UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO – BCC

ALEXANDRE JERONIMO SEHNEM

FORMALIZAÇÃO DA TRADUÇÃO DAS LÓGICAS CLÁSSICA E INTUICIONISTA PARA A LÓGICA LINEAR EM COQ

JOINVILLE
2023

ALEXANDRE JERONIMO SEHNEM

FORMALIZAÇÃO DA TRADUÇÃO DAS LÓGICAS CLÁSSICA E INTUICIONISTA PARA A LÓGICA LINEAR EM COQ

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Karina Girardi Roggia
Coorientador: Paulo Henrique Torrens

JOINVILLE
2023

ALEXANDRE JERONIMO SEHNEM

FORMALIZAÇÃO DA TRADUÇÃO DAS LÓGICAS CLÁSSICA E INTUICIONISTA PARA A LÓGICA LINEAR EM COQ

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Karina Girardi Roggia
Coorientador: Paulo Henrique Torrens

BANCA EXAMINADORA:

Orientadora:

Dra. Karina Girardi Roggia
UDESC

Coorientador:

Me. Paulo Henrique Torrens
University of Kent

Membros:

Dr. Cristiano Damiani Vasconcellos
UDESC

Me. Rafael Castro Gonçalves Silva
University of Copenhagen

Joinville, 28 de junho de 2023

RESUMO

Desde que a lógica linear foi inicialmente proposta, sua relevância tem aumentado constantemente, pois seu funcionamento permite que ela modele problemas que não são abrangidos pela lógica clássica, como seu não determinismo. Em virtude disso, se faz necessário a definição de ferramentas que permitam a verificação de propriedades em sistemas equivalentes a lógica linear. Para tanto, através do assistente de provas *Coq*, é possível criar um formalismo que permita que se construa uma tradução da lógica clássica para a linear e assim utilizar os recursos e propriedades da lógica para criar provas que contornem o problema do não determinismo e que ainda estejam formalizadas e possam ser reutilizadas. Adicionalmente, essa tradução para a lógica linear em *Coq* permite que se defina de forma simples e automatizada um valor computacional pra as provas construídas através do mesmo.

Palavras-chave: Lógica Linear. Coq. Confluência. Lógica Proposicional Clássica. Lógica Intuicionista.

ABSTRACT

Since linear logic was initially proposed, it's relevance has steadily increased, as its operation allows it to model problems that are not covered by classical logic, such as it's non determinism. As a result, it is necessary to define the tools that allow the verification of properties in systems equivalent to linear logic. Therefore, through the Coq proof assistant, it is possible to create a formalism that allows the construction of a translation of classical logic to linear logic and then, utilize the resources and properties of the logic to create proofs that circumvent the problem of non determinism and that are still formalized so that they can be reutilized. Additionally, this translation to linear logic in Coq allows a simple and automated definition of a computational value for the proofs built through it.

Keywords: Linear Logic. Coq. Confluence. Classical Propositional Logic. Intuitionistic Logic.

LISTA DE ILUSTRAÇÕES

Figura 1 – Gramática da Lógica Clássica 12
Figura 2 – Comutação e distributividade para lógica clássica 14
Figura 3 – (i) Regra do terceiro excluído, (ii) eliminação da dupla negação 14
Figura 4 – Dedução natural para a lógica intuicionista 16
Figura 5 – Regras de *weakening* e *contraction* para dedução natural 16
Figura 6 – Regras de dedução natural adicionais para a lógica clássica 17
Figura 7 – Gramática da Lógica Linear 19
Figura 8 – Negação Linear 22
Figura 9 – Cálculo de Sequentes para a lógica linear 23
Figura 10 – Comparação de um sequente da lógica linear com um sequente da lógica clássica 24
Figura 11 – Distribuição Linear 26
Figura 12 – Comutatividade dos Operadores 26
Figura 13 – Distributividade da conjunção multiplicativa sobre a disjunção aditiva . . . 27
Figura 14 – Distributividade da disjunção multiplicativa sobre a conjunção aditiva . . . 28
Figura 15 – Isomorfismo do exponencial "!" 29
Figura 16 – Isomorfismo do exponencial "?" 29
Figura 17 – Modelagem das ações de Liga/Desliga de uma máquina 30
Figura 18 – Eliminação do corte: *tensor* e *par* 32
Figura 19 – Eliminação do corte: *identidade* 33
Figura 20 – Não conformidade da lógica linear com a propriedade *Termination* 35
Figura 21 – Exemplo de *Term Rewriting* confluente 36
Figura 22 – Exemplo de *Term Rewriting* não confluente 37
Figura 23 – Removendo Corte na Lógica Clássica 38
Figura 24 – Não confluência de cálculo lambda *call/cc* 39
Figura 25 – Regras de tradução da Lógica Intuicionista para a Lógica Linear 42
Figura 26 – Regra base para tradução de LI 42
Figura 27 – Regras de tradução *call-by-name* da Lógica Intuicionista para a Lógica Linear 45
Figura 28 – Regras de tradução da Lógica Clássica para a Lógica Linear 46
Figura 29 – Definição formal da tradução *call-by-value* da lógica intuicionista 48
Figura 30 – Formalismo para o teorema da tradução *call-by-value* e *call-by-name* da lógica intuicionista 48

LISTA DE TABELAS

Tabela 1 – Comparativo entre os trabalhos relacionados 51

SUMÁRIO

1 INTRODUÇÃO 9
1.1 OBJETIVO GERAL 10
1.2 OBJETIVOS ESPECÍFICOS 10
1.3 METODOLOGIA 10
1.4 ESTRUTURA DO TRABALHO 10
2 LÓGICA CLÁSSICA E INTUICIONISTA 12
2.1 LÓGICA CLÁSSICA 12
2.2 LÓGICA INTUICIONISTA 14
2.3 DEDUÇÃO NATURAL 15
2.3.1 Dedução Natural para Lógica Clássica 17
3 LÓGICA LINEAR 18
3.1 LINGUAGEM 19
3.2 SISTEMA DEDUTIVO 21
3.2.1 Cálculo de Sequentes 21
3.2.2 Cálculo de Sequentes para a Lógica Linear 21
3.3 PROPRIEDADES DOS CONECTIVOS 24
3.3.1 Teorema da Dedução e Equivalência 25
3.3.2 Comutatividade e Distributividade 25
3.3.3 Exponenciais 27
3.4 CODIFICAÇÃO DE ESTADOS 30
3.5 ELIMINAÇÃO DA REGRA DO CORTE 31
4 *TERM REWRITING* E CONFLUÊNCIA 34
4.1 CONFLUÊNCIA 35
4.1.1 Call-by-name *vs* Call-by-value 37
4.2 CONFLUÊNCIA APLICADA À LÓGICA 38
5 PROVA DAS TRADUÇÕES 41
5.1 TRADUÇÃO DA LÓGICA INTUICIONISTA 42
5.1.1 Variações da Tradução para a lógica intuicionista 45
5.2 TRADUÇÃO DA LÓGICA CLÁSSICA 45
5.3 FORMALIZAÇÃO DAS TRADUÇÕES NO COQ 47
6 TRABALHOS RELACIONADOS 49
6.1 XAVIER (2017) 49
6.2 LAURENT (2017) 50
6.3 KALVALA E PAIVA (1995) 50
6.4 CONSIDERAÇÕES SOBRE OS TRABALHOS RELACIONADOS 51

7 CONSIDERAÇÕES FINAIS 52
7.1 TRABALHOS FUTUROS **52**
REFERÊNCIAS 53

1 INTRODUÇÃO

Desde que foi inicialmente proposta, a lógica como uma área de estudo tem se aprimorado e evoluído constantemente através dos séculos, o que culmina na descoberta de sua relação com a matemática no início do século XX, possibilitando a fundação dos sistemas formais que são utilizados até os dias de hoje pela Ciência da Computação (BORNAT, 2005). Estes conceitos, após serem refinados, passaram a ser chamados de lógica clássica. Entretanto, como Priest (2009) atesta, há fórmulas que não podem ser propriamente descritas pela lógica clássica, e assim novas propostas de lógicas foram criadas com o intuito de suprir esta deficiência de representatividade.

A estas lógicas se é dado o nome de lógicas não clássicas, e temos nesta categoria, por exemplo, a lógica intuicionista, a qual tem como principal característica seu tratamento de fórmulas como provas e o não cumprimento do princípio do terceiro excluído. Nesta mesma linha de reimaginar a interpretação de suas fórmulas, temos a lógica *fuzzy*, a qual possui um sistema multivalorado onde suas fórmulas, em vez de representar verdades ou falsidades absolutas, permite a representação do conceito de verdades parciais.

E ainda seguindo esta subversão da interpretação das fórmulas, a lógica linear é uma das lógicas que propõem uma expansão dos conceitos propostos pela lógica clássica, sem deixar de atingir o nível de representação da mesma (GIRARD, 1987). Para tanto, a lógica linear modifica a noção atribuída às fórmulas na lógica clássica e na intuicionista, onde ao invés de ter seu foco voltado respectivamente para verdades e provas, as fórmulas passam a ser tratadas como recursos (COSMO; MILLER, 2019).

Estas propriedades inerentes à lógica linear fazem com que ela possua uma verosimilhança muito maior com o nosso mundo, já que o funcionamento da grande maioria dos sistemas mecânicos e até mesmo orgânicos funcionam a partir do pressuposto que os recursos existente são finitos e são consumidos pelo sistema (XAVIER, 2017). Portanto, a lógica linear encontra usos em áreas extremamente diversas, como na Biologia Molecular (CHAUDHURI; DESPEYROUX, 2013), Computação tradicional (ALEXIEV, 1994), Linguística (MOOT; PIAZZA, 2001) e Computação Quântica (LAGO; FAGGIAN, 2012). Além destas áreas, existe um paralelo interessante que pode ser traçado entre a lógica linear e as redes de Petri, como discutido por Martí-Oliet e Meseguer (1989), em que é demonstrada a facilidade com que a lógica linear pode representar sistemas concorrentes.

Com a aparição de novos problemas e aplicações para a lógica linear, surgem formalizações que permitem a verificação de propriedades da lógica e na especificação de softwares que a usem. E os assistentes de provas, como Coq e Isabelle, são ótimas ferramentas para fornecer auxílio no desenvolvimento formal de provas. Segundo Xavier (2017), o assistente de provas *Coq* demonstra ser a o assistente de provas mais confiável para a formalização de sistemas, permitindo por exemplo, a implementação do cálculo de sequentes em *Coq* como um sistema dedutivo para a lógica linear. E esta característica se deve, em grande parte, a sua linguagem de

especificação que permite a construção de axiomas para teorias próprias e seu sistema de táticas que permite o desenvolvimento de provas (PAULIN-MOHRING, 2011).

E como demonstrado por Nishizaki (1993), a lógica linear pode inclusive ser utilizada para compensar fraquezas presentes em outras lógicas, como por exemplo, o não determinismo da lógica clássica. Assim, surge o interesse de se desenvolver um formalismo em *Coq* para a tradução da lógica proposicional clássica para a lógica linear, como proposto por Girard (1987), e verificar os méritos que tal abordagem pode proporcionar.

1.1 OBJETIVO GERAL

Este trabalho tem como objetivo a criação de um formalismo para a tradução da lógica proposicional clássica para a lógica linear, através do uso do assistente de provas *Coq*.

1.2 OBJETIVOS ESPECÍFICOS

Com base no objetivo geral, são definidos os seguintes objetivos específicos:

- Apresentar de forma didática e coesa a lógica linear.
- Demonstrar a capacidade de se criar provas para a lógica linear em *Coq*.
- Formalizar a tradução da lógica proposicional clássica em *Coq*.
- Comprovar a capacidade de confluência da lógica linear.

1.3 METODOLOGIA

O procedimento metodológico utilizado neste trabalho tem como base o levantamento de uma fundamentação teórica sobre a lógica linear, a lógica clássica e a lógica intuicionista, dissecando o funcionamento de cada uma delas, bem como de suas propriedades. Também foi realizado um estudo sobre *Term Rewriting System* e, em conjunto, foi realizado uma revisão bibliográfica nos trabalhos relacionados para se traçar possíveis abordagens que possam ser utilizadas no desenvolvimento das tarefas propostas.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho está organizado no seguinte formato: o Capítulo 2 apresenta a lógica linear e a intuicionista, introduzindo para ambas o seu funcionamento, gramática e sistema dedutivo, além de também demonstrar diferenças entre as duas. O Capítulo 3 apresenta a lógica linear, sua linguagem, sistema dedutivo e propriedades da lógica.

Em seguida, o Capítulo 4 explica o conceito de *Term Rewriting* e a relação que as lógicas possuem com a confluência. O Capítulo 5 detalha em profundidade como é feito a tradução das

lógicas clássica e intuicionista para a lógica linear, ressaltando as diferenças existentes e suas finalidades.

O Capítulo 6 apresenta trabalhos relacionados ao objetivo deste e discute possíveis aprendizados obtidos através da análise dos mesmos. O capítulo 7 discute os aprendizados obtidos neste do trabalho e explana possibilidades para a continuação do projeto.

2 LÓGICA CLÁSSICA E INTUICIONISTA

As lógicas clássica e intuicionista possuem abordagens distintas em seu tratamento de fórmulas lógicas e interpretação de seus significados. Enquanto a lógica clássica possui raízes profundas na lógica aristotélica, que busca demonstrar a verdade ou falsidade dos fatos, a lógica intuicionista abstrai este conceito através da interpretação de fórmulas como provas e da rejeição de diversas propriedades inerentes da lógica clássica (TROELSTRA; SCHWICHTENBERG, 2000).

Entretanto, mesmo com estas diferenças, é inegável as similaridades que ambas também possuem. Assim, neste Capítulo estas propriedades serão provadas e exemplificadas. Na Seção 2.1 será explicado o funcionamento da lógica clássica, através da definição de seus operadores e o comportamento dos mesmos, bem como uma demonstração das regras inerentes à lógica. Na Seção 2.2, será demonstrado o funcionamento da lógica intuicionista e também serão explicadas as diversas diferenças e similaridades compartilhadas pelas lógicas. Por fim, na Seção 2.3, será apresentado um sistema dedutivo para ambas as lógicas, e também as diferenças que este sistema deve ter para representar cada uma das lógicas.

2.1 LÓGICA CLÁSSICA

A lógica clássica, também conhecida como lógica aristotélica ou lógica tradicional, é um ramo da lógica formal que tem suas origens na Grécia antiga. Ela desempenhou um papel significativo no desenvolvimento da matemática, filosofia e ciência da computação. A história da lógica clássica remonta às obras de Aristóteles, que estabeleceu as bases para o raciocínio lógico e desenvolveu o sistema de lógica silogística. As teorias lógicas de Aristóteles forneceram um quadro para analisar proposições e tirar conclusões válidas, formando a base da lógica clássica (PRIEST, 2017).

Atualmente, a lógica clássica é vista como um sistema formal que visa provar a validade de proposições lógicas a partir de um conjunto de regras e operadores. Assim, ela tem como foco interpretar se uma fórmula ou proposição é verdadeira ou falsa. Para construir estas valorações, a lógica clássica define em sua gramática (descrita na Figura 1) três operadores principais: conjunção ($\wedge$), disjunção ($\vee$) e negação ($\neg$), duas constantes ($\bot$ e $\top$) (PRIEST, 2017).

$$\begin{array}{rcl} F & ::= & a \mid \neg a \\ & \mid & \bot \mid \top \\ & \mid & F \wedge F \\ & \mid & F \vee F \end{array}$$

Figura 1 – Gramática da Lógica Clássica

Além destes, ainda há o operador de implicação ($\Rightarrow$) presente na lógica clássica. Entretanto ele não está explicitado na gramática pois é construído a partir da utilização do operador

de disjunção da seguinte forma: $(a \Rightarrow b)$ equivale a $(\neg a \vee b)$ (PRIEST, 2017). Dessa maneira, podemos definir a interpretação da valoração de cada um dos operadores da seguinte forma:

- As constantes $\bot$ e $\top$ são sempre falsa e verdadeira respectivamente.
- Uma fórmula a só é verdade se a fórmula $\neg a$ for falsa.
- Uma fórmula $A \wedge B$ é verdade somente se ambos, A e B, forem verdadeiros.
- Uma fórmula $A \vee B$ é verdade se A for verdade, se B for verdade ou se ambos forem verdadeiros.
- Uma fórmula $A \Rightarrow B$ é sempre verdade se A for falso. Porém se A for verdadeiro, então ela somente será verdade se B também for verdadeiro.

Com estas regras definidas, podemos facilmente utilizar esta gramática para construir frases ou expressões de linguagens naturais dentro da lógica clássica (ENDERTON, 2001). Por exemplo, a frase "Não irá chover e também não fará sol", pode ser traduzida para: $(\neg A \wedge \neg B)$, onde A representa chover e B representa fazer sol. Pode-se verificar que esta tradução está de fato correta pois, se essa fórmula for interpretada tendo em mente as definições já estabelecidas para a lógica, veremos que ela só pode ser verdade se A e B forem falsos, ou seja, se não chover nem fizer sol respectivamente.

Entretanto, existe outra fórmula clássica que também atenderia as limitações definidas pela fórmula do exemplo anterior, sendo ela: $\neg(A \vee B)$. Isso se deve ao fato que, tanto a fórmula anterior como $(\neg A \wedge \neg B)$ são equivalentes. E o que permite essa característica é uma das propriedades do operador de negação na lógica clássica, conhecida como *De Morgan* (ENDERTON, 2001). Essa regra determina as seguintes equivalências:

- $\neg(A \vee B) \equiv (\neg A \wedge \neg B)$
- $\neg(A \wedge B) \equiv (\neg A \vee \neg B)$

Uma característica interessante da lógica clássica que pode ser derivada do uso da regra de *De Morgan*, é a capacidade de se remover da gramática o operador de disjunção ou o de conjunção, sem que a lógica perca sua capacidade de expressão. Isso fica claro pois, como *De Morgan* define uma equivalência, basta substituir um operador por seu equivalente conforme a regra, que este operador se tornaria obsoleto para a expressividade da lógica (ENDERTON, 2001).

Outras regras observadas dentro da lógica clássica e fundamentais para a interpretação das fórmulas e construção de provas, são as regras de comutação e distributividade, como definidas na Figura 2. Note que o operador de implicação, por mais que seja definido através do operador de disjunção, não pode ser comutado de forma direta.

Comutação	Distributividade
$A \wedge B \equiv B \wedge A$	$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
$A \vee B \equiv B \vee A$	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
	$A \Rightarrow (B \wedge C) \equiv (A \Rightarrow B) \wedge (A \Rightarrow C)$

Figura 2 – Comutação e distributividade para lógica clássica

Finalmente, existem duas características implícitas a regras da linguagem que basicamente definem a lógica clássica. A primeira é a regra do terceiro excluído, a qual determina que uma fórmula do tipo $(A \vee \neg A)$ é sempre uma verdade. A última regra é chamada de eliminação da dupla negação e define que, a negação de uma fórmula negada é a própria fórmula (ENDERTON, 2001). Ambas as regras são formalmente definidas na Figura 3.

$$(A \vee \neg A) \equiv \top \qquad \neg\neg A \equiv A$$
$$\text{(i)} \qquad\qquad \text{(ii)}$$

Figura 3 – (i) Regra do terceiro excluído, (ii) eliminação da dupla negação

2.2 LÓGICA INTUICIONISTA

O intuicionismo como uma lógica foi pela primeira vez apresentado e formalizado pelo matemático e lógico holandês Arend Heyting em 1928. Entretanto o intuicionismo como um conceito matemático precede em muitos anos o seu fragmento lógico e é a base direta de onde a mesma foi derivada. O intuicionismo se baseia na construção de fatos e inferências que podem ser descritos por um linguagem e, devido a características da matemática intuicionista, essas descrições acabam apresentando padrões linguísticos. A validade desses padrões e de suas inferências são derivadas da validade das premissas contidas pelos mesmos, e é para o estudo matemático dessas inferências que a lógica intuicionista é utilizada (ATTEN, 2022).

Efetivamente a lógica intuicionista (LI) se baseia na construção de provas de seus elementos, onde uma prova de um elemento atômico é uma construção matemática intuicionista que demonstra que este elemento é verdade (ATTEN, 2022). Assim, em termos mais simples, a lógica intuicionista é uma dissidência da lógica clássica que não permite a utilização da regra do terceiro excluído e a remoção da dupla negação (MOSCHOVAKIS, 2022).

Como a lógica intuicionista possui a mesma gramática que a lógica clássica podemos, como demonstrado por Priest (2009), expandir o conceito intuicionista de provas para descrever o funcionamento dos outros conectivos da lógica clássica que também estão presentes na lógica intuicionista:

- Uma prova de $(A \wedge B)$, é um par composto por uma prova de A e uma prova de B.
- Uma prova de $(A \vee B)$, se obtém a partir de uma prova de A ou B.
- Uma prova de $(A \Rightarrow B)$, é uma construção que indica que, dada uma prova de A qualquer, é possível aplicá-la para criar uma prova de B .

- Uma prova de $(\neg A)$, indica que não é possível construir uma prova de A.

Através dessas definições, se torna simples deduzir as limitações estruturais impostas pelas regras da lógica intuicionista já mencionadas. A dupla negação $(\neg\neg A)$, por exemplo, indica a inviabilidade de construir uma prova de que não há prova de A. Então, diferente do que acontece na lógica clássica, não podemos inferir nenhuma informação sobre A através desta interpretação.

Da mesma forma, podemos expandir ainda mais essas definições e demonstrar a não conformidade da regra do terceiro excluído $(A \vee \neg A)$ ao aplica-lá em um problema real, como o da infinidade de número primos gêmeos.

Este problema visa provar a existência de infinitas combinações de números primos cuja diferença é igual a dois (Ex: 11 e 13, 59 e 61). Assim, determinando que:

- A seja uma possível prova para o problema.
- $\neg A$ seja uma prova de que não é possível construir uma prova para o problema.
- E que existe uma prova de $(A \vee \neg A)$.

Na lógica clássica, bastaria encontrar uma prova de que $\neg A$ é falso, ou seja, que existe uma prova de $\neg\neg A$, para que seja possível determinar que A existe. Entretanto, essa interpretação estaria ignorando uma terceira opção, a de que também não é possível construir uma prova de A (PRIEST, 2009).

Este exemplo torna óbvio a utilidade que a lógica intuicionista cumpre, especialmente para a computação. Se pudéssemos aplicar a regra do terceiro excluído sem nenhuma restrição junto ao Isomorfismo de Curry-Howard, por exemplo, poderíamos através da prova de $\neg\neg A$ construir uma prova e, por consequência do isomorfismo, um código que resolvesse A, mesmo sem que conseguíssemos de fato criar uma prova para A.

2.3 DEDUÇÃO NATURAL

A dedução natural é um sistema dedutivo presente em uma grande parcela das lógicas, incluindo a intuicionista e a clássica. Este sistema permite a criação de provas para uma fórmula a partir da criação de uma árvore de prova para essa mesma fórmula através de suposições abertas localizada nos nós do topo da árvore, também chamados de folhas (TROELSTRA; SCHWICHTENBERG, 2000).

Uma dedução neste sistema é construída de forma indutiva, onde sua base é uma árvore com um único nó que possui uma suposição. E o passo indutivo é construído a partir da aplicação de uma das regras descritas na Figura 4:

É importante destacar que as regras apresentadas possuem certas características que as diferenciam das regras de dedução natural para a lógica intuicionista encontradas na literatura.

$$\frac{\begin{array}{c}\vdots\\ \Gamma \vdash A\end{array} \quad \begin{array}{c}\vdots\\ \Delta \vdash B\end{array}}{\Gamma, \Delta \vdash A \wedge B}\ (\wedge \mathrm{I}) \qquad \frac{\begin{array}{c}\vdots\\ \Gamma \vdash A \wedge B\end{array}}{\Gamma \vdash A}\ (\wedge E_R) \qquad \frac{\begin{array}{c}\vdots\\ \Gamma \vdash A \wedge B\end{array}}{\Gamma \vdash B}\ (\wedge E_L)$$

$$\frac{\begin{array}{c}\vdots\\ \Gamma, A \vdash B\end{array}}{\Gamma \vdash A \Rightarrow B}\ (\Rightarrow I) \qquad \frac{\begin{array}{c}\vdots\\ \Gamma \vdash A \Rightarrow B\end{array} \quad \begin{array}{c}\vdots\\ \Delta \vdash A\end{array}}{\Gamma, \Delta \vdash B}\ (\Rightarrow E) \qquad \frac{\begin{array}{c}\vdots\\ \Gamma \vdash \bot\end{array}}{\Gamma \vdash A}\ (\bot E)$$

$$\frac{\begin{array}{c}\vdots\\ \Gamma \vdash A\end{array}}{\Gamma \vdash A \vee B}\ (\vee I_R) \qquad \frac{\begin{array}{c}\vdots\\ \Gamma \vdash B\end{array}}{\Gamma \vdash A \vee B}\ (\vee I_L) \qquad \frac{\begin{array}{c}\vdots\\ \Gamma \vdash A \vee B\end{array} \quad \begin{array}{c}\vdots\\ \Delta, A \vdash C\end{array} \quad \begin{array}{c}\vdots\\ \Sigma, B \vdash C\end{array}}{\Gamma, \Delta, \Sigma \vdash C}\ (\vee E)$$

Figura 4 – Dedução natural para a lógica intuicionista

Ao se comparar com as definições providas por Troelstra e Schwichtenberg (2000), se torna evidente a ausência das regras de "para todo"($\forall$) e "existe"($\exists$). Isso se deve ao escopo deste projeto, já que optou-se por trabalhar somente com o fragmento proposicional das lógicas abordadas e, portanto, nenhum dos operadores de primeira ordem serão utilizados.

Outra diferença é a utilização explícita de um conjunto de conjunções para expressar as fórmulas presentes no contexto junto ao símbolo de derivação. Aqui, a notação $\Gamma \vdash A$ indica que, a partir de uma ou mais fórmulas presentes em Γ, é possível provar A.

Essa mudança foi feita inicialmente a partir de um ponto de vista didático, pois isso facilitará no entendimento das provas da tradução da lógica intuicionista para a lógica linear, já que esse método de construção das regras da dedução natural as torna muito similar aos sequentes utilizados no sistema dedutivo da lógica linear. Entretanto, essa mudança acaba se tornando extremamente relevante para a construção da tradução, pois a partir dela podemos criar uma prova que se aplique para casos que possuam um conjunto qualquer em seu contexto, o que claramente torna a prova muito mais significativa.

$$\frac{\begin{array}{c}\vdots\\ \Gamma \vdash\end{array}}{\Gamma \vdash A}\ (wk) \qquad \frac{\begin{array}{c}\vdots\\ \Gamma \vdash A, A\end{array}}{\Gamma \vdash A}\ (co)$$

$$(weakening) \qquad (contraction)$$

Figura 5 – Regras de *weakening* e *contraction* para dedução natural

Por fim, a dedução natural possui diversas regras estruturais que podem ser utilizadas para auxiliar na construção das provas. Porém neste momento, vale a pena citar a regra de *weakening* e a de *contraction* (5), que tanto para a lógica intuicionista como para a clássica são implícitas junto com o sistema dedutivo. Aqui, diferente do que acontece na lógica linear, estas regras não possuem nenhum limite de uso atrelados a elas e normalmente só são usadas de forma explícita em um prova para melhorar o entendimento da mesma (GIRARD, 1987).

2.3.1 Dedução Natural para Lógica Clássica

Como já apresentou-se neste capítulo, a lógica clássica compartilha diversas características com a lógica intuicionista, incluindo as regras de dedução natural que são definidas pela mesma. Entretanto, somente as regras já definidas não são o suficientes para compor um sistema dedutivo para a lógica clássica (TROELSTRA; SCHWICHTENBERG, 2000).

$$\frac{\begin{array}{c}\vdots\\ \Gamma, \neg A \vdash \bot\end{array}}{\Gamma \vdash A}\ (\neg E) \qquad \frac{}{\vdash A \vee \neg A}\ (\mathit{magic})$$

Figura 6 – Regras de dedução natural adicionais para a lógica clássica

Portanto, a Figura 6 apresenta duas novas regras que podem ser adicionadas ao sistema dedutivo. Basta adicionar uma destas regras nas já definidas para a dedução natural intuicionista que se obtém um sistema dedutivo que pode representar provas para a lógica clássica. Note que, ambas as regras definidas acima são propriedades da lógica clássica que a lógica intuicionista ativamente evita implementar, sendo que, a primeira dessas regras representa a eliminação da dupla negação ($\neg E$), enquanto que a segunda regra (*magic*) representa a incorporação do terceiro excluído (TROELSTRA; SCHWICHTENBERG, 2000).

3 LÓGICA LINEAR

A lógica linear é um produto da lógica clássica e da lógica intuicionista, criada com o intuito de refinar conceitos propostos por ambas, sem deixar de atingir o nível de representação das mesmas. Desde quando foi inicialmente proposta em 1987, como uma lógica capaz de representar certos aspectos de sistemas concorrentes (GIRARD, 1987), novos usos para a lógica linear foram encontrados em áreas extremamente diversas, como na Biologia Molecular (CHAUDHURI; DESPEYROUX, 2013), Computação tradicional (ALEXIEV, 1994), Linguística (MOOT; PIAZZA, 2001) e Computação Quântica (LAGO; FAGGIAN, 2012).

O principal objetivo da lógica linear é trazer para suas regras uma verossimilhança maior na forma com que as regras da lógica clássica funcionam em situações reais. Para tanto, em vez de manter o conceito de que fórmulas são verdades ou provas, como acontece respectivamente na lógica clássica e na lógica intuicionista, a lógica linear passa a tratar as fórmulas como recursos (COSMO; MILLER, 2019).

Assim, se faz necessária a adição de causalidade em suas regras, o que permite que a lógica linear emule a utilização de recursos e seu consumo (GIRARD, 2005). Para alcançar tal feito, a lógica linear como descrita por Girard (1987), restringe o uso das regras *weakening* e *contraction* como presentes na lógica clássica, e modifica a regra de implicação para que ela não permita uma nova iteração dependendo de como seu contexto foi modificado.

Este conceito pode ser mais facilmente entendido através do funcionamento de uma Máquina de Vendas. Supondo que em tal máquina pode ser inserido o valor de um real e ela irá lhe fornecer uma bala, utilizando a lógica linear é possível modelar este sistema da seguinte forma:

- Assumindo A como a ação de gastar um real.
- Assumindo B como a obtenção de uma bala.
- A fórmula $A \multimap B$ representa o funcionamento da máquina.

Assim, se um contexto possuir as seguintes premissas A e $A \multimap B$, é possível modificar permanentemente estas premissas de forma a se obter B. Dessa forma, fica explícito o fato de que o recurso um real foi consumido para que o recurso bala pudesse ser produzido, o que evidencia o foco da lógica linear em tratar suas fórmulas como recursos.

Entretanto, após entender o funcionamento da implicação linear e seu consumo de recursos, um grande problema surge na proposta inicial da lógica linear: a de expandir a lógica clássica sem perder a expressividade da mesma. Portanto, neste capítulo será explicado de forma mais detalhada o funcionamento da lógica linear e os operadores que são utilizados para suprir as disparidades com a lógica clássica. A Seção 3.1, apresenta a gramática da lógica linear, introduz os operadores exponenciais e explica: o funcionamento dos operadores binários, a implementação do operador dual e a construção da implicação linear. A Seção 3.2 introduz o cálculo

de sequentes e explica o funcionamento do mesmo na lógica linear, além de exemplificar as diferenças dos sequentes com seu equivalente clássico e definir regras de De Morgan para a negação linear. A Seção 3.3, demonstra como expressar equivalências na lógica linear, prova a existência das propriedades de comutação e distributividade e explica de forma mais detalhada o funcionamento dos exponenciais. Por fim, a Seção 3.4 traz uma das formas possíveis de se interpretar as fórmulas da lógica linear e apresenta um exemplo de como modelar uma máquina real utilizando a lógica.

3.1 LINGUAGEM

Uma lógica normalmente possui uma linguagem formal, a qual é composta por símbolos e um determinado meio de se construir fórmulas desta lógica a partir da utilização desses símbolos (SHAPIRO; KISSEL, 2021). Tendo isso em mente, a Figura 7 define a gramática para a linguagem da lógica linear, nela se assume F como uma fórmula qualquer que pertence a lógica linear.

$$
\begin{array}{rcl}
F & ::= & F \mid F^{\perp} \\
 & \mid & 0 \mid 1 \mid \perp \mid \top \\
 & \mid & F \otimes F \mid F \,\&\, F \\
 & \mid & F \oplus F \mid F \parr F \\
 & \mid & ?F \mid !F
\end{array}
$$

Figura 7 – Gramática da Lógica Linear

Para implementar a casualidade nas fórmulas presentes na lógica linear, se faz necessária a adição de diversos novos símbolos e operadores à lógica. Inicialmente os que mais se destacam são os dois operadores de conjunção e dois operadores de disjunção. O primeiro destes operadores é a conjunção multiplicativa (*tensor*), também representado pelo símbolo $\otimes$ e que possui 1 como identidade, indica a viabilidade de executar ambas as ações. Relembrando o exemplo da máquina de vendas utilizado na seção anterior, se o preço de uma bala fosse de dois reais, é possível remodelar a máquina para que seu funcionamento seja descrito como $A \otimes A \multimap B$. Graças ao operador *tensor* é possível indicar a obrigatoriedade de se gastar duas moedas de um real para se obter uma bala.

O próximo operador presente na lógica linear é a conjunção aditiva (*with*), representado pelo símbolo & e que possui $\top$ como identidade, indica a viabilidade de somente uma das ações, a qual será escolhida por nós. Utilizando mais uma vez o exemplo da máquina de vendas, podemos definir uma nova máquina da seguinte forma:

- Assumindo A como a ação de gastar um real.
- Assumindo B como a obtenção de uma bala.
- Assumindo C como a obtenção de um suco.

- A fórmula $A \multimap B \& C$, representa o funcionamento da máquina.

Neste caso, a máquina ao receber um real entregará uma bala ou suco, porém nunca ambos, e a decisão de qual produto será entregue cabe ao consumidor.

A disjunção aditiva (*plus*), representado pelo símbolo $\oplus$ e que possui 0 como identidade, indica de forma semelhante a viabilidade de somente uma das ações, entretanto não é possível escolher o resultado. Como exemplo podemos utilizar o funcionamento de uma máquina caça-níqueis, onde após o depósito de uma certa quantia, a máquina irá decidir de forma arbitrária entre a vitória ou derrota. Este sistema pode ser modelado na lógica linear como:

- Assumindo A como a ação de gastar um real.
- Assumindo V como a ação de ganhar.
- Assumindo D como a ação de perder.
- A fórmula $A \multimap V \oplus D$, representa o funcionamento da máquina.

Por fim a disjunção multiplicativa (*par*), representado pelo símbolo $\parr$ e que possui $\bot$ como identidade, indica a viabilidade de ambas as ações, porém nunca ambas ao mesmo tempo. Para entender melhor este operador, podemos modificar levemente o sistema da máquina de vendas de forma que:

- Assumindo A como a ação de NÃO gastar um real.
- Assumindo B como a obtenção de uma bala.
- A fórmula $A \parr B$, representa o funcionamento da máquina.

Neste modelo, é possível observar que a máquina permite que a bala ou o dinheiro sejam mantidos pela pessoa, porém nunca ambos ao mesmo tempo. Assim, a partir do momento que a ação A deixa de existir no contexto, ou seja, uma pessoa gasta um real, imediatamente a ação B se faz presente.

Observe que para exemplificar a definição do operador acima, utilizou-se a negação de uma fórmula que havia sido utilizada em um exemplo anterior. Para facilitar esta representação, é feita a adição de um conectivo na lógica linear chamado de negação linear, o qual é representado por $(.)^{\bot}$, e que permite a definição do dual de uma determinada fórmula. Por exemplo, se A é a ação de gastar um real, então $A^{\bot}$ é a ação de NÃO gastar um real.

Conhecendo a definição da negação linear se torna claro que a implicação linear não é necessariamente uma operação inerente à lógica, como pode ser visto abaixo, e sim uma fórmula obtida através da utilização do operador de disjunção multiplicativo, fazendo com que não seja necessária a adição explícita deste operador à lógica linear.

$$A \multimap B \equiv A^{\bot} \parr B \tag{1}$$

Por fim, para manter a expressividade da lógica linear no mesmo nível da intuicionista e clássica, dois operadores, chamados de exponenciais (representados por ! e ?), são adicionados à lógica. Juntos, eles permitem a reiteração de ações as quais estes conectivos estão associados, permitindo o controle da aplicação das regras de *weakening* e *contraction* dentro da lógica linear.

3.2 SISTEMA DEDUTIVO

Como descrito por Shapiro e Kissel (2021), um sistema dedutivo é criado com o intuito de verificar a validade de argumentos e realizar a prova de teoremas de uma determinada lógica. Através do mesmo é possível derivar novas fórmulas que são consequências lógicas do conjunto de fórmulas que compõem o sistema (SILVA, 2006). Existem diversos métodos que podem ser utilizados para se representar o sistema dedutivo de uma lógica, entretanto, na lógica linear, o cálculo de sequentes é o método mais aceito e difundido.

3.2.1 Cálculo de Sequentes

Proposto originalmente em 1935 como uma alternativa ao método de dedução natural, o cálculo de sequentes é um estilo de argumentação lógica, onde cada uma de suas linhas é composta por uma tautologia, a qual chamaremos de sequente.

Um sequente possui forma $\Gamma \vdash \Delta$, onde o lado esquerdo do sequente é visto como uma conjunção das fórmulas lógicas pertencentes ao conjunto Γ e o lado direito é interpretado como uma disjunção das fórmulas lógicas pertencentes ao conjunto Δ, sendo ambos estes conjuntos finitos. Através dessa expressão, o sequente indica que a partir das fórmulas presentes em Γ é possível derivá-las para obter as fórmulas em Δ.

Para se construir a prova de uma determinada fórmula, são utilizadas regras como a apresentada abaixo:

$$\frac{P_1, P_2, P_3, ..., P_n}{C}$$

A qual indica que uma conclusão C é verdade sempre que as premissas $P_1,\ P_2,\ P_3, ...,\ P_n$ são verdadeiras. Assim uma prova normalmente é constituída por múltiplas regras distintas ou não que compõem a sua construção. Entretanto, é importante notar que as construções de provas utilizando o cálculo de sequentes normalmente se dá de baixo para cima (*bottom-up*), ou seja, a partir da fórmula que se deseja provar são aplicadas as regras do sistema, para que assim se descubra como a fórmula foi criada e, por consequência, sua validade.

3.2.2 Cálculo de Sequentes para a Lógica Linear

Antes de apresentar as regras dos sequentes para a lógica linear, é importante entender o funcionamento do operador de negação linear. Através dele, é possível definir o dual de qual-

quer fórmula da lógica, e por ser um operador involutivo (ou seja, a fórmula $((A)^{\perp})^{\perp}$ pode ser identificada simplesmente como A), é possível a construção de regras para a lógica linear semelhante às regras de De Morgan presentes na lógica clássica. O dual dos operadores da lógica linear e, por consequência, as regras de De Morgan que estão vinculadas com os mesmos são apresentados na Figura 8.

$$\begin{array}{cc} 1^{\perp} := \perp & \perp^{\perp} := 1 \\ \top^{\perp} := 0 & 0^{\perp} := \top \\ (A \otimes B)^{\perp} := A^{\perp} \parr B^{\perp} & (A \parr B)^{\perp} := A^{\perp} \otimes B^{\perp} \\ (A \& B)^{\perp} := A^{\perp} \oplus B^{\perp} & (A \oplus B)^{\perp} := A^{\perp} \& B^{\perp} \\ (!A)^{\perp} := ?A^{\perp} & (?A)^{\perp} := !A^{\perp} \end{array}$$

Figura 8 – Negação Linear

Para facilitar a definição das regras do cálculo de sequentes da lógica linear, as regras de derivação à esquerda serão, em sua maioria, omitidas. Isso se torna possível graças às regras de derivação da negação linear, que permitem que uma fórmula à esquerda da derivação seja reescrita à direita como seu dual. Assim, se define as regras do cálculo de sequentes para a lógica linear como apresentado na Figura 9.

Note que, através da regra *par* é possível concluir que as vírgulas que separam as diferentes fórmulas de um sequente têm a mesma função que o operador de disjunção multiplicativa. Através disso, podemos, por exemplo, construir uma das tautologias mais simples da lógica clássica que também se aplica a lógica linear, a regra $A \multimap A$. Demonstrando de forma efetiva a criação de uma regra de sequente para a implicação linear, devido à existência da equivalência entre a implicação e o operador de disjunção multiplicativa.

$$\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp}, A}\,(\text{id})}{\vdash A^{\perp} \parr A}\,(\text{par})}{\vdash A \multimap A^{\perp}}\,(\text{implication})$$

Agora, vejamos o exemplo abaixo onde, além de ser possível observar o funcionamento do cálculo de sequentes, também temos a demonstração de um dos lados das regras de De Morgan mencionadas anteriormente e da utilização da regra de negação linear para transpor uma fórmula da esquerda para a direita de um sequente.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp}, A}\,(\text{id})}{\vdash A, A^{\perp}}\,(\text{ex})}{\vdash A, A^{\perp} \oplus B^{\perp}}\,(\text{left plus}) \quad \dfrac{\dfrac{\dfrac{}{\vdash B^{\perp}, B}\,(\text{id})}{\vdash B, B^{\perp}}\,(\text{ex})}{\vdash B, A^{\perp} \oplus B^{\perp}}\,(\text{right plus})}{\vdash A \& B \ , \ A^{\perp} \oplus B^{\perp}}\,(\text{with})}{(A \& B)^{\perp} \vdash A^{\perp} \oplus B^{\perp}}\,(\text{left dual})$$

$$\frac{}{\vdash A^{\perp}, A}\ \text{(identity)} \qquad \frac{\vdash \Gamma, A \qquad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\ \text{(cut)}$$

$$\frac{\vdash \Gamma, B, A, \Delta}{\vdash \Gamma, A, B, \Delta}\ \text{(exchange)} \qquad \frac{\vdash \Gamma, A \qquad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\ \text{(cut)}$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, A^{\perp} \vdash \Delta}\ \text{(left dual)} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash A^{\perp}, \Delta}\ \text{(right dual)}$$

$$\frac{}{\vdash 1}\ \text{(one)} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\ \text{(false)} \qquad \frac{}{\vdash \Gamma, \top}\ \text{(true)}$$

$$\frac{\vdash \Gamma, A \qquad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}\ \text{(times)} \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\ \text{(par)}$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}\ \text{(left plus)}$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \& B}\ \text{(with)}$$

$$\frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\ \text{(right plus)}$$

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\ \text{(of course)} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\ \text{(weakening)}$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\ \text{(dereliction)} \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\ \text{(contraction)}$$

Figura 9 – Cálculo de Sequentes para a lógica linear

Perceba que a regra chamada *exchange* foi utilizada para modificar o sequente e tornar possível a aplicação da regra *identity*. Essa regra permite a troca de posição de fórmulas separadas por vírgula e é inerente a definição de cálculo de sequentes, já que um sequente nada mais é do que um multiconjunto de fórmulas. Além disso, essa regra recebe um reforço ainda maior na lógica linear, pois a vírgula pode ser interpretada como um operador de disjunção multiplicativa, e o mesmo é comutativo. Tal regra não necessariamente precisa ser utilizada de forma explícita, porém para um melhor entendimento das provas e para manter consistente o sentido computacional das mesmas (ver Seção 6.1), este será o método abordado.

Agora, para exemplificar algumas das propriedades já demonstradas e levantar alguns pontos importantes sobre a construção dos sequentes, vamos traçar na Figura 10 um comparativo entre um sequente para uma fórmula da lógica clássica (b) e um potencial equivalente na lógica linear (a).

A princípio a única característica que impede que a prova do sequente da esquerda seja completada é a impossibilidade de aplicar a regra *weakening* na linha "*" do sequente, assim como ocorre no que aparenta ser seu equivalente da lógica clássica. Poderíamos então concluir

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A,A}\,(*)\quad\dfrac{}{\vdash B^{\perp},B}\,(\text{id})}{\vdash A^{\perp},A,A\otimes B^{\perp},B}\,(\text{tensor})}{\vdash A^{\perp},A\otimes B^{\perp},A,B}\,(\text{ex})}{A,A^{\perp}\parr B\vdash A,B}\,(\text{L dual})}{A,A\multimap B\vdash A,B}\,(\text{implication})$$

(a)

$$\dfrac{\dfrac{\dfrac{}{A\vdash A}\,(\text{id})}{A\vdash A,A}\,(\text{w?})\quad\dfrac{}{B\vdash B}\,(\text{id})}{A,A\to B\vdash A,B}\,(\text{L}\to)$$

(b)

Figura 10 – Comparação de um sequente da lógica linear com um sequente da lógica clássica

que, basta adicionar mais uma fórmula A à esquerda do sequente para então ser possível derivar A,B. Entretanto, esta interpretação está errada, pois como foi visto anteriormente, as vírgulas à direita do sequente podem ser interpretadas como disjunções multiplicativas ($\parr$) e este operador indica a potencialidade de somente uma das regras, ao invés da obrigatoriedade de ambas, que é o que a fórmula à esquerda do sequente deveria demonstrar. Assim, o sequente abaixo utiliza o operador de conjunção multiplicativa para corretamente expressar a derivação desejada.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A}\,(\text{id})\quad\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A}\,(\text{id})\quad\dfrac{}{\vdash B^{\perp},B}\,(\text{id})}{\vdash A^{\perp},A\otimes B^{\perp},B}\,(\text{tensor})}{\vdash B,A^{\perp},A\otimes B^{\perp}}\,(\text{ex})}{\vdash A^{\perp},A\otimes B,A^{\perp},A\otimes B^{\perp}}\,(\text{tensor})}{\vdash A^{\perp},A^{\perp},A\otimes B^{\perp},A\otimes B}\,(\text{ex})}{A,A,A^{\perp}\parr B\vdash A\otimes B}\,(\text{left dual})}{A,A,A\multimap B\vdash A\otimes B}\,(\text{implication})$$

É interessante perceber que a fórmula à esquerda do sequente acima não pode derivar a fórmula $(B\otimes B)$, já que a própria possibilidade de executar uma transformação de A para B também é consumida se a mesma for efetuada. Se voltarmos a analisar o exemplo da máquina de vendas previamente proposto, é como se a quantidade de fórmulas $(Real\multimap Bala)$ existentes no contexto do funcionamento da máquina indicasse o estoque de balas que a mesma possui.

3.3 PROPRIEDADES DOS CONECTIVOS

Um sistema dedutivo é fundamental para a construção de provas sobre propriedades e teoremas de uma determinada lógica (SILVA, 2006). Portanto, tendo definido o funcionamento do cálculo de sequentes para a lógica linear, se torna possível demonstrar a validade de certas propriedades referentes à lógica. Esta seção focará em explicar e provar algumas das propriedades dos operadores da lógica linear e da implicação linear.

3.3.1 Teorema da Dedução e Equivalência

Proposto originalmente por Herbrand (1930), o teorema da dedução é um metateorema que facilita a construção de provas condicionais. Esse teorema permite que, em uma prova de uma implicação $A \to B$, se assuma A como uma hipótese e, a partir dela, se derivar B. Ou seja, para quaisquer fórmulas A, B, se $\Gamma, A \vdash B$, então $\Gamma \vdash A \to B$

Assim, como demonstrado abaixo, é possível a partir de $A \vdash B$ construir um sequente que tem como resultado $A \multimap B$, o que efetivamente prova que o metateorema da dedução também é válido para a implicação linear.

$$\dfrac{\dfrac{\dfrac{A \vdash B}{\vdash A^{\perp}, B}\ \text{(right dual)}}{\vdash A^{\perp} \parr B}\ \text{(par)}}{\vdash A \multimap B}\ \text{(implication)}$$

Tendo provado a validade deste teorema para a lógica linear, podemos utilizá-lo para facilitar a construção de diversas provas da mesma. E uma das classes de teoremas que mais se utiliza dessa propriedade é a de equivalência, pois dado duas fórmulas A, B quaisquer, podemos interpretar a equivalência entre elas como:

$$A \leftrightarrow B \equiv (A \multimap B) \& (B \multimap A)$$

Além disso, observe o início da construção de um sequente que prova essa mesma equivalência:

$$\dfrac{\vdash (A \multimap B) \quad \vdash (B \multimap A)}{\vdash (A \multimap B) \& (B \multimap A)}\ \text{(with)}$$

Fica claro que a única ação possível de ser tomada no inicio deste sequente é a aplicação da regra *with*, por consequência o mesmo se aplica para todos os outros sequentes que provam equivalências. Assim, para facilitar o entendimento e a construção dos sequentes, podemos provar cada uma das implicações separadamente, e ainda assim obter efetivamente o mesmo resultado.

Além disso, podemos utilizar o metateorema da dedução para tornar os sequentes ainda mais concisos. Assim, como demonstrado no exemplo presente na Figura 11, podemos construir a prova da fórmula $(A \otimes (B \parr C)) \multimap ((A \otimes B) \parr C)$, a qual é chamada de distribuição linear.

3.3.2 Comutatividade e Distributividade

Entre todas as propriedades dos operadores da lógica linear, uma das mais simples e fundamentais é a comutatividade, a qual dita que trocar a ordem dos operandos em uma operação binária não altera o resultado da mesma. Como provado na Figura 12, todos os operadores binários da lógica linear são comutativos.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A}\ (\text{identity})\quad \dfrac{\dfrac{}{\vdash B,B^{\perp}}\ (\text{identity})\quad \dfrac{}{\vdash C^{\perp},C}\ (\text{identity})}{\vdash B,(B^{\perp}\otimes C^{\perp}),C}\ (\text{tensor})}{\vdash A^{\perp},(A\otimes B),(B^{\perp}\otimes C^{\perp}),C}\ (\text{tensor})}{\vdash A^{\perp},(B^{\perp}\otimes C^{\perp}),(A\otimes B),C}\ (\text{exchange})}{\vdash A^{\perp},(B^{\perp}\otimes C^{\perp}),(A\otimes B)\parr C}\ (\text{par})}{\vdash A^{\perp}\parr(B^{\perp}\otimes C^{\perp}),(A\otimes B)\parr C}\ (\text{par})}{(A\otimes(B\parr C))\vdash((A\otimes B)\parr C)}\ (\text{left dual})$$

Figura 11 – Distribuição Linear

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B^{\perp},B}\ (\text{identity})\quad \dfrac{}{\vdash A,A^{\perp}}\ (\text{identity})}{\vdash B^{\perp},B\otimes A,A^{\perp}}\ (\text{tensor})}{\vdash A^{\perp},B^{\perp},B\otimes A}\ (\text{exchange})}{\vdash A^{\perp}\parr B^{\perp},B\otimes A}\ (\text{par})}{A\otimes B\vdash B\otimes A}\ (\text{left dual})$$

$$\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A}\ (\text{identity})}{\vdash A^{\perp},B\oplus A}\ (\text{right plus})\quad \dfrac{\dfrac{}{\vdash B^{\perp},B}\ (\text{identity})}{\vdash B^{\perp},B\oplus A}\ (\text{left plus})}{\vdash A^{\perp}\&B^{\perp},B\oplus A}\ (\text{with})}{A\oplus B\vdash B\oplus A}\ (\text{left dual})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A,A^{\perp}}\ (\text{identity})\quad \dfrac{}{\vdash B^{\perp},B}\ (\text{identity})}{\vdash A,A^{\perp}\otimes B^{\perp},B}\ (\text{tensor})}{\vdash A^{\perp}\otimes B^{\perp},B,A}\ (\text{exchange})}{\vdash A^{\perp}\otimes B^{\perp},B\parr A}\ (\text{par})}{A\parr B\vdash B\parr A}\ (\text{left dual})$$

$$\dfrac{\dfrac{\dfrac{}{\vdash B^{\perp},B}\ (\text{identity})}{\vdash A^{\perp}\oplus B^{\perp},B}\ (\text{right plus})\quad \dfrac{\dfrac{}{\vdash A^{\perp},A}\ (\text{identity})}{\vdash A^{\perp}\oplus B^{\perp},A}\ (\text{left plus})}{\vdash A^{\perp}\oplus B^{\perp},B\&A}\ (\text{with})}{A\&B\vdash B\&A}\ (\text{left dual})$$

Figura 12 – Comutatividade dos Operadores

Em seguida temos a distributividade, propriedade que assim como a comutatividade só atua sobre operadores binários. Na lógica linear há somente duas relações de distributividade presentes entre seus operadores. A primeira delas é da conjunção multiplicativa sobre a disjunção aditiva, como demonstrado na fórmula a seguir:

$$A\otimes(B\oplus C)\equiv(A\otimes B)\oplus(A\otimes C)$$

É importante ressaltar que o oposto, ou seja, a distributividade da disjunção aditiva sobre a conjunção multiplicativa, não é possível. Isso se torna claro se analisarmos a seguinte fórmula: $(A\oplus(B\otimes C))$, e aplicarmos nela a distributividade de forma errônea. Essa fórmula descreve que uma escolha entre A ou $(B\otimes C)$ deve ser feita, e coloca ênfase em B e C serem escolhidos obrigatoriamente juntos. Assim, ao aplicar a distributividade de forma incorreta, obtém-se $((A\otimes B)\oplus(A\otimes C))$, uma fórmula que distorce completamente o significado de seu contra-partido original, pois a opção de escolha entre A ou B junto a C, foi removida do contexto.

A Figura 13 demonstra a prova desta distributividade, note que a prova desta equivalência está dividida em duas partes para facilitar a leitura e entendimento da mesma.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A}\,(\text{identity})\quad\dfrac{}{\vdash B,B^{\perp}}\,(\text{identity})}{\vdash A^{\perp},(A\otimes B),B^{\perp}}\,(\text{tensor})}{\vdash A^{\perp},B^{\perp},(A\otimes B)}\,(\text{exchange})}{\vdash A^{\perp},B^{\perp},(A\otimes B)\oplus(A\otimes C)}\,(\text{left plus})\quad\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A}\,(\text{identity})\quad\dfrac{}{\vdash C,C^{\perp}}\,(\text{identity})}{\vdash A^{\perp},(A\otimes C),C^{\perp}}\,(\text{tensor})}{\vdash A^{\perp},C^{\perp},(A\otimes C)}\,(\text{exchange})}{\vdash A^{\perp},C^{\perp},(A\otimes B)\oplus(A\otimes C)}\,(\text{right plus})}{\vdash A^{\perp},B^{\perp}\&C^{\perp},(A\otimes B)\oplus(A\otimes C)}\,(\text{with})$$

$$\dfrac{\dfrac{\vdash A^{\perp}\parr(B^{\perp}\&C^{\perp}),(A\otimes B)\oplus(A\otimes C)}{A\otimes(B\oplus C)\vdash(A\otimes B)\oplus(A\otimes C)}\,(\text{left dual})}{}\,(\text{par})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A}\,(\text{identity})\quad\dfrac{\dfrac{}{B,B^{\perp}}\,(\text{identity})}{B\oplus C,B^{\perp}}\,(\text{left plus})}{\vdash A^{\perp},A\otimes(B\oplus C),B^{\perp}}\,(\text{tensor})}{\vdash A^{\perp},B^{\perp},A\otimes(B\oplus C)}\,(\text{exchange})}{\vdash A^{\perp}\parr B^{\perp},A\otimes(B\oplus C)}\,(\text{par})\quad\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A}\,(\text{identity})\quad\dfrac{\dfrac{}{C,C^{\perp}}\,(\text{identity})}{B\oplus C,C^{\perp}}\,(\text{left plus})}{\vdash A^{\perp},A\otimes(B\oplus C),C^{\perp}}\,(\text{tensor})}{\vdash A^{\perp},C^{\perp},A\otimes(B\oplus C)}\,(\text{exchange})}{\vdash A^{\perp}\parr C^{\perp},A\otimes(B\oplus C)}\,(\text{par})}{\vdash(A^{\perp}\parr B^{\perp})\&(A^{\perp}\parr C^{\perp}),A\otimes(B\oplus C)}\,(\text{with})$$

$$\dfrac{\vdash(A^{\perp}\parr B^{\perp})\&(A^{\perp}\parr C^{\perp}),A\otimes(B\oplus C)}{(A\otimes B)\oplus(A\otimes C)\vdash A\otimes(B\oplus C)}\,(\text{left dual})$$

Figura 13 – Distributividade da conjunção multiplicativa sobre a disjunção aditiva

A segunda regra se refere à distributividade da disjunção multiplicativa sobre a conjunção aditiva, como demonstrado na fórmula abaixo.

$$A\parr(B\&C)\equiv(A\parr B)\&(A\parr C)$$

Assim como a primeira regra de distributividade, não é possível distribuir a conjunção aditiva sobre disjunção multiplicativa. De forma ainda semelhante à regra anterior, essa distribuição não funciona pois, se aplicada, ela muda completamente o sentido da fórmula original. Finalmente, a prova da distributividade, apresentada na Figura 14, também está divida em duas partes.

A última regra demonstrada ainda pode ser expandida para denotar uma regra de distributividade para a implicação linear, da seguinte forma:

$$A\multimap(B\&C)\equiv(A\multimap B)\&(A\multimap C)$$

A prova desta regra não será demonstrada pois, se relembrarmos que a definição do operador em questão é definida pela Fórmula 1, torna-se claro que a partir de uma aplicação direta dessa definição do operador se obtém a regra citada acima.

3.3.3 Exponenciais

Como mencionado anteriormente, as regras de exponenciais são utilizadas para emular o funcionamento da lógica clássica na lógica linear. Para alcançar tal feito, existem três regras para

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}\ \text{(identity)}\quad \dfrac{\dfrac{}{\vdash B^{\perp}, B}\ \text{(identity)}}{\vdash B^{\perp} \oplus C^{\perp}, B}\ \text{(left plus)}}{\vdash A, A^{\perp} \otimes (B^{\perp} \oplus C^{\perp}), B}\ \text{(tensor)}}{\vdash A^{\perp} \otimes (B^{\perp} \oplus C^{\perp}), A, B}\ \text{(exchange)}}{\vdash A^{\perp} \otimes (B^{\perp} \oplus C^{\perp}), A \parr B}\ \text{(par)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}\ \text{(identity)}\quad \dfrac{\dfrac{}{\vdash C^{\perp}, C}\ \text{(identity)}}{\vdash (B^{\perp} \oplus C^{\perp}), C}\ \text{(right plus)}}{\vdash A, A^{\perp} \otimes (B^{\perp} \oplus C^{\perp}), C}\ \text{(tensor)}}{\vdash A^{\perp} \otimes (B^{\perp} \oplus C^{\perp}), A, C}\ \text{(exchange)}}{\vdash A^{\perp} \otimes (B^{\perp} \oplus C^{\perp}), A \parr C}\ \text{(par)}}{\vdash A^{\perp} \otimes (B^{\perp} \oplus C^{\perp}), (A \parr B) \& (A \parr C)}\ \text{(with)}}{A \parr (B \& C) \vdash (A \parr B) \& (A \parr C)}\ \text{(left dual)}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}\ \text{(identity)}\quad \dfrac{}{\vdash B^{\perp}, B}\ \text{(identity)}}{\vdash A, (A^{\perp} \otimes B^{\perp}), B}\ \text{(tensor)}}{\vdash (A^{\perp} \otimes B^{\perp}), A, B}\ \text{(exchange)}}{\vdash (A^{\perp} \otimes B^{\perp}) \oplus (A^{\perp} \otimes C^{\perp}), A, B}\ \text{(left plus)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}\ \text{(identity)}\quad \dfrac{}{\vdash C^{\perp}, C}\ \text{(identity)}}{\vdash A, (A^{\perp} \otimes C^{\perp}), C}\ \text{(tensor)}}{\vdash (A^{\perp} \otimes C^{\perp}), A, C}\ \text{(exchange)}}{\vdash (A^{\perp} \otimes B^{\perp}) \oplus (A^{\perp} \otimes C^{\perp}), A, C}\ \text{(right plus)}}{\vdash (A^{\perp} \otimes B^{\perp}) \oplus (A^{\perp} \otimes C^{\perp}), A, (B \& C)}\ \text{(with)}}{\vdash (A^{\perp} \otimes B^{\perp}) \oplus (A^{\perp} \otimes C^{\perp}), A \parr (B \& C)}\ \text{(par)}}{(A \parr B) \& (A \parr C) \vdash A \parr (B \& C)}\ \text{(left dual)}$$

Figura 14 – Distributividade da disjunção multiplicativa sobre a conjunção aditiva

o operador "?" e uma regra para o operador "!" dentro do conjunto de sequentes apresentados. As regras: *weakening*, *dereliction* e *contraction*, são utilizadas para representar a potencialidade de se utilizar uma determinada fórmula infindáveis vezes. Por outro lado, a regra *of course*, por mencionar que seu contexto deve ser ?Γ, pode ser interpretada como a obrigatoriedade de uma determinada fórmula possuir recursos infinitos em seu contexto para poder ser provada.

Para melhor compreender o funcionamento dessas regras, podemos como exemplo, compará-las com o funcionamento de uma copiadora. Enquanto uma fórmula $!A$, e por consequência a regra *of course*, representa a capacidade de criar cópias sem restrições, as outras três regras realizam efetivamente esta potencialidade ao simular as seguintes ações: apagar uma cópia (*weakening*), criar uma única cópia (*dereliction*) e a duplicando a própria máquina (*contraction*).

Para contextualizar estar propriedades, podemos modelar o funcionamento de uma máquina montadora de lápis. Esta máquina recebe grafite e madeira (representados respectivamente por G, M) e as transforma em um lápis (representado por L). A princípio a ação executada por esta máquina poderia ser representada pela fórmula: $((G \otimes M) \multimap L)$. Entretanto, por mais que a máquina tenha um limite de lápis que a mesma possa montar baseado em sua expectativa de vida, não há como efetivamente mensurar um limite de produção para a máquina. Assim, podemos descrever o funcionamento da máquina como $!((G \otimes M) \multimap L)$, efetivamente removendo o limite de montagens que podem ser feitas pela mesma.

Tendo esclarecido o funcionamento dos exponenciais, podemos analisar a prova de dois isomorfismos pertinentes a estes operadores, como apresentado nas Figuras 15 e 16.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A}\,(\text{identity})}{\vdash A^{\perp}\oplus B^{\perp},A}\,(\text{left plus})}{\vdash ?(A^{\perp}\oplus B^{\perp}),A}\,(\text{dereliction})}{\vdash ?(A^{\perp}\oplus B^{\perp}),!A}\,(\text{of course}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B,B^{\perp}}\,(\text{identity})}{\vdash B,A^{\perp}\oplus B^{\perp}}\,(\text{right plus})}{\vdash B,?(A^{\perp}\oplus B^{\perp})}\,(\text{dereliction})}{\vdash !B,?(A^{\perp}\oplus B^{\perp})}\,(\text{of course})}{\dfrac{\dfrac{\dfrac{\vdash ?(A^{\perp}\oplus B^{\perp}),!A\otimes !B,?(A^{\perp}\oplus B^{\perp})}{\vdash ?(A^{\perp}\oplus B^{\perp}),?(A^{\perp}\oplus B^{\perp}),!A\otimes !B}\,(\text{exchange})}{\vdash ?(A^{\perp}\oplus B^{\perp}),!A\otimes !B}\,(\text{contraction})}{!(A\&B)\vdash !A\otimes !B}\,(\text{left dual})}\,(\text{tensor})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A}\,(\text{identity})}{\vdash ?A^{\perp},A}\,(\text{dereliction})}{\vdash ?A^{\perp},?B^{\perp},A}\,(\text{weakening}) \qquad \dfrac{\dfrac{\dfrac{}{\vdash B^{\perp},B}\,(\text{identity})}{\vdash ?B^{\perp},B}\,(\text{dereliction})}{\vdash ?A^{\perp},?B^{\perp},B}\,(\text{weakening})}{\vdash ?A^{\perp},?B^{\perp},(A\&B)}\,(\text{with})}{\dfrac{\dfrac{\vdash ?A^{\perp},?B^{\perp},!(A\&B)}{\vdash ?A^{\perp}\parr ?B^{\perp},!(A\&B)}\,(\text{par})}{!A\otimes !B\vdash !(A\&B)}\,(\text{left dual})}\,(\text{of course})$$

Figura 15 – Isomorfismo do exponencial "!"

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp},A}\,(\text{identity})}{\vdash A^{\perp},?A}\,(\text{dereliction})}{\vdash A^{\perp},?A,?B}\,(\text{weakening}) \qquad \dfrac{\dfrac{\dfrac{}{\vdash B^{\perp},B}\,(\text{weakening})}{\vdash B^{\perp},?B}\,(\text{dereliction})}{\vdash B^{\perp},?A,?B}\,(\text{weakening})}{\vdash A^{\perp}\&B^{\perp},?A,?B}\,(\text{with})}{\dfrac{\dfrac{\vdash !(A^{\perp}\&B^{\perp}),?A,?B}{\vdash !(A^{\perp}\&B^{\perp}),?A\parr ?B}\,(\text{par})}{?(A\oplus B)\vdash ?A\parr ?B}\,(\text{left dual})}\,(\text{of course})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A,A^{\perp}}\,(\text{identity})}{\vdash A\oplus B,A^{\perp}}\,(\text{left plus})}{\vdash ?(A\oplus B),A^{\perp}}\,(\text{dereliction})}{\vdash ?(A\oplus B),!A^{\perp}}\,(\text{of course}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B^{\perp},B}\,(\text{identity})}{\vdash B^{\perp},A\oplus B}\,(\text{right plus})}{\vdash B^{\perp},?(A\oplus B)}\,(\text{dereliction})}{\vdash !B^{\perp},?(A\oplus B)}\,(\text{of course})}{\dfrac{\dfrac{\dfrac{\vdash ?(A\oplus B),!A^{\perp}\otimes !B^{\perp},?(A\oplus B)}{\vdash !A^{\perp}\otimes !B^{\perp},?(A\oplus B),?(A\oplus B)}\,(\text{exchange})}{\vdash !A^{\perp}\otimes !B^{\perp},?(A\oplus B)}\,(\text{contraction})}{?A\parr ?B\vdash ?(A\oplus B)}\,(\text{left dual})}\,(\text{tensor})$$

Figura 16 – Isomorfismo do exponencial "?"

3.4 CODIFICAÇÃO DE ESTADOS

A característica que as fórmulas na lógica linear têm de representar recursos as torna um ótimo meio para caracterizar estados de um sistema, ou seja, permite que represente premissas que em certo momento do sistema sejam verdade, porém após alguma transição as mesmas se tornam falsas. Esta característica é naturalmente introduzida a lógica através do funcionamento do operador implicação linear. Por exemplo, podemos descrever um determinado estado de um contexto como $\pi_1 : A, !(A \multimap B)$ e, após o consumo de A pela implicação, temos como resultado um novo estado $\pi_2 :!(A \multimap B), B$.

Vejamos com um outro exemplo, a partir da máquina de estados finitos representada na Figura 17, onde há dois possíveis estados que uma máquina hipotética pode assumir, ou ligado (*On*) ou desligado (*Off*).

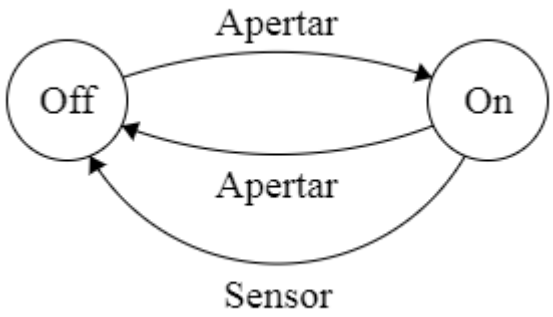

Figura 17 – Modelagem das ações de Liga/Desliga de uma máquina

Ao analisar esta máquina de estados, e considerando que $A = Apertar$, $O = On$, $F = Off$ e $S = Sensor$, podemos obter mais alguns detalhes sobre o sistema que a mesma representa:

- Se a máquina estiver desligada e um botão for apertado, o estado da máquina então muda para ligado. Esta transição pode ser representada como:

$$T_1 = (F \otimes A) \multimap O$$

- Da mesma forma, se a máquina estiver ligada e o mesmo botão for pressionado, o estado da máquina então muda para desligado. Esta transição pode ser representada por:

$$T_2 = (O \otimes A) \multimap F$$

- Por fim, a máquina ainda possui um sensor que pode detectar uma falha no sistema, desligando-a caso isso ocorra. Esta transição pode ser representada por:

$$T_3 = (O \otimes S) \multimap F$$

Como estas transições podem ocorrer indefinidamente, e considerando que a análise da máquina tem início com a mesma desligada, nós podemos representar seu estado inicial através do seguinte sequente:

$$\pi_{off} = F, !T1, !T2, !T3 \vdash$$

Tendo esta representação, podemos demonstrar que ao apertar o botão a máquina mudará para um novo estado $\pi_{on} = (O, !T_1, !T_2, !T_3)$, como pode ser visto na prova abaixo:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp}, A}\,(\text{id}) \quad \dfrac{}{\vdash F, F^{\perp}}\,(\text{id})}{\vdash A^{\perp}, A * F, F^{\perp}}\,(\text{tensor})}{\vdash A^{\perp}, F^{\perp}, A * F}\,(\text{ex}) \quad \dfrac{}{\vdash *O^{\perp}, O}\,(\text{id})}{\vdash A^{\perp}, F^{\perp}, (A * F) * O^{\perp}, O}\,(\text{tensor})}{\vdash A^{\perp}, F^{\perp}, ?((A * F) * O^{\perp}), O}\,(\text{d?})}{\vdash A^{\perp}, F^{\perp}, ?T_1^{\perp}, O}\,(\text{T}_1\text{ def}) \quad \dfrac{\dfrac{\dfrac{}{\vdash T_1, T_1^{\perp}}\,(\text{id})}{\vdash T_1, ?T_1^{\perp}}\,(\text{d?})}{\vdash !T_1, ?T_1^{\perp}}\,(\text{o!})}{\vdash A^{\perp}, F^{\perp}, ?T_1^{\perp}, (O \otimes !T_1), ?T_1^{\perp}}\,(\text{tensor}) \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash T_2^{\perp}, T_2}\,(\text{id})}{\vdash ?T_2^{\perp}, T_2}\,(\text{d?})}{\vdash ?T_2^{\perp}, !T_2}\,(\text{o!}) \quad \dfrac{\dfrac{\dfrac{}{\vdash T_3, T_3^{\perp}}\,(\text{id})}{\vdash T_3, ?T_3^{\perp}}\,(\text{d?})}{\vdash !T_3, ?T_3^{\perp}}\,(\text{o!})}{\vdash ?T_2^{\perp}, (!T_2 \otimes !T_3), ?T_3^{\perp}}\,(\text{tensor})}{\vdash (!T_2 \otimes !T_3), ?T_2^{\perp}, ?T_3^{\perp}}\,(\text{ex})}{\vdash A^{\perp}, F^{\perp}, ?T_1^{\perp}, (O \otimes !T_1), ?T_1^{\perp}}\,(\text{ex})}{\vdash A^{\perp}, F^{\perp}, ?T_1^{\perp}, ?T_1^{\perp}, (O \otimes !T_1)}\,(\text{c?})}{\vdash A^{\perp}, F^{\perp}, ?T_1^{\perp}, (O \otimes !T_1 \otimes !T_2 \otimes !T_3), ?T_2^{\perp}, ?T_3^{\perp}}\,(\text{tensor})$$

$$\dfrac{\dfrac{\vdash A^{\perp}, F^{\perp}, ?T_1^{\perp}, ?T_2^{\perp}, ?T_3^{\perp}, (O \otimes !T_1 \otimes !T_2 \otimes !T_3)}{\vdash A^{\perp}, F^{\perp}, ?T_1^{\perp}, (O \otimes !T_1 \otimes !T_2 \otimes !T_3), ?T_2^{\perp}, ?T_3^{\perp}}\,(\text{ex})}{\vdash A^{\perp}, F^{\perp}, ?T_1^{\perp}, (O \otimes !T_1 \otimes !T_2 \otimes !T_3), ?T_2^{\perp}, ?T_3^{\perp}}\,(\text{tensor})}{A, F, !T_1, !T_2, !T_3 \vdash (O \otimes !T_1 \otimes !T_2 \otimes !T_3)}\,(\text{left dual})$$

As transições criadas pelo operador de implicação linear podem ser interpretadas como *multi-set rewriting*, e portanto, qualquer sistema computacional que utiliza este conceito é passível de ser modelado através da lógica linear. Sistemas que possuem concorrência na utilização de recursos, por exemplo, são ótimos candidatos para serem modelados utilizando a lógica linear como base, já que a mudança de alocação de recursos podem ser naturalmente interpretados como mudanças de estados (MARTÍ-OLIET; MESEGUER, 1989).

3.5 ELIMINAÇÃO DA REGRA DO CORTE

A regra do corte, como definida por Girard (1987), é uma regra estrutural explícita que faz parte do cálculo de sequentes da lógica linear. Essa regra é muito útil em alguns casos para facilitar a construção de provas.

Um exemplo de sua utilidade pode ser visto neste trabalho, pois na "simulação"das regras de *weakening* e *contraction* irrestritas da lógica clássica e intuicionista durante a sua tradução, a regra do corte é amplamente utilizada. Entretanto, vale notar que esta regra é opcional ao seu sistema dedutivo, já que qualquer prova de sequente da lógica linear que utiliza a regra

do corte pode ser substituída por uma prova equivalente que não utilize esta regra (XAVIER, 2017).

O teorema da eliminação da regra do corte fornece duas propriedades muito importantes para a lógica linear. A primeira é a garantia de que as fórmulas das premissas sempre serão sub fórmulas da conclusão. A segunda, e mais importante, é a remoção da ambiguidade gerada pela regra do corte. Como a lógica linear tem uma preocupação muito grande em controlar a quantidade de regras e fórmulas que estão sendo utilizadas (vide a escassez de recursos), é contra intuitivo permitir a utilização de uma regra que permite a adição irrestrita de novas fórmulas que contenham exponenciais (XAVIER, 2017).

Teorema 1. *Seja* $(\vdash \Gamma, A)$ *um sequente qualquer da lógica linear que possui uma ou mais aplicações da regra do corte em sua prova, é possível reescrever a prova para esse sequente sem a utilização da regra do corte.*

Esse teorema foi originalmente demonstrado por Girard (1987), e fazia o uso de *proof nets* na construção de sua prova. Embora pudéssemos utilizar esta mesma prova de eliminação da regra do corte com base na correspondência entre as *proof nets* e o cálculo de sequentes, existe uma prova escrita por Lincoln et al. (1992) que é nativa ao cálculo de sequentes e se encaixa melhor nas necessidades geradas pelas traduções descritas neste trabalho. Devido ao tamanho da descrição da prova deste teorema, optou-se por não demonstrá-la em sua íntegra aqui. Entretanto, uma breve descrição da prova e alguns dos casos de indução relevantes ainda serão descritos.

A prova é definida por indução através de dois parâmetros, a *altura* do sequente onde o corte é feito e sobre o *grau* do corte. Aqui, altura é definida como a posição na árvore de derivação do sequente onde o corte foi aplicado e grau do corte é definido como o número de símbolos presentes nas premissas onde o corte foi aplicado. Neste caso, também devemos definir que consideramos como símbolos todas as constantes e conectivos, bem como cada uma das proposições presentes no sequente (LINCOLN et al., 1992).

A construção da prova é feita a partir da demonstração de dois teoremas distintos. O primeiro teorema determina que, a partir da prova de um sequente que termina com uma aplicação de um corte com um grau $d > 0$, e que nenhuma das premissas deste corte possua um corte de grau maior que d, é possível construir uma prova para este sequente que termine com um corte que tenha grau menor que d.

$$\dfrac{\dfrac{\dfrac{\vdots}{\vdash \Gamma, A} \quad \dfrac{\vdots}{\vdash B, \Delta}}{\vdash \Gamma, A \otimes B, \Delta}\ times \quad \dfrac{\dfrac{\vdots}{\vdash \Psi, A^{\perp}, B^{\perp}}}{\vdash \Psi, A^{\perp} \parr B^{\perp}}\ par}{\vdash \Gamma, \Delta, \Psi}\ cut \quad \Rightarrow \quad \dfrac{\dfrac{\vdots}{\vdash \Gamma, A} \quad \dfrac{\dfrac{\vdots}{\vdash \Delta, B} \quad \dfrac{\vdots}{\vdash \Psi, A^{\perp}, B^{\perp}}}{\vdash \Delta, \Psi, A^{\perp}}\ cut}{\vdash \Gamma, \Delta, \Psi}\ cut$$

Figura 18 – Eliminação do corte: *tensor* e *par*

Ao analisar uma das partes da eliminação da regra do corte demonstradas na Figura 18, é visto que as mesmas premissas foram utilizadas em ambas as construções dos sequentes e que a quantidade de símbolos no corte é menor após remover o corte designado. Logo, o grau do novo sequente é menor nesta transformação e, portanto, a hipótese é valida para esta regra.

É curioso notar que a eliminação de um corte não precisa garantir que nenhum corte seja usado em seu resultado, tanto que, como demonstrado no exemplo acima, a quantidade de cortes aumentou após a eliminação do corte original. E isso é válido, pois pela indução basta gerar uma prova com grau do corte menor, para que eventualmente o corte seja completamente eliminado nas folhas da árvore de dedução do sequente (na altura máxima). E é isso que o segundo teorema auxiliar para a eliminação do corte visa provar. Ele dita que, a partir de um sequente qualquer de grau $d > 0$, é possível construir um sequente com grau menor que d. Assim, como se pode intuir a partir da Figura 19, algumas regras podem ser tidas como "base" da prova por indução, já que elas são as que de fato eliminam completamente o uso do corte (LINCOLN et al., 1992)

$$\dfrac{\dfrac{}{\vdash c, c^{\perp}}\ \text{id} \quad \dfrac{\vdots}{\vdash \Psi, c}}{\vdash c, \Psi}\ (\text{cut}) \quad \Rightarrow \quad \dfrac{\vdots}{\vdash \Psi, c}$$

Figura 19 – Eliminação do corte: *identidade*

Por fim, a prova proposta por Lincoln et al. (1992), que foi brevemente explicada neste capítulo, também permite que se utilize as regras presentes na prova indutiva para a construção de um algoritmo que transforme uma prova que utilize o corte em uma prova sem o uso da regra. Entretanto, por mais que existam diversas vantagens em sua remoção, é necessário mencionar que em alguns casos (como em aplicações do corte sobre os exponenciais) a remoção da regra do corte pode gerar uma explosão no tamanho da prova do sequente. Assim, cabe cautela na decisão se vale ou não a pena aplicar este algoritmo.

4 *TERM REWRITING* E CONFLUÊNCIA

Dentro das áreas de estudo da computação, uma de suas áreas que se intersecta com a lógica e a matemática é o estudo da forma com que fórmulas podem ser reescritas antes de serem resolvidas, a essa área é dado o nome de *Term Rewriting*. Este método tem aplicações em várias áreas da ciência da computação, incluindo raciocínio automatizado, linguagens de programação e demonstração de teoremas (BAADER; NIPKOW, 1998).

Formalmente, *Term Rewriting* é um modelo computacional que envolve a transformação de expressões, conhecidas como termos, usando regras de reescrita. A ideia é substituir determinados padrões dentro dos termos por outras expressões de acordo com regras predefinidas. Esse processo é repetido iterativamente até que não seja possível realizar mais reescritas ou quantas vezes o processo determinar necessário. Assim, ao definirmos regras de reescrita e as expressões que compõem os termos, obtemos um processo ao qual pode ser chamado de *Term Rewriting System* (BAADER; NIPKOW, 1998).

Como um exemplo, analisemos a descrição de um *Term Rewriting System* para a soma de números naturais provida por Baader e Nipkow (1998). Para tanto, inicialmente definimos o termo 0 como a constante base e s como o passo indutivo que permite a construção dos demais números. Em seguida, descrevemos as identidades que representam as regras de reescrita para a execução da soma:

$$\begin{aligned} R1: &\quad x+0 &\equiv &\quad x \\ R2: &\quad x+s(y) &\equiv &\quad s(x+y) \end{aligned}$$

Com estas regras definidas, podemos utilizá-las para efetuar o cálculo de dois números. Vejamos como isso se aplica ao calcular a soma de 1 (representado por $s(0)$) com 2 (representado por $s(s(0))$):

$$s(0)+s(s(0)) \underset{R2}{\equiv} s(s(0)+s(0)) \underset{R2}{\equiv} s(s(s(0)+0)) \underset{R1}{\equiv} s(s(s(0)))$$

Neste exemplo, todas as regras de equivalência foram aplicadas da esquerda para a direita, ou seja, a partir do momento que é encontrado uma correspondência com uma fórmula à esquerda do símbolo de equivalência, ela era substituída pela fórmula correspondente à direita da equivalência. Entretanto o processo inverso, uma substituição da direita para a esquerda, também é permitido neste caso (BAADER; NIPKOW, 1998).

Note que essa propriedade permitiria que a partir do número 3, pudéssemos construir uma soma completamente irrelevante, como: "0 + 3" ($0+s(s(s(0)))$). Assim, as regras podem ser definidas como tendo sentido único, para evitar resultados indesejados durante o processo de reescrita.

Assim como demonstrado acima, um *Term Rewriting System* pode possuir algumas propriedades estruturais em seu processo de reescrita, sendo o resultado de como suas das regras foram definidas. Uma destas propriedades é conhecida como *Termination*, ela dita que para

um *Term Rewriting System*, existe um limite implícito na aplicação das regras de reescritas em uma fórmula desse sistema, ou seja, haverá um ponto em que nenhuma regra será equivalente a fórmula reescrita (BAADER; NIPKOW, 1998). No exemplo da soma de naturais, poderia-se obter essa propriedade ao forçar com que as regras só fossem aplicadas da esquerda para a direita, como consequência disso, o comportamento notado no parágrafo anterior não poderia acontecer.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp}, A}\,(identity) \quad \dfrac{}{\vdash B, B^{\perp}}\,(identity)}{\vdash A^{\perp}, A \otimes B, B^{\perp}}\,(tensor)}{\vdash A^{\perp}, B \otimes A, B^{\perp}}\,(tensor\ comm)}{\vdash A^{\perp}, A \otimes B, B^{\perp}}\,(tensor\ comm)$$

Figura 20 – Não conformidade da lógica linear com a propriedade *Termination*

A não conformidade com a regra de *Termination* é prejudicial para a capacidade de construção automática de provas. Como visto na Figura 20, a regra de comutação pode ser realizada quantas vezes for desejado no sequente, sem que altere o resultado da prova. Infelizmente, essa é uma característica que na maioria das lógicas não pode ser eliminada, pois isso acabaria com seu nível de expressão (ROBINSON; VORONKOV, 2001).

Entretanto, há mais uma propriedade presente nos *Term Rewriting Systems* que é de extrema relevância para a construção de provas e algoritmos, e ela é conhecida como Confluência. Esta propriedade possui problemas similares a de *Termination* entretanto, como demonstrado por Nishizaki (1993), há algumas formas de se adaptar as lógicas para que elas atendam essa propriedade. Assim, este capítulo irá explicar o funcionamento da propriedade de confluência e como ela se relaciona com cada uma das lógicas apresentadas neste trabalho.

4.1 CONFLUÊNCIA

Uma possível interpretação para melhor visualizar uma série de aplicações de reescritas é aquela de uma árvore. Entretanto, isso parece contra intuitivo com o que foi apresentado até agora, já que todas as regras de reescrita geravam uma sequência linear de transformações. Mas e se fosse adicionado a um *Term rewriting system* regras que permitissem uma bifurcação na árvore de transformações, ou seja, uma fórmula que possua mais de uma regra para reescrevê-la.

Vejamos então o que isso implicaria a um sistema, através do seguinte exemplo proposto por Baader e Nipkow (1998). Inicialmente, é definido um sistema que permite a reescrita de fórmulas matemáticas. Esse *Term rewriting system* tem como termos constantes 0, 1, X, Y e possui as regras de reescrita unilaterais como descritas a seguir:

$$
\begin{array}{lccc}
R1: & D_x(X) & \Rightarrow & 1 \\
R2: & D_x(Y) & \Rightarrow & 0 \\
R3: & D_x(u+v) & \Rightarrow & D_x(u)+D_x(v) \\
R4: & D_x(u*v) & \Rightarrow & (u*D_x(v))+(D_x(u)*v) \\
R5: & u+0 & \Rightarrow & 0
\end{array}
$$

Como podemos observar na Figura 21, a reescrita da fórmula $D_x(x*x)$ tem um comportamento diferente do que foi visto até o momento. A partir da aplicação da regra $R4$, obtém-se uma fórmula em que é possível executar duas aplicações distintas da regra $R1$, e que portanto, geram fórmulas reescritas que também tão distintas. E é esse comportamento o que faz com que o processo de reescrita de um termo possa ser interpretado como uma árvore.

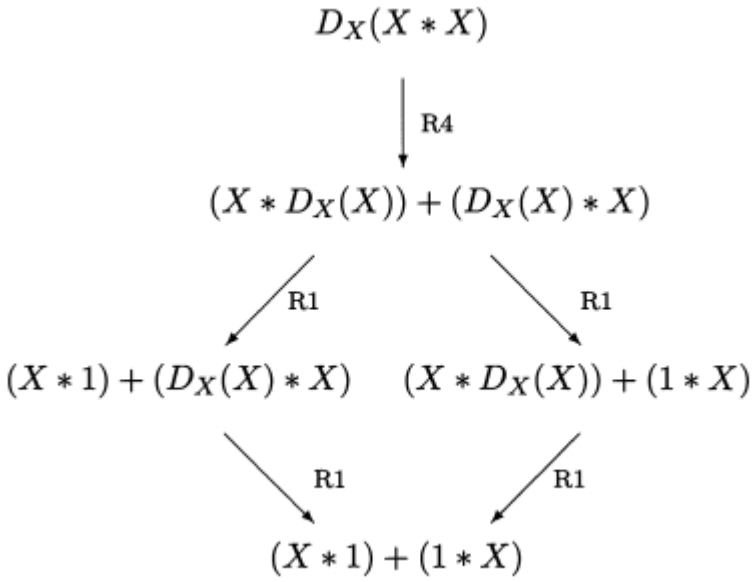

Figura 21 – Exemplo de *Term Rewriting* confluente

Note ainda que, após a divisão causada por aplicações distintas de $R1$, através da aplicação de novas regras em cada lado da árvore, foi possível gerar uma fórmula que unificasse novamente a árvore em um único nó e que, além disso, essa fórmula também é final, pois não há mais nenhuma nova regra de reescrita que pode ser aplicada. Para esta propriedade presente em termos de um *Term rewriting system* é que damos o nome de Confluência.

Assim surge a questão, é possível que existam processos de reescrita que não sejam confluentes? Para verificar este questionamento, vejamos outro caso provido por Baader e Nipkow (1998). Neste exemplo, utilizamos o sistema de reescrita de fórmulas matemáticas para verificar o processo de *Term Rewriting* da fórmula $D_x(X+0)$.

A princípio, como demonstrado na Figura 22, a fórmula inicial permite que duas regras distintas sejam aplicadas, o que gera uma bifurcação na árvore. Os resultados do processo de reescrita são diferentes, como é esperado, entretanto ao continuar as transformações em cada um dos galhos dessa árvore, obtém-se duas fórmulas finais distintas. Como claramente não

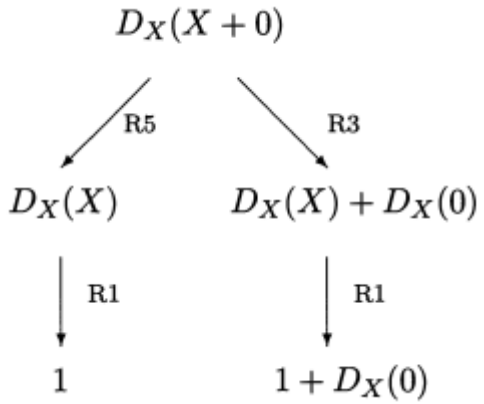

Figura 22 – Exemplo de *Term Rewriting* não confluente

existe uma regra que permita unificar os nós dessa árvore, isso implica que este processo de *Term Rewriting* não é confluente.

Portanto, podemos generalizar a propriedade de confluência de um único termo para definir formalmente a propriedade para um *Term rewriting system* como:

Definição 1. *Um Term rewriting system é considerado confluente se o processo de Term Rewriting para todos os seus termos é confluente.*

Sistemas computacionais que não possuem a propriedade de confluência apresentam um problema grave de não determinismo, pois ele surge a partir do momento que um programa com a mesma entrada, pode assumir resultados diferentes. Logo, traçando um paralelo com a confluência, uma ou mais escolhas feitas durante a execução do programa levam o mesmo a exibir um comportamento diferente do esperado (MCDERMOTT; MYCROFT, 2023).

Um dos fragmentos da computação que está diretamente vinculado ao não determinismo e problemas de confluência são as estratégias de avaliação, pois uma mesma função pode ser avaliada de formas diferentes (Ex: *call-by-name* ou *call-by-value*) e não exibir o mesmo resultado (NISHIZAKI, 1993).

4.1.1 Call-by-name *vs* Call-by-value

Na avaliação *call-by-value*, as expressões utilizadas como argumento são avaliadas antes de serem passadas para uma função. Isso significa que os valores dos argumentos são calculados e associados aos seus respectivos nomes de parâmetro antes da função ser invocada. Como resultado, a função recebe cópias dos valores dos argumentos, e quaisquer alterações feitas nos parâmetros dentro da função não afetam os argumentos originais (MCDERMOTT; MYCROFT, 2023).

Por outro lado, na avaliação *call-by-name*, as expressões utilizadas como argumento não são avaliadas antes de serem passadas para uma função. Em vez disso, elas são passadas em sua forma base não processada. Durante a execução da função, o programa pode escolher

quando e se quer avaliar os argumentos. Isso significa que as expressões de argumento são reavaliadas cada vez que são referenciadas dentro da função. Como resultado, as alterações feitas nos parâmetros dentro da função podem afetar os argumentos originais (MCDERMOTT; MYCROFT, 2023).

O comportamento não confluente de ambos os métodos de avaliação pode ser observado no exemplo da Figura 22. Consideremos que D_x é uma função computacional e que $X+0$ é o argumento passado a ela. Neste caso, a divisão causada pelas regras $R5$ e $R3$ podem ser interpretadas como aplicações de *call-by-value* e *call-by-name*.

Para $R3$, o argumento $X+0$ é avaliado imediatamente, gerando assim um novo valor único que será usado como argumento para D_x. Este é um claro caso de *call-by-value*, pois o argumento foi processado antes da função poder modificá-lo. Por outro lado, em $R5$ temos uma execução considerada *call-by-name* pois o argumento não é processado de imediato, o que permite que a função o altere e, ao mesmo tempo, torne impossível o processamento feito por $R3$.

Assim, fica clara a relação que a confluência tem com o não determinismo que os diferentes métodos de avaliação podem gerar em um sistema. Portanto, podemos concluir que basta garantir a confluência de um sistema e de seu método de avaliação para remover o não determinismo indesejado de um sistema (NISHIZAKI, 1993).

4.2 CONFLUÊNCIA APLICADA À LÓGICA

Como foi demonstrado até agora, a confluência é uma propriedade inerente a *Term rewriting systems* com consequências que abrangem até mesmo o funcionamento de programas. Tendo em vista esse aspecto computacional, podemos utilizar a correspondência de Curry-Howard para traçar um paralelo entre provas lógicas e a confluência (NISHIZAKI, 1993).

Este formalismo demonstra que há uma relação direta entre provas lógicas e programas de computador, ou seja, que há uma correspondência que permite que provas sejam transformadas em código. Portanto, o meio como a prova de um sequente, por exemplo, é construída pode alterar o funcionamento de um programa derivado dessa prova.

$$\dfrac{\dfrac{\dfrac{\dfrac{\pi_1}{\vdash A_1}}{\vdash A_1, B}\,(wk_R) \quad \dfrac{\dfrac{\pi_2}{\vdash A_2}}{B \vdash A_2}\,(wk_L)}{\vdash A_1, A_2}\,(Cut)}{\vdash A}\,(co_R)$$

$$\swarrow \qquad\qquad \searrow$$

$$\dfrac{\dfrac{\dfrac{\pi_1}{\vdash A}}{\vdash A_1, A_2}\,(wk)}{\vdash A}\,(co_R) \qquad\qquad \dfrac{\dfrac{\dfrac{\dfrac{\pi_2}{\vdash A}}{\vdash A_2, A_1}\,(wk_R)}{\vdash A_1, A_2}\,(ex)}{\vdash A}\,(co_R)$$

Figura 23 – Removendo Corte na Lógica Clássica

Vejamos como exemplo a prova em dedução natural da lógica clássica presente na Figura 23, onde também foi demonstrado o processo de reescrever a prova removendo a utilização da regra do corte. Note ainda que, neste exemplo, será utilizado $A \equiv A_1 \equiv A_2$, para facilitar a distinção de fórmulas iguais dentro da prova.

É evidente que, neste caso, há dois meios possíveis de se reescrever a prova para que se obtenha o mesmo resultado final. Entretanto, é preciso lembrar que, como dita a correspondência de Curry-Howard, a estrutura da prova altera o resultado do código. Assim, neste exemplo, não se observa confluência das provas, pois mesmo que elas obtenham o mesmo resultado final, sua estrutura não é a mesma (NISHIZAKI, 1993).

Este comportamento de não confluência computacional pode também ser observado no exemplo proposto por Nishizaki (1993) e representado pela Figura 24. Aqui o autor propõe um exemplo de cálculo lambda que possua a primitiva *call-with-current-continuation* (*call/cc*). Neste exemplo, se a computação utilizar um método de avaliação *call-by-value*, os argumentos serão processados antes de serem chamados pela função, o que fará com que a função gere a resposta "2", como resultado da função **exit** que estava dentro do argumento. Por outro lado, se for feita uma avaliação *call-by-name*, o primeiro **exit** é chamado, gerando 1 como resultado. Como ambos os resultados são diferentes, fica nítido o não determinismo presente entre os diferentes métodos de avaliação.

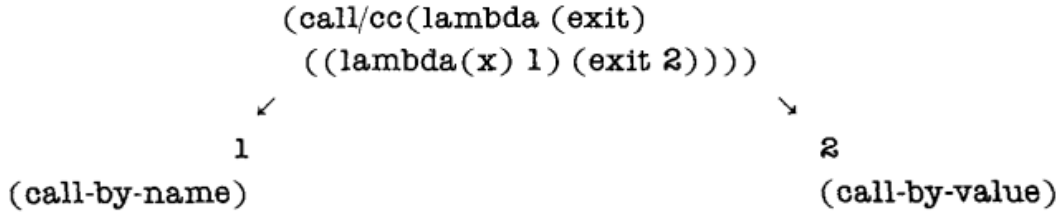

Figura 24 – Não confluência de cálculo lambda *call/cc*

Infelizmente, essa característica de múltiplas provas possíveis que não são confluentes é inerente ao funcionamento da lógica clássica. Esta característica provém do uso irrestrito das regras de *weakening* e *contraction* em ambos os lados do símbolo de derivação. Obviamente, este problema pode ser evitado através de uma restrição no funcionamento dessas regras e da manipulação estrutural da prova. Entretanto, realizar estas modificações a torna muito mais engessada e, ainda assim, não eliminam totalmente a não confluência gerada por regras estruturais como a comutação (NISHIZAKI, 1993).

E é aqui que a lógica linear mostra sua relevância. Como já foi descrito extensivamente ao longo do Capítulo 3, a lógica linear possui, através das regras dos operadores exponenciais, um controle extremamente preciso sobre como e quando as regras de *weakening* e *contraction* podem ser utilizadas.

Assim, a partir da tradução de uma fórmula clássica para uma fórmula equivalente na lógica linear, é possível garantir a construção de múltiplos sequentes que provem a fórmula

traduzida (e por consequência a original também), e que elas serão confluentes. Assim sendo, a tradução garante que provas construídas com base na lógica clássica serão deterministas, pois elas estarão sendo representadas pela lógica linear (NISHIZAKI, 1993).

Por fim, como um exemplo da capacidade computacional provida por uma possível tradução, podemos analisar a interpretação computacional da regra do terceiro excluído. Basicamente o terceiro excluído ao ser chamado, retorna o termo a direita do operador de disjunção ($\neg A \equiv A \Rightarrow \bot$). Caso esta função seja invocada, o programa então retorna a sua operação ao estado que estava no momento que o terceiro excluído foi executado e então retorna o termo a esquerda da disjunção (WADLER, 2003).

5 PROVA DAS TRADUÇÕES

As traduções da lógica clássica e da lógica intuicionista para a lógica linear existem desde a primeira publicação da lógica linear provida pelo criador da mesma Girard (1987). Portanto, as traduções que serão formalizadas em *Coq* e explicadas neste capítulo, terão como base o que já foi provido na literatura. Entretanto, fez-se necessário realizar algumas modificações em como a prova é construída para que assim se torne mais fácil o seu entendimento e a construção de algumas táticas.

A principal mudança está no meio como uma regra da dedução natural é traduzida para um sequente. Enquanto Girard (1987) optou por utilizar um cálculo de sequentes que possui regras para ambos os lados da derivação, a tradução presente neste trabalho utilizará somente as regras do lado direito do sequente.

Esta mudança foi feita pois a biblioteca da lógica linear em *Coq* implementada por Laurent (2017), que será utilizada para a construção das traduções somente possui as regras para o lado direito do sequente. Como demonstrado na Seção 3.2.2, todo sequente pode ser transformado em um sequente que possui somente regras à direita do símbolo de derivação através do uso da regra *left dual*. Além disso, o cálculo de sequentes somente com regras à direita possui o mesmo nível de expressividade que o cálculo de sequentes bilateral, portanto não há nenhuma perda na capacidade de construção de provas através do uso deste método de tradução.

Outra mudança menos significativa realizada na estrutura da prova foi feita devido à necessidade de deixar explícita a utilização da regra de *weakening* em um sequente para gerar um conjunto de fórmulas da lógica linear. Na prova da tradução como descrita por Girard (1987), essa regra é usadas de forma implícita. Entretanto, como o intuito dessa prova é criar um formalismo em um assistente de provas, todo teorema utilizado deve ser explicitamente descrito. Portanto, para facilitar a correlação das provas descritas neste capítulo com aquelas implementadas em *Coq*, o teorema como descrito acima será redefinido e provado a seguir.

Teorema 2. *Seja* $(?\Gamma)$ *uma lista qualquer de fórmulas da lógica linear onde todas as suas fórmulas são precedidas pelo operador lógico "?", a partir de um sequente qualquer é possível deduzir* $(?\Gamma)$.

A prova deste Teorema é construída a partir de uma indução sobre o número de fórmulas em $(?\Gamma)$. Para a base, a solução é trivial já que $(?\Gamma)$ é uma lista vazia. Para o passo, temos como hipótese de indução $\vdash (?\Gamma)$ e provamos que a partir dela é possível construir o sequente $\vdash (?[\Gamma, A])$, como mostrado abaixo:

$$\dfrac{\dfrac{\dfrac{}{\vdash ?\Gamma}\,H}{\vdash ?A, ?\Gamma}\,weakening}{\vdash ?[\Gamma, A]}$$

5.1 TRADUÇÃO DA LÓGICA INTUICIONISTA

A tradução da lógica intuicionista fornecida por Girard (1987), possui uma tradução para todos os operadores presentes na lógica intuicionista clássica, como visto na Figura 25. A terminologia utilizada para expressar a tradução de uma fórmula A da lógica intuicionista para a lógica linear seguirá a seguinte notação: $()^t$. Assim sendo, A^t é uma fórmula pertencente à lógica linear.

$$\begin{array}{cc} A \wedge B = A^t \& B^t & A \vee B = !(A^t) \oplus !(B^t) \\ A \Rightarrow B = !(A^t) \multimap B^t & \neg A = (A^t) \multimap 0 \end{array}$$

Figura 25 – Regras de tradução da Lógica Intuicionista para a Lógica Linear

A princípio, ao analisar as traduções fornecidas, a primeira conclusão que se obtém é que pode-se utilizar a regra da implicação para construir a tradução de uma regra da dedução natural do tipo $\Gamma \vdash A$, como demonstrado na Figura 26. Assim, todas as regras da dedução natural provindas do lado esquerdo do símbolo de dedução serão traduzidas para a lógica linear como $?(\Gamma^t)$.

$$\frac{\overline{\Gamma \vdash A}}{\vdash \Gamma \Rightarrow A}\ (\Rightarrow I) \quad \Longrightarrow \quad \cfrac{\cfrac{\overline{!(\Gamma^t) \vdash A^t}}{\vdash ?(\Gamma^t)^{\perp}, A^t}\ (right\ dual)}{\vdash ?(\Gamma^t)^{\perp} \parr A^t}\ (par)}{\vdash !(\Gamma^t) \multimap A^t}\ (implication)$$

Figura 26 – Regra base para tradução de LI

Tendo definido a forma base de como as regras da dedução serão traduzidas, podemos começar a construir e provar a tradução para o sistema dedutivo da lógica intuicionista. A tradução é definida através de uma indução no tamanho da dedução e sua prova é feita a partir dessa indução, sendo aplicada a cada uma das dez regras presentes na dedução natural para a lógica intuicionista.

Teorema 3. *Seja uma prova qualquer para a lógica intuicionista que termine em* $\Gamma \vdash A$*, e que foi construída a partir de uma das regras de dedução natural intuicionista. Pode-se utilizar o cálculo de sequentes da lógica linear para construir uma prova que tenha como conclusão* $(\vdash (\Gamma^t)^{\perp}, A^t)$.

i. O primeiro passo para provar a tradução de uma prova por dedução natural baseia-se em demonstrar a validade do axioma do sistema dedutivo. Assim, tomando como verdade a hipótese $(\Gamma \vdash A)$, ou seja $(A \in \Gamma)$, devemos provar $(\vdash ?(\Gamma^t)^{\perp}, A^t)$. Assim, a partir da regra de identidade da lógica linear, utilizamos a regra de *weakening* e a hipótese (H) para construir $?(\Gamma^t)^{\perp}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash (A^t)^\perp, A^t}\,(identity)}{\vdash ?(A^t)^\perp, A^t}\,(dereliction)}{\vdash ?(\Delta^t)^\perp, ?(A^t)^\perp, A^t}\,(weakening)}{\vdash ?(\Gamma^t)^\perp, A^t}\,(H)$$

ii. Para o próximo passo, provamos a tradução da introdução da implicação. Para tanto, demonstramos a partir da hipótese $(?(\Gamma^t)\perp, ?(A)^\perp, B)$ é possível construir o sequente $(?(\Gamma^t)^\perp, !(A) \multimap B)$.

$$\dfrac{\dfrac{\dfrac{}{\vdash ?(\Gamma^t)^\perp, ?(A^t), B^t}\,(H)}{\vdash ?(\Gamma^t)^\perp, ?(A^t) \parr B^t}\,(par)}{\vdash ?(\Gamma^t)^\perp, !(A^t) \multimap B^t}\,(implication)$$

iii. Em seguida, provamos a eliminação da implicação (*modus ponens*). Aqui, assumimos as hipóteses H': $(\vdash ?(\Gamma^t)^\perp, !(A^t) \multimap B^t)$ e H'': $(\vdash ?(\Delta^t)^\perp, (A^t))$ para construir a prova. Note que, na construção do sequente a partir de H'', foi necessário utilizar a regra $of\ course$. E só foi possível utilizar a mesma pois, como definido anteriormente, todas as regras do conjunto $?(\Delta^t)$ são precedidas pelo operador ?.

$$\dfrac{\dfrac{\dfrac{}{\vdash ?(\Gamma^t)^\perp, !(A^t) \multimap B^t}\,(H') \quad (\multimap)}{\vdash ?(\Gamma^t)^\perp, ?(A^t)^\perp \parr B^t} \qquad \dfrac{\dfrac{\dfrac{}{\vdash ?(\Delta^t)^\perp, (A^t)}\,(H'')}{\vdash ?(\Delta^t)^\perp, !(A^t)}\,(of\ course) \qquad \dfrac{}{\vdash (B^t)^\perp, B^t}\,(identity)}{\vdash ?(\Delta^t)^\perp, !(A^t) \otimes (B^t)^\perp, B^t}\,(times)}{\vdash ?(\Gamma^t)^\perp, ?(\Delta^t)^\perp, B^t}\,(cut)$$

iv. Agora, focamos em provar a regra que define a dedução natural intuicionista, a introdução do absurdo. Perceba que, como definido de forma implícita na Figura 25, utilizamos o 0 para representar a tradução de $\perp$. Portanto, a hipótese $(\vdash ?(\Gamma^t)^\perp, (\perp)^t)$ pode ser interpretada como $(\vdash ?(\Gamma^t)^\perp, 0)$

$$\dfrac{\dfrac{}{\vdash ?(\Gamma^t)^\perp, 0}\,(H) \qquad \dfrac{}{\vdash A^t, \top}\,(true)}{\vdash ?(\Gamma^t)^\perp, A^t}\,(cut)$$

v. A próxima regra a ser provada é a introdução da conjunção. Aqui, conforme as regras para a dedução natural, assumimos as hipóteses H': $(\vdash ?(\Gamma^t)^\perp, A^t)$ e H'': $(\vdash ?(\Delta^t)^\perp, B^t)$. Note que, como provado no Teorema 2, precisamos aplicar a regra de *weakening* de um conjunto para que seja possível unificar ambos os sequentes que contém as hipóteses e assim, construir a tradução.

$$\dfrac{\dfrac{\dfrac{}{\vdash ?(\Gamma^t)^\perp, A^t}\,(H')}{\vdash ?(\Gamma^t)^\perp, ?(\Delta^t)^\perp, A^t}\,(weakening) \qquad \dfrac{\dfrac{}{\vdash ?(\Delta^t)^\perp, B^t}\,(H'')}{\vdash ?(\Gamma^t)^\perp, ?(\Delta^t)^\perp, B^t}\,(weakening)}{\vdash ?(\Gamma^t)^\perp, ?(\Delta^t)^\perp, A^t \& B^t}\,(with)$$

vi. A eliminação da conjunção na dedução natural possui duas regras. Para a primeira regra, tomamos $(\vdash ?(\Gamma^t)^\perp, A^t \& B^t)$ como hipótese para construir o sequente.

$$\dfrac{\dfrac{}{\vdash ?(\Gamma^t)^\perp, A^t \& B^t}\,(H') \quad \dfrac{\dfrac{\dfrac{}{\vdash A^t, (A^t)^\perp}\,(identity)}{\vdash A^t, (A^t)^\perp \oplus (B^t)^\perp}\,(left\ plus)}{\vdash A^t, (A^t \& B^t)^\perp}\,(dual)}{\vdash ?(\Gamma^t)^\perp, A^t}\,(cut)$$

vii. Para a segunda regra da eliminação da conjunção, temos a mesma hipótese que foi utilizada acima, porém aplicamos a outra regra do operador *plus*, gerando assim um sequente que "espelha" a prova acima.

$$\dfrac{\dfrac{}{\vdash ?(\Gamma^t)^\perp, A^t \& B^t}\,(H') \quad \dfrac{\dfrac{\dfrac{}{\vdash B^t, (B^t)^\perp}\,(identity)}{\vdash B^t, (A^t)^\perp \oplus (B^t)^\perp}\,(right\ plus)}{\vdash B^t, (A^t \& B^t)^\perp}\,(dual)}{\vdash ?(\Gamma^t)^\perp, B^t}\,(cut)$$

viii. As três ultimas regras que precisam ser demonstradas são as regras para o operador de disjunção. A primeira delas a ser provada é a eliminação da disjunção. Aqui, será construído o sequente a partir da hipótese H: $(\vdash ?(\Gamma^t)^\perp, !(A^t) \oplus !(B^t))$ e das hipóteses D: $(\vdash ?(\Delta^t)^\perp, C^t, ?(A^t)^\perp)$ e D': $(\vdash ?(\Psi^t)^\perp, C^t, ?(B^t)^\perp)$.

$$\dfrac{\dfrac{}{\vdash ?(\Gamma^t)^\perp, !(A^t) \oplus !(B^t)}\,(H) \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash ?(\Delta^t)^\perp, C^t, ?(A^t)^\perp}\,(D)}{\vdash ?(\Delta^t)^\perp, ?(\Psi^t)^\perp, C^t, ?(A^t)^\perp}\,(wk) \quad \dfrac{\dfrac{}{\vdash ?(\Psi^t)^\perp, C^t, ?(B^t)^\perp}\,(D')}{\vdash ?(\Delta^t)^\perp, ?(\Psi^t)^\perp, C^t, ?(B^t)^\perp}\,(wk)}{\vdash ?(\Delta^t)^\perp, ?(\Psi^t)^\perp, C^t, ?(A^t)^\perp \& ?(B^t)^\perp}\,(with)}{\vdash ?(\Delta^t)^\perp, ?(\Psi^t)^\perp, C^t, (!(A^t) \oplus !(B^t))^\perp}\,(dual)}{\vdash ?(\Gamma^t)^\perp, ?(\Delta^t)^\perp, ?(\Psi^t)^\perp, C^t}\,(cut)$$

ix. Por fim, é necessário provar a tradução para as duas regras de introdução da disjunção. Para provar a primeira regra, assumimos $(\vdash ?(\Gamma^t)^\perp, A^t)$ como hipótese para construir o sequente.

$$\dfrac{\dfrac{\dfrac{}{\vdash ?(\Gamma^t)^\perp, A^t}\,(H)}{\vdash ?(\Gamma^t)^\perp, !(A^t)}\,(of\ course)}{\vdash ?(\Gamma^t)^\perp, !(A^t) \oplus !(B^t)}\,(left\ plus)$$

x. De forma similar, para a segunda regra da introdução da disjunção, assim assumimos $(\vdash ?(\Gamma^t)^\perp, B^t)$ como hipótese para construir o sequente. Note que, em ambas as provas para a introdução da disjunção, pudemos usar a regra *of course* pois, todas as regras presentes no sequente no momento de aplicação da mesma estavam precedidas pelo operador ?.

$$\dfrac{\dfrac{\dfrac{}{\vdash ?(\Gamma^{\,t})^{\perp}, B^{t}}\,(H)}{\vdash ?(\Gamma^{\,t})^{\perp}, !(B^{t})}\,(of\ course)}{\vdash ?(\Gamma^{\,t})^{\perp}, !(A^{t}) \oplus !(B^{t})}\,(left\ plus)$$

5.1.1 Variações da Tradução para a lógica intuicionista

Além da tradução demonstrada no início desta seção, em sua publicação original da lógica linear Girard (1987) apresentou mais um meio de realizar esta tradução, a qual suas regras estão definidas de forma adaptada na Figura 27.

$$\begin{array}{cc} A \wedge B = !(A^{t}) \otimes !(B^{t}) & A \vee B = !(A^{t}) \oplus !(B^{t}) \\ A \to B = !(A^{t}) \multimap !(B^{t}) & \neg A = !(A^{t}) \multimap !0 \end{array}$$

Figura 27 – Regras de tradução *call-by-name* da Lógica Intuicionista para a Lógica Linear

Esta tradução funciona de forma similar a que já foi apresentada, pois ela também se estende a uma tradução de provas. Assim, é possível construir para esta tradução uma prova para o Teorema 3 de forma extremamente similar à que já foi apresentada. Portanto, nesta seção não será demonstrada esta prova, só será discutido o propósito que ambas as traduções possuem.

Entretanto, vale ressaltar uma mudança da construção da prova. Como fica evidente através da análise das novas regras, houve uma mudança em como a implicação é traduzida. Assim, com base nesta regra, a tradução para uma regra de dedução natural base do tipo $\Gamma \vdash A$, será $\vdash ?(\Gamma^{\,t}), !(A^{t})$

A existência de duas traduções serve um propósito interessante para a proposta da resolução do problema de confluência da lógica intuicionista. Veja, como se pode induzir do que foi demonstrado no Capítulo 4, o próprio processo de tradução também é um *Term rewriting System*. Assim, a ordem de avaliação é algo que deve ser levado em conta baseado no contexto no qual a tradução será efetuada.

Assim, podemos definir que a tradução da Figura 25 possui ordem de avaliação *call-by-value*. Por outro lado, a segunda tradução, apresentada na Figura 27, possui ordem de avaliação *call-by-name* Girard (1987).

5.2 TRADUÇÃO DA LÓGICA CLÁSSICA

A lógica clássica, assim como a lógica intuicionista, possui duas traduções possíveis propostas por Girard (1987), que seguem o mesmo propósito demonstrado na Seção 5.1.1. Entretanto, neste trabalho só foi possível demonstrar a tradução *call-by-value* da lógica clássica. Essa tradução é feita de forma similar às outras já apresentadas, como exposto na Figura 28.

Uma propriedade importante desta tradução para a construção de provas é o fato que toda fórmula da lógica clássica quando traduzida, é precedida pelo operador exponencial ?. Isso permite que seja possível, para um sequente qualquer do tipo $\vdash \Gamma, !A$, deduzir este sequente a

$$A \wedge B = ?(A^t \,\&\, B^t) \qquad A \vee B = ?(!(A^t) \oplus !(B^t))$$
$$A \to B = !(A^t) \multimap B^t \qquad \neg A = ??(A^t)^{\perp}$$

Figura 28 – Regras de tradução da Lógica Clássica para a Lógica Linear

partir da utilização da regra de *of course* no sequente $\vdash \Gamma, A$, desde que todos os membros de Γ sejam precedidos por ? ou sejam uma tradução $()^t$.

É possível também construir e provar a tradução para o sistema dedutivo da lógica clássica. A tradução é definida de forma similar à lógica intuicionista, por meio de uma indução no tamanho da dedução e sua prova é feita a partir dessa indução. Entretanto, por mais que seja possível provar o Teorema 3 para esta tradução de forma similar ao que já foi demonstrado para a lógica intuicionista, isso não é o suficiente para provar esta tradução pois, como demonstrado na Seção 2.3.1, a lógica clássica possui mais duas regras para a dedução natural.

Assim, é necessário expandir a prova do Teorema 3 para que inclua a regra de eliminação da dupla negação e a regra de incorporação do terceiro excluído. Para tanto, definimos os seguintes passos indutivos:

i. Para a eliminação da dupla negação, precisa-se definir uma tradução da constante $\perp$. Ao analisar as regras definidas para a tradução, podemos inferir que a constante $\perp$ da lógica clássica é traduzida como $?\perp$, sendo esta última uma constante da lógica linear. Assim, podemos construir a prova assumindo como verdade a hipótese H: $(\vdash ?(\Gamma^t)^{\perp}, ?!!(A^t), ?\perp)$.

$$\dfrac{\dfrac{\dfrac{}{\vdash ?(\Gamma^t)^{\perp}, ?!!(A^t), ?\perp}\,(H)}{\vdash ?(\Gamma^t)^{\perp}, (?!!(A^t) \parr ?\perp)}\,(parr) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^t, (A^t)^{\perp}}\,(identity)}{\vdash A^t, ?(A^t)^{\perp}}\,(dereliction)}{\vdash A^t, ??(A^t)^{\perp}}\,(dereliction)}{\vdash A^t, !??(A^t)^{\perp}}\,(ofcourse) \quad \dfrac{\dfrac{}{\vdash 1}\,(one)}{\vdash !1}\,(ofcourse)}{\vdash A^t, (!??(A^t)^{\perp} \otimes !1)}\,(times)}{\vdash (?!!(A^t) \parr ?\perp)^{\perp}, A^t}\,(dual)}{\vdash ?(\Gamma^t)^{\perp}, A^t}\,(cut)$$

ii. Para a incorporação do terceiro excluído não há nenhuma hipótese. Portanto, basta construir a tradução da regra utilizando o cálculo de sequentes da lógica linear.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash (A^t)^{\perp},(A^t)}\,(identity)}{\vdash ?(A^t)^{\perp},(A^t)}\,(dereliction)}{\vdash ??(A^t)^{\perp},(A^t)}\,(dereliction)}{\vdash ??(A^t)^{\perp},!(A^t)}\,(ofcourse)}{\vdash ??(A^t)^{\perp},(!(A^t)\oplus !??(A^t)^{\perp})}\,(leftplus)}{\vdash ??(A^t)^{\perp},?(!(A^t)\oplus !??(A^t)^{\perp})}\,(dereliction)}{\vdash !??(A^t)^{\perp},?(!(A^t)\oplus !??(A^t)^{\perp})}\,(ofcourse)}{\vdash (!(A^t)\oplus !??(A^t)^{\perp}),?(!(A^t)\oplus !??(A^t)^{\perp})}\,(rightplus)}{\vdash ?(!(A^t)\oplus !??(A^t)^{\perp}),?(!(A^t)\oplus !??(A^t)^{\perp})}\,(dereliction)}{\vdash ?(\Gamma\, ^t)^{\perp},?(!(A^t)\oplus !??(A^t)^{\perp}),?(!(A^t)\oplus !??(A^t)^{\perp})}\,(weakening)}{\vdash ?(\Gamma\, ^t)^{\perp},?(!(A^t)\oplus !??(A^t)^{\perp})}\,(contracion)$$

A utilização dos operadores exponenciais nesta tradução faz com que a prova das regras descritas acima sejam grandes. Esse aumento também é observado na tradução para as outras regras da dedução natural. Porém, este fato não força a necessidade de demonstrar de forma explícita a prova de cada uma das regras, pois como já foi dito, as regras são provadas ao que já foi demonstrado no Teorema 3. Ou seja, as hipóteses são as mesmas, porém adaptadas para estar de acordo com as regras de tradução.

5.3 FORMALIZAÇÃO DAS TRADUÇÕES NO COQ

As provas de todas as traduções formalizadas neste trabalho segue o mesmo método de prova já discutido extensivamente no decorrer deste capítulo. Como sua implementação é longa, não cabe incluir a mesma em sua íntegra neste texto. Entretanto, ainda há alguns pontos relevantes sobre o formalismo que merecem ser mencionados.

O primeiro destes tem relação a como foi definido a tradução dos operadores e do sequente base para todas as três traduções formalizadas. Na Figura 29, há um exemplo de como esta regra foi construída para a tradução *call-by-value* da lógica intuicionista. Veja que, a tradução dos operadores é relativamente simples, o *Fixpoint* **cbv** aplica uma recursão que traduz cada uma das sub-fórmulas contidas dentro de uma fórmula.

Neste mesmo exemplo, também é possível analisar a tradução do sequente base (definida por **dual_set_cbv**), onde é utilizado novamente uma recursão para que, cada fórmula lógica contida dentro do conjunto Γ é traduzida utilizando **cbv**. Além disso, como visto na Figura 26, também foi necessário utilizar a regra da implicação para a tradução das fórmulas. Dessa maneira, é garantido que todas fórmulas traduzidas serão precedidas pelo operador ? e que elas serão o dual da tradução base, já que só serão usadas regras a direita do sequente para a construção do formalismo.

Vale ainda notar que as outras duas traduções formalizadas são construídas de forma similar, obviamente tendo como diferença que as definições formalizadas levam em conta suas

respectivas regras de tradução.

```
Fixpoint cbv (a: PropF) : formula :=
match a with
  | Var A => var A
  | Disj A B => aplus (oc (cbv A)) (oc(cbv B))
  | Impl A B => parr (wn (dual (cbv A))) (cbv B)
  | Neg A => parr (wn (dual (cbv A))) zero
  | Bot => zero
  | Conj A B => awith(cbv A) (cbv B)
end.


Fixpoint dual_set_cbv (a: list PropF) : list formula :=
match a with
  | [] => []
  | A::t => (wn (dual (cbv A)))::(dual_set_cbv t)
end.
```

Figura 29 – Definição formal da tradução *call-by-value* da lógica intuicionista

Outro ponto interessante para se ressaltar pode ser observado na Figura 30, onde é apresentado a definição do formalismo para o Teorema 3, tanto para a tradução *call-by-value* intuicionista, quanto para a tradução *call-by-name* intuicionista. Veja que, na definição da tradução *call-by-name* a tradução da fórmula A é feita utilizando o operador !. Esta diferença entre a tradução *cbn* e a tradução *cbv* se da pelo meio como a tradução da implicação é definida em *cbn*, assim o sequente base é traduzido como já demonstrado na Seção 5.1.1.

```
Theorem proof_cbv: forall Γ A, Γ \- A -> (ll ((cbv A)::(dual_set_cbv Γ))).
Proof.
intros. dependent induction H.

Theorem proof_cbn: forall Γ A, Γ \- A -> (ll ((oc (cbn A))::(dual_set_cbn Γ))).
Proof.
intros. induction H.
```

Figura 30 – Formalismo para o teorema da tradução *call-by-value* e *call-by-name* da lógica intuicionista

Além disso, é possível evidenciar outras duas características interessantes de como este formalismo foi definido. O primeiro ponto é o método de construção de prova que, como já foi demonstrado neste capítulo, é baseado em uma indução sobre as regras da dedução natural. O segundo ponto está relacionado a como as fórmulas foram dispostas no sequente traduzido, que no formalismo está como $\vdash A^t, (\Gamma\ ^t)^{\perp}$. Esta comutação é válida, pois a comutação de fórmulas em um sequente é permitido na lógica linear, e ela foi feita para facilitar a aplicação das regras do cálculo de sequentes durante a construção das provas dentro do assistentes de provas.

6 TRABALHOS RELACIONADOS

Neste capítulo serão discutidos trabalhos que possuem implementações para a lógica linear em um assistente de provas. O desenvolvimento de um sistema de provas para a lógica linear utilizando o assistente de provas Coq é algo abordado por diversos autores, porém com diferentes peculiaridades em suas implementações, como será discutido nas seções abaixo. Além destas, existe uma proposta feita por Trifunovski (2017), a qual foi implementada em OCaml, e que propõe a construção de um assistente de provas especificamente para a lógica linear.

6.1 XAVIER (2017)

Este autor teve como objetivo modelar um sistema primariamente sintático para a lógica linear utilizando o assistente de provas Coq e utilizar este sistema para criar e manipular provas e metateoremas referentes à lógica. Para alcançar tal feito, Xavier (2017) traduziu a sintaxe da lógica para o Coq junto com a definição da negação linear e a prova de que a mesma é involutiva, além de formalizar o cálculo de sequentes.

Sua formalização do cálculo de sequentes é baseada em um módulo que foi implementado especificamente para ser utilizado neste trabalho e é baseado em multiconjuntos, que são uma modificação da definição de conjuntos para que permita múltiplas instâncias de um elemento. Como consequência disso, o autor demonstra que não é necessário a utilização da regra *exchange*, devido a propriedade de permutação que os multiconjuntos possuem. Além disso, múltiplos sistemas de sequentes foram implementados pelo autor, entre eles foram implementados:

- O sistema *two-sided*, onde cada conectivo possui regras *left* e *right*;
- O sistema monádico, onde os conectivos possuem somente as regras *right*;
- O sistema com altura, que modifica as regras do sistema monádico de forma a permitir a utilização de regras de indução;
- E o sistema com altura, regra do corte e número de cortes.

As provas construídas a partir do sistema proposto se dividem em duas partes. A primeira consiste em provar propriedades sobre os sistemas de cálculo de sequentes indutivos que foram propostos e demonstrar o funcionamento de certas regras neste sistema. A segunda parte utiliza as provas que foram construídas no trabalho para provar a eliminação da regra do corte nos diversos sistemas de sequentes.

Em particular, a regra do corte, que é o principal foco deste trabalho, foi provada de forma indutiva a partir da introdução de duas novas características para as fórmulas da lógica linear e seus sequentes. A primeira delas é o tamanho do corte e a segunda é o tamanho das premissas, o qual é definido a partir do número de regras e da altura das provas. O que efetivamente

faz com que seja possível aplicar a indução, pois ela permite que se interprete a formação de um sequente a partir de uma base e uma sequência de passos.

6.2 LAURENT (2017)

Inspirando seu trabalho no projeto citado anteriormente, Laurent (2017) propõe criar uma biblioteca que prove uma *deep embedding* para a lógica linear em Coq. As semelhanças nas propostas dos projetos são claras, porém este trabalho possui uma diferença fundamental: ela trata os sequentes como listas ao invés de multiconjuntos.

O principal motivo desta mudança se deve à dificuldade de se atribuir valor computacional às provas que usam multiconjuntos através da correspondência de Curry-Howard. Essencialmente, ao utilizar multiconjuntos para representar um sequente, se torna impossível distinguir múltiplas ocorrências de uma mesma fórmula. Por outro lado, ao interpretar um sequente como uma lista e criar uma regra de *exchange* explícita, é possível rastrear as ocorrências de fórmulas iguais através das derivações.

E o mais importante é que, mesmo com estas modificações no formato computacional dos sequentes, o autor ainda assim conseguiu provar a remoção da regra do corte de forma semelhante ao trabalho anterior, sendo que a principal diferença é a quantidade maior de casos que precisam ser provados, pois é necessário levar em conta a influência da regra *exchange*.

6.3 KALVALA E PAIVA (1995)

Este artigo teve como proposta modelar um sistema para a lógica linear utilizando o assistente Isabelle. As autoras Kalvala e Paiva (1995) optaram por utilizar o cálculo de sequentes como o sistema dedutivo para a construção das provas, entretanto, diferente dos outros artigos mencionados, neste trabalho foi utilizado uma variante de uma formalização do cálculo de sequentes de Gentzen (LK), ou seja, o sistema possui regras de sequentes para os dois lados da derivação.

Nesta implementação há um foco grande destinado à automatização na busca de provas para as fórmulas. O artigo menciona três regras do cálculo de sequente que geram problemas para algoritimos de busca: *exchange*, *contraction* e *cut*. O que é uma conclusão óbvia, já que estas três regras são as únicas que fazem com que buscas por exaustão não funcionem.

Para remover o problema causado pela regra *exchange*, as autoras propõem que os sequentes deixem de ser uma lista e passem a ser interpretados como um multiconjunto. Os problemas gerados pela regra *contraction* foram parcialmente mitigados pela adição de diversos lemas extras que substituem a utilização desta regra e assim reduz a quantidade de vezes que a esta regra é utilizada. A regra *cut*, por outro lado, não foi removida ou modificada devido a sua grande utilidade na criação de provas.

Em suma, o projeto teve êxito em implementar métodos de construir de forma manual provas para a lógica linear no assistente de provas Isabelle e, além disso, conseguiu provar de

forma automática diversos lemas propostos na literatura.

6.4 CONSIDERAÇÕES SOBRE OS TRABALHOS RELACIONADOS

Ao criarmos um comparativo entre os trabalhos apresentados nas sessões anteriores, como apresentado na Tabela 1, podemos utilizá-lo para traçar algumas conclusões.

Autores	**Cálculo de Sequentes**	**Representação dos Sequentes**	**Sistema Semântico**	**Eliminação do Corte**
Xavier	*One-sided*	Multiconjuntos	Não abordado	Provado
Laurent	*One-sided*	Listas	Não abordado	Provado
Kalvala e Paiva	*Two-sided*	Multiconjuntos	Não abordado	Não provado

Tabela 1 – Comparativo entre os trabalhos relacionados

Podemos concluir que o melhor sistema dedutivo para se trabalhar é o cálculo de sequentes *one-sided* devido a reduzida quantidade de regras que o mesmo possui, fazendo com que seja mais fácil o entendimento e a construção das provas. Da mesma forma, a representação dos sequentes como multiconjuntos também facilita a criação de provas, entretanto ela talvez não seja a melhor forma de se representar os sequentes, pois tem o potencial de limitar a expressividade computacional de uma prova.

E também podemos determinar que é possível provar a eliminação da regra do corte em um assistente de provas através de uma indução relativamente padrão sobre a altura do corte e a altura das premissas. Por outro lado, por mais que a lógica linear possua as propriedade de correção e completude, como provado por Girard (1987), os autores decidiram não abordar estas provas devido a complexidade do sistema semântico da lógica, e portanto este sistema também está fora do escopo deste projeto.

7 CONSIDERAÇÕES FINAIS

A proposta inicial deste trabalho, visava em criar um formalismo para a tradução da lógica clássica e intuicionista para a lógica linear utilizando o provador de teoremas *Coq*. O intuito desta tradução é permitir que haja um meio de garantir que ambas as lógicas estejam em conformidade com a propriedade de *Term Rewriting Systems* chamada confluência.

Este projeto obteve êxito na conclusão destes objetivos ao implementar um formalismo para duas traduções da lógica intuicionista e uma da lógica clássica (podem ser acessadas na íntegra em <https://github.com/AlexandreJSehnem/coq-cpl-to-ll-translation-proof/>). Além disso, estes formalismos cumprem o propósito de garantir que ambas as lógicas possuam a propriedade de confluência, pois as traduções formalizadas já possuem esta propriedade.

A implementação deste formalismo utilizou em seu fragmento linear a biblioteca *NanoYalla*, a qual foi implementada por Laurent (2017). Devido ao alto nível de recursos fornecidos por esta biblioteca, a mesma permite que o formalismo implementado neste trabalho seja utilizado junto a outras propriedades da lógica linear já provadas (Ex: remoção do corte).

Além disso, a explicação das diversas propriedades da lógica linear e o comparativo de como seu funcionamento se compara com as outras lógicas apresentadas, auxilia o leitor a compreender as propriedades da lógica linear, seu propósito e inclusive, a aprender funcionalidades que saem do escopo das traduções, como por exemplo, a interpretação das fórmulas como recurso.

7.1 TRABALHOS FUTUROS

Como proposta de trabalhos futuros para este trabalho, há múltiplas melhorias possíveis que podem ser implementadas. A finalização da tradução para a lógica proposicional clássica, ao construir a prova da tradução *call-by-name*. Em seguida, a expansão das traduções já descritas, através da implementação de regras para os conectivos de primeira ordem.

Em uma outra vertente, também seria interessante criar a prova inversa para cada uma das traduções, ou seja, a partir de uma fórmula da lógica linear, obter uma fórmula correspondente na lógica clássica ou na intuicionista. Por mais que está tradução não tenha o propósito de confluência, ainda seria interessante para reforçar a validade das provas já feitas.

REFERÊNCIAS

ALEXIEV, Vladimir. Applications of linear logic to computation: An overview. **Logic Journal of IGPL**, v. 2, n. 1, p. 77107, 1994. Citado 2 vezes nas páginas 9 e 18.

ATTEN, Mark van. The Development of Intuitionistic Logic. In: ZALTA, Edward N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Summer 2022. [S.l.]: Metaphysics Research Lab, Stanford University, 2022. Citado na página 14.

BAADER, F.; NIPKOW, T. **Term Rewriting and All That**. Cambridge University Press, 1998. (Term Rewriting and All that). ISBN 9780521779203. Disponível em: <https://books.google.com.br/books?id=N7BvXVUCQk8C>. Citado 3 vezes nas páginas 34, 35 e 36.

BORNAT, Richard. **Proof and disproof in formal logic**: an introduction for programmers. [S.l.]: Oxford University Press, 2005. Citado na página 9.

CHAUDHURI, Kaustuv; DESPEYROUX, Joëlle. A hybrid linear logic for constrained transition systems with applications to molecular biology. **arXiv preprint arXiv:1310.4310**, 2013. Citado 2 vezes nas páginas 9 e 18.

COSMO, Roberto Di; MILLER, Dale. Linear Logic. In: ZALTA, Edward N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Summer 2019. [S.l.]: Metaphysics Research Lab, Stanford University, 2019. Citado 2 vezes nas páginas 9 e 18.

ENDERTON, H.B. **A Mathematical Introduction to Logic**. Elsevier Science, 2001. ISBN 9780080496467. Disponível em: <https://books.google.com.br/books?id=dVncCl_EtUkC>. Citado 2 vezes nas páginas 13 e 14.

GIRARD, Jean-Yves. Linear logic. **Theoretical Computer Science**, v. 50, n. 1, p. 1101, 1987. Citado 10 vezes nas páginas 9, 10, 16, 18, 31, 32, 41, 42, 45 e 51.

GIRARD, J.-Y. Linear logic: its syntax and semantics. **Advances in Linear Logic**, p. 142, 2005. Citado na página 18.

HERBRAND, Jacques. Recherches sur la théorie de la démonstration. In: . [s.n.], 1930. (Thèses de l'entre-deux-guerres, 110). Disponível em: <http://archive.numdam.org/item/THESE_1930__110__1_0/>. Citado na página 25.

KALVALA, Sara; PAIVA, Valeria de. Mechanizing linear logic in isabelle. In: . [S.l.: s.n.], 1995. Citado na página 50.

LAGO, Ugo Dal; FAGGIAN, Claudia. On multiplicative linear logic, modality and quantum circuits. **Electronic Proceedings in Theoretical Computer Science**, v. 95, p. 5566, 2012. Citado 2 vezes nas páginas 9 e 18.

LAURENT, Olivier. Preliminary report on the yalla library. In: . [S.l.: s.n.], 2017. Citado 3 vezes nas páginas 41, 50 e 52.

LINCOLN, Patrick et al. Decision problems for propositional linear logic. **Annals of Pure and Applied Logic**, v. 56, n. 1, p. 239–311, 1992. ISSN 0168-0072. Disponível em: <https://www.sciencedirect.com/science/article/pii/016800729290075B>. Citado 2 vezes nas páginas 32 e 33.

MARTÍ-OLIET, Narciso; MESEGUER, José. From petri nets to linear logic. In: PITT, David H. et al. (Ed.). **Category Theory and Computer Science**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989. p. 313–340. ISBN 978-3-540-46740-3. Citado 2 vezes nas páginas 9 e 31.

MCDERMOTT, Dylan; MYCROFT, Alan. **Galois connecting call-by-value and call-by-name**. 2023. Citado 2 vezes nas páginas 37 e 38.

MOOT, R.; PIAZZA, M. Linguistic applications of first order intuitionistic linear logic. **Journal of Logic, Language and Information**, v. 10, p. 211232, 2001. Citado 2 vezes nas páginas 9 e 18.

MOSCHOVAKIS, Joan. Intuitionistic Logic. In: ZALTA, Edward N.; NODELMAN, Uri (Ed.). **The Stanford Encyclopedia of Philosophy**. Winter 2022. [S.l.]: Metaphysics Research Lab, Stanford University, 2022. Citado na página 14.

NISHIZAKI, Shin ya. Programs with continuations and linear logic. **Science of Computer Programming**, v. 21, n. 2, p. 165–190, 1993. ISSN 0167-6423. Disponível em: <https://www.sciencedirect.com/science/article/pii/016764239390005A>. Citado 6 vezes nas páginas 10, 35, 37, 38, 39 e 40.

PAULIN-MOHRING, Christine. Introduction to the coq proof-assistant for practical software verification. In: 0001, Bertrand Meyer; NORDIO, Martin (Ed.). **LASER Summer School**. [S.l.]: Springer, 2011. (Lecture Notes in Computer Science, v. 7682), p. 45–95. ISBN 978-3-642-35745-9; 978-3-642-35746-6. Citado na página 10.

PRIEST, Graham. **An introduction to non-classical logic**. 2. ed. New York: Cambridge University Press, 2009. Citado 3 vezes nas páginas 9, 14 e 15.

PRIEST, G. **Logic: A Very Short Introduction**. Oxford University Press, 2017. (Very short introductions). ISBN 9780198811701. Disponível em: <https://books.google.com.br/books?id=oZ83DwAAQBAJ>. Citado 2 vezes nas páginas 12 e 13.

ROBINSON, A.J.A.; VORONKOV, A. **Handbook of Automated Reasoning**. Elsevier Science, 2001. (Handbook of Automated Reasoning, v. 1). ISBN 9780444829498. Disponível em: <https://books.google.com.vc/books?id=X3z8ujBRgmEC>. Citado na página 35.

SHAPIRO, Stewart; KISSEL, Teresa Kouri. Classical Logic. In: ZALTA, Edward N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Spring 2021. [S.l.]: Metaphysics Research Lab, Stanford University, 2021. Citado 2 vezes nas páginas 19 e 21.

SILVA, Flavio. **Logica para computacao**. Sao Paulo: Thomson Learning, 2006. ISBN 8522105170. Citado 2 vezes nas páginas 21 e 24.

TRIFUNOVSKI, Maksim. An interactive proof assistant for linear logic. In: . [S.l.: s.n.], 2017. Citado na página 49.

TROELSTRA, A. S.; SCHWICHTENBERG, H. **Basic Proof Theory**. 2. ed. [S.l.]: Cambridge University Press, 2000. (Cambridge Tracts in Theoretical Computer Science). Citado 4 vezes nas páginas 12, 15, 16 e 17.

WADLER, Philip. Call-by-value is dual to call-by-name. In: . [S.l.: s.n.], 2003. v. 8. Citado na página 40.

XAVIER, Bruno Francisco. **Formalização da Lógica Linear em Coq**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, Natal, 2017. Citado 3 vezes nas páginas 9, 32 e 49.