Pontifícia Universidade Católica
do Rio de Janeiro

José Flávio Cavalcante Barros Júnior

Uma Abordagem Experimental sobre a Compressão de Provas em Dedução Natural Minimal Implicacional

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática da PUC-Rio.

Orientador: Prof. Edward Hermann Haeusler

Rio de Janeiro
Abril de 2019

PUC-Rio - Certificação Digital Nº 1712666/CA

Pontifícia Universidade Católica
do Rio de Janeiro

José Flávio Cavalcante Barros Júnior

Uma Abordagem Experimental sobre a Compressão de Provas em Dedução Natural Minimal Implicacional

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo.

Prof. Edward Hermann Haeusler
Orientador
Departamento de Informática – PUC-Rio

Prof. Luiz Carlos Pinheiro Dias Pereira
Departamento de Filosofia – PUC-Rio

Prof. Bruno Lopes Vieira
Universidade Federal Fluminense – UFF

Rio de Janeiro, 26 de Abril de 2019

PUC-Rio - Certificação Digital Nº 1712666/CA

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

José Flávio Cavalcante Barros Júnior

Graduou-se Bacharel em Sistemas de Informação pela Universidade Federal do Ceará no Campus de Quixadá em 2016.

PUC-Rio - Certificação Digital Nº 1712666/CA

Ficha Catalográfica

Barros Júnior, José Flávio Cavalcante

Uma Abordagem Experimental sobre a Compressão de Provas em Dedução Natural Minimal Implicacional / José Flávio Cavalcante Barros Júnior; orientador: Edward Hermann Haeusler. – Rio de janeiro: PUC-Rio, Departamento de Informática , 2019.

62 f: il. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática .

Inclui bibliografia

1. Informática – Teses. 2. Teoria da Prova. 3. Compressão de Provas. 4. Lógica Minimal. I. Haeusler, Edward Hermann. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática . III. Título.

CDD: 004

PUC-Rio - Certificação Digital Nº 1712666/CA

Aos meus pais.

Agradecimentos

À minha família, que me apoiou desde o início na decisão de cursar esse mestrado. Toda a ajuda de vocês foi determinante no sucesso dessa jornada, nunca vou poder retribuir o que fizeram por mim nesse momento da minha vida.

Ao meu orientador, professor Edward Hermann Haeusler, pela disponibilidade, sugestões, comentários, atenção e compreensão durante todo o mestrado. Por todas as conversas descontraídas, que sempre ajudaram a tranquilizar nos momentos difíceis. Espero que nossos times de futebol tenham mais sucesso no futuro!

Aos membros da bancas das defesas da proposta de dissertação e da dissertação, Bruno Lopes e Luiz Carlos Pereira. Obrigado por todos os comentários, dicas e sugestões que elevaram a qualidade do trabalho.

Aos meus amigos, por todas as conversas e conselhos que sempre renovaram minhas forças nos bons e nos maus momentos.

Aos meus colegas do TecMF que contribuíram direta ou indiretamente no trabalho.

Ao CNPq pelo apoio financeiro à essa pesquisa.

PUC-Rio - Certificação Digital Nº 1712666/CA

Resumo

Barros Júnior, José Flávio Cavalcante; Haeusler, Edward Hermann (Orientador). **Uma Abordagem Experimental sobre a Compressão de Provas em Dedução Natural Minimal Implicacional**. Rio de Janeiro, 2019. 62p. Dissertação de Mestrado – Departamento de Informática , Pontifícia Universidade Católica do Rio de Janeiro.

O tamanho das provas formais possui algumas importantes implicações teóricas na área da complexidade computacional. O problema de determinar se uma fórmula é uma tautologia da Lógica Proposicional Intuicionista e do fragmento puramente implicacional da Lógica Minimal (M$\supset$) é PSPACE-Completo. Qualquer lógica proposicional com um sistema de dedução natural que satisfaça o princípio da subfórmula possui o problema de determinar tautologias em PSPACE. Saber se qualquer tautologia em M$\supset$ admite provas de tamanho polinomialmente limitado está relacionado com saber se NP = PSPACE. Técnicas de compressão de provas reportadas na literatura utilizam duas abordagens principais para comprimir provas: gerar provas já compactadas; comprimir uma prova já gerada. Proposta por Gordeev e Haeusler (6), a *Compressão Horizontal* é uma técnica de compressão de provas em dedução natural da M$\supset$ que utiliza grafos direcionados para representar as provas. Dada a prova de uma tautologia qualquer da M$\supset$, que pode possuir tamanho exponencial em relação ao tamanho da conclusão, o objetivo da Compressão Horizontal é que a prova resultante da compressão possua tamanho polinomialmente limitado em relação ao tamanho da conclusão. Nosso trabalho apresenta a primeira implementação da Compressão Horizontal, juntamente com os primeiros resultados da aplicação da técnica sobre provas de tautologias da M$\supset$, além disso, compara as taxas de compressão obtidas com técnicas tradicionais de compressão de dados.

Palavras-chave

Teoria da Prova; Compressão de Provas; Lógica Minimal.

PUC-Rio - Certificação Digital Nº 1712666/CA

Abstract

Barros Júnior, José Flávio Cavalcante; Haeusler, Edward Hermann (Advisor). **An Experimental Approach on Minimal Implicational Natural Deduction Proofs Compression**. Rio de Janeiro, 2019. 62p. MSc Dissertation – Departamento de Informática , Pontifícia Universidade Católica do Rio de Janeiro.

The size of formal proofs has some important theoretical implications in computational complexity theory. The problem of determining if some formula of Intuitionistic Propositional Logic and the purely implicational fragment of Minimal Logic (M$\supset$) is a tautology is PSPACE-Complete. Any propositional logic with a natural deduction system that satisfies the subformula principle is PSPACE. To know if any tautology in M$\supset$ admits polynomially sized proof is related to whether NP = PSPACE. Proof compression techniques reported in literature use two main approaches to proof compressing: generating already compressed proofs; compressing an already generated proof. Proposed by Gordeev and Haeusler (6), the Horizontal Compression is a natural deduction proof compression technique that uses directed graphs to represent proofs. Given a tautology proof in M$\supset$, which may have an exponential size in relation to conclusion length, the goal of Horizontal Compression is that the compressed proof has a polynomially limited size in relation to conclusion length. Our work presents the first implementation of Horizontal Compression, together with the first results of the execution of the technique on proofs of M$\supset$ tautologies.

PUC-Rio - Certificação Digital Nº 1712666/CA

Keywords

Proof Theory; Proof Compression; Minimal Logic.

Sumário

PUC-Rio - Certificação Digital Nº 1712666/CA

Lista de figuras

PUC-Rio - Certificação Digital Nº 1712666/CA

Lista de tabelas

PUC-Rio - Certificação Digital Nº 1712666/CA

PUC-Rio - Certificação Digital Nº 1712666/CA

O trabalho da ciência não consiste na criação, mas na descoberta de pensamentos verdadeiros.

Gottlob Frege, *primeiro capítulo do livro não finalizado, The Thought: A Logical Inquiry (1918).*

1
Introdução

Provas matemáticas existem desde a Grécia antiga e têm o objetivo de corroborar a veracidade de uma afirmação, além de transmitir conhecimento entre especialistas através do tempo. As provas matemáticas utilizadas até o final do século XIX, conhecidas como provas informais, são geralmente construídas em linguagem natural e não possuem uma forma precisa, sendo estruturadas de acordo com a vontade do autor. Apesar de serem bastante utilizadas, as provas informais têm um importante problema: inexiste um método efetivo de verificação de erros. Entre o final do século XIX e o início do século XX, alguns matemáticos e lógicos abordaram diferentes maneiras para fundamentar a matemática através de um aparato teórico que permitisse uma checagem de erros. Entre as várias contribuições para a lógica matemática, surgiu o ramo da *teoria da prova* juntamente com o conceito de prova formal.

Ao contrário das provas informais, as provas formais são estruturadas através de rigorosas regras de construção. Uma prova formal é a derivação de uma sentença (conclusão) a partir de outras sentenças (axiomas e hipóteses), tal derivação é uma sequência de sentenças, onde cada elemento da sequência é uma hipótese ou axioma, ou é resultado da aplicação de uma regra de inferência de um sistema dedutivo. Cada sentença de uma prova formal, também chamada de fórmula, é um elemento de uma linguagem formal, linguagens da Lógica Proposicional e da Lógica de Primeira Ordem são exemplos de linguagens formais.

Além da utilização puramente teórica na lógica matemática, provas formais podem ser utilizadas para diversas finalidades em diferentes contextos. No desenvolvimento de *softwares*, provas podem ser necessárias para validar o funcionamento de porções de código. Na fabricação de *hardwares*, um projeto de circuito pode ser validado através da prova de uma fórmula que o descreve. Em alguns casos, tais provas podem ser grandes, possuindo tamanhos exponenciais em relação ao tamanho da conclusão, o que aumenta significativamente a complexidade da construção e demanda uma automação.

A Prova Automática de Teoremas (*Automated Theorem Proving* - ATP) é a área da Ciência da Computação que utiliza programas de computador para a geração automatizada ou semiautomatizada de provas. No entanto, um pro-

PUC-Rio - Certificação Digital Nº 1712666/CA

vador de teoremas ainda pode gerar provas muito grandes. O tamanho de uma prova pode prejudicar sua utilização prática, visto que a extração de alguma informação útil ao contexto pode se tornar inviável, além de que manipular grandes volumes de dados pode acarretar problemas de implementação para o provador. Esses problemas reforçam a importância do esforço para compressão das provas geradas e/ou na geração de provas já comprimidas.

Além de possíveis problemas de implementação nos provadores, o tamanho das provas possui algumas importantes implicações teóricas na área da complexidade computacional. Statman mostrou que o problema de determinar se uma fórmula é uma tautologia ($TAUT$) da Lógica Proposicional Intuicionista e do fragmento puramente implicacional da Lógica Minimal (M$\supset$) é PSPACE-Completo (1). M$\supset$ é capaz de simular a Lógica Proposicional Intuicionista através de uma tradução polinomial. Qualquer lógica proposicional com um sistema de dedução natural que satisfaça o princípio da subfórmula também possui seu respectivo $TAUT$ em $PSPACE$ (2). Um sistema de dedução natural que satisfaz o princípio da subfórmula é capaz de gerar provas onde cada ocorrência de fórmula é uma subfórmula da conclusão ou é uma subfórmula de alguma hipótese. Saber se $TAUT$ da M$\supset$ possui um certificado polinomial para qualquer tautologia, i.e, saber se qualquer tautologia possui uma prova de tamanho polinomialmente limitado em relação ao tamanho da conclusão, está relacionado com saber se $NP = PSPACE$.

Provas em dedução natural podem ser representadas em diferentes formatos. No estilo de Gentzen-Prawitz, as provas possuem o formato de uma árvore, onde as fórmulas são representadas pelos nós, e as regras e os rótulos de descarte são representados pelas arestas. No estilo de Jaśkowski-Fitch, as provas são sequências de passos numerados seguidos pela identificação da regra e sua referida justificativa. O tamanho de uma prova pode ser aferido a partir de diferentes pontos de vista. A quantidade de nós (Gentzen-Prawitz), a quantidade de linhas (Jaśkowski-Fitch) e até a quantidade de símbolos podem ser utilizados para mensurar o tamanho de uma prova. Para a complexidade computacional, o tamanho de uma prova é a quantidade de símbolos de sua representação.

É bem conhecido que provas podem ser muito grandes. Na M$\supset$, algumas fórmulas possuem provas em dedução natural com tamanhos com limite inferior exponencial (3). A redução do tamanho de provas pode ser realizada, principalmente, através de duas abordagens: gerar provas já comprimidas através de um cálculo de dedução natural que seja capaz de gerar provas menores que a dedução natural usual; comprimir provas em dedução natural que já tenham sido geradas.

PUC-Rio - Certificação Digital Nº 1712666/CA

Em (4, 5), Paleo propõe um cálculo de dedução natural para a M⊃ utilizando *deep inference*, que permite aplicar as regras de inferência diretamente nas subfórmulas. Para algumas fórmulas, esse cálculo pode gerar provas quadraticamente menores que a dedução natural usual.

Em (6), Gordeev e Haeusler propõem o método de Compressão Horizontal que permite reduzir o tamanho das provas através da fusão de nós com fórmulas idênticas que estejam no mesmo nível na árvore de derivação (estilo Gentzen-Prawitz). Essa técnica de compressão é utilizada como uma ferramenta para a prova da conjectura $NP = PSPACE$. O objetivo da compressão é que a prova compactada possua tamanho polinomialmente limitado em relação ao tamanho da conclusão.

A Compressão Horizontal e outras técnicas de compressão de provas para outros sistemas dedutivos, (7, 8, 9), utilizam a redundância de dados nas representações para obter a redução do espaço necessário para representar as provas. Essa característica também é a base para inúmeras técnicas tradicionais de compressão de dados, e.g., codificação de Huffman.

Esta dissertação de mestrado se propõe a realizar um estudo comparativo empírico entre técnicas de compressão de provas na M⊃ em dedução natural. A revisão da literatura identificou apenas duas técnicas de compressão com essa característica (dedução natural da M⊃). A primeira é a Dedução Natural Contextual (DNc) proposta em 2013 (4) com experimentos reportados em 2015 (5), que demonstraram que a técnica é capaz de gerar provas menores que a dedução natural usual em alguns casos. A segunda é a compressão horizontal de provas proposta em 2016 (6) para comprimir provas na M⊃ de tautologias arbitrárias com garantia que a compactação gera provas de tamanho polinomialmente limitado em relação ao tamanho da conclusão.

Apresentamos a primeira implementação da Compressão Horizontal juntamente com os resultados da aplicação das técnicas de compressão de provas e dados para um conjunto de provas na M⊃.

No Capítulo 2, introduzimos os principais conceitos relativos a provas formais utilizados no restante do trabalho. Iniciamos com um pequeno resumo histórico sobre o surgimento das provas formais. Apresentamos o fragmento puramente implicacional da lógica minimal, detalhando sua linguagem, sistema de dedução natural e semântica. Mostramos como uma prova em dedução natural de uma mesma fórmula pode possuir diferentes representações

No Capítulo 3, apresentamos em detalhes a Compressão Horizontal e a a codificação de Huffman. Exemplificamos como a Compressão Horizontal atua na compressão das provas no Capítulo 4 e detalhamos passo a passo a compressão de uma prova. No Capítulo 5, apresentamos o compressor de

PUC-Rio - Certificação Digital Nº 1712666/CA

provas *compressing*, que implementa a Compressão Horizontal. Detalhamos os principais aspectos do projeto e justificamos algumas decisões de projeto na implementação do compressor.

O Capítulo 6 apresenta os resultados dos experimentos dos algoritmos da Compressão Horizontal e codificação de Huffman, detalhando a definição do conjunto de provas utilizadas e os *softwares* utilizados direta ou indiretamente na experimentação. Concluímos o trabalho no Capítulo 7, no qual avaliamos os objetivos inicialmente estabelecidos e os resultados obtidos, destacamos as contribuições do trabalho e exploramos possibilidades de trabalhos futuros.

PUC-Rio - Certificação Digital Nº 1712666/CA

2
Provas Formais

Este capítulo apresenta os principais conceitos relacionados às provas formais utilizados no trabalho. A Seção 2.1 apresenta um resumo histórico do surgimento das provas formais desde Frege até a criação da dedução natural por Gentzen. A Seção 2.2 apresenta a lógica proposicional minimal, detalhando sua linguagem, sistema dedutivo de dedução natural e semântica. A Seção 2.3 apresenta os dois principais estilos de representação de provas em dedução natural propostos por Gentzen e Jaśkowski.

2.1
Provas Formais *vs* Provas Informais

Segundo o dicionário *Michaelis* da língua portuguesa, uma *prova* é "aquilo que demonstra a veracidade de uma afirmação ou de um fato; confirmação, comprovação, evidência". Na matemática, as provas são utilizadas para certificar e comunicar o conhecimento entre especialistas. Provas matemáticas existem desde a Grécia antiga, no entanto, sua forma e organização mudaram bastante ao longo do tempo.

O conceito de prova formal foi gradualmente construído entre o final do século XIX e o início do século XX, as provas matemáticas utilizadas antes desse período são chamadas aqui de *provas informais*. Uma prova informal é expressa em linguagem natural e, possivelmente, contém símbolos e figuras (10). Esse tipo de prova tem o objetivo de convencer o leitor que a proposição matemática em questão é verdadeira ou falsa através da exposição de uma sequência de argumentos encadeados. Normalmente, para facilitar a compreensão, algumas informações básicas e passos óbvios de raciocínio não são adicionados à prova, o que pode deixar algumas lacunas na argumentação. No restante do texto, o termo "prova" será sempre utilizado em referência à prova formal.

No final do século XIX iniciou-se um movimento entre alguns matemáticos, conhecido como *logicismo*, com o objetivo de solidificar os fundamentos da matemática através da lógica. A forma como o conhecimento matemático era construído e repassado já estava bem estabelecida, e até então havia se mostrada eficiente através da prática matemática. No entanto, as provas in-

PUC-Rio - Certificação Digital Nº 1712666/CA

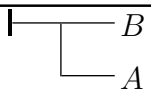
formais eram passíveis a erros, que poderiam ser propagados caso não fossem identificados. Uma solução desejável para esse problema seria um método para construir e especificar provas que, por definição, admita um processo de checagem mecânica (11).

Uma das primeiras e mais importantes contribuições ao logicismo foi realizada pelo matemático, lógico e filósofo alemão Gottlob Frege (1848 - 1925), que em 1879 publicou o livro *Begriffsschrift — escrita conceitual, conceitografia* — considerado um dos principais precursores da lógica moderna. Com o objetivo de fundamentar a aritmética por meios puramente lógicos, Frege percebeu que a linguagem natural não era adequada, pois apresentava importantes imperfeições que a limitavam na tarefa de expressar conceitos matemáticos com exatidão e clareza (12). Esse problema, observado por Frege como inerente à linguagem natural, foi uma das principais motivações para a produção do *Begriffsschrift*, onde Frege introduz uma linguagem baseada puramente em fórmulas (a *conceitografia*), que possui suas sentenças e regras definidas de forma clara e precisa.

A conceitografia possui símbolos básicos representando a *implicação* e a *negação*, e ainda, possui representações de quantificação universal e existencial (Tabela 2.1). Considerada como a primeira linguagem formal, a conceitografia possui um sistema dedutivo composto por nove axiomas e uma regra de inferência. O sistema dedutivo (axiomático) de Frege tem como princípios que: axiomas expressam verdades lógicas básicas; outras verdades são derivadas dos axiomas através da regra de inferência *modus ponens* (13). Provas construídas construídas com termos (fórmulas) de uma linguagem formal e seguindo regras de um sistema dedutivo são chamadas de *provas formais*.

PUC-Rio - Certificação Digital Nº 1712666/CA

Tabela 2.1: Simbologia utilizada no *Begriffsschrift*.

Definição	**Símbolo**
Implicação (Se A, então B)	
Negação	
Quantificação universal	
Quantificação existencial	

Após a publicação do *Begriffsschrift*, Frege continuou se dedicando ao objetivo de formalizar a aritmética por meios puramente lógicos. Em 1893, publicou o primeiro de dois volumes do *Grundgesetze der Arithmetik — Leis Básicas da Aritmética* — onde define cinco Leis Básicas (I, II, III, IV, V) originadas a partir de refinamentos do *Begriffsschrift*, exceto pela Lei V, que

introduz a noção de *extensão de um conceito*. Nas vésperas do lançamento do segundo volume, em 1902, Frege recebeu uma carta do matemático e filósofo britânico Bertrand Russel (1872 - 1970) comunicando a descoberta de um problema com as Leis Básicas do primeiro volume. A partir da Lei V, Russel conseguiu obter uma contradição, tornando a teoria do *Grundgesetze der Arithmetik* inconsistente. Os detalhes da descoberta desse problema, que ficou conhecido como o *paradoxo de Russel*, foram publicados por Russel no livro *The Principles of Mathematics* em 1903.

Com a descoberta que a teoria desenvolvida por Frege é inconsistente, o logicismo necessitava de mais aparato teórico para atingir seu objetivo. Pelos anos seguintes Russel se dedicou a encontrar uma solução para o paradoxo, que inicialmente julgava ser simples, no entanto, só chegou a uma solução em 1908 através da Teoria dos Tipos em *Mathematical Logic as Based on the Theory of Types*. Durante esse período, Russel começou a colaborar com seu ex-professor Alfred North Whitehead (1861 - 1947). O resultado dessa colaboração foi a publicação dos três volumes do *Principia Mathematica*, respectivamente, em 1910, 1912 e 1913.

O *Principia Mathematica* é considerada a obra mais ambiciosa e importante do logicismo; Russel e Whitehead ansiavam reduzir toda a matemática à lógica. Apesar de compartilhar a mesma motivação filosófica sobre o logicismo com Frege, Russel utilizou no *Principia* e em trabalhos anteriores uma notação próxima à utilizada em 1889 pelo matemático italiano Guiseppe Peano (1858 - 1932) no *Arithmetices principia, nova methodo exposita*. Juntos, os três volumes são divididos em seis partes e abordam números reais, ordinais e cardinais, e teoria dos conjuntos. Um quarto volume foi iniciado abordando a geometria, mas não chegou a ser concluído.

A parte I, presente no primeiro volume, introduz alguns conceitos e notações referentes à lógica proposicional. O sistema formal para o fragmento proposicional do *Principia Mathematica* é baseado em dois conectivos primitivos, negação e disjunção, e a partir desses os outros três conectivos são definidos, conjunção, implicação e equivalência (Tabela 2.2). O sistema dedutivo original é composto por cinco axiomas e oito regras de inferência (14), no entanto, Paul Bernays mostrou que um dos axiomas é redundante e pode ser obtido a partir dos outros quatro (15).

Anos antes do lançamento do *Principia Mathematica*, em 1898, o matemático alemão David Hilbert (1862 - 1953) publicou o livro *Grundlagen der Geometrie* — Fundamentos da Geometria — contendo importantes avanços no desenvolvimento do método axiomático. Segundo Hilbert, a construção de um sistema axiomático possui dois pilares principais: a independência dos axiomas

PUC-Rio - Certificação Digital Nº 1712666/CA

Tabela 2.2: Conectivos lógicos utilizados no *Principia Mathematica*.

Conectivo lógico	**Sentido**	**Símbolo**	**Definição**
Negação	A é falso	$\sim$A	*primitivo*
Disjunção	A ou B	A $\vee$ B	*primitivo*
Conjunção	A e B	A.B	$\sim(\sim A \vee \sim B)$
Implicação	Se A, então B	A $\supset$ B	$\sim A \vee B$
Equivalência	A é equivalente a B	A $\equiv$ B	A $\supset$ B.B $\supset$ A

e a consistência dos axiomas, i.e, garantia de que a partir dos axiomas não é possível obter uma contradição. Pelos anos seguintes, Hilbert trabalhou para provar a consistência dos axiomas da geometria através da redução à prova da consistência da análise (16). No entanto, diversos fatores atrasaram esse desenvolvimento. A descoberta do paradoxo de Russel e críticas a um esboço da prova da consistência da análise evidenciaram a necessidade de desenvolvimento dos formalismos lógicos para os sistemas axiomáticos.

Após a publicação do *Principia*, Hilbert retomou os trabalhos relacionados a prova da consistência de sistemas axiomáticos. Em 1917, Paul Bernays inciou uma série de contribuições ao trabalho de Hilbert, essas contribuições resultaram em alguns importantes avanços na lógica formal, tais como o tratamento da lógica proposicional e da lógica de primeira ordem como sistemas axiomáticos distintos e a primeira prova da completude do cálculo proposicional do *Principia*. No entanto, muitos problemas relacionados com a axiomatização da matemática continuavam em aberto. Por volta de 1920, Hilbert iniciou um programa de pesquisa, conhecido posteriormente como o *programa de Hilbert*, com o objetivo formalizar a matemática através de um sistema axiomático que possua uma prova direta da sua consistência (17).

Em 1931, o filósofo, matemático e lógico austríaco Kurt Gödel publicou os dois *teoremas da incompletude*. O primeiro afirma que qualquer sistema axiomático capaz de expressar a aritmética elementar não pode ser simultaneamente completo e consistente. O segundo afirma que qualquer sistema dedutivo capaz de expressar a aritmética elementar é capaz de provar sua própria consistência, se e somente se for inconsistente. Esses dois teoremas mostraram que os objetivos iniciais do programa de Hilbert jamais poderiam ser alcançados.

Após os resultados da incompletude de Gödel, o matemático e lógico alemão Gerhard Gentzen (1909 - 1945) iniciou estudos para provar a consistência da aritmética. Um dos produtos dessa pesquisa foi sua tese de doutorado publicada em duas partes, em 1934 e em 1935, na qual se propunha investigar

PUC-Rio - Certificação Digital Nº 1712666/CA

como as provas matemáticas realmente ocorrem na prática. Inicialmente, Gentzen observou que as provas matemáticas não seguiam a estrutura dos sistemas axiomáticos de Hilbert. No lugar de utilizar axiomas, as provas partiam de pressupostos para provar proposições (13). Outras observações são referentes ao fato de que as conclusões das provas são obtidas em partes, por exemplo, para obter a proposição "A e B" é necessário obter A e B separadamente

$$\frac{A \qquad B}{A \;\&\; B}$$

e que os pressupostos das provas são considerados em termos de seus componentes, por exemplo, a partir do pressuposto "A e B" é possível obter A separadamente ou B separadamente.

$$\frac{A \;\&\; B}{A}$$

$$\frac{A \;\&\; B}{B}$$

Aos procedimentos de obter a conclusão por partes e considerar os pressupostos em termos de seus componentes, Gentzen nomeou, respectivamente, de *regras de introdução* e *regras de eliminação*. O resultado da investigação foi a criação do sistema dedutivo chamado de *dedução natural* (DN), que é composto por regras de introdução e eliminação para os conectivos lógicos (18).

Se em uma prova em dedução natural ocorre uma regra de introdução seguida por sua respectiva regra de eliminação é dito que a prova contém um "desvio" que pode ser eliminado. Uma prova sem "desvios" é chamada de *prova normal*. O processo de transformar uma prova não-normal em normal é chamado de *normalização*. Provas normais satisfazem o princípio da subfórmula.

Com a criação da dedução natural, o objetivo de Gentzen era utilizá-la na prova da consistência da aritmética, mas para isso era necessário provar que toda prova em dedução natural da lógica clássica possui uma forma normal (teorema da normalização). No entanto, Gentzen só obteve o teorema da normalização para a lógica intuicionista. Para resolver esse problema, Gentzen criou um outro sistema dedutivo chamado de *cálculo de sequentes* (CS), para o qual forneceu um teorema equivalente ao da normalização, o teorema da eliminação do corte. Utilizando DN e CS, Gentzen construiu quatro provas para a consistência da aritmética entre 1934 e 1939 (19).

Posteriormente, o teorema da normalização para a lógica clássica foi apresentado por Prawitz (20). Hoje, DN e CS são dois dos principais cálculos de dedução para a lógica clássica e intuicionista (proposicional e primeira ordem). Apesar de terem a origem na matemática, possuem grande influência em outras áreas, por exemplo, a computação.

PUC-Rio - Certificação Digital Nº 1712666/CA

2.2 Lógica Proposicional Minimal

A Lógica Proposicional Minimal (LPM) é obtida da Lógica Proposicional Intuicionista (LPI) através da exclusão do *ex falso quodlibet* — princípio da explosão — que afirma que qualquer proposição poder ser obtida do absurdo.

Para estabelecer um sistema formal lógico é necessário definir três componentes: sintaxe da linguagem, que define como os elementos da linguagem do sistema formal são construídos; o sistema de derivação (dedutivo), que estabelece regras para a derivação de fórmulas a partir de outras fórmulas da linguagem; a semântica formal, que atribui significado aos elementos da linguagem. Nessa Seção, apresentamos o sistema formal da LPM e definimos o seu fragmento puramente implicacional.

2.2.1 Linguagem

Para definir uma linguagem é necessário estabelecer dois componentes principais: o *alfabeto*, que contém os símbolos pelos quais os elementos da linguagem são compostos; as *regras de formação* dos elementos da linguagem. O alfabeto da linguagem da LPM ($\mathcal{LP}_{\mathcal{M}}$) possui três conjuntos disjuntos de símbolos: símbolos proposicionais, conectivos e símbolos auxiliares. Os *símbolos proposicionais* representam as proposições, i.e, afirmações sobre as quais faz sentido perguntar "é verdadeira?". Os conectivos associam elementos da linguagem para formar um novo elemento. Os símbolos auxiliares servem para organizar a estrutura interna dos elementos. O alfabeto de $\mathcal{LP}_{\mathcal{M}}$ é composto por:

- **Símbolos proposicionais**: $\{\bot, A, B, C, ..., A_1, B_1, ...\}$
- **Conectivos**: $\{\wedge, \vee, \supset, \neg\}$
- **Símbolos auxiliares**: $\{(,)\}$

Os elementos da $\mathcal{LP}_{\mathcal{M}}$, também chamados de *fórmulas bem formadas* (ou somente, *fórmulas*), são sequências de símbolos compostas por símbolos pertencentes à união desses conjuntos e devem obedecer às regras de formação da linguagem. Convencionamos representar fórmulas por letras minúsculas do alfabeto grego ($\alpha, \beta, \gamma, ...$).

Definição 2.1. Fórmula bem formada. Uma sequência de símbolos do alfabeto de $\mathcal{LP}_{\mathcal{M}}$ é uma *fórmula bem formada* se e somente se for criada a partir dessas regras:

PUC-Rio - Certificação Digital Nº 1712666/CA

1. Qualquer símbolo proposicional é uma fórmula (nesse caso, também chamado de fórmula atômica).

2. Se α é uma fórmula, '$\neg\alpha$' é uma fórmula.

3. Se α e β são fórmulas, '$(\alpha \wedge \beta)$' é uma fórmula.

4. Se α e β são fórmulas, '$(\alpha \vee \beta)$' é uma fórmula.

5. Se α e β são fórmulas, '$(\alpha \supset \beta)$' é uma fórmula.

Apenas os conectivos $\wedge$, $\vee$ e $\supset$ são considerados primitivos da linguagem, o conectivo $\neg$ em uma fórmula γ é uma abreviação para $\gamma \supset \bot$.

2.2.2 Sistema Dedutivo

Um sistema dedutivo, ou de derivação, estabelece mecanismos para a obtenção de uma fórmula a partir de um conjunto de fórmulas. Esses mecanismos são baseados em procedimentos puramente sintáticos e não dependem de quaisquer significados atribuídos às fórmulas. Um sistema dedutivo é composto por um conjunto de axiomas (possivelmente vazio) e por um conjunto de regras de inferência (não vazio).

A obtenção de uma fórmula em um sistema dedutivo é realizado através de uma derivação, também chamada de dedução. Essa derivação ocorre a partir de dois conjuntos de fórmulas, denominados de *hipóteses* e axiomas, por meio de aplicações sequenciais de regras de inferência. Uma regra de inferência permite concluir uma fórmula (conclusão) a partir de outras fórmulas (premissas). Quando uma fórmula α é derivada a partir de um conjunto de hipóteses Γ, utilizamos a seguinte notação:

$$\Gamma \vdash \alpha$$

Definição 2.2. Derivação. Seja um sistema dedutivo com um conjunto de axiomas Λ, uma *derivação* de $\Gamma \vdash \phi_k$ é composta por uma sequência de fórmulas $\langle \phi_0, \phi_1, ..., \phi_k \rangle$, onde cada ϕ_i, para $i = 0$ até k, satisfaz uma das seguintes condições:

- $\phi_i \in \Lambda \cup \Gamma$.
- ϕ_i é a conclusão de uma regra de inferência e possui as fórmulas $\{\phi_j, \phi_{j-1}, ...\phi_{j-n}\}$ como premissas, tal que $0 \leq n \leq j < i$.

PUC-Rio - Certificação Digital Nº 1712666/CA

Derivações podem ter diferentes representações visuais. No sistema de dedução, as derivações podem ser representadas através de uma sequência de fórmulas ou através de uma estrutura de árvore (Seção 2.3).

Se a derivação de uma fórmula β ocorre a partir de um conjunto de hipóteses vazio

$$\vdash \beta$$

dizemos que a derivação é uma *prova* de β e que β é um *teorema*.

Um sistema formal lógico pode possuir vários sistemas dedutivos. A seguir, apresentamos o sistema dedutivo de dedução natural para a LPM, que possui o conjunto de axiomas vazio e regras de inferência de introdução e eliminação para cada conectivo (Figura 2.1).

$$\frac{\begin{array}{c}[\alpha]\\ \bot\end{array}}{\neg\alpha}\,\neg{-}I \qquad \frac{\alpha \quad \neg\alpha}{\bot}\,\neg{-}E$$

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta}\,\wedge{-}I \qquad \frac{\alpha \wedge \beta}{\alpha}\,\wedge{-}E_1 \qquad \frac{\alpha \wedge \beta}{\beta}\,\wedge{-}E_2$$

$$\frac{\alpha}{\alpha \vee \beta}\,\vee{-}I_1 \qquad \frac{\beta}{\alpha \vee \beta}\,\vee{-}I_2 \qquad \frac{\alpha \vee \beta \quad \begin{array}{c}[\alpha]\\ \gamma\end{array} \quad \begin{array}{c}[\beta]\\ \gamma\end{array}}{\gamma}\,\vee{-}E$$

$$\frac{\begin{array}{c}[\alpha]\\ \beta\end{array}}{\alpha \supset \beta}\,\supset{-}I \qquad \frac{\alpha \quad \alpha \supset \beta}{\beta}\,\supset{-}E$$

Figura 2.1: Regras de inferência da Dedução Natural para a LPM

Nas regras de inferência $\supset{-}E$ e $\vee{-}E$, a premissa α da $\supset{-}E$ e as premissas γ da $\vee{-}E$ são chamadas de *premissas menores*, todas as outras premissas que não são menores são chamadas de *premissas maiores*. Em uma derivação, um segmento $\langle\alpha_1, ..., \alpha_n\rangle$, onde α_1 é uma conclusão de uma regra de introdução e α_n é a premissa maior de uma regra de eliminação, é chamado de *segmento maximal*. Uma derivação que não contém um segmento maximal é chamada de *derivação normal* (20). Prawitz mostrou com o Teorema da Normalização que para o sistema de dedução natural da LPM, se $\Gamma \vdash \alpha$, então existe uma derivação normal de α a partir de Γ (20). Uma derivação normal satisfaz o *princípio da subfórmula*, que estabelece que toda ocorrência

PUC-Rio - Certificação Digital Nº 1712666/CA

de fórmula em uma derivação $\Gamma \vdash \alpha$ é uma subfórmula de α ou de alguma fórmula pertencente a Γ.

2.2.3 Semântica

Seguindo a semântica para a LPI proposta por Kripke (21). Um *modelo* $\mathcal{M}$ é uma tripla $(W, \mathcal{R}, \Phi)$, onde W é um conjunto não vazio de mundos; $\mathcal{R}$ é uma relação de ordem parcial em W; Φ é um função binária, $\Phi : P \times W \rightarrow \{V, F\}$, onde P é conjunto dos símbolos proposicionais, tal que, se $\Phi(p, h) = V$ e $h\mathcal{R}h'$, então $\Phi(p, h') = V$. Se $\Phi(p, h) = V$ em um modelo $\mathcal{M}$, denotamos $\mathcal{M}, h \vDash p$. O valor de Φ para fórmulas proposicionais é definido por indução estrutural na definição de fórmulas:

1. $\mathcal{M}, h \vDash \alpha \wedge \beta$, se e somente se $\mathcal{M}, h \vDash \alpha$ e $\mathcal{M}, h \vDash \beta$.
2. $\mathcal{M}, h \vDash \alpha \vee \beta$, se e somente se $\mathcal{M}, h \vDash \alpha$ ou $\mathcal{M}, h \vDash \beta$.
3. $\mathcal{M}, h \vDash \alpha \supset \beta$, se e somente se para todo $h' \in W$ e $h\mathcal{R}h'$, $\mathcal{M}, h' \nvDash \alpha$ ou $\mathcal{M}, h' \vDash \beta$.
4. $\mathcal{M}, h \vDash \neg\alpha$, se e somente se para todo $h' \in W$ e $h\mathcal{R}h'$, $\mathcal{M}, h' \nvDash \alpha$.

Se para todo $h \in W$ em um modelo $\mathcal{M}$, $M, h \vDash \alpha$, dizemos que α é *válida* em $\mathcal{M}$. Se uma fórmula é válida em todos os modelos, dizemos que ela é uma *tautologia*.

2.2.4 Fragmento Puramente Implicacional

O fragmento puramente implicacional da LPM (M$\supset$) é restrito somente ao conectivo $\supset$. A Linguagem e a semântica de Kripke são restritas às fórmulas com apenas o conectivo $\supset$. O sistema de dedução natural contém apenas as regras de inferência $\supset -I$ e $\supset -E$ e é *correto* e *completo* em relação à semântica de Kripke, ou seja, uma fórmula α é um teorema, $\vdash \alpha$, se e somente se α é uma tautologia, $\vDash \alpha$.

Statman mostrou que os problemas de determinar se uma fórmula é uma tautologia da LPI e da M$\supset$ são PSPACE-Completos (1). Saber se qualquer tautologia de M$\supset$ admite provas com tamanho polinomialmente limitado está relacionado com saber se NP = PSPACE.

PUC-Rio - Certificação Digital Nº 1712666/CA

2.3 Estilos de Provas em Dedução Natural

Jaśkowski e Gentzen conduziram pesquisas paralelas e independentes em lógica, coincidentemente, publicaram em 1934 duas abordagens para a dedução natural. Apesar da não interação entre as pesquisas, os dois métodos de representação de provas em dedução natural resultantes das duas abordagens são equivalentes para a LPI e a LPC (22) e possuem traduções entre si (23). Cada representação possui vantagens e desvantagens em relação a outra, no entanto, a expressividade de ambas as representações é a mesma, sendo melhores denominadas como *estilos de representação* de provas em dedução natural.

No estilo de Gentzen, uma prova possui o formato de uma árvore, onde cada nó corresponde a ocorrência de uma fórmula na derivação e a fórmula associada a raiz é a conclusão da derivação. Denominaremos esse estilo de representação como "Gentzen-Prawitz", apresentada com mais detalhes na Seção 2.3.1, dado que as convenções que utilizaremos aqui foram estabelecidas por Dag Prawitz (20). As provas no estilo de Jaśkowski são uma sequência linear de fórmulas, onde a última fórmula é a conclusão da derivação. Na Seção 2.3.2 apresentamos mais detalhes desse estilo de representação, que chamaremos de "Jaśkowski-Fitch", pois, apesar de ser proposta primeiramente por Jaśkowski, a versão mais utilizada na literatura é um refinamento elaborado por Fitch (24).

2.3.1 Estilo de Gentzen-Prawitz

As derivações nesse estilo são representadas por meio de árvores, onde cada nó é rotulado com uma fórmula e cada aresta é rotulada com uma regra inferência. As fórmulas associadas às folhas são as hipóteses e a fórmula associada à raiz é a conclusão da derivação. Esse estilo é baseado na forma como as regras de inferência da Figura 2.1 são estruturadas. A representação de uma derivação é um agrupamento gráfico de sucessivas aplicações de regras de inferência, onde cada fórmula é: exclusivamente uma premissa de uma regra, simultaneamente uma premissa de uma regra e conclusão de outra regra, ou exclusivamente a conclusão de uma regra. Como exemplo de derivação nesse estilo, a Figura 2.2 mostra a derivação da fórmula $(A \supset (B \supset C)) \supset (B \supset (A \supset C))$ em M$\supset$.

A árvore que representa a derivação não possui uma representação usual de árvores, onde cada par de nós adjacentes são conectados por arestas. Cada linha horizontal representa a aplicação de uma regra de inferência, onde as

PUC-Rio - Certificação Digital Nº 1712666/CA

$$\dfrac{\dfrac{[B]^{(2)} \qquad \dfrac{[A]^{(3)} \qquad [A \supset (B \supset C)]^{(1)}}{B \supset C} \supset\!-E}{C} \supset\!-E}{\dfrac{A \supset C}{B \supset (A \supset C)} \supset\!-I^{(2)}} \supset\!-I^{(3)}}{(A \supset (B \supset C)) \supset (B \supset (A \supset C))} \supset\!-I^{(1)}$$

Figura 2.2: Exemplo de derivação no estilo de Gentzen-Prawitz

premissas são localizadas acima e a conclusão é localizada abaixo da linha. Cada fórmula na árvore é uma *ocorrência de fórmula*. Duas ocorrências de fórmulas são idênticas se são ocorrências da mesma fórmula. Uma *fórmula inicial* é uma ocorrência de fórmula que não está imediatamente abaixo de nenhuma outra ocorrência de fórmula. Uma *fórmula final* é uma ocorrência de fórmula que não está imediatamente acima de nenhuma outra ocorrência de fórmula. A conclusão, única fórmula final na derivação, depende de um subconjunto (possivelmente vazio) do conjunto das fórmulas inciais. As fórmulas inciais das quais a conclusão não depende são descartadas na derivação, sendo chamadas de *hipóteses descartadas*. Em M$\supset$, a única maneira de descartar uma hipótese é através da aplicação da regra $\supset\!-I$. Para identificar quais hipóteses são descartadas e onde são descartadas, são associados numerais às hipóteses e às regras de inferência onde elas são descartadas. Na derivação da Figura 2.2, as hipóteses $A \supset (B \supset C)$, B e A são descartadas, respectivamente, nas regras $\supset\!-I^{(1)}$, $\supset\!-I^{(2)}$ e $\supset\!-I^{(3)}$.

2.3.2 Estilo de Jaśkowski-Fitch

As derivações são representadas por uma sequência numerada de passos, onde cada passo é composto por uma fórmula e sua referida justificativa. As hipóteses são identificadas na justificativa e são sublinhadas, a conclusão é a fórmula do último passo da sequência. A Figura 2.2 mostra um exemplo de derivação da fórmula $(A \supset (B \supset C)) \supset (B \supset (A \supset C))$ em M$\supset$.

PUC-Rio - Certificação Digital Nº 1712666/CA

1	$A \supset (B \supset C)$	hip
2	$\quad B$	hip
3	$\quad\quad A$	hip
4	$\quad\quad A \supset (B \supset C)$	Rep, 1
5	$\quad\quad B \supset C$	$\supset -E$, 3, 4
6	$\quad\quad B$	Rep, 2
7	$\quad\quad C$	$\supset -E$, 5, 6
8	$\quad B \supset C$	$\supset -I$, 3–7
9	$B \supset (A \supset C)$	$\supset -I$, 2–8
10	$(A \supset (B \supset C)) \supset (B \supset (A \supset C))$	$\supset -I$, 1–9

Figura 2.3: Exemplo de derivação no estilo de Jaśkowski-Fitch

Diferentemente do estilo de Gentzen-Prawitz, as subprovas são explicitamente representadas pelos diferentes níveis da prova. Cada vez que uma hipótese é introduzida, um novo nível é aberto na prova. Esse nível, representado por uma linha vertical, identifica a subprova onde a hipótese ainda é válida.

PUC-Rio - Certificação Digital Nº 1712666/CA

3
Compressão de Provas e Dados

3.1
Compressão Horizontal

Proposta por Gordeev e Haeusler (6), a *Compressão Horizontal*[1] (CH) é uma técnica de compressão de provas em dedução natural da M$\supset$ que utiliza grafos direcionados para representar as provas. Essa técnica de compressão é utilizada como uma ferramenta para provar a conjectura $NP = PSPACE$. Dada a prova de uma tautologia qualquer da M$\supset$, que pode possuir tamanho exponencial em relação ao tamanho da conclusão, o objetivo da CH é que a prova resultante da compressão possua tamanho polinomialmente limitado em relação ao tamanho da conclusão. Na representação de provas utilizando grafos direcionados, cada vértice do grafo é rotulado com uma fórmula. O processo de compressão ocorre através da fusão de vértices rotulados com fórmulas idênticas que estejam localizados na mesma seção horizontal da derivação, iniciando no nível da conclusão e indo até os níveis das hipóteses.

PUC-Rio - Certificação Digital Nº 1712666/CA

3.1.1
Definições sobre grafos

Esta seção apresenta os principais conceitos sobre grafos utilizados para apresentar a Compressão Horizontal.

Definição 3.1. Grafo. Um *grafo* é uma tripla ordenada $G = \langle V(G), A(G), I_G \rangle$, onde $V(G)$ é um conjunto não vazio, $V(G)$ e $A(G)$ são conjuntos disjuntos e I_G é uma função de incidência que associa cada elemento de $A(G)$ a um par não ordenado de elementos de $V(G)$, $I_G : A(G) \to V(G) \times V(G)$. Elementos de $V(G)$ são chamados de *vértices* e os elementos de $A(G)$ são chamados de *arestas*.

Exemplo 3.2. Se $V(G) = \{v_1, v_2, v_3\}$, $A(G) = \{a_1, a_2, a_3\}$ e I_G é dado por $I_G(a_1) = (v_1, v_2)$, $I_G(a_2) = (v_2, v_3)$ e $I_G(a_3) = (v_1, v_2)$, então $\langle V(G), A(G), I_G \rangle$ é um grafo.

[1]Além da referência (6), a descrição da Compressão Horizontal apresentada neste trabalho é originada de notas não publicadas (28) e comunicação pessoal com Haeusler

Essa definição de grafo permite que existam mais de uma aresta entre um mesmo par de vértices (multiarestas), além disso, permite que existam arestas que conectam um vértice a ele mesmo (laços). Grafos que que possuem multiarestas e/ou laços são chamados de *multigrafos*. Grafos que permitem no máximo uma aresta entre cada par de vértices são chamados de grafos simples.

Definição 3.3. Grafo simples. Um *grafo simples* é um par ordenado $\langle V(G), A(G)\rangle$, onde $A(G)$ é um subconjunto de pares não ordenados de $V(G)$, $A(G) \subseteq V(G) \times V(G)$. Se $a, b \in V(G)$ e $(a, b) \in A(G)$, então $a \neq b$.

Os grafos que possuem o conjunto de arestas composto por pares não ordenados do conjunto de vértices são chamados de *grafos não direcionados*. Os grafos com o conjunto de arestas composto por pares ordenados são chamados de *grafos direcionados*.

Definição 3.4. Grafo direcionado. Um *grafo direcionado* é um par ordenado $\langle V(G), A(G)\rangle$, onde $A(G)$ é um subconjunto de pares ordenados de $V(G)$, $A(G) \subseteq V(G) \times V(G)$. Se $a, b \in V(G)$ e $(a, b) \in A(G)$, então $(a, b) \neq (b, a)$.

Exemplo 3.5. Se $V(G) = \{v_1, v_2, v_3\}$, $A(G) = \{(v_1, v_3), (v_2, v_3), (v_2, v_1)\}$, onde cada elemento de $A(G)$ é um par ordenado, então $\langle V(G), A(G)\rangle$ é um grafo simples e um grafo direcionado.

Nas definições seguintes, os grafos são direcionados e simples.

Definição 3.6. Grafos rotulados. Dado um grafo $(V(G), A(G))$ e um conjunto de rótulos R. $\langle V(G), A(G), r_v\rangle$ é um grafo rotulado nos vértices, onde r_v é uma função que associa cada vértice a um rótulo de R, $r_v : V(G) \to R$. Um grafo rotulado nas arestas possui uma função r_a das arestas para os rótulos, $r_a : A(G) \to R$.

Definição 3.7. Grafos coloridos. Seja um grafo $G = \langle V(G), A(G)\rangle$, um grafo com arestas coloridas possui uma coloração de arestas c, $c : A(G) \to S$, onde S é um conjunto de cores. Um grafo G é um grafo com arestas n-coloridas, se $|S| = n$.

A definição anterior de grafos coloridos nas arestas é um pouco diferente da usual, pois permite que arestas da mesma cor sejam adjacentes. Em um grafo direcionado, duas arestas são adjacentes se possuem um vértice em comum.

Definição 3.8. Caminho. Dado um grafo $\langle V(G), A(G)\rangle$. Um *caminho* com tamanho n é uma lista de n arestas de $A(G)$, $\langle (v_1^1, v_2^1), ..., (v_1^n, v_2^n)\rangle$, onde para todo i, j de 1 até n, $v_2^i = v_1^{i+1}$, $v_1^i \neq v_1^j$ e $(v_1^i, v_2^i) \neq (v_1^j, v_2^j)$.

PUC-Rio - Certificação Digital Nº 1712666/CA

Definição 3.9. Árvore. Dado um grafo $\langle V(G), A(G)), (V(G), A(G), r\rangle$ é uma *árvore*, onde $r \in V(G)$ e para todo $v \in V(G)$, existe apenas um caminho de r para v.

Definição 3.10. Grau de saída. Dado um grafo direcionado $\langle V(G), A(G)\rangle$. O *grau de saída* de um vértice $v \in V$ é a quantidade de arestas que possuem v na primeira posição do par ordenado.

Exemplo 3.11. Considere o grafo ordenado do Exemplo 3.5. Os vértices v_1, v_2 e v_3 possuem o grau de saída igual a, respectivamente, 1, 2 e 0.

Definição 3.12. Grau de entrada. Dado um grafo direcionado $\langle V(G), A(G)\rangle$. O *grau de entrada* de um vértice $v \in V$ é a quantidade de arestas que possuem v na segunda do par ordenado.

Exemplo 3.13. Considere o grafo ordenado do Exemplo 3.5. Os vértices v_1, v_2 e v_3 possuem o grau de entrada igual a, respectivamente, 1, 0 e 2.

No restante do texto, para um grafo G, omitimos a identificação de G nos conjuntos de vértices e arestas. Um grafo G é denotado por $G = (V, A)$.

PUC-Rio - Certificação Digital Nº 1712666/CA

3.1.2 Grafos direcionados como derivações

No estilo de Gentzen-Prawitz, uma derivação é uma árvore onde os nós e as arestas são rotulados com, respectivamente, fórmulas e regras de inferência. Baseado nesse estilo, uma derivação na CH é representada por um grafo direcionado com arestas coloridas que possui fórmulas como rótulos nos vértices.

Nas definições a seguir, considere as seguintes funções que mapeiam fórmulas e conjunto de fórmulas em suas respectivas subfórmulas. Sub_f é uma função que mapeia uma fórmula em um conjunto com todas as suas subfórmulas; Sub_c é uma função que mapeia um conjunto de fórmulas em um conjunto da união dos conjuntos de subfórmulas de cada fórmula.

Exemplo 3.14. Seja a fórmula $\alpha = A \supset (B \supset C)$. O conjunto de subfórmulas de α é definido por:

$$Sub_f(\alpha) = \{A,\ B,\ C,\ B \supset C,\ A \supset (B \supset C)\}$$

Exemplo 3.15. Seja o conjunto de fórmulas

$$\Delta = \{(A \supset (B \supset C)),\ (B \supset (A \supset C))\}$$

O conjunto da união dos conjuntos de subfórmulas de cada fórmula de Δ é definido por:

$$Sub_c(\Delta) = \{A,\ B,\ C,\ B \supset C,\ A \supset (B \supset C),\ A \supset C,\ B \supset (A \supset C)\}$$

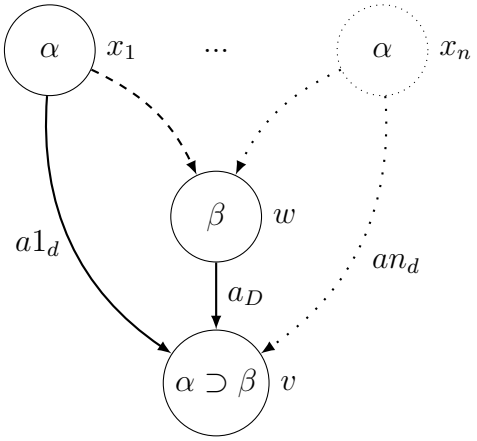
Sejam um grafo direcionado $\langle V, A\rangle$ e um subconjunto das arestas $A' \supseteq A$. As funções $gs : V \times A' \to \mathbb{N}$ e $ge : V \times A' \to \mathbb{N}$ mapeiam um vértice em seus graus, respectivamente, de saída e entrada, considerando apenas as arestas em V'.

Definição 3.16. Grafo de derivação. Seja uma derivação Π de $\Gamma \vdash \alpha$, um *grafo de derivação* é um grafo simples, direcionado, colorido nas arestas e rotulado nos vértices $\langle V, A, r, c, l\rangle$, onde: V é o conjunto não vazio de vértices; A é o conjunto de arestas, $A \subseteq V \times V$; c é a função de coloração das arestas, $c : A \to \{0, 1\}$, onde as arestas do conjunto $A_D = \{\langle u, v\rangle \mid c(\langle u, v\rangle) = 0\}$ são chamadas de *arestas de dedução* e as arestas do conjunto $A_d = \{\langle u, v\rangle \mid c(\langle u, v\rangle) = 1\}$ são chamadas de *arestas de descarte*; o conjunto de arestas é formado pela união dos conjuntos das arestas de dedução e de descarte, $A = A_D \cup A_d$; toda aresta é exclusivamente de dedução ou exclusivamente de descarte, $A_D \cap A_d = \emptyset$; $\langle V, A_D, r\rangle$ é uma árvore; l é a função que rotula os vértices com as fórmulas, $l : V \to Sub_c(\Gamma) \cup Sub_f(\alpha)$; tal que:

1. O vértice r não possui arestas dedutivas saindo dele, $gs(A_D, r) = 0$, e possui a conclusão da derivação como rótulo, $l(r) = \alpha$.

2. Todo vértice $v \in V$ possui $ge(A_D, v)$ igual a 0, 1 ou 2.

3. Para todo vértice $v \in V$, se $ge(A_D, v) = 1$, então $ge(A_d, v) \geq 1$ e $l(v)$ é a conclusão de uma regra $\supset -I$, tal que

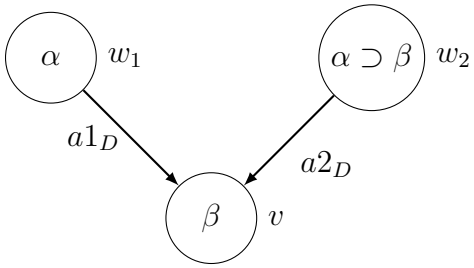


o rótulo do vértice w associado a v por uma aresta de dedução (a_D) é idêntico ao consequente de $l(v)$; os rótulos dos vértices $(x_1, ..., x_n)$

PUC-Rio - Certificação Digital Nº 1712666/CA

associados a v por arestas de descarte $(a1_d, ..., an_d)$ são idênticos ao antecedente de $l(v)$.

4. Para todo vértice $v \in V$, se $ge(A_D, v) = 2$, então $l(v)$ é a conclusão de uma regra $\supset -E$, tal que



sejam os vértices w_1 e w_2 associados a v, onde $|l(w_1)| < |l(w_2)|$, o rótulo de w_1 é idêntico ao antecedente de $l(w_2)$, o rótulo de v é idêntico ao consequente de $l(w_2)$.

5. Para todo vértice $v \in V$, se $ge(A_D, v) = 0$ e $gs(A_d, v) = 0$, então $l(v)$ é uma hipótese de Π.

Para exemplificar a representação de derivações por Grafos de Prova, a Figura 3.1 mostra o grafo que representa a derivação de

$$\vdash (A \supset (B \supset C)) \supset (B \supset (A \supset C))$$

As arestas com linhas tracejadas são arestas de descarte.

Seja uma derivação Π de $\Gamma \vdash \alpha$, a conclusão α depende do conjunto de hipóteses Γ. Em um grafo de derivação que representa Π, o conjunto de hipóteses é composto pelos rótulos dos vértices que não possuem arestas de dedução incidentes e que não foram descartados por arestas de descarte. No entanto, as arestas de descarte (A_d) adicionam ciclos no grafo $\langle V, A_D \rangle$ (desconsiderando a direção das arestas).

Definição 3.17. Seja uma derivação Π de β tendo o conjunto Γ de hipóteses. Uma aplicação de $\supset -I$ é gulosa, se e somente se, produz $\alpha \supset \beta$ e descarta cada ocorrência não descartada de α em Π na qual β depende.

Lema 3.18. Seja uma derivação Π de α. A derivação Π' resultante da transformação de todas as regras $\supset -I$ de Π em gulosas ainda é uma derivação válida de α em M$\supset$ (28).

Com as regras $\supset -I$ gulosas, as informações de dependências podem ser diretamente determinadas nos vértices (V) ou nas arestas de dedução (A_D)

PUC-Rio - Certificação Digital Nº 1712666/CA

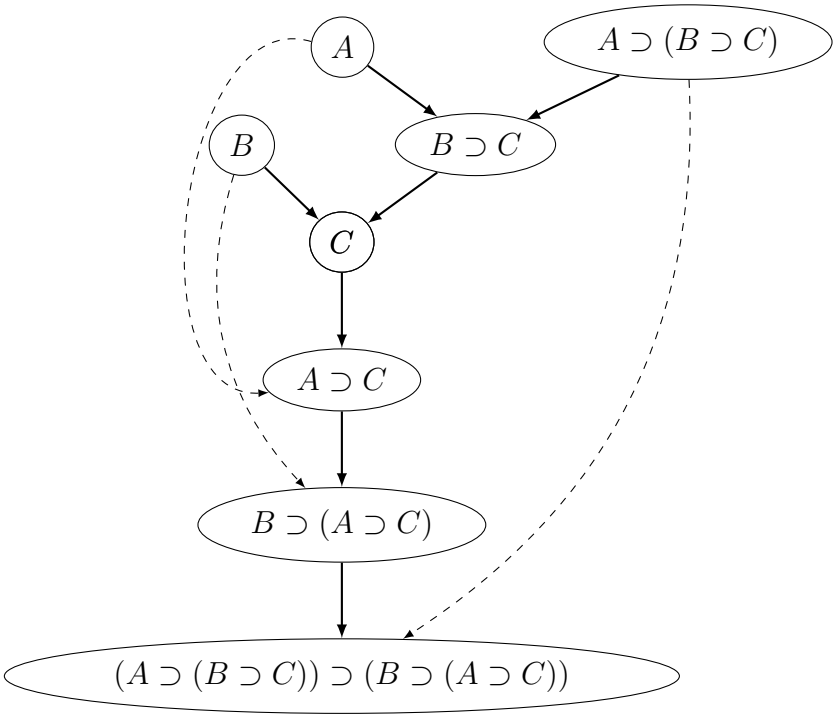


Figura 3.1: Exemplo de grafo de derivação

PUC-Rio - Certificação Digital Nº 1712666/CA

e portanto, as arestas de descarte deixam de ser necessárias. As definições a seguir utilizam grafos de provas com todas as regras $\supset -I$ gulosas.

Definição 3.19. Seja Λ um conjunto de fórmulas de tamanho n e $\mathcal{O}(\Lambda)$ uma ordenação linear das fórmulas de Λ, $\mathcal{O}(\Lambda) = \{\beta_1, \beta_2, ..., \beta_n\}$. Uma cadeia de *bits* em $\mathcal{O}(\Lambda)$ é uma sequência de *bits* $\langle b_1, b_2, ..., b_n\rangle$, onde cada $b_i \in \{0, 1\}$ com i variando de 1 até n. Para todo conjunto de fórmulas Λ de tamanho n, existe uma função bijetora f entre o conjunto de todas as cadeias de bits de tamanho n e o conjunto das partes de Λ, $f : B(\mathcal{O}(\Lambda)) \to \mathcal{P}(\Lambda)$, $f(\langle b_1, ..., b_n\rangle) = \{\beta_i | b_i = 1\}$.

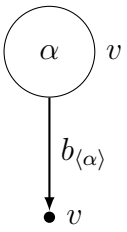
Exemplo 3.20. Seja $\Lambda = \{A, B, C, (A \supset B)\}$ e uma ordenação linear $\mathcal{O}(\Lambda) = [(A \supset B), A, C, B]$, onde $(A \supset B)$, A, C e B possuem os índices 1, 2, 3 e 4, respectivamente. Para cadeias de *bits* de tamanho 4, a função f da definição anterior possui o seguinte comportamento para as seguintes cadeias de *bits*: $f(\langle 0, 1, 1, 0\rangle) = \{A, C\}$; $f(\langle 1, 0, 1, 0\rangle) = \{(A \supset B), C\}$.

Em uma derivação de $\Gamma \vdash \alpha$, sendo o conjunto $\Lambda = Sub_c(\Gamma) \cup Sub_f(\alpha)$, $\mathcal{O}(\Lambda)$ pode ser utilizado para associar cada ocorrência de fórmula com suas respectivas dependências.

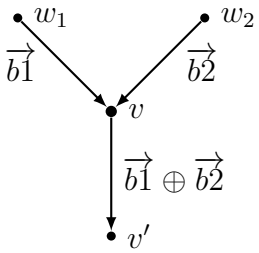
Na definição a seguir, considere $b_{\langle\beta\rangle}$ como a cadeia de *bits* que contém apenas o *bit* referente à fórmula β igual a 1.

Definição 3.21. Grafo de derivação decorado. Sejam uma derivação Π de $\Gamma \vdash \alpha$, um conjunto de fórmulas $\Lambda = Sub_c(\Gamma) \cup Sub_f(\alpha)$, um grafo de derivação de Π $\langle V, A_D, A_d, r, c, l\rangle$ e uma ordenação linear $\mathcal{O}(\Lambda)$. $\langle V, A_D, r, c, l, d\rangle$ é um *grafo de derivação decorado*, onde d é uma função que associa cada aresta $(v1, v2) \in A_D$ a uma cadeia de bits que representa as dependências de $l(v1)$, $d : A_D \to B(\mathcal{O}(\Lambda))$, tal que:

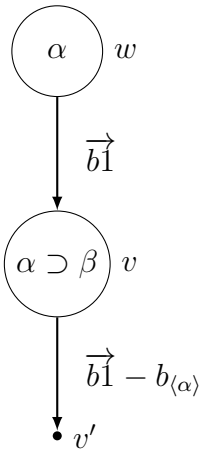
1. Para todo $v \in V$, se v é uma hipótese e $l(v) = \alpha$, então $d(\langle v, v'\rangle) = b_{\langle\alpha\rangle}$ para algum $v' \in V$.



2. Para todo $v \in V$, se v é a conclusão de uma regra $\supset\!-E$ e possui $w1, w2 \in V$ como premissas, onde $d(\langle w1, v\rangle) = \overrightarrow{b1}$ e $d(\langle w2, v\rangle) = \overrightarrow{b2}$, então $d(\langle v, v'\rangle) = \overrightarrow{b1} \oplus \overrightarrow{b2}$ para algum $v' \in V$.



3. Para todo $v \in V$, se v é a conclusão de uma regra $\supset\!-I$, possui $l(v) =$ '$\alpha \supset \beta$' e $w \in V$ como premissa, onde $d(\langle w, v\rangle) = \overrightarrow{b1}$ e $l(w) = \alpha$, então $d(\langle v, v'\rangle) = \overrightarrow{b1} - b_{\langle\alpha\rangle}$ para algum $v' \in V$.



PUC-Rio - Certificação Digital Nº 1712666/CA

A Figura 3.2 mostra um exemplo de grafo de derivação decorado obtido a partir do grafo de derivação da Figura 3.1. Considerando

$$\alpha = (A \supset (B \supset C)) \supset (B \supset (A \supset C))$$

e ainda considerando Δ igual a $Sub_f(\alpha)$, onde

$$Sub_f(\alpha)$$

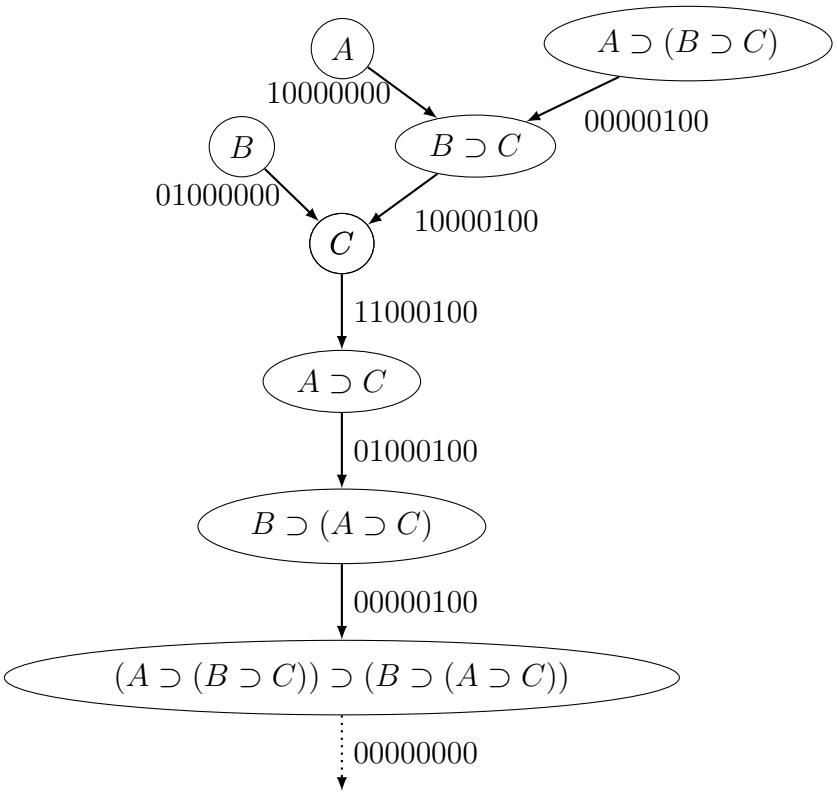


Figura 3.2: Exemplo de grafo de derivação decorado

Um grafo de derivação decorado pode ser visto como uma árvore. A Compressão Horizontal ocorre em um grafo de derivação decorado fundindo os vértices que possuem rótulos idênticos e que estejam no mesmo nível da árvore. Durante a compressão, algumas informações adicionais são inseridas no grafo, essas informações são utilizadas para verificar se o grafo resultante da compressão corresponde a uma derivação válida. A seguir, definimos a estrutura do grafo de derivação utilizado pela Compressão Horizontal.

Definição 3.22. Estrutura de grafo de derivação decorado (EGDD). Seja Π uma derivação de $\Gamma \vdash \alpha$, $\Delta = Sub_c(\Pi) \cup Sub_f(\alpha)$, um grafo de

PUC-Rio - Certificação Digital Nº 1712666/CA

derivação decorado de Π $\langle V, A_D, l, r, c, d\rangle$ e uma ordenação linear $\mathcal{O}(\Delta)$. $\langle V, A_D, A_A, r, l, g, d, p\rangle$ é uma EGDD, onde:

1. V é o conjunto não vazio de vértices.
2. A_D é o conjunto das *arestas de dedução*.
3. $A_A \in V \times V$ é o conjunto das *arestas de ancestralidade*.
4. $l : V \to \Delta$ é a função que rotula os vértices com suas respectivas fórmulas em Π.
5. r é a raiz de $\langle V, (A_D^i)_{i\in\mathcal{O}(\Delta)}\rangle$, onde $l(r) = \alpha$.
6. $g : A_D \to \{1, ... |\Delta|\}$ é a função de coloração das arestas de dedução. Cada elemento do conjunto $\{1, ... |\Delta|\}$ pode ser visto como uma cor para uma aresta de dedução.
7. $d : \bigcup_{i\in\mathcal{O}(\Delta)} A_D^i \to B(\mathcal{O}(\Delta))$ é a função que associa cada aresta de dedução $\langle v1, v2\rangle$ à cadeia de bits que denota a dependência de $v1$.
8. $p : A_A \to \{1, ... |\Delta|\}^\star$, onde para cada aresta $\langle v1, v2\rangle \in A_A$, $p(\langle v1, v2\rangle)$ é uma cadeia de caracteres $\langle c_1, ..., c_n\rangle$, onde cada c_i, com $i = 1$ até n, é um índice da ordenação linear $\mathcal{O}(\Delta)$.

Antes de iniciar a compressão, a EGDD possui apenas arestas de dedução. Todas as arestas de dedução possuem cor 0 e todos os vértices possuem o grau de saída igual a 1, exceto a raiz, que possui grau de saída igual a 0.

3.1.3 Algoritmo de compressão

A Compressão Horizontal comprime a EGDD através da fusão de vértices rotulados com fórmulas idênticas que estejam no mesmo nível da árvore de derivação. Para cada fórmula em cada nível da árvore, uma estrutura de fila do tipo *FIFO*[2] contendo os vértices rotulados com a respectiva fórmula é criada. As fusões ocorrem com os vértices que estejam em uma fila com mais de um vértice. Em cada fila com mais de um elemento, os vértices são desenfileirados[3] e colapsados em pares. O colapso consiste na operação de remover dois vértices do grafo e adicionar um novo vértice, que é o resultante do colapso dos vértices removidos. O colapso sempre ocorre com pelo menos um vértice que ainda não foi colapsado. No caso de filas com mais de dois vértices, a partir do segundo,

[2] *First In, First Out* - primeiro elemento inserido é o primeiro a ser removido.
[3] Operação de remover um elemento da fila

PUC-Rio - Certificação Digital Nº 1712666/CA

os colapsos ocorrem entre o vértice resultante do colapso anterior e um vértice retirado da fila.

A Compressão Horizontal é executada pelo Algoritmo 1. A função *vérticesRótulosIdênticos* retorna uma lista de filas para cada fórmula presente em determinado nível da árvore de derivação. A função *desenfileirar* executa a operação de remoção em uma fila.

Algoritmo 1: Compressão Horizontal

Entrada: EGDD

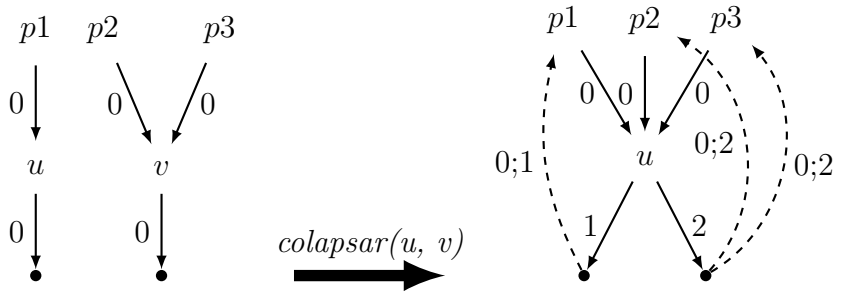
Saída: EGDD comprimida

```
1  para nível = 1 até altura(G) faça
2      vértices_nível ← vérticesRótulosIdênticos(G, nível)
3      para vértices ∈ vértices_nível faça
4          vértice_1 ← desenfileirar(vértices)
5          enquanto tamanho(vértices) > 0 faça
6              vértice_2 ← desenfileirar(vértices)
7              vértice_1 ← colapsar(vértice_1, vértice_2)
8          fim
9      fim
10 fim
```

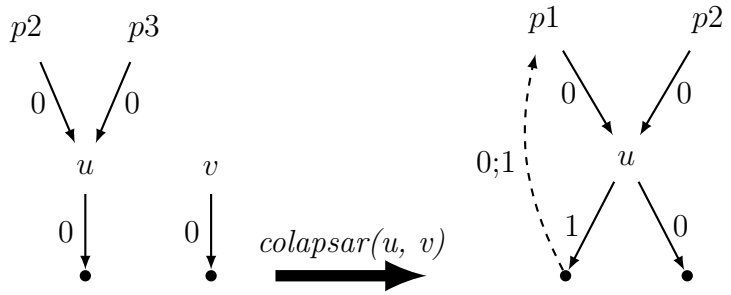
A função *colapsar* recebe dois vértices, executa o colapso e retorna o vértice colapsado. Ao todo, a função possui 18 regras distintas para executar o colapso. A regra aplicada em cada colapso é determinada de acordo com as características dos vértices. A seguir, listamos e ilustramos as principais características dos vértices consideradas no momento do colapso:

– **Arestas de ancestralidade**

Ao colapsar dois vértices, as premissas de ambos os vértices passam a apontar para o vértice colapsado. As arestas de ancestralidade servem para identificar as premissas após o colapso, que são adicionadas somente se o vértice possui premissa(s) antes do colapso.



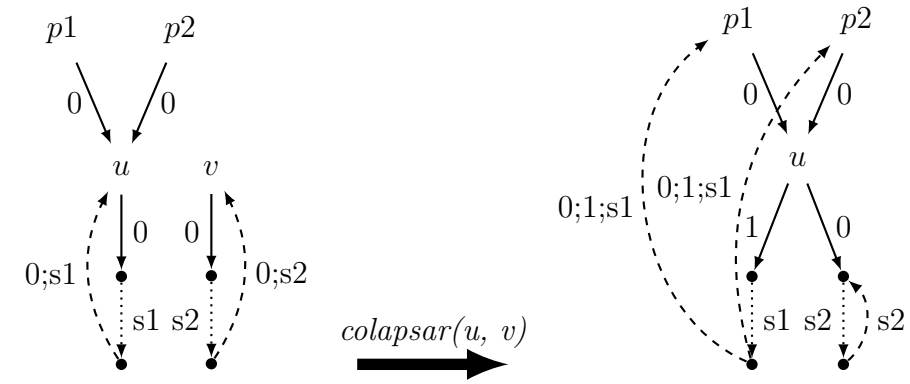
PUC-Rio - Certificação Digital Nº 1712666/CA



Cada aresta de ancestralidade possui como rótulo a informação necessária para refazer o caminho entre o destino e a origem através das arestas de dedução, esse caminho é identificado através das cores das arestas de dedução. Se o vértice possui premissas antes do colapso, é necessário atribuir uma cor para sua respectiva aresta dedutiva de saída após o colapso. Todas as arestas de dedução que saem de um determinado vértice possuem cores distintas, exceto para a cor 0.

– **Arestas de ancestralidade incidentes**

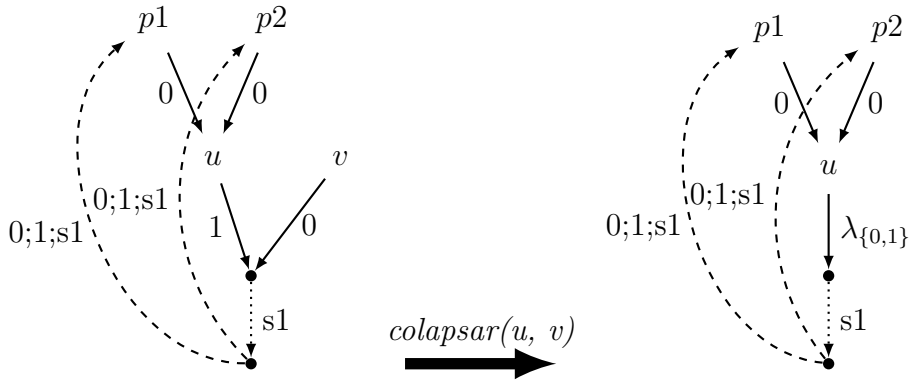
Caso algum dos vértices a serem colapsados possua uma aresta de ancestralidade incidente, é necessário atualizar o destino e consequentemente o rótulo da aresta. Se o vértice possuir premissas, a aresta de ancestralidade é redirecionada para as premissas, caso contrário, a aresta é redirecionada para o vértice imediatamente inferior.



– **Colapso de arestas**

Nem sempre os destinos das arestas de dedução que saem dos vértices a serem colapsados possuem destinos diferentes. Caso os destinos já tenham sido colapsados no nível inferior, além de colapsar os vértices, é necessário colapsar as arestas.

PUC-Rio - Certificação Digital Nº 1712666/CA



λ é a cor especial utilizada para indicar que a aresta foi colapsada. As arestas colapsadas ainda mantêm a informação das cores originais das arestas.

3.2 Codificação de Huffman

Proposta em 1952 por David Huffman (25), a *Codificação de Huffman* é uma técnica de compressão de dados sem perdas, que permite recuperar o dado original a partir do dado comprimido, baseada na substituição de símbolos do dado por códigos. O objetivo é minimizar a quantidade de espaço necessário para representar o dado atribuindo códigos menores para os símbolos mais frequentes e códigos maiores para os menos frequentes. Para otimizar o uso do espaço, normalmente, utiliza-se uma codificação binária para comprimir os dados, no entanto, a codificação de Huffman é extensível para codificações n-árias.

Definição 3.23. Codificação binária. Seja uma mensagem m com um alfabeto Σ, uma *codificação binária* é uma função F que mapeia cada símbolo de Σ em uma cadeia de bits, $F : \Sigma \to \{0,1\}^*$, tal que:

1. Para todo símbolo $a \in \Sigma$, $F(a) \neq \epsilon$, $F(a)$ não é vazia.
2. Para todos símbolos $a, b \in \Sigma$, $F(a) \neq F(b)$.
3. Para todos símbolos $a, b \in \Sigma$, $F(a)$ não é prefixo de $F(b)$.

A condição (3) permite que, dada uma mensagem $m = \langle a_1, ..., a_n \rangle$ e uma codificação binária F, a mensagem codificada $m_c = \langle F(a_1), ..., F(a_n) \rangle$ seja decodificada sem informações adicionais. Por exemplo, seja uma mensagem *babcb* e uma codificação binária, $F(a) = 111$, $F(b) = 010$ e $F(c) = 000$. Uma mensagem codificada 010111010000010 possui apenas uma segmentação

PUC-Rio - Certificação Digital Nº 1712666/CA

possível baseada na codificação binária do alfabeto, 010|111|010|000|010, que corresponde exatamente a mensagem original *babcb*.

A construção da codificação binária utilizada por Huffman é baseada na probabilidade de ocorrência dos símbolos do alfabeto na mensagem a ser codificada, utilizando tabelas auxiliares para atribuir códigos a cada símbolo do alfabeto. Seja (s, p), um símbolo e sua respectiva probabilidade de ocorrência, a construção das tabelas auxiliares segue os seguintes passos:

1. Inicialmente, uma tabela com todos os símbolos e suas respectivas probabilidades é criada, as entradas são ordenadas em ordem decrescente das probabilidades.

2. Seleciona as entradas com as menores probabilidades, (s_1, p_1) e (s_2, p_2), cria uma entrada auxiliar $((s_1, s_2), p_1+p_2)$ e adiciona em uma nova tabela com as entradas de todos os símbolos, exceto s_1 e s_2.

3. Repete o passo 2 até que a tabela criada tenha apenas uma entrada.

A Figura 3.3 mostra um exemplo de criação das tabelas auxiliares do alfabeto $\Sigma = \{a, r, j, l, m, p\}$, onde as probabilidades de a, r, j, l, m e p são, respectivamente, 0.35, 0.20, 0.15, 0.13, 0.12 e 0.05. Para facilitar a visualização, as entradas auxiliares estão destacadas em negrito.

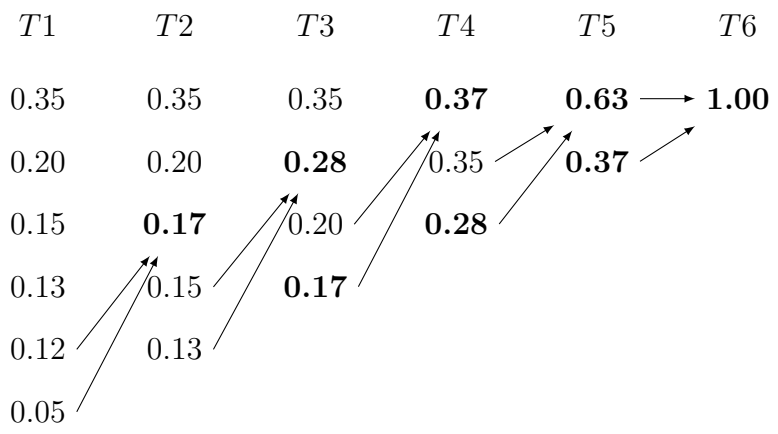


Figura 3.3: Exemplo de tabelas auxiliares da codificação de Huffman

A partir das tabelas auxiliares é possível construir uma árvore binária para a obtenção do código de cada símbolo, tal construção ocorre percorrendo as tabelas na direção inversa da criação, onde a raiz da árvore é a entrada auxiliar com probabilidade igual a 1.0 da última tabela e as folhas são as entradas originais (tabela T1) das probabilidades dos símbolos do alfabeto.

PUC-Rio - Certificação Digital Nº 1712666/CA

Para cada fusão de entradas realizada pelo passo 2 é criada uma ramificação na árvore, onde a entrada com a menor probabilidade é atribuída ao filho direito e a entrada com maior probabilidade é atribuída ao filho esquerdo. Cada arco da árvore é rotulado com 1 ou 0, os arcos dos filhos esquerdos e direitos são rotulados, respectivamente, com 0 e 1. A Figura 3.4 mostra a árvore criada a partir das tabelas da Figura 3.3.

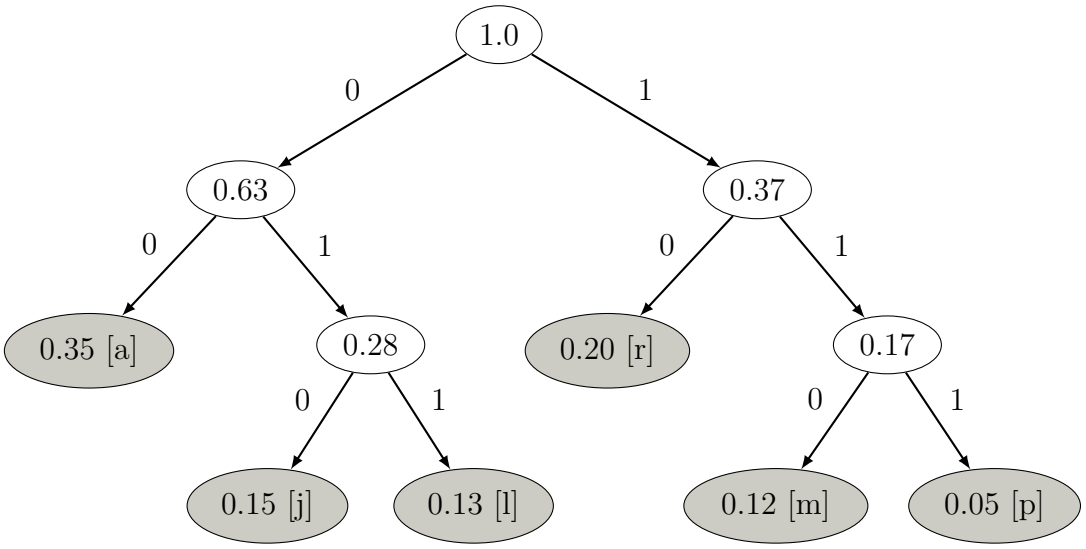


Figura 3.4: Exemplo de árvore binária criada a partir das tabelas auxiliares da codificação de Huffman

Os códigos de cada símbolo são obtidos percorrendo os caminhos da raiz até cada folha e armazenando os rótulos dos arcos percorridos. A Tabela 3.1 mostra os códigos obtidos da árvore da Figura 3.4. Os símbolos com as maiores probabilidades, 'a' e 'r', ficaram com os menores códigos, respectivamente, 00 e 10.

Tabela 3.1: Exemplos de códigos obtidos a partir da árvore binária da codificação de Huffman.

Símbolo	**Probabilidade**	**Código**
a	0.35	00
r	0.20	10
j	0.15	010
l	0.13	011
m	0.12	110
p	0.05	111

PUC-Rio - Certificação Digital Nº 1712666/CA

4
Aplicação da Compressão Horizontal

Como apresentado na Seção 3.1, a CH adiciona algumas informações à prova antes da e durante a compressão, essas informações são posteriormente utilizadas na validação da derivação compactada. A CH só é realmente efetiva na compactação se executar uma certa quantidade de colapsos durante a compressão. A prova da Figura 3.1, por exemplo, não contém nenhum par de vértices a ser colapsado, nesse caso, a aplicação da CH resultaria no aumento do espaço necessário para representar a prova.

Para exemplificar a aplicação da CH, considere a prova da seguinte fórmula denominada Fib_4 (Figura 4.1)

$$((A1 \supset A2) \supset ((A1 \supset (A2 \supset A3)) \supset ((A2 \supset (A3 \supset A4)) \supset (A1 \supset A4))))$$

que na representação de grafos direcionados contém 17 vértices, sendo que 7 deles serão colapsados durante a compressão.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[A1]_4 \quad [(A1 \to A2)]_1}{A2} \to \text{elim}_4 \quad \dfrac{[A1]_4 \quad [(A1 \to (A2 \to A3))]_2}{(A2 \to A3)} \to \text{elim}_5}{A3} \to \text{elim}_2 \quad \dfrac{\dfrac{[A1]_4 \quad [(A1 \to A2)]_1}{A2} \to \text{elim}_4 \quad [(A2 \to (A3 \to A4))]_3}{(A3 \to A4)} \to \text{elim}_3}{A4} \to \text{elim}_1}{(A1 \to A4)} \to \text{intro}_4}{((A2 \to (A3 \to A4)) \to (A1 \to A4))} \to \text{intro}_3}{((A1 \to (A2 \to A3)) \to ((A2 \to (A3 \to A4)) \to (A1 \to A4)))} \to \text{intro}_2}{((A1 \to A2) \to ((A1 \to (A2 \to A3)) \to ((A2 \to (A3 \to A4)) \to (A1 \to A4))))} \to \text{intro}_1$$

Figura 4.1: Prova da fórmula Fib_4

Neste Capítulo utilizamos como métrica para o tamanho de uma prova o tamanho do grafo direcionado que o representa (*quantidade de vértices* + *quantidade de arestas*).

O grafo que representa a prova de Fib_4 possui tamanho 40 (17 *vértices* + 16 *arestas dedutivas* + 7 *arestas de descarte*), antes de iniciar a compressão, as arestas de descarte são substituídas pelas cadeias de bits de dependências associadas a cada aresta dedutiva, logo, o tamanho do grafo antes de inciar a compressão é 33.

Considerando o nível da conclusão como o nível 1, o primeiro colapso é realizado no nível 7 entre os 2 vértices rotulados com a fórmula 'A2'.

PUC-Rio - Certificação Digital Nº 1712666/CA

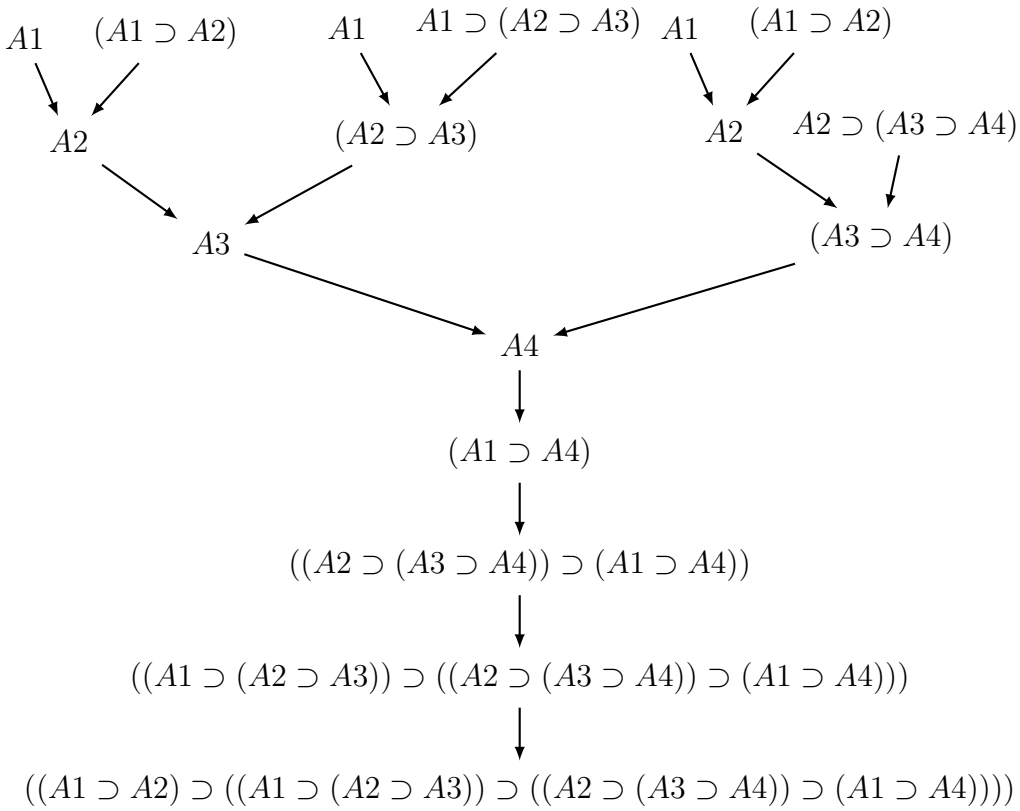
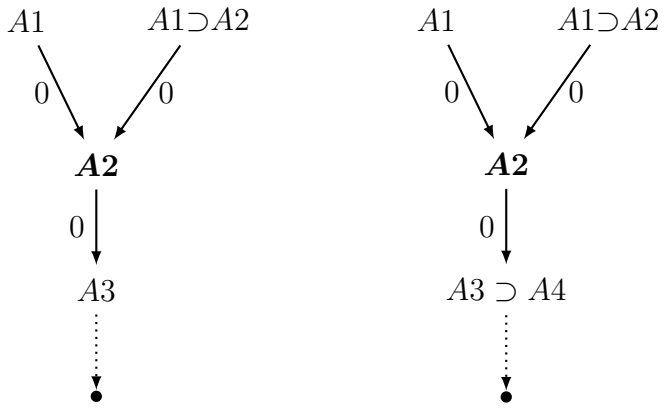
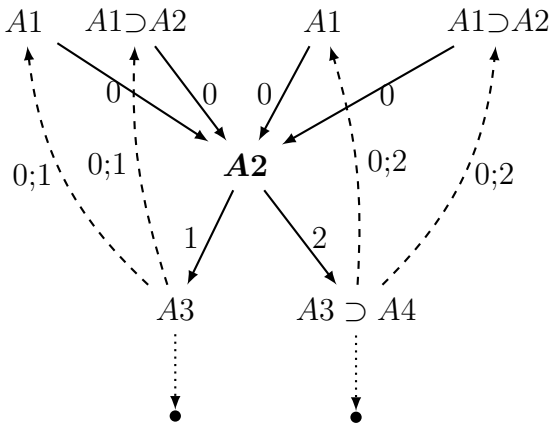


Figura 4.2: EGDD da derivação de Fib_4

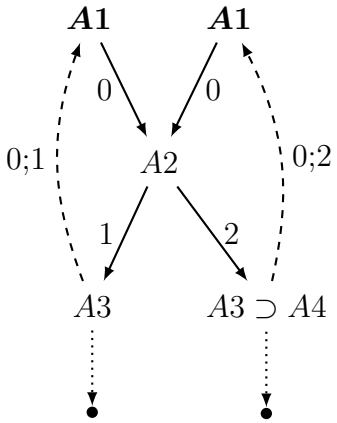


Como ambos os vértices possuem 2 premissas e ainda não foram colapsados, são adicionadas 4 arestas de ancestralidade.

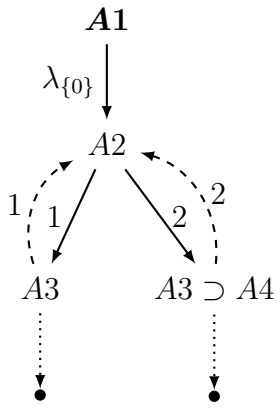
PUC-Rio - Certificação Digital Nº 1712666/CA



No nível 8, o primeiro de dois colapsos dos vértices rotulados com 'A1' é realizado com os dois vértices que possuem uma aresta dedutiva apontando para o vértice rotulado com 'A2', colapsado no nível inferior. Ambos os vértices são premissas e cada um possui uma aresta de ancestralidade incidente.



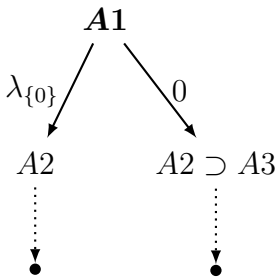
Além do colapso dos vértices, as arestas dedutivas que possuem o vértice 'V2' como destino também são colapsadas. A aresta colapsada é rotulada com a cor especial λ e com as cores das arestas dedutivas colapsadas, que nesse caso é somente a cor 0. Como os vértices não possuem premissas, as arestas de ancestralidade incidentes são rebaixadas e seus respectivos rótulos são atualizados.



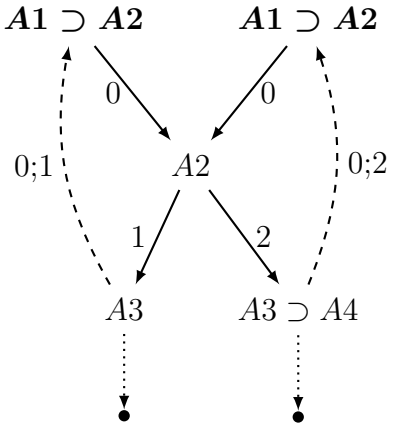
PUC-Rio - Certificação Digital Nº 1712666/CA

O vértice resultante do colapso anterior é colapsado com o outro vértice rotulado com 'A1' no nível 8 e que também não possui premissas. O vértice 'A2' possui 2 arestas de ancestralidade incidentes, mas elas não interferem no colapso.

Após o colapso, apenas um vértice é removido do grafo.

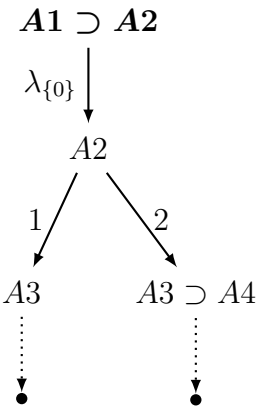


O último colapso, ainda no nível 8, envolve os dois vértices rotulados com $A1 \supset A2$. Ambos os vértices não possuem premissas e cada um possui uma aresta de ancestralidade incidente.



Esse colapso é semelhante ao primeiro colapso dos vértices $A1$, no entanto, as arestas de ancestralidade são removidas em vez de serem rebaixadas, pois, já existem arestas de ancestralidade de A3 para A2, e de $A3 \supset A4$ para A2 com os mesmos rótulos. Esse colapso remove 3 arestas, sendo 1 dedutiva e 2 de ancestralidade.

PUC-Rio - Certificação Digital Nº 1712666/CA



A Tabela 4.1 mostra o tamanho do grafo em cada colapso. Apesar de não ser a métrica que descreve o real de tamanho da prova, pois são adicionadas informações às arestas durante a compressão, o tamanho do grafo evidencia o comportamento da Compressão Horizontal durante a compressão de uma prova. Nos primeiros colapsos o tamanho do grafo de prova é superior ao tamanho do grafo antes da compressão, mas a medida que colapsos vão sendo executados e vértices e arestas são retirados, o tamanho do grafo tende a diminuir.

Tabela 4.1: Tamanho do grafo direcionado em cada colapso

Passo	**Tamanho (vértices + arestas)**
grafo inicial	33 (17 + 16)
colapso 1	36 (16 + 20)
colapso 2	34 (15 + 19)
colapso 3	33 (14 + 19)
colapso 4	29 (13 + 16)

O Capítulo 5 apresenta como os grafos de prova (EGDD) da Compressão Horizontal são representados em arquivos e o Capítulo 6 apresenta os resultados de compressão utilizando a representação em arquivos de texto.

PUC-Rio - Certificação Digital Nº 1712666/CA

5
Implementação da Compressão Horizontal

Esse capítulo apresenta a implementação do *compressing*[1], compressor de provas que executa o algoritmo da Compressão Horizontal apresentado por Gordeev e Haeusler (6). O objetivo desta implementação é fornecer os primeiros resultados empíricos da compressão e auxiliar na definição do algoritmo, aperfeiçoando procedimentos e estruturas de dados utilizadas durante o processo de compressão.

5.1
Decisões de Projeto

A finalização da formalização do algoritmo da Compressão Horizontal foi realizada paralelamente à implementação do *compressing*, essa implementação tinha o objetivo inicial de retroalimentar a formalização do algoritmo com informações sobre a sua execução. Para gerar as informações sobre a execução do algoritmo seria necessário documentar os passos de execução através da geração de visualizações do grafo de prova durante o processo de compressão. Além da geração das visualizações, outro fator importante considerado durante a definição das decisões de projeto foi a necessidade de realizar várias modificações no código do compressor durante o desenvolvimento.

O *compressing* foi implementado utilizando a linguagem de programação *Python*. A escolha da linguagem levou em consideração a existência de uma implementação inicial da Compressão Horizontal escrita em Python, além de que a linguagem não demanda muito esforço para alterações no código e ainda possui várias bibliotecas com suporte a operações em grafos que possuem integração com o pacote de ferramentas de visualizações de grafos *graphviz*.

5.2
Estrutura do Projeto

Utilizamos o padrão de projeto *Adapter* para estruturar o projeto. A adoção do Adapter na estruturação do sistema permite a integração de classes externas ao sistema, convertendo a interface das classes externas à uma interface esperada pelas classes internas.

[1] https://github.com/flavio-barros/compressing

PUC-Rio - Certificação Digital Nº 1712666/CA

A Figura 5.1 mostra um exemplo de como as classes são estruturadas utilizando o *Adapter*. Na *Interface* são definidos os métodos que uma classe interna espera. A classe *Adaptador* implementa a interface e utiliza os métodos da classe externa (*ClasseAdaptada*) para definir os métodos esperados pela classe interna (*ClasseCliente*).

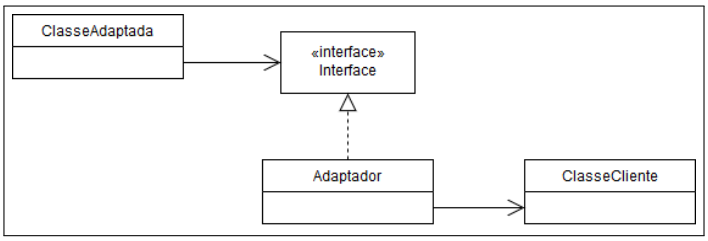


Figura 5.1: Exemplo de diagrama de classes do *Adapter*

No *compressing*, as classes adaptadas são dos pacotes que fornecem as funcionalidades de manipulação e visualização de grafos. A Figura 5.2 mostra o diagrama de classes simplificado do *compressing*. A classe *GraphAdapter* possui um objeto da classe *DiGraph* e implementa a interface *Graph*, que define todos os métodos de manipulação de grafos utilizados no *compressing*. A implementação de cada método da classe *GraphAdapter* adapta as funcionalidades oferecidas pela *DiGraph*, que pertence ao pacote *networkx*[2], para as funcionalidades definidas pela interface *Graph*. A classe *ProofGraph* implementa a EGDD definida na Seção 3.1 utilizando um objeto da classe *GraphAdapter*. De maneira semelhante, a classe *VisualGraphAdapter* converte as funcionalidades da classe *Agraph*, do pacote pygraphviz[3], implementando os métodos definidos pela interface *VisualGraph*. A principal diferença é que a *VisualGraphAdapter* é utilizada por duas classes: a classe *VisualProofGraph* implementa a visualização do do grafo de um objeto da *ProofGraph*; a classe *VisualCollapseGraph* implementa a geração da visualização de um colapso do processo de compressão, que é um subgrafo do grafo de um objeto da *ProofGraph*.

5.3 Representação das Provas

Os grafos de prova são representados por arquivos DOT, linguagem de descrição de grafos do *graphviz*. A linguagem DOT pode representar grafos direcionados e não direcionados; subgrafos; atributos de grafos, vértices e arestas. O *compressing* aceita somente arquivos *.dot* que seguem as seguintes convenções:

[2]https://networkx.github.io/
[3]https://pygraphviz.github.io/

PUC-Rio - Certificação Digital Nº 1712666/CA

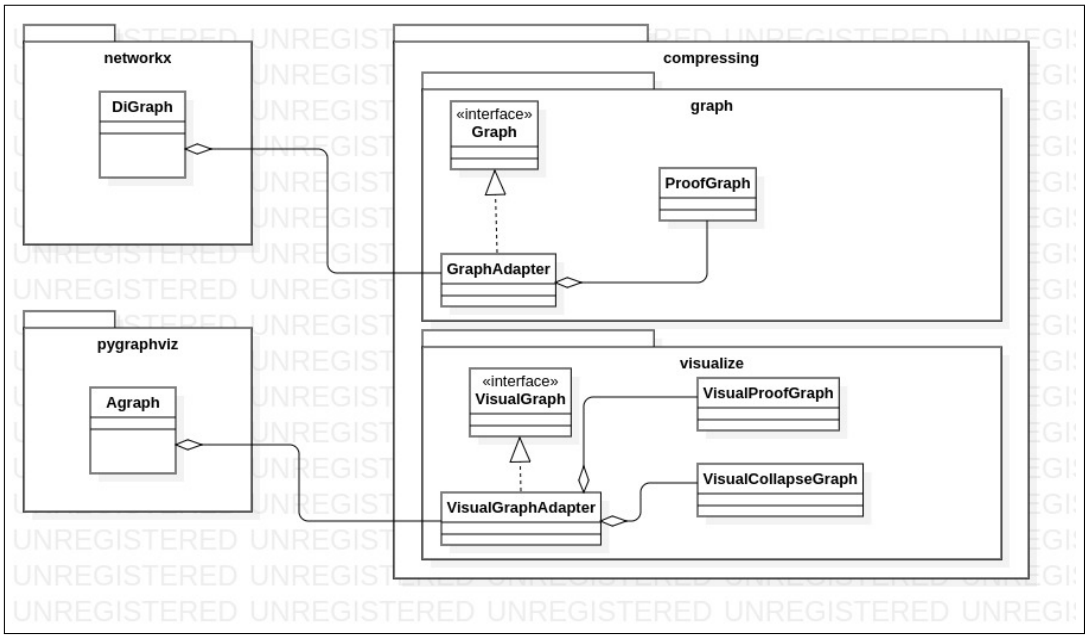


Figura 5.2: Diagrama de classes simplificado do *compressing*

PUC-Rio - Certificação Digital Nº 1712666/CA

– Cada vértice do grafo é identificado por um *id* e possui um atributo *label*, que tem como valor a fórmula a qual o vértice é associado.
– Cada aresta de descarte possui um atributo *comment* com o valor *discharge*.
– As arestas dedutivas não possuem qualquer informação adicional.

Cada arquivo de saída possui as informações da EGDD adicionadas durante a compressão na forma de atributos de grafo, vértices e arestas. Cada arquivo gerado pelo *compressing* segue as seguintes convenções:

– A representação dos vértices é idêntica à dos arquivos de entrada.
– As arestas dedutivas que **não foram** colapsadas durante a compressão possuem três atributos: *collapsed*, que possui valor *False* indicando que a aresta não foi colapsada; *color*, que possui como valor o numeral que indica sua cor; *dependencies*, que possui como valor a cadeia de bits que indicam a dependência da fórmula do vértice que é sua origem.
– As arestas dedutivas que **foram** colapsadas durante a compressão possuem dois atributos: *collapsed*, que possui valor *True* indicando que a aresta foi colapsada; *lambda-colors*, que possui como valor as cores das arestas dedutivas que a originaram.
– As arestas de ancestralidade possuem apenas um atributo, *path*, que possui uma lista de cores que representa o caminho, através de arestas dedutivas, entre o vértice que é seu destino e o vértice que é a sua origem.

– O grafo possui um atributo, *order*, que possui como valor a lista de fórmulas representando a ordenação linear utilizada para a construção das cadeias de bits de dependências.

PUC-Rio - Certificação Digital Nº 1712666/CA

6
Experimentos

Este capítulo apresenta os resultados dos experimentos das técnicas de compressão aplicadas a um conjunto de provas de tautologias da M⊃.

6.1
Ambiente Computacional

Todos os experimentos relatados nesta Seção foram executados em uma máquina com processador Intel Core i3-6100U 2,30GHz, 8GB de memória RAM e com sistema operacional Ubuntu 18.04.1 LTS. O *gerador de fórmulas*[1] descrito na Seção 6.2 e a *implementação da codificação de Huffman*[1], cujos resultados de compressão são mostrados na Seção 6.3, foram implementados utilizando a linguagem *Python*.

6.2
Provas de Entrada

Provas com tamanho super-polinomial em relação ao tamanho da conclusão são redundantes, quanto maior a prova, maior a quantidade de redundâncias (26). Em um árvore de prova, essas redundâncias são ramos idênticos que se repetem ao longo da prova no mesmo nível da derivação. Portanto, para a Compressão Horizontal, quanto maior a prova, maior a quantidade de colapsos e consequentemente, maior a taxa de compressão. Como observado no Capítulo 4, quando aplicada a provas que permitem poucos colapsos, a CH gera provas maiores que as originais. Logo, o conjunto de provas utilizadas nos experimentos deve ser necessariamente composto por provas grandes, que possuem tamanhos exponenciais em relação ao tamanho da conclusão, para que seja possível mensurar a capacidade de compressão da CH.

Inicialmente, consideramos três famílias de fórmulas para serem utilizadas nos experimentos. A primeira família de fórmulas ψ_n definida em (3), não possui provas normais, para qualquer $n > 0$, com menos de 2^n hipóteses descartadas. ψ_n é definida com segue:

Definição 6.1. Fórmulas Haeusler. Seja $\chi[X, Y] = (((X \supset Y) \supset X) \supset X) \supset Y$. Considere os símbolos proposicionais C e D_i, para $i > 0$. A família

[1] https://github.com/flavio-barros/proof-compressions

PUC-Rio - Certificação Digital Nº 1712666/CA

de fórmulas ξ_i é definida recursivamente como segue:

$$\xi_1 = \chi[D_1, C]$$
$$\xi_{i+1} = \chi[D_{i+1}, \xi_i]$$

Utilizando ξ_i, a família de fórmulas ψ_n, para $n > 0$, é definida como segue, para $i \geq 1$:

$$\psi_{i+1} = \xi_{i+1} \supset C$$

A segunda família de fórmulas foi retirada da Biblioteca ILTP[2] (*Intuitionistic Logic Theorem Proving (ILTP) Library*) que fornece uma plataforma para testes para provadores de teoremas da lógica proposicional e de primeira ordem intuicionista. Os problemas da biblioteca estão agrupados em 24 domínios, incluindo o SYJ, domínio dos problemas sintáticos intuicionistas (27). Um dos problemas, o SYJ204, refere-se a uma família de fórmulas ϕ_n, tendo apenas a implicação como conectivo, que possui somente provas normais de tamanhos exponenciais. ϕ_n é definida como segue:

Definição 6.2. Fórmulas SYJ204. Sejam A_i, com $i \geq 0$, símbolos proposicionais. A família de fórmulas ω_i é definida recursivamente como segue:

$$\omega_1 = (A_1 \supset (A_1 \supset A_0))$$
$$\omega_{i+1} = \omega_i \supset (A_{i+1} \supset (A_{i+1} \supset A_i))$$

Utilizando ω_i, a família de fórmulas ϕ_n, para $n > 0$, é definida como segue, para $i \geq 1$:

$$\phi_{i+1} = A_{i+1} \supset (\omega_{i+1} \supset A_0)$$

A terceira família de fórmulas foi retirada de um exemplo de compressão em (6). Fib_n possui provas normais maiores ou iguais que *fibonacci(n)* e é definida como segue:

Definição 6.3. Fórmulas Fibonacci. Sejam A_i, com $i \geq 3$, símbolos proposicionais. A família de fórmulas σ_i é definida recursivamente como segue:

$$\sigma_3 = (A_1 \supset (A_2 \supset A_3))$$
$$\sigma_i = \sigma_{i-1} \supset (A_{i-2} \supset (A_{i-1} \supset A_n))$$

[2]http://www.iltp.de/

PUC-Rio - Certificação Digital Nº 1712666/CA

Utilizando σ_i, a família de fórmulas Fib_n, para $n > 2$, é definida como segue:

$$Fib_n = (A_1 \supset A_2) \supset (\sigma_n \supset (A_1 \supset A_n))$$

As provas utilizadas nos experimentos foram geradas utilizando o provador de teoremas *NatDProver*[3,4], desenvolvido no TecMF[5], que recebe fórmulas da M$\supset$ e devolve um arquivo *.dot* que contém o grafo de prova caso a prova seja gerada com sucesso. O NatDProver ainda está em desenvolvimento, alguns erros e comportamento inesperados podem ocorrer durante a geração das provas.

Executamos o NatDProver para instâncias das três famílias de fórmulas selecionadas para os experimentos. O provador pode apresentar cinco tipos de retorno em cada execução: *sucesso*, geração da prova e do arquivo *.dot* ocorreram com sucesso; *erro geração DOT*, a geração da prova ocorreu com sucesso, mas a geração do arquivo *.dot* apresentou erro; *erro geração prova*, ocorreu um erro na geração da prova; *falha geração prova*, a geração da prova falhou, mas não ocorreram erros durante a execução; *timeout*, o tempo de execução do provador atingiu o limite de tempo definido.

A Tabela 6.1 mostra o retorno obtido do NatDProver para cada instância das famílias de fórmulas consideradas. Executamos o provador para 20 instâncias de cada família, o limite de tempo de execução para cada instância foi de 30 minutos.

Tabela 6.1: Retornos do NatDProver para cada instâncias das famílias de fórmulas

Família de fórmulas	**Valor de n**	**Retorno**
ψ_n (fórmulas Haeusler)	1	*erro geração DOT*
	2 - 20	*falha geração prova*
ϕ_n (SYJ204)	1	*sucesso*
	2 - 4	*erro geração DOT*
	5 - 20	*falha geração prova*
Fib_n (fórmulas Fibonacci)	3 - 15	*sucesso*
	16 - 22	*timeout*

Após as execuções, o NatDProver não gerou nenhum arquivo *.dot* com provas das fórmulas Haeusler, um arquivo para as fórmulas SYJ204 e 13 arquivos para as fórmulas Fibonacci. Como o objetivo do experimento é

[3]https://github.com/Bpalkmim/NatDProver (Implementação original)

[4]https://github.com/flavio-barros/NatDProver (Implementação alterada para aceitar entradas por linha de comando)

[5]http://www.tecmf.inf.puc-rio.br/

PUC-Rio - Certificação Digital Nº 1712666/CA

observar as taxas de compressão das provas obtidas por cada técnica a medida que o tamanho das provas aumenta, utilizamos apenas as fórmulas Fibonacci para gerar os resultados mostrados na próxima seção.

6.3 Resultados

Nos gráficos seguintes, as provas das fórmulas de Fib_n são mostradas em termos dos tamanhos das fórmulas no lugar dos valores de n, a prova da fórmula de $n = 3$ possui a conclusão com tamanho 13, $n = 4$ possui tamanho 19, e assim por diante, até $n = 15$ com a conclusão com tamanho 85.

A Figura 6.1 mostra o gráfico que relaciona os tamanhos das fórmulas da família Fib_n, considerando a quantidade de símbolos proposicionais, e os tamanhos de suas respectivas provas, considerando o tamanho do grafo de prova. Na Figura 6.2, os tamanhos das provas são mostrados considerando os tamanhos (em *kilobytes*) dos arquivos DOT que as representa.

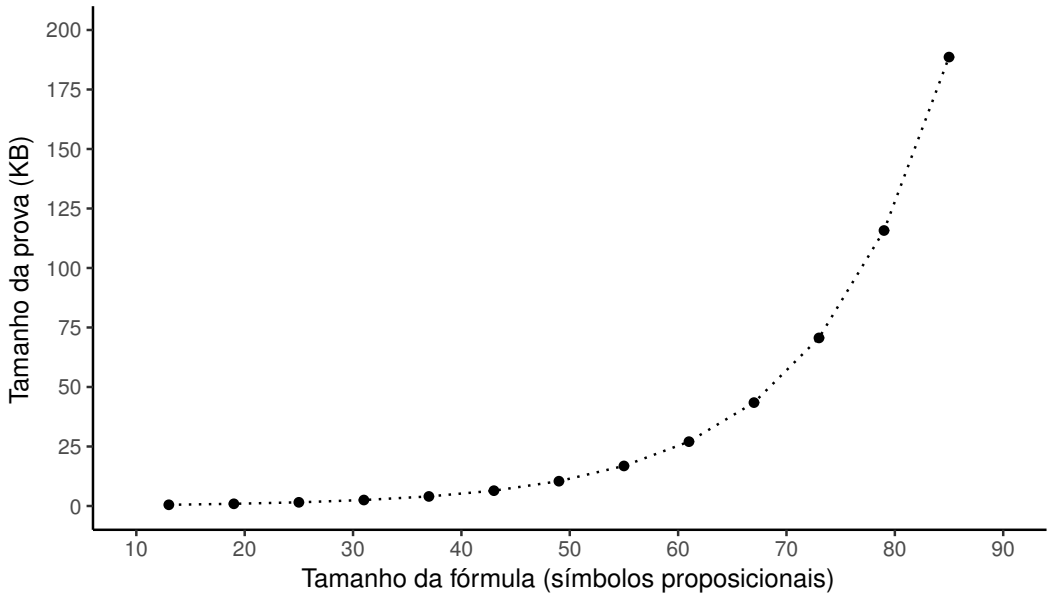


Figura 6.1: Tamanho das fórmulas e seus respectivos grafos de prova

A seguir apresentamos os resultados das aplicações dos algoritmos de compressão às provas das fórmulas de Fib_n. Os resultados são mostrados para os algoritmos da codificação de Huffman e da Compressão Horizontal.

A Figura 6.3 mostra os tamanho dos arquivos originais e dos resultantes dos algoritmos de compressão, o eixo dos tamanhos dos arquivos está em escala logarítmica. Como já esperado, a Compressão Horizontal não possui bons resultados para as provas menores, apresentando uma taxa de compressão

PUC-Rio - Certificação Digital Nº 1712666/CA

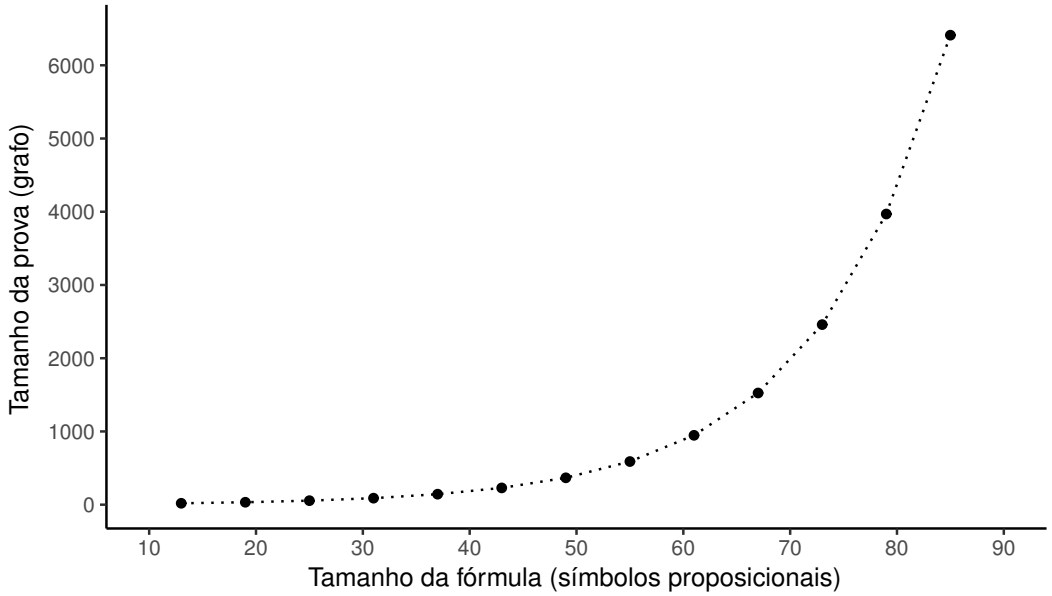


Figura 6.2: Tamanho das fórmulas e seus respectivos arquivos de prova

PUC-Rio - Certificação Digital Nº 1712666/CA

efetiva apenas a partir da fórmula com tamanho 37 ($n = 7$). A partir da fórmula com tamanho 30 ($n = 8$), a Compressão Horizontal obteve resultados melhores que a codificação de Huffman. A maior prova, da fórmula com tamanho 85 ($n = 15$), possui tamanho original 188,6 KB e foi comprimida para 113 KB pela codificação de Huffman e para 9,36 KB pela Compressão Horizontal.

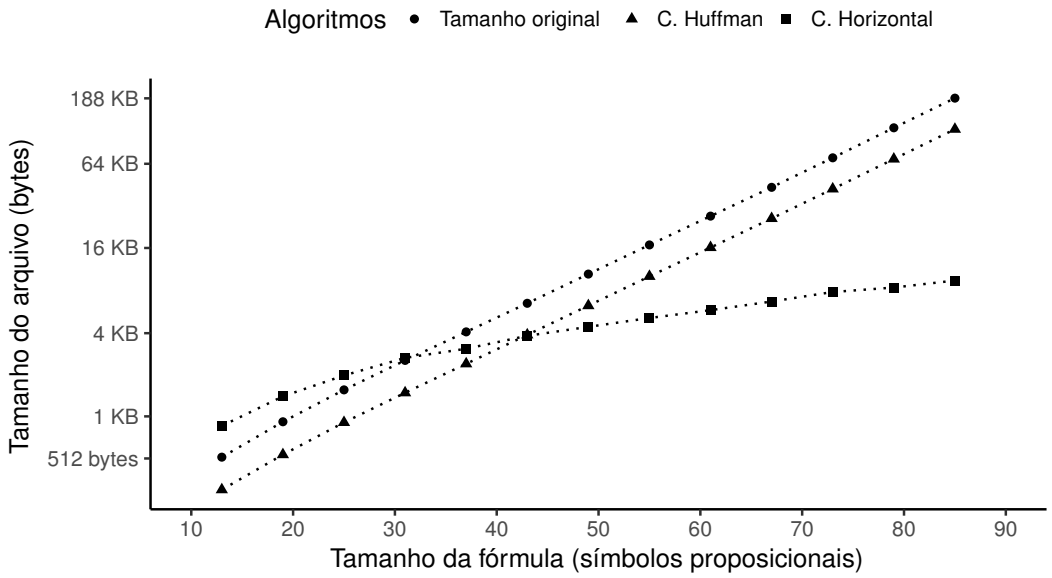


Figura 6.3: Tamanhos dos arquivos de prova após a compressão

A Figura 6.4 mostra as taxas de compressão obtidas para cada fórmula. Para todas as provas, a codificação de Huffman obteve taxa de compressão de aproximadamente 40%, enquanto que as taxas de compressão da Compressão Horizontal variam bastante de acordo com o tamanho da prova. Para a fórmula de tamanho 13 ($n = 3$), a taxa de compressão obtida foi de -67% (aumento de 67% do tamanho do arquivo original) e para a prova da fórmula de tamanho 85 ($n = 15$), a taxa de compressão obtida foi de 95%.

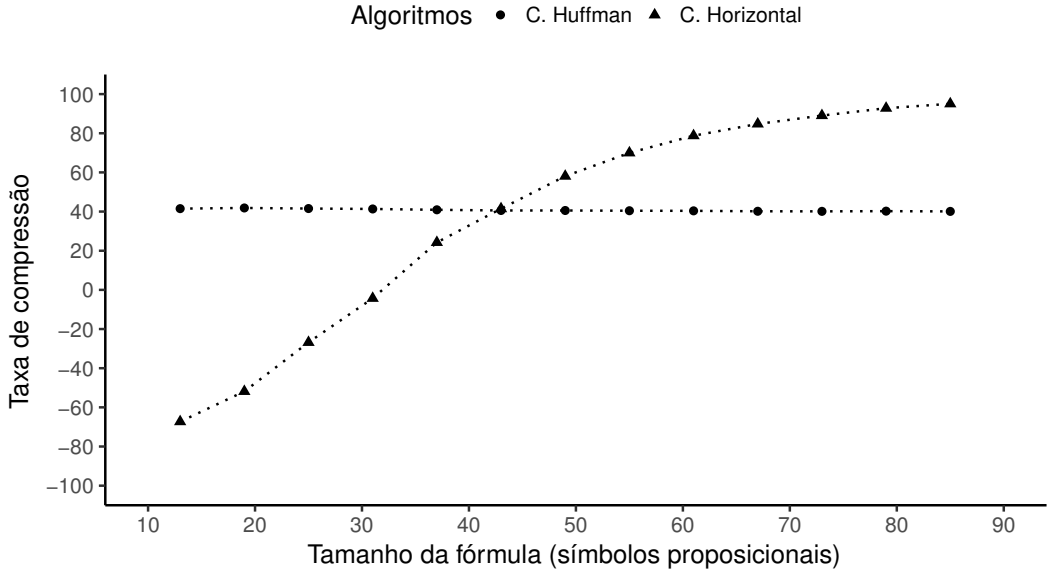


Figura 6.4: Taxa de compressão dos algoritmos

Enquanto que para as taxas de compressão, a Compressão Horizontal obteve resultados mais satisfatórios, para os tempos de execução, a codificação de Huffman apresentou tempos menores para todas as fórmulas. O gráfico da Figura 6.5 mostra os tempos de execuções dos dois algoritmos para todas as provas, o eixo dos tempos de execução está em escala logarítmica.

PUC-Rio - Certificação Digital Nº 1712666/CA

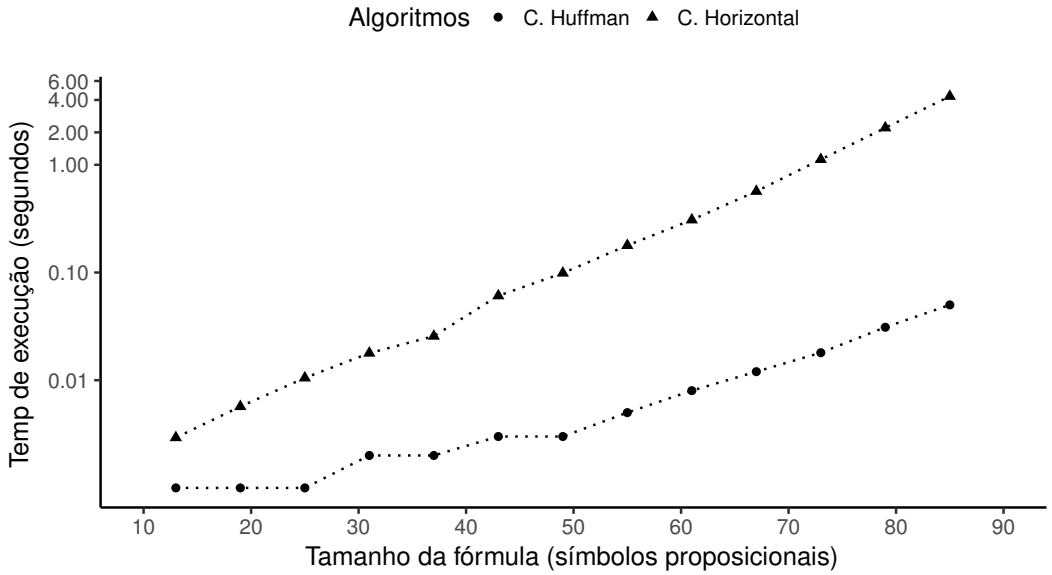


Figura 6.5: Tempo de execução dos algoritmos de compressão

PUC-Rio - Certificação Digital Nº 1712666/CA

7
Conclusão e Trabalhos Futuros

Com o objetivo inicial de implementar a Compressão Horizontal e comparar seu desempenho na compressão de provas em dedução natural da M$\supset$ com outras técnicas de compressão relatadas na literatura, concluímos esta dissertação com o objetivo parcialmente atingido. Identificamos na literatura apenas a Dedução Natural Contextual (4) como técnica de compressão de provas em dedução natural da $M \supset$, no entanto, seu provador (5) não foi capaz de gerar nenhuma prova das fórmulas selecionadas para o experimento.

Implementamos o *compressing*, compressor de provas em Dedução Natural da M$\supset$ que utiliza o algoritmo da Compressão Horizontal, e propomos formatos e convenções para os arquivos que contêm as provas submetidas à compressão e para os arquivos que contêm as provas comprimidas. Projetamos o compressor como o padrão de projetos *adapter* para que possíveis alterações de componentes externos (manipulação e visualização de grafos) sejam facilitadas.

Na preparação dos experimentos, selecionamos famílias de fórmulas na literatura com as características adequadas para mostrar a capacidade de compressão da Compressão Horizontal, ou seja, fórmulas que possuem provas grandes, preferencialmente, com tamanho exponencial em ralação ao tamanho da conclusão. Entre as famílias de fórmulas selecionadas, o provador utilizado, o NatDProver, gerou provas apenas para a Fib_n.

Nos resultados dos experimentos, reportamos as informações das execuções da Compressão Horizontal e da codificação de Huffman para as fórmulas de Fib_n. A Compressão obteve taxas de compressão de até 95%, enquanto que a codificação de Huffman obteve aproximadamente 40% de taxa de compressão para todas as provas.

Nossa principal contribuição é a implementação do *compressing*, que implementa o algoritmo da Compressão Horizontal, capaz de comprimir qualquer prova da M$\supset$ para um tamanho polinomialmente limitado em relação ao tamanho da conclusão. Se qualquer tautologia da M$\supset$ possui provas com tamanho polinomialmente limitado, então $NP = PSPACE$. No entanto, a base de provas utilizadas para a obtenção dos resultados das taxa de compressão é limitada, sendo composta apenas por fórmulas que compartilham a mesma

PUC-Rio - Certificação Digital Nº 1712666/CA

estrutura.

Listamos a seguir os possíveis trabalhos futuros:

- Diversificar a base de provas para os experimentos do *compressing*, adaptando um outro provador já existente ou corrigindo as falhas do NatDProver.
- Adicionar o algoritmo de verificação da Compressão Horizontal, que verifica se a derivação comprimida é válida, ao *compressing*.
- Melhorar a interação do usuário com o *compressing*, implementando uma interface por linha de comando.
- Otimizar os algoritmos internos do *compressing* para melhorar os resultados de tempos de execução.

PUC-Rio - Certificação Digital Nº 1712666/CA

Referências bibliográficas

[1] STATMAN, R.. **Intuitionistic propositional logic is polynomial-space complete**. Theoretical Computer Science, 9(1):67–72, 1979.

[2] HAEUSLER, E. H.. **Propositional logics complexity and the sub-formula property**. Electronic Proceedings in Theoretical Computer Science, 179, 01 2014.

[3] HAEUSLER, E. H.. **How many times do we need an assumption to prove a tautology in minimal logic? examples on the compression power of classical reasoning**. Electronic Notes in Theoretical Computer Science, 315:31–46, 2015.

[4] PALEO, B. W.. **Contextual natural deduction**. In: Artemov, S.; Nerode, A., editors, LOGICAL FOUNDATIONS OF COMPUTER SCIENCE, p. 372–386, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[5] PALEO, B. W.. **Implementation and evaluation of contextual natural deduction for minimal logic**. In: INTERNATIONAL ANDREI ERSHOV MEMORIAL CONFERENCE ON PERSPECTIVES OF SYSTEM INFORMATICS, p. 314–324. Springer, 2015.

[6] GORDEEV, L.; HAEUSLER, E. H.. **NP vs PSPACE**. CoRR, 2016.

[7] VYSKOČIL, J.; STANOVSKÝ, D. ; URBAN, J.. **Automated proof compression by invention of new definitions**. In: INTERNATIONAL CONFERENCE ON LOGIC FOR PROGRAMMING ARTIFICIAL INTELLIGENCE AND REASONING, p. 447–462. Springer, 2010.

[8] AMJAD, H.. **Data compression for proof replay**. Journal of Automated Reasoning, 41(3-4):193–218, 2008.

[9] BOUDOU, J.; FELLNER, A. ; PALEO, B. W.. **Skeptik: A proof compression system**. In: INTERNATIONAL JOINT CONFERENCE ON AUTOMATED REASONING, p. 374–380. Springer, 2014.

[10] BUSS, S. R.. **Handbook of proof theory**, volume 137, chapter 1. An Introduction to Proof Theory. Elsevier, 1998.

PUC-Rio - Certificação Digital Nº 1712666/CA

[11] MARFORI, M. A.. **Informal proofs and mathematical rigour**. Studia Logica, 96(2):261–272, 2010.

[12] FREGE, G.. **Conceitografia: Uma linguagem formular do pensamento puro decalcada sobre a da aritmética**. PPGFIL-UFRRJ, 1 edition, 2018.

[13] VON PLATO, J.. **The development of proof theory**. In: Zalta, E. N., editor, THE STANFORD ENCYCLOPEDIA OF PHILOSOPHY. Metaphysics Research Lab, Stanford University, winter 2018 edition, 2018. URL = `https://plato.stanford.edu/archives/win2018/entries/proof-theory-development/`.

[14] LEARY, D. J. O.. **The propositional logic of "principia mathematica"and some of its forerunners**. Russell: The Journal of Bertrand Russell Studies, 8:92, 1988.

[15] BERNAYS, P.. **Axiomatische untersuchung des aussagen-kalkuls der "principia mathematica"**. Mathematische Zeitschrift, 25(1):305–320, 1926.

[16] ZACH", R.. **Hilbert's program then and now**. In: Jacquette, D., editor, PHILOSOPHY OF LOGIC, Handbook of the Philosophy of Science, p. 411–447. North-Holland, Amsterdam, 2007.

[17] ZACH, R.. **Hilbert's program**. In: Zalta, E. N., editor, THE STANFORD ENCYCLOPEDIA OF PHILOSOPHY. Metaphysics Research Lab, Stanford University, spring 2016 edition, 2016. `https://plato.stanford.edu/archives/spr2016/entries/hilbert-program/`.

[18] GENTZEN, G.. **Investigations into logical deduction. tradução publicada em m. szabo the collected papers of gerhard gentzen**. American philosophical quarterly, 1(4):288–306, 1964.

[19] KAHLE, R.; RATHJEN, M.. **Gentzen's Centenary: The Quest for Consistency**. Springer, 2015.

[20] PRAWITZ, D.. **Natural Deduction: A Proof-Theoretical Study**. Dover Publications, 1965.

[21] KRIPKE, S. A.. **Semantical analysis of intuitionistic logic I**. In: Crossley, J.; Dummett, M., editors, FORMAL SYSTEMS AND RECURSIVE FUNCTIONS, volume 40, p. 92–130. Elsevier, 1965.

PUC-Rio - Certificação Digital Nº 1712666/CA

[22] HAZEN, A.; PELLETIER, F. J.. **Gentzen and Jaśkowski natural deduction: Fundamentally similar but importantly different**. Studia Logica, 102(6):1103–1142, 2014.

[23] STANDEFER, S.. **Translations between Gentzen–Prawitz and Jaśkowski–Fitch natural deduction proofs**. Studia Logica, p. 1–32, 2018.

[24] FITCH, F. B.. **Symbolic Logic**. New York: Ronald Press Co., 1952.

[25] HUFFMAN, D. A.. **A method for the construction of minimum-redundancy codes**. Proceedings of the IRE, 40(9):1098–1101, 1952.

[26] GORDEEV, L.; HAEUSLER, E. H.. **Huge proofs, redundant proofs and some reasons in favor of NP=PSPACE**. In: EM III TüBINGEN CONFERENCE ON PROOF-THEORETIC SEMANTICS, Tubinga, Alemanha, Mar 2019.

[27] RATHS, T.; OTTEN, J. ; KREITZ, C.. **The ILTP problem library for intuitionistic logic**. Journal of Automated Reasoning, 38(1-3):261–271, 2007.

[28] GORDEEV, L.; HAEUSLER, E. H.. **Horizontal compression of dag-derivations in $\mathbf{m}\supset$**. Manuscrito não publicado.

PUC-Rio - Certificação Digital Nº 1712666/CA