UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DANIEL VAZ DOS SANTOS

REPRESENTAÇÃO COMPUTACIONAL PARA GRAFOS-DE-PROVA

TRABALHO DE DIPLOMAÇÃO

PONTA GROSSA

2012

DANIEL VAZ DOS SANTOS

REPRESENTAÇÃO COMPUTACIONAL PARA GRAFOS-DE-PROVA

Trabalho de Diplomação apresentado à Coordenação de Tecnologia em Análise e Desenvolvimento de Sistemas como requisito parcial para obtenção do grau de Tecnólogo no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná.

Orientador: Gleifer Vaz Alves

PONTA GROSSA

2012

TERMO DE APROVAÇÃO

REPRESENTAÇÃO COMPUTACIONAL PARA GRAFOS-DE-PROVA

por

DANIEL VAZ DOS SANTOS

Este Trabalho de Conclusão foi apresentado em 6 de junho de 2012 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof Dr. Gleifer Vaz Alves
Orientador

Prof Msc. Danillo Leal Belmonte
Membro titular

Profa. Msc. Helyane B.Borges
Responsável pelos Trabalhos
De Conclusão de Curso

Prof Msc. Saulo Jorge Beltrão de Queiroz
Membro titular

Profa. Msc. Simone de Almeida
Coordenadora do Curso de Tecnologia em
Análise e Desenvolvimento de Sistemas
UTFPR – Câmpus Ponta Grossa

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

RESUMO

SANTOS, Daniel Vaz dos. REPRESENTAÇÃO COMPUTACIONAL PARA GRAFOS-DE-PROVA. 65 f. Trabalho de Diplomação – Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2012.

N-Grafos é um sistema lógico baseado em Dedução Natural. Com a criação dos N-Grafos, tornou-se possível representar provas da lógica proposicional através de grafos-de-prova, substituindo a tradicional representação de provas como árvores pela representação através de grafos direcionados (dígrafos). Um dos questionamentos que passaram a serem feitos foi sobre a possibilidade de aplicações computacionais que possam fazer uso dos N-Grafos. Para isso faz-se necessário a definição de uma representação computacional para os N-Grafos. Este trabalho tem como principal objetivo a definição de um *Schema* XML para a representação dos N-Grafos. A linguagem definida por esse *Schema* será chamada N-GraphML.

Palavras-chave: Grafos-de-Prova, N-Grafos, GraphML, N-GraphML

ABSTRACT

SANTOS, Daniel Vaz dos. COMPUTATIONAL REPRESENTATION FOR PROOF-GRAPHS. 65 f. Trabalho de Diplomação – Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2012.

N-Graphs is a logical system based on Natural Deduction. With the creation of N-graphs, proofs of propositional logic can now be represented by proofs-graphs, replacing the traditional representation of proofs as trees by representation as directed graphs (digraphs). One of the questions that have to be made was about the possibility of computer applications that can make use of N-Graphs. For this reason it is necessary to define a computational representation for the N-graphs. This work has as main goal the definition of XML *Schema* for the representation of N-Graphs. The language defined by the *Schema* will be called N-GraphML.

Keywords: Proof-graphs, N-Graphs, GraphML, N-GraphML

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

N-Grafos é um sistema formal de provas que utiliza regras lógicas da Dedução Natural com regras estruturais no estilo de cálculo de sequentes. O conceito de N-Grafos foi desenvolvido na tese de doutorado defendida por De Oliveira (OLIVEIRA, 2001).

Com a criação dos N-Grafos, tornou-se possível representar provas da lógica proposicional através de grafos-de-prova, substituindo a tradicional representação de provas como árvores pela representação através de grafos direcionados (dígrafos).

De acordo com (SCHROEDER, 2008), demonstrações em lógica podem tornar-se demasiada extensas e complexas. Um assistente de demonstração geral (provador automático de teoremas) deve integrar ferramentas que ajudem a especificar as lógicas, as equações, os conjuntos de regras, e as estratégias de busca (semi) automática de demonstrações. Vê-se na criação de um provador uma das possíveis aplicações computacionais dos N-Grafos.

Uma outra aplicação é a criação de ferramentas específicas para os N-Grafos. Em um trabalho recente (KASPCZAK A. E RODRIGUES, 2011) desenvolvido no grupo de Grupo de Métodos Formais e Fundamentos da Computação (GM2FC), do departamento de informática da UTFPR, foi criada uma ferramenta para a verificação de ciclos simples em N-Grafos. Atualmente está sendo desenvolvida uma extensão desta ferramenta, onde será criado um editor mais completo para os N-Grafos, capaz de verificar outras propriedades relacionadas com os grafos-de-prova. Dessa forma cria-se a necessidade de definir-se uma representação computacional para os N-Grafos, a qual possa ser aplicada nas duas ferramentas citadas previamente.

Dito isso, o principal objetivo deste trabalho é a definição de uma linguagem baseada em XML para a representação de Grafos-de-prova, o que poderá viabilizar a utilização dos N-Grafos em aplicações computacionais, como por exemplo, provadores automáticos de teoremas baseados em grafos.

1.1 PROBLEMAS E OBJETIVOS

Nesta Seção está descrito o objetivo geral e os objetivos específicos do trabalho.

1.1.1 OBJETIVO GERAL

Como objetivo geral deste trabalho, destaca-se definir uma linguagem baseada em XML para a representação computacional de Grafos-de-prova, a N-GraphML, construída a partir da linguagem GraphML, uma linguagem de marcação desenvolvida para a representação computacional de grafos.

1.1.2 OBJETIVOS ESPECÍFICOS

- A definição de uma representação computacional para os grafos-de-prova, trabalho futuro proposto em (KASPCZAK A. E RODRIGUES, 2011);
- Estudar conceitos e definições da teoria dos grafos;
- Estudar lógica formal e sistemas de prova;
- Estudar Grafos-de-Prova;
- Estudar as linguagens XML e sua definição;
- Estudar a linguagem GraphML;
- Criar a linguagem N-GraphML, a partir de GraphML;
- Contribuir com o conteúdo abordado para trabalhos futuros na área de representação computacional de Grafos-de-Prova.

1.2 ORGANIZAÇÃO DO TRABALHO

Para atingir-se o objetivo final deste trabalho, algumas etapas de estudo foram definidas. A Figura 1 demonstra de forma geral as principais etapas e elementos de estudo envolvidos neste trabalho. Foram definidas duas frentes de estudo, que seguiram em paralelo. De um lado houve os estudos de Lógica Formal e Dedução Natural, que juntamente com o estudo de Teoria dos Grafos possibilitou o entendimento dos N-Grafos. De outro lado os estudos das linguagens XML, DTD, XML *Schema* e GraphML. A convergência das duas partes do estudo resultaram na definição do *Schema* para a representação dos N-Grafos.

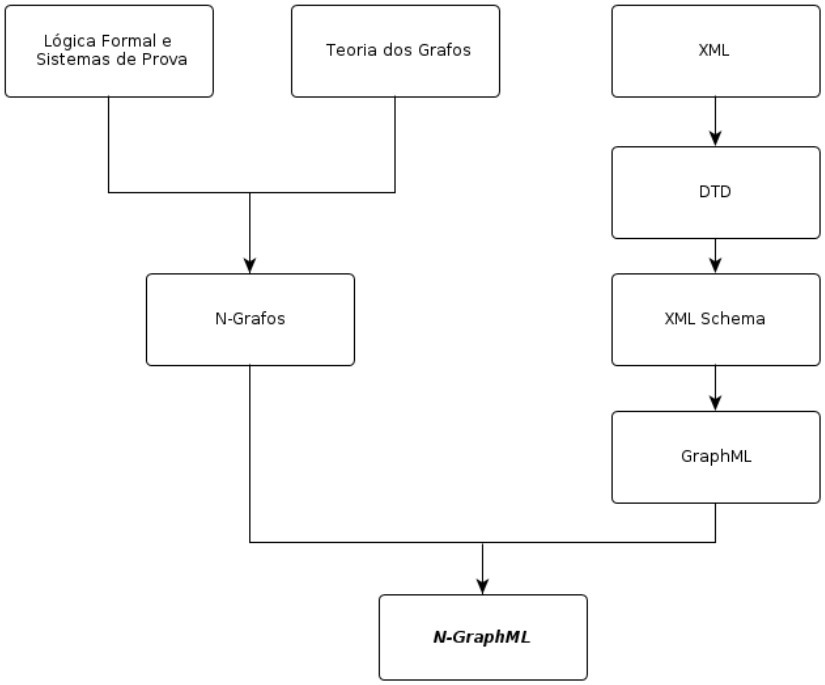

Figura 1: Etapas do trabalho
Fonte: autoria própria

Este trabalho está organizado da seguinte forma:

- O Capítulo 2 define alguns conceitos sobre a teoria dos grafos;
- O Capítulo 3 apresenta alguns conceitos de lógica formal, proposições e representações simbólicas;
- O Capítulo 4 demonstra as regras lógicas do sistema de dedução natural;
- O Capítulo 5 define alguns conceitos sobre os N-Grafos, construção de ciclos e critérios de corretude;
- O Capítulo 6 apresenta a conceitos e exemplos do funcionamento da linguagem GraphML;
- O Capítulo 7 apresenta o resultado final do trabalho, o *Schema* para a representação de N-Grafos gerado a partir da GraphML.

1.3 FERRAMENTAS UTILIZADAS

Para elaborar as figuras deste trabalho foram utilizadas as ferramentas gratuitas para representação de grafos, Yed (YWORKS, 2011) e GraphViz (BILGIN et al., 2011). Para o texto

utilizou-se o software livre Kile (KILE, 2011). As referências foram criadas com o auxílio da ferramenta BibTEX Express (Versão 1.0) (FONSECA, 2012).

2 TEORIA DOS GRAFOS

Este Capítulo aborda definições sobre teoria dos grafos que servem de base para todo o trabalho. Na Seção 2.1 estão as definições básicas sobre grafos, na Seção 2.2 as definições sobre grafos direcionados, grafos isomorfos, caminhos em grafos e grafos planares. Por último, na Seção 2.3 estão algumas das maneiras de se representar computacionalmente grafos.

2.1 DEFINIÇÃO DE GRAFO

Informalmente pode-se definir um grafo como um conjunto de vértices e arestas, onde cada aresta se conecta a dois vértices. Uma figura que poderia ilustrar bem tal definição seria o mapa de um bairro onde, por exemplo, o encontro de duas ruas (esquinas) seria o vértice e as ruas (representadas por linhas ligando os pontos) seriam as arestas. Vértices também são chamados de **nó** e arestas de **arcos**.

Definição 2.1.1 (Grafo) *Um grafo G é um par (V,E), onde V é um conjunto não vazio de* **p** *pontos, denominados vértices (ou nós); e E é um conjunto de* **q** *pares de vértices distintos de V. Cada par $e = (u,v)$ de vértices em E é denominado aresta ou linha de G, e é dito que agrega u e v; u e v são vértices adjacentes; e o vértice u e a aresta e são incidentes um com outro, assim como v e e (HARARY, 1969).*

Ex.: Grafos com vértices $(1,2,3)$, arestas (a_1,a_2,a_3) e funções g(a_1)= 1-2, g(a_2) = 2-3, g(a_3) = 3-1.

A representação de tal grafo seria um triângulo, visto na Figura 2. A função g(a_1) indica que uma de suas arestas tem como extremidades os vértices 1 e 2, e assim por diante. Porém uma forma triangular não seria a única maneira de representar o grafo descrito, pois os vértices poderiam estar alinhados por exemplo. Neste caso os grafos são denominados **isomorfos**, o que será explicado na Seção 2.2.

Se o conjunto não-vazio de vértices e o conjunto de arestas são finitos pode-se afirmar que o grafo *G* é finito.

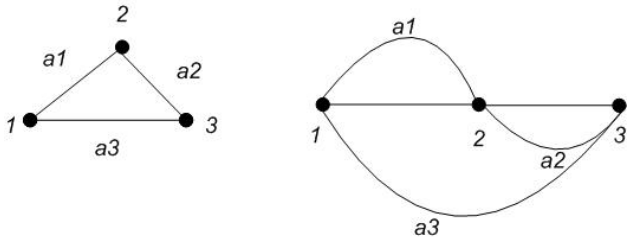

Figura 2: Exemplo de isomorfismo

Fonte: (GERSTING, 2001)

Um grafo também pode ser direcionado, ou seja, quando está definido em qual vértice começa e termina cada aresta. Um grafo direcionado também é chamado de **dígrafo.**

Além das definições comuns para um grafo, existem extensões da sua definição que podem incluir alguns elementos adicionais, como nomes identificadores aos nós (**grafo rotulado**), cores aos nós e vértices (**grafo colorido**), e valores ou pesos a cada aresta (**grafo com pesos**). Por exemplo, um grafo poderia representar um mapa, onde os nós seriam rotulados com nomes de cidades e os pesos poderiam ser a distância em km entre os nós, ou custos, como pedágios, etc (FURTADO, 1973).

Algumas definições básicas de teoria dos grafos podem causar confusão terminológica. A nomenclatura pode variar de autor para autor, deve-se portanto atentar às definições e entender suas diferenças. Abaixo uma listagem das principais definições (ALVES, 2006).

Definição 2.1.2 (Nós Adjacentes) *Dois nós podem ser ditos adjacentes quando ambos são extremidades de uma mesma aresta, independente do grafo ser direcionado ou não.*

Definição 2.1.3 (Nó isolado) *Nó que não é adjacente de nenhum outro, ou seja, nenhuma aresta começa ou termina no mesmo.*

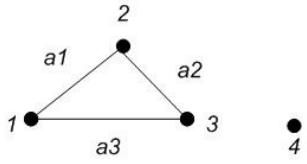

Figura 3: Exemplo de vértices adjacentes e isolados

Fonte: autoria própria

Na Figura 3 os nós 1 e 2 são exemplos de nós adjacentes e o nó 4 exemplo de nó isolado.

Definição 2.1.4 (Grau de um nó) *Número de arestas que começam/terminam em determinado nó. Um nó isolado, por exemplo, tem grau 0, já o nó de um triângulo será de grau 2.*

Definição 2.1.5 (Laço) *Diz-se de uma aresta que começa e termina em um mesmo nó, aresta com extremidades n-n em um nó n.*

Um exemplo de grafo com laços tem-se na Figura 4.

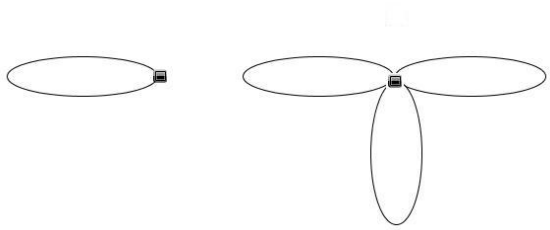

Figura 4: Exemplos de grafos com laços
Fonte: autoria própria

Definição 2.1.6 (Arestas paralelos) *Duas arestas que tenham os mesmos nós como extremidades.*

Definição 2.1.7 (Grafo simples) *Grafos que não possui nenhuma aresta paralela e nenhum laço.*

Definição 2.1.8 (Grafo completo) *Grafo no qual todo par de nós distintos é adjacente.*

Definição 2.1.9 (Subgrafo) *Um subgrafo é um grafo que se obtêm removendo parte de um grafo. Ou seja, um conjunto de nós e arestas subconjuntos de um grafo original, onde as extremidades de cada aresta devem ser as mesmas em ambos.*

Definição 2.1.10 (Caminho) *O caminho de um grafo G é uma sequência de vértices e arestas*

$$n_0, a_0, n_1, a_1, ..., n_{K-1}, n_K$$

que comece e termine em vértices, em que cada aresta ligue o vértice imediatamente precedente e o seguinte. Uma segunda maneira seria onde as arestas entre os nós ficassem subentendidas, não representando as mesmas:

$$n_0, n_1, ..., n_{K-1}, n_K$$

Definição 2.1.11 (Caminho fechado) *Um caminho fechado é aquele onde o vértice inicial é o mesmo que o final.*

$$v_0 = v_n$$

Definição 2.1.12 (Caminho simples) *Um caminho simples é um caminho onde nenhum vértice ou aresta se repete.*

Definição 2.1.13 (Trilha) *Uma trilha é um caminho onde nenhuma aresta se repete.*

Definição 2.1.14 (Comprimento) *O comprimento é o número de arestas necessárias para se ir de um nó à outro, se arestas forem repetidos são contados novamente.*

Definição 2.1.15 (Ciclo) *Ciclo é um caminho simples, ou seja, um caminho que não tenha repetições de vértices ou arestas à exceção do primeiro e último* $(v_o = v_n)$ *e o número de vértices é maior ou igual a 3* $(n \geqslant 3)$*. Um grafo sem ciclos é chamado* **acíclico***. Um ciclo é um caminho fechado, mas o contrário não é necessariamente verdade.*

Definição 2.1.16 (Grafo conexo) *Grafo em que exista obrigatoriamente ao menos um caminho de qualquer nó para qualquer outro, ou seja, não podem existir nós isolados.*

Definição 2.1.17 (Grafo bipartido completo) *É um grafo onde podemos dividir seus nós em dois conjuntos distintos, não vazios,* n_1 *e* n_2*, de forma que entre os nós de um mesmo grupo não existam nós adjacentes.*

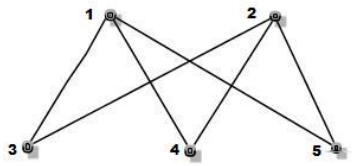

Figura 5: Grafo bipartido completo
Fonte: autoria própria

Note que a Figura 5 não retrata um **grafo completo**, pois nem todos os nós são adjacentes à outros nós, mas é **bipartido completo**, pois pode ser dividido em dois conjuntos distintos, $(1,2)$ e $(3,4,5)$ onde não há adjacência entre nós do mesmo conjunto.

2.2 DEFINIÇÕES COMPLEMENTARES

Nesta Seção estão alguns conceitos complementares da teoria dos grafos, como grafos direcionados, isomorfos e planares.

2.2.1 GRAFOS DIRECIONADOS

Esta Subseção contém algumas definições sobre grafos direcionados (dígrafos).

Definição 2.2.1 (Grafo direcionado) *Um grafo direcionado ou* **dígrafo** *G consiste em um conjunto V de vértices não vazio finito e um conjunto E de pares ordenados com vértices distintos. Em um dígrafo as arestas são denominadas arestas direcionadas (HARARY, 1969).*

Definição 2.2.2 (Grau de entrada) *Em um dígrafo o grau de entrada é o número de arestas que apontam para ele.*

Definição 2.2.3 (Grau de saída) *Em um dígrafo o grau de saída é o número de arestas que apontam para fora do mesmo.*

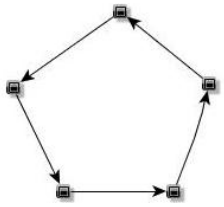

Figura 6: Grafo direcionado
Fonte: autoria própria

Neste exemplo de grafo direcionado (Figura 6) cada vértice apresenta 1 grau de entrada e 1 grau de saída.

Definição 2.2.4 (Caminho direcionado) *Em dígrafos um caminho direcionado é uma sequência de vértices e arestas* $(v_0; x_1; v_1; ...; x_n; v_n)$ *, em que cada aresta* x_i *é dada pela função* (v_{i-1}, v_i).

Definição 2.2.5 (Caminho direcionado fechado) *É o caminho direcionado que inicia e termina no mesmo vértice.*

Definição 2.2.6 (Caminho simples direcionado) *É um caminho direcionado onde não há repetição de vértices.*

Definição 2.2.7 (Ciclo direcionado) *Um ciclo direcionado é um caminho fechado de vértices distintos, à exceção do primeiro e último.*

A Figura 7 representa um grafo com caminho direcionado e caminho direcionado fechado, definido pela função $(1;a1;2a;2;3;a3;4;a4;1)$, não é um caminho simples direcionado, pois há a repetição de vértice. Pode ser considerado um ciclo direcionado, pois o único vértice que se repete é o primeiro. Já na Figura 8 tem-se um exemplo de caminho simples direcionado.

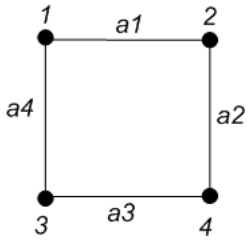

Figura 7: Caminho direcionado
Fonte: autoria própria

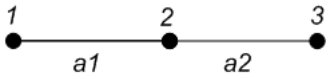

Figura 8: Caminho simples direcionado
Fonte: autoria própria

Definição 2.2.8 (Semi-caminho) *É uma sequência alternada de vértices e arestas* $(v_0;x_1;v_1;...;x_n;v_n)$ *onde cada aresta* x_i *deve ter a função* (v_{i-1},v_i) *ou* (v_i,v_{i-1}).

Definição 2.2.9 (Semi-caminho simples) *É uma sequência alternada de vértices e arestas distintos* $(v_0;x_1;v_1;...;x_n;v_n)$ *onde cada aresta* x_i *deve ter a função* (v_{i-1},v_i) *ou* (v_i,v_{i-1}).

A Figura 9 representa um grafo com semi-caminho, definido pela função $(1;a1;2;a2;3;a3;1)$. Na Figura 10 tem-se um grafo com semi-caminho simples, pois nenhum vértice se repete.

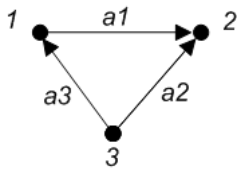

Figura 9: Semi-caminho
Fonte: autoria própria

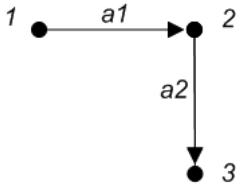

Figura 10: Semi-caminho simples
Fonte: autoria própria

Definição 2.2.10 (Semiciclo) *Em uma sequência alternada de vértices e arestas distintos* $(v_0;x_1;v_1;...;x_n;v_n)$*,* v_0 *é uma semiciclo contanto que a sequência* $v_1;v_2;...;v_n$ *seja um semi caminho simples.*

A Figura 11 representa um grafo com semiciclo.

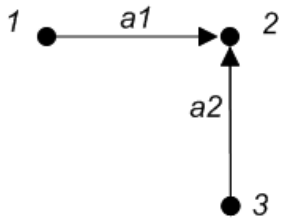

Figura 11: Semiciclo
Fonte: autoria própria

2.2.2 GRAFOS ISOMORFOS

Dois grafos podem parecer muito diferentes em sua representação visual, como na Figura 12, e ainda assim serem iguais em uma definição formal. Tal ocorrência é conhecida como **isomorfismo**. Para provar que duas estruturas são isomorfas utiliza-se uma bijeção entre os elementos das duas estruturas, ou seja, uma mudança de nomes de nós e arestas.

Definição 2.2.11 (Grafos Isomorfos) *Dois grafos* (N_1,A_1,g_1) *e* (N_2,A_2,g_2) *são* **isomorfos** *se existem bijeções* $f_1 : N_1 \to N_2$ *e* $f_2 : A_1 \to A_2$ *tais que, para cada aresta* $\in A_1, g_1(a) = x - y$ *se, e somente se,* $g_2[f_2(a)] = f_1(x) - f_1(y)$*. Onde* ***a*** *representa uma aresta, x e y representam vértices* g_1*,*g_2*,* f_1 *e* f_2 *são funções,* N_1 *e* N_2 *são conjuntos de vértices e* A_1 *e* A_2 *são conjuntos de arestas.*

Abaixo tem-se a função de correspondência entre os vértices e as arestas do grafo da Figura 12.

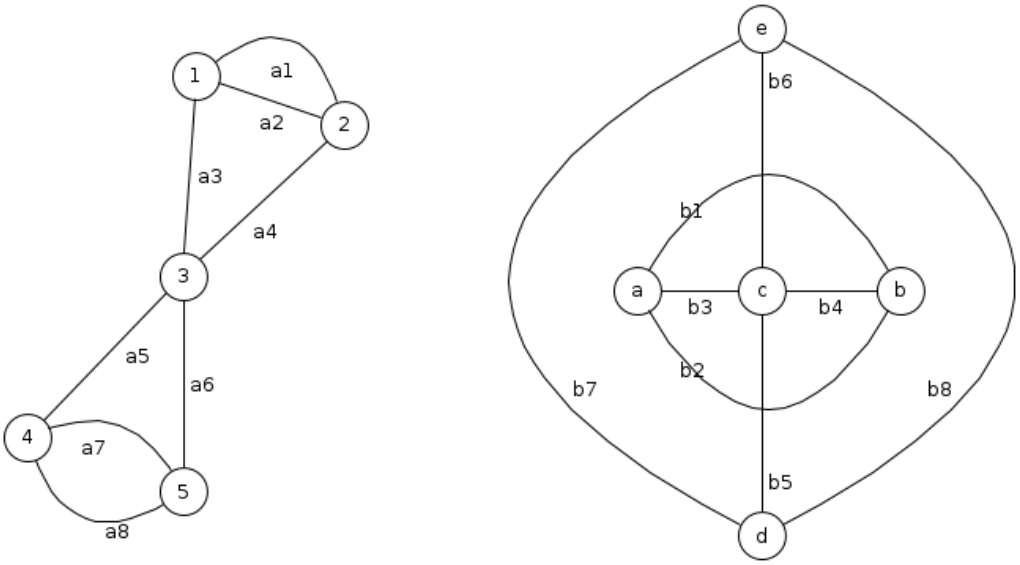

Figura 12: Grafo com isomorfismo

Vértices:

$1 \rightarrow a$

$2 \rightarrow b$

$3 \rightarrow c$

$4 \rightarrow d$

$5 \rightarrow e$

Arestas:

$a1 \rightarrow b1$

$a2 \rightarrow b2$

$a3 \rightarrow b3$

$a4 \rightarrow b4$

$a5 \rightarrow b5$

$a6 \rightarrow b6$

$a7 \rightarrow b7$

$a8 \rightarrow b8$

2.2.3 GRAFOS PLANARES

Um grafo é chamado de planar quando pode ser representado em um plano de modo que suas arestas se intersectam apenas nos vértices, como representado na figura 13. Quanto aos grafos planares tem-se o seguinte teorema (GERSTING, 2001):

Teorema 2.2.1 (Teorema sobre o Número de Vértices e Arestas) *Para um grafo planar simples e conexo com n nós e a arestas:*

1. *Se a representação planar divide o plano em r regiões, então*
   $n - a + r = 2$
2. *Se* $n \geqslant 3$*, então*

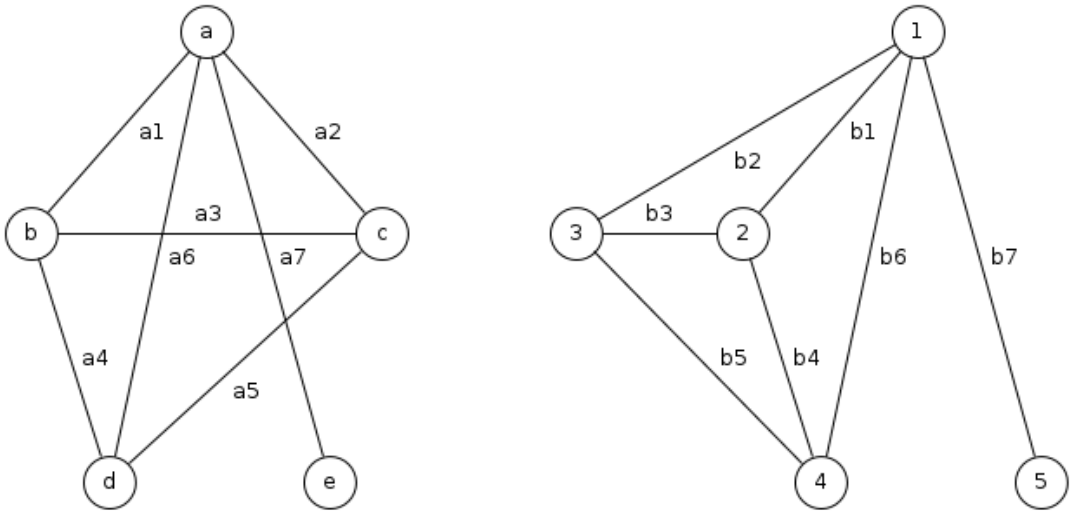

Figura 13: Grafo planar

Fonte: autoria própria

$a \leqslant 3n - 6$

3. *Se* $n \geqslant 3$ *e se não existem ciclos de comprimento 3, então*
   $a \leqslant 2n - 4$

2.3 REPRESENTAÇÃO COMPUTACIONAL DE GRAFOS

Existem diversas maneiras de se representar computacionalmente grafos. Representam-se as adjacências entre vértices e arestas. Algumas das principais maneiras são através de:

- matriz de adjacência
- lista de adjacência

2.3.1 MATRIZ DE ADJACÊNCIA

Através de uma matriz de adjacência pode-se representar tanto grafos direcionados como não-direcionados. Suponha um grafo não direcionado com n nós, numerados $n_1, n_2, ..., n_3$. A ordem da matriz será n x n, onde o elemento i, j é o número de arestas entre os nós n_1 e n_j. Essa matriz pode ser chamada **matriz de adjacência A** do grafo. Assim:

$$a_{ij} = p, \text{se existem } p \text{ arestas entre } n_i \text{ e } n_j$$

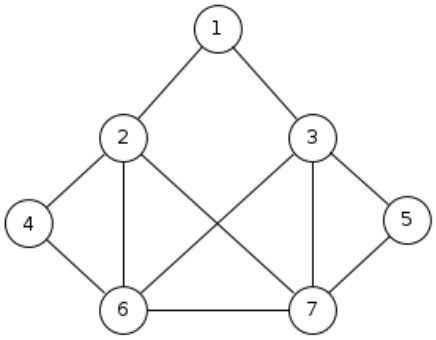

Figura 14: Grafo não-direcionado
Fonte: (GERSTING, 2001)

O elemento 1,1 é 0, pois não há laço no nó 1. O elemento 2,1 é 1 porque existe uma aresta entre os nós 1 e 2 e assim por diante.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Se fosse em um grafo direcionado, a matriz de adjacência **A** iria refletir o sentido da orientação das arestas.

$$a_{ij} = p \text{ se existem } p \text{ arestas de } n_i \text{ para } n_j$$

Ou seja, na matriz acima, se estivesse de um dígrafo o elemento 2,1 ser 1 iria significar que a aresta está direcionada do nó 2 para o nó 1.

2.3.2 LISTA DE ADJACÊNCIA

Alguns grafos podem conter muitos nós isolados, sua representação através de matrizes de adjacência nestes casos não é a mais indicada, pelo número grande de zeros (conhecida como *matriz de adjacência esparsa*) que tais matrizes contém. A melhor maneira de representar nestes casos é armazenando apenas os elementos não-nulos da matriz de adjacência. Essa

representação é uma lista para cada nó, de todos os nós adjacentes a ele. Para ir de um item ao próximo na lista são utilizado ponteiros. Essa estrutura pode ser implementada através de **lista encadeada**. Existe um arranjo de ponteiros, um para cada nó, para começar cada lista. Para encontrar todos os nós adjacentes a n_i, é preciso percorrer a lista encadeada para n_i, que pode ter bem menos elementos que teríamos que examinar em uma matriz de adjacência. A estrutura da Figura 15 representa um grafo e sua respectiva lista de adjacência:

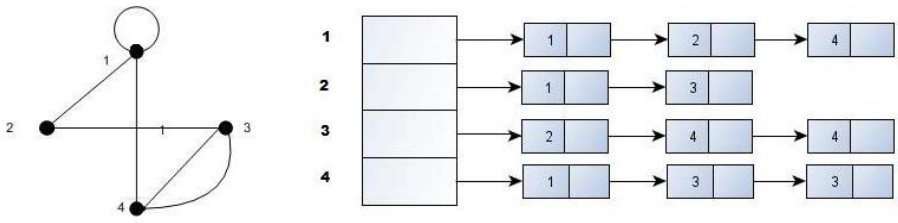

Figura 15: Lista de adjacência

Fonte: (GERSTING, 2001)

Essa lista é um arranjo de ponteiros com 4 elementos, uma para cada nó. O ponteiro de cada nó aponta para um nó adjacente, que aponta para outro nó adjacente e assim por diante. O ponto indica um ponteiro nulo, ou seja, não há mais nada a indicar, ou que se chegou ao final da lista. Em um grafo não direcionado cada aresta é representado duas vezes. Para grafos rotulados ou com pesos, dados podem ser adicionados ao rótulo do nó na lista de adjacência, como mostar a Figura 23.

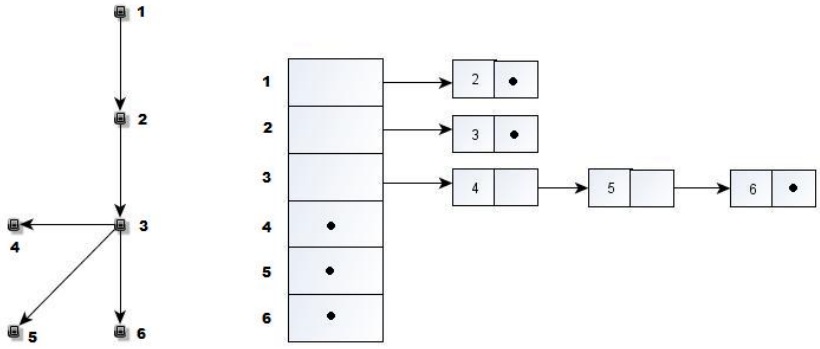

Figura 16: Grafo direcionado e lista de adjacência

Fonte: (GERSTING, 2001)

A Figura 17 representa um grafo direcionado e com pesos. Para cada registro da lista, o primeiro componente é o nó, o segundo o peso da aresta (que poderia ser qualquer informação adicional), e o terceiro é o ponteiro. No quarto nó não existe nenhuma aresta saindo, portanto o arranjo é nulo.

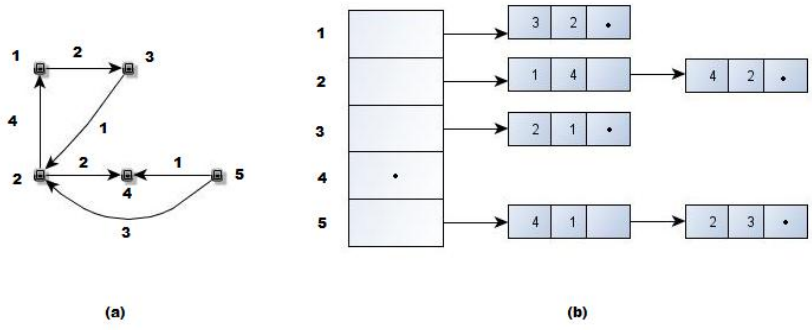

Figura 17: Grafo direcionado ponderado e lista de adjacência
Fonte: (GERSTING, 2001)

Após uma análise preliminar, conclui-se como melhor opção a utilização da representação através de lista de adjacência neste trabalho. Estudos futuros mais aprofundados poderão confirmar essa escolha.

2.4 RESUMO

Neste Capítulo foram abordadas as definições relativas à Teoria dos Grafos, como definições sobre grafos, dígrafos, isomorfismo, grafos planares, e representação computacional de grafos através de lista e matriz de adjacências. Aqui destacam-se os seguintes conceitos que serão necessários para o entendimento dos N-Grafos e criação da N-GraphML: todas as definições apresentadas sobre grafos direcionados e representação computacional de grafos.

3 LÓGICA PROPOSICIONAL E SISTEMAS DE PROVA

Este Capítulo apresenta alguns sistemas de provas que servem como base para o funcionamento e entendimento dos N-Grafos. Dessa forma, na Seção 3.1 estão os conceitos de proposição, suas representações simbólicas e as definições de definições de tautologia, contradição e contingência. Na Seção 3.2 descreve-se o sistema de Dedução Natural, explicando o funcionamento de cada uma de suas regras lógicas. Por último, a Seção 3.3 trata do Cálculo de Sequentes.

3.1 PROPOSIÇÕES E REPRESENTAÇÕES SIMBÓLICAS

Lógica formal serve para representar proposições ou fatos. Uma sentença pode ser verdadeira ou falsa. Se uma sentença é falsa ou verdadeira diz-se que é uma **proposição**. Uma pergunta não é considerada uma proposição, pois não afirma ou nega nada. Na lógica clássica aristotélica, que é utilizada como base para este trabalho, uma proposição não pode ser verdadeira e falsa ao mesmo tempo. São exemplos de proposições válidas:

- A afirmação número 1 está errada.
- O carro é mais veloz que o avião.

Não são proposições:

- O que você disse?
- Aquilo está errado.

Proposições geralmente fazem uso de conectivos lógicos, como *e, ou* etc. Em uma representação gráfica, afirmações podem ser representadas por letras maiúsculas (A,B,C...) e conectivos pelos símbolos $\wedge$ e $\vee$, que representam *e* e *ou*, respectivamente. Exemplo:

Estou com fome e vou comer

Estou com fome atribui-se A, *vou comer* atribui-se B. A frase seria representada assim:

$$A \wedge B$$

Tal expressão é chamada de **conjunção** de A e B, que são os fatores da conjunção. Se o conectivo entre os fatores fosse *ou* seria chamada **disjunção**.

A Tabela 3.1 é a tabela verdade da conjunção $A \wedge B$.

A	B	$A \wedge B$
V	V	V
V	F	F
F	V	F
F	F	F

Tabela 1: tabela verdade da conjunção $A \wedge B$

Fonte: (GERSTING, 2001)

A Tabela 3.1 é a tabela verdade da disjunção $A \vee B$.

A	B	$A \vee B$
V	V	V
V	F	V
F	V	V
F	F	F

Tabela 2: tabela verdade da disjunção $A \vee B$

Fonte: (GERSTING, 2001)

Proposições também podem ser condicionais, como "se proposição 1, então proposição 2", o conectivo lógico neste caso é o **condicional** (ou a **implicação**). A representação fica:

$$A \rightarrow B$$

Neste caso A é a proposição **antecedente** e B a proposição **consequente**. Lê-se "A implica B". Existe também um conectivo **bicondicional**, simbolizado por $\leftrightarrow$. Uma expressão $A \leftrightarrow B$ é na verdade a conjunção de $A \rightarrow B$ e $B \rightarrow A$. Na tabela 5 nota-se que $A \leftrightarrow B$ é verdadeira quando A e B tem os mesmos valores lógicos.

A	B	$A \to B$	$B \to A$	$(A \to B) \wedge (B \to A)$
V	V	V	V	V
V	F	F	V	F
F	V	V	F	F
F	F	V	V	V

Tabela 3: tabela verdade bicondicional

Fonte: (GERSTING, 2001)

A	$\neg A$
V	F
F	V

Tabela 4: tabela conectivo de negação

Fonte: (GERSTING, 2001)

Os conectivos vistos acima são ditos **conectivos binários**, pois juntam duas expressões através de um conectivo lógico e produz uma terceira expressão. Um exemplo de **conectivo unário** é o conectivo de negação, na tabela 3.1, representado por $\neg A$ (lê-se *não A*).

O conectivo da negação $\neg$ dá origem à **função verdade** $f^{\neg}$. A função a partir do conjunto $\{\text{V, F}\}$ para si próprio dada pela tabela-verdade: $f^{\neg}(V) = F$ e $f^{\neg}(F) = V$.

Uma expressão válida é dita **fórmula bem formada** ou **fbf**. A Definição 3.1.1 foi retirada de (ALVES, 2011b).

Definição 3.1.1 (Fórmula bem formada) *Uma fórmula bem formada (**fbf**) é:*

1. *Uma proposição atômica p,q,r ...*
2. *Se A e B são **fbf**, então:*
   - $(A \wedge B)$
   - $(A \vee B)$
   - $(A \to B)$
   - $(A \leftrightarrow B)$
   - $\neg A$

   são fórmulas bem formadas.

Quando houver conectivos dentro de vários parênteses, efetua-se primeiro as expressões dos parênteses mais internos:

$$((A \vee B) \wedge C) \rightarrow (B \vee \neg C)$$

O conectivo principal aqui é $\rightarrow$, uma maneira simplificada da representação, omitindo detalhes seria:

$$P \rightarrow Q$$

Onde P e Q representam respectivamente $((A \vee B) \wedge C)$ e $(B \vee \neg C)$.

3.1.1 TAUTOLOGIA, CONTRADIÇÃO E CONTINGÊNCIA

Seguem-se as definições formais (ALVES, 2011a):

Definição 3.1.2 (Tautologia) *Uma forma proposicional é uma tautologia se todos os possíveis valores verdade atribuídos resultam sempre em V(verdadeiros).*

Definição 3.1.3 (Contradição) *Uma forma proposicional é uma contradição se todos os possíveis valores verdade atribuídos resultam sempre em F (falsos).*

Definição 3.1.4 (Contingência) *Uma forma proposicional que não é uma tautologia nem uma contradição é dita ser uma contingência.*

Um exemplo de tautologia é a expressão $A \vee \neg A$, pois todos os valores para A são verdadeiros. $A \wedge \neg A$ é uma contradição e $(A \vee B) \rightarrow \neg A$ é uma contingência.

3.2 DEDUÇÃO NATURAL

O sistema de dedução natural serve para demonstrar de forma clara o passo-a-passo de cada decisão tomada, até a chegada de um resultado final. Esta sequência de passos ou ideias é demonstrada de forma explícita, ou seja, como tudo deve ser demonstrado e provado, não existem axiomas (sentença que não é demonstrada ou provada) em dedução natural, sendo assim formando um conjunto de regras lógicas (KASPCZAK A. E RODRIGUES, 2011).

A ideia básica de dedução natural é uma assimetria: a prova é uma estrutura vagamente semelhante a uma árvore com uma ou mais hipóteses (possivelmente nenhuma) mas uma única conclusão. Em dedução natural a maioria das regras são simétricas, com algumas exceções como, por exemplo, as regras do conectivo da disjunção ($\vee$) que são assimétricas. A simetria

profunda do cálculo é mostrada pela introdução e eliminação de regras que combinam entre si exatamente. Observa-se, aliás, que em uma estrutura de árvore, sempre se pode decidir qual a regra utilizada pela última vez, o que é algo que não se poderia dizer se houvesse várias conclusões (GIRARD, 1989).

3.2.1 REGRAS LÓGICAS - CONJUNÇÃO ($\wedge$)

1. Eliminação ($\wedge$)

$$\frac{P \wedge Q}{P} E-\wedge$$

$$\frac{P \wedge Q}{Q} E-\wedge$$

   Eliminando-se a conjunção das premissas P e Q fica apenas P ou Q.

2. Introdução ($\wedge$)

$$\frac{P \qquad Q}{P \wedge Q} I-\wedge$$

   Premissas P e Q, introduz-se o conectivo $\wedge$ resulta em $P \wedge Q$.

3.2.2 REGRAS LÓGICAS - DISJUNÇÃO ($\vee$)

1. Eliminação ($\vee$)

$$\frac{\begin{array}{ccc} & [P] & [Q] \\ & \vdots & \vdots \\ P \vee Q & P \rightarrow R & Q \rightarrow R \end{array}}{R} E-\vee$$

   $P \vee Q$, eliminando-se a disjunção P está para R assim como Q está para R, ou somente R.

2. Introdução ($\vee$)

$$\frac{P}{P \vee Q} I-\vee$$

$$\frac{Q}{P \vee Q} I-\vee$$

   Premissa P, introduz-se a disjunção resulta em P ou uma premissa que pode ser dita Q.

3.2.3 REGRAS LÓGICAS - IMPLICAÇÃO ($\rightarrow$)

1. Eliminação ($\rightarrow$)

$$\frac{P \qquad P \rightarrow Q}{Q} E_{\rightarrow}$$

   Premissa P, P implicando em Q, elimina-se a implicação fica apenas Q.

2. Introdução ($\rightarrow$)

$$\begin{array}{c} [P] \\ \vdots \\ \dfrac{R}{P \rightarrow R} \end{array} I_{\rightarrow}$$

   De uma hipótese P conclui-se R, então P implica em R.

3.2.4 REGRAS LÓGICAS - NEGAÇÃO ($\neg$)

1. Eliminação ($\neg$)

$$\frac{\neg\neg P}{P} E_{\neg}$$

   $\neg\neg P$, é o mesmo que apenas P pois é o mesmo que o contrário do contrário da premissa.

2. Introdução ($\neg$)

$$\begin{array}{c} [P] \\ \vdots \\ \dfrac{\textit{Contradição}}{\neg P} \end{array} I_{\neg}$$

   De uma hipótese P chega-se a uma contradição, introduzindo a negação na hipótese P esta fica $\neg P$.

3.2.5 REGRAS LÓGICAS - BICONDICIONAL ($\leftrightarrow$)

1. Eliminação ($\leftrightarrow$)

$$\frac{P \leftrightarrow Q}{P \rightarrow Q} E_{\leftrightarrow}$$

$$\frac{P \leftrightarrow Q}{Q \rightarrow P} E_{\leftrightarrow}$$

Se uma hipótese P é bicondicional à Q, eliminando-se a bicondição diz-se que P implica em Q, mas o contrário não é necessariamente verdade.

2. Introdução ($\leftrightarrow$)

$$\frac{P \rightarrow Q \qquad Q \rightarrow P}{P \leftrightarrow Q} I_{\leftrightarrow}$$

Se uma hipótese P implica em Q e a hipótese Q implica em P, pode-se introduzir a bicondicional. Ou seja, B está para P assim como P está para Q.

3.2.6 EXEMPLOS DE PROVA

Abaixo são demonstrados alguns exemplos de deduções:

1. $D \rightarrow V, D \vdash R$

$$\frac{D \qquad D \rightarrow V}{V} E_{\rightarrow}$$

Premissa D implicando em V, eliminando a implicação conclui-se apenas V.

2. $P, Q \rightarrow R, P \rightarrow Q \vdash R$

$$\frac{\dfrac{P \qquad P \rightarrow Q}{Q} E_{\rightarrow} \qquad Q \rightarrow R}{R} E_{\rightarrow}$$

Premissa P, P implicando em R, P implicando em Q. Elimina-se implicação até chegar na premissa esperada, R.

3. $P \rightarrow V, \neg V \vdash \neg P$

$$\frac{\dfrac{\dfrac{[P]^a \qquad P \rightarrow V}{V} E_{\rightarrow} \qquad \neg V}{V \wedge \neg V} I_{\wedge}}{\neg P} I^a_{\neg}$$

Cria-se uma hipótese P para eliminar a implicação, fica apenas V. Utilizando a negação de V e introduzindo a conjunção chega-se à uma contradição entre V e não V. Introduz a negação na hipótese a, para assim chegar na conclusão $\neg P$.

4. $P \to Q \vdash \neg P \vee Q$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[P]^a \qquad P \to Q}{Q} E_{\to}}{\neg P \vee Q} I_{\vee} \qquad [\neg(\neg P \vee Q)]^b}{(\neg P \vee Q) \wedge \neg(\neg P \vee Q)} I_{\wedge}}{\neg P} I^a_{\neg}}{\neg P \vee Q} I_{\vee} \qquad \neg(\neg P \vee Q)}{(\neg P \vee Q) \wedge \neg(\neg P \vee Q)} I_{\wedge}}{\neg\neg(\neg P \vee Q)} I^b_{\neg}}{\neg P \vee Q} E_{\neg}$$

- Cria a hipótese $[P]$ para assim eliminar a implicação, resultando em Q;
- Introduz-se a implicação inserindo um $\neg P$ à direita do Q;
- Cria-se uma hipótese b, oposta ao resultado do passo anterior;
- Introduz-se a conjunção para assim gerar uma contradição;
- Introduz-se a negação na hipótese a, eliminando ela, e produzindo um $\neg P$;
- Introduz uma disjunção ao lado esquerdo com a premissa Q;
- Utiliza-se de novo a hipótese b, unindo-as com a conjunção, produzindo assim uma contradição;
- Introduzir a negação na hipótese b, cancelando esta. O resultado será $\neg\neg(\neg P \vee Q)$;
- Eliminando a negação chegamos a conclusão.

3.3 CÁLCULO DE SEQUENTES

O cálculo de sequentes é um sistema de provas que possui similaridades com a dedução natural. No entanto, o cálculo de sequentes tem um aspecto mais sistemático e possui várias aplicações computacionais, como, por exemplo, em provadores automático de teoremas (GIRARD, 1989).

Entretanto, aplica as regras (diretamente) através de associações, suporta provas com retrocesso de forma natural e não faz uso de suposições temporárias. Com essas características, ele ganha em facilidade na automação de provas. Consequentemente, é utilizado em aplicações que combinam estruturas de controle, como os provadores de teoremas (ALVES,).

3.3.1 DEFINIÇÃO

As definições de sequentes desta Subseção baseiam-se em (GIRARD, 1989):

Definição 3.3.1 (Sequente) *Um sequente é uma expressão $A \vdash B$, onde A e B são sequências finitas da fórmula: $A, A_1, \ldots A_n \vdash B, B_1, \ldots B_n$. Por denotação tem-se que a conjunção de A implica na disjunção de B.*

Tem-se:

- Se A está vazio, o sequente afirma a disjunção de B;
- Se A é vazio e B é apenas B_1, confirma-se B_1;
- Se B está vazio, nega-se a conjunção A de A_1;
- Se A e B estão vazios, confirma-se que há contradição.

3.4 REGRAS DO CÁLCULO DE SEQUENTES

As regras do cálculo de sequentes são dividas em dois grupos de regras: lógicas e estruturais.

Na Tabela 3.4 tem-se as seguintes regras estruturais: inclusão (Inc), enfraquecimento (Enf), corte (Cor), intercâmbio (Int) e contração (Con).

	esquerda	*direita*
Inc	$\dfrac{}{\Gamma, \alpha \vdash \alpha, \Delta}$	Cor $\dfrac{\Gamma \vdash \alpha \qquad \Gamma, \alpha \vdash \Delta}{\Gamma \vdash \Delta}$
Enf	$\dfrac{\Gamma \vdash \Delta}{\Gamma, \alpha \vdash \Delta}$	$\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash \alpha, \Delta}$
Int	$\dfrac{\Gamma, \alpha, \beta, \Delta \vdash \Theta}{\Gamma, \beta, \alpha, \Delta \vdash \Theta}$	$\dfrac{\Gamma \vdash \Delta, \alpha, \beta, \Theta}{\Gamma \vdash \Delta, \beta, \alpha, \Theta}$
Con	$\dfrac{\Gamma, \alpha, \alpha, \vdash \Delta}{\Gamma, \alpha \vdash \Delta}$	$\dfrac{\Gamma \vdash \alpha, \alpha\Delta}{\Gamma \vdash \alpha, \Delta}$

Tabela 3.4: Regras Estruturais

Fonte: (GIRARD, 1989)

Dada a conhecida lógica usada para criar hipóteses em deduções naturais ou executar cálculos sequentes, se tem que é possível criar variações lógicas através de novos operadores, desde que a simetria seja preservada. Na Tabela 3.5, são listadas algumas destas regras lógicas:

	esquerda	*direita*
$\neg$	$\dfrac{\Gamma \vdash \alpha, \Delta}{\Gamma, \neg\alpha \vdash \Delta}$	$\dfrac{\Gamma, \alpha \vdash \Delta}{\Gamma \vdash \neg\alpha, \Delta}$
$\wedge$	$\dfrac{\Gamma, \alpha \qquad \beta, \Delta}{\Gamma, \alpha \wedge \beta \vdash \Delta}$	$\dfrac{\Gamma \vdash \alpha, \Delta \qquad \Gamma \vdash \beta, \Delta}{\Gamma \vdash \alpha \wedge \beta, \Delta}$
$\vee$	$\dfrac{\Gamma, \alpha \vdash \Delta \qquad \Gamma, \beta \vdash \Delta}{\Gamma, \alpha \vee \beta \vdash \Delta}$	$\dfrac{\Gamma \vdash \alpha, \beta, \Delta}{\Gamma \vdash \alpha \vee \beta, \Delta}$
$\rightarrow$	$\dfrac{\Gamma \vdash \alpha, \Delta \qquad \Gamma, \beta \vdash \Delta}{\Gamma, \alpha \rightarrow \beta \vdash \Delta}$	$\dfrac{\Gamma, \alpha \vdash \beta, \Delta}{\Gamma \vdash \alpha \rightarrow \beta, \Delta}$
$\leftrightarrow$	$\dfrac{\Gamma, \alpha, \beta \vdash \Delta \qquad \Gamma \vdash \alpha, \beta, \Delta}{\Gamma, \alpha \leftrightarrow \beta \vdash \Delta}$	$\dfrac{\Gamma, \alpha \vdash \beta, \Delta \qquad \Gamma, \beta \vdash \alpha, \Delta}{\Gamma \vdash \alpha \leftrightarrow \beta, \Delta}$

Tabela 3.5: Regras Lógicas

Fonte: (GIRARD, 1989)

3.5 RESUMO

Neste Capítulo foram abordadas definições relativas à lógica proposicional e sistemas de provas. Detalhou-se as regras da dedução natural, inclusive com exemplos de provas. Por último, as definições de cálculo de sequentes e suas regras lógicas e estruturais. Destaca-se que no próximo Capítulo será apresentado o sistema dos N-Grafos, o qual faz uso da dedução natural e do cálculo de sequente.

4 N-GRAFOS

Este Capítulo apresenta o conceito central do trabalho: N-Grafos. Na Seção 4.1 estão o conceito e algumas definições de importância sobre os N-Grafos e na Seção 4.2 estão os conceitos mais avançados em N-Grafos, como ciclos e critérios de corretude.

4.1 DEFINIÇÕES SOBRE N-GRAFOS

N-Grafos é um sistema formal de provas que utiliza regras lógicas da Dedução Natural com regras estruturais no estilo de cálculo de sequentes (ALVES, 2009). Derivações são representadas graficamente na forma de grafos direcionados (dígrafos) rotulados, inclusive com estrutura de múltipla conclusão de prova. Com isto, tem-se uma simetria entre as regras de introdução e de eliminação, onde também são permitidas estruturas com ciclos em derivações.

O conceito de N-Grafos, desenvolvido na tese de doutorado defendida por (OLIVEIRA, 2001), é baseado no sistema de provas com múltiplas conclusões da Dedução Natural. Nas provas os vértices são rotulados com ocorrência de fórmulas e as arestas representam passos atômicos na derivação. A linguagem proposicional dos N-Grafos tem os seguintes elementos:

- *Constantes proposicionais:* $\bot, \top$;
- *Conectivos:* $\neg, \wedge, \vee, \rightarrow$;
- *Fórmulas de ocorrências:* $A, B, C, \ldots$

E a gramática da linguagem é dada pela tupla de 4 elementos (V, Σ, R, S) onde:

- $V = \{S, C, L, F\}$;

  Possui os elementos que definem a gramática dos N-Grafos.

- $\Sigma = \{\bot, \top, A, B, C, \ldots\}$;

Σ é o conjunto finito e não-vazio de símbolos ou caracteres (HOPCROFT et al., 1939), também chamado de alfabeto da gramática dos N-Grafos.

- S é a variável inicial;
- R é o conjunto de regras definidas da seguinte forma:

$S \rightarrow C \mid L \mid F$
$C \rightarrow \bot \mid \top$
$L \rightarrow A \mid B \mid C \mid \ldots$
$F \rightarrow \neg F \mid F \wedge F \mid F \vee F \mid F \rightarrow F \mid L$

Seguem-se algumas definições sobre N-Grafos (ALVES, 2009):

Definição 4.1.1 (Sub-fórmula) *O conceito de sub-fórmula é definido indutivamente por:*

1. A é a sub-fórmula de A.
2. Se $\neg B, B \wedge C, B \vee C$, ou $B \rightarrow C$ é uma sub-fórmula de A, então também são B e C.

Definição 4.1.2 (Ponto de ramificação) *Um ponto de ramificação é um vértice em um dígrafo com três arestas ligadas a ele.*

Definição 4.1.3 (Ponto de ramificação convergente) *Um ponto de ramificação convergente é um vértice em um dígrafo com duas arestas orientadas para ele.*

Definição 4.1.4 (Ponto de ramificação divergente) *Um ponto de ramificação divergente é um vértice em um dígrafo com duas arestas orientadas para fora dele.*

Definição 4.1.5 (i) *Um link convergente é um conjunto $\{(u_1),(u_2)\}$ em que v é chamado de ponto de ramificação (ou ponto de ramificação convergente) deste link. u_1 e u_2 são as premissas e v é a conclusão.*

Definição 4.1.6 (ii) *Um link divergente é um conjunto $\{(u_1),(u_2)\}$ em que u é chamado de ponto de ramificação (ou ponto de ramificação divergente) deste link. v_1 e v_2 são as conclusões e u é a premissa.*

Definição 4.1.7 (iii) *Um link simples é uma aresta (u,v) que não pertence nem ao link convergente e nem ao link divergente. u é a premissa e v é a conclusão.*

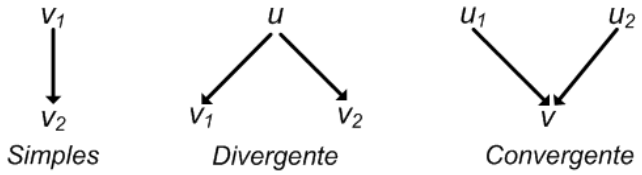

Figura 18: Três tipos diferentes de links

Fonte: (ALVES, 2009)

A Figura 18 mostra três tipos de diferentes de links, simples, divergente e convergente.

Definição 4.1.8 (Grafo-de-prova) *Um Grafo-de-prova é um grafo conexo direcionado G definido da seguinte maneira:*

- *G é construído por meio dos seguintes tipos de links: simples, convergente e divergente. Especificamente, N-Grafos são divididos em links lógicos e estruturais. Em N-Grafos utiliza-se o termo link onde termo regras também poderia ser utilizado.*
- *Existem dois tipos de arestas: meta-arestas e arestas sólidas. Meta-arestas são rotuladas com* ***m*** *para identificar sua respectiva aresta, onde a meta aresta é usada para representar o cancelamento da hipótese. Arestas sólidas não são rotuladas pela esquerda.*
- *Cada vértice é rotulado com uma fórmula de ocorrência.*
- *Todo vértice em um grafo-de-prova é rotulado com uma conclusão de um único link e é a premissa de mais de um link.*

A Figura 19 demonstra links lógicos e estruturais.

Definição 4.1.9 (Link conjuntivo) *Os links* $\wedge - I$, $\neg - E$, $\rightarrow - E$, $\top$ *- enfraquecimento convergente e link de expansão são chamados conjuntivos.*

Definição 4.1.10 (Link disjuntivo) *Os links* $\vee - E$, $\neg - I$, $\rightarrow - I$, $\bot$ *- enfraquecimento divergente e links de contração são chamados de disjuntivos.*

Definição 4.1.11 (Grau) *O* ***grau*** *de uma fórmula A é definido como o número de ocorrências de conectivos lógicos em A, exceto as constantes* $\bot$ *e* $\top$.

Definição 4.1.12 (Grau de entrada sólido de *v* **em um grafo-de-prova)** *O grau de entrada sólido de um vértice v em um grafo-de-prova é o número de arestas sólidas voltadas para ele.*

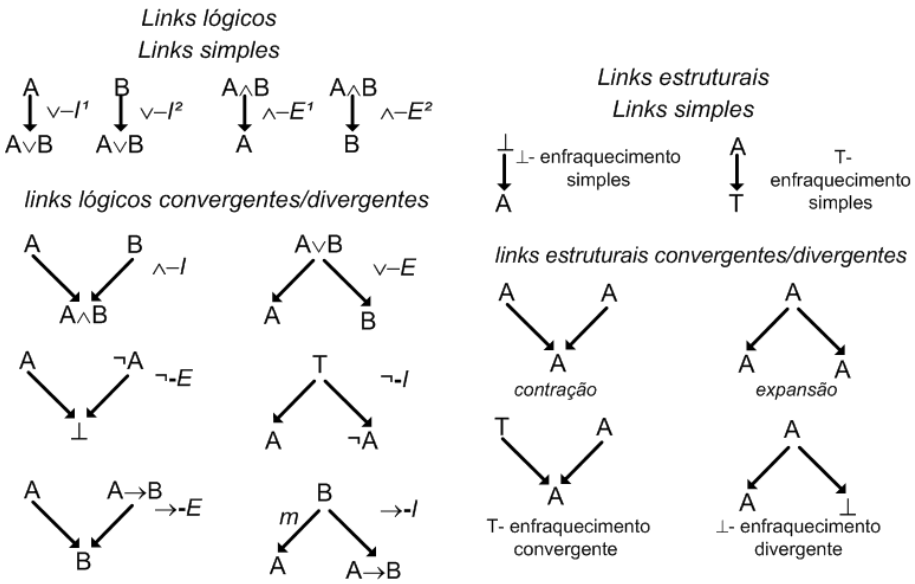

Figura 19: Links em N-Grafos

Fonte: (ALVES, 2009)

Definição 4.1.13 (Grau de saída sólido de v em um grafo-de-prova) *O grau de saída sólido de um vértice v em um grafo-de-prova é o número de arestas sólidas orientadas para fora do mesmo.*

Definição 4.1.14 (Teorema em N-Grafos) *Um N-Grafo G é um teorema se o conjunto de premissas (G) tem uma ou mais ocorrências de $\top$ constantes e o conjunto de hipóteses (G) tem zero ou mais suposições descarregadas.*

4.2 CICLOS, META-ARESTAS E CORRETUDE EM N-GRAFOS

Nesta Seção tem-se algumas definições a respeito dos N-Grafos, como o tratamento de ciclos, metas arestas e critérios de corretude. Seguem-se as definições (ALVES, 2009):

Definição 4.2.1 (Ciclo N-Grafo) *Um ciclo N-Grafo é um **semiciclo**, onde um semiciclo em um dígrafo é uma sequencia alternada de vértices e arestas distintas $v_0, v_1, \ldots, x_n, v_0$ em que cada aresta x_i pode tanto ser (v_{i-1}, v_i) como (v_i, v_{i-1}). Em outras palavras, um semiciclo é um ciclo em um dígrafo onde a direção das arestas é indeferida.*

Em um grafo-de-prova tem-se várias fórmulas como vértices de premissa e vértices de conclusão. As fórmulas restantes que não são nem vértices de premissas ou de conclusões, são chamadas apenas de vértices.

Definição 4.2.2 (Premissa de ciclo/ vértice de conclusão) *Em um ciclo N-Grafo G, o conjunto de premissas* (G) *tem o então chamado vértice de premissa de ciclo. Além disso, G tem o conjunto de conclusões* (G) *que contém o então chamado vértice de conclusão de ciclo. Ainda restam os vértice que não são nem do primeiro e nem do segundo tipo, que são chamados apenas de vértices de ciclo.*

Cada link lógico tem dois tipos de fórmulas: ***corte central*** *e* ***corte periférico****. O tipo corte central é a fórmula que contém o conectivo que está sendo introduzido ou eliminado, enquanto o tipo de corte periférico é a fórmula que não contém nem o conectivo que está sendo introduzido e nem o que está sendo eliminado.*

4.2.1 CONSTRUÇÃO DE CICLOS N-GRAFOS

Com o critério para ciclos N-Grafos pode-se extrair o padrão de um ciclo N-Grafo simples válido. Um ciclo N-Grafo deve ter um dos dois padrões:

1. O vértice de premissa de ciclo é a premissa de um link de *expansão* e o vértice de conclusão de ciclo de um link *convergente* e *conjuntivo*.
2. O vértice de premissa de ciclo é premissa de um link *divergente* e *disjuntivo* e o vértice de conclusão de ciclo de um link de *contração*.

A Figura 20 tem alguns exemplos de estruturas de ciclo.

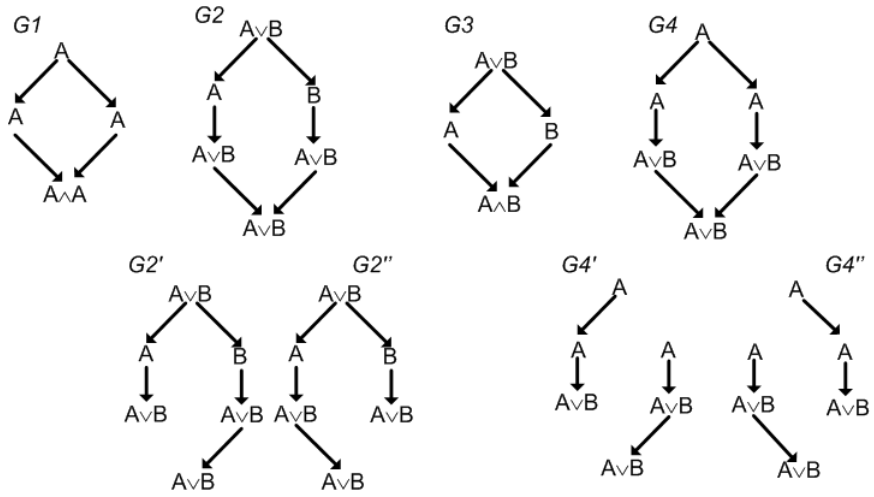

Figura 20: Exemplos de ciclos em grafos-de-prova
Fonte: (ALVES, 2009)

Para fins de esclarecimento tem-se a seguinte definição:

Definição 4.2.3 (Link de chaveamento) *Todo link de expansão ou contração.*

Na Figura 20 tem-se quatro exemplos de ciclos N-Grafos. Note que $G1$ e $G2$ são ciclos N-Grafos válidos, enquanto $G3$ e $G4$ são ciclos N-Grafos não válidos, porque $G3$ não tem link de chaveamento. Lembrando que ao menos um link de chaveamento é necessário em ciclo, desde que o critério de corretude definido abaixo na Subseção 4.2.2 seja verdadeiro.

4.2.2 CRITÉRIO DE CORRETUDE

Abaixo tem-se algumas definições que determinam o critério de corretude para N-Grafos:

Definição 4.2.4 (Grafo de chaveamento) *Dado um grafo-de-prova G, um grafo de chaveamento G associado com um G abrangente subgrafo de G em que as seguintes arestas são removidas:*

- *Uma das duas arestas de cada link de contração ou expansão;*
- *Todas as meta-arestas.*

Definição 4.2.5 (Expansão de chaveamento) *Dado um grafo-de-prova G, um grafo de expansão de chaveamento S' associado com G é um subgrafo de abrangência de G no qual todos as meta-arestas são removidas, assim como uma das duas arestas de cada link de expansão é removida.*

Definição 4.2.6 (Meta-condição) *Dado um grafo-de-prova G diz-se que a meta-condição para cada meta-aresta $(u,v)^m$ do link divergente $\{(u,v)^m,(u,w)\}$ em G é que existe um caminho ou semi-caminho sem passar através de (u,w), de v para u em cada grafo de expensão de chaveamento S' associado com G e o grau sólido de entrada de v é igual a zero.*

Definição 4.2.7 (Derivação de N-Grafos) *Um grafo-de-prova G é uma derivação de N-Grafo se existe uma meta-condição para G e cada grafo de chaveamento associado à G é acíclico e conectado.*

4.3 RESUMO

Neste Capítulo foram abordadas as definições relativas aos N-Grafos. Destacam-se as seguintes definições: links dos N-Grafos, meta-arestas e tipos de links (simples, convergente e divergente), as quais serão necessárias para a criação da linguagem N-GraphML.

5 GRAPHML

Este Capítulo está dividido da seguinte forma: a Seção 5.1 apresenta os conceitos de XML, DTD e *Schema* XML, justificando a escolha do *Schema*. Na Seção 5.2 está a definição da linguagem GraphML, exemplos da sua sintaxe na Seção 5.3. Na Seção 5.4 são demonstrados atributos adicionais que podem ser utilizados na GraphML. A Seção 5.5 explica como estender a GraphML e na Seção 5.6 encontra-se o detalhamento do *Schema* da GraphML.

5.1 XML E *SCHEMA* XML

Um documento XML "Válido" ou "Bem Formado" deve estar em conformidade com as regras de um DTD (Document Type Definition) (W3SCHOOLS, 2011a). O propósito de um DTD é definir a estrutura de um documento XML. Nele estão descritos a lista dos elementos permitidos, tipo de dados e atributos que cada um poderá ter e o correto aninhamentos de elementos. Um documento XML *Schema* (W3SCHOOLS, 2011b) tem o mesmo propósito do DTD, porém trata-se de um sucessor, oferecendo algumas vantagens:

- São extensíveis;
- São mais ricos em detalhamento que documento DTD (W3SCHOOLS, 2011b);
- São escritos em XML, o que elimina a necessidade do aprendizado de uma nova linguagem;
- Suporta tipos de dados;
- Suporta *namespaces*, o que evita conflitos entre nomes de elementos.

No Código 5.1 tem-se um documento simples em XML. Sua definição está demonstrada de duas maneiras, nos documentos *Nota.dtd* (Código 5.2) e *Nota.xsd* (Código 5.3). Deve-se reparar como o arquivo XSD (**X**ml **S**chema **D**efinition) provê uma descrição bem mais detalhada. Para adicionar extensões ao exemplo do Código 5.1 pode-se fazer modificações nos

arquivos .dtd ou .xsd, alterando a ordem dos elementos, modificando elementos, acrescentando elementos, etc.

```
<?xml version="1.0"?>
  <nota>
    <origem>Daniel</origem>
    <destino>Daniel</destino>
    <cabecalho>Lembrete</cabecalho>
    <corpo>Escrever TCC<corpo>
  </nota>
```

Código 5.1: documento simples em XML

```
<!ELEMENT nota (origem, destino, cabecalho, corpo)>
<!ELEMENT origem (#PCDATA)>
<!ELEMENT destino (#PCDATA)>
<!ELEMENT cabecalho (#PCDATA)>
<!ELEMENT corpo (#PCDATA)>
```

Código 5.2: Nota.dtd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="nota">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="origem" type="xs:string"/>
      <xs:element name="destino" type="xs:string"/>
      <xs:element name="cabecalho" type="xs:string"/>
      <xs:element name="corpo" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Código 5.3: Nota.xsd

5.2 DEFINIÇÃO DA LINGUAGEM GRAPHML

GraphML é uma linguagem de marcação desenvolvida para a representação computacional de grafos (BRANDES et al., 2011). Conta com *tags* específicas para os elementos de um grafo, como vértices e arestas.

O propósito de um documento em GraphML é definir um grafo. O documento GraphML consiste de um elemento "graphml" e uma variedade de *tags* que representam seus subelementos, como grafo, vértice, aresta.

A escolha da linguagem GraphML para a elaboração do *schema* justifica-se pelo fato de ser derivada XML (W3SCHOOLS, 2011c), uma linguagem onde os dados são armazenados

em um formato de texto explanado, o que pode facilitar o compartilhamento de dados entre aplicações diferentes, que muitas vezes poderiam trabalhar com tipos incompatíveis de dados.

Algumas das vantagens de se trabalhar com uma linguagem baseada em XML são:

- Separação dos dados e conteúdo de formatação. Pode-se utilizar-se de XML para o armazenamento de dados e HTML para formatação dos mesmos. Em uma aplicação Web, por exemplo, com poucas linhas de código JavaScript pode-se fazer a leiura de um arquivo externo XML e atualizar uma página;
- Maior facilidade para troca de informações entre bases de dados e aplicações incompatíveis, uma vez que as informações são armazenadas em forma de texto simples, tendo assim independência de tecnologias;
- Pela maneira como suas informações são armazenadas torna-se relativamente simples o transporte de dados e até mesmo mudança de plataformas;
- É uma linguagem facilmente extensível, podendo-se definir as próprias *tags* e tipos de dados.

Além das qualidades enunciadas acima, a GraphML contém uma mecanismo próprio para sua extensão.

5.3 SINTAXE DA GRAPHML

Para exemplificar a utilização da linguagem GraphML considere-se o código do dígrafo descrito no Código 5.4, onde cada vértice tem um atributo *id* e cada aresta um vértice de saída e um vértice de entrada, representados respectivamente por *target* e *source*:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <graphml xmlns="http://graphml.graphdrawing.org/xmlns"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
5        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
6    <graph id="G" edgedefault="directed">
7      <node id="n0"/>
8      <node id="n1"/>
9      <node id="n2"/>
10     <node id="n3"/>
11     <node id="n4"/>
12     <node id="n5"/>
13     <node id="n6"/>
14     <node id="n7"/>
15     <edge source="n1" target="n0"/>
16     <edge source="n2" target="n1"/>
17     <edge source="n0" target="n3"/>
18     <edge source="n1" target="n4"/>
```

```
19      <edge source="n2" target="n5"/>
20      <edge source="n6" target="n7"/>
21      <edge source="n3" target="n6"/>
22      <edge source="n3" target="n4"/>
23      <edge source="n5" target="n4"/>
24    </graph>
25  </graphml>
```

Código 5.4: dígrafo descrito em GraphML

A Figura 21 representa o dígrafo que foi descrito no Código 5.4:

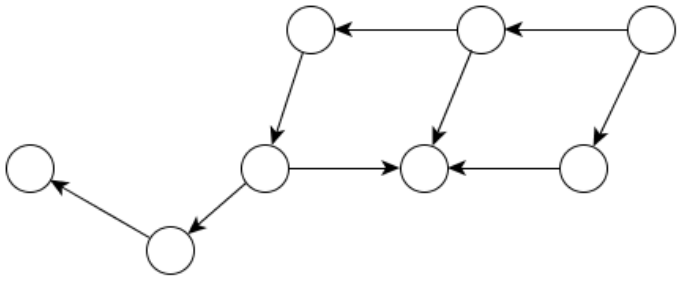

Figura 21: Exemplo de um dígrafo
Fonte: autoria própria

Um grafo é denotado pelo elemento "graph". Aninhados dentro de um elemento graph existem declarações de vértices e arestas. Um vértice é declarado com um elemento "node" e aresta com um elemento "edge". A Figura 22 representa trecho do código que descreve um grafo direcionado.

```
1  <graph id="G" edgedefault="directed">
2      <node id="n0"/>
3      <node id="n1"/>
4       ...
5      <node id="n10"/>
6      <edge source="n0" target="n2"/>
7      <edge source="n1" target="n2"/>
8       ...
9      <edge source="n8" target="n10"/>
10 </graph>
```

Código 5.5: definição de um grafo

Grafos em GraphML são mistos, ou seja, podem conter arestas direcionadas e não-direcionadas ao mesmo tempo. Se nenhuma direção for especificada a direção padrão é aplicada à aresta. A direção padrão é declarada em XML como *edgedefault* do *graph element*. Dois valores são possíveis: *directed* e *undirected*. A direção padrão deve ser especificada. Também pode-se adicionar um atributo *id*, para referenciar o grafo.

5.4 ATRIBUTOS ADICIONAIS

GraphML tem suporte à diversos tipos de atributos adicionais. Como exemplo tem-se o Código 5.6 que é a representação da Figura 22, onde tem-se arestas com pesos e vértices com cores.

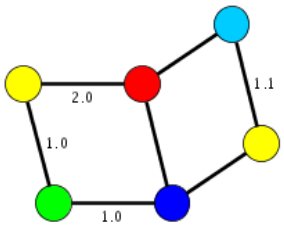

Figura 22: Grafo com coloração e pesos

Fonte: (BRANDES et al., 2011)

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
         http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <key id="d0" for="node" attr.name="color" attr.type="string">
    <default>yellow</default>
  </key>
  <key id="d1" for="edge" attr.name="weight" attr.type="double"/>
  <graph id="G" edgedefault="undirected">
    <node id="n0">
      <data key="d0">green</data>
    </node>
    <node id="n1"/>
    <node id="n2">
      <data key="d0">blue</data>
    </node>
    <node id="n3">
      <data key="d0">red</data>
    </node>
    <node id="n4"/>
    <node id="n5">
      <data key="d0">turquoise</data>
    </node>
    <edge id="e0" source="n0" target="n2">
      <data key="d1">1.0</data>
    </edge>
    <edge id="e1" source="n0" target="n1">
      <data key="d1">1.0</data>
    </edge>
    <edge id="e2" source="n1" target="n3">
      <data key="d1">2.0</data>
    </edge>
    <edge id="e3" source="n3" target="n2"/>
    <edge id="e4" source="n2" target="n4"/>
    <edge id="e5" source="n3" target="n5"/>
    <edge id="e6" source="n5" target="n4">
      <data key="d1">1.1</data>
    </edge>
  </graph>
</graphml>
```

Código 5.6: grafo com atributos

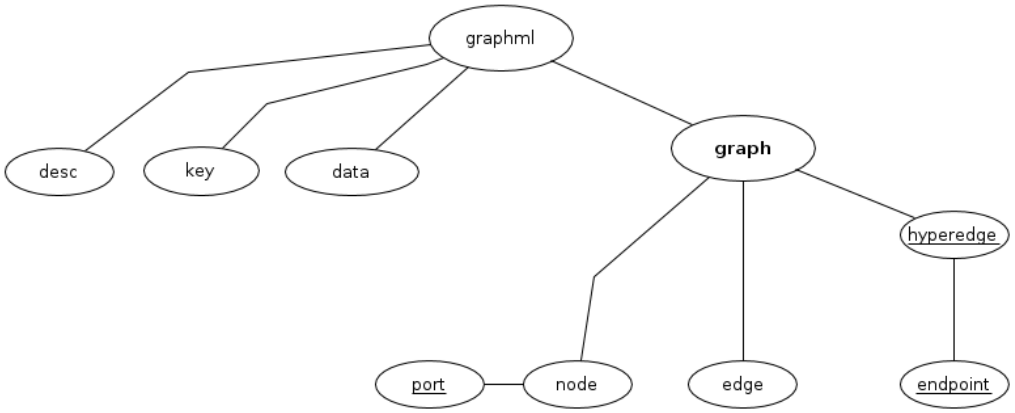

Figura 23: Hierarquia de elementos graphml
Fonte: autoria própria

A Figura 23 demonstra de forma concisa a hierarquia dos principais elementos da GraphML. Pode-se notar que *graphml* é o elemento principail e dele derivam seus subelementos primários, *desc*, *key*, *data* e *graph*. Dos elementos primários derivam os secundários, e assim por diante. É importante ressaltar que não foram retratados todos os elementos da graphml, apenas os mais relevantes para o contexto do trabalho.

5.5 ESTENDER A GRAPHML

Por se tratar de uma linguagem derivada de XML, a GraphML é pode ser facilmente estendida. Nesta Seção tem-se exemplos de uma das possibilidades de estender a GraphML, através da adição de atributos XML aos elementos GraphML.

Para adicionar um atributo XML para elementos GraphML deve-se estender a GraphML. Esta extensão pode ser definida por um XML *Schema*. O exemplo 5.7 abaixo mostra como o atributo `href` é adicionado ao `node`.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema
3      targetNamespace="http://graphml.graphdrawing.org/xmlns"
4      xmlns="http://graphml.graphdrawing.org/xmlns"
5      xmlns:xlink="http://www.w3.org/1999/xlink"
6      xmlns:xs="http://www.w3.org/2001/XMLSchema"
7      elementFormDefault="qualified"
8      attributeFormDefault="unqualified"
9  >
10
11 <xs:import namespace="http://www.w3.org/1999/xlink"
12             schemaLocation="xlink.xsd"/>
13
14 <xs:redefine
15       schemaLocation="http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
16
17   <xs:attributeGroup name="node.extra.attrib">
18     <xs:attributeGroup ref="node.extra.attrib"/>
```

```
19      <xs:attribute ref="xlink:href" use="optional"/>
20    </xs:attributeGroup>
21
22  </xs:redefine>
23
24  </xs:schema>
```

Código 5.7: Estendendo GraphML através de atributos

5.6 ESPECIFICAÇÃO

Na página da linguagem GraphML (BRANDES et al., 2011) podem ser encontrados os documentos que compõe o *Schema* da mesma. São eles os arquivos `graphml-attributes.xsd`, `graphml-parseinfo.xsd` e `graphml-structure.xsd`. É disponibilizado também o documento `graphml.dtd`, que apesar de obsoleto pode ser necessário em determinadas aplicações. Deve-se ressaltar a função de cada um dos arquivos da documentação:

- `graphml-structure.xsd`

  O *schema* correspondente à este documento define as regras estruturais do GraphML. No exemplo 5.8, tem-se um trecho de código que enumera os valores aceitos pelo elemento *endpoint*.

```
1
2   ...
3
4   <xs:simpleType name="endpoint.type.type" final="#all">
5     <xs:annotation>
6       <xs:documentation
7           source="http://graphml.graphdrawing.org/"
8           xml:lang="en">
9         Simple type for the type attribute of &lt;endpoint>.
10        endpoint.type.type is a restriction of xs:NMTOKEN
11        Allowed values: in, out, undir.
12      </xs:documentation>
13    </xs:annotation>
14    <xs:restriction base="xs:NMTOKEN">
15      <xs:enumeration value="in"/>
16      <xs:enumeration value="out"/>
17      <xs:enumeration value="undir"/>
18    </xs:restriction>
19  </xs:simpleType>
20
21  ...
```

Código 5.8: Fragmento do documento graphml-structure.xsd

- `graphml-attributes.xsd`

  Este documento define a lista de atributos possíveis na linguagem GraphML, assim como as suas regras de aninhamento. O exemplo 5.9 demonstra um trecho desse documento, onde são apresentados os tipos de dados aceitos pela linguagem.

```
...

<xs:simpleType name="key.type.type" final="#all">

  <xs:annotation>
    <xs:documentation
        source="http://graphml.graphdrawing.org/(Dokumentation der Attributes Erweiterung; entsprechende Stelle.html)"
        xml:lang="en">
      Simple type for the attr.type attribute of &lt;key&gt;.
      key.type.type is final, that is, it may not be extended
                    or restricted.
      key.type.type is a restriction of xs:NMTOKEN
      Allowed values: boolean, int, long, float, double, string.
    </xs:documentation>
  </xs:annotation>

  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="boolean"/>
    <xs:enumeration value="int"/>
    <xs:enumeration value="long"/>
    <xs:enumeration value="float"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="string"/>
  </xs:restriction>

</xs:simpleType>

...
```

Código 5.9: Fragmento do documento graphml-attributes.xsd

- `graphml-parseinfo.xsd`

  Neste documento encontram-se as informações necessárias para a extensão da GraphML. O exemplo 5.10 demonstra um trecho do código do documento, que será utilizado pelo parser (analisador sintático) da linguagem GraphML.

```
...

<xs:simpleType name="node.indegree.type" final="#all">

  <xs:annotation>
    <xs:documentation
        source="http://graphml.graphdrawing.org/"
        xml:lang="en">
      Simple type for the parse.indegree attribute of &lt;node&gt;.
      node.indegree.type is final, that is, it may not be extended
                    or restricted.
      node.indegree.type is a restriction of xs:nonNegativeInteger
      Allowed values: (no restriction).
    </xs:documentation>
  </xs:annotation>

  <xs:restriction base="xs:nonNegativeInteger"/>
</xs:simpleType>

<xs:simpleType name="node.outdegree.type" final="#all">

  <xs:annotation>
    <xs:documentation
        source="http://graphml.graphdrawing.org/"
        xml:lang="en">
      Simple type for the parse.outdegree attribute of &lt;node&gt;.
      node.outdegree.type is final, that is, it may not be extended
                    or restricted.
      node.outdegree.type is a restriction of xs:nonNegativeInteger
      Allowed values: (no restriction).
```

```
31      </xs:documentation>
32    </xs:annotation>
33
34  ...
```

Código 5.10: Fragmento do documento graphml-parseinfo.xsd

No Capítulo 6 são descritos de forma detalhada os trechos da documentação da GraphML que foram modificados, de forma à gerar a documentação da N-GraphML.

5.7 RESUMO

Neste Capítulo foram abordadas as definições relativas às linguagens XML e GraphML. Dessa forma obteve-se o entendimento de uma estrutura para a criação de uma linguagem de marcação, assim como do *schema* da própria linguagem GraphML, em conjunto com seus respectivos elementos. Essas questões são fundamentais para a definição que nomeamos N-GraphML, e será vista no Capítulo 6.

6 N-GRAPHML

Este Capítulo está dividido da seguinte forma: na Seção 6.1 tem-se uma lista com os elementos retirados do *Schema*, assim como a justificativa para tal, na Seção 6.2 tem-se a lista dos elementos que foram modificados, na Seção 6.3 a lista dos atributos que foram incluídos, na Seção 6.4 tem-se a explanação da maneira definida para a representação dos caracteres matemáticos, na Seção 6.5 está a forma adotada para a representação das regras lógicas através da N-GraphML e na Seção 6.6 estão os exemplos de N-Grafos representados através da linguagem N-GraphML.

6.1 ELEMENTOS RETIRADOS

Dois elementos foram retirados do *Schema*, são eles: `hyperegde` e `port`. O elemento `hyperegde` serve para representar uma generalização de arestas, não apenas a relação de uma para outra, como pode ser observado na Figura 24 e no Código 6.1.

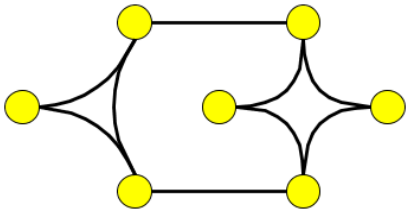

Figura 24: Hyperedge
Fonte: (BRANDES et al., 2011)

Um vértice pode especificar diferentes localizações lógicas para as arestas e hyperedges se conectarem. As localizações lógicas são chamadas “portas”(do inglês port). Como uma analogia, pode-se pensar num grafo como uma placa-mãe, os vértices como circuitos integrados e as arestas como conectores. Assim cada pino no circuito integrado iria corresponder ao `port` de um vértice. Um exemplo da utilização de um `port` pode ser observado no Código 6.2. A princípio os conceitos de *hyperedge* e *port* não terão aplicação nos N-Grafos, portanto foram

retirados do schema da N-GraphML, a fim de simplificar o mesmo.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="G" edgedefault="directed">
    <node id="n0"/>
    <node id="n1"/>
    <node id="n2"/>
    <node id="n3"/>
    <node id="n4"/>
    <node id="n5"/>
    <node id="n6"/>
    <hyperedge>
      <endpoint node="n0"/>
      <endpoint node="n1"/>
      <endpoint node="n2"/>
    </hyperedge>
    <hyperedge>
      <endpoint node="n3"/>
      <endpoint node="n4"/>
      <endpoint node="n5"/>
      <endpoint node="n6"/>
    </hyperedge>
    <hyperedge>
      <endpoint node="n1"/>
      <endpoint node="n3"/>
    </hyperedge>
    <edge source="n0" target="n4"/>
  </graph>
</graphml>
```

Código 6.1: Exemplo de utilização do elemento hyperedge

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="G" edgedefault="directed">
    <node id="n0">
      <port name="North"/>
      <port name="South"/>
      <port name="East"/>
      <port name="West"/>
    </node>
    <node id="n1">
      <port name="North"/>
      <port name="South"/>
      <port name="East"/>
      <port name="West"/>
    </node>
    <node id="n2">
      <port name="NorthWest"/>
      <port name="SouthEast"/>
    </node>
    <node id="n3">
      <port name="NorthEast"/>
      <port name="SouthWest"/>
    </node>
    <edge source="n0" target="n3" sourceport="North" targetport="NorthEast"/>
    <hyperedge>
      <endpoint node="n0" port="North"/>
      <endpoint node="n1" port="East"/>
      <endpoint node="n2" port="SouthEast"/>
    </hyperedge>
  </graph>
</graphml>
```

Código 6.2: Exemplo de utilização do elemento port

Os códigos dos elementos retirados estão representados nos Códigos 6.3 e 6.4.

```
(...)
<xs:element name="hyperedge">
  <xs:complexType>
    <xs:complexContent>
      <xs:attribute name="id" type="xs:integer" use="implied"/>
    </xs:complexContent>
(...)
```

Código 6.3: Elemento retirado hyperegde

```
(...)
<xs:attribute name="port" type="xs:integer" use="implied"/>
(...)
```

Código 6.4: Elemento retirado port

6.2 ELEMENTOS MODIFICADOS

O elemento `graph` teve uma modificação em sua regra, o atributo `edgedefault`, que aceitava os valores `directed` e `undirected` passou a aceitar apenas o valor `directed`. A justificativa para isso é que todo N-Grafo tem somente arestas direcionadas.

O Código 6.5 mostra como era o elemento `graph` antes das modificações, e o Código 6.6 mostra como ficou após as modificações.

```
(...)
<xs:element name="graph">
  <xs:complexType>
  <xs:complexContent>
    <xs:attribute name="id" type="xs:integer" use="implied"/>
    <xs:attribute name="edgedefault" use="required">
      <xs:restriction base="xs:string">
      <xs:pattern value="directed|undirected"/>
      </xs:restriction>
    </xs:attribute>
  </xs:complexContent>
(...)
```

Código 6.5: Elemento a ser modificado

```
(...)
<xs:element name="graph">
  <xs:complexType>
    <xs:complexContent>
      <xs:attribute name="id" type="xs:integer" use="implied"/>
      <xs:attribute name="edgedefault" use="required">
        <xs:restriction base="xs:string">
          <xs:pattern value="directed"/>
        </xs:restriction>
      </xs:attribute>
    </xs:complexContent>
(...)
```

Código 6.6: Elemento modificado

6.3 ATRIBUTOS INCLUÍDOS

Para a representação de meta-arestas (ver Definição 4.1.8) foi incluído o atributo `meta`, que é opcional e está contido no elemento `edge`.

O Código 6.7 representa a adição do atributo `meta`.

```
<xs:element name="edge">
  <xs:complexType>
    <xs:complexContent>
      <xs:attribute name="id" type="xs:integer" use="implied"/>
      <xs:attribute name="source" type="xs:integer" use="required"/>
      <xs:attribute name="label" type="xs:string" use="required"/>
      <xs:attribute name="sourceport" type="xs:integer" use="implied"/>
      <xs:attribute name="target" type="xs:integer" use="required"/>
      <xs:attribute name="targetport" type="xs:integer" use="implied"/>
      <xs:attribute name="meta" use="implied">
        <xs:restriction base="xs:string">
          <xs:pattern value="true|false"/>
        </xs:restriction>
      </xs:attribute>
      <xs:attribute name="contour" use="implied">
        <xs:restriction base="xs:string">
          <xs:pattern value="dashed"/>
        </xs:restriction>
      </xs:attribute>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

Código 6.7: Elemento modificado

6.4 REPRESENTAÇÃO DE CARACTERES MATEMÁTICOS COM A N-GRAPHML

Um dos problemas a serem resolvidos na criação da N-GraphML foi encontrar uma maneira de representar as fórmulas de ocorrência dos vértices. Por conter símbolos lógicos matemáticos, uma representação teve que ser definida. A primeira possível solução foi a utilização de trechos da linguagem MathML (BOS et al., 2011), que por ter como objetivo a representação de fórmulas matemáticas em XML contém suporte à caracteres especiais. Essa alternativa porém foi descartada, pois a maneira de sua representação acabou por não se mostrar como a mais adequada ao problema. Por fim, a solução adotada foi a criação de entidades de referência em XML, que tomariam como base a numeração dos símbolos da Tabela Unicode (UNICODE, 2011). A tabela 5 representa os caracteres necessários para a N-GraphML.

Com a utilização de caractere do tipo Hexadecimal, acrescenta-se o seguinte símbolo: `&#`, seguido do respectivo hexadecimal (HERBORTH, 2011). Deve-se lembrar que o próprio símbolo `&` é um caractere reservado em XML, devendo ser representado com a entidade de referência `&`.

Tabela 5: caracteres e seu código Unicode

Caractere	Nome do caractere	Código Unicode
$\top$	TOP	22A4
$\bot$	BOTTOM	22A5
$\bigwedge$	AND	22C0
$\bigvee$	OR	22C1
$\rightarrow$	RIGHTWARDS ARROW	27F6
$\neg$	NOT	00AC

Fonte: (UNICODE, 2011)

6.5 REPRESENTAÇÃO DOS LINKS DOS N-GRAFOS

O Código 6.8 demonstra o elemento que foi incluído para possibilitar a representação do link aplicado na prova do N-Grafo. Tem-se um elemento *link*, com um atributo *type*, que pode assumir três valores: *Defocussing*, *Simple* ou *Focussing*, que representam respectivamente os *links* convergente, simples e divergente. Cada um dos tipos de *links* tem subatributos:

- Defocussing $\rightarrow$ source, target1 e target2
- Simple $\rightarrow$ source e target
- Focussing $\rightarrow$ source1, source2 e target.

```
<xs:element name="link">
  <xs:complexType>
    <xs:complexContent>
      <xs:attribute name="type" use="implied">
        <xs:restriction base="xs:string">
          <xs:pattern value="Defocussing|Simple|Focussing"/>
        </xs:restriction>
      </xs:attribute>

      <xs:attribute name="rule" use="implied">
        <xs:restriction base="xs:string">
          <xs:pattern value="expansion|contraction"/>
        </xs:restriction>
      </xs:attribute>

      <xs:attribute name="source" tpe="xs:integer" use="implied"/>
      <xs:attribute name="source1" type="xs:integer" use="implied"/>
      <xs:attribute name="target1" type="xs:integer" use="implied"/>
      <xs:attribute name="source2" type="xs:integer" use="implied"/>
      <xs:attribute name="target2" type="xs:integer" use="implied"/>
      <xs:attribute name="target" type="xs:integer" use="implied"/>

      <xs:attribute name="link" use="implied">
        <xs:restriction base="xs:string">
          <xs:pattern value="IC|EC|ID|ED|IN|EN|II|EI|EFBottom|EFTop|Cont|Exp|ECTop|ECBottom"/>
        </xs:restriction>
      </xs:attribute>

    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

Código 6.8: Elemento incluído

6.6 EXEMPLOS DA REPRESENTAÇÃO DE N-GRAFOS PELA N-GRAPHML

Tem-se nesta Seção exemplos de N-Grafos e as suas respectivas representações em N-GraphML.

De forma análoga à Figura 23, apresentada no Capítulo 5, na Figura 25 tem-se a hierarquia dos elementos da N-GraphML, onde destacam-se os elementos que foram removidos e modificados.

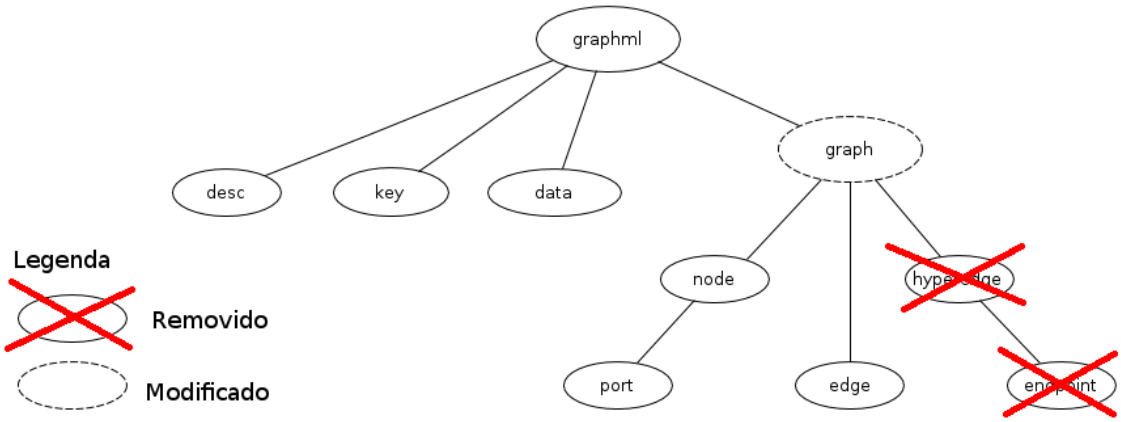

Figura 25: Hierarquia de elementos N-GraphML

Fonte: autoria própria

O Código 6.9 representa o grafo da Figura 26, pode-se se notar nesse N-Grafo exemplos de links divergentes e convergentes. Um link divergente pode-se ilustrar como o link com premissa $R \vee S$ e conclusões $R \vee S$ e $R \vee S$. Nas linhas 7 à 20 tem-se os vértices e seus rótulos (fórmulas). Nas linhas 21 à 36 encontram-se as arestas, e por último, nas linhas 38 à 47 está o código relativo aos links.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <graphml xmlns="http://graphml.graphdrawing.org/xmlns"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
5        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
6    <graph id="N" edgedefault="directed">
7      <node id="n0" label = "R &amp;#22C1; S"/>
8      <node id="n1" label = "R &amp;#22C1; S"/>
9      <node id="n2" label = "R &amp;#22C1; S"/>
10     <node id="n3" label = "R"/>
11     <node id="n4" label = "S"/>
12     <node id="n5" label = "R"/>
13     <node id="n6" label = "S"/>
14     <node id="n7" label = "R &amp;#22C1; S"/>
```

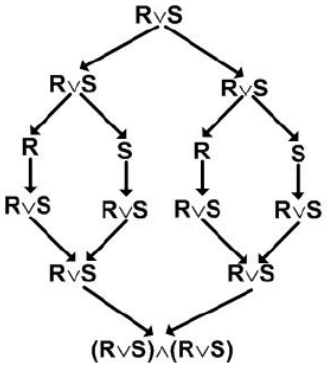

Figura 26: N-Grafo 01

Fonte: (ALVES, 2009)

```
15      <node id="n8" label = "R &amp;#22C1; S"/>
16      <node id="n9" label = "R &amp;#22C1; S"/>
17      <node id="n10" label = "R &amp;#22C1; S"/>
18      <node id="n11" label = "R &amp;#22C1; S"/>
19      <node id="n12" label = "R &amp;#22C1; S"/>
20      <node id="n13" label = "(R &amp;#22C1; S) &amp;#22C0; (R &amp;#22C1; S)"/>
21      <edge source="n0" target="n1" />
22      <edge source="n0" target="n2"/>
23      <edge source="n1" target="n3"/>
24      <edge source="n1" target="n4"/>
25      <edge source="n2" target="n5"/>
26      <edge source="n2" target="n6"/>
27      <edge source="n3" target="n7"/>
28      <edge source="n4" target="n8"/>
29      <edge source="n5" target="n9"/>
30      <edge source="n6" target="n10"/>
31      <edge source="n7" target="n11"/>
32      <edge source="n8" target="n11"/>
33      <edge source="n9" target="n12"/>
34      <edge source="n10" target="n12"/>
35      <edge source="n11" target="n13"/>
36      <edge source="n12" target="n13"/>
37      <!— Code relative to the link !—>
38      <link type="Defocussing" rule="expansion"source="n0" target1="n1" target2="n2"/>
39      <link type="Defocussing" source="n1" target1="n4" target2="n5"/>
40      <link type="Defocussing" source="" target1="n6" target2="n7"/>
41      <link type="Simple" source="n4" target="n8"/>
42      <link type="Simple" source="n5" target="n9"/>
43      <link type="Simple" source="n6" target="n10"/>
44      <link type="Simple" source="n7" target="n11"/>
45      <link type="Focussing" source1="n8" source2="n9" target="n12"/>
46      <link type="Focussing" source1="n10" source2="n11" target="n13"/>
47      <link type="Focussing" source1="n12" source2="n13" target="n14"/>
48    </graph>
49  </graphml>
```

Código 6.9: N-Grafo 01

O Código 6.10 representa o grafo da Figura 27,Nas linhas 7 à 18 tem-se os vértices e seus rótulos (fórmulas). Nas linhas 19 à 30 encontram-se as arestas, e por último, nas linhas 31 à 39 está o código relativo aos links.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <graphml xmlns="http://graphml.graphdrawing.org/xmlns"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

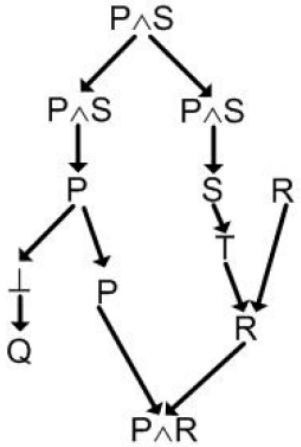

Figura 27: N-Grafo 02

Fonte: (ALVES, 2009)

```
4     xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
5       http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
6     <graph id="N" edgedefault="directed">
7       <node id="n0" label = "P &amp;#22C0; S"/>
8       <node id="n1" label = "P &amp;#22C0; S"/>
9       <node id="n2" label = "P &amp;#22C0; S"/>
10      <node id="n3" label = "P"/>
11      <node id="n4" label = "S"/>
12      <node id="n5" label = "R"/>
13      <node id="n6" label = "&amp;#22A5;"/>
14      <node id="n7" label = "P"/>
15      <node id="n8" label = "&amp;#22A4;"/>
16      <node id="n9" label = "Q"/>
17      <node id="n10" label = "R"/>
18      <node id="n11" label = "P &amp;#22C0; R"/>
19      <edge source="n0" target="n1" />
20      <edge source="n0" target="n2"/>
21      <edge source="n1" target="n3"/>
22      <edge source="n2" target="n4"/>
23      <edge source="n3" target="n6"/>
24      <edge source="n3" target="n7"/>
25      <edge source="n4" target="n8"/>
26      <edge source="n5" target="n10"/>
27      <edge source="n6" target="n9"/>
28      <edge source="n7" target="n11"/>
29      <edge source="n8" target="n10"/>
30      <edge source="n10" target="n11"/>
31      <!— Code relative to the link !—>
32      <link type="Defocussing" source="n0" target1="n1" target2="n2"/>
33      <link type="Defocussing" source="n0" target1="n1" target2="n2"/>
34      <link type="Simple" source="n4" target="n8"/>
35      <link type="Simple" source="n4" target="n8"/>
36      <link type="Simple" source="n4" target="n8"/>
37      <link type="Simple" source="n4" target="n8"/>
38      <link type="Focussing" source1="n8" source2="n9" target="n12"/>
39      <link type="Focussing" source1="n8" source2="n9" target="n12"/>
40    </graph>
41  </graphml>
```

Código 6.10: N-Grafo 02

O Código 6.11 representa o grafo da Figura 28. Nas linhas 7 à 18 tem-se os vértices e seus rótulos (fórmulas). Nas linhas 19 à 30 encontram-se as arestas, e por último, nas linhas 32

à 40 está o código relativo aos links. Nota-se que na linha 31 está a notação para a meta-aresta.

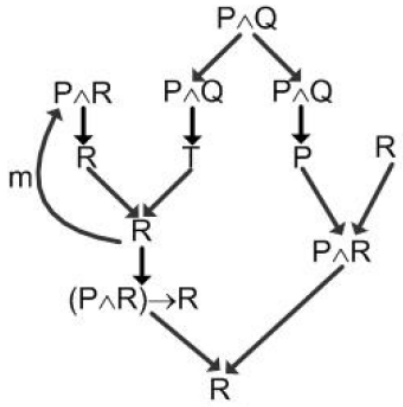

Figura 28: N-Grafo 03

Fonte: (ALVES, 2009)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <graphml xmlns="http://graphml.graphdrawing.org/xmlns"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
5        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
6    <graph id="N" edgedefault="directed">
7      <node id="n0" label = "P &amp;#22C0 Q"/>
8      <node id="n1" label = "P &amp;#22C0 R"/>
9      <node id="n2" label = "P &amp;#22C0 Q"/>
10     <node id="n3" label = "P &amp;#22C0 Q"/>
11     <node id="n4" label = "R"/>
12     <node id="n5" label = "&amp;#22A4"/>
13     <node id="n6" label = "P"/>
14     <node id="n7" label = "R"/>
15     <node id="n8" label = "R"/>
16     <node id="n9" label = "P &amp;#22C0 R"/>
17     <node id="n10" label = "(P &amp;#22C0 R) &amp;#22F6 R"/>
18     <node id="n11" label = "R"/>
19     <edge source="n0" target="n2"/>
20     <edge source="n0" target="n3"/>
21     <edge source="n1" target="n4"/>
22     <edge source="n2" target="n5"/>
23     <edge source="n3" target="n6"/>
24     <edge source="n4" target="n8"/>
25     <edge source="n5" target="n8"/>
26     <edge source="n6" target="n9"/>
27     <edge source="n7" target="n9"/>
28     <edge source="n8" target="n10"/>
29     <edge source="n10" target="n11"/>
30     <edge source="n9" target="n11"/>
31     <edge source="n8" target="n1" meta="true" contour="dashed"/>
32     <!-- Code relative to the link !-->
33     <link type="Defocussing" source="n0" target1="n2" target2="n3"/>
34     <link type="Simple" source="n1" target="n4"/>
35     <link type="Simple" source="n2" target="n5"/>
36     <link type="Simple" source="n3" target="n6"/>
37     <link type="Simple" source="n8" target="n10"/>
38     <link type="Focussing" source1="n4" source2="n5" target="n8"/>
39     <link type="Focussing" source1="n6" source2="n7" target="n9"/>
40     <link type="Focussing" source1="n9" source2="n10" target="n11"/>
41   </graph>
42 </graphml>
```

Código 6.11: N-Grafo 03

O Código 6.12 representa o grafo da Figura 29. Nas linhas 7 à 15 tem-se os vértices e seus rótulos (fórmulas). Nas linhas 16 à 23 encontram-se as arestas, e por último, nas linhas 25 à 29 está o código relativo aos links.

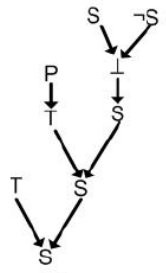

Figura 29: N-Grafo 04

Fonte: (ALVES, 2009)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <graphml xmlns="http://graphml.graphdrawing.org/xmlns"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
5        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
6    <graph id="N" edgedefault="directed">
7      <node id="n0" label = "S"/>
8      <node id="n1" label = "&amp;#00ACS"/>
9      <node id="n2" label = "P"/>
10     <node id="n3" label = "&amp;#22A5"/>
11     <node id="n4" label = "&amp;#22A4"/>
12     <node id="n5" label = "S"/>
13     <node id="n6" label = "&amp;#22A4"/>
14     <node id="n7" label = "S"/>
15     <node id="n8" label = "S"/>
16     <edge source="n0" target="n3"/>
17     <edge source="n1" target="n3"/>
18     <edge source="n2" target="n4"/>
19     <edge source="n3" target="n5"/>
20     <edge source="n4" target="n6"/>
21     <edge source="n5" target="n7"/>
22     <edge source="n6" target="n8"/>
23     <edge source="n7" target="n8"/>
24     <!-- Code relative to the link !-->
25     <link type="Simple" source="n2" target="n4"/>
26     <link type="Simple" source="n3" target="n5"/>
27     <link type="Focussing" source1="n0" source2="n1" target="n3"/>
28     <link type="Focussing" source1="n4" source2="n5" target="n7"/>
29     <link type="Focussing" source1="n6" source2="n7" target="n8"/>
30   </graph>
31 </graphml>
```

Código 6.12: N-Grafo 04

O Código 6.13 representa o grafo da Figura 30.Nas linhas 7 à 17 tem-se os vértices e seus rótulos (fórmulas). Nas linhas 18 à 30 encontram-se as arestas, e por último, nas linhas 32 à 38 está o código relativo aos links.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <graphml xmlns="http://graphml.graphdrawing.org/xmlns"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
```

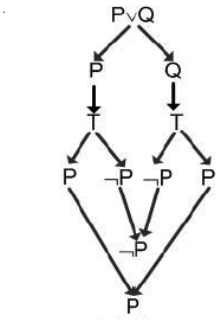

Figura 30: N-Grafo 05

Fonte: (ALVES, 2009)

```
5       http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
6     <graph id="N" edgedefault="directed">
7       <node id="n0" label = "P &amp;#22C1; Q"/>
8       <node id="n1" label = "P"/>
9       <node id="n2" label = "Q"/>
10      <node id="n3" label = "&amp;#22A4"/>
11      <node id="n4" label = "&amp;#22A4"/>
12      <node id="n5" label = "P"/>
13      <node id="n6" label = "&amp;#00AC P"/>
14      <node id="n7" label = "&amp;#00AC P"/>
15      <node id="n8" label = "P"/>
16      <node id="n9" label = "&amp;#00AC P"/>
17      <node id="n10" label = "P"/>
18      <edge source="n0" target="n1"/>
19      <edge source="n0" target="n2"/>
20      <edge source="n1" target="n3"/>
21      <edge source="n2" target="n4"/>
22      <edge source="n3" target="n5"/>
23      <edge source="n3" target="n6"/>
24      <edge source="n3" target="n7"/>
25      <edge source="n4" target="n7"/>
26      <edge source="n4" target="n8"/>
27      <edge source="n5" target="n10"/>
28      <edge source="n6" target="n9"/>
29      <edge source="n7" target="n9"/>
30      <edge source="n8" target="n10"/>
31      <!— Code relative to the link !—>
32      <link type="Defocussing" source="n0" target1="n1" target2="n2"/>
33      <link type="Defocussing" source="n3" target1="n5" target2="n6"/>
34      <link type="Defocussing" source="n4" target1="n7" target2="n8"/>
35      <link type="Simple" source="n1" target="n3"/>
36      <link type="Simple" source="n2" target="n4"/>
37      <link type="Focussing" source1="n6" source2="n7" target="n9"/>
38      <link type="Focussing" source1="n5" source2="n8" target="n10"/>
39    </graph>
40  </graphml>
```

Código 6.13: N-Grafo 05

6.7 RESUMO

Para a representação dos N-Grafos foi utilizado como base o *Schema* GraphML. Abaixo estão listadas as modificações necessárias para a elaboração do *Schema* da N-GraphML:

- Definição de uma representação para caracteres matemáticos;
- Definição de elementos a serem retirados do *Schema* da GraphML;
- Definição de elementos a serem modificados no *Schema* da GraphML;
- Definição dos atributos a serem incluídos no *Schema* da GraphML;
- Definição de uma representação para indicar a regra (ou link) aplicado na prova dos N-Grafos.

7 CONCLUSÃO

Com a criação dos N-Grafos, provas da lógica proposicional representadas através de grafos-de-prova poderão ser mais facilmente empregadas em aplicações computacionais. Uma das principais aplicações que terão como ser implementadas a partir deste trabalho são provadores automáticos de teoremas, que se valerão da contribuições dos N-Grafos como diferencial em relação aos já existentes. Portanto, o objetivo geral deste trabalho, consistiu da definição de um *Schema* XML para a representação computacional de Grafos-de-Prova. Esse *Schema* define a linguagem denominada N-GraphML.

Como resultado da convergência dos estudos de teoria dos grafos, lógica, dedução natural e das linguagens XML e GraphML tem-se a primeira versão da N-GraphML. Trata-se de um *Schema* XML, baseado no *Schema* da linguagem de marcação GraphML, onde foram feitas algumas alterações para oferecer suporte à representação dos N-Grafos. Na Figura 31 tem-se uma ilustração da interseção entre GraphML e a N-GraphML. Estão listados abaixo os principais elementos retirados ou modificados e atributos adicionados.

- Elementos retirados
  - hyperedge
  - port
- Elementos modificados
  - graph
- Atributos incluídos
  - meta
  - link
  - rule

O *Schema* está disponibilizado em:

`http://ngraphml.wikinet.org`

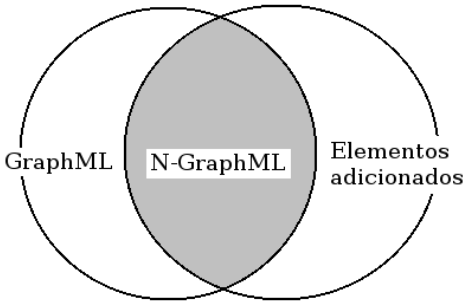

Figura 31: GraphML x N-GraphML

Fonte: Autoria Própria

7.1 CONTRIBUIÇÕES DO TRABALHO

Como principais contribuições do trabalho pode-se enunciar:

- Possibilidade de definir uma melhor representação para os N-Grafos utilizada na ferramenta desenvolvida em (KASPCZAK A. E RODRIGUES, 2011);
- Viabilizar a construção de um editor para os N-Grafos, que está sendo desenvolvido no do Grupo de Métodos Formais e Fundamentos da Computação (GM2FC), da UTFPR, através da linguagem que servirá como formato de entrada e saída para o mesmo;
- Facilitar a utilização e aplicação computacional dos N-Grafos, através de uma representação própria, que se relaciona com diferentes plataformas e aplicações.

7.2 TRABALHOS FUTUROS

Como trabalhos futuros pode-se enunciar:

- Eventuais correções ou melhor adequação nas soluções propostas pelo *Schema* da N-GraphML;
- Implementação de provadores automatizados de teoremas que façam uso do *Schema*;
- Possibilidade de utilizar os N-Grafos para o ensino de lógica computacional;

- Expansão do editor para abrir, fazer alterações e gerar uma vizualização para os N-Grafos, utilizando-se de uma biblioteca gráfica, possivelmente a Jgraph (BENSON; ALDER, 2012);
- Outras implementações computacionais que possam se utilizar dos N-Grafos.

REFERÊNCIAS

ALVES, G. V. **Implementação de Estruturas de Controle para Estratégias de Prova em um Provador de Teoremas**. Trabalho de Conclusão de Curso - Universidade Católica de Pelotas - Rio Grande do Sul, 2002.

ALVES, G. V. **Aspectos introdutórios da topologia de grafos**. [S.l.]: Trabalho Individual. Universidade Federal de Pernambuco, 2006.

ALVES, G. V. **Transformations for proof-graphs with cycle treatment augmented via geometric perspective techniques**. Tese (Doutorado) — Universidade Federal de Pernambuco, 2009.

ALVES, G. V. **Notas de Aulas - Lógica Matemática - Aula 02 - Lógica Proposicional (tabelas-verdade)**. 2011.

ALVES, G. V. **Notas de Aulas - Lógica Matemática - Aula 03 - Lógica Proposicional (propriedades e fbf)**. 2011.

BENSON, D.; ALDER, G. **Jgraph - Connecting the dots**. 2012. Disponível em: <http://www.jgraph.com/>. Acesso em: 04 de maio 2012.

BILGIN, A.; CALDWELL, D. et al. **Graphviz - Graph Visualization Software**. 2011. Disponível em: <http://www.graphviz.org//>. Acesso em: 28 de outubro de 2011.

BOS, B. et al. **W3C Math Home**. 2011. Disponível em: <http://www.w3.org/Math/>. Acesso em: 17 de setembro de 2011.

BRANDES, U.; EIGLSPERGER, M. et al. **The GraphML File Format**. 2011. Disponível em: <graphml.graphdrawing.org>. Acesso em: 28 de outubro de 2011.

FONSECA, P. **BibTeXpress - BibTeX Entry Builder (Version 1.0)**. 2012. Disponível em: <http://www.cin.ufpe.br/ paguso/bibtexpress/>. Acesso em: 16 de maio 2012.

FURTADO, A. L. **Teoria dos Grafos: Algoritmos**. [S.l.]: Livros Técnicos e Científicos Editora S.A., 1973.

GERSTING, J. L. **Fundamentos matemáticos para a ciência da computação**. Quarta edição. [S.l.]: LTC, 2001.

GIRARD, J.-Y. **Proofs and Types**. [S.l.]: Press Syndicate of the University of Cambridge, 1989.

HARARY, F. **Graph Theory**. [S.l.]: Addison-Wesley, 1969.

HERBORTH, C. **IBM - Developer Works**. 2011. Disponível em: <http://www.ibm.com/developerworks/xml/library/x-entities/>. Acesso em: 30 de outubro 2011.

HOPCROFT, J. E.; ULLMAN, J. D.; MOTWANI, R. **Introdução à Teoria de Autômatos, Linguagens e Computação**. Segunda edição. [S.l.]: Addison-Wesley, 1939.

KASPCZAK A. E RODRIGUES, L. G. 2011. **Implementação de um algoritmo para verificação de ciclos em grafos-de-prova**. Trabalho de Conclusão de Curso - Universidade Tecnológica Federal do Paraná - Campus Ponta Grossa.

KILE. **Kile - an Integrated LaTeX Environment**. 2011. Disponível em: <http://kile.sourceforge.net/>. Acesso em: 28 de outubro de 2011.

OLIVEIRA, A. G. de. **Proofs from a Geometric Perspective**. Tese (Doutorado) — Centro de Informática - Universidade Federal de Pernambuco, abr. 2001.

SCHROEDER, B. **A Graph Based Theorem Proving Platform with Strategies**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, set. 2008.

UNICODE. **The Unicode Consortium - Unicode 6.0 Character Code Charts**. 2011. Disponível em: <http://www.unicode.org/charts/#symbols>. Acesso em: 30 de outubro 2011.

W3SCHOOLS. **DTD Tutorial**. 2011. Disponível em: <http://www.w3schools.com/dtd/default.asp>. Acesso em: 28 de outubro de 2011.

W3SCHOOLS. **XML Schema Tutorial**. 2011. Disponível em: <http://www.w3schools.com/schema/default.asp>. Acesso em: 28 de outubro de 2011.

W3SCHOOLS. **XML Tutorial**. 2011. Disponível em: <http://www.w3schools.com/xml/default.asp>. Acesso em: 28 de outubro de 2011.

YWORKS. **yEd Graph Editor**. 2011. Disponível em: <http://www.yworks.com/en/products_yed_about.html>. Acesso em: 28 de outubro de 2011.