER SCIENCE, 1994. **Anais...** [S.l.: s.n.], 1994. p.288–303.

KREOWSKI, H.-J. A comparison between petri-nets and graph grammars. In: GRAPHTHEORETIC CONCEPTS IN COMPUTER SCIENCE, 1981, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 1981. p.306–317.

KRÜGER. V., J.; FOSS, L.; REISER, R.; CAVALHEIRO, S. **Primeiros passos da Tradução de Redes de Petri Fuzzy para Gramática de Grafos com Atributos**. V Workshop-Escola de Informática Teórica (WEIT 2019).

LIU, H.-C.; YOU, J.-X.; LI, Z.; TIAN, G. Fuzzy Petri nets for knowledge representation and reasoning: A literature review. **Engineering Applications of Artificial Intelligence**, [S.l.], v.60, p.45–56, 2017.

MOCKOR, J. α-Cuts and models of fuzzy logic. **International Journal of General Systems - INT J GEN SYSTEM**, [S.l.], v.42, p.1–12, jan 2012.

MOTTA JAFELICE, R. S. da; BARROS, L. C. de; BASSANEZI, R. C. **Notas em Matemática aplicada 17**. São Carlos/SP: [s.n.], 2005. Sociedade Brasileira de Matemática Aplicada e Computacional.

NETO, L.; HENRIQUE, P.; COELHO, G. MINICURSO DE SISTEMA ESPECIALISTA NEBULOSO. In: SPRINGER BERLIN HEIDELBERG, 2019. **Anais...** [S.l.: s.n.], 2019.

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

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

PARASYUK, I.; YERSHOV, S. Transformational approach to the development of software architectures on the basis of fuzzy graph models. **Cybernetics and Systems Analysis**, [S.l.], v.44, n.5, p.749–759, 2008.

PAVLISKA, V. **Petri nets as fuzzy modeling tool**. [S.l.]: University of Ostrava Institute for Research and Applications of Fuzzy Modeling, 2006.

PEDRYCZ, W.; CAMARGO, H. Fuzzy timed Petri nets. **Fuzzy Sets and Systems**, [S.l.], v.140, n.2, p.301–330, 2003.

REDDY, P. V. S. Fuzzy logic based on Belief and Disbelief membership functions. In: 2017. **Anais...** Elsevier, 2017. v.9, n.4, p.405–422.

RIBEIRO, L. **Métodos formais de especificação gramáticas de grafos**. [S.l.: s.n.], 2006.

ROSS, T. J. et al. **Fuzzy logic with engineering applications**. [S.l.]: Wiley Online Library, 2004. v.2.

ROZENBERG, G. **Handbook of Graph Grammars and Computing by Graph Transformation**. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1997. v.1.

SANCHEZ, E. C. M. **Controle por aprendizado acelerado e neuro-fuzzy de sistemas servo-hidráulicos de alta frequência**. 2009. Dissertação (Mestrado em Ciência da Computação) — PUC-Rio.

SANTOS, M. C. d. **Relações formais entre gramáticas de grafos e redes de Petri**. 1999. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.

SILVA, F. F. B. **Desvendando a lógica Fuzzy**. 2011. Dissertação (Mestrado em Matemática) — Universidade Federal de Uberlândia.

SILVA, M. **Introducing Petri nets**. Dordrecht: Springer Netherlands, 1993. p.1–62.

SURAJ, Z. Generalised Fuzzy Petri Nets for Approximate Reasoning in Decision Support Systems. In: CS&P, 2012. **Anais...** [S.l.: s.n.], 2012. p.370–381.

SURAJ, Z. PNeS: Tools for the Design and Analysis Petri Nets. In: INTERNATIONAL CONFERENCE ON CONTROL, DECISION AND INFORMATION TECHNOLOGIES (CODIT), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.391–396.

SURAJ, Z.; GROCHOWALSKI, P. Petri nets and PNeS in modeling and analysis of concurrent systems. In: INTERNATIONAL WORKSHOP ON CONCURRENCY, SPECIFICATION AND PROGRAMMING, 2017. **Proceedings...** [S.l.: s.n.], 2017.

VEUNG, D.; LIU, J. N.; SHIU, S. C.; FUNG, G. S. Fuzzy coloured Petri nets in modelling flexible manufacturing systems. In: MEXICO-USA COLLABORATION IN INTELLIGENT SYSTEMS TECHNOLOGIES., 1996. **Proceedings...** [S.l.: s.n.], 1996. p.100–107.

VIRTANEN, H. **A study in fuzzy Petri nets and the relationship to fuzzy logic programming**. [S.l.]: Citeseer, 1995.

WATKINS, G. S. 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.

WATKINS, G. S. **A Framework for Interpreting Noisy, Two-dimensional Images, based on a Fuzzification of Programmed, Attributed Graph Grammars**. 1997. Tese (Doutorado em Ciência da Computação) — Rhodes University.

YEUNG, D. S.; YSANG, E. C. C. A multilevel weighted fuzzy reasoning algorithm for expert systems. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans**, [S.l.], v.28, n.2, p.149–158, mar 1998.

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

ZHOU, K.-Q.; ZAIN, A. M. Fuzzy Petri nets and industrial applications: a review. **Artificial Intelligence Review**, [S.l.], v.45, n.4, p.405–446, 2016.

ZURAWSKI, R.; ZHOU, M. Petri nets and industrial applications: A tutorial. **IEEE Transactions on Industrial Electronics**, [S.l.], v.41, n.6, p.567–583, 1994.