Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

PROJETO DE UM ASSISTENTE DE PROVAS PARA OS N-GRAFOS

Laís Sousa de Andrade

TRABALHO DE GRADUAÇÃO

Recife
12 de julho de 2012

Universidade Federal de Pernambuco
Centro de Informática

Laís Sousa de Andrade

PROJETO DE UM ASSISTENTE DE PROVAS PARA OS N-GRAFOS

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: *Anjolina Grisi de Oliveira*

Recife
12 de julho de 2012

Dedico este trabalho aos meus pais, Amaury e Ziza, por todo o apoio que sempre me deram.

AGRADECIMENTOS

Quero agradecer a todos que de alguma forma me ajudaram durante a minha graduação:

à meus pais, Amaury e Ziza, por todo o suporte material e emocional que me deram, não apenas durante o periodo da graduação. É graças ao esforço deles que eu pude me dedicar plenamente a este curso, e foi com seu apoio e compreensão durante os momentos de dificuldade que obtive força e determinação pra chegar até aqui.

à meu irmão Felipe, que sempre esteve presente, torcendo por mim e me ajudando, especialmente quando abdicava do acesso ao computador para meu uso quase que exclusivo.

à minha orientadora, Anjolina, que me indicou um caminho e me ajudou a trilhá-lo, sempre com muita dedicação, paciência e carinho.

aos meus colegas da maratona de programação, que me ajudaram a construir grande parte do meu conhecimento adquirido na graduação. Um agradecimento especial à professora Liliane Salgado, que conduz o projeto com muita dedicação, e sempre esteve ao meu lado me dando força e me orientando em minhas escolhas. Outro agradecimento especial a meus companheiros de time Renan Pires e Filipe Martins, que partilharam comigo um ano de trabalho duro, alegrias e aprendizado. E a todos os outros participantes da época, que com maior ou menor intensidade fizeram parte desta etapa da minha graduação.

aos meus amigos de turma, que partilharam comigo muitas noites no CIn durante os projetos. Entre eles, Amora, Anália, Caio, Irineu, Ivson e Lorena. Juntos passamos por grandes momentos de estresse e dificuldades, mas outros ainda maiores de alegria, divesão e conquistas.

a Ruan Carvalho, que trabalhou comigo durante boa parte da elaboração deste trabalho enquanto trabalhava com sua tese de mestrado. Juntos, sob a orientação da professora Anjolina, pudemos entrar no mundo dos N-grafos e desenvolver nossos trabalhos de maneira colaborativa.

RESUMO

Na lógica clássica existem diversos sistemas de formalização de provas. Um deles é a *Dedução Natural*, onde as regras de derivação são definidas da maneira mais *natural* possível, simulando o raciocínio humano. Baseando-se nele, de Oliveira apresenta no seu trabalho *"Proofs From a Geometric Perspective"* os *N-grafos*: um sistema de prova com mútiplas conclusões para o cálculo clássico proposicional. Tal sistema visa o estudo dos aspectos estruturais de provas.

Apesar da naturalidade do cálculo de dedução natural, ele pouco influenciou o desenvolvimento de provadores automáticos. Para este fim, outros formalismos fundamentados no trabalho de Herbrand e no cálculo de sequentes de Gentzen tem sido mais influentes. Este trabalho tem como objetivo projetar um assistente de provas para os N-grafos, desenvolvendo um mecanismo automático de prova para este sistema. Para isso, serão apresentadas estratégias para a *verificação* e *geração* de provas para o fragmento com a $\neg, \wedge$ e $\vee$ dos N-grafos.

Palavras-chave: N-grafos, grafo de prova, dedução natural, assistente de provas, provador automático de teoremas, resolução.

ABSTRACT

There are many formalism for proof in classical logic. One of them is the *Natural Deduction*, where rules and derivations are defined as natural as possible, coming very close to the human reasoning. Based on it, de Oliveira presents on her work called *"Proofs From a Geometric Perspective"* the *N-graphs*: A multiple conclusion proof system for classic propositional logic. This system aims the study of structural aspects of proofs.

Despite being a very natural calculus, the natural deduction has little influence on automated proving systems. Other formalisms grounded on the work of Herbrand and on Gentzen' sequent calculus has been fundamental for this purpose. This work aims to design a proof assistant for N-graphs, developing an automated proof generation mechanism. To achieve that, strategies for proof *verification* and *generation* will be made for the $\neg, \wedge$ and $\vee$ N-graph fragment.

Keywords: N-graphs, proof graph, natural deduction, proof assistant, automated theorem proving, resolution.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

Assistentes de prova são sistemas de computação que auxiliam no desenvolvimento de provas e definições formais. Estas provas são construídas através de uma colaboração entre homem e máquina, de maneira interativa. O sistema é formado por uma interface de edição, pela qual o usuário pode guiar o processo da construção da prova, e um núcleo de processamento, que provê alguns passos da prova automaticamente e verifica a corretude da mesma. O escopo desta ferramenta pode variar desde a expressão de provas matemáticas e teoremas, como no Coq [Coq12], Isabelle [Isa12] e no Yarrow [Yar12], até a verificação formal de software e hardware, como no ACL2 [ACL12].

No campo da lógica clássica existem diversos sistemas de formalização de provas de sentenças. Um deles é a *Dedução Natural*, proposta por Gentzen no seu trabalho *"Investigations into Logical Deduction"* [Gen35]. Neste sistema, as regras de derivação são definidas da maneira mais *natural* possível, simulando o raciocínio humano para a dedução lógica de sentenças a partir de hipóteses.

Baseando-se na dedução natural, de Oliveira apresenta no seu trabalho *"Proofs From a Geometric Perspective"* [dO01] um sistema de prova com mútiplas conclusões para o cálculo clássico proposicional, chamado *N-grafos*. Tal sistema tem um foco diferente dos demais cálculos de mútiplas conclusões, como os propostos por Kneale [Kne58] e Ungar [Ung92]. Ele visa o estudo dos aspectos estruturais de provas formais da dedução natural, e por isso é definido na forma de grafos direcionados. Assim como no cálculo de Gentzen, as derivações nos N-grafos tentam representar de maneira mais natural o processo de dedução.

Apesar da naturalidade do cálculo de dedução natural, ele pouco influenciou o desenvolvimento de provadores automáticos de teoremas. Para este fim, outros formalismos fundamentados no trabalho de Herbrand e no cálculo de sequentes de Gentzen tem sido mais influentes. Por que este cálculo não foi tão explorado para a construção automática de provas? Sieg em [SB98] afirma que a resposta para esta pergunta deve observar três

pontos importantes: (1) como especificar por um cálculo apenas provas normalizadas; (2) como construir um espaço de busca que permita a definição de estratégias para encontrar provas e (3) como provar que tais estratégias sempre terminam.

Sieg et al. em [SS92] e [SB98] apresentam uma técnica de geração de provas para a dedução natural, que é utilizada por eles para a definição de um *Proof Tutor*, que auxilia na construção de tais provas. Baseado no trabalho dele, e também nas técnicas tradicionais como no *Princípio da Resolução*, este trabalho apresenta um conjunto de estratégias para a *verificação* e *geração* de provas para o sistema de N-grafos, no fragmento com a $\neg, \wedge$ e $\vee$.

1.1 ESTRUTURA DO TRABALHO

Esta monografia está dividida em oito capítulos. O Capítulo 2 apresenta uma introdução à assistentes de prova, com alguns conceitos importantes envolvidos com a geração automática de provas. O Capítulo 3 apresenta uma introdução à Dedução Natural de Gentzen, fazendo um destaque dos aspectos positivos e dos pontos fracos de tal sistema. Neste ponto, o Capítulo 4 entra propondo soluções para alguns dos problemas apontados no capítulo anterior, através da definição de cálculos de múltipla conclusão para as lógicas clássica e intuicionista.

No Capítulo 5 é apresentado o sistema dos *N-grafos* como definido por de Oliveira. Neste capítulo, os aspectos importantes que garantem a corretude do cálculo são mostrados com mais detalhes, já orientados para a definição de estratégias para verificação de tais provas, que são apresentadas no Capítulo 6. Neste capítulo, uma nova forma de verificação mais eficiente do critério de corretude apresentado por de Oliveira em [dO01] é proposta para o fragmento $\neg, \wedge$ e $\vee$, juntamente com o algoritmo para esta verificação.

No último capítulo, duas formas de geração de provas no sistema dos N-grafos para o fragmento $\neg, \wedge$ e $\vee$ são apresentadas, sendo uma delas baseada no método da resolução e a outra na técnica proposta por Sieg et al. em [SS92] para a dedução natural.

1.2 TERMINOLOGIA

Neste trabalho, o foco será o tratamento da lógica proposicional, uma vez que os N-grafos foram definidos para a mesma. A seguinte terminologia será adotada:

- Constantes proposicionais: $\bot$ (a proposição *verdadeiro*) e $\top$ (a proposição *falso*)
- Conectivos: $\neg$, $\vee$, $\wedge$ e $\rightarrow$
- Fórmulas proposicionais: $A, B, C, \ldots$

Obs.: por vezes, os símbolos que representam fórmulas podem ser aplicados no contexto de "ocorrência de fórmulas" (representando simplesmente uma ocorrência daquela fórmula na derivação). Letras minúsculas $u, v, w, \ldots$ podem também aparecer em algum contexto representando tais ocorrências, principalmente quando o aspecto geométrico de uma prova nos N-grafos for mais relevante.

CAPÍTULO 2

PROVAS ASSISTIDAS POR COMPUTADOR

De acordo com [CL73], um *teorema* pode ser definido como a afirmação de que uma fórmula deriva de um conjunto de outras fórmulas por meio de um raciocínio lógico. Uma demonstração de tal afirmação é chamada *prova* do teorema. O *problema da prova automática de teoremas* é considerar métodos técnicos para encontrar provas de teoremas. Uma prova assistida por computador entra neste contexto como uma maneira de construir uma prova utilizando alguns passos gerados automaticamente, mas contando com o auxílio de um humano para guiar o processo de busca e construção.

Uma prova assistida é muito mais flexível do que uma gerada automaticamente. Nesta última, o mecanismo de prova consiste de um conjunto de procedimentos de decisão fechado que permite que fórmulas em um formato restrito possam ser provadas automaticamente. Por ter tal estrutura, tais sistemas possuem expressividade limitada. Já uma prova assistida tem apenas um conjunto de técnicas para abordar certos passos de uma prova automaticamente, mas não todos. Parte das decisões tomadas durante a construção da prova vêm de um usuário humano, que interage com o sistema.

2.1 O PAPEL DAS PROVAS

Geuvers em [Geu09] define dois papéis para uma prova na matemática:

i) Uma prova *convence* o leitor de que a afirmação é correta.

ii) Uma prova *explica* porque a afirmação é correta.

O primeiro papel consiste de atividades metódicas de verificação da corretude dos pequenos passos de dedução e da conexão entre eles, validando a prova como um todo. Neste papel, o computador entra como uma ferramente valiosa a partir da definição de procedimentos de *verificação de provas* (do inglês, *proof checking*). Os primeiros assistentes de prova eram constituídos simplemente deste passo: eles recebiam como entrada um texto formatado em uma sintaxe específica e verificavam se o mesmo era uma prova.

Um teorema possui geralmente mais de uma prova. Isto mostra que uma prova não apenas verifica a corretude do teorema, mas também a explica. Um teorema pode ter diferentes explicações, onde cada uma mostra de um novo ponto de vista porquê a prova está correta. Este é o segundo papel de uma prova, e nele um assistente de prova entra como uma ferramenta de apoio, provendo alguns passos da dedução automaticamente.

Os fundamentos das técnicas de prova automática de teoremas foram introduzidos por Herbrand em 1930 [CL73]. O método sugerido na época era impossível de ser aplicado na prática, até a invenção do computado digital. Porém, foi apenas com o trabalho de Robinson em 1965, onde o mesmo introduz o *princípio da resolução*, que grandes passos foram tomados em direção a uma implementação mais realista de provadores de teoremas.

2.2 PRINCÍPIO DA RESOLUÇÃO

O princípio da resolução é uma regra de inferência que define uma técnica de prova por refutação. A idéia essencial do princípio é verificar se um conjunto S de sentenças da lógica proposicional ou de primeira ordem contém a cláusula vazia $\varnothing$. Se S contém $\varnothing$, então é insatisfatível. Senão, deve-se verificar se $\varnothing$ é derivável a partir de S por aplicações da seguinte regra de inferência:

$$\frac{a_1 \vee \cdots \vee a_i \vee \cdots \vee a_n \qquad b_1 \vee \cdots \vee b_j \vee \cdots \vee b_m}{a_1 \vee \cdots \vee a_{i-1} \vee a_{i+1} \vee \cdots \vee a_n \vee b_1 \vee \cdots \vee b_{j-1} \vee b_{j+1} \vee \cdots \vee b_m}$$

Os termos $a_i, 1 \leq i \leq n$ e $b_i, 1 \leq i \leq n$ são literais. Um *literal*, na lógica proposicional, é uma variável (chamada *átomo*) ou a negação de uma. Dois literais são ditos complementares se um é a negação do outro. A regra atua sobre duas cláusulas que contenham literais complemenares, gerando uma terceira sem estes literais. No caso acima, o literal a_i é complementar ao b_j. A cláusula produzida pela regra da resolução é chamada *resolvente*.

É provado que esta regra ao ser aplicada a C_1 e a C_2, gerando o resovente C_3, o faz de modo que C_3 é uma consequência lógica de C_1 e C_2. Com isso, podemos definir uma *dedução*.

Definição 2.1 (resolução/dedução). Dado um conjunto S de cláusulas, uma (*resolução*) *dedução* de C a partir de S é uma sequência finita $C_1, C_2, \ldots, C_k$ de cláusulas tal que

(1)	$P \vee Q$	
(2)	$\neg P \vee Q$	
(3)	$P \vee \neg Q$	
(4)	$\neg P \vee \neg Q$	
(5)	Q	$de(1)e(2)$
(6)	$\neg Q$	$de(3)e(4)$
(7)	$\varnothing$	$de(5)e(6)$

Tabela 2.1 Resolução de $S = \{P \vee Q, \neg P \vee Q, P \vee \neg Q, \neg P \vee \neg Q\}$

cada C_i ou é uma cláusula de S ou um resolvente de cláusulas precedento C_i, e $C_k = C$. Uma dedução de $\varnothing$ a partir de S é chamada uma *refutação*, ou uma *prova* de S.

Um exemplo de dedução pode ser visto na Tabela 2.1.

O princípio também pode ser aplicado para a lógica de primeira ordem. Para cláusulas contendo variáveis, a tarefa de encontrar cláusulas complementares fica mais complicada. Para resolver este problema, a técnica de *substituição* é utilizada, seguida da aplicação de um algoritmo para a *unificação* destes literais. Como este trabalho trata apenas da lógica proposicional, tais aspectos da resolução não serão detalhados.

O princípio da resolução é completo para as lógicas proposicional e de primeira ordem. Isto significa que um conjunto S de fórmulas é insatisfatível se e somente se (*sse*) existe uma dedução da cláusula vazia $\varnothing$ a partir de S.

Apesar de ser bastante eficiente comparada com as técnicas da época, a resolução ainda é capaz de gerar muitas cláusulas irrelevantes para a prova final. A fim de diminuir o número de cláusulas inúteis, várias estratégias foram desenvolvidas, cada uma com o seu mérito em remover certas classes de cláusulas irrelevantes a ser geradas. Entre elas, Chang e Lee em [CL73] destacam a *estratégia de deleção* e os seguintes refinamentos: *resolução semântica*, proposta por Slagle em 1967, que unifica outras técnicas previamente propostas; *resolução de lock* (do ingles, *lock resolution*), proposta por Boyer em 1971, que é uma técnica bastante eficiente; e a *resolução linear*, proposta independentemente por Loveland e Luckham em 1970.

Muitas das técnicas aplicadas hoje para prova automática de teoremas se baseiam na resolução. O assistente de provas *Isabelle* [Isa12] tem o como seu principal método de prova uma versão da resolução para a lógica de alto nível (em inglês, *Higher-order*

logic), baseado na *unificação em alto nível*. Existe ainda um sistema de inferência chamado *SLD resolution*, que é completo e correto para as cláusulas de Horn [1]. Esta técnica permite a verificação do problema de satisfatibilidade para cláusulas de Horn (chamado *HORNSAT*) em tempo linear, e por isso é muito utilizada para verificação automática de teoremas. Sistemas de inferência como o *Prolog* [SS94] utilizam estas regras para construir o seu motor de inferência.

2.3 CONSTRUÇÃO DE PROVAS

O papel principal de um asistente de prova é a construção de provas formais em algum sistema dedutivo. Para isto, a tradição teórica tem se fundamentado no trabalho de Herbrand e no cálculo de sequentes definido por Gentzen, desenvolvendo novas técnicas baseadas em *resolução*, *tableaux* e *programação lógica* principalmente.

A resolução e demais técnicas de refutação são bastante utilizadas por provadores automáticos de teoremas, mesmo que apenas internamente como motor de prova. Porém, o formato deste tipo de prova é bem diferente das provas matemáticas construtivas, que são mais utilizadas no meio acadêmico em geral. As provas que os assistentes devem ajudar a construir devem ser mais amigáveis ao usuário final, ou seja, mais próximas da maneira como um ser humano desenvolve provas formais.

Usualmente os assistentes definem uma linguagem própria para a expressão de sentenças matemáticas, como acontece com o *Coq* e o *Isabelle*. Ela tem como principal objetivo diminuir a distância entre a linguagem formal matemática e uma que possa ser entendida pelo computador. É por meio dela que fica estabelecida a comunicação do usuário com o núcleo de processamento: o usuário entra com fórmulas nesta linguagem, e o motor de inferência transforma de volta os resultados obtidos automaticamente para exibição.

Em outra vertente existe o trabalho desenvolvido por Sieg et al. para a criação do *AProS* (Automated Proof Search) [APr12]. Neste projeto eles propõem uma técnica para a geração de provas na *Dedução Natural* de Gentzen, buscam uma maneira de gerar

[1] Uma cláusula de Horn é uma cláusula com no máximo um literal positivo. Elas são nominadas em homenagem ao lógico Alfred Horn, que investigou as propriedades matemáticas de sentenças similares a estas. Claúsulas de Horn têm um papel básico em programação lógica, e são muito importantes para a lógica construtiva.

automaticamente derivações na dedução natural. O objetivo inicial deste trabalho foi desenvolver um tutor para fins didáticos [SS92], chamado de *Proof Tutor*, mas o projeto foi crescendo com o tempo e se tornou uma plataforma para investigação do cálculo de dedução natural.

Nos trabalhos [SS92] e [SB98], Sieg define um novo cálculo, chamado cálculo de intercalação (do inglês, *Intercalation Calculus*). Ele prova a sua corretude e completude para a lógica clássica proposicional. Este trabalho depois foi extendido para ser aplicado na lógica de predicados, intuicionista e minimal. Por lidar com a dedução natural clássica e permitir tanto a geração automática de uma prova quanto um método interativo de construção da mesma, esta técnica será vista com mais detalhes no Capítulo 7, onde uma variação da mesma será proposta.

CAPÍTULO 3

DEDUÇÃO NATURAL

Gentzen em [Gen35] apresenta um sistema de dedução que ele denomina *"natural"* para as lógicas intuicionista (NJ) e clássica (NK). Neste sistema formal não existem axiomas, como nos sistemas utilizados em provas matemáticas na época, mas apenas regras que permitem a introdução e eliminação de operadores lógicos. O objetivo deste sistema era definir uma formalização da dedução lógica, como desenvolvida por Frege, Russell e Hilbert, que chegasse o mais próximo possível do raciocínio humano.

De acordo com Gentzen, os formalismos desenvolvidos até então estavam muito distantes da maneira que provas formais matemáticas eram elaboradas na prática. Nestes sistemas, chamados por ele de cálculos logísticos, uma fórmula verdadeira é derivada a partir de uma sequência de fórmulas básicas, por meio da aplicação de algumas poucas formas de inferência. Este conjunto de fórmulas básicas são os *axiomas*, e qualquer construção formal que intenda provar uma proposição deve derivá-la destes axiomas iniciais.

Diferentemente destes sistemas, na dedução natural parte-se de hipóteses às quais derivações lógicas são aplicadas. Por meio destas derivações, o resultado pode se tornar independente das hipóteses. Logo, na dedução natural não é necessário um conjunto inicial de proposições (axiomas) para a construção de derivações. Externamente, esta é a diferença essencial entre estes dois cálculos.

3.1 DERIVAÇÕES

Provas em dedução natural são representadas por derivações, que são definidas a seguir:

Definição 3.1 (figura de inferência). Uma *figura de inferência*, escrita como

$$\frac{A_1 \quad A_2 \quad \dots \quad A_n}{B} \qquad (n \geq 1),$$

representa uma operação que pode ser aplicada sobre $A_1, \dots, A_n$, com o uso de um

dos operadores lógicos, para gerar B. As fórmulas $A_1, \ldots, A_n$ são chamadas *fórmulas superiores* e B é chamado *fórmula inferior* da figura de inferência.

Definição 3.2 (**figura de prova *(derivação)***)**.** Uma *figura de prova*, ou simplesmente *derivação*, consiste de um número de fórmulas (pelo menos uma) combinadas através de figuras de inferência da seguinte maneira:

i) cada fórmula é a fórmula inferior de no máximo uma figura de inferência;

ii) cada fórmula é uma fórmula superior de pelo menos uma figura de inferência, com a exceção de apenas uma, chamada *fórmula final*;

iii) o sistema de figuras de inferência não é circular, ou seja, não existe uma sequência de fórmulas $A_1, \ldots, A_n$, onde A_i é uma fórmula superior e A_{i+1} a fórmula inferior de uma figura de inferência, para $1 \leq i < n$, de modo que A_n é uma fórmula superior da figura de inferência da qual A_1 é a fórmula inferior.

As derivações tratadas pela dedução natural consistem de fórmulas arranjadas em uma *estrutura de árvore* onde:

1. cada fórmula é uma fórmula superior de no máximo uma figura de inferência;

2. todas as fórmulas que não são a fórmula inferior de nenhuma figura de inferência são chamadas *fórmulas iniciais* de uma derivação (folhas da árvore), e representam as *premissas* (ou hipóteses) da derivação;

3. a *fórmula final* (raiz da árvore) representa a conclusão (ou resultado) da derivação.

As figuras de inferências, chamadas regras de derivação, da dedução natural são apresentadas na Tabela 3.1. Elas podem ser divididas em dois tipos: regras de *introdução* e de *eliminação* de operadores lógicos. Prawitz afirma que esta organização das regras é feita de tal modo que os papeis dedutivos de cada conectivo ficam separados [Pra65].

A regras definidas para a lógica proposicional, que são $\forall - I$, $\forall - E$, $\exists - I$ e $\exists - E$ deve obedecer à uma série de restrições sobre as variáveis envolvidas nas fórmulas. Como o escopo deste trabalho é a lógica proposicional, tais regras serão negligenciadas daqui em diante.

As regras de introdução e eliminação da negação podem ser consideradas como um caso especial das regras sobre a implicação, uma vez que $\neg A$ pode ser visto como uma

representação da fórmula $A \to \bot$. Assim, as regras ¬-I e ¬-E podem ser eliminadas do cálculo, junto com o operador ¬.

Toda fórmula que está abaixo de uma premissa na derivação, porém acima de uma figura de inferência que descarta a mesma, é dita ser dependente desta premissa. As regras que permitem o descarte de hipóteses, adicionando premissas à lista de hipóteses descartadas, são $\to -I$, $\vee - E$, $\exists - E$ e $\neg - I$. Nestas regras, as fórmulas entre colchetes representam um número arbitrário de premissas que estão na mesma sub-árvore da derivação da fórmula superior da figura de inferência, e que estão sendo descartadas da prova da fórmula inferior. Por exemplo, na figura

$$\dfrac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \to B} \to -I$$

Temos o descarte de ocorrências da hipótese A (possivelmente nenhuma) sobre a dedução da fórmula $A \to B$. Girard em [Gir87] se refere a esta possibilidade de descarte de premissas definindo dois estados para uma fórmula: viva ou morta. Ele diz que uma fórmula está "viva" no seu estado natural, quando ela interfere de maneira ativa na prova, ou seja, quando outras fórmulas dependem dela. Ele define o estado "morta" para a fórmula que não tem mais um papel ativo na prova, ou seja, a conclusão da derivação não depende mais desta hipótese.

Girard também fez uma análise sobre a estrutura de árvore definida por Gentzen como sendo a estrutura de uma derivação. Ele observou que quando uma regra de descarte de hipóteses é aplicada, é necessário se manter o registro de que fórmulas foram "mortas" por esta regra, uma vez que esta escolha de fórmulas é arbitrária. Para manter esta informação é necessário ligar a regra às ocorrências que serão descartadas, mas com isso a estrutura final deixa de ser uma árvore. Gentzen tinha observado este detalhe em [Gen35], e propõe a atribuição de uma numeração apropriada para cada regra de eliminação de hipótese e as ocorrências canceladas. Semanticamente falando, esta atribuição equivale a uma ligação entre regra e hipóteses, que quebra toda a estrutura de árvore, como visto no exemplo da Tabela 3.2.

Introdução	Eliminação
$\dfrac{A \quad B}{A \wedge B}\ \wedge\!-I$	$\dfrac{A \wedge B}{A}\ \wedge\!-E_1 \qquad \dfrac{A \wedge B}{B}\ \wedge\!-E_2$
$\dfrac{A}{A \vee B}\ \vee\!-I_1 \qquad \dfrac{B}{A \vee B}\ \vee\!-I_2$	$\dfrac{A \vee B \qquad \begin{array}{cc}[A] & [B] \\ \vdots & \vdots \\ C & C\end{array}}{C}\ \vee\!-E$
$\dfrac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \to B}\ \to\!-I$	$\dfrac{A \quad A \to B}{B}\ \to\!-E$
$\dfrac{\begin{array}{c}[A] \\ \vdots \\ \bot\end{array}}{\neg A}\ \neg\!-I$	$\dfrac{A \quad \neg A}{\bot}\ \neg\!-E$
$\dfrac{P(a)}{\forall x\, P(x)}\ \forall\!-I$	$\dfrac{\forall x\, P(x)}{P(b)}\ \forall\!-E$
$\dfrac{P(a)}{\exists x\, P(x)}\ \exists\!-I$	$\dfrac{\exists x\, P(x) \qquad \begin{array}{c}[P(a)] \\ \vdots \\ C\end{array}}{C}\ \exists\!-E$

Tabela 3.1 Regras de inferência da Dedução Natural

$$\dfrac{\dfrac{\dfrac{}{A \vee (B \wedge C)}\,2 \qquad \dfrac{\dfrac{\dfrac{}{A}\,1}{A \vee B}\vee\text{-}I \quad \dfrac{\dfrac{}{A}\,1}{A \vee C}\vee\text{-}I}{(A \vee B) \wedge (A \vee C)}\wedge\text{-}I \qquad \dfrac{\dfrac{\dfrac{}{(B \wedge C)}\,1}{B}\wedge\text{-}E}{A \vee B}\vee\text{-}I \quad \dfrac{\dfrac{\dfrac{}{(B \wedge C)}\,1}{C}\wedge\text{-}E}{A \vee C}\vee\text{-}I}{(A \vee B) \wedge (A \vee C)}\wedge\text{-}I}{(A \vee B) \wedge (A \vee C)}\vee\text{-}E_1}{(A \vee (B \wedge C)) \rightarrow ((A \vee B) \wedge (A \vee C))}\rightarrow\text{-}I_2$$

Tabela 3.2 Exemplo de atribuição de números à regras de eliminação

3.2 CÁLCULO INTUICIONISTA X CLÁSSICO

O cálculo de dedução natural foi inicialmente definido por Gentzen para a lógica intuicionista, no que ele chamou de cálculo NJ. Para isso, ele adicionou às regras definidas na Tabela 3.1 a figura de inferência chamada *regra do absurdo intuicionista*, ou *regra da negação intuicionista* (Λ_I):

$$\frac{\bot}{A}$$

Para abranger a lógica clássica, e assim definir o chamado cálculo NK, o mesmo propõe uma das seguintes alterações:

1) adicionar a *"lei do terceito excluído"*, que permite o uso da premissa $A \vee \neg A$, para toda fórmula A.
2) adicionar uma nova figura de inferência $\dfrac{\neg\neg A}{A}$.

A primeira alternativa foi considerada "mais natural" por Gentzen, uma vez que todo o sistema permanece inalterado internamente, apenas com a adição externa de uma nova fórmula básica. Porém, desta maneira esta fórmula acaba ocupando um lugar especial dentro do sistema, fora de todo o esquema de regras de introdução e eliminação. Já a segunda alternativa altera diretamente este esquema de regras, uma vez que uma nova regra de eliminação para a negação é adicionada sem um alternativo análogo para a introdução. Desta maneira o sistema de dedução continua a se sustentar apenas nas figuras de inferência, sem a necessidade de nenhuma fórmula especial ou axioma.

Em ambos os casos, o padrão de introdução e eliminação, onde a dedução natural se sustenta, fica comprometido. E é através deste padrão que algumas das simetrias

que existem na lógica ficam destacadas na DN. Girard, em [Gir87], chega a afirmar que a dedução natural é limitada à lógica intuicionista, argumentando que para a lógica clássica este cálculo não possui propriedades particularmente boas. Na Seção 3.4 estes aspectos serão detalhados.

Prawitz em [Pra65] e [Pra71] define um sistema de dedução natural para a lógica clássica adicionando a *regra do absurdo clássico* (Λ_C), também chamado de *princípio do absurdo clássico*:

$$\dfrac{\begin{array}{c}[\neg A]\\ \vdots\\ \bot\end{array}}{A}$$

ao sistema de regras proposto por Gentzen. Esta regra veio da aplicação da segunda opção de Gentzen. Admitindo a equivalencia entre $\neg\neg A$ e A, temos o seguinte esquema:

$$\dfrac{\begin{array}{c}[\neg A]\\ \vdots\\ \bot\end{array}}{\neg A \to \bot}\,{\scriptstyle \to -I} \quad \equiv \quad \dfrac{\begin{array}{c}[\neg A]\\ \vdots\\ \bot\end{array}}{\neg\neg A} \quad \equiv \quad \dfrac{\begin{array}{c}[\neg A]\\ \vdots\\ \bot\end{array}}{A}$$

3.3 HAUPTSATZ E NORMALIZAÇÃO

Ao introduzir a Dedução Natural (DN) em [Gen35], Gentzen se propunha a investigar as propriedades específicas deste cálculo de dedução. Porém, ele não conseguiu grandes avanços com o sistema DN. Gentzen então definiu um novo cálculo, chamado cálculo de sequentes. Dessa forma ele conseguiu definir um teorema sobre provas formais lógica (clássica e intuicionista): o *Hauptsatz* (chamado *teorema da eliminação do corte* no cálculo de sequentes).

Este teorema afirma que uma prova puramente lógica pode ser reduzida a uma *forma normal*. Esta forma normal é bem definida, porém não é única para uma dada derivação. Uma prova formal é dita estar na sua forma normal se ela for uma prova sem *redundâncias*: nenhum conceito (fórmula) entra na prova se não for relevante para o resultado. Em outras palavras, apenas as fórmulas que estejam contidas no resultado e

cujo uso na derivação seja essencial para alcançá-lo aparecem em provas normalizadas.

Para provar tal teorema, ele precisou definir um cálculo natural logístico, chamado por ele de LJ para a lógica intuicionista e LK para a clássica. Tal cálculo é conhecido como *cálculo de sequentes*, e é visto com detalhes no Apêndice B. Ele afirma que apesar da DN ter as propriedades necessárias para a validade do Hauptsatz, ela o faz satisfatoriamente apenas para a lógica intuicionista. Isso ocorre por causa da lei do terceiro excluído, que ocupa um lugar especial em relação a estas propriedades.

Prawitz, em [Pra65] e [Pra71], investigou e identificou algumas propriedades da DN para definir uma normalização para a mesma. Ele apresenta uma definição de forma normal e uma normalização para três fragmentos do cálculo:

i) *lógica minimal* (chamado sistema M), formado apenas pelas regras da Tabela 3.1;

ii) *lógica intuicionista* (chamado sistema I), formado pelo sistema M mais a regra Λ_I;

iii) *lógica clássica* (chamado sistema C), formado pelo sistema M sem os conectivos $\vee$ e $\exists$, mais a regra Λ_C

Ele apresenta *teoremas de forma normal*, que afirma que toda derivação tem uma forma normal, e *teoremas de normalização*, que oferecem um procedimento sistemático de aplicações de *regras de redução* para obter a forma normal de uma dada derivação. Isto para os três sistemas lógicos definidos acima. Como citado em [dO01], outros resultados sobre a normalização da DN para toda a lógica clássica são encontrados nos trabalhos de J. Seldin, L. C. Pereira e C. Massi, entre outros.

3.4 O PROBLEMA DA SIMETRIA

De acordo com Girard em [Gir89], a lógica possui uma dicotomia fundamental entre o seu sentido, significado, e a denotação utilizada para a sua manipulação e formalização. É impossível separar estes dois lados, uma vez que o sentido contem a denotação, mesmo que implicitamente. A única realidade tangível do sentido é a maneira como ele é escrito, ou seja, o formalismo. O problema é que o formalismo acabou se tornando o principal objeto de estudo, uma vez que ainda não foi dado um significado operacional para este sentido abstrato.

O problema com os formalismos está no fato da sintaxe frequentemente ofuscar as simetrias profundas existentes no significado mais abstrato da lógica. Girard chega a afirmar que não existem simetrias na sintaxe, apenas no sentido, mas que nem por isso uma abordagem puramente sintatica é inválida. O *Hauptsatz* é um exemplo de como algumas simetrias podem ser mostradas em nível sintático.

Na dedução natural, as simetrias existentes se resumem às regras de introdução e eliminação de operadores, e na verdade se restringem apenas a alguns deles ($\wedge, \rightarrow, \forall$). Para todos os operadores as regras de introdução são bem naturais. Para este conjunto satisfatório, as regras de eliminação seguem um padrão intuitivo de virar as regras de introdução "de cabeça para baixo". Porém, nas regras de eliminação para $\vee$ e $\exists$, isso não acontece. Existe uma terceita fórmula, C, a qual Girard chama de "fórmula parasita", que tem uma ligação com as demais apenas com o papel de "contexto".

Esta adoção de contextos para definir regras de eliminação na verdade é um passo na direção ao cálculo sequente. Ao aplicar este conceito na dedução natural, a dedução de uma fórmula não consegue diferenciar de modo satisfatório as derivações:

$$\dfrac{\dfrac{\begin{array}{ccc} \vdots & [A] & [B] \\ A \vee B & \vdots & \vdots \\ & C & C \end{array}}{C}\ {\scriptstyle \vee -E}}{D}\ {\scriptstyle r} \qquad\qquad \dfrac{\begin{array}{ccc} & [A] & [A] \\ \vdots & \vdots & \vdots \\ A \vee B & C & C \\ & D & D \end{array}}{D}\ {\scriptstyle r}\ {\scriptstyle r}\ {\scriptstyle \vee -E}$$

Isso mostra que no fundo uma dedução verdadeira é na realidade uma classe de equivalência de derivações módulo regras de comutação. Girard chega a afirmar que a necessidade destas comutações para este tipo de identificação revela uma inadequação da sintaxe para estes operadores.

Além destas assimetrias, para a lógica clássica o padrão de introdução e eliminação é perdido, como destacado na Seção 3.2. Esta falta de simetria foi denominada por Getnzen em [Gen35] como sendo o "calcanhar de Aquiles" da DN. Para conseguir provar o Hauptsatz, Gentzen teve que utilizar o cálculo de sequentes, que de acordo com Girard consegue mostrar as simetrias da lógica de uma maneira paticularmente satisfatória. Girard afirma que na verdade a idéia básica da DN é uma assimetria: A estrutura de árvore

descreve uma relação desigual entre múltiplas hipóteses e uma única conclusão. Já no cálculo de sequentes, existem múltiplas premissas e múltiplas conclusões.

CAPÍTULO 4

CÁLCULOS COM MÚLTIPLAS CONCLUSÕES

Como visto no Capítulo 3, a falta de simetria é um dos grandes problemas na Dedução Natural. Para tentar minimizar esta falta de simetria, foi proposta a criação de um cálculo que permitisse múltiplas conclusões. Com isso, a estrutura de árvore definida por Gentzen seria abandonada, e a dedução se tomaria uma estrutura mais genérica, onde um conjunto de premissas pode levar a um conjunto de conclusões. Duas abordagens se destacam na tentativa de criar tal cálculo: as *tabelas de desenvolvimento* de Kneale e a Dedução Natural de Ungar.

Kneale foi um pioneiro nesta área. Em seus trabalhos [Kne58] e [KK62] ele tenta modificar a estrutura das regras de descarte de premissas ($\rightarrow -I, \neg -I, \vee -E$ *e* $\exists -E$), alegando que as mesmas são a principal fonte de assimetria na DN. Já Ungar em [Ung92] tenta uma abordagem geométrica para representar as provas, utilizando grafos no lugar de árvores (uma vez que o cálculo proposto permite múltiplas conclusões).

4.1 TABELAS DE DESENVOLVIMENTO DE KNEALE

As *tabelas de desenvolvimento* de Kneale definem um esquema para o desenvolvimento de uma prova lógica formal. O padrão para um desenvolvimento é:

$$\frac{P \qquad Q}{R \qquad S}$$

onde P e Q são as premissas e R e S as conclusões, chamadas aqui de *limites do desenvolvimento*. A dedução de um conjunto de fórmulas é definida de maneira similar à dedução natural de Gentzen, onde a partir de um conjunto de fórmulas iniciais, e com a aplicação de dos desenvolvimentos definidos na Tabela 4.1, pode-se chegar às fórmulas finais da derivação.

O asterisco significa que qualquer fórmula pode aparecer, independente de P e Q. A regra $\neg_1$ é na verdade uma formulação da lei do terceiro excluído, como proposta por

$(\wedge_1)$ $\dfrac{P \quad Q}{P \wedge Q}$	$(\wedge_{2a})$ $\dfrac{P \wedge Q}{P}$	$(\wedge_{2b})$ $\dfrac{P \wedge Q}{Q}$
$(\vee_{1a})$ $\dfrac{P}{P \vee Q}$	$(\vee_{1b})$ $\dfrac{Q}{P \vee Q}$	$(\vee_2)$ $\dfrac{P \vee Q}{P \quad Q}$
$(\neg_1)$ $\dfrac{*}{P \quad \neg P}$	$(\neg_2)$ $\dfrac{P \quad \neg P}{*}$	
$(\rightarrow_{1a})$ $\dfrac{*}{P \quad P \rightarrow Q}$	$(\rightarrow_{1b})$ $\dfrac{Q}{P \rightarrow Q}$	$(\rightarrow_2)$ $\dfrac{P \quad P \rightarrow Q}{Q}$

Tabela 4.1 Esquema básico de desenvolvimento para a lógica proposicional

Gentzen, porém adicionada como um desenvolvimento, não como uma fórmula especial. A regra $\neg_2$ representa o princípio da não contradição. A regra $\rightarrow_{1a}$ é a mais diferente das existentes na dedução natural. Kneale explica que a mesma é um equivalente ao princípio de que uma implicação é válida quando o antecedente é falso.

A diferença aqui entre o cálculo definido por Kneale e o de Gentzen está no fato de uma derivação poder gerar mais de uma fórmula final, perdendo a estrutura de árvore sobre a qual a DN se firma. Além disso, existem desenvolvimentos como $\vee_2$, $\neg_1$ e $\rightarrow_1$, que permitem a geração de mais de uma fórmula, diferentemente de todas as figuras de inferências definidas para a DN.

As regras $\vee_2$ e $\rightarrow_{1b}$ têm como equivalentes na DN as regras $\vee - E$ e $\rightarrow - E$, respectivamente. Estas úlimas são as responsáveis pelo descarte de hipóteses na lógica proposicional, e aqui elas tomam um aspecto diferente. A regra $\vee_2$ não depende mais de uma terceira fórmula C, que representava um contexto na dedução, e não mais necessita dos termos P e Q como premissas para ser aplicada. Já a regra $\rightarrow_{1b}$ não precisa mais de P como premissa para concluir $P \rightarrow Q$, uma vez que Q foi concluída previamente na derivação. Assim, nas tabelas de desenvolvimento de Kneale não há mais o descarte de premissas, e assim não existe também a ligação semântica entre a regra e as premissas descartadas,

que foi apontada por Girard em [Gir87].

Além de remover o descarte de hipóteses, as tabelas de Kneale também oferecem regras que apresentam uma simetria entre introdução e eliminação de operadores mesmo no caso do $\vee$. Além disso, existe uma simetria entre os operadores $\vee$ e $\wedge$, que antes não era explicitada pela DN. De um modo geral, muitos dos problemas de simetria da dedução natural foram atacados com esta abordagem, e outras simetrias e dicotomias, que antes eram "borradas" pela sintaxe do cálculo da DN, ficaram claras agora.

Para evitar falácias geradas pela possibilidade de múltiplas conclusões, como

$$\dfrac{\dfrac{A \vee B}{A \qquad B}}{A \wedge B}$$

Kneale impõe sobre as fórmulas a seguinte restrição: fórmulas que estão ligadas horizontalmente, diretamente por uma ligação ou indiretamente por algumas ligações, não podem se conectar novamente de nenhuma outra maneira. P e Q estão ligadas horizontalmente se ambas fazem parte da conclusão de uma mesma regra.

Diferentemente da dedução natural de Gentzen, as tabelas de Kneale consistem de um cálculo para a lógica clássica. Para obter uma versão para a lógica intiocionista, a regra $\neg_1$ deve ser removida, e a correspondente da dedução natural $(\neg - I)$ adicionada no seu lugar. Isto revela uma desvantagem no sistema para a lógica intuinionista, uma vez que voltamos ao sistema de descarte de hipóteses da DN.

4.2 INADEQUAÇÃO

O problema da inadequação é inerente a sistemas de prova com múltiplas conclusões. Ele foi primeiramente identificado por Shoesmith e Smiley em [SS78] sobre as tabelas de desenvolvimento de Kneale, e depois foi referenciado por Ungar em [Ung92] como o "problema da substituição" na eliminação do $\vee$. Ele caracteriza-se pela inexistencia de um desenvolvimento de S para Q, mesmo quando existem desenvolvimentos de S para R e de R para Q.

Este tipo de obstáculo não ocorre em cálculos de uma única conclusão, como a dedução natural, porque não existem repetições de fórmulas na conclusão e as premissas são agru-

padas em classes de equivalências. Na DN, por exemplo, quando temos uma dedução de S para R:

$$\Pi : \quad \dfrac{\begin{array}{ccc}[S] & [A] & [B] \\ \vdots & \vdots & \vdots \\ A \vee B & R & R\end{array}}{R} \vee\!-E$$

e uma dedução de R para Q:

$$\Pi' : \quad \dfrac{\begin{array}{c}[R] \ldots [R] \\ \vdots \\ Q\end{array}}{} r$$

podemos construir uma derivação de S para Q facilmente substituindo cada ocorrência de R como fórmula inicial de Π', por Π:

$$\dfrac{\begin{array}{ccc}\Pi & & \Pi \\ R & \cdots & R \\ & \vdots & \\ & Q &\end{array}}{} r$$

Já com um cálculo de múltipla conclusão isso nem sempre é possível. Isso porque quando a derivação Π possui mais de uma conclusão, ela já não pode mais substituir uma ocorrência apenas de R em Π'. Além disso, no caso das tabelas de Kneale, se duas ocorrências de R se ligam em Π', elas não podem vir da mesma derivação Π, por causa da restrição sobre fórmulas com ligações horizontais. Este caso é ilustrado na Figura 4.1.

Shoesmith e Smiley mostraram um exemplo de uma tautologia $(A \to A) \wedge (A \vee (A \to A))$, que não possui desenvolvimento nas tabelas de Kneale. Então eles mostraram um desenvolvimento Π_1 para $A \to A$ e um Π_2 de $A \to A$ para $(A \to A) \wedge (A \vee (A \to A))$. A Tabela 4.2 mostra tais desenvolvimentos, e a tentativa falha de gerar um desenvolvimento Π para a tautologia.

No sistema de Kneale, a restrição sobre fórmulas ligadas horizontalmente foi inserida para evitar falácias, como visto na seção anterior, e é a responsável pela inadequação. Ao

restringir o uso de mais de uma conclusão de um mesmo desenvolvimento como premissas de uma regra, ele acaba restringindo alguns casos onde seria permitido fazer o mesmo, sem gerar inconsistência. Por exemplo, na Tabela 4.2 temos Π_1 gerando conclusões iguais, e que semanticamente poderiam ser utilizadas como premissas em uma mesma derivação, apesar de ligadas horizontalmente. Se Π_1 e Π_2 fossem concatenadas, gerando o desenvolvimento Π desejado, da maneira mostrada na Tabela 4.3, teríamos um desenvolvimento logiamente correto, mas inválido no sistema de Kneale.

Analisando de maneira geométrica um desenvolvimento, percebe-se que uma falácia ocorre na presença de um ciclo no fluxo lógico das ocorrências das fórmulas. Isso porque a semântica de uma regra é: a conjunção das premissas implicam na disjunção das conclusões. Porém, nem todo ciclo em uma prova é inválido, como mostra a Tabela 4.4. A restrição de Kneale pode ser traduzida aqui com uma que impede a geração de qualquer ciclo. Como mostrado por Shoesmith e Smiley, apesar de deixar o sistema correto, ele leva à sua incompletude.

Shoesmith e Smiley propuseram refinamentos sobre as tabelas de Kneale para tratar a inadequação de uma maneira mais flexível do que a proposta pelo mesmo. Eles analisaram os desenvolvimentos como grafos, e propõem três soluções. Entre elas duas que merecem destaque: a segunda, cujo procedimento gera cópias das duas derivações Π_1 e Π_2 e as combina de modo a gerar uma única derivação logicamente válida; e a terceira, que busca identificar premissas e conclusões nos desenvolvimentos e as colide em uma ocorrência única, permitindo a junção das duas em um único ponto. Elas são ilustradas na Figura 4.2 e Figura 4.3. Outra solução foi a proposta Ungar, que desenvolveu um outro sistema de múltiplas conclusões e tratou o problema da inadequação no mesmo.

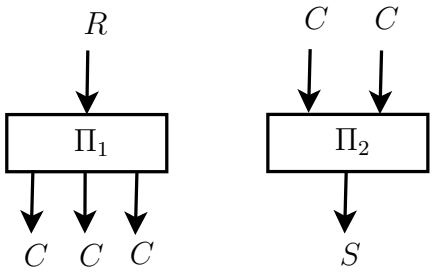

Figura 4.1 Grafos genéricos para Π_1 e Π_2.

Π_1	Π_2
$\dfrac{\dfrac{A}{A \to A} \qquad A \to A}{}$	$\dfrac{A \to A \qquad \dfrac{A \to A}{A \vee (A \to A)}}{(A \to A) \wedge (A \vee (A \to A))}$

Tabela 4.2 Exemplo da inadequação nas tabelas de desenvolvimento de Kneale.

$$\dfrac{\dfrac{\dfrac{A}{A \to A}}{A \vee (A \to A)} \qquad A \to A}{(A \to A) \wedge (A \vee (A \to A))}$$

Tabela 4.3 Ligando dois desenvolvimentos de forma inválida: a parte superior contém o desenvolvimento Π_1 e a inferior Π_2, ambos mostrados na Tabela 4.2.

derivação de uma falácia	derivação válida
$\dfrac{A \vee B}{\dfrac{A \quad B}{A \wedge B}}$	$\dfrac{A \vee A}{\dfrac{A \quad A}{A \wedge A}}$

Tabela 4.4 Ciclos no cálculo com múltiplas conclusões: O primeiro exemplo mostra um ciclo logicamente incorreto, enquando o segundo é válido.

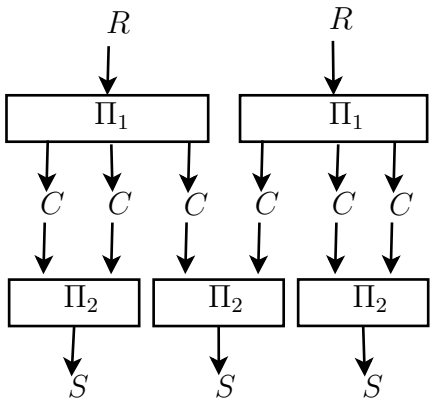

Figura 4.2 Junções múltiplas.

4.3 DEDUÇÃO NATURAL POR UNGAR

Ungar propôs em [Ung92] um tratamento revisado da dedução natural através de um cálculo de múltiplas conclusões, com o objetivo de estudar a correspondencia entre a eliminação do corte e a normalização (ou seja, tentando reconciliar DN e o cálcuo sequente). De acordo com ele, existem alguns aspectos da dedução natural que poderiam ser melhorados em um cálculo revisado. Entre eles, o principal que Ungar destaca é regra da eliminação, que para ele não representa de maneira satisfatória o significado por trás. Esta observação já tinha sido feita por Girard em (capítulo 10, [Gir87]), quando ele destaca vários defeitos do cálculo de Gentzen.

Ungar afirma que, uma vez que as derivações deveriam representar os "objetos reais" envolvidos em uma prova formal, a presença das regras de eliminação não representa a estrutura da prova como ela deveria ser formalizada. Logo, uma versão revisada do cálculo deve incluir uma reformulação da regra para a forma $\frac{A \vee B}{A \quad B}$, semelhantemente a versão de Kneale.

Neste sistema, temos:

- Cada ocorrência de fórmula tem um número natural subescrito, que indica a classe a qual esta ocorrência pertence;
- A notação A_n indica que todas as ocorrências de A_n são descartadas por esta regra;
- Derivação é um grafo cujos vértices são rotulados com ocorrências de fórmulas;
- Cada axioma A_m é representado por um vértice rotulado com A_m.

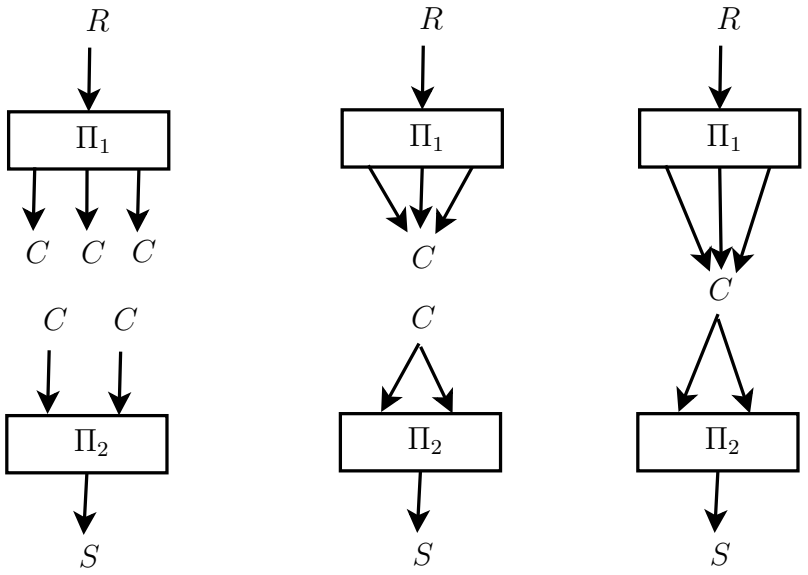

Figura 4.3 Unificação de premissas e conclusões para junção simples.

Para evitar falácias do tipo

$$\dfrac{\dfrac{A \vee B}{A \quad B}}{A \wedge B}$$

Ungar estripula que as premissas das regras $\wedge_1$ e $\rightarrow_2$ não podem ser ambas conclusões de uma mesma derivação. Esta restrição é mais leve do que a aplicada por Kneale às tabelas de desenvolvimento, e isso acaba por gerar um sistema menos rígido ao lidar com os ciclos.

Ungar estuda em seu sistema um problema que é equivalente à inadequação escontrada por Schoesmith e Smiley, que ele denomina problema da substituição. A solução que ele considera ideal para tratar os ciclos deve permitir a existencia de ciclos válidos, diferentemente do que Kneale fez. Ungar propõe a aplicação de uma operação sobre o grafo, chamada *combinação*, que quando aplicada de modo a gerar um grafo logicamente válido passa a se chamar *substituição*. A idéia por trás é gerar cópias das derivações que serão juntas, e combiná-las de maneira a gerar um grafo que represente uma derivação Π da conclusão. Na Figura 4.4 é ilustrada a aplicação da técnica.

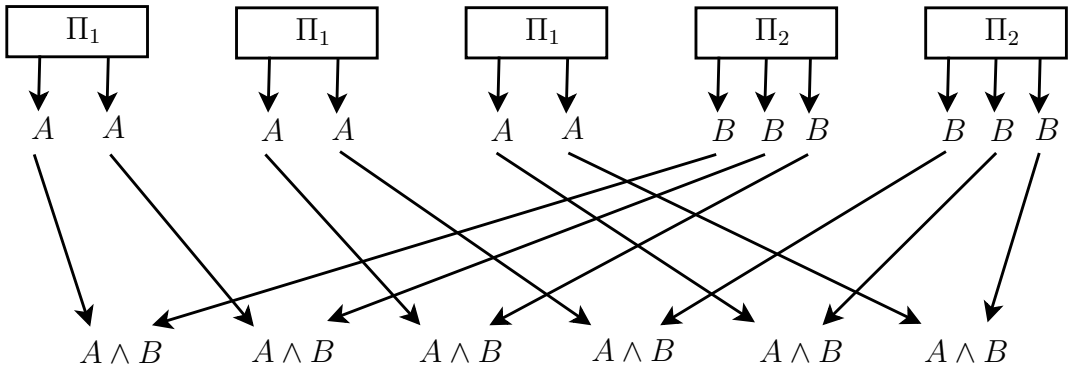

Figura 4.4 Exemplo da substituição de Ungar.

Axiomas:	A_m

Regras:

$$(\wedge_1)\ \frac{A_m \quad B_n}{A \wedge B_p} \qquad (\wedge_{2a})\ \frac{A \wedge B_p}{A_m} \qquad (\wedge_{2b})\ \frac{A \wedge B_p}{B_n}$$

$$(\vee_{1a})\ \frac{A_m}{A \vee B_p} \qquad (\vee_{1b})\ \frac{B_n}{A \vee B_p} \qquad (\vee_2)\ \frac{A \vee B_p}{A_m \quad B_n}$$

$$(\rightarrow_1)\ \frac{B_m}{A \rightarrow B_p\{A_n\}} \qquad (\rightarrow_2)\ \frac{A_n \quad A \rightarrow B_p}{B_m}$$

$$(\perp_1)\ \frac{\perp_p}{A_n} \qquad (\perp_2)\ \frac{A_n}{\perp_p}$$

Tabela 4.5 Sistema de Ungar

CAPÍTULO 5

N-GRAFOS

N-grafos é um sistema de formalização de provas, proposto por de Oiveira em [dO01], baseado nas regras da dedução natural (DN) com a adição de algumas regras estruturais similares às do cálculo de sequentes. Define-se como um sistema de prova com múltiplas conclusões para a lógica clássica proposicional na forma de grafos direcionados (*"digrafo"*). O principal objetivo deste sistema é oferecer uma perspectiva geométrica para as provas em DN, permitindo um estudo dos aspectos estruturais de uma prova formal. Para isso, ela torna explícita algumas simetrias por meio da extração do fluxo de ocorrências de uma fórmula, removendo assim o peso sintático que tornam implícitas tais simetrias nos demais formalismos.

O sistema de N-grafos foi concebido com a incorporação de idéias vindas de outros estudos realizados sobre sistemas de prova com múltiplas conclusões e sobre o aspecto geométricos destas provas. As derivações são baseadas no cálculo ND simétrico de múltiplas conclusões definido por Kneale e Ungar, como visto no Capítulo 4. Os critérios de validade (*soundness*) e de corretude foram provados por uma perspectiva geométrica, inspirados pelo trabalho de Danos-Regnier [DR89] sobre as redes de prova (do inglês, *proof-nets*) de Girard.

A seguir definimos grafos de prova, conceito similar ao de estruturas de prova (do inglês, *proof-structures*) definido por Girard em [Gir87]. Assim como Girard definiu *estruturas de prova* como grafos que representam provas na lógica linear, e *redes de prova* como estruturas de prova logicamente válidas, de Oliveira criou o conceito de *grafos de prova*, que são grafos que representam provas na lógica clássica, e o de *N-grafos* como grafos de prova logicamente válidos. As noções de teoria dos grafos aqui utilizadas vem do Apêndice A.

5.1 GRAFOS DE PROVA

Os grafos de prova são grafos direcionados, onde cada vértice é rotulado com a ocorrência de uma fórmula, e as ligações entre estas ocorrências (arestas) são definidas a partir do conceito de "links", onde cada link representa uma derivação atômica. Abaixo algumas definições importantes retiradas de [dO01]:

Definição 5.1 (**ponto de ramificação).** Um *ponto de ramificação* em um digrafo é um vértice (ou nó) com três arestas ligadas ao mesmo.

Definição 5.2 (**ponto de ramificação convergente).** Um *ponto de ramificação convergente* é um vértice em um digrafo com duas arestas direcionadas ao mesmo.

Definição 5.3 (**ponto de ramificação divergente).** Um *ponto de ramificação divergente* é um vértice em um digrafo com duas arestas saindo do mesmo.

Definição 5.4 (**link convergente).** Um *link convergente* é um conjunto $\{(u_1, v), (u_2, v)\}$ de arestas de um digrafo onde v é um ponto de ramificação convergente, como mostra Figura 5.1. Os vértices u_1 e u_2 são chamados premissas do link, e v conclusão.

Definição 5.5 (**link divergente).** Um *link divergente* é um conjunto $\{(u, v_1), (u, v_2)\}$ de arestas de um digrafo onde u é um ponto de ramificação divergente, como mostra Figura 5.1. Os vértices v_1 e v_2 são chamados conclusões do link, e u premissa.

Definição 5.6 (**link simples).** Um *link simples* é uma aresta (u, v) de um digrafo que não pertence a nenhum link convergente ou divergente, como mostra Figura 5.1. O vértice u é chamado premissa do link, e v conclusão.

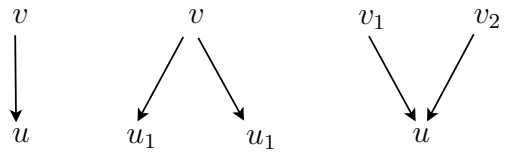

Figura 5.1 Links em um grafo de prova.

Definição 5.7 (**grafo de prova).** Um *grafo de prova* é um grafo conexo direcionado onde:

i) cada vértice é rotulado com uma ocorrência de uma fórmula;

ii) as arestas são de dois tipos (*"sólida"* e *"meta"*), onde as meta-arestas são rotuladas com um *"m"* ($(u, v)^m$).

iii) os links são de três tipos (*simples*, *convergente* e *divergente*), que por sua vez são divididos em links *lógicos* e *estruturais*, como mostram Figura 5.2 e Figura 5.3;

iv) cada vértice é rotulado como a conclusão de um único link e é a premissa de no máximo um link.

Em um grafo de prova, as direções das arestas formam um fluxo lógico sobre as ocorrências de fórmulas na prova, não um fluxo temporal. Um *link lógico* é um link que representa a aplicação de uma derivação da DN, com a incorporação do $\top - link$ que representa a *lei to terceiro excluído* (como formulada por Gentzen para a lógica clássica). Um *link estrutural* é um link que representa a aplicação de uma regra estrutural similar às definidas no cálculo de sequentes, e portanto permitem o enfraquecimento da prova, a duplicação de "recursos" (premissas) e a junção de ocorrências da mesma fórmula em uma única ocorrência (contração). Não existe um equivalente à regra de troca (do inglês, *interchange*) do cálculo de sequentes, uma vez que nos grafos de prova a ordem das premissas não é relevante para a aplicação das regras de derivação.

Os links podem ainda ser classificados como:

Definição 5.8 (**link conjuntivo).** Os links $\wedge - I$, $\bot - link$, $\rightarrow - E$, $\top - enfraq. - convergente$ e de *expansão* são chamados *conjuntivos*.

Definição 5.9 (**link disjuntivo).** Os links $\vee - E$, $\top - link$, $\bot - enfraq. - divergente$ e de *contração* são chamados *disjuntivos*.

Os links conjuntivos e disjuntivos são definidos com relação ao valor semântico que possuem, não geométrico. Por isso a diferença entre links conjuntivos (disjuntivos) de links convergentes (divergentes). Apesar da maioria dos links convergentes (divergentes) serem conjuntivos (disjuntivos), os links de contração e expansão tem uma geometria contrária a sua semântica.

Um link de contração tem o significado de uma disjunção de suas premissas, uma vez que uma delas está sendo "descartada" da prova (em outras palavras, apenas uma das derivações será relevante para a conclusão, não ambas). Um link de expansão tem o significado de uma conjunção de suas conclusões, uma vez que a prova final necessita

LINKS LÓGICOS

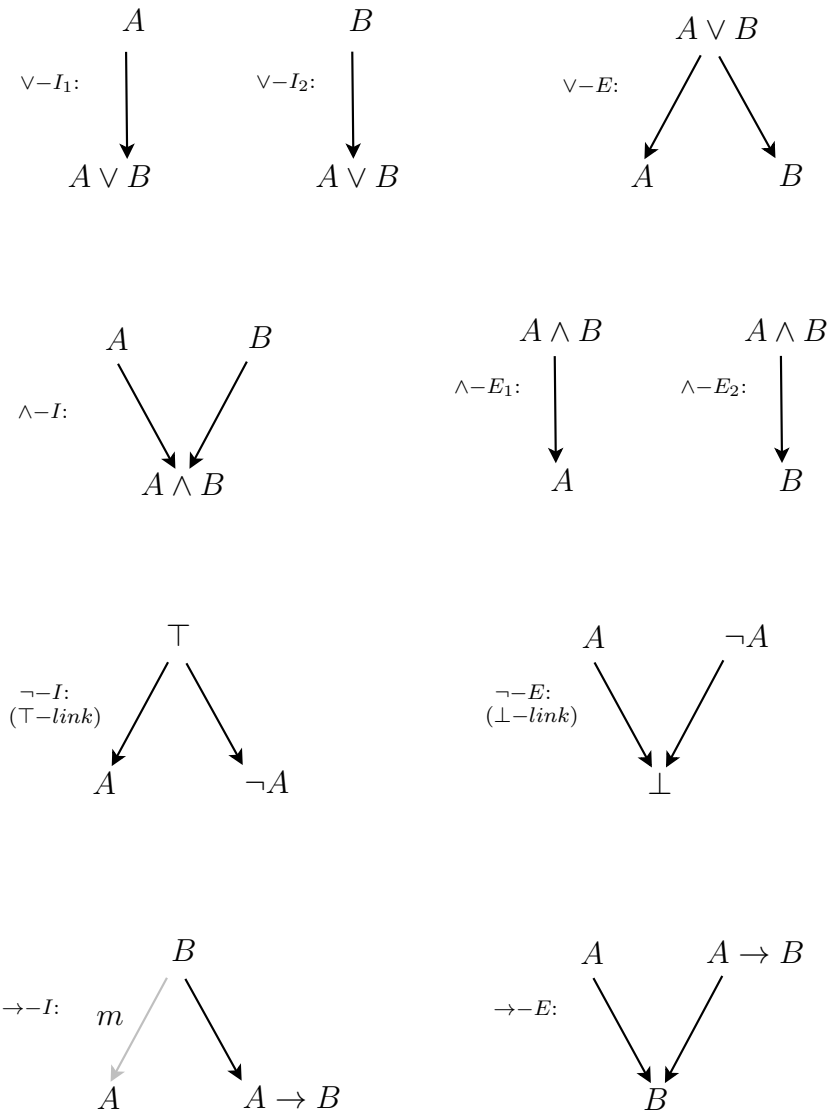

Figura 5.2 Links lógicos

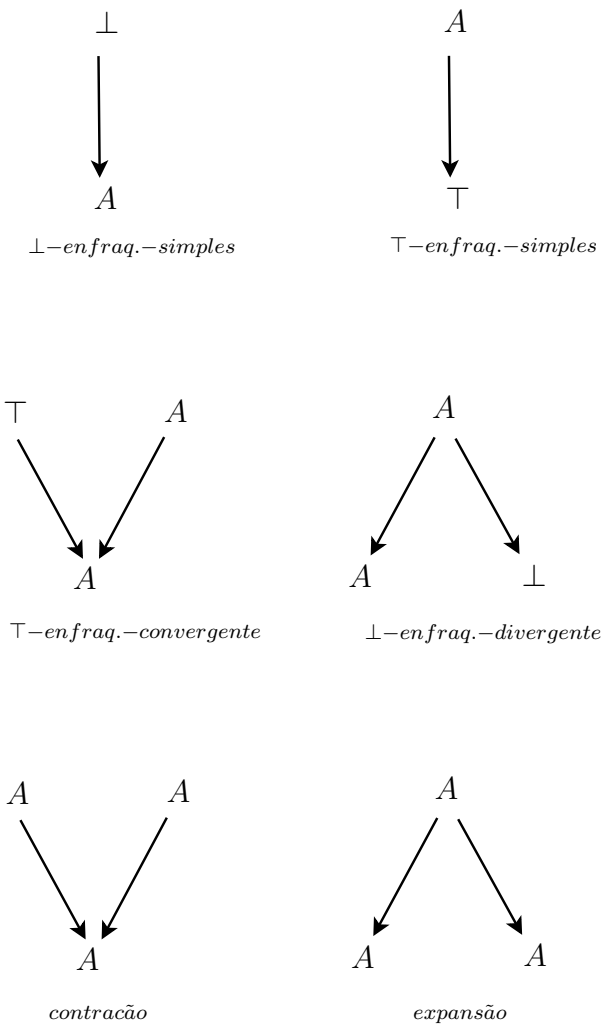

Figura 5.3 Links estruturais

de ambas derivações. Quando forem definidos os N-grafos, o significado destes links será examinado com mais cuidado.

É importante destacar que a meta-aresta carrega o significado do cancelamento de uma hipótese. Nos grafos de prova, isto ocorre apenas na introdução da implicação ($\rightarrow -I$). Na dedução natural existe outra maneira de se remover hipóteses (eliminação do $\vee$), porém esta regra é justamente um exemplo de como o açúcar sintático da DN obscurece uma simetria inerente à lógica deste operador, como destacado no Capítulo 3. Ao definir este operador para um cálculo de mútiplas conclusões, de Oliveira utiliza a solução proposta por Kneale, e assim destaca melhor a simetria deste operador e sua dualidade com o $\wedge$.

A seguir mais alguns conceitos importantes sobre um grafo de prova G:

- Se G não tem arestas, ele representa um *axioma*;
- O conjunto de vértices com o grau de saída igual a zero represenenta as conclusões de G ($CONC(G)$);
- O conjunto de vértices com o grau de entrada igual a zero represenenta as premissas de G ($PREMIS(G)$);
- O conjunto de vértices com o grau de entrada igual a um, onde esta é uma meta-aresta, representa as hipóteses canceladas de G ($HIPOT(G)$);
- O número de arestas sólidas direcionadas a um vértice v é chamado *grau de entrada sólido* de v. De modo análogo é definido o *grau de entrada meta* de v;
- O número de arestas sólidas que partem de um vértice v é chamado *grau de saída sólido* de v. De modo análogo é definido o *grau de saída meta* de v;

5.2 INADEQUAÇÃO

Como um cálculo de múltiplas conclusões, os grafos de prova possuem o mesmo problema de inadequação encontrado nas tabelas de desenvolvimento de Kneale [Kne58], discutido por Ungar em [Ung92] como o problema da substituição, como visto no Capítulo 4. A solução proposta por de Oliveira foi inspirada pela análise do problema feita por Shoesmith-Smiley e Ungar, mas com um toque da simplicidade da técnica utilizada na

teoria das redes de prova por Danos-Regnier.

A idéia principal é diferenciar um link que representa uma derivação lógica de um que simplesmente manipula a estrutura da derivação. Assim, a condição imposta por Kneale e Ungar sobre as regras $\wedge - I$ e $\rightarrow -E$ usarem premissas de derivações distintas pode ser relaxada. A ocorrência de alguns ciclos [1] será permitida nos grafos de provas, mas de maneira controlada, por meio do uso de links estruturais explícitos para contração e expansão.

5.2.1 Links de expansão e contração

Em cálculos de múltipla conclusão, o principal causador de problemas é a ocorrência de ciclos em derivações. O que de Oliveira fez ao avaliar o problema da inadequação nos N-grafos foi tentar distinguir os ciclos válidos dos inválidos, extraindo algum padrão que os pudesse identificar. Analisando semanticamente o problema nas tabelas de desenvolvimento de Kneale, observa-se que um ciclo inválido ocorre quando uma conjunção é feita sobre dois termos gerados por uma disjunção, como pode ser visto na Figura 5.4.

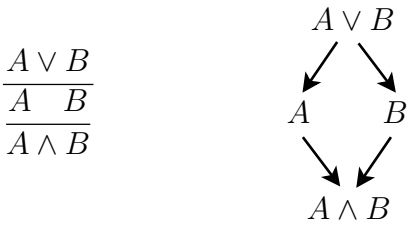

Figura 5.4 Grafo de prova inadequado: o grafo à direita representa a derivação inadequada à esquerda.

Já um ciclo válido ocorre quando uma disjunção gera uma mesma fórmula, que podem depois serem ambas premissas de um mesmo link conjuntivo, uma vez que ambas tem o mesmo valor semântico. Para tratar este caso onde ciclos são válidos, de Oliveira propõe o agrupamento de conclusões em classes de equivalência, assim como na dedução natural isto é feito com premissas. É neste contexto que entra o papel do link de contração: agrupar conclusões iguais (ocorrências distintas de uma mesma fómula) em uma mesma

[1] Daqui em diante, o termo ciclo será aplicado quando, estritamente falando, na verdade se tratar de um semi-ciclo. Ver Apêndice A para mais detalhes.

conclusão. Com este artifício, ciclos válidos como o mostado no grafo de prova mais à direita da Figura 5.5 podem ser gerados sem o problema descrito acima: o ciclo superior deste grafo é aberto por uma disjunção e fechado por outra disjunção (a contração).

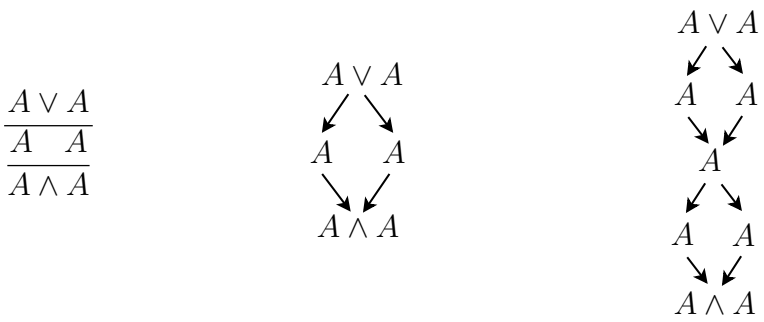

Figura 5.5 Grafos de prova para $A \vee A \vdash A \wedge A$: à esquerda um grafo inadequado e à direita o grafo correto.

O link de expansão tem o papel de simular o efeito que existe em cálculos de única conclusão de geração de ocorrências de uma mesma premissa. Ele tem o efeito de agrupar ocorrências distintas de uma mesma fórmula em uma mesma classe de equivalência. Para permitir a conjunção de uma fórmula com ela mesma (por exemplo, $A \wedge A$), é necessário a existencia de duas ocorrências desta mesma fórmula (dois vértices no grafo de prova). Com este artifício a prova de $A \vee A \vdash A \wedge A$, mostrada na Figura 5.5, pode ser completada. Como o valor semântico de uma expansão é uma conjunção, o ciclo inferior do exemplo contém dois links conjuntivos, e portanto é válido (uma vez que não cai no padrão de fazer uma conjunção de duas premissas vindas de uma disjunção).

O link de expansão pode ser visto ainda como uma contração à esquerda do cálculo de sequentes, enquando o link de contração representa uma contração à direita. Deste modo fica claro o papel de ambos ao definir classes de equivalência entre premissas e conclusões, respectivamente. Por ter este papel bem definido, podemos observar em Figura 5.6 e Figura 5.7 que uma expansão não deve ter como conclusão uma conclusão da derivação, nem uma contração deve atuar sobre premissas da prova.

Como visto no Capítulo 4, o problema da inadequação gerado nas tabelas de Kneale vem da restrição severa imposta sobre os ciclos. Uma das soluções dadas por Shoesmith e Smiley foi justamente o agrupamento de conclusões e premissas para permitir a junção de duas provas Π_1 de $R \vdash C, C, C$ e Π_2 de $C, C \vdash S$ em uma prova final Π de $R \vdash S$. Aqui

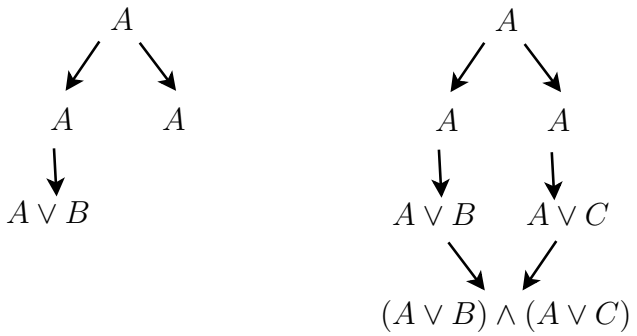

Figura 5.6 Links de expansão: exemplos de uma aplicação incorreta (à esquerda, para $A \vdash A \vee B, A$) e uma correta (à direita, para $A \vdash (A \vee B) \wedge (A \vee C)$) do link de expansão.

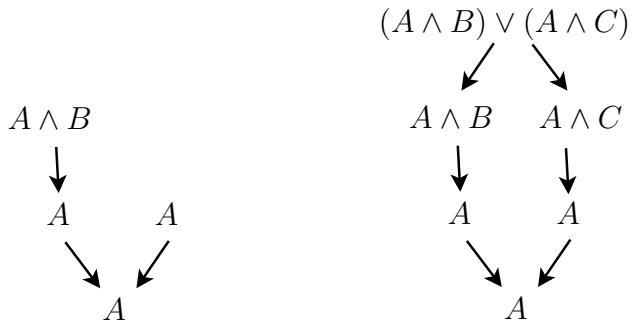

Figura 5.7 Links de contração: exemplos de uma aplicação incorreta (à esquerda, para $A \wedge B, A \vdash A$) e uma correta (à direita, para $(A \vee B) \wedge (A \vee C) \vdash A$) do link de contração.

a solução é bastante similar, porém já incorporada nas regras de derivação do cálculo definido por de Oliveira.

5.2.2 Meta-aresta e descarte de hipóteses

Outra possível fonte de grafos inválidos é o descarte de hipóteses. Nos sistemas de Kneale não existem regras que anulam premissas. Já no sistema de Ungar, e na própria dedução natural de Gentzen, o descarte é feito de maneira livre, onde uma regra pode eliminar um número arbitrário de premissas (possivelmente nenhuma) e gerar uma conclusão independente desta fórmula. No sistema aqui apresentado, este descarte é feito de uma maneira mais controlada, e por isso merece mais atenção.

Nos grafos de prova a única regra que pode descartar hipóteses é $\rightarrow -I$. Nela, temos uma meta-aresta que liga a premissa deste link a um vértice v, representando o descarte

desta ocorrência de v como premissa na prova. Isto explicita a ligação semântica que existe entre a regra e as hipóteses que ela descarta e destrói a estrutura de árvore definida por Gentzen, exatamente como Girard afirmou em [Gir89]. Este novo ciclo que pode surgir não torna a derivação inválida caso obedeça a algumas restrições.

Primeiramente deve-se observar que o ciclo gerado por esta meta-aresta é válido, uma vez que ela apenas acentua a ligação semântica entre as duas fórmulas em questão para a geração da implicação entre elas. Porém, é preciso se certificar que aquela ocorrência da fórmula v foi de fato descartada na prova, ou seja, nenhuma outra cópia da mesma é necessária para a geração de outras conclusões. Observe que aqui estamos falando de cópias de uma ocorrência da fórmula, não da fórmula em si (que pode ter outras ocorrências além da descartada).

O conceito de cópia aqui citado se refere ao poder do link de expansão de gerar novas ocorrências a partir de uma. Sempre que uma ocorrência descartada por uma meta-aresta for "copiada" por meio de uma expansão, estas cópias não devem gerar outras conclusões além da implicação, como mostra a Figura 5.8. Se isto ocorrer, seria possível a prova de falácias como $\vdash A \vee B$.

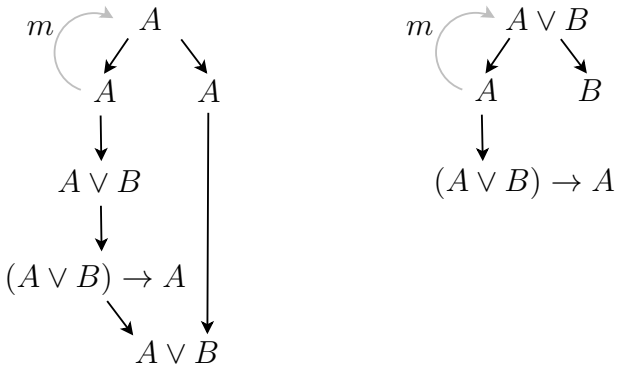

Figura 5.8 Meta-aresta: exemplos de uma aplicação incorreta (à esquerda, para $\vdash A \vee B$) e uma correta (à direita, para $\vdash (A \vee B) \to A$) do descarte de hipóteses por meta-arestas.

Porém, se a ocorrência de v descartada por uma meta-aresta for premissa de uma outra conclusão (além da implicação que a cancela), mas isso ocorrer sem a utilização de expansões pelo caminho, então o grafo de prova final é válido. Isto se dá pela característica disjuntiva dos demais links divergentes. Se o vértice v é capaz de gerar duas conclusões, u e z, sem o uso de expansões para separar a prova em duas conclusões, isto é

equivalente ao sequente $v \vdash u, z$. Logo, v pode ser descartado do conjunto de premissas e uma nova conclusão aparece: $v \to u$ (aplicando a regra equivalente $\to -IS$ do cálculo de sequentes). Temos então o sequente $\vdash v \to u, z$. Mas, como o link de expansão tem um significado conjuntivo, não é possível simplesmente cancelar v como premissa da prova, se apenas a conlusão u o descarta, uma vez que ambas tenham sido geradas separadamente por meio de uma (ou mais) expansões no caminho. Isso porque, com o uso destas expansões, temos u e z a partir de v, não mais u ou z.

Os grafos de prova na Figura 5.8 ilustram este caso. O primeiro exemplo mostra como o cancelamento de A por uma das cópias da mesma não é refletido na outra, que logo depois é utilizada em um link conjuntivo como verdade e gera uma falácia. Já o segundo caso reflete a característica disjuntiva do link divergente aplicado, de modo que apenas um dos ramos da prova descartou a premissa A, mas o outro ramo não afetou o primeiro de maneira conjuntiva (e nem poderia de maneira válida, como visto na subseção anterior), de modo que ambas as conclusões têm uma relação disjuntiva e podem ser independentes de A.

Como em um link $\to -I$ temos apenas uma meta-aresta, apenas uma ocorrência de v pode ser descartada com a introdução da implicação. Para simular o significado que temos na dedução natural, onde o descarte é de um número arbitrário de ocorrências, temos a seguinte abordagem: para simular o descarte de mais de uma ocorrência, estas ocorrências devem ser agrupadas em uma mesma classe de equivalência, através do uso de links de expansão, e deve-se ligar a meta-aresta à ocorrência inicial que foi expandida. A Figura 5.9 mostra como isso deve ser feito. Aqui deve-se ter cuidado com o caso inválido visto na Figura 5.8.

Não é possível nos grafos de prova termos o cancelamento de nenhuma hipótese na introdução da implicação. O que podemos fazer é simular cancelando uma hipótese que semanticamente não é relevante para nenhuma das conclusões, como mostrado na Figura 5.10.

5.3 CRITÉRIO DE CORRETUDE

Na definição de um critério de corretude sobre os grafos de prova, entra o conceito de *N-grafos*: grafos de prova semanticamente válidos. Nos N-grafos, apenas os ciclos válidos

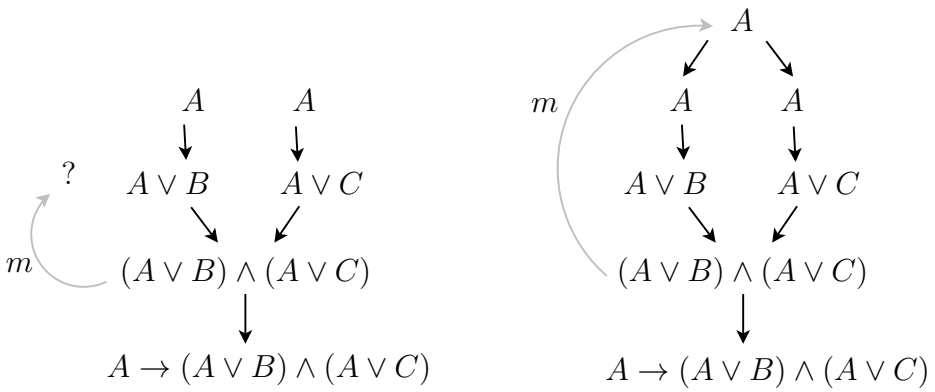

Figura 5.9 Cancelamento de mais de uma ocorrência com meta-aresta.
Exemplo da prova de $\vdash A \to (A \vee B) \wedge (A \vee C)$

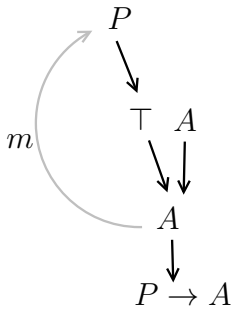

Figura 5.10 Cancelamento de "nenhuma" ocorrência com meta-aresta.
Exemplo da prova de $A \vdash P \to A$

são permitidos, e além disso a meta-aresta é tratada de maneira especial para que o descarte de hipóteses ocorra de maneira similar à DN.

Primeiramente analisando os ciclos, foi visto na seção anterior que os ciclos válidos são aqueles os quais não permitem a conjunção de fórmulas vindas de uma disjunção. Para tratar isto, os links de contração e expansão tem um valor semântico intuitivamente diferente de seu aspecto geométrico: a contração é convergente e disjuntiva, e a expansão é divergente e conjuntiva. Neste ponto temos um problema semelhante ao encontrado nas redes de prova: como distinguir links conjuntivos de links disjuntivos em uma prova geométrica, quando os mesmos nem sempre são convergentes e divergentes, respectivamente?

Para solucionar este problema, o critério criado por de Oliveria se baseia na proposta de Danos-Regnier para as redes de prova [DR89]. De maneira análoga aos *grafos D-R*

(do inglês, *D-R graph*), é definido o conceito de *grafo de chaveamento*.

Definição 5.10 (**grafo de chaveamento).** Dado um grafo de prova G, o *grafo de chaveamento* $S(G)$ associado a G é um subgrafo gerador de G onde as seguintes arestas são removidas:

- uma das duas arestas de todos os links de expansão;
- uma das duas arestas de todos os links de contração;
- todas as meta-arestas.

Para verificar os casos que envolvem a implicação, é necessário definir um conceito similar ao de grafo de chaveamento, porém apenas para os links de expansão e as meta-arestas. O *grafo de chaveamento de expansão* é introduzido:

Definição 5.11 (**grafo de chaveamento de expansão).** Dado um grafo de prova G, o *grafo de chaveamento de expansão* $S_e(G)$ associado a G é um subgrafo gerador de G onde as seguintes arestas são removidas:

- uma das duas arestas de todos os links de expansão;
- todas as meta-arestas.

Definição 5.12 (**meta-condição).** Dado um grafo de prova G, dizemos que a *meta-condição* é satisfeita por G sse para toda meta-aresta $(u, v)^m$ de um link divergente $\{(u, w), (u, v)^m\}$ em G existe um caminho ou semi-caminho de v para u sem passar pela aresta (u, w) em todo grafo de chaveamento de expansão $S_e(G)$ e o grau de entrada sólido de v é igual a zero.

Definição 5.13 (**derivação N-grafo (N-grafo)).** Um grafo de prova G é uma *derivação N-grafo* (ou simplismente *N-grafo*) sse a meta-condição for satisfeita por G e todo grafo de chaveamento $S(G)$ for acíclico e conexo.

A definição de N-grafo é bem similar à de rede de prova dada pelo critério de Danos-Regnier, restringindo os subgrafos gerados pela remoção de uma aresta de alguns links específicos a serem acíclicos e conexos. Temos apenas a adição da meta-condição para validar o descarte de hipóteses, operação esta que é possível nos N-grafos, mas que não

existe nas redes de prova.

Com definição de N-grafos foi possível provar a corretude e completude deste sistema a partir de um mapeamento do mesmo com o cálculo de sequentes para lógica clássica (LK) definido por Gentzen em [Gen35]. O mapeamento foi definido com estes dois teoremas:

Teorema 5.1 (**mapeamento para os N-grafos).** *Dada uma derivação* Π *de* $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ *no cálculo de sequentes clássico, é possível construir um N-grafo correspondente* $NG(\Pi)$ *cujos elementos em* $PREMIS(NG(\Pi))$ *e* $CONC(NG(\Pi))$ *tem uma correspondencia um-para-um com as fórmulas* $A_1, \ldots, A_n$ *e* $B_1, \ldots, B_m$*, respectivamente.*

Teorema 5.2 (**sequentização).** *Dado um N-grafo* G*, existe uma derivação em cálculo sequente clássico* $SC(G)$ *de* $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ *cujas fórmulas em* $A_1, \ldots, A_n$ *e* $B_1, \ldots, B_m$ *tem uma correspondencia um-para-um com os elementos de* $PREMIS(G)$ *e* $CONC(G)$*, respectivamente.*

Os dois teoremas foram provados em [dO01].

5.3.1 Análise do fragmento $\wedge$, $\vee$ e $\neg$

Para o fragmento dos N-grafos com operadores lógicos $\vee$, $\wedge$ e $\neg$ apenas, ou seja, sem a implicação, a inadequação ocorre quando temos um ciclo inválido. Porém, os grafos de prova considerados inválidos pelo critério visto na seção anterior não são apenas os que contém tais ciclos: os grafos de prova que estejam associados a grafos de chaveamento desconexos também são considerados inválidos, mesmo sem ciclos. Este caso é exemplificado na Figura 5.11.

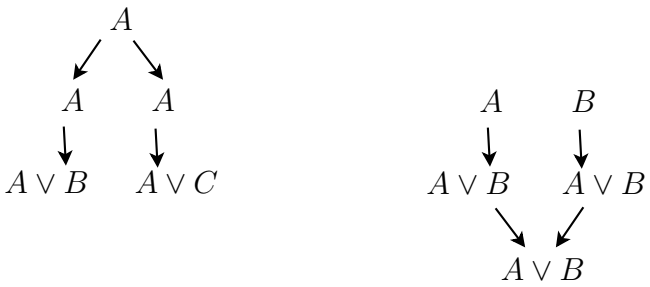

Figura 5.11 Grafos de prova sem ciclos, mas inválidos.

$$\dfrac{\dfrac{A \vdash A}{A \vdash A \vee B}\vee\text{-}IS}{A, B \vdash A \vee B} \qquad \dfrac{\dfrac{B \vdash B}{B \vdash A \vee B}\vee\text{-}IS}{A, B \vdash A \vee B}$$

Tabela 5.1 Derivaçoes de $A, B \vdash A \vee B$

Considerando a prova apenas pelo aspecto algébrico, as provas exemplificadas pela Figura 5.11 são consideradas corretas, pois $A, B \vdash A \vee B$ e $A \vdash (A \vee B), (A \vee C)$ são sempre verdade. Porém, semanticamente, o que é feito nestes casos é a junção de duas provas distintas em uma única prova, a partir do agrupamento das premissas e conclusões por meio da expansão e da contração. No primeiro exemplo, temos duas provas de $A \vee B$, e no segundo temos duas provas a partir de A.

Se este tipo de operação fosse permitida, isto seria equivalente a obter uma derivação de $A \vee B$ a partir de duas premissas (A e B) ao mesmo tempo, quando na verdade na lógica clássica podemos chegar a esta conclusão a partir de uma premissa apenas. A segunda premissa é adicionada por enfraquecimento da prova, como vemos na Tabela 5.1. Por isso, mesmo esta prova sendo algebricamente uma afirmação verdadeira, ela não pode ser considerada uma prova válida na lógica clássica.

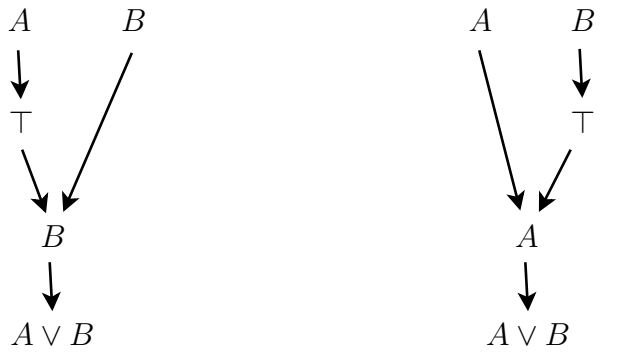

Figura 5.12 N-grafos de $A, B \vdash A \vee B$ em cálculo de sequentes.

De maneira análoga, para o exemplo com o link de expansão $(A \vdash (A \vee B), (A \vee C))$ o mesmo se aplica. O enfraquecimento aqui é feito no sucessor do sequente para gerar as duas conclusões, como mostrado na Tabela 5.2.

$$\dfrac{\dfrac{A \vdash A}{A \vdash A \vee B}\vee\text{-}IS}{A \vdash A \vee B, A \vee C} \qquad \dfrac{\dfrac{A \vdash A}{A \vdash A \vee C}\vee\text{-}IS}{A \vdash A \vee B, A \vee C}$$

Tabela 5.2 Derivaçoes de $A \vdash (A \vee B), (A \vee C)$ em cálculo de sequentes.

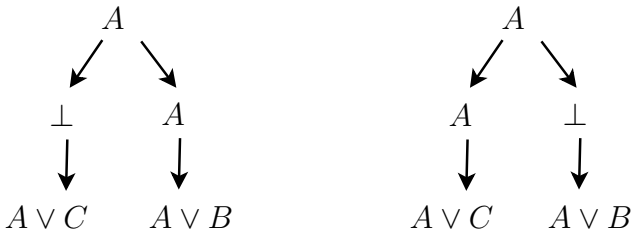

Figura 5.13 N-grafos de $A \vdash A \vee B, A \vee C$.

5.3.2 Análise da meta-condição

A intuição por trás da meta-condição é a mesma descrita na Seção 5.2: garantir o cancelamento de todas as cópias geradas a partir da ocorrência descartada. O critério, ao restringir todas os grafos de chaveamento de expansão, tem a intensão de controlar as expansões entre v e u. Este controle visa o seguinte critério: se entre v e u existe uma fórmula A que é expandida, ela deve fechar um ciclo antes de se chegar em u, como mostra a Figura 5.14.

Isto garante que os dois ramos gerados por uma conjunção (expansão) a partir de v (ou de alguma conclusão obtida a partir dele), foram utilizados para chegar em u, e não serão utilizados posteriormente com $v \to u$ (ou com alguma conclusão obtida a partir dele) de maneira conjuntiva.

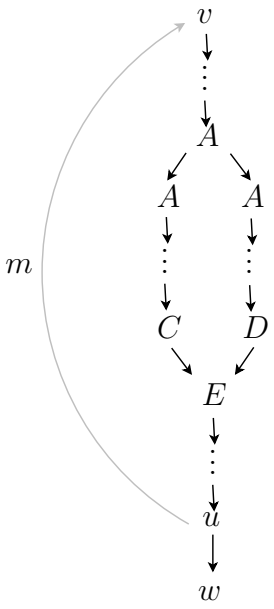

Figura 5.14 Meta-aresta com expansão.

CAPÍTULO 6

VERIFICAÇÃO DE GRAFOS DE PROVA

A primeira parte do assistente a ser definida para os N-grafos é uma técnica de verificação de grafos de prova. Como visto no Capítulo 2, esta é a parte mais simples de um assistente, onde os passos atômicos devem ser verificados um a um, assim como a conexão dos mesmos em uma derivação final. Por trabalharmos com grafos de prova, que naturalmente são estruturas geométricas, a estrutura geral do verificador será abandonada em favor de uma abordagem mais direcionada ao sistema com o qual estamos trabalhando.

Certificar a aplicação de cada derivação atômica ainda faz parte do processo de verificação. Porém, para a definição das técnicas de validação a seguir, assume-se que o grafo em questão já é um grafo de prova, ou seja, todas as regras representadas no grafo são válidas. Será verificado a validade do grafo de prova, se as derivações contidas no mesmo formam uma estrutura que representa uma prova logicamente correta. Em outras palavras, se o grafo de prova é um N-grafo.

Pode-se dizer que os teoremas e algoritmos descritos neste capítulo são o núcleo do verificador de grafos de prova, e uma estrutura para a validação da entrada dada pelo usuário deve ser adicionada ao transformar este projeto em um programa executável. Sobre como esta validação deve ser feita, fica a critério de implementação, dependendo da interface de comunicação com o usuário e os requisitos do produto a ser desenvolvido. O importante é filtrar previamente os grafos que não constituem grafos de prova.

6.1 ANÁLISE SOBRE CICLOS

No critério de corretude definido por de Oliveira, todo grafo de chaveamento associado a G deve ser acíclico para que o mesmo possa ser um N-grafo. Desta maneira, ela definiu quais são os ciclos em G válidos: aqueles que não aparecem em nenhum grafo de chaveamento $S(G)$. As arestas que podem não existir em um chaveamento de G são as pertencentes à um link de expansão ou contração e as meta-arestas. Elas têm um papel

chave para a identificação de tais ciclos. Porém, durante a análise de um grafo de chaveamento $S(G)$ qualquer associado a G, não pode-se assumir a existência de tais arestas. Por esta característica, elas são denominadas aqui *arestas voláteis*.

Definição 6.1 (**aresta volátil**). Uma aresta (u, v) de um grafo de prova G é dita *volátil* se a mesma pertencer a um link de expansão ou contração, ou for uma meta-aresta.

Além disso, duas arestas de um link de expansão ou contração têm uma ligação semântica muito forte ao se analisar os grafos de chaveamento: se uma delas estiver em um dado chaveamento $S(G)$, a outra não pode estar. Por isso, estas duas arestas serão chamadas de *arestas conjugadas*.

Definição 6.2 (**aresta conjugada**). Sejam e_1 e e_2 arestas pertencentes a um link de expansão ou contração de um grafo de prova G. A *aresta conjugada* de e_1, representada por $\hat{e}_1$, é a aresta e_2, e virce-versa. Observe que, sendo $S(G)$ um grafo de chaveamento associado a G, temos $e \in S(G) \iff \hat{e} \notin S(G)$.

Seja c um ciclo em um grafo de prova G. Define-se como $E_V(c)$ o conjunto das arestas voláteis de c. Analisando um ciclo qualquer quanto à presença de arestas voláteis, temos os seguintes tipos de ciclo, ilustrados na Figura 6.1:

i) ciclos sem arestas voláteis;

ii) ciclos c com um conjunto de arestas voláteis $E_V(c) = \{e_1, \ldots, e_n\}, n > 0$, onde $\forall\ e_i\ \in E_V(c)\ \rightarrow\ \hat{e}_i\ \notin\ E_V(c)$;

iii) ciclos c com um conjunto de arestas voláteis $E_V(c) = \{e_1, \ldots, e_n\}, n > 0$, onde $\exists\ e_i\ \in E_V(c)\ \rightarrow\ \hat{e}_i\ \in\ E_V(c)$;

Estudando cada um destes ciclos separadamente, percebemos que os do tipo i são sempre inválidos, pois como não possuem aresta volátil, ele está presente em todos os grafos de chaveamento associados a G.

Os ciclos do tipo ii podem não ocorrer em alguns chaveamentos, quando alguma das arestas voláteis do mesmo não pertencer a $S(G)$. Porém, quando nenhuma destas ares-

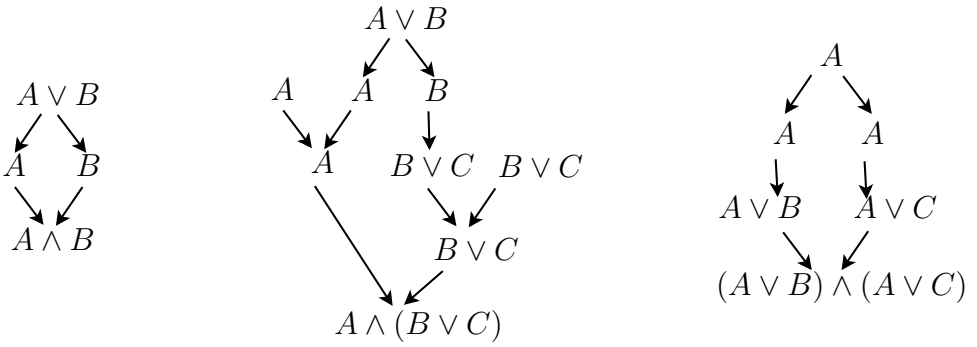

Figura 6.1 Ciclos em um grafo de prova.

tas voláteis é uma meta-aresta, existe pelo menos um chaveamento que contém [1] este ciclo c: basta pegar um que contém todas as arestas de $E_V(c)$. Em outras palavras, $E_V(c) \subseteq S(G)$ para algum chaveamento $S(G)$. Este chaveamento de G existe, pois não há um par de arestas conjugadas em $E_V(c)$: logo, é possível remover as conjugadas de todas as arestas de $E_V(c)$, criando um chaveamento que contém este ciclo. Se uma das arestas voláteis deste ciclo for uma meta-aresta o ciclo é válido, pois todos os grafos de chaveamento associados a G não possuem nenhuma meta-aresta. Senão, o ciclo é inválido.

O último caso é o mais interessante, pois ele define apenas os ciclos válidos. Qualquer ciclo do tipo *iii* é tal que nenhum grafo de chaveamento $S(G)$ o contêm. Isso porque, como existe uma aresta cuja conjugada também está no ciclo, uma das duas deve "desaparecer" em todo chaveamento associado a G. Estes ciclos são exatamente os que contém pelo menos um link de expansão ou contração.

A partir disto, temos o seguinte teorema:

Teorema 6.1 (**ciclo válido**). *Seja G um grafo de prova. Defina G' como o subgrafo gerador de G sem as meta-arestas. Um ciclo c em G' contém um link de expansão ou contração sse ele não existir em todo grafo de chaveamento $S(G)$ associado a G.*

Demonstração.

i) *Se um ciclo c em G' contém um link de expansão ou contração, então ele não existe em todo grafo de chaveamento $S(G)$ associado a G.*
Seja l um link de expansão ou contração contido em c. Seja $S(G)$ um grafo de

[1] Aqui, ciclos, grafos de chaveamento e links são tratados como conjuntos de arestas. Portanto, os termos "conter um ciclo c", "o ciclo c contém o link l", e semelhantes significam a relação *contém* padrão entre conjuntos: X está contido em Y ($X \subseteq Y$) sse todo elemento de X for elemento de Y ($\forall x \in X \mid x \in Y$).

chaveamento associado a G. Como $S(G)$ deve eliminar uma das arestas de l, então o ciclo c vai ser desconectado e não pode existir em $S(G)$. Como o $S(G)$ é qualquer, temos que todos os grafos de chaveamento associados a G não contêm o ciclo c.

ii) *Se todo grafo de chaveamento $S(G)$ associado a G não contém o ciclo c, então c contém um link de expansão ou contração.*
Pela contrapositiva, suponha que o ciclo c não contém nenhum link de expansão ou contração (observe que aqui ele não contém o link completo, mas pode conter apenas uma das arestas voláteis de alguns links). Tome como $E_V(c)$ o conjunto de todas as arestas volateis de c. Como $E_V(c)$ não possui nenhuma meta-aresta, uma vez que o ciclo existe em G', e não possui nenhum par de arestas conjugadas e, $\hat{e}$, existe um grafo de chaveamento $S(G)$ associado a G tal que $\forall e_i \in E_V(c) \mid e_i \in S(G)$. Logo, este ciclo existe no chaveameno $S(G)$. Assim, existe um grafo de chaveamento asociado a G que contém o ciclo c.

∎

Estes ciclos são denominados ciclos válidos. A partir desta prova segue o seguinte corolário:

Corolário 6.1 (**aciclicidade**). *Seja G um grafo de prova. Defina G' como o subgrafo gerador de G sem todas as meta-arestas. Todo ciclo em G' contém um link de expansão ou contração sse todo grafo de chaveamento $S(G)$ associado a G for acíclico.*

6.2 ANÁLISE SOBRE A CONECTIVIDADE

O critério definido por de Oliveira não restringe os grafos de chaveamento de um grafo de prova a serem acíclicos apenas. Eles devem ser acíclicos e *conexos*. Um grafo não direcionado acíclico e conexo define uma *árvore*. Para verificar a validade de todo chaveamento de um grafo de prova basta então verificar se todos eles são árvores [2]. Por definição, uma árvore com n vértices contém exatamente $n - 1$ arestas [Har72].

Teorema 6.2 (**propriedade de árvores**). *Seja $G = (V, E)$ um grafo não orientado. Se G é acíclico, então $|E| = |V| - 1$.*

[2] Aqui novamente é ressaltado que para a verificação de conectividade, as direções das arestas são irrelevantes. Logo, quando se falar em caminho e ciclo, na verdade o significado, estritamente falando, é semi-caminho e semi-ciclo. Ver mais detalhes no Apêndice A.

Com o Teorema 6.2, é possível extender o seguinte teorema para os grafos de prova:

Teorema 6.3 (**conectividade**). *Seja G um grafo de prova tal que todo grafo de chaveamento $S(G)$ associado a G seja acíclico. Então todo grafo de chaveamento $S(G)$ associado a G é uma árvore sse a seguinte fórmula é válida:*

$$|E(G)| - |L_E(G)| - |L_C(G)| - |E_M(G)| = |V(G)| - 1$$

onde $E(G)$ é o conjunto de todas as arestas de G, $L_E(G)$ e $L_C(G)$ são os conjuntos de todos os links de expansão e contração de G, respectivamente, $E_M(G)$ é o conjunto de todas as meta-arestas de G, e $V(G)$ é o conjunto de todos os vértices de G.

Demonstração. Em todo grafo de chaveamento associado a G, todas as meta-arestas são removidas, assim como uma aresta de cada link de expansão e contração. Assim, todo chaveamento de G possui $|E(G)| - |L_E(G)| - |L_C(G)| - |E_M(G)|$ arestas. Pelo Teorema 6.2, se todo chaveamento de G for acíclico, então todo chaveamento de G é uma árvore sse o número de arestas for igual ao número de vértices menos 1. Como todo grafo de chaveamento tem exatamente $|E(G)| - |L_E(G)| - |L_C(G)| - |E_M(G)|$ arestas, e $|V(G)|$ vértices, todo grafo de chaveamento $S(G)$ associado a G é uma árvore sse $|E(G)| - |L_E(G)| - |L_C(G)| - |E_M(G)| = |V(G)| - 1$ ∎

6.3 O CASO DA META-CONDIÇÃO

A meta-condição a princípio pode parecer um caso particular da condição de conectividade proposta sobre os grafos de chavemento, mas, ao estudarmos mais a fundo a estrutura dos grafos de prova com meta-arestas válida e inválidas, vemos que se trata de um caso bem especial. Inicialmente ela restringe os vértices v de todas as meta-arestas $(u, v)^m$ a terem o grau de entrada sólido igual a zero. Esta condiçao é na verdade a mais simples de verificar, com um custo linear sobre o número de meta-arestas: basta percorrer todas as meta-arestas, verificando se a lista de adjacência do vértice v de um link $(u, v)^m$ contém apenas esta aresta (uma vez que G é um grafo de prova, e nos grafos de prova cada vértice pode ser conclusão de no máximo um link).

A outra restrição imposta pela meta-condição é mais traiçoeira: todo grafo de chaveamento de extensão associado a G deve conter um caminho de u para v sem passar pela aresta (u, w), para todo link $\{(u, w), (u, v)^m\}$. Esta afirmação pode ser traduzida para a seguinte: todo grafo de chaveamento de extensão associado a G, com a remoção da aresta

(u, w), tem u e v no mesmo componente conexo. Daí temos a similaridade com a restrição de conectividade imposta sobre os chaveamentos associados a G.

Porém, embora tenha sido alcançado um teorema para o caso dos chaveamentos, que verifica a conectividade do mesmo por meio de uma simples aplicação de fórmula, o mesmo não pode ser feito para este caso. A figura Figura 6.2 mostra dois grafos de prova com o mesmo número de vértices, arestas, links de expansão e contração e meta-arestas, onde o primeiro é um exemplo de aplicação válida do descarte de hipótese, mas o segundo não.

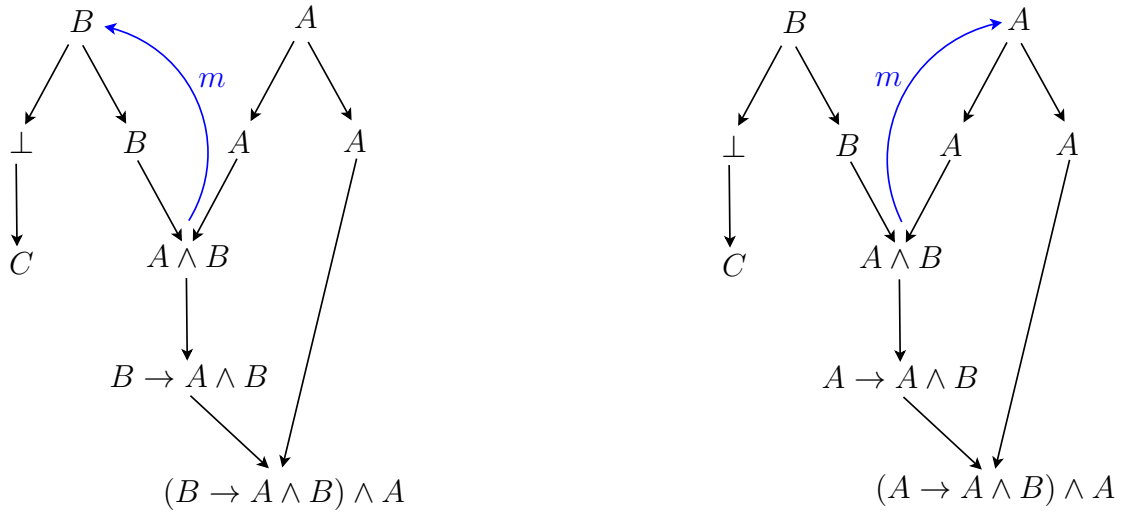

Figura 6.2 Exemplos de grafos com meta-aresta: o primero é um N-grafo, e o segundo é inválido.

Neste exemplo fica claro que a diferença está ligada à estrutura do grafo de uma maneira mais intrínseca, que não pode ser traduzida de maneira direta a alguma propriedade externa do mesmo.

O problema aqui surge quando, ao invés de utilizar grafos de chaveamento, de Oliveira define a meta-condição apenas sobre os grafos de chaveamento de *expansão*. Na prática, o que isto faz é devolver as arestas de contração que são removidas de todo chaveamento, quando existem links de contração no grafo. Como todo grafo de chaveamento deve ser uma árvore, conforme visto na seção anterior, se o grafo contém links de contração, então todo grafo de chaveamento de expansão é cíclico. Estes ciclos que podem existir nos chaveamentos de expansão na verdade é o que torma mais difícil a tarefa de verificar se

todo chaveamento de expansão conecta u e v sem passar por (u, w). Isto pode ser visto na abordagem a seguir.

Uma vez que temos definido teoremas que facilitam a verificação de grafos de prova sem a meta-aresta, uma maneira de lidar com a mesma seria aplicar uma transformação τ sobre o grafo de prova G de modo que G é um N-grafo sse $\tau(G)$ o é, e $\tau(G)$ não contém meta-arestas. Esta tranformação baseia-se na seguinte igualdade da lógica proposicional:

$$P \to Q \iff \neg P \vee Q$$

A idéia é definir τ de tal modo que toda ocorrência da fórmula $v \to u$ seria substituída por $\neg v \vee u$, e as seguintes alterações seriam feitas:

- Toda vez que $\to -E$ fosse aplicada em $(v \to u)$ e v, esta aplicação seria substituída por uma $\vee - E$ em $\neg v \vee u$, seguida de um $\perp - link$ em $\neg v$ e v, sobrando como conclusões $\perp$ e u. Esta transformação é ilustrada na Figura 6.3.

- Toda hipótese cancelada v por uma meta-aresta $\{(u, w), (u, v)^m\}$, adicionamos o componente mostrado na Figura 6.4, como ilustrado na Figura 6.5.

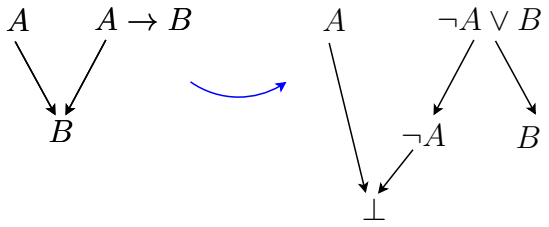

Figura 6.3 Substituição de $\to -E$ por $\vee - E$ e $\perp - link$.

Assim, os vértices v de todas as meta-arestas $(u, v)^m$ deixam de ser premissa, o cálculo tem a mesma semântica, apesar de ter fórmulas diferentes para uma mesma afirmação, e apenas premissas do tipo $\top$ e conclusões do tipo $\perp$ foram adicionadas, o que deixaria as provas equivalentes.

Se, além de transformar mantendo o significado semântico da prova, esta transformação não tornasse o N-grafo inválido quando o original o era, e virse-versa, seria a transformação ideal para a verificação eficiente da meta-condição. Isto porque esta transformação também é polinomial, dependendo apenas do número de meta-arestas

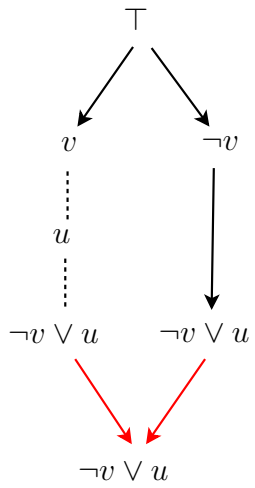

Figura 6.4 Componente removedor de meta-aresta.

existentes no grafo. Primeiramente, vamos verificar G *inválido* $\rightarrow \tau(G)$ *inválido*:

i) se existe ciclo em algum chaveamento de G, τ não o quebra pois apenas adiciona componentes.

ii) se algum chaveamento de G é desconexo, continua sendo, pois o componente adicionado por τ não é um caminho em nenhum chaveamento de $\tau(G)$, pois tem um link de contração completo. Outra maneira de verificar é pela fórmula do Teorema 6.3: $\tau(G)$ adiciona 4 vértices, 3 arestas, 1 link de contração e remove 1 meta-aresta. Assim, temos que na fórmula de τ fica:

$$|E(\tau(G))| - |L_C(\tau(G))| - |L_E(\tau(G))| - |E_M(\tau(G))| = |V(\tau(G))| - 1$$
$$(|E(G)| + 5 - 1) - (|L_C(G)| + 1) - |L_E(G)| - (|E_M(G)| - 1) = (|V(G)| + 4) - 1$$

Entretanto, a volta não é verdadeira. Ela é capaz de gerar um ciclo em alguns grafos de chaveamento de $\tau(G)$, mesmo quando G é válido. Isto é mostrado no contra-exemplo da Figura 6.6. Analisando este caso, vemos que isso ocorre porque existe um caminho em alguns chaveamentos de G entre v e w, sem passar por u. Isto significa que não existe caminho em nenhum chaveamento entre v e u sem passar por (u, w), pois senão existiriam chaveamentos cíclicos em G. Mas o fato de não existir tal chaveamento não significa que não exista ligação em todo chaveamento de expansão. Este é o caso onde o ciclo que há nos chaveamentos de expansão quebra toda a intuição por trás da transformação.

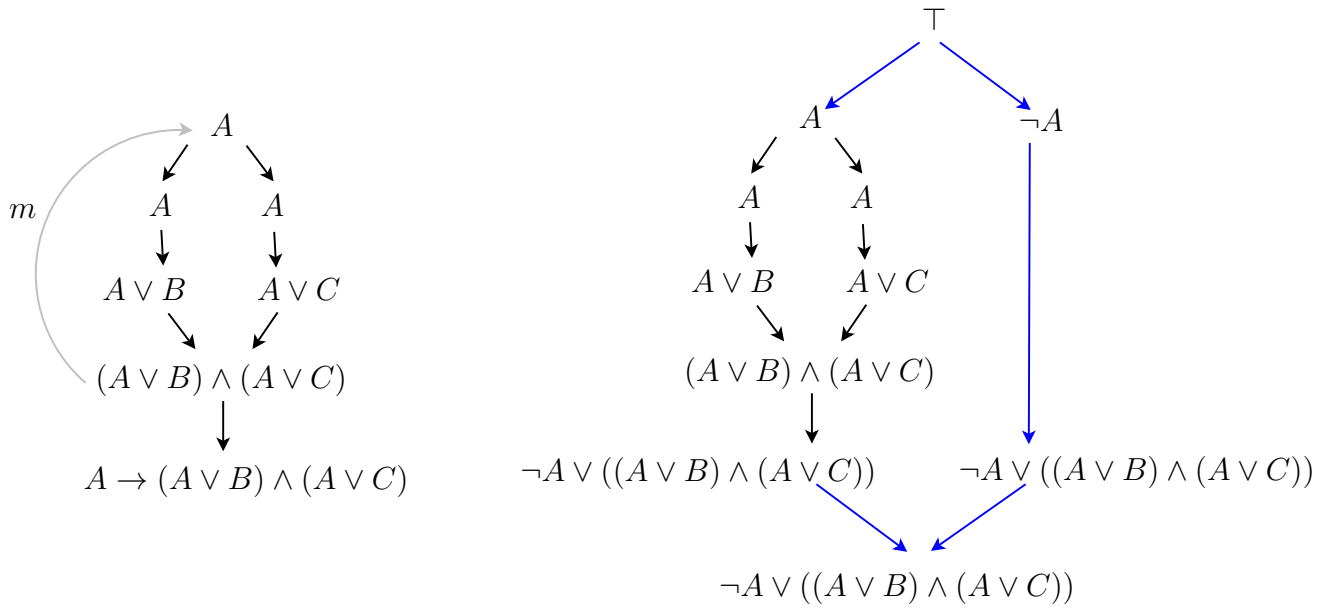

Figura 6.5 Exemplo de adição do componente removedor de meta-aresta.

Podemos afirmar, porém, que se o grafo não possui link de contração, a transformação τ preserva a validade de G. Assim, ela pode ser aplicada em toda meta-aresta para validar a meta-condição em G. Isto porque, como não existem contrações, grafos de chaveamento e grafos de chaveamento de expansão são a mesma coisa. Assim, se um ciclo se fechar, o que ocorre apenas quando existe caminho entre v e w sem passar por u, como vemos na Figura 6.7, significa que não pode existir caminho em nenhum chaveamento (de extensão) entre v e u sem passar por (u, w) (senão G tem chaveamento cíclico). Assim, a meta-condição não é válida.

6.4 NOVO MECANISMO DE VALIDAÇÃO

A partir do Corolário 6.1 e do Teorema 6.3, podemos redefinir N-grafos:

Teorema 6.4 (**N-grafos (reformulado)**). *Seja G um grafo de prova e G' o subgrafo gerador de G sem todas as meta-arestas. Defina $V(G)$ e $E(G)$ como o conjunto de vértices e arestas de G, respectivamente, e $L_C(G)$, $L_E(C)$ e $E_M(G)$ como os conjuntos dos links de contração, expansão e das meta-arestas, respectivamente. G é um* N-grafo *sse as seguintes afirmações são válidas:*

i) todo ciclo em G' contém um link de expansão *ou* contração*;*

ii) a fórmula $|E(G)| - |L_E(G)| - |L_C(G)| - |E_M(G)| = |V(G)| - 1$ é válida;

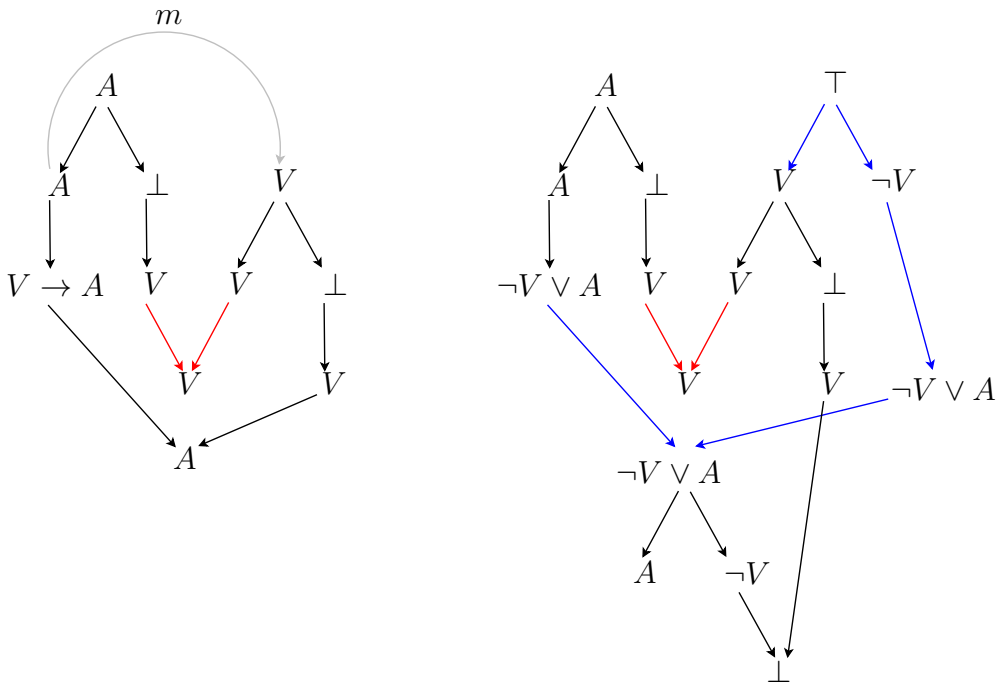

Figura 6.6 Exemplo onde um ciclo é adicionado por τ.

iii) a meta-condição é satisfeita por G.

Demonstração. A afirmativa i, pelo Corolário 6.1, diz que todo grafo de chaveamento associado a G é acíclico. De acordo com o Teorema 6.3, a afirmativa ii implica em todo chaveamento de G ser uma árvore. Esta última afirmação é equivalente à utilizada por de Oliveira ao definir os N-grafos como tendo todo chaveamento acíclico e conexo. Com a adição da meta-condição, as duas definições de N-grafos são equivalentes. ∎

Este teorema mostra uma outra forma de verificar a validade de um grafo de prova, porém muito mais eficiente para o fragmento do $\wedge, \vee$ e $\neg$. Para o caso da implicação a verificação permanece a mesma, ainda checando a meta-condição definida por de Oliveira. O custo da verificação do item i é polinomial, e na verdade pode ser linear, uma vez que é necessária apenas a verificação de todos os ciclos em um grafo não-direcionado. O item ii tem um custo constante na aplicação da fórmula definida sobre o tamanho do grafo G, resumido apenas à contagem dos links e arestas relevantes para a mesma (pode ser considerado aqui um custo linear para esta contagem, mas a mesma pode ser inclusa na leitura do grafo, ou seja, pode ser considerado uma informação a priori para o algoritmo).

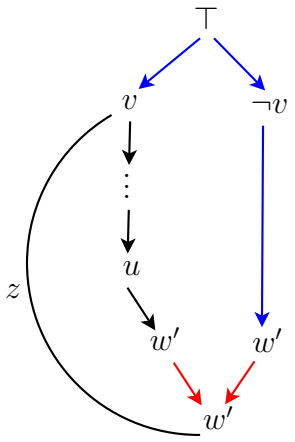

Figura 6.7 Caso genérico onde um ciclo é adicionado por τ: A ligação de v a w' por z, juntamente com o novo componente, gera um ciclo que existe em alguns grafos de chaveamento.

Para o caso da meta-aresta, como visto na seção anterior, não foi possível encontrarmos uma equivalência com alguma propriedade cujo custo de verificação fosse polinomial. Neste trabalho, a verificação completa, aplicável para todos os N-grafos, tem um custo exponencial por causa desta condição.

6.5 ALGORITMO

O algoritmo para a verificação segue dos teoremas definidos anteriormente. Um procedimento será definido para verificar cada um dos três itens definidos no Teorema 6.4, e um procedimento geral será formulado a partir dos demais.

O primeiro procedimento a ser definido é uma variação da *busca em profundidade* (em inglês, *Depth First Search*, ou *DFS)*). A versão que será utilizada como base para este algoritmo foi tirada do livro de Cormen [THCS02]. A estratégia aqui é montar uma árvore geradora do grafo, em profundidade, de modo que quando uma aresta de retorno for identificada, delimitando um ciclo no grafo original, o critério i possa ser validado em tempo constante.

Para isso, durante a busca é necessário manter um registro das arestas voláteis visitadas durante o caminho. Com este registro, quando um ciclo for definido será possível identificar os links completos que foram "descobertos" na busca, mas que ainda não foram

"terminados", e ainda que estão entre o vértice inicial e final do ciclo. A intuição fica mais clara com o exemplo utilizando o grafo de prova descrito em Figura 6.8.

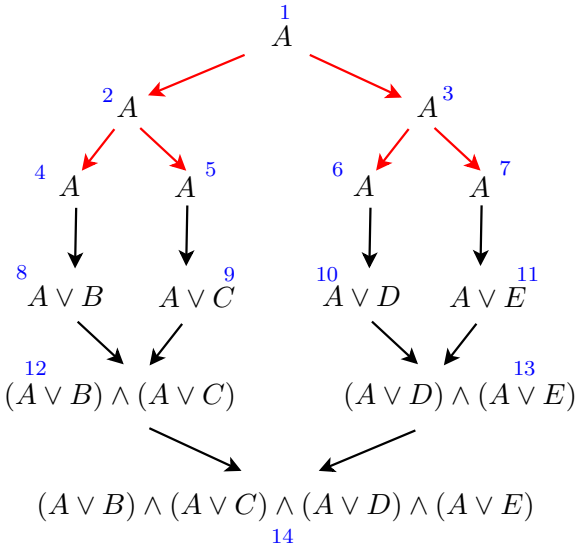

Figura 6.8 Grafo de prova utilizado para demonstração do algoritmo. As arestas de expansão estão destacadas, e os vértices estão numerados para facilidar futuras referências.

Observe que no exemplo visto na figura Figura 6.9, o ciclo $c1$ encontrado é válido. Porém, no caminho desde a raiz foram "descobertos" três arestas voláteis. Como verificar se um par de arestas descobertas são conjugadas? E além disso, a primeira aresta volátil percorrida na busca não está inclusa no ciclo. Como identificar se uma dada aresta volátil percorrida está no ciclo recém encontrado? E, mais importante, como fazer isso de maneira mais eficiente?

Para isto, uma extensão sobre os conceitos do algoritmo de busca em profundidade será proposta. Além dos vértices visitados, se manterá um registro das arestas percorridas. Cormen em [THCS02] sugere uma notação a ser aplicada nos vértices durante a execução da busca que define estados para os mesmos. Um vértice em uma busca em profundidade pode ter sido *não visitado*, quando a busca ainda são passou por este vértice, *descoberto*, quando ele foi alcançado, mas as arestas incidentes ao mesmo ainda não foram todas percorridas, ou pode estar *terminado*, quando todas as arestas incidentes ao mesmo foram percorridas.

De maneira similar, definimos aqui os mesmos estados para arestas, onde uma aresta

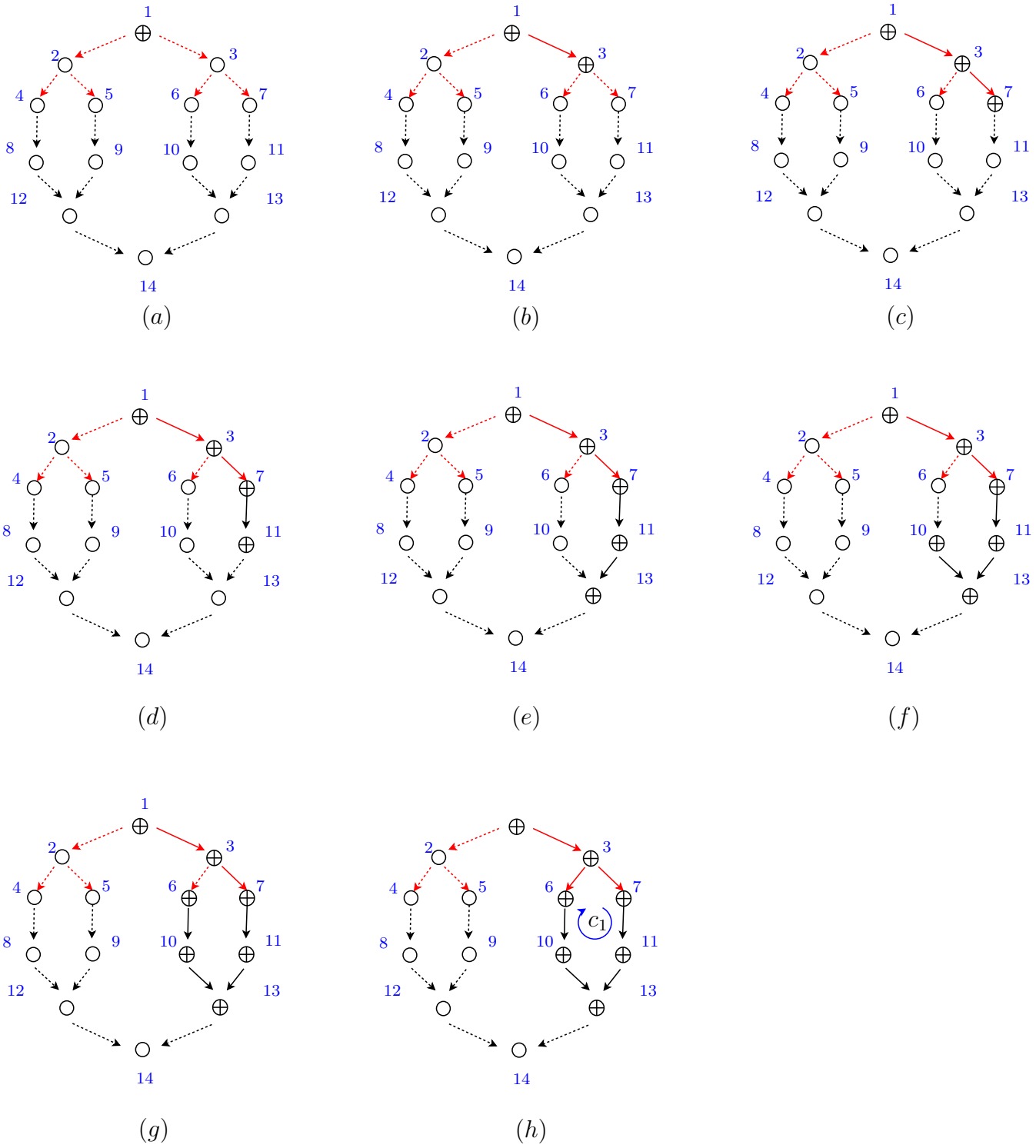

Figura 6.9 Exemplo do algoritmo de busca. As bolas vazias representam vértices não visitados, as bolas com uma cruz representam vértices descobertos e as bolas com um X representam vértices terminados.

pode não ter sido visitada, quando ela ainda não foi percorrida pela busca, ter sido descoberta, quando ela foi percorrida em um sentido na ida, mas não na volta, ou ela pode estar terminada, quando a mesma foi percorrida na ida e na volta da recursão.

Além destes estados, a DFS também identifica os vértices por um *carimbo de tempo*. Cada vértice v tem dois carimbos de tempo: o primeiro, chamado $d[v]$ registra quando v é descoberto pela primeira vez, e o segundo carimbo de tempo $f[v]$, que registra o momento em que a busca pelas arestas incidentes a v foi finalizada, e o vértice foi para o estado terminado. Estes carimbos são bastante utilizados em muitos algoritmos de grafos e são úteis no raciocínio sobre o comportamento da busca em profundidade.

De maneira análoga novamente, estes carimbos são extendidos para que possam ser aplicados em vértices e arestas. Assim, cada vértice v recebe um carimbo $\delta[v]$ sempre que na DFS ele recebe um $d[v]$, e um carimbo $\zeta[v]$ sempre que ele recebe um $f[v]$, registrando os mesmos momentos de descoberta e finalização da busca. Cada aresta e tem os mesmos carimbos, mas com um significado diferente: o primeiro, $\delta[e]$, registra quando a aresta é percorrida pela primeira vez, e o segundo $\zeta[e]$ quando ela é percorrida de volta, no retorno da busca recursiva feita pela mesma.

O instante de tempo utilizado para registrar a pasagem por uma aresta não é um instante novo, mas sim um mesmo utilizado em algum dos vértices incidente à mesma. Assim, a passagem por arestas continua sendo "instantânea", como é naturalmente na busca original. Os carimbos de tempo registrados nas arestas coincide com os carimbos de alguns vértices. Isso gera um problema em definir uma relação de *maior que* entre os rótulos, cujos valores são utilizados para provar diversas propriedades da busca. O ideal seria que, agora que arestas também podem ser carimbadas, elas pudessem ser ordenadas entre si e entre os vértices, em uma ordenação total.

Para isso, a seginte ordenação é definida sobre os carimbos: (A intuição destes operadores pode ser vista na Figura 6.10).

Definição 6.3. Dados dois elementos de um grafo $G(V, E)$, u e v, define-se o operador $\delta[u] \prec \delta[v]$:
a) Se u e v são vértices, temos $\delta[u] \prec \delta[v]$ *sse* $\delta[u] < \delta[v]$;
b) Se u e v são arestas, temos $\delta[u] \prec \delta[v]$ *sse* $\delta[u] < \delta[v]$;

c) Se u é uma aresta e v um vértice, temos $\delta[u] \prec \delta[v]$ *sse* $\delta[u] \leq \delta[v]$.

Definição 6.4. Dados dois elementos de um grafo $G(V, E)$, u e v, define-se o operador $\zeta[u] \prec \zeta[v]$:
a) Se u e v são vértices, temos $\zeta[u] \prec \zeta[v]$ *sse* $\zeta[u] < \zeta[v]$;
b) Se u e v são arestas, temos $\zeta[u] \prec \zeta[v]$ *sse* $\zeta[u] < \zeta[v]$;
c) Se v um vértice e u é uma aresta, temos $\zeta[v] \prec \zeta[u]$ *sse* $\zeta[v] \leq \zeta[u]$.

Obs.: Aqui, o operador $<$ é utilizado como o operador comum de comparação entre inteiros.

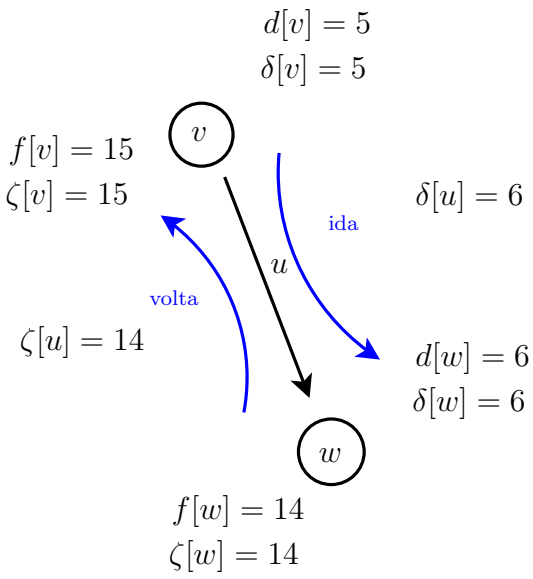

Figura 6.10 Ordenação dos valores de δ e ζ. Temos a seguinte ordenação:
$\delta[v] \prec \delta[u] \prec \delta[w] < \zeta[w] \prec \zeta[u] \prec \zeta[v]$

Na busca em profundidade original, temos sempre $d[v] < f[v]$. Com esta nova valoração isto continua sendo verdade, pois para os vértices utilizamos os mesmos carimbos d e f para definir δ e ζ. Para as arestas isto também ocorre, pois para uma aresta $e = \{u, v\}$, o valor δ virá sempre do valor registrado para o vértice u, se a mesma foi percorrida no sentido $u \to v$, e o valor ζ sempre virá de v, que irá percorrê-la na volta no sentido $v \to u$. Assim, temos $\delta[e] \prec \delta[v] < \zeta[v] \prec \zeta[e]$.

O teorema dos parêntesis, mostrado como Teorema A.1 no Apêndice A, também continua válido, e ainda tem o significado análogo extendido para as arestas: uma elemento

(vértice ou aresta) v de um grafo G é dito descendente de uma aresta u se ele foi descoberto depois de u ter sido percorrida na ida, e foi terminado antes de voltar por u. Ou seja, $\delta[u] \prec \delta[v] < \zeta[v] \prec \zeta[u]$.

Com estas expansões, é possível definir o seguinte algoritmo:

Algoritmo *VALIDAR-CICLOS*
Entrada $G(V, E)$: grafo de prova
Saída $TRUE$ se todos os ciclos sao válidos, senão $FALSE$
1 **para todo** vértice $v \leftarrow V(G)$ **faça**
2 $cor[v] \leftarrow BRANCO$
3 $\pi[v] \leftarrow NIL$
4 **para todo** link $l \leftarrow L_C(G) \cup L_E(G)$ **faça**
5 $ativo[l] \leftarrow FALSE$
6 $valido \leftarrow TRUE$
7 $tempo \leftarrow 0$
8 **para todo** vértice $v \in V(G)$ **faça**
9 **se** $cor[v] = BRANCO$ **então**
10 $DFS - VISITE(v)$ **devolva** $valido$
Algoritmo *DFS-VISITE*
Entrada u: vértice a ser visitado
1 $cor[u] \leftarrow CINZA$
2 $tempo \leftarrow tempo + 1$
3 $\delta[u] \leftarrow tempo$
4 **para todo** aresta $(u, v) \in Adj[u]$ **faça**
5 $\delta[(u, v)] \leftarrow tempo$
6 **se** (u, v) é volátil **então**
7 $l \leftarrow$ link que contém (u, v)
8 **se** $ativo[l]$ **então**{*O link já foi ativado, agora está sendo completo*}
9 $pilha \leftarrow l$
10 $ativou \leftarrow FALSE$
11 **senão**{*O link está sendo ativado agora*}
12 $ativo[l] \leftarrow TRUE$
13 $ativou \leftarrow TRUE$
14 **se** $cor[v] = BRANCO$ **então**

15 $\pi[v] \leftarrow u$
16 $DFS - VISITE(v)$
17 **senão**{*Esta é uma aresta de retorno*}
18 $\{e_1, e_2\} \leftarrow$ *topo da pilha*
19 **se** *pilha vazia* OU $min(\delta[e_1], \delta[e_2]) \prec \delta[v]$ **então**{*O último link completo está antes do ciclo*}
20 $valido \leftarrow FALSE$ {*Ciclo inválido encontrado*}
21 **se** (u, v) é volátil E *ativou* **então**{*O link foi ativado por esta aresta*}
22 $ativo[l] \leftarrow FALSE$
23 **senão**{*O link foi completado por esta aresta*}
24 *remove topo da pilha*
25 $\zeta[(u, v)] \leftarrow tempo$ {*Uma aresta não conta o tempo*}
26 $cor[u] \leftarrow PRETO$
27 $\zeta[u] \leftarrow tempo \leftarrow tempo + 1$

O segundo algoritmo é a simples aplicação da fórmula definida no Teorema 6.3.

Algoritmo *VALIDAR-CONECTIVIDADE*
Entrada $G(V, E)$: grafo de prova acíclico
Saída $TRUE$ se todos os grafos de chaveamento de G sao árvores, senão $FALSE$
1 $e \leftarrow |E(G)|$
2 $v \leftarrow |V(G)|$
3 $l_C \leftarrow$ número de links de contração em G
4 $l_E \leftarrow$ número de links de expansão em G
5 $e_M \leftarrow$ número de meta-arestas em G
6 **devolva** $e - l_C - l_E - e_M = v - 1$

O algoritmo final para o fragmento $\wedge, \vee$ e $\neg$ é uma composição dos dois algoritmos acima.

Algoritmo *VALIDAR-GRAFO*
Entrada $G(V, E)$: grafo de prova sem meta-arestas
Saída $TRUE$ se G é válido, senão $FALSE$
1 $valido \leftarrow VALIDAR - CICLOS(G)$

```
2 se valido então
3     valido ← VALIDAR − CONECTIVIDADE(G)
4 devolva valido
```

6.5.1 Corretude

A corretude do Algoritmo 6.5 vem do Corolário 6.1, cuja demonstração vem do Teorema 6.1. Com uma busca em profundidade, todos os ciclos de um grafo ficam destacados a partir das arestas de retorno do mesmo, que "fecham" estes ciclos. Quando tal aresta é encontrada, o algoritmo verifica se existe um link de expansão ou contração que foi percorrido completamente durante a formação do ciclo.

Uma aresta de retorno aponta para um nó $CINZA$. Ao chegar em um vértice u que aponta para o vértice v, podemos ter as seguintes situações:

1) Nenhuma aresta volátil foi percorrida: neste caso, nenhum link foi colocado na pilha, e logo o ciclo não contém nenhum link de expansão ou contração.

2) Uma aresta volátil está terminada: como ela foi terminada, ou ela ativou um link, e ao terminar na volta da recursão ela o desativa, ou ela adicionou o link na pilha, e ao terminar ela o removeu do topo. Em ambos os casos, o link que a contém não pode estar ativo nem na pilha.

3) Uma aresta volátil está $CINZA$, mas sua conjugada não foi visitada: logo o link foi ativado, mas não foi adicionado na pilha. Apenas quando a segunda aresta for percorrida enquanto esta aresta ainda está $CINZA$ é que ele será consolidado na pilha.

4) Uma aresta volátil está $CINZA$, mas sua conjugada foi terminada: como a conjugada foi terminada, pelo teorema dos parentesis ela já foi totalmente processada (ou seja, percorrida na ida e na volta da recursão). Assim, ela ativou o link quando foi descoberta, e o desativou quando foi terminada. Agora a aresta atual irá novamente ativar este link, mas ele não será colocado na pilha.

5) Duas arestas conjugadas estão $CINZA$: neste caso, o link que as contém foi adicionado na pilha. Ao encontrar uma aresta de retorno neste estado, deve-se apenas observar se este link do topo da pilha está entre os vértices inicial e final do ciclo encontrado.

6) Todas as arestas voláteis estão terminadas: neste estado a pilha está vazia e todos os links estão desativados, pois todas as arestas já foram percorridas, e consequentemente já ativaram e desativaram seus links, ou adicionaram e removeram o mesmo da pilha.

Para provar que tal algoritmo faz esta análise dos links, temos os seguintes lemas:

Lema 6.1. *Se um link $l = \{e_1, e_2\}$ está na pilha no momento em que o vértice v é visitado, então v é descendente de e_1 e e_2.*

Demonstração. Como o link está na pilha, isto significa que no momento em que a segunda aresta de l foi percorrida, a primeira já tinha ativado o link. Isto significa que uma aresta é descendente da outra, e ambas estão ativas. Quando a busca visita o vértice v, temos a seguinte configuração: $\delta[e_1] \prec \delta[e_2] \prec \delta[v]$. Pelo teorema dos parêntesis, a configuração final deve ser $\delta[e_1] \prec \delta[e_2] \prec \delta[v] < \zeta[v] \prec \zeta[e_2] \prec \zeta[e_1]$. ∎

Lema 6.2. *Se um link l está na pilha, então qualquer aresta de retorno para um vértice v tal que $\delta[v] \prec min(\delta[e_1], \delta[e_2])$ ou $\delta[v] = min(\delta[e_1], \delta[e_2])$ define um ciclo que contém este link.*

Demonstração. Se o link l está na pilha, ambas as suas arestas estão ativas. Isto significa que este vértice u é descendente de ambas. Se uma aresta de retorno for encontrada, o ciclo que será fechado é definido pelo caminho na árvore entre o vértice u e o v mais a aresta de retorno. Como este caminho entre v e u é único, temos três casos, ilustrados na Figura 6.11:

i) v é descendente de ambas as arestas de l, e assim $min(\delta[e_1], \delta[e_2]) \prec max(\delta[e_1], \delta[e_2]) \prec \delta[v]$. Neste caso, o ciclo não contém nenhuma das arestas de l, pois se restringe ao caminho entre u e v na árvore mais a aresta de retorno, e nenhum deles passa pelas arestas de l.

ii) e_1 é descendente de v, mas e_2 não é, e assim $min(\delta[e_1], \delta[e_2]) \prec \delta[v] \prec max(\delta[e_1], \delta[e_2])$. Neste caso, o caminho de v para u na árvore passa apenas por e_2, única descendente de v em l. Assim, o link não está no ciclo.

iii) ambas as arestas de l são descendentes de v, e assim $\delta[v] \prec min(\delta[e_1], \delta[e_2]) \prec max(\delta[e_1], \delta[e_2])$. Neste caso, o caminho entre v e u passa por e_1 e por e_2, e todos os demais descendentes de v até u. Assim o ciclo contém ambas as arestas. Uma das arestas pode ser a própria aresta de retorno, pois ela é analisada antes do vértice, e pode colocar l na pilha neste momento.

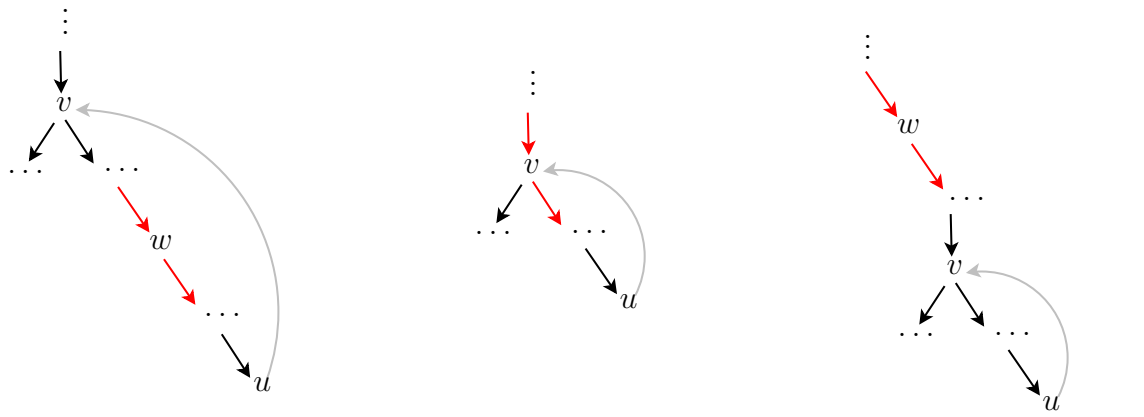

Figura 6.11 Casos onde o link l está na pilha ao chegar em u.

A corretude dos Algoritmos 6.5 e 6.5 decorre do Teorema 6.3 e do Teorema 6.4.

6.5.2 Complexidade

O esqueleto do algoritmo é uma busca em profundidade em um grafo conexo. O custo de tal busca é linear sobre as *arestas* do grafo, pois cada aresta é percorrida uma vez apenas (a idéia de percorrer de volta é representada pela volta da recursão). A única adição foi um acesso direto aos links de expansão e contração, o que pode ser feito em tempo constante com um mapeamento similar ao utilizado para definir os valores de δ e ζ para todo vértice e aresta. Outro ponto é a recuperação de um link a partir de uma aresta volátil. Isto também pode ser executado com custo constante se um mapeamento de acesso direto for utilizado para armazenamento e recuperação de tais links (por exemplo, uma tabela de *hash*).

A pilha é utilizada para simular a recursão que está sendo executada, porém levando em conta os links completos que foram ativados (o que ocorre quando ambas estão no estado descoberto). Com esta pilha, o acesso ao último link que foi completo fica direto, sem a sobrecarga que seria exigida por uma busca entre todos os links.

Apesar do custo ser linear sobre as arestas, o que poderia levar a um algoritmo quadrático sobre os vértices para grafos quaisquer, isto não é um problema para os grafos de prova. Como cada vértice pode ser premissa e conclusão de no máximo um link, e um

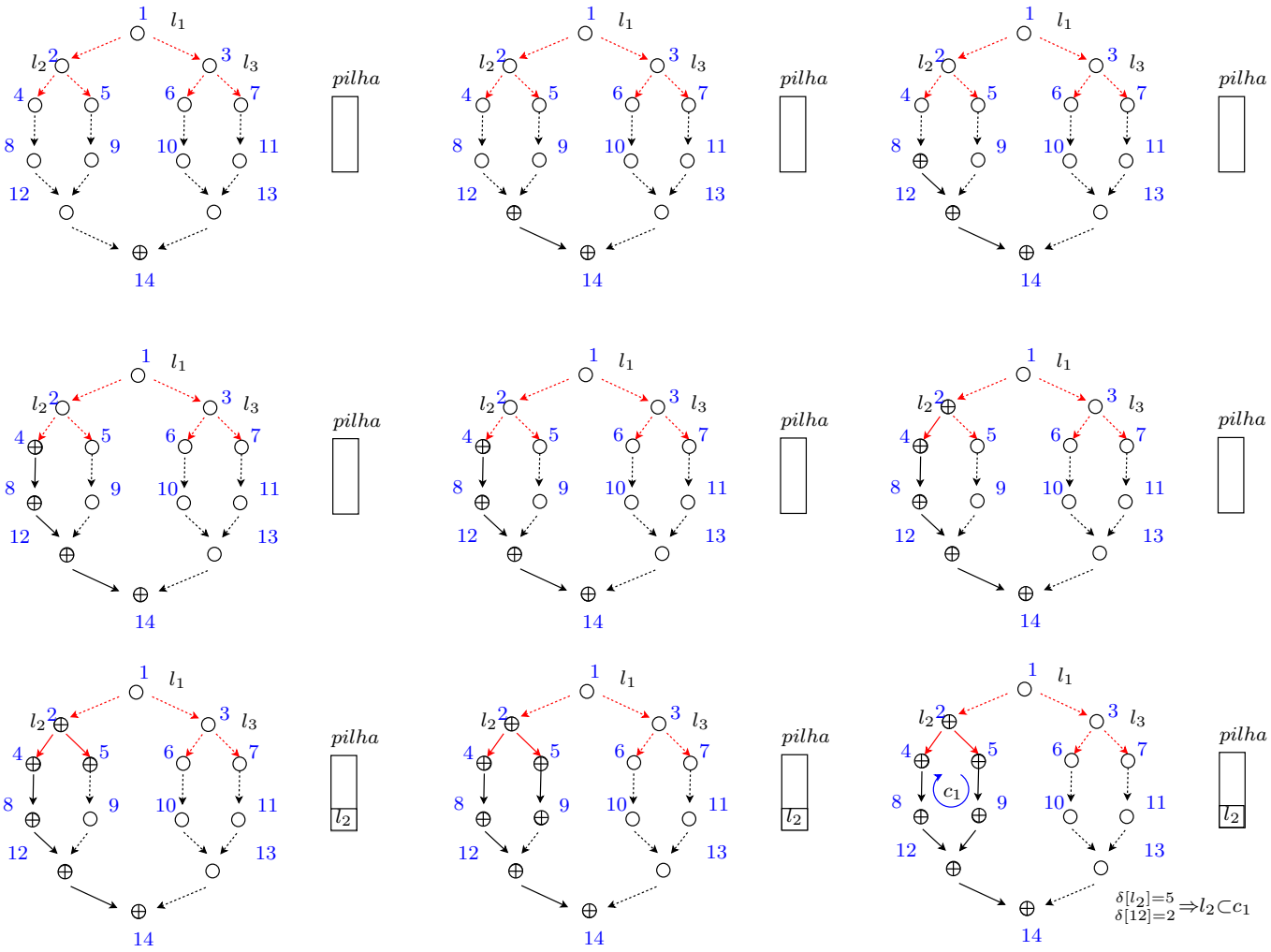

Figura 6.12 Exemplo do algoritmo de busca. As bolas vazias representam vértices não visitados, as bolas com uma cruz representam vértices descobertos e as bolas com um X representam vértices terminados.

vértice em um link tem no máximo duas arestas incidentes a ele, um grafo de prova tem no máximo $4|V|$ arestas, o que é linear com a constante sendo 4.

6.6 EXEMPLOS

A seguir vemos alguns exemplos da aplicação do algoritmo sobre grafos de prova. Como primeiro exemplo, temos o mesmo N-grafo da Figura 6.8. Desta vez, o algoritmo começa da conclusão, e descobre os dois ciclos do grafo. O algoritmo está ilusrado em Figura 6.12 e Figura 6.13.

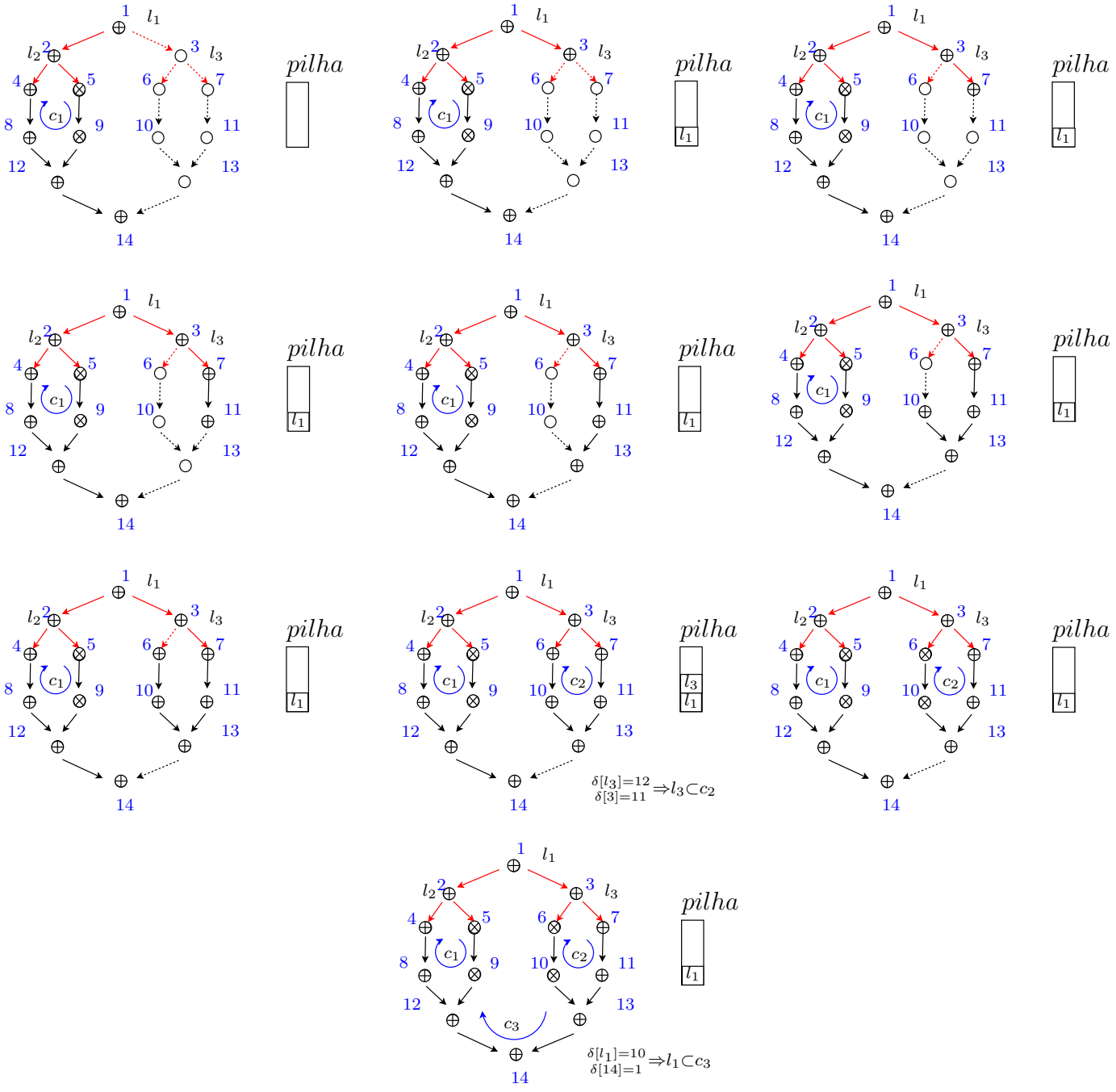

Figura 6.13 Continuação do exemplo do algoritmo de ciclos. As bolas vazias representam vértices não visitados, as bolas com uma cruz representam vértices descobertos e as bolas com um X representam vértices terminados.

No exemplo mostrado na Figura 6.14, vemos um caso de grafo sem prova acíclico, porém inválido pelo Teorema 6.3.

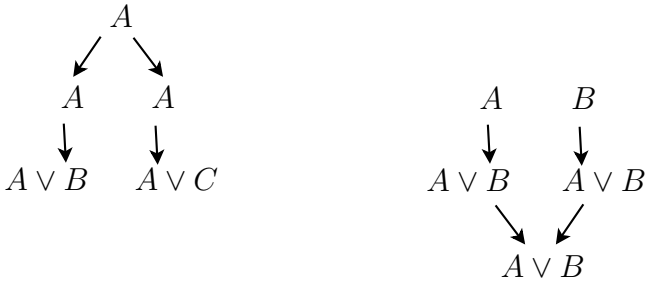

Figura 6.14 Grafos de prova inválidos sem ciclos: A fórmula do Teorema 6.3 não é satisfeita. Temos $4 - 1 \neq 5 - 1$

Outro exemplo mais simples na Figura 6.16 aplica o algoritmo no grafo da Figura 6.15.

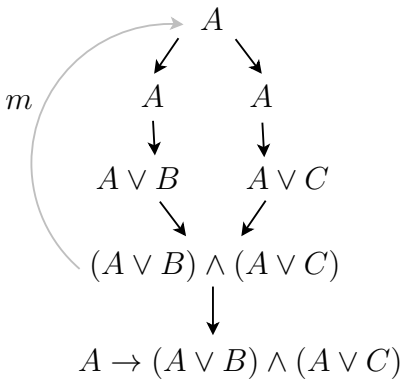

Figura 6.15 Exemplo da prova de $\vdash A \to (A \vee B) \wedge (A \vee C)$

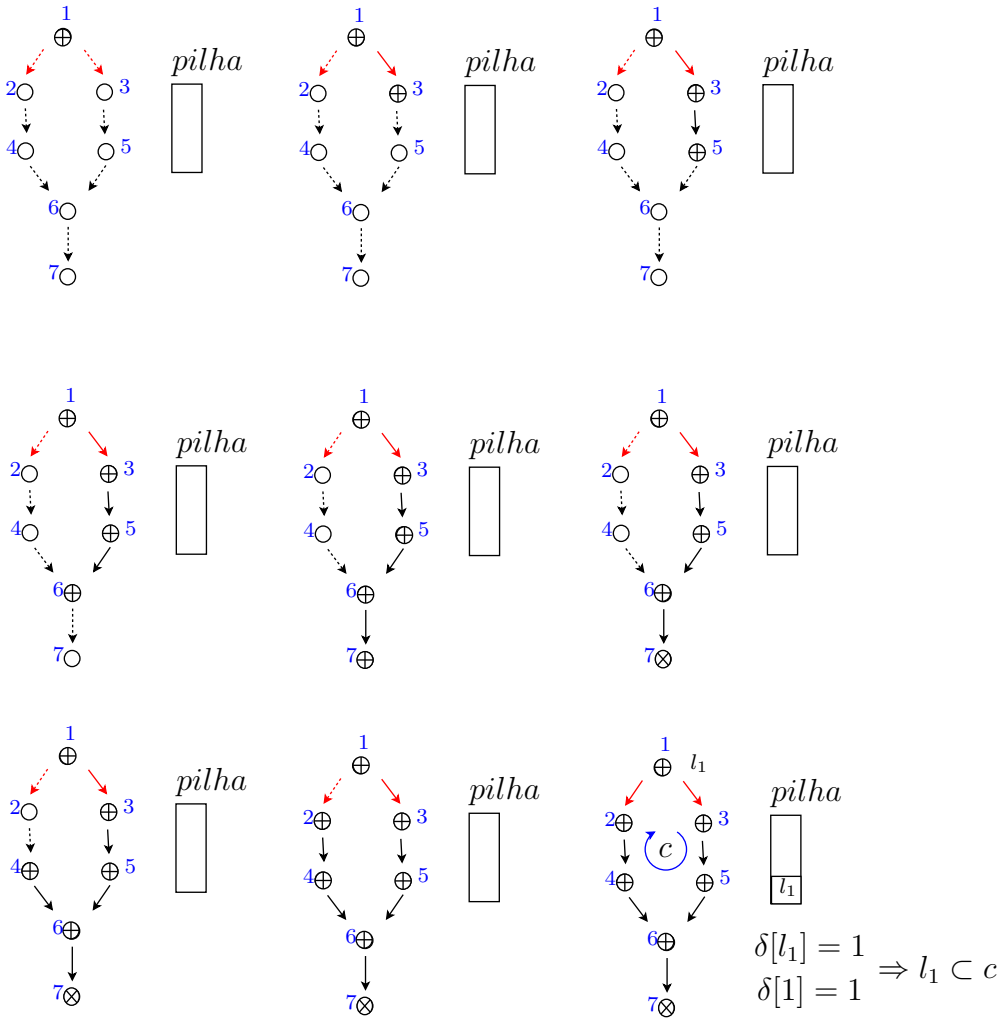

Figura 6.16 Exemplo de aplicação do algoritmo no grafo da Figura 6.15.

CAPÍTULO 7

GERAÇÃO AUTOMÁTICA DE PROVAS

Técnicas para a geração automática de provas é o núcleo de todo assistente de provas. O mecanismo deve ser flexível o suficiente para permitir ao usuário uma manipulação quase que livre do processo de prova, mas restrita no sentido de permitir apenas a construção de provas válidas. A técnica desenvolvida deve ser *correta* e *completa*, de modo que o usuário sempre chegue em um estado terminal do procedimento interativo de dedução.

O procedimento de geração pode englobar um mecanismo de construção de provas totalmente independente de interação com o usuário. Assim, o assistente também pode ser utilizado como *provador automático de teoremas*. Neste capítulo duas abordagens autônomas são apresentadas: a primeira delas é mais voltada para a construção de um provador automático, e a segunda é própria para um assistente interativo, com o resultado sendo uma prova em um formato mais amigável para o usuário final.

7.1 GERAÇÃO POR RESOLUÇÃO

Como visto no Capítulo 2, o princípio da resolução é um dos mais importantes para a definição de técnicas de prova por refutação. Varios algoritmos de poda da árvore de busca foram e continuam sendo definidos para melhorar a performance deste método, e muitos trabalhos sobre esta área já são consolidados na teoria e na lógica.

Uma perspectiva interessante para a geração de N-grafos a partir de uma fórmula é fazer um mapeamento da regra de inferência da resolução para uma operação geométrica sobre grafos de prova, de modo que este processo seja capaz de gerar um N-grafo para $\vdash \neg S$ sse S for insatisfatível.

Na Figura 7.1 é mostrada a regra de inferência aplicada a duas cláusulas [1] C_1 e C_2

[1] Uma cláusula na lógica proposicional é uma sentença formada apenas pela disjunção de literais. Este literal pode ser uma variável proposiconal ou a negação de uma.

sobre o literal A. Esta regra separa os literais complementares A e $\neg A$ das cláusulas, e cancela a ocorrência dos dois por meio de um $\perp - link$. Depois, ela gera a partir de cada cláusula o resolvente, por meio de $\vee - I$. Por fim, estes resolventes são contraídos em uma única ocorrência. Este componente por si é válido, e representa o N-grafo de $C_1, C_2 \vdash C_1' \vee C_2', \perp$, onde C_i' representa a cláusula C_i sem a ocorrência do literal que foi utilizado na resolução. Na Figura 7.2 esta estrutura de resolução é representada de maneira simplificada, e esta notação será utilizada nos demais exemplos.

Nesta seção o termo S será utilizado ora para representar um conjunto de cláusulas, como naturalmente é utlizado na resolução, ora para representar uma fórmula na FNC (*Forma Normal Conjuntiva*), que tem a semântica equivalente à do conjunto S. Sempre que S for tratado como uma fórmula, ao se falar de provas em N-grafos, esta fórmula será a FNC que representa o conjunto S.

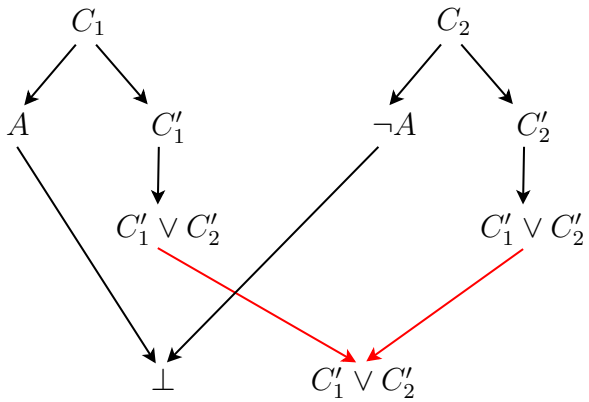

Figura 7.1 Resolução de C_1 e C_2 pelo literal A: C_1' representa C_1 sem A, e C_2' representa C_2 sem $\neg A$.

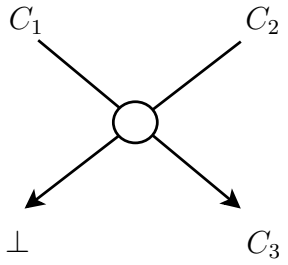

Figura 7.2 Representação de C_1 e C_2 gerando o resolvente C_3.

Quando uma cláusula vazia é deduzida a partir de duas cláusulas, como na figura Fi-

gura 7.3, temos uma estrutura mais simples para representar a resolução. A fim de manter a notação, a representação de tal resolução será como mostrado na figura Figura 7.4. Não existe um vértice para representar a fórmula vazia nos grafos de prova, então o símbolo $\varnothing$ tem apenas um papel ilustrativo, para mostrar que esta resolução chegou à $\varnothing$. A única conclusão deste componente é $\perp$ (o sequente que representa este passo é $C_1, C_2 \vdash \perp$).

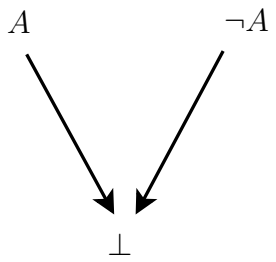

Figura 7.3 Dedução de $\varnothing$ por resolução.

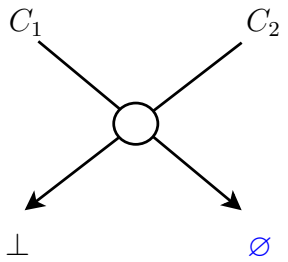

Figura 7.4 Representação da dedução de $\varnothing$.

A representação de todo o procedimento da resolução é montada a partir destes blocos, que são ligados uns aos outros cada vez que a regra de inferência é aplicada a duas cláusulas. Esta ligação é representada no exemplo da Figura 7.5. Quando a cláusula vazia não puder ser deduzida a partir do conjunto inicial S, temos um conjunto de fórmulas que é satisfatível. Por ser uma prova por refutação, a resolução não é capaz de provar que este conjunto nem a sua negação são uma tautologia.

Com isto, os N-grafos que representam provas da resolução serão do tipo $\vdash \neg S$, dado um conjunto S de proposições. Para provar tal afirmação, define-se a partir do vértice S um fluxo lógico que deve convergir em ocorrências de $\perp$. Uma vez que a cláusula vazia foi deduzida, as demais cláusulas que não foram utilizadas na inferência de $\varnothing$ (irrelevantes e redundantes na prova) são eliminadas do conjunto de conclusões da prova por outros $\perp$

gerados, utilizando o link $\wedge - I$ [2]. A Figura 7.6 mostra como isto é feito.

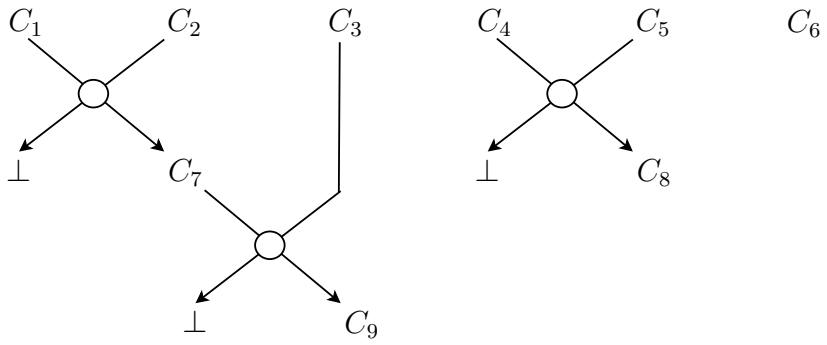

Figura 7.5 Exemplo da aplicação da resolução.

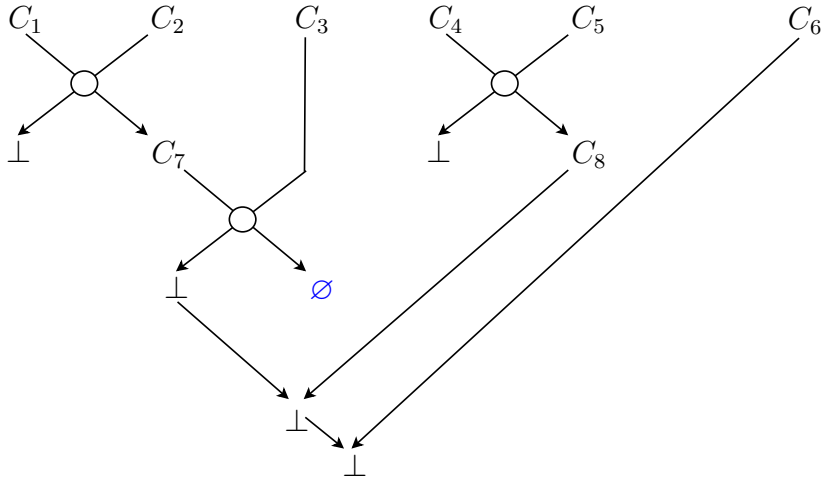

Figura 7.6 Exemplo de cancelamento de conclusões.

As premissas do processo de resolução, que constituem das cláusulas de S, precisam ser conectadas na prova. Por meio da fórmula de S é possível fazer tal ligação, e esta fórmula deve depois ser ligada à $\neg S$, que representa o objetivo da prova (conclusão). Para isso, S deve gerar cada uma das cláusulas iniciais por meio de $\wedge - E$ e *expansões*. O processo é ilustrado na Figura 7.7.

Quando uma cláusula é necessária em mais de uma resolução, um vértice que representa uma ocorrência dela pode ser duplicado pelo uso da *extensão*. Podemos então

[2] Aqui outro abuso de notação é utilizado. A maneira correta de eliminar uma cláusula seria aplicando $\wedge - I$ com um $\bot$, seguido de um $\wedge - E$ que de fato elimina a cláusula. Para simplificar os gráficos, será considerado que $C_i \wedge \bot \equiv \bot$.

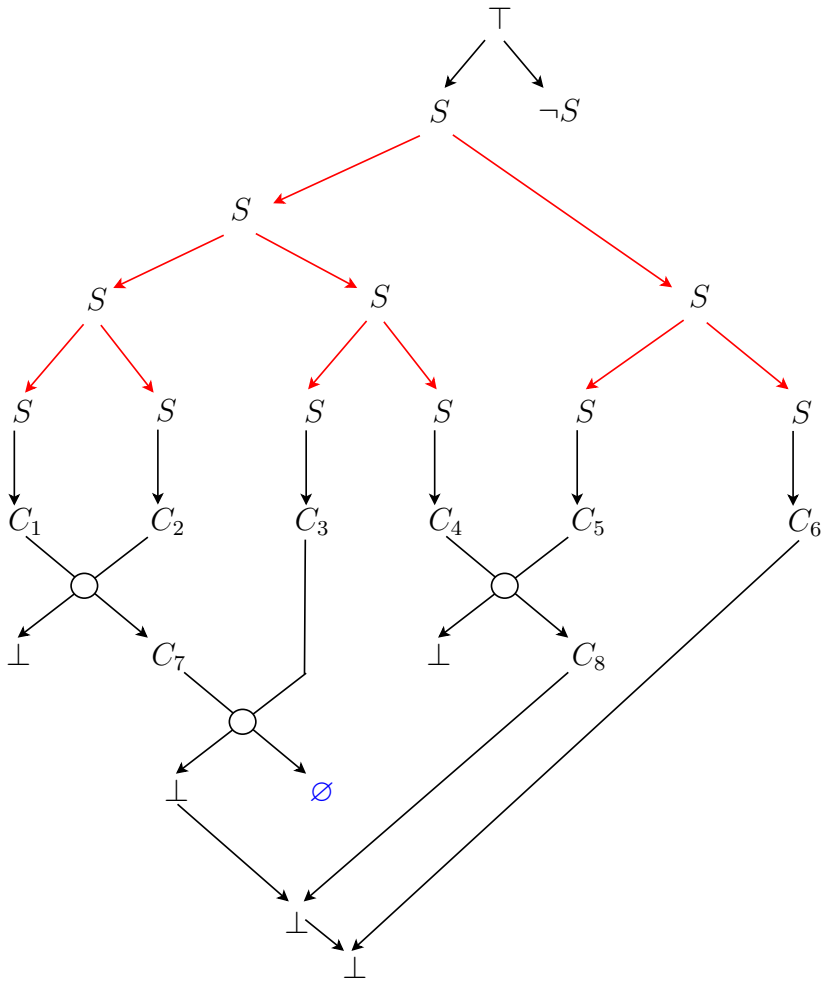

Figura 7.7 N-grafo para $\neg S$, onde $\neg S = \neg(C_1 \wedge C_2 \wedge \cdots \wedge C_6)$

resumir o processo de construção do N-grafo da resolução.

Definição 7.1 (grafo de resolução). O *grafo de resolução* de um conjunto de cláusulas S é um grafo de prova construído da seguinte maneira:

1) Etapa de resolução
   a) Crie um vértice para cada cláusula $C_i \in S$;
   b) Aplique sucessivamente a regra da resolução, adicionando as ligações do componente da Figura 7.1 às duas cláusulas a serem resolvidas, para gerar novos resolventes;
   c) Sempre que uma cláusula C_i precisar ser utilizada em mais de uma regra de dedução, adicione links de extensão em C_i e ligue cada uma das cópias geradas

a uma aplicação da regra de inferência;

d) Quando $\varnothing$ for deduzida, ou quando novas cláusulas não puderem ser mais geradas, termine;

2) Etapa de cancelamento de cláusulas irrelevantes

a) Para cada cláusula irrelevante (não utilizada em nenhuma resolução), aplique o link $\wedge - I$ nela e no $\bot$ do componente que gerou a cláusula vazia (Figura 7.3) sucessivamente, até que exista apenas um conjunto de resolução (Definição 7.2).

3) Etapa de eliminação de premissas

a) Crie um vértice para S, e por meio de expansões gere o mesmo número de ocorrências de S quanto o número de cláusulas iniciais;

b) Ligue cada ocorrência de S à cláusula C_i por meio de um $\wedge - E$;

4) Etapa de finalização

a) Cancele a premissa S adicionando um $\top - link$, e adicionando $\neg S$ à lista de conclusões.

Definição 7.2 (**conjunto de resolução).** Para cada vértice v do grafo de resolução G que não é premissa de nenhum link, com exceção do $\bot$, defina como seu *conjunto de resolução* o conjunto com todos os vértices de G que estão ligados à v sem passar por arestas de expansão.

Definição 7.3 ($\bot_{\varnothing}$)**.** No componente de resolução que gerar a cláusula vazia, o seu único vértice conclusão $\bot$ é denominado $\bot_{\varnothing}$.

Os estados terminais deste processo de construção da prova são os seguintes:

$$\vdash \neg S$$

$$ou$$

$$\vdash \neg S, C_1, \ldots, C_n$$

No primeiro caso temos uma prova de que S é insatisfatível, e ele ocorre quando na resolução chegamos à cláusula vazia. O segundo acontece quando não é possível deduzir $\varnothing$ a partir de S. Neste caso, S é satisfatível, e o grafo de resolução pode não ser válido, pois podem existir cláusulas que são serão conectadas pela Etapa de cancelamento de cláusulas irrelevantes.

Para que este processo seja válido, é necessário provar que os grafos gerados são N-grafos quando S é insatisfatível.

Lema 7.1. *Cada conjunto de resolução tem exatamente uma cláusula como conclusão que não é* $\perp$.

Demonstração. Prova por indução sobre a aplicação da regra de redução sobre duas cláusulas C_1 e C_2:

Inicialmente, cada cláusula é um conjunto de resolução.

Ao resolver duas cláusulas C_1 e C_2, duas conclusões são transformadas em um único resolvente C_3, desconsiderando-se o $\perp$ gerado.

Assumindo como hipótese de indução que todos os conjuntos de resolução possuem exatamente uma cláusula diferente de $\perp$, fica claro que resolver duas cláusulas significa juntar dois conjuntos de resolução disjuntos em um novo conjunto, que tem como única conclusão diferente de $\perp$ a fórmula C_3. Temos duas possibilidades para C_3: ou ele é uma fórmula não vazia, ou é igual a $\varnothing$. No primeiro caso, fica provado que o passo indutivo gerou um novo conjunto de resolução com exatamente uma cláusula que não é $\perp$. Já no segundo, vemos que o passo indutivo gerou um novo conjunto que não é um conjunto de resolução, pela sua definição (nenhum vértice deste novo componente se liga mais a conclusões irrelevantes). Como neste caso apenas removemos dois conjuntos de resolução, os demais continuam obedecendo ao Lema 7.1.

Assim fica provado por indução o Lema 7.1. ∎

Lema 7.2 (**aciclicidade do grafo de resolução**). *Todo ciclo no grafo de prova G gerado pelo procedimento da Definição 7.1 é válido.*

Demonstração. Todo ciclo deve conter pelo menos um link divergente e um convergente [Alv09]. Como visto no Capítulo 6, um ciclo é válido quando ele contém pelo menos um link de expansão ou contração. No processo de construção definido em Definição 7.1, os únicos links divergentes utilizados são $\top - link$, $\vee - E$ e *links de expansão*. Os convergentes são $\wedge - I$, $\perp - link$ e *links de contração*.

O $\top - link$ não pode gerar nenhum ciclo, pois um das suas conclusões é $\neg S$, que não se liga a mais nenhuma fórmula. Assim, para gerar um ciclo inválido um $\vee - E$ deve ser o único link divergente utilizado. Todo link $\vee - E$ está dentro de um componente de resolução. Os únicos vértices que se ligam com outros fora deste componente são C_1, C_2 e C_3, e por isso são a única possibilidade de se fechar um ciclo evitando-se o caminho pela contração. Pela estrutura destes componentes, um ciclo que o contenha e que não passe pela contração deve ter a estrutura $C_1, A, \bot, \neg A, C_2, \ldots, C_3, C_3, C_1', C_1$, ilustrada na Figura 7.8.

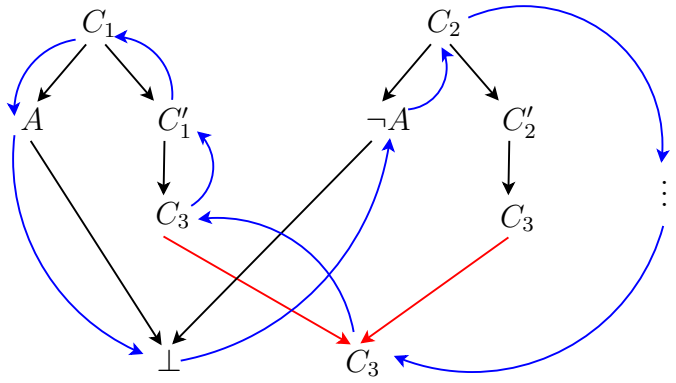

Figura 7.8 Ciclo envolvendo o link $\vee - E$.

No final da Etapa de resolução, as cláusulas C_2 e C_3 estão no mesmo conjunto de resolução, ou estão ambos ligados à $\bot_\varnothing$. Em ambos os casos, eles não podem se ligar passando por um $\wedge - I$, uma vez que eles são introduzidos na Etapa de cancelamento de cláusulas irrelevantes ligando conjuntos de resolução *disjuntos* ao componente com o $\bot_\varnothing$. Assim, para gerar o ciclo inválido, o mesmo deve ser fechado por um $\bot - link$.

Pelo Lema 7.1, sempre que se "entra" em um conjunto de resolução pela conclusão diferente de $\bot$, deve-se "sair" por uma das premissas. Assim, como C_2 deve ser a conclusão de um outro componente ou uma cláusula de S, sempre que caminhamos a partir de C_2 para fora do componente em questão e entramos em um outro componente, voltamos ao caso onde teremos de sair de uma das premissas de um componente. Os outros casos seriam:

1) chegar a uma cláusula inicial de S. A partir daí o caminho deve ou "subir" na estrutura e passar por um link de expansão para voltar por outro caminho e se encontrar com C_3 (caso ilustrado na Figura 7.9);

2) chegar a uma conclusão que foi expandida. Neste ponto ele pode escolher percorrer parciamente as arestas de expansão, e entrar em outro componente pela conclusão (o que o força a sair dele como premissa, e o leva novamente a uma destes casos finais), ou percorrer um destes links de expansão completamente. Este caso é ilustrado na Figura 7.10.

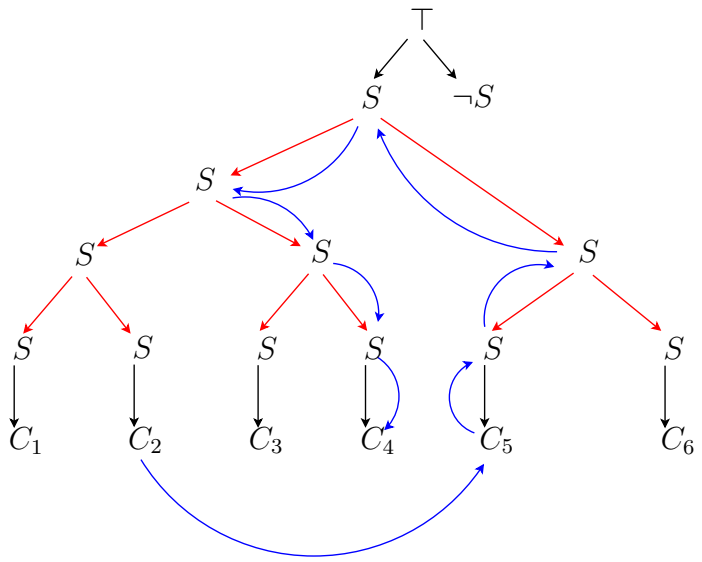

Figura 7.9 Caso onde o caminho a partir de C_2 passa por uma ocorrência de S.

Em todos estes casos, de alguma maneira um link de contração ou expansão será percorrido, e o ciclo inválido não será gerado. ∎

Lema 7.3 (**conectividade do grafo de resolução**). *Todo grafo de chaveamento associado a um grafo de resolução de um conjunto S insatisfatível é conexo.*

Demonstração. Analisando o grafo de resolução G por etapa de construção podemos mostrar que todo chaveamento associado a G é conexo.

As únicas arestas voláteis que existem nos grafos de resolução são as pertencentes a links de contração e de expansão. Todo link de contração é inserido junto com os componentes de resolução, dentro de um ciclo simples (como visto na Figura 7.1). Ao remover uma das arestas deste link, o componente continua sendo conexo. Utilizando o mesmo como uma caixa preta, o comportamento que se observa quando à conectividade é o mesmo: as quatro pontas C_1, C_2, C_3 e $\perp$ continuam ligadas, conectando possivelmente quarto outros componentes em todo chaveamento. Isto significa que os links de contração, por causa da estrutura especial na qual eles sempre se encontram, não afetam

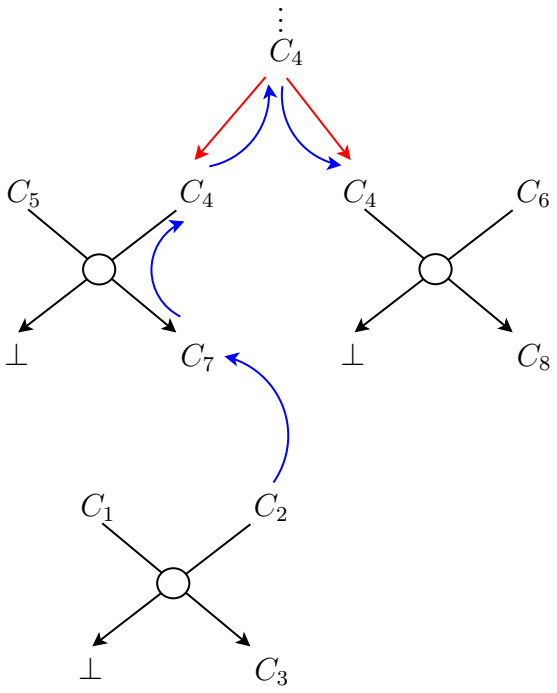

Figura 7.10 Caso onde o caminho a partir de C_2 por uma expansão de uma cláusula D_4.

a conectividade do grafo de resolução.

Outra maneira de observar o caso do link de contração é considerando o componente de resolução como um suporte a este link. Para todo caminho que passe por uma aresta deste link, existe um simétrico que passa pela outra, que é o complemento do caminho escolhido no ciclo. Sempre existem dois caminhos, um dando suporte ao outro em caso dele não ocorrer em um chaveamento.

Já os links de expansão podem aparecer em dois contextos. O primeiro deles é na derivação de todas as cláusulas iniciais a partir da fórmula de S. A estrutura montada cria uma árvore, onde S é a raiz e cada uma das cláusulas $\{C_i \mid C_i \in S\}$ é uma folha. Nesta estrutura, cada chaveamento associado ao grafo tem exatamente um caminho de S até uma cláusula C_i, dado que cada link deve ter uma de suas arestas no chaveamento. Logo S, juntamente com o $\top - link$ que o gera, estão ligados sempre a uma das cláusulas iniciais. Considerando que o componente gerado pelas etapas de resolução e cancelamento de cláusulas irrelevantes é conexo quando S é insatisfatível (ver na Figura 7.7), o grafo total é conexo apensar destas expansões.

Para mostrar que o componente gerado pelas duas etapas iniciais é conexo se S é insatisfatível, é preciso avaliar o segundo caso onde links de expansão podem ocorrer: em uma cláusula C_i que seja inicial ou resolvente. De maneira genéria, sejam dois vértices desta parte do grafo de resolução. Uma das seguintes afirmações é verdade:

1) Ambos estão ligados por um caminho sem arestas de expansão. Neste caso, eles fazem parte de um mesmo conjunto de resolução, ou estão ligados à $\perp_\varnothing$. Ambas estas opções definem componentes que estão ligados em todo chaveamento, pois as únicas arestas voláteis envolvidas são as da contração, que como visto antes não são capazes de desconectar o grafo onde ela se insere. Assim, eles estão conectados em todo chaveamento associado a G.
2) Ambos estão ligados de modo que não existe caminho entre eles sem passar por arestas de expansão. Logo eles pertencem a conjuntos de resolução disjuntos. Durante a etapa de cancelamento de cláusulas irrelevantes, estes conjuntos são ligados de maneira não-volátil ao conjunto que contém $\perp_\varnothing$, de modo que no final não existam mais conjuntos de resolução, pois todos os vértices serão ligados por um caminho que não passa por arestas de expansão (caindo no item 1).

∎

A partir destes lemas, pode-se provar a corretude dos grafos de resolução.

Teorema 7.1 (**validade do grafo de resolução).** *Todo grafo de resolução de um conjunto S insatisfatível é válido.*

Demonstração. Como o grafo de resolução não possui meta-arestas, a meta-condição já é satisfeita por vacuidade. É necessário provar então que todos os grafos de chaveamentos associados a S são *acíclicos* e *conexos*. Pelo Lema 7.2, todos os ciclos que existem nestes grafos são válidos. Isto significa que todo grafo de chaveamento associado com um grafo de resolução é acíclico. Pelo Lema 7.3, todo chaveamento associado ao mesmo grafo é conexo quando S é insatisfatível. Logo, podemos concluir que um grafo de resolução de um conjunto insatisfatível é um N-grafo. ∎

7.1.1 Considerações sobre o assistente

Apesar da resolução ser uma técnica utilizada para geração automática, esta conversão para grafos de prova pode tornar mais intuitivo e visual o processo de geração de cláusulas. Para fins didáticos, um assitente interativo poderia ser definido a partir desta transformação.

Uma aplicação mais interessante seria o estudo dos aspectos geométricos do procedimento de resolução. Que tipo de estrutura ele forma, que otimizações sobre a estrutura de grafo que podem ser geradas, entre outros aspectos que poderiam ser analisadas sobre o mapeamento com os N-grafos. Outra análise pode ser uma investigação sobre os algoritmos que existem hoje em dia para refinar a resolução, e como eles poderiam ser convetidos para grafos de prova e até mesmo melhorados neste ambiente, onde alguma propriedade geométrica da resolução que antes era ignorada pode se mostrar interessante.

Outro ponto relevante para a implementação de um assistente seria deixar este processo de geração do N-grafo como sendo uma etapa apenas da geração da prova, inerente ao núcleo de processamento. Uma segunda etapa poderia atuar sobre este N-grafo, definindo uma transformação que preserve a corretude do mesmo e o deixe mais amigável ao usuário final.

7.1.2 Exemplos

Aqui um exemplo de grafo de prova gerado por resolução é mostrado na Figura 7.11. O conjunto $S = \{(P \vee Q), (P \vee \neg Q), (\neg P \vee Q), (\neg P \vee \neg Q)\}$ será utilizado.

7.2 GERAÇÃO POR DEDUÇÃO NATURAL

A abordagem definida na seção anterior se baseia em um método de prova por contradição. Na verdade, o que se constrói é uma prova da negação de S, ou seja, uma prova indireta sobre o mesmo. Uma abordagem mais interessante para o usuário final seria gerar uma prova de maneira construtiva, partindo de um conjunto de premissas para chegar em um conjunto de conclusões.

A maneira ideal de se trabalhar com este tipo de prova aqui é utilizando a dedução natual como sistema de dedução automático. Um motivo para tal abordagem vem do fato deste sistema ter sido criado por Gentzen por ser mais próxima da maneira natural do ser humano de desenvolver um raciocínio lógico. Outro motivo é por ser também a base de toda a geometrização que foi definida nos N-grafos. Assim, uma dedução natural não só é mais intuitiva, como explora os links dos N-grafos pela semântica que foi aderida aos mesmos.

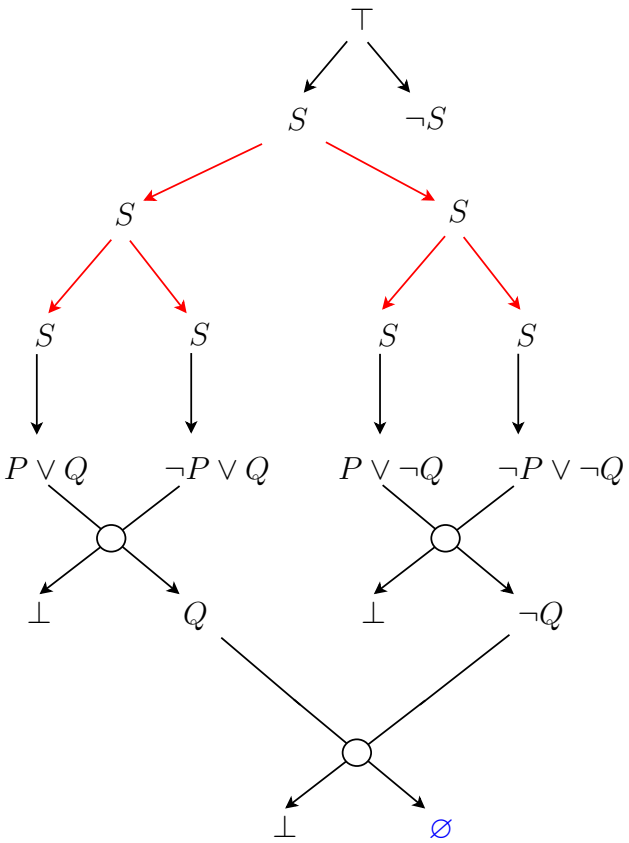

Figura 7.11 N-grafo para $\neg S = \neg(P \vee Q) \wedge (P \vee \neg Q) \wedge (\neg P \vee Q) \wedge (\neg P \vee \neg Q)$

Sieg e Scheines em [SS92] definem uma técnica para a busca automática de provas na dedução natural. Ela se baseia em um cálculo que foi desenvolvido justamente para este propósito, o *cálculo de interpolação*, e que foi provado ser completo e correto neste trabalho. Nesta seção é proposta uma metodologia para a transformação de uma derivação neste cálculo em um grafo de prova, extendendo a técnica que foi desenvolvida para um cálculo de única conclusão para se tornar aplicável em um cálculo de múltiplas conclusões como os N-grafos.

7.2.1 Cálculo de Interpolação

O cálculo de interpolação, como definido em [SS92] e [SB98], tem como objetivo preencher o vazio (do inglês, *gap*) que existe entre um conjunto de premissas e uma conclusão. A pergunta que ele tenta responder é: Como podemos derivar uma conclusão φ a partir das

premissas $\phi_1, \ldots, \phi_n$? A notação utilizada para representar tal pergunta é

$$\phi_1, \ldots, \phi_n ? \varphi$$

onde o símbolo ? deve, a partir da aplicação de algumas *regras inversas* de introdução e eliminação, ser preenchido por fórmulas intermediárias.

As regras de preenchimento do vazio são definidas como inversos das regras de derivação da dedução natural. São chamadas de regras-$\uparrow$ as regras inversas às de *introdução*, e de regras-$\downarrow$ as inversas às regras de *eliminação*. Um exemplo simples é a aplicação da técnica para resolver $P \wedge Q \vdash Q \wedge P$:

$$\begin{array}{c} P \wedge Q \\ ? \\ Q \wedge P \end{array}$$

Aplicando a inversa da $\wedge - I$, temos:

$$\dfrac{\begin{array}{cc} P \wedge Q & P \wedge Q \\ ? & ? \\ Q & P \end{array}}{Q \wedge P}$$

Que pode ser resolvido com a aplicação da inversa da $\wedge - E$ nos dois lados:

$$\dfrac{\begin{array}{cc} \dfrac{P \wedge Q}{Q} & \dfrac{P \wedge Q}{P} \\ ? & ? \\ Q & P \end{array}}{Q \wedge P}$$

E assim o vazio foi preenchido. A tabela Tabela 7.1 mostra todas as regras inversas na notação ?. Nela, α e β representam sequências de fórmulas e a concatenação $\alpha\beta$ indica a juntaposição destas sequências.

Estas regras definem um espaço de busca de fórmulas intermediárias, que cresce como uma árvore binária. Na Figura 7.12 esta árvore de busca é mostrada. As ramificação representadas por um quadrado indicam uma disjunção (apenas um dos ramos deve chegar à uma prova), e aquelas representadas por um quadrado com um ponto no centro indicam uma conjunção (os dois ramos devem chegar na prova).

As regras que utilizam o termo *OU* indicam que apenas um dos vazios gerados deve

$\downarrow \wedge : \alpha;\beta?G$	$\varphi_1 \wedge \varphi_2 \in \alpha\beta$	$\Rightarrow$	$\alpha;\beta;\varphi_1?G \; OU \; \alpha;\beta;\varphi_2?G$
$\downarrow \vee : \alpha;\beta?G$	$\varphi_1 \vee \varphi_2 \in \alpha\beta$	$\Rightarrow$	$\alpha;\beta;\varphi_1?G \; E \; \alpha;\beta;\varphi_2?G$
$\downarrow \rightarrow : \alpha;\beta?G$	$\varphi_1 \rightarrow \varphi_2 \in \alpha\beta, \varphi_1 \in \alpha\beta$	$\Rightarrow$	$\alpha;\beta;\varphi_2?G$
$\uparrow \wedge : \alpha;\beta?\phi_1 \wedge \phi_2$		$\Rightarrow$	$\alpha;\beta?\phi_1 \; E \; \alpha;\beta?\phi_2$
$\uparrow \vee : \alpha;\beta?\phi_1 \vee \phi_2$		$\Rightarrow$	$\alpha;\beta;?\phi_1 \; OU \; \alpha;\beta?\phi_2$
$\uparrow \rightarrow : \alpha;\beta?\phi_1 \rightarrow \phi_2$		$\Rightarrow$	$\alpha;\beta;\phi_1?\phi_2$
$\bot_C : \alpha;\beta?\phi$	$\phi \neq \bot$	$\Rightarrow$	$\alpha;\beta;\neg\phi_1?\bot$
$\bot_I : \alpha;\beta?\neg\phi$		$\Rightarrow$	$\alpha;\beta;\phi?\bot$
$\bot_F : \alpha;\beta?\bot$	$\phi \in f\acute{o}rmulas(\alpha\beta)$	$\Rightarrow$	$\alpha;\beta?\phi \; E \; \alpha;\beta?\neg\phi$

Tabela 7.1 Derivaçoes do cálculo de intercalação

ser preenchido. Isso ocorre para a regra de eliminação do $\wedge$ da dedução natural, onde há uma escolha a ser feita sobre qual fórmula será descartada, e para a introdução do $\vee$, que também requer uma escolha de que fórmula servirá de base. As regras que utilizam E também dividem a busca em duas partes, mas para ser válido ambos os vazios devem ser preenchidos. Isso acontece com as regras conjuntivas que têm mais de uma premissa $\vee - E$ e $\wedge - I$. As regras da negação também incluem a *lei do terceiro excluído*, pois trabalha com a lógica clássica.

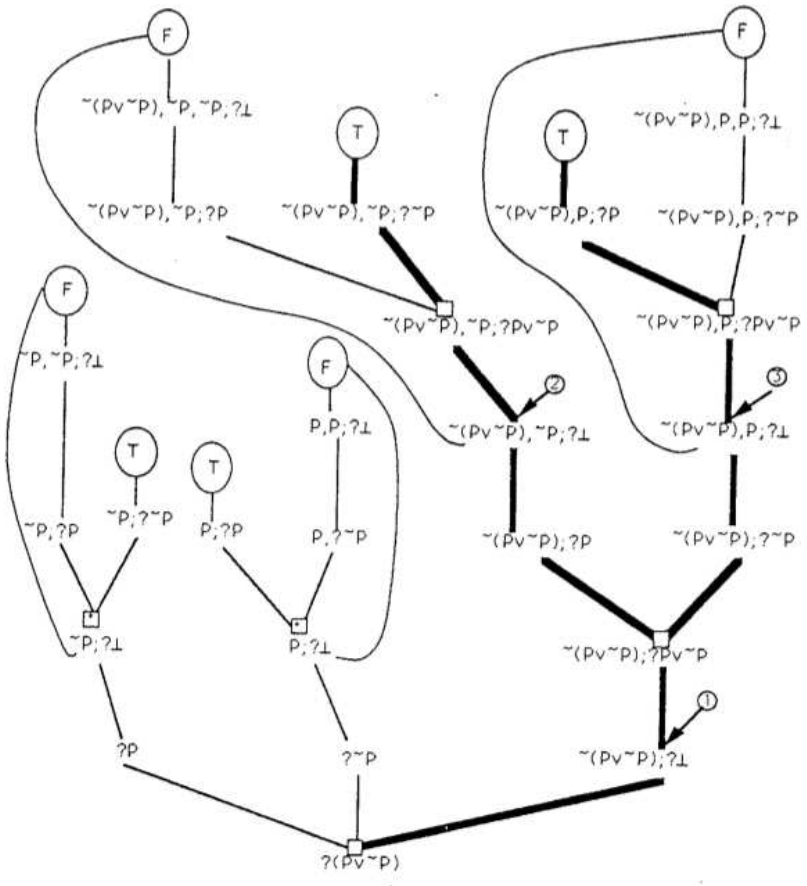

Figura 7.12 Árvore de busca da prova, tirada de [SS92]. Aqui, a notação p é utilizada para representar $\neg p$.

O *teorema da completude* para o cálculo de interpolação foi provado em [SS92]. Nele, Sieg afirma que:

- O procedimento de decisão definido para a lógica proposicional pelo o cálculo é finito, uma vez que a árvore de busca é finita;

- Existem maneiras canônicas para se extrair derivações em várias formas da dedução natural a partir de uma árvore de busca calculada;

- As derivações extraídas estão na forma normal e satisfazem a propriedade da sub-fórmula [3].

7.2.2 Extensão para os N-grafos

Cada uma destas regras tem uma relação íntima com cada uma das regras de derivação da dedução natural. Utilizando o mesmo cálculo de interpolação, porém com um mapeamento de cada regra do cálculo com uma regra dos N-grafos, podemos converter uma derivação de interpolação em um grafo de prova válido.

Neste mapeamento para o caso da $\rightarrow$ será ignorado, por causa das peculiaridades que uma prova da validade da meta-condição exige. Nos casos onde temos um *OU* na Tabela 7.1, apenas um dos lados precisa ser derivado. Pode tamém ser feita uma geração das duas ramificações no grafo de prova, um para cada linha de desenvolvimento, e depois ambos são juntos novamente. Esta junção deve ser feita de modo que um ramo deve ser enfraquecido e "absorvido" belo outro. Isto está ilustrado pelo componente em azul na Figura 7.1.

Nos casos onde temos um E, é necessário construir os dois lados da prova. Para isso um link conjuntivo $\wedge - I$ deve ser utilizado para o $\uparrow \wedge$). No caso do $\downarrow \vee$ temos um caso onde a regra é diferente nos grafos de prova: por ser um cálculo de múltiplas conclusões, os N-grafos geram duas conclusões a partir da eliminação do *ou*. Aqui, a estratégia é gerar as duas fórmulas do *ou* com o link $\vee - E$, e no final contrair as duas conclusões, que devem ser ocorrências da mesma fórmula. A Figura 7.13 ilustra como fazer esta conversão.

Para gerar provas com múltiplas conclusões, é necessário extender o esquema apresentado.

Definição 7.4 (grafo de interpolação). Um *grafo de interpolação* de uma proposição $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ é um grafo de prova construído da seguinte forma:

i) Construa uma prova de $A_1, \ldots, A_n \vdash B_1 \vee \cdots \vee B_m$ pelo cálculo de interpolaçao;

[3] A propriedade da sub-fórmula afirma que todas as fórmulas que aparecem na prova são subfórmulas daquelas que estão na conclusão.

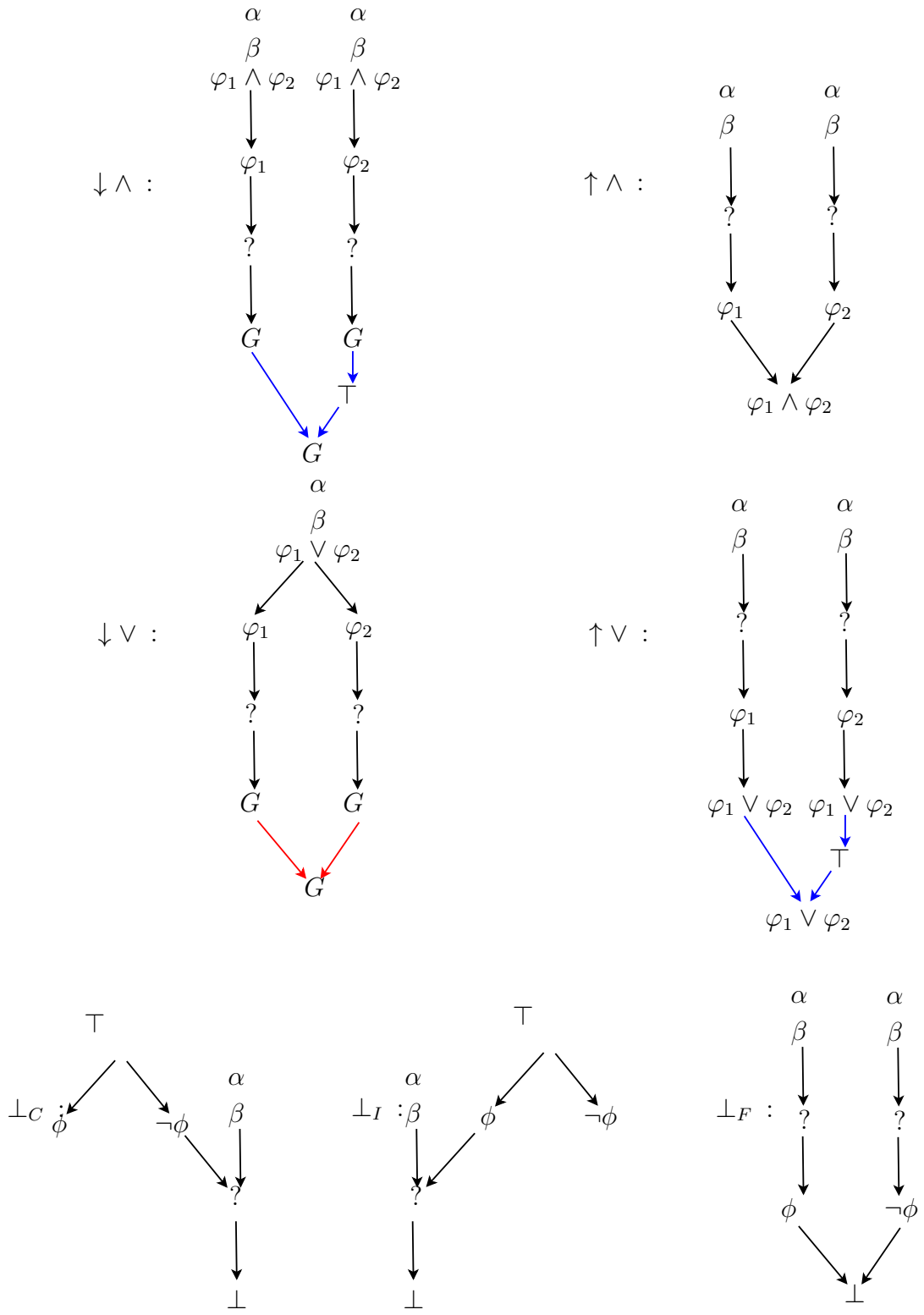

Figura 7.13 Mapeamento das regras de inferência do cálculo de interpolação para os N-grafos.

ii) Para a prova encontrada no passo anterior, um grafo de prova válido é definido utilizando o mapeamento definido na Figura 7.13;
iii) Se a prova gerada não utilizar todas as premissas, a regra de enfraquecimento à esquerda deverá ser aplicada para gerar as premissas que não foram necessárias na prova, da maneira proposta por de Oliveira em [dO01], na prova de completude dos N-grafos;
iv) A conclusão deve ser quebrada utilizando a regra $\vee - E$, de modo a gerar uma prova de $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$;
v) Todas as premissas que ocorrerem mais de uma vez devem ser agrupadas na mesma classe de equivalência pelo uso de links de *expansão*.

Teorema 7.2 (**validade do grafo de interpolação).** *Todo grafo de interpolação é válido.*

Demonstração. Todos os mapeamentos são similares aos da dedução natural, que vêm de maneira canônica com o cálculo de interpolação. A única regra diferente é a da eliminação do $\vee$, que gera um ciclo. Como toda a estrutura da construção da prova é feita em formato de àrvore (projetada assim para desenvolver uma dedução natural, que é em forma de àrvore), estes ciclos são os únicos que irão existir. Como existe um link de contração contido neste ciclo, ele é válido. Além disso, como esse é o único lugar onde pode existir arestas voláteis no grafo gerado, e estes links estão contidos em ciclos, o grafo final terá todos os grafos de chaveamento associados conexos.

Dado que os passos i e ii geram uma prova válida com conclusão apenas, basta provarmos que os passos $iii - v$ preservam a corretude do N-grafo. Caso a prova não possa ser gerada utilizando cálculo de interpolação, pela completude deste cálculo podemos dizer que a afirmação não é válida pelo conta-exemplo gerado.

Considere que um N-grafo G foi construído pelos passos i e ii. O passo iii aplicado em G preserva a sua corretude, como já foi provado por de Oliveira em sua tese. O passo iv também preserva a corretude trivialmente, uma vez que não adiciona ciclos na prova nem arestas voláteis. O único paso que pode gerar problemas é o v, uma vez que adiciona arestas voláteis. Todo ciclo que possa ser fechado pelos links de expansão inseridos é válido, uma vez que deve conter uma das expansões. Além disso, todo grafo de chaveamento associado o grafo final G' é conexo, pois cada chaveamento vai conter exatamente um caminho de cada vértice adicionado até uma das premissas agrupadas por este nó. Isto pode ser observado pela estrutura de árvore binária criada por estas expansões, e pelo fato de uma aresta de cada link permanecer, o que garante a cada nó

interno da árvore a conexão com um de seus filhos. Uma vez que G é um N-grafo, o que indica que o componente ao qual os links de expansão foram acoplados tinha todo chaveamento conexo, o componente final tem também todo chaveamento conexo.

∎

7.2.3 Considerações sobre o assistente

Um assistente de prova, ao utilizar este tipo de geração de grafos de prova, fica muito mais livre para definir o grau de interatividade que o sistema terá. Este procedimento de geração pode ser totalmente autônomo, e neste cenário pode também ser otimizado para gerar uma prova percorrendo o mínimo do espaço de busca e abusando da regra do enfraquecimento. Por outro lado, ele pode colocar poder total sobre o processo de prova na mão do usuário, onde o mesmo possa escolher que regra ele prefere aplicar em uma determinada hora, que ramo da derivação ele prefere seguir na busca e que ramos ele prefere ignorar.

O usuário pode até ter o poder para escolher como prefere atacar as fórmulas, se a partir das premissas ele pretente chegar na conclusão, ou se ele prefere ao contrário. Ele pode até percorrer o processo de geração da prova de trás para frente, aplicando e depois removendo as regras de dedução do cálculo de interpolação para preencher ou aumentar os vazios da prova partial.

Uma das finalidades do tutor gerado por Sieg no projeto era atuar como uma plataforma de aprendizagem para raciocínio lógico e dedução natural, onde os alunos pudessem tentar encontrar provas para sentenças com o auxílio do sistema, que mostrava alguns caminhos possíveis e gerava dicas durante o desenvolvimento [SS92]. Esta finalidade didátia pode estar presente também em um assistente para os N-grafos.

Para definir um sistema automático de prova de teoremas, o uso da dedução natural não seria a técnica recomendada se o foco for eficiência. Aqui o foco seria em encontrar provas mais próximas daquelas que um ser humano definiria para uma dada proposoção da lógica clássica. Nos pontos onde a busca pela prova se abre em dois ramos disjuntivos, temos uma explosão exponencial no espaço de busca da prova. Uma maneira de se sobrepor a tal deficiência seria utilizar as demais regras do cálculo para gerar os passos diretos (que não criam ramificações na árvore de busca) e convergentes automaticamente, e permitir que o usuário pudesse interagir na hora de escolher qual caminho seguir na

busca em uma destas ramificações.

Uma solução dada no mapeamento definido aqui para os N-grafos foi permitir a geração dos dois ramos da árvore, para uma posterior junção. Em casos onde o interesse seja mais prático e menos didático, este esquema pode ser abandonado e adotar-se um onde fosse necessário o uso de apenas um dos ramos.

7.2.4 Exemplos

A Figura 7.14 mostra como o N-grafo de $A \wedge (B \vee C) \vdash (A \wedge B) \vee (A \wedge C)$ é gerado pelo método. Na Figura 7.15 temos a parte da árvore de busca utilizada para encontrar este grafo. Neste caso, foram utilizadas regras-$\uparrow$ e regras-$\downarrow$.

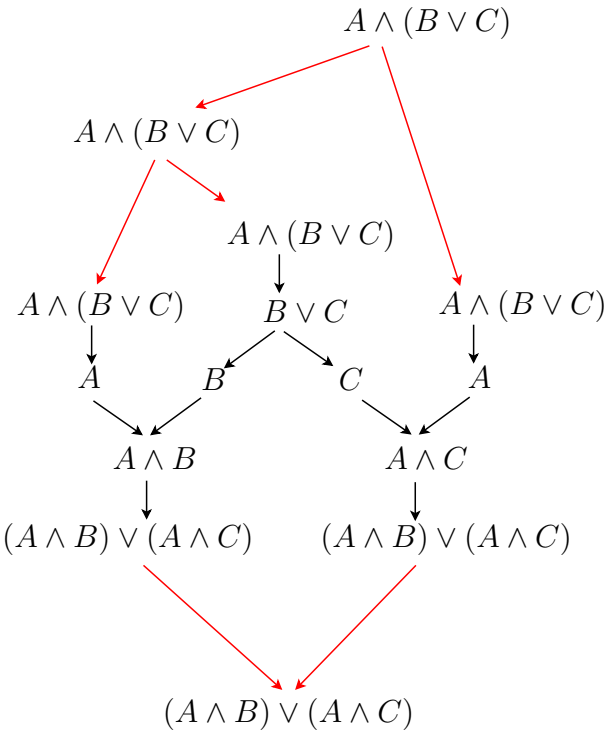

Figura 7.14 N-grafo de $A \wedge (B \vee C) \vdash (A \wedge B) \vee (A \wedge C)$.

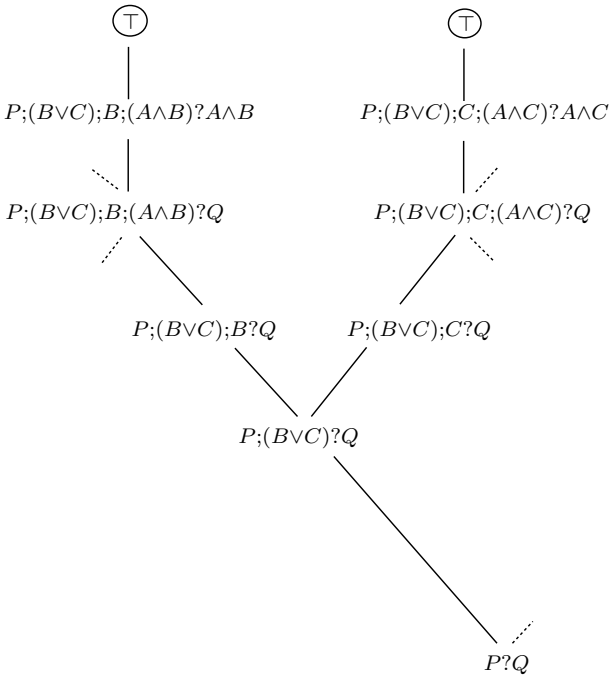

Figura 7.15 Árvore de busca de $A \wedge (B \vee C) \vdash (A \wedge B) \vee (A \wedge C)$. Na figura, $P = A \wedge (B \vee C)$ e $Q = (A \wedge B) \vee (A \wedge C)$. As linhas pontilhadas representam ramos disjuntos que não foram analisados.

CAPÍTULO 8

CONCLUSÃO

Este trabalho estudou os sistemas dedutivos existentes para a lógica clássica mais próximos no raciocínio natural humano: os baseados na *Dedução Natural* de Gentzen. Foi visto como estes sistemas evoluíram a partir de uma forma inicial, que permitia de geração de uma única conclusão, para sistemas mais simétricos que permitem múltiplas premissas e conclusões.

Este processo de evolução, passando por problemas de simetria e o da inadequação, foi apresentado de forma a culminar no sistema objetivo deste trabalho: os *N-grafos*. Neste último, uma prova tem todas as características dos sistemas anteriores, como naturalidade das regras de derivação, simetria nos operadores, geração de múltiplas conclusões, acrescido da explicitação dos aspectos estruturais de uma prova.

Este aspectos geométricos, que podem ser manipulados na estrutura de um grafo de prova, permitem que este possa ser analisado como um simples grafo. Assim, é possível a extensão de algoritmos clássicos de uma área que hoje está bem fundamentada, a *teoria dos grafos*, para a aplicação em provas da lógica clássica proposicional. Uma aplicação vista neste trabalho foi a extensão da *busca em profundidade*, aplicada para verificar a aciclicidade de todos os grafos de chaveamento de um grafo de prova, que permitiu a verificação mais eficiente do critério de corretude.

Para a geração automática de provas, foi extendida uma versão definida para o cálculo de dedução natural, de modo que pudesse ser aplicável nos N-grafos, e outra a partir do princípio da resolução, que é capaz de gerar um N-grafo que representa este processo. Ambas possuem aspectos relevantes: a primeira gera grafos mais naturais, considerando um mecanismo de prova mais próximo da maneira construtiva que as provas são construídas; e o segundo é baseado em uma prova por refutação, e abre portas para trabalhos que definam um mapeamento das técnicas de otimização já definidas para a resolução para os N-grafos.

Neste trabalho foram apresentados mecanismos para a verificação de provas no sistema dos N-grafos, e para a geração automática de grafos de prova válidos para teoremas escritos na lógica proposicional. A partir de tais técnicas, um trabalho futuro seria a implementação de um gerador automático ou um assistente de provas para os N-grafos.

Outro trabalho futuro é o estudo da meta-condição. Pode-se buscar um algoritmo polinomial para sua verificação, caso exista. Além disso, as técnicas de geração de grafos de prova podem ser extendidas para lidar com toda a lógica clássica, dado que aqui os apresentados são restridos ao fragmento com o $\neg, \wedge$ e $\vee$.

APÊNDICE A

TEORIA DOS GRAFOS

Aqui apresentamos algumas noções da teoria dos grafos adotadas neste trabalho. Estas definições foram extraídas do livro [Har72].

Definição A.1 (grafo)**.** Um *grafo* G consiste de um par (V, E), onde V é um conjunto não vazio de pontos, chamados *vértices* (ou *nós*); e E é um conjunto prescrito de pares não ordenados de vertices distintos de V. Cada par $e = \{u, v\}$ de vértices em E é chamado *aresta* (ou *linha*) de G, e liga u a v; e o vértice u e a aresta e são incidentes um no outro, assim como v e e.

Usaremos a notação $V(G)$ e $E(G)$ para denotas o conjunto dos vértices e arestas de um grafo G, respectivamente. Letras minúsculas serão utilizadas para representar vértices, com a exceção da letra e, que representará arestas. Grafos são representados graficamente por diagramas, onde cada vértice é um ponto e cada aresta uma linha ligando dois pontos existentes.

Definição A.2 (vértices adjacentes)**.** Em um grafo G, dois vértices u e v são ditos *adjacentes* se existe uma aresta $e = \{u, v\}$ em $E(G)$.

Definição A.3 (arestas adjacentes)**.** Se duas arestas distintas e_1 e e_2 de um grafo G são incidentes a um vértice em comum, então elas são *arestas adjacentes*.

Definição A.4 (subgrafo)**.** Um *subgrafo* de um grafo G é um grafo que contém todos os seus vértices e todas as suas arestas em G. Ou seja, se G' é um subgrafo de G, temos que $V(G') \subseteq V(G)$ e $E(G') \subseteq E(G)$.

Definição A.5 (subgrafo gerador)**.** Um *subgrafo gerador* é um subgrafo G' de um grafo G contendo todos os vértices de G.

Definição A.6 (grafo direcionado)**.** Um *grafo direcionado* ou *digrafo* G consiste de um conjunto não vazio de vértices e um conjunto de pares ordenados de vértices distintos. Estas arestas são denominadas *arestas direcionadas*.

Definição A.7 (caminho)**.** Um *caminho* $v_1, \ldots, v_n, comn > 1$, em um grafo G é uma sequência de de vértices distintos onde (v_i, v_{i+1}) é uma aresta de G para $1 \leq i < n$.

Definição A.8 (subcaminho)**.** Um *subcaminho* de um caminho $v_1, \ldots, v_n$ é um caminho $v_i, ..., v_j$ com $1 \leq i < j \leq n$.

Definição A.9 (semicaminho)**.** Um *semicaminho* em um digrafo é uma sequência alternada de vértices e arestas distintos $v_0, e_1, v_1, \ldots, e_n, v_n$, onde cada aresta e_i pode ser (v_{i-1}, v_i) ou (v_i, v_{i-1}).

Definição A.10 (ciclo)**.** Uma sequência de vértices $v_1, \ldots, v_n, v_1$ é um *ciclo* se a sequência $v_1, \ldots, v_n$ for um caminho e existir uma aresta $e = \{v_1, v_n\}$ no grafo G.

Definição A.11 (semiciclo)**.** Uma sequência alternante de vértices e arestas $v_0, e_1, v_1, \ldots, e_n, v_0$ é um *semiciclo* se a sequência $v_0, \ldots, v_n$ é um semicaminho e a aresta $e_n = (v_n, v_0)$ ou $e_n = (v_0, v_n)$.

Definição A.12 (grafo conexo)**.** Um grafo G é *conexo* se existe pelo menos um caminho entre todos os pares de vértices distintos de G. Senão, G é *desconexo*.

Definição A.13 (árvore)**.** Um grafo é acíclico se não possui nenhum ciclo. Uma *árvore* é um grafo *acíclico* e *conexo*.

Definição A.14 (vértice alcançável)**.** Em um grafo direcionado G, se existe um caminho do vértice v para o vértice u, u é dito ser *alcançável* a partir de v.

Definição A.15 (vértice inicial/final/intermediário)**.** Em um digrafo G, um vértice é *inicial* se não existe aresta direcionada a ele e *final* se nenhuma aresta parte dele. Um vértice que não é inicial nem final é *intermediário*.

Definição A.16 (grau)**.** O *grau* de um vértice em um grafo G é o número de arestas incidentes a ele.

O seguinte teorema foi tirado do livro do Cormen [THCS02]:

Teorema A.1 (**teorema dos parêntesis**)**.** *Em qualquer busca em profundidade em um grafo $G(V, E)$, para dois vértices quaisquer u e v, exatamente uma das três condições a seguir é válida:*

- *Os intervalos $[d[u], f[[u]]$ e $[d[v], f[[v]]$ são completamente disjuntos, e nem u nem v é um descendente do outro na árvore de profundidade;*

- *O intervalo* $[d[u], f[[u]]$ *está totalmente contido no intervalo* $[d[v], f[[v]]$*, e* u *é descendente de* v *na árvore de profundidade;*
- *O intervalo* $[d[v], f[[v]]$ *está totalmente contido no intervalo* $[d[u], f[[u]]$*, e* v *é descendente de* u *na árvore de profundidade.*

APÊNDICE B

CÁLCULO DE SEQUENTES

Em [Gen35], Gentzen introduz um cálculo de dedução que se baseia na introdução e eliminação de símbolos lógicos, assim como a dedução natural, mas que exclui das derivações as fórmulas tomadas como suposições.

Uma fórmula $\mathfrak{B}$, derivada pela dedução natural a partir das suposições $\mathfrak{A}_1, ..., \mathfrak{A}_\mu$, pode ser escrita como

$$(\mathfrak{A}_1 \wedge \ldots \wedge \mathfrak{A}_\mu) \rightarrow \mathfrak{B}$$

que por si só é uma fórmula válida, livre de hipóteses. Para poder lidar com estas tautologias introduzindo e eliminando símbolos, como intencionado, é necessário reescrever estas fórmulas sem estes símbolos adicionais. Para isso ele criou o conceito de *sequente*. No lugar de utilizar a fórmula acima, agora utiliza-se o sequente

$$\mathfrak{A}_1, \ldots, \mathfrak{A}_\mu \vdash \mathfrak{B}$$

que difere da primeira na estrutura formal, mas mantém o significado semântico.

De maneira geral, um sequente é uma expressão $\Gamma \vdash \Delta$, onde Γ, chamado antecessor, e Δ, sucessor, são sequências finitas de fórmulas $\mathfrak{A}_1, \ldots, \mathfrak{A}_n$ e $\mathfrak{B}_1, \ldots, \mathfrak{B}_m$. A semântica de um sequente é a *disjunção* de Γ implica na *conjunção* de Δ, ou formalmente:

$$(\mathfrak{A}_1 \wedge \ldots \wedge \mathfrak{A}_n) \rightarrow (\mathfrak{B}_1 \vee \ldots \vee \mathfrak{B}_m)$$

Quando o antecessor é vazio, o sequente se reduz à fórmula $\mathfrak{B}_1 \vee \ldots \vee \mathfrak{B}_m$. Quando o sucessor é vazio, o sequente significa a negação do antecessor, $\neg(\mathfrak{A}_1 \wedge \ldots \wedge \mathfrak{A}_n)$, ou simplesmente $(\mathfrak{A}_1 \wedge \ldots \wedge \mathfrak{A}_n) \rightarrow \perp$. Se ambos forem vazios, temos o sequente que significa o falso, ou simplesmente $\perp$.

As derivações no cálculo de sequentes consistem de sequentes iniciais (*átomos* ou *axiomas*), da forma $\mathfrak{A} \vdash \mathfrak{A}$, onde $\mathfrak{A}$ é uma fórmula qualquer, e de figuras de inferência

que devem obedecer a uma das regras de inferência, que são divididas em dois grandes drupos:

- Regras estruturais (Tabela B.1)
- Regras operacionais (lógicas) (Tabela B.2)

Thinning	$\dfrac{\Gamma \vdash \Delta}{\mathfrak{A}, \Gamma \vdash \Delta}$	$\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \mathfrak{B}}$
Contração	$\dfrac{\mathfrak{A}, \mathfrak{A}, \Gamma \vdash \Delta}{\mathfrak{A}, \Gamma \vdash \Delta}$	$\dfrac{\Gamma \vdash \Delta, \mathfrak{B}, \mathfrak{B}}{\Gamma \vdash \Delta, \mathfrak{B}}$
Interchange	$\dfrac{\Lambda, \mathfrak{A}_1, \mathfrak{A}_2, \Gamma \vdash \Delta}{\Lambda, \mathfrak{A}_2, \mathfrak{A}_1, \Gamma \vdash \Delta}$	$\dfrac{\Gamma \vdash \Theta, \mathfrak{B}_1, \mathfrak{B}_2, \Delta}{\Gamma \vdash \Theta, \mathfrak{B}_2, \mathfrak{B}_1, \Delta}$
Corte	$\dfrac{\Gamma \vdash \Delta, \Sigma \qquad \Sigma, \Lambda \vdash \Theta}{\Gamma, \Lambda \vdash \Delta, \Theta}$	

Tabela B.1 Regras estruturais

As regras estruturais são figuras de inferências que não se integram ao sistema original de introduções e eliminações de símbolos, mas manipulam a estrutura do sequente. Girard em [Gir89] declara que estas regras são as mais importantes em todo o cálculo, pois sem escrever nenhum símbolo, o comportamento futuro dos operadores lógicos é determinado por elas. Já as regras operacionais, ou lógicas, são bastante similar às utilizadas na dedução natural. A grande diferença está na simetria entre introdução e eliminação de todos os conectivos lógicos, que não existe nos casos da disjunção e da implicação na dedução natural.

O cálculo sequente foi criado para suprir a necessidade de uma simetria que não existia na dedução natural. Gentzen buscava resolver a incompatibilidade entre o sistema intrinsecamente simétrico (a lógica clássica) e o cálculo de dedução natural clássico NK, cuja simetria falha ao lidar com a *"lei do terceiro excluído"*.

No cálculo de sequentes, a estrutura é na verdade a raiz de toda a simetria. Com ela,

$$\wedge - IS: \quad \frac{\Gamma \vdash \Delta, \mathfrak{A} \qquad \Gamma \vdash \Delta, \mathfrak{B}}{\Gamma \vdash \Delta, \mathfrak{A} \wedge \mathfrak{B}} \wedge\text{-}esq$$

$$\wedge - IA: \quad \frac{\mathfrak{A}, \Gamma \vdash \Delta}{\mathfrak{A} \wedge \mathfrak{B}, \Gamma \vdash \Delta} \wedge\text{-}dir_1 \qquad \frac{\mathfrak{B}, \Gamma \vdash \Delta}{\mathfrak{A} \wedge \mathfrak{B}, \Gamma \vdash \Delta} \wedge\text{-}dir_2$$

$$\vee - IA: \quad \frac{\mathfrak{A}, \Gamma \vdash \Delta \qquad \mathfrak{B}, \Gamma \vdash \Delta}{\mathfrak{A} \vee \mathfrak{B}, \Gamma \vdash \Delta} \vee\text{-}esq$$

$$\vee - IS: \quad \frac{\Gamma \vdash \Delta, \mathfrak{A}}{\Gamma \vdash \Delta, \mathfrak{A} \vee \mathfrak{B}} \vee\text{-}dir_1 \qquad \frac{\Gamma \vdash \Delta, \mathfrak{B}}{\Gamma \vdash \Delta, \mathfrak{A} \vee \mathfrak{B}} \vee\text{-}dir_2$$

$$\rightarrow - IS: \quad \frac{\mathfrak{A}, \Gamma \vdash \Delta, \mathfrak{B}}{\Gamma \vdash \Delta, \mathfrak{A} \rightarrow \mathfrak{B}} \rightarrow\text{-}dir$$

$$\rightarrow - IA: \quad \frac{\Gamma \vdash \Delta, \mathfrak{A} \qquad \mathfrak{B}, \Lambda \vdash \Theta}{\mathfrak{A} \rightarrow \mathfrak{B}, \Gamma, \Lambda \vdash \Delta, \Theta} \rightarrow\text{-}esq$$

$$\neg - IS: \quad \frac{\mathfrak{A}, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg\mathfrak{A}} \neg\text{-}dir$$

$$\neg - IA: \quad \frac{\Gamma \vdash \Delta, \mathfrak{A}}{\neg\mathfrak{A}, \Gamma \vdash \Delta} \neg\text{-}esq$$

Tabela B.2 Regras operacionais

temos as dualidades entre os lados esquerdo e direito do *'turnstile'*:

$$\begin{array}{c} \textit{negativo} \vdash \textit{positivo} \\ \textit{conjunção} \vdash \textit{disjunção} \end{array}$$

além da simetria entre as regras $\wedge - IS$ e $\vee - IA$, $\wedge - IA$ e $\vee - IS$, que não existe na dedução natural por causa da regra de eliminação do $\vee$. É graças também a esta estrutura que as regras de introdução e eliminação tem uma relação de inverso.

Com o cálculo de sequentes Gentzen conseguiu provar o *Hauptsatz*. Este teorema afirma que qualquer derivação com aplicações da regra do corte pode ser transformada em uma a mesma, e ainda oferece um procedimento sistemático para sua eliminação.

REFERÊNCIAS BIBLIOGRÁFICAS

[ACL12] *ACL2*, 2012 (acessado em 04 Jul 2012). http://www.cs.utexas.edu/ moore/acl2.

[Alv09] Gleifer Vaz Alves. *Transformations for proof-graphs with cycle treatment augmented via geometric perspective techniques*. PhD thesis, Universidade Federal de Pernambuco, Recife, June 2009.

[APr12] *AProS: automated proof search*, 2012 (acessado em 04 Jul 2012). http://www.phil.cmu.edu/projects/apros/index.php?page=generator.

[CL73] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press Inc. (London) LTD., 1973.

[Coq12] *The Coq Proof Assistant*, 2012 (acessado em 04 Jul 2012). http://coq.inria.fr.

[dO01] Anjolina Grisi de Oliveira. *Proofs From a Geometric Perspective*. PhD thesis, Universidade Federal de Pernambuco, Recife, February 2001.

[DR89] V. Danos and L. Regnier. The structure of multiplicatives. *Archive for Mathematical Logic 28:181-203*, 1989.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische schliessen. *Mathematische Zeitschirift*, pages 176–210 and 405–431, 1935. Tradução para o inglês: "Investigations into Logical Deduction" in The Collected Works of Gerhard Getzen, ed. M.E. Szabo, North-Holland Pub Co., 1969.

[Geu09] H. Geuvers. Proof assistants: History, ideas and future. *Sādhanā Vol. 34, Part 1, pp 3-25*, 2009.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 1987. London Mathematical 50:1, pp. 1-102.

[Gir89] Jean-Yves Girard. *Proofs and Types*. Cambridge University Press, 1989.

[Har72] Frank Harary. *Graph Theory*. Addison-Wesley Publishing Company, 1972.

[Isa12] *Isabelle*, 2012 (acessado em 04 Jul 2012). http://www.cl.cam.ac.uk/research/hvg/isabelle.

[KK62] W. Kneale and M. Kneale. The development of logic. *Oxford*, 1962.

[Kne58] W. Kneale. The province of logic. *Contemporary British Philosophy*, 1958.

[Pra65] Dag Prawitz. *Natural Deduction. A Proof-Theoretical Study*, volume 3 of *Acta Universitatis Stockholmiensis*. Almqvist and Wiksell, Stockholm, 1965.

[Pra71] Dag Prawitz. Ideas and results in proof theory. *Proceedings 2nd Scandinavian Logic Symposium, North-Holland*, 1971. In J. E. Fenstad, editor.

[SB98] Wilfried Sieg and John Byrnes. Normal natural deduction proofs (in classical logic). *Studia Logica 60, 67-106*, 1998.

[SS78] D. J. Shoesmith and T. J. Smiley. Multiple-conclusion logic. *Cambridge University Press*, 1978.

[SS92] Wilfried Sieg and Richard Scheines. Search for proofs (in sentential logic). *Philosophy and the Computer (L. Burkholder, editor), 137-159*, 1992.

[SS94] Leon Sterling and Ehud Shapiro. *The Art of Prolog*. The MIT Press, 1994.

[THCS02] Ronald L. Rivest Thomas H. Cormen, Charles E. Leiserson and Clifford Stein. *Introduction to Algorithms*. McGraw-Hill Science/Engineering/Math, 2nd edition, 2002.

[Ung92] A. M. Ungar. Normalization, cut-elimination and the theory of proofs. *Number 28 of CSLI Lecture Notes, Center for the Study of Language and Information, Stanford*, 1992.

[Yar12] *Yarrow Home Page*, 2012 (acessado em 04 Jul 2012). http://www.cs.ru.nl/ janz/yarrow.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).