UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MESTRADO ACADÊMICO EM COMPUTAÇÃO

FRANCISCO LEONARDO BATISTA MARTINS

AVALIANDO O DESEMPENHO DE GRANDES MODELOS DE LINGUAGEM NA REALIZAÇÃO DE PROVAS DE DEDUÇÃO NATURAL EM LÓGICA PROPOSICIONAL E LÓGICA DE PREDICADOS

QUIXADÁ

2025

FRANCISCO LEONARDO BATISTA MARTINS

AVALIANDO O DESEMPENHO DE GRANDES MODELOS DE LINGUAGEM NA REALIZAÇÃO DE PROVAS DE DEDUÇÃO NATURAL EM LÓGICA PROPOSICIONAL E LÓGICA DE PREDICADOS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Computação. Área de Concentração: Algoritmos e Teoria da Computação.

Orientador: Prof. Dr. Davi Romero de Vasconcelos.
Coorientadora: Profa. Dra. Maria Viviane de Menezes.

QUIXADÁ

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M343a Martins, Francisco Leonardo Batista.
Avaliando o desempenho de grandes modelos de linguagem na realização de provas de dedução natural em lógica proposicional e lógica de predicados / Francisco Leonardo Batista Martins. – 2025.
82 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2025.
Orientação: Prof. Dr. Davi Romero de Vasconcelos.
Coorientação: Profa. Dra. Maria Viviane de Menezes.

1. Dedução natural. 2. Lógica proposicional. 3. Lógica de predicados. 4. Raciocínio lógico. 5. Grandes modelos de linguagem. I. Título.

CDD 005

FRANCISCO LEONARDO BATISTA MARTINS

AVALIANDO O DESEMPENHO DE GRANDES MODELOS DE LINGUAGEM NA REALIZAÇÃO DE PROVAS DE DEDUÇÃO NATURAL EM LÓGICA PROPOSICIONAL E LÓGICA DE PREDICADOS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Computação. Área de Concentração: Algoritmos e Teoria da Computação.

Aprovada em: 28 de Agosto de 2025

BANCA EXAMINADORA

Prof. Dr. Davi Romero de Vasconcelos (Orientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Maria Viviane de Menezes (Coorientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Bruno Lopes Vieira
Universidade Federal Fluminense (UFF)

Prof. Dr. Alexandre Matos Arruda
Universidade Federal do Ceará (UFC)

Aos meus pais Francisco e Carmem, que fizeram o possível para que eu tivesse as oportunidades que eles nunca tiveram.

AGRADECIMENTOS

Agradeço a Deus, em primeiro lugar, a toda a minha família, especialmente ao meu pai, Francisco, e à minha mãe, Carmem, pelo apoio.

À professora Viviane, por sua ajuda e paciência, e ao professor Davi, por seu apoio, confiança e por ter me motivado a não desistir.

Agradeço aos meus amigos da igreja, por me ajudarem a manter a paz de espírito durante esse período, e a todos que me colocaram em suas orações.

Agradeço à Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP), pelo apoio financeiro.

"E sabemos que todas as coisas contribuem juntamente para o bem daqueles que amam a Deus, daqueles que são chamados segundo o seu propósito."
(Apóstolo Paulo)

RESUMO

A crescente utilização de agentes conversacionais tem despertado um interesse cada vez maior entre pesquisadores, educadores e instituições de ensino em todo o mundo. Além disso, com a recente popularização dos Grandes Modelos de Linguagem (do inglês *Large Language Models* - LLMs), muitos estudos têm sido conduzidos no sentido de explorar a utilização dessas ferramentas para auxiliar no processo de ensino e aprendizagem. Esses sistemas têm a capacidade de compreender e processar enormes quantidades de dados, o que lhes permite, por exemplo, fornecer suporte individualizado aos alunos na resolução de exercícios.

Entretanto, é fundamental considerar que tais sistemas, geralmente, possuem um processo de raciocínio baseado em métodos estatísticos, como algoritmos de aprendizado de máquina, os quais podem produzir respostas incorretas em tarefas que exigem *raciocínio lógico*. Este trabalho tem como objetivo avaliar a habilidade de Grandes Modelos de Linguagem , na resolução de exercícios de Dedução Natural em Lógica Proposicional e Lógica de Predicados.

Para tanto, foram realizados experimentos utilizando os modelos *GPT 4.1-mini*, *GPT 4o* e *GPT 3.5-turbo* na resolução de exercícios de dedução natural. Esses experimentos foram conduzidos, *a priori*, sem treinamento e, posteriormente, foram realizados os experimentos no modelo *GPT 4.1-mini* treinado, a fim de verificar eventuais melhorias de desempenho. Os resultados apontam uma melhora significativa após o treinamento, embora o modelo ainda apresente um número considerável de erros. Portanto, caso seja empregado para essa tarefa, recomenda-se o uso com a devida prudência.

Palavras-chave: dedução natural; lógica proposicional; lógica de predicados; raciocínio lógico; grandes modelos de linguagem.

ABSTRACT

The growing use of conversational agents has sparked increasing interest among researchers, educators, and educational institutions worldwide. Moreover, with the recent popularization of Large Language Models (LLMs), numerous studies have been conducted to explore the use of these tools to support the teaching and learning process. These systems are capable of understanding and processing vast amounts of data, which allows them, for example, to provide individualized support to students in solving exercises.

However, it is essential to consider that such systems generally employ reasoning processes based on statistical methods, such as machine learning algorithms, which may produce incorrect answers in tasks requiring *logical reasoning*. This study aims to evaluate the ability of Large Language Models (LLMs) to solve Natural Deduction exercises in Propositional Logic and Predicate Logic.

To this end, experiments were carried out using the *GPT 4.1-mini*, *GPT 4o*, and *GPT 3.5-turbo* models to solve natural deduction exercises. These experiments were conducted *a priori*, without training, and subsequently, further experiments were performed on the trained *GPT 4.1-mini* model to assess potential performance improvements. The results indicate a significant improvement after training, although the model still exhibits a considerable number of errors. Therefore, if employed for this task, its use is recommended with due caution.

Keywords: natural deduction; propositional logic; predicate logic; logical reasoning; large language models.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A utilização de Grandes Modelos de Linguagem (LLMs, do inglês *Large Language Models*) (Zhao *et al.*, 2023) na educação tem despertado um crescente interesse de pesquisadores, educadores e instituições de ensino em todo o mundo (Tlili *et al.*, 2023; Kasneci *et al.*, 2023). A capacidade desses sistemas em compreender e processar grandes volumes de dados, além de sua habilidade em aprender e adaptar-se a novas informações, oferece oportunidades para auxiliar no processo de ensino-aprendizagem. Um dos principais benefícios de se utilizar tais ferramentas é a possibilidade de oferecer suporte individualizado aos alunos. Entretanto, é fundamental considerar que esses sistemas geralmente possuem um processo de raciocínio baseado em métodos estatísticos, como algoritmos de aprendizado de máquina (Russell, 2010) , o que pode resultar em *respostas incorretas* em determinadas situações, especialmente quando lidam com tarefas envolvendo *raciocínio lógico* (Liu *et al.*, 2023).

A Lógica para Computação (Huth; Ryan, 2004) é uma disciplina abordada em grande parte dos cursos de graduação na área de Tecnologia da Informação e Comunicação (TICs). O objetivo da lógica na computação é desenvolver linguagens para modelar situações do mundo e dos sistemas, de modo que possamos analisá-las formalmente, construindo argumentos sobre elas para serem apresentados e *justificados rigorosamente*. Geralmente, na ementa desta disciplina, são apresentadas duas lógicas clássicas, a saber: a Lógica Proposicional e a Lógica de Predicados (Huth; Ryan, 2004). Nesse contexto, A Dedução Natural é um dos conteúdos mais importantes, sendo utilizada para derivar *conclusões* a partir de sentenças dadas como verdadeiras (as quais são denominadas *premissas*), seguindo regras específicas (Pelletier, 1999; Huth; Ryan, 2004). A construção de uma prova de dedução natural é um exercício criativo: **não é óbvio quais regras aplicar e em qual ordem, a partir das premissas, para obter a conclusão desejada**.

Neste contexto, a questão de pesquisa central desse trabalho é:

"Os *LLMs são capazes de gerar respostas corretas na tarefa de solucionar exercícios de Dedução Natural em Lógica Proposicional e Lógica de Predicados?*".

Para responder a esta pergunta de pesquisa, propomos avaliar os modelos *GPT 4.1-mini*, *GPT 4o* e *GPT 3.5-turbo* na tarefa de realizar provas de dedução natural em Lógica Proposicional e Lógica de Predicados. Para isso, elaboramos bases de dados referentes a enunciados de exercícios de Dedução Natural, bem como suas respostas corretas, e avaliamos as respostas produzidas pelos modelos em corretas e incorretas. Inicialmente, avaliamos o

desempenho dos LLMs sem treinamento, para verificar a capacidade atual dos mesmos na resolução deste tipo de prova. Em seguida, os modelos foram treinados apresentando exemplos de enunciados e respostas corretas e, por fim, avaliamos a capacidade de aprendizado dos mesmos em relação a este tipo de prova.

1.1 Objetivos

1.1.1 Objetivo Geral

Avaliar o desempenho de Grandes Modelos de Linguagem *GPT 4.1-mini*, *GPT 4o* e *GPT 3.5-turbo* na realização de provas de dedução natural em lógica proposicional e lógica de predicados.

1.1.2 Objetivos Específicos

Como objetivos específicos temos:

- Criação de bases de dados de exercícios de Dedução Natural em Lógica Proposicional e Lógica de Predicados com suas respectivas respostas corretas;
- Avaliação e Comparação do desempenho dos modelos: *GPT 4.1-mini*, *GPT 4o* e *GPT 3.5-turbo* na tarefa de dedução natural sem treinamento;
- Avaliação e Comparação do desempenho do *GPT-4.1-mini* na tarefa de dedução natural sem treinamento e com treinamento.

1.1.3 Organização do Trabalho

O restante do trabalho está organizado como mostrado a seguir. No Capítulo 2, apresentaremos a fundamentação teórica do trabalho, consistindo nos conceitos de Lógica Proposicional, Lógica de Predicados, a ferramenta *NADIA* e os conceitos de Aprendizado de Máquina e Redes Neurais Transformacionais. No Capítulo 3, apontamos alguns trabalhos relacionados sobre o uso de LLMs em tarefas de raciocínio lógico. No Capítulo 4, descrevemos os passos metodológicos para desenvolvimento desse trabalho. No Capítulo 5, apresentamos os resultados obtidos. E, no Capítulo 6 são apresentadas as conclusões e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos fundamentais utilizados no desenvolvimento deste trabalho. Inicialmente iremos destacar os conceitos relacionados a Lógica: Lógica Proposicional (2.1), Lógica de Predicados (2.2) e iremos apresentar a ferramenta *NADIA* (2.3). Por fim, serão apresentados os conceitos relacionados a Aprendizado de Máquina (2.4) e Redes Neurais Transformacionais (2.5).

2.1 Lógica Proposicional

A Lógica Proposicional baseia-se em *proposições* ou *frases declarativas* que se pode argumentar sobre sua veracidade ou falsidade. As frases declarativas são consideradas como *atômicas* (e.g, "*o dólar sobe*", "*os produtos ficam mais caros*") de certa forma que podemos atribuir símbolos distintos $P, Q, R, \cdots$ a cada uma destas frases (e.g, P : "*o dólar sobe*", Q : "*os produtos ficam mais caros*"). Para codificar frases mais complexas usamos os conectivos lógicos: $\neg$ (negação), $\wedge$ (conjunção), $\vee$ (disjunção) e $\rightarrow$ (implicação) (Enderton, 1972). Por exemplo: $P \wedge Q$ codifica a frase "O dólar sobe *e* os produtos ficam mais caros"; $P \vee Q$ codifica a frase "O dólar sobe *ou* os produtos ficam mais caros."; $P \rightarrow Q$ representa "*se* O dólar sobe *então* os produtos ficam mais caros." e; $\neg P$: "O dólar não subiu".

O conjunto dos átomos proposicionais, juntamente com os conectivos e os símbolos '(', ')' formam o alfabeto da linguagem da Lógica Proposicional. A sintaxe da Lógica Proposicional pode ser definida por uma gramática na forma de *Backus Naur* (BNF - *Backus Naur Form*) como a seguir (Huth; Ryan, 2004):

$$\varphi ::= \bot \mid P \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi)$$

em que $\bot$ e P representam, respectivamente, a contradição e qualquer proposição atômica e cada ocorrência φ a direita de '::=' representa qualquer fórmula já construída.

A Dedução Natural é um sistema dedutivo que permite a construção de uma prova formal, estabelecendo uma conclusão φ a partir de um conjunto de premissas $\Gamma = \{\varphi_1, \varphi_2, \cdots, \varphi_n\}$, denotado por $\Gamma \vdash \varphi$, aplicando-se sucessivamente *regras de demonstração*. O trabalho de (Pelletier, 2000) compara o estilo de demonstrações de Dedução Natural de 33 livros-texto, sendo o estilo *Fitch* (caixas) o adotado na maioria deles. Neste estilo, as demonstrações são apresentadas de forma linear e sequencial, na qual cada uma das linhas da prova é numerada, tem uma afirmação e uma justificativa. As justificativas são definidas por serem *premissas* da

prova ou pela aplicação de uma das *regras do sistema dedutivo*. As subprovas dentro de uma prova maior têm *caixas* ao redor que servem para delimitar o escopo de hipóteses temporárias. Provas podem ter caixas dentro de caixas, ou pode-se abrir outras caixas depois de fechar outras, obedecendo as regras de demonstração. Uma fórmula só pode ser utilizada em uma prova em um determinado ponto se essa fórmula aconteceu anteriormente e se nenhuma caixa que contenha essa ocorrência da fórmula tenha sido fechada.

Uma demonstração em Dedução Natural é construída a partir da enumeração das premissas e da aplicação das *regras de Dedução Natural*. Em geral, cada conectivo possui uma regra para adicionar e outra regra para eliminar este conectivo. A construção de uma demonstração é um exercício criativo: **não é óbvio quais regras aplicar e em qual ordem, a partir das premissas, para obter a conclusão desejada**. A seguir, serão apresentadas as regras básicas de Dedução Natural em Lógica Proposicional, a saber: $\wedge$-introdução ($\wedge i$), $\wedge$-eliminação ($\wedge e$), $\rightarrow$-introdução ($\rightarrow i$), $\rightarrow$-eliminação ($\rightarrow e$), $\vee$-introdução ($\vee i$), $\vee$-eliminação ($\vee e$), $\bot$-eliminação ($\bot e$), $\neg$-introdução ($\neg i$), $\neg$-eliminação ($\neg e$), raa (redução ao absurdo) e copie.

2.1.1 Premissas

O primeiro passo em uma demonstração em Dedução Natural no estilo *Fitch* é enumerar as premissas da prova. A Figura 1 apresenta a estrutura geral da regra das premissas, na qual $\varphi_1, \varphi_2, \ldots, \varphi_n$ são representadas em uma linha cada, seguindo uma numeração sequencial e como justificativa "premissa".

Figura 1 – Enumeração das premissas.

1.	φ_1	premissa
2.	φ_2	premissa
$\vdots$	$\vdots$	$\vdots$
n.	φ_n	premissa
$\vdots$	$\vdots$	$\vdots$

Fonte: (Vasconcelos; Menezes, 2024).

2.1.2 Conjunção Introdução ($\wedge i$)

A regra da introdução da conjunção ($\wedge i$) é apresentada na Figura 2(a) (ou Figura 2(b)), na qual a fórmula $\varphi \wedge \psi$ pode ser concluída em uma linha p se φ e ψ foram demonstradas

nas linhas m (ou n) e n (ou m), respectivamente, anteriores a linha p e que não foram descartadas. A Figura 2(c) exibe a aplicação $\wedge i$ da fórmula $A \wedge B$ na linha 3 a partir das fórmulas A e B, definidas nas linhas 1 e 2, respectivamente, que são anteriores a linha 3.

Figura 2 – Conjunção Introdução ($\wedge i$)

$$\begin{array}{lll} \vdots & \vdots & \vdots \\ m. & \varphi & \\ \vdots & \vdots & \vdots \\ n. & \psi & \\ \vdots & \vdots & \vdots \\ p. & \varphi \wedge \psi & \wedge\text{i } m,n \end{array}$$

(a) Regra $\wedge i$

$$\begin{array}{lll} \vdots & \vdots & \vdots \\ m. & \psi & \\ \vdots & \vdots & \vdots \\ n. & \varphi & \\ \vdots & \vdots & \vdots \\ p. & \varphi \wedge \psi & \wedge\text{i } m,n \end{array}$$

(b) Regra $\wedge i$

$$\begin{array}{lll} 1. & A & \text{premissa} \\ 2. & B & \text{premissa} \\ 3. & A \wedge B & \wedge i\ 1,2 \end{array}$$

(c) $A, B \vdash A \wedge B$

Fonte: (Vasconcelos; Menezes, 2024)

2.1.3 Conjunção Eliminação ($\wedge e$)

A regra da eliminação da conjunção ($\wedge e$) é apresentada na Figura 3(a) (ou Figura 3(b)), na qual a fórmula φ (ou ψ) pode ser concluída na linha p a partir da eliminação à esquerda (ou à direita) da conjunção da fórmula $\varphi \wedge \psi$ da linha n (anterior a m e não foi descartada). A Figura 3(c) exibe uma aplicação da regra na qual A é obtida na linha 3 pela eliminação da conjunção à esquerda da fórmula $A \wedge B$ da linha 1.

Figura 3 – Conjunção Eliminação ($\wedge e$)

$$\begin{array}{lll} \vdots & \vdots & \vdots \\ m. & \varphi \wedge \psi & \\ \vdots & \vdots & \vdots \\ p. & \varphi & \wedge\text{e } m \end{array}$$

(a) Regra $\wedge e$ à esquerda

$$\begin{array}{lll} \vdots & \vdots & \vdots \\ m. & \varphi \wedge \psi & \\ \vdots & \vdots & \vdots \\ p. & \psi & \wedge\text{e } m \end{array}$$

(b) Regra $\wedge e$ à direita

$$\begin{array}{lll} 1. & A \wedge B & \text{premissa} \\ 2. & C & \text{premissa} \\ 3. & A & \wedge e\ 1 \\ 4. & A \wedge C & \wedge i\ 3,2 \end{array}$$

(c) $A \wedge B, C \vdash A \wedge C$

Fonte: (Vasconcelos; Menezes, 2024).

2.1.4 Implicação Eliminação (→e)

A regra da eliminação da implicação (→e), também conhecida como *Modus Ponens*, é apresentada na Figura 4(a) (ou Figura 4(b)), na qual a fórmula ψ pode ser concluída na linha p a partir da eliminação da implicação da fórmula $\varphi \to \psi$ da linha m (ou n) e φ da linha n (ou m), anteriores a p e não descartadas. A Figura 4 exibe uma aplicação da regra na qual a fórmula C é obtida na linha 4 pela eliminação da implicação das fórmulas $B \to C$ e B das linha 2 e 3, respectivamente.

Figura 4 – Implicação Eliminação ($\to e$)

⋮	⋮	⋮		⋮	⋮	⋮	
$m.$	$\varphi \to \psi$			$m.$	φ		
⋮	⋮	⋮		⋮	⋮	⋮	
$n.$	φ			$n.$	$\varphi \to \psi$		
⋮	⋮	⋮		⋮	⋮	⋮	
$p.$	ψ	→e m,n		$p.$	ψ	→e m,n	

(a) Regra $\to e$ (b) Regra $\to e$

1.	$A \wedge B$	premissa
2.	$B \to C$	premissa
3.	B	$\wedge e$ 1
4.	C	$\to e$ 2,3

(c) $A \wedge B, B \to C \vdash C$

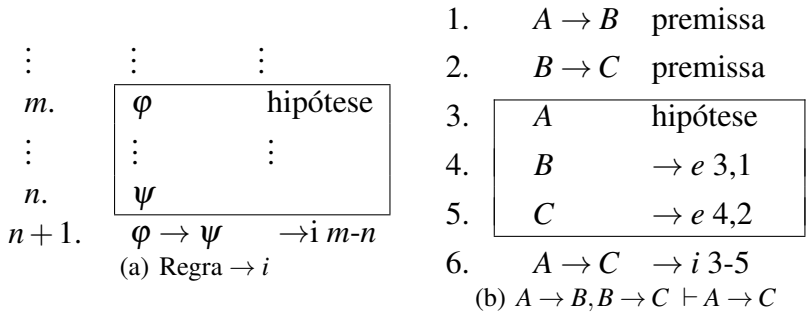
Fonte: (Vasconcelos; Menezes, 2024).

2.1.5 Implicação Introdução (→i)

A regra da introdução da implicação (→i) constrói condicionais que não ocorrem como premissas. Para construção de um condicional é necessário realizar *raciocínio hipotético*, isto é, supor *temporariamente* que uma dada fórmula é verdadeira. Chamamos esta fórmula de *hipótese*. Assim, utilizamos *caixas de demonstração*, que servem para delimitar o *escopo da hipótese temporária*. Observe na Figura 5(a) que para provar o condicional $\varphi \to \psi$ na linha $n+1$, colocamos φ como *hipótese* no topo de uma caixa (linha m), aplicamos um número finito de regras de forma a obter ψ na linha n. Todo o raciocínio para obter ψ depende da veracidade de φ e, por isso, as fórmulas resultante deste raciocínio ficam delimitadas na caixa. Na linha seguinte $(n+1)$ podemos aplicar a regra $\to i$ para obter $\varphi \to \psi$, sendo que este condicional não mais depende da hipótese φ. Na justificativa da linha $n+1$ utilizamos o nome da regra seguido

das linhas inicial e final da caixa ($\rightarrow i$ *m-n*). A Figura 5(b) exibe uma aplicação da regra na qual a fórmula $A \rightarrow C$ é obtida na linha 6 a partir da caixa das linhas 3 a 5 em que A é a hipótese e C é a conclusão da caixa.

Figura 5 – Implicação Introdução ($\rightarrow i$)



(a) Regra $\rightarrow i$

(b) $A \rightarrow B, B \rightarrow C \vdash A \rightarrow C$

Fonte: (Vasconcelos; Menezes, 2024).

Demonstrações podem ter caixas dentro de caixas, ou pode-se abrir novas caixas depois de fechar outras. No entanto, existem regras sobre quais fórmulas podem ser utilizadas em que ponto na demonstração. Em geral, só podemos usar uma fórmula em um determinado ponto se esta fórmula ocorre *antes* desse ponto e se nenhuma caixa que contenha a ocorrência desta fórmula tenha sido fechada (Huth; Ryan, 2004).

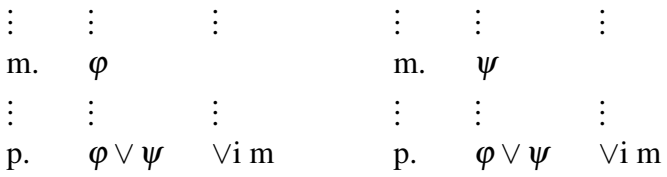
2.1.6 Disjunção Introdução

A regra da introdução da disjunção ($\vee i$) é a apresentada na Figura 6(a) (ou Figura 6(b)), na qual dizemos que temos $\varphi \vee \psi$ em uma linha p se φ (ou ψ) ocorre em uma linha m anterior a p e que não foi descartada. A Figura 6(c) ilustra a aplicação da introdução da disjunção na linha 3 com a introdução de $A \vee B$ a partir da fórmula A definida na linha 2.

2.1.7 Disjunção Eliminação ($\vee e$)

A regra da disjunção eliminação ($\vee e$) é a apresentada na Figura 7(a), na qual dizemos que podemos concluir uma fórmula α na linha $p+1$ se eliminarmos a disjunção da fórmula $\varphi \vee \psi$ na linha m e se fizermos a suposição de φ em uma caixa, na linha m, e a suposição de ψ em outra caixa, na linha $n+1$, tal que ambas as caixas tenham como conclusão α, nas linhas n e p, respectivamente, por meio de uma sequência finita de passos (regras). Note que essa regra se assemelha a introdução da implicação no sentido de que fazemos raciocínio hipotético, nas caixas de $(m+1)-n$ e $(n+1)-p$. A Figura 7(b) ilustra a aplicação da eliminação da disjunção

Figura 6 – Disjunção Introdução ($\vee i$)



(a) Regra $\vee i$ à esquerda (b) Regra $\vee i$ à direita

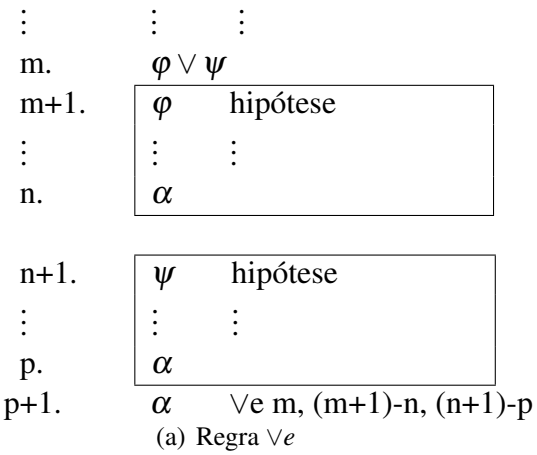
1.	$(A \vee B) \to C$	premissa
2.	A	hipótese
3.	$A \vee B$	$\vee$ i 2
4.	C	$\to$ e 3,1
5.	$A \to C$	$\to$ i 2-4

(c) $(A \vee B) \to C \vdash A \to C$

Fonte: (Vasconcelos; Menezes, 2024).

na linha 8, na qual concluímos C, a partir da disjunção de $A \vee B$ na linha 3 e das caixas $4-5$ e $6-7$ onde supomos A na linha 4 e concluímos C na linha 5 e supomos B na linha 6 e concluímos C na linha 7.

Figura 7 – Disjunção Eliminação ($\vee e$)



(a) Regra $\vee e$

1.	$A \to C$	premissa
2.	$B \to C$	premissa
3.	$A \vee B$	premissa
4.	A	hipótese
5.	C	$\to$ e 4,1
6.	B	hipótese
7.	C	$\to$ e 6,2
8.	C	$\vee$ e 3, 4-5, 6-7

(b) $A \to C, B \to C, A \vee B \vdash C$

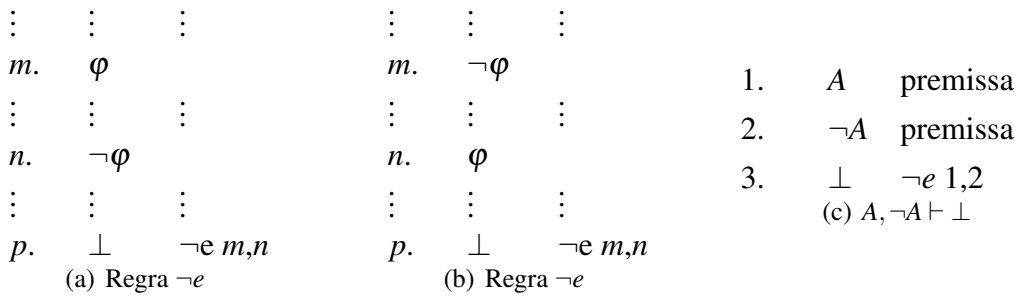
Fonte: (Vasconcelos; Menezes, 2024).

2.1.8 Negação Eliminação ($\neg e$)

A regra da negação eliminação ($\neg$e) envolve a noção de *contradição*. Contradições são expressões da forma $\varphi \wedge \neg\varphi$ ou $\neg\varphi \wedge \varphi$ onde φ é qualquer fórmula da Lógica Proposicional. A fórmula $\bot$ é utilizada para denotar uma contradição e este fato é expresso na regra $\neg e$. Na Figura 8(a) (ou Figura 8(b)) temos a fórmula φ na linha m (ou n) e a sua negação $\neg\varphi$ na linha n (ou m) sendo combinadas para aparecimento da contradição $\bot$ na linha p com a aplicação da

regra $\neg e$. A Figura 8(c) exibe uma aplicação da regra na qual a contradição $\bot$ é obtida na linha 3 devido às fórmulas A na linha 1 e $\neg A$ na linha 2.

Figura 8 – Negação Eliminação ($\neg e$)



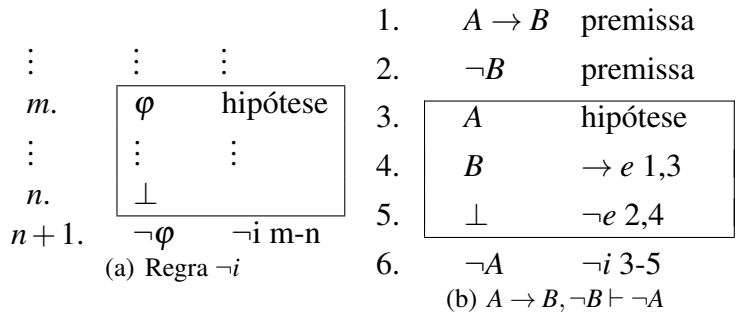
(a) Regra $\neg e$ (b) Regra $\neg e$ (c) $A, \neg A \vdash \bot$

Fonte: (Vasconcelos; Menezes, 2024).

2.1.9 *Negação Introdução ($\neg i$)*

A regra da negação introdução ($\neg i$) é a apresentada na Figura 9(a). Esta é uma regra que envolve raciocínio hipotético e contradição. Se tomarmos φ como hipótese (linha m) e, após a aplicação de um número finito de regras, chegarmos a uma contradição $\bot$ na linha n, significa que a hipótese não pode ser verdadeira. Desse modo, finalizamos nosso raciocínio hipotético introduzindo a negação na hipótese e obtendo $\neg\varphi$ na linha $n+1$. A Figura 9(b) mostra um exemplo da aplicação da regra $\neg i$ em para provamos $\neg A$ na linha 6, assumimos A como hipótese no topo da caixa na linha 3 e chegamos a uma contradição no final da caixa na linha 5.

Figura 9 – Negação Introdução ($\neg i$)



(a) Regra $\neg i$ (b) $A \to B, \neg B \vdash \neg A$

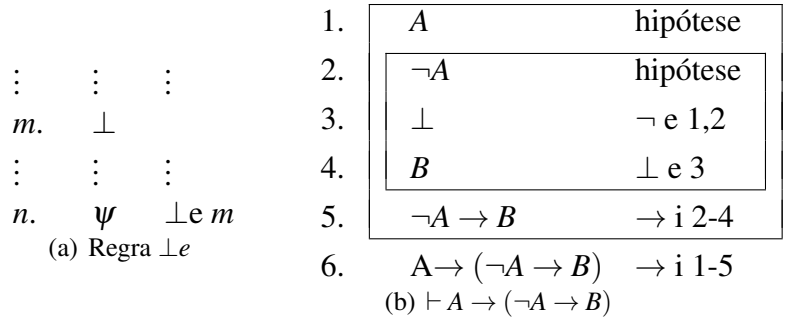
Fonte: (Vasconcelos; Menezes, 2024).

2.1.10 *Contradição Eliminação ($\bot e$)*

A regra da contradição eliminação ($\bot e$) é exibida na Figura 10(a), na qual podemos concluir uma fórmula qualquer ψ na linha n se demonstramos em uma linha m, anterior a n, a

contradição. A Figura 10(b) apresenta a demonstração de B na linha 4 a partir da eliminação da contradição $\bot$ da linha 3.

Figura 10 – Regra da Eliminação por Contradição ($\bot e$)



(a) Regra $\bot e$

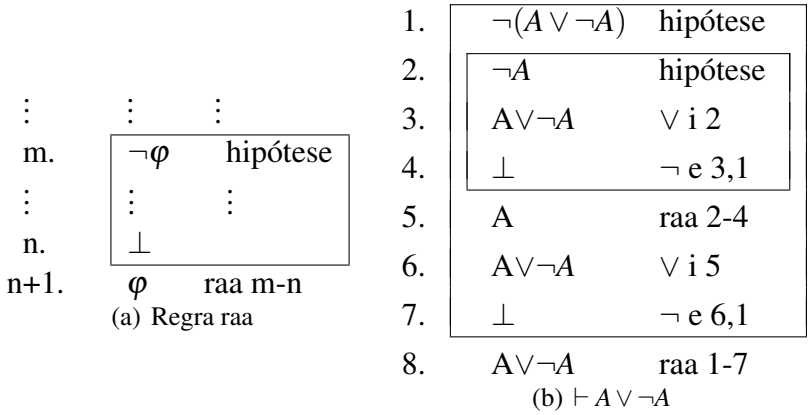
(b) $\vdash A \to (\neg A \to B)$

Fonte: (Vasconcelos; Menezes, 2024).

2.1.11 Redução ao Absurdo (raa)

A regra de redução ao absurdo é apresenta na Figura 11(a), na qual para provarmos uma fórmula φ em uma linha $n+1$, iremos supor temporariamente a negação da fórmula, $\neg\varphi$, em uma caixa que inicia na linha m e que conclui a contradição, $\bot$, na linha n, após uma sequência de aplicações de regras. A Figura 11(b) exibe a demonstração de $A \vee \neg A$, também conhecido como terceiro-excluído. Para provarmos $A \vee \neg A$ na linha 8, fazemos a suposição de $\neg(A \vee \neg A)$, na linha 1 (início da caixa) e concluímos a contradição $\bot$, na linha 7 (fim da caixa).

Figura 11 – Redução ao Absurdo (raa)



(a) Regra raa

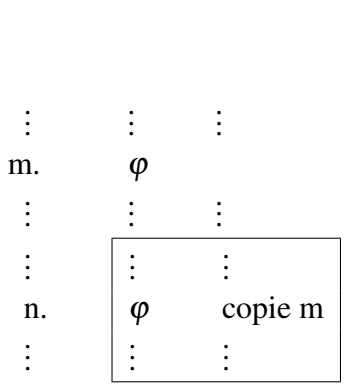
(b) $\vdash A \vee \neg A$

Fonte: (Vasconcelos; Menezes, 2024).

2.1.12 Regra Copie

A regra copie, apresentada na Figura 12(a), é necessária, no estilo de Fitch, para permitir concluir uma caixa com uma fórmula que já apareceu anteriormente na demonstração. Por exemplo, para demonstrar $A \rightarrow B$, veja Figura 12(b), na linha 10, é preciso que a caixa que justifica a introdução da implicação inicie com A, linha 8, e termine com B, linha 9. Ocorre que a justificativa de B já havia sido realizada e, portanto, a justificativa da linha 9 é a cópia da fórmula da linha 7.

Figura 12 – Regra Copie



(a) Regra copie

1.		$\neg A \vee B$	premissa
2.		$\neg A$	hipótese
3.		A	hipótese
4.		$\perp$	$\neg$ e 2,3
5.		B	$\perp$ e 4
6.		A$\rightarrow B$	$\rightarrow$ i 3-5
7.		B	hipótese
8.		A	hipótese
9.		B	copie 7
10.		A$\rightarrow B$	$\rightarrow$ i 8-9
11.		A$\rightarrow B$	$\vee$ e 1, 2-6, 7-10

(b) $\neg A \vee B \vdash A \rightarrow B$

Fonte: (Vasconcelos; Menezes, 2024).

2.2 Lógica de Predicados

A Lógica de Predicados surgiu da necessidade de representar aspectos mais ricos da linguagem natural do que apenas as frases declarativas da Lógica Proposicional. Esta lógica foi desenvolvida de forma independente por Gottlob Frege e Charles Sanders Peirce (Ferreirós, 2001; Hammer, 1998). As fórmulas da lógica de predicados são compostas por predicados, constantes, variáveis, símbolos funcionais, quantificadores e conectivos.

Um *predicado* é um símbolo que representa uma propriedade ou uma relação entre objetos do mundo. Por exemplo, na fórmula $Estudante(maria)$ o predicado *Estudante* expressa a propriedade de que um símbolo constante *maria* é uma estudante. Já na fórmula *Professor(joão, maria)* o predicado *Professor* relaciona as constantes *joão* e *maria*, representando que *joão* é

professor de *maria*. Essas relações também podem ser estabelecidas para variáveis de um certo domínio. Além disso, a Lógica de Predicados inclui *quantificadores* com os quais é possível representar a noção de "*existe*" (*existe um x que é estudante*) e "*para todo*" (*todos os x são estudantes*). Também é possível representar *símbolos funcionais* que são associados a objetos do mundo. Por exemplo, *professor(maria)*[1] é uma função que devolve a constante *joão*.

Devido ao seu maior poder de expressividade, a linguagem da Lógica de Predicados é mais complexa do que a da Lógica Proposicional. Nesta lógica há dois tipos de expressões: os termos e as fórmulas. Termos são expressões que relacionam-se a objetos de um domínio: indivíduos como *joão* e *maria* são exemplos; variáveis como x e; símbolos funcionais que também referem-se a objetos, como exemplo *professor(maria)* refere-se ao objeto *joão*. Já as fórmulas são expressões que podem ser relacionadas a um valor verdade: verdadeiro ou falso.

O conjunto dos termos da linguagem da Lógica de Predicados pode ser definido por uma gramática na forma BNF (Huth; Ryan, 2004):

$$t ::= x \mid c \mid f(t, \cdots, t),$$

em que x é uma variável, c é uma constante e f é um símbolo funcional com aridade $n > 0$.

Já as fórmulas da Lógica de Predicados podem ser definidas em BNF, como a seguir (Huth; Ryan, 2004):

$$\varphi ::= \bot \mid P(t_1, t_2, \cdots, t_n), \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\forall x\ \varphi) \mid (\exists x\ \varphi),$$

em que $\bot$ e P representam, respectivamente, a contradição e um símbolo predicado n-ário com $n \geq 0$, $t_1, \cdots t_n$ são termos e x é uma variável.

Nas fórmulas da Lógica de Predicados, as variáveis podem ocorrer *livres* ou *ligadas* (presas). Considere a seguinte fórmula φ:

$$(\exists x(P(x) \wedge Q(x)) \vee R(x, y).$$

Dizemos que a variável x é *livre* em φ se ela não é capturada por um quantificador ($\forall x$ ou $\exists x$) em φ, isto é, se a variável x não está no escopo de um quantificador $\forall x$ ou $\exists x$. Caso contrário, dizemos que a variável é ligada (presa) na fórmula φ. Na fórmula anterior, as duas primeiras ocorrências da variável x são *ligadas* (estão no escopo do quantificador $\exists x$) e a terceira ocorrência

[1] Representamos funções com letras minúsculas para diferenciar dos predicados que são representados com letras maiúsculas.

desta variável é *livre* (não está no escopo do quantificador $\exists x$). Já a única ocorrência da variável y é *livre*.

Definimos φ_t^x como a expressão obtida da fórmula φ pela substituição da variável x, sempre que ela ocorrer livre em φ, pelo termo t. Por exemplo, $(H(x) \to \forall x M(x))_y^x = (H(y) \to \forall x M(x))$. Dizemos que o termo t é substituível para x em φ se não existe uma variável y em t tal que ela é capturada por um ($\forall y$ ou $\exists y$) quantificador de φ_t^x. Por exemplo, o termo z é substituível para y em $\forall x P(x,y)$. Por outro lado, o termo x não é substituível para y em $\forall x P(x,y)$.

As demonstrações de Dedução Natural para a Lógica de Predicados são semelhantes às demonstrações para a Lógica Proposicional com a adição de novas regras para tratar dos *quantificadores*. Deste modo, qualquer regra de Dedução Natural para a Lógica Proposicional pode ser aplicada também para as fórmulas da Lógica de Predicados. A seguir, serão descritas as regras para introdução e eliminação dos quantificadores universal e existencial.

2.2.1 Eliminação do Universal ($\forall e$)

A regra da eliminação do universal ($\forall e$) é apresentada na Figura 13(a), na qual a fórmula φ_t^x pode ser concluída em uma linha p se $\forall x \varphi$ foi demonstrada na linha m, desde que o termo t seja substituível pela variável x em φ. A Figura 13(b) exibe a aplicação $\forall e$ da fórmula $H(s) \to M(s)$ na linha 3 a partir da fórmula $\forall x(H(x) \to M(x))$, definida na linha 1.

Figura 13 – Regra Universal Eliminação ($\forall e$)

⋮	⋮	⋮
m.	$\forall x \varphi$	
⋮	⋮	⋮
p.	φ_t^x	$\forall$e m

t é substituível para x em φ

(a) Regra Universal Eliminação

1.	$\forall x(H(x) \to M(x))$	premissa
2.	H(s)	premissa
3.	H(s) $\to M(s)$	$\forall e$ 1
4.	M(s)	$\to e$ 2,3

(b) $\forall x(H(x) \to M(x)), H(s) \vdash M(s)$

Fonte: (Vasconcelos; Menezes, 2024).

2.2.2 Introdução do Universal ($\forall i$)

A regra da introdução do universal ($\forall i$) é apresentada na Figura 14(a), na qual para provarmos $\forall x \varphi$, na linha $n+1$, iremos supor que para uma variável a qualquer (arbitrária) em uma caixa que inicia na linha m e que conclui φ_a^x, na linha n. Dizemos que a variável a é qualquer se ela é uma variável nova na linha m, ou seja, a não ocorre como variável livre de

qualquer fórmula que aconteça anteriormente a linha m que não esteja em uma caixa fechada. Na Figura 14(b), ilustramos a introdução do universal na linha 7 para provar $\forall x M(x)$ a partir da escolha de a, no início da caixa, na linha 3, e demonstramos $M(a)$ ao final da caixa, na linha 6. Note que a variável a escolhida não é uma variável livre das fórmulas das linhas 1 e 2.

Figura 14 – Regra Universal Introdução ($\forall i$)

⋮	⋮	⋮
m.	a	
⋮	⋮	⋮
n.	$\varphi(a)$	
n+1.	$\forall x \varphi(x)$	$\forall$i m-n

a é uma variável nova

(a) Universal Introdução ($\forall i$)

1.	$\forall x(H(x) \rightarrow M(x))$	premissa
2.	$\forall x H(x)$	premissa
3.	a	
4.	H(a)$\rightarrow M(a)$	$\forall e$ 1
5.	H(a)	$\forall e$ 2
6.	M(a)	$\rightarrow e$ 5,4
7.	$\forall x M(x)$	$\forall i$ 3-6

(b) $\forall x(H(x) \rightarrow M(x)), \forall x H(x) \vdash \forall x M(x)$

Fonte: (Vasconcelos; Menezes, 2024).

2.2.3 Introdução do Existencial ($\exists i$)

A regra da introdução do existencial ($\exists i$) é apresentada na Figura 15(a), na qual a fórmula $\exists x \varphi$ pode ser concluída em uma linha p se φ_t^x foi demonstrada na linha m, desde que o termo t seja substituível para a variável x em φ. A Figura 15(b) exibe a aplicação $\exists i$ da fórmula $\exists x P(x)$ na linha 3 a partir da fórmula $P(a)$, definida na linha 1.

Figura 15 – Regra Existencial Introdução ($\exists i$)

⋮	⋮	⋮
m.	φ_t^x	
⋮	⋮	⋮
p.	$\exists x \varphi$	$\exists$i m

t é substituível para x em φ

(a) Existencial ($\exists i$)

1.	P(a)	premissa
2.	$\exists x P(x) \rightarrow B$	premissa
3.	$\exists x P(x)$	$\exists$ i 1
4.	B	$\rightarrow$ e 3,2

(b) $P(a), \exists x P(x) \rightarrow B \vdash B$

Fonte: (Vasconcelos; Menezes, 2024).

2.2.4 Eliminação do Existencial ($\exists e$)

A regra da eliminação do existencial ($\exists e$) é apresentada na Figura 16(a), na qual para provarmos uma fórmula α, na linha $p+1$, iremos eliminar o existencial da fórmula $\exists x \varphi$,

na linha m, supondo a fórmula φ_a^x para alguma variável a em uma caixa que inicia na linha n e que concluiu com uma fórmula α ao final da caixa, na linha p, desde que a não ocorra na conclusão α. Nesta regra sabemos que a fórmula φ vale para algum elemento. Entretanto, não podemos assumir nenhuma propriedade específica para esta variável. Assim, a variável a deve ser uma variável nova na linha n, ou seja, a não ocorre como variável livre de qualquer fórmula que aconteça anteriormente a linha n que não esteja em uma caixa fechada e nem pode estar na conclusão α da regra. Na Figura 16(b), ilustramos a eliminação do existencial para concluir $\perp$ na linha 9, a partir da fórmula $\exists x H(x)$, na linha 3, e pela caixa que inicia na linha 4 com a suposição da fórmula $H(a)$ com a (nova) variável a e que termina a caixa na linha 8 com a conclusão $\perp$.

Figura 16 – Regra Existencial Eliminação ($\exists e$)

⋮	⋮	⋮
m.	$\exists x \varphi(x)$	
⋮	⋮	⋮
n.	a $\varphi(a)$	hipótese
⋮	⋮	⋮
p.	α	
p+1.	α	$\exists$e m, n-p

a é uma variável nova
a não ocorre em α

(a) Existencial Eliminação ($\exists e$)

1.	$\forall x(H(x) \rightarrow M(x))$		premissa
2.	$\forall x \neg M(x)$		premissa
3.	$\exists x H(x)$		hipótese
4.	a	H(a)	hipótese
5.		H(a)$\rightarrow M(a)$	$\forall e$ 1
6.		M(a)	$\rightarrow e$ 4,5
7.		$\neg M(a)$	$\forall e$ 2
8.		$\perp$	$\neg e$ 6,7
9.	$\perp$		$\exists e$ 3, 4-8
10.	$\neg \exists x H(x)$		$\neg i$ 3-9

(b) $\forall x(H(x) \rightarrow M(x)), \forall x \neg M(x) \vdash \neg \exists x H(x)$

Fonte: (Vasconcelos; Menezes, 2024).

2.3 NADIA

A ferramenta NADIA (*Natural Deduction Proof Assistant*), é um assistente de provas para o sistema de Dedução Natural em Lógica Proposicional e Lógica de Predicados, no estilo de Fitch (Vasconcelos; Menezes, 2024). A ferramenta verifica automaticamente se a demonstração está correta e, caso contrário, exibe os erros encontrados.NADIA também possui uma linguagem própria de escrita acessível. No Capítulo 7 podemos observar como se dar a conversão de um texto em LaTeX para a linguagem aceita pela ferramenta NADIA, bem como, a explicação de como se constrói provas de dedução natural nessa linguagem.

– Os átomos e os predicados são escritos em letras maiúsculas (e.g.‘A’,‘B’,‘H(x)’).

– As variáveis são escritas com a primeira letra em minúsculo, podendo ser seguida de letras e números (e.g. ‘x’, ‘x0’,‘xP0’).
– Os símbolos de $\bot, \vdash$ e os conectivos $\neg, \wedge, \vee, \rightarrow$ são escritos por ‘@’, ‘|-’, ‘∼’, ‘&’, ‘|’, ’->’ respectivamente.
– As fórmulas com o ∀x e ∃x serão representadas por ‘Ax’ e ‘Ex’ (‘A’ e ‘E’ seguidos da variável ‘x’). Por exemplo, ‘Ax (H(x)->M(x))’ representa $\forall x(H(x) \rightarrow M(x))$.
– A ordem de precedência dos quantificadores e dos conectivos lógicos é definida por $\neg, \forall, \exists, \wedge, \vee e \rightarrow$ com alinhamento à direita. Por exemplo:
  – A fórmula ‘∼A&B->C’ representa a fórmula $(((\neg A) \wedge B) \rightarrow C)$.
  – O teorema‘∼Ax P(x) |- Ex ∼P(x)’ representa o teorema $(\neg((\forall x)P(x))) \vdash ((\exists x)(\neg P(x)))$.
– As palavras Premissa e Hipótese são representadas por ‘pre’ e ‘hip’, respectivamente.

A Figura 17 exibe o Teorema $A \rightarrow B, B \rightarrow C \vdash A \rightarrow C$, escrito de três formas distintas: escrita e compilada em LaTeX (Figura 17(a)), na linguagem aceita pela ferramenta NADIA (Figura 17(b)) e na escrita em LaTeX de forma original não compilada (Figura 17(c)).

Figura 17 – Exemplo de Escrita do Teorema $A \rightarrow B, B \rightarrow C \vdash A \rightarrow C$

1.	$A \rightarrow B$	premissa
2.	$B \rightarrow C$	premissa
3.	A	hipótese
4.	B	$\rightarrow e$ 3, 1
5.	C	$\rightarrow e$ 4, 2
6.	$A \rightarrow C$	$\rightarrow i$ 3-5

(a) Prova escrita e compilada em LaTeX

```
1. A -> B pre
2. B -> C pre
3. { A hip
4. B ->e 3,1
5. C ->e 4,2
}
6. A -> C ->i 3-5
```

(b) Prova escrita na linguagem aceita pela ferramenta NADIA

```
\begin{logicproof}{6}
    A \rightarrow B & premissa\\
    B \rightarrow C & premissa\\
    \begin{subproof}
        A & hipótese \\
        B & $\rightarrow e$ 3,1\\
        C & $\rightarrow e$ 4,2
    \end{subproof}
    A \rightarrow C & $\rightarrow i$ 3-5
\end{logicproof}
```

(c) Prova escrita em LaTeX

Fonte: Próprio autor (2025)

O Quadro 1 apresenta um resumo da correspondência entre os símbolos escritos em LaTeX e na Linguagem aceita pela ferramenta NADIA. As respostas escritas na linguagem aceita pela ferramenta NADIA podem ser convertidas para o LaTeX utilizando a implementação da ferramenta NADIA disponível de forma online[2].

Quadro 1 – Correspondência entre símbolos, comandos em LaTeX e no NADIA

Símbolo	$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\forall x$	$\exists x$	$\bot$	premissa	hipótese	caixa	$\vdash$
LaTeX	`\lnot`	`\land`	`\lor`	`\rightarrow`	`\forall x`	`\exists x`	`\bot`	premissa	hipótese	caixa	`\vdash`
NADIA	`~`	`&`	`|`	`->`	`Ax`	`Ex`	`@`	pre	hip	`{}`	`-`

Fonte: (Vasconcelos; Menezes, 2024).

2.4 Aprendizado de Máquina

Aprendizado de Máquina é uma subárea da Inteligência Artificial que estuda modelos e algoritmos capazes de aprender a partir de dados.

As *Redes Neurais Artificiais* (RNAs) fundamentam-se na estrutura e funcionamento do cérebro humano, consistindo em um conjunto interconectado de unidades de processamento chamadas neurônios artificiais, que são organizados em camadas. Cada neurônio recebe entradas, aplica uma função de ativação e gera uma saída. As informações são passadas por meio das camadas até que a saída seja gerada. As RNAs são usadas em uma enorme variedade de aplicações, como reconhecimento de padrões, classificação, regressão, visão computacional e processamento de linguagem natural (Russell, 2010). A Figura 18 ilustra a arquitetura básica de uma RNA com uma camada de entrada, duas camadas ocultas e uma camada de saída.

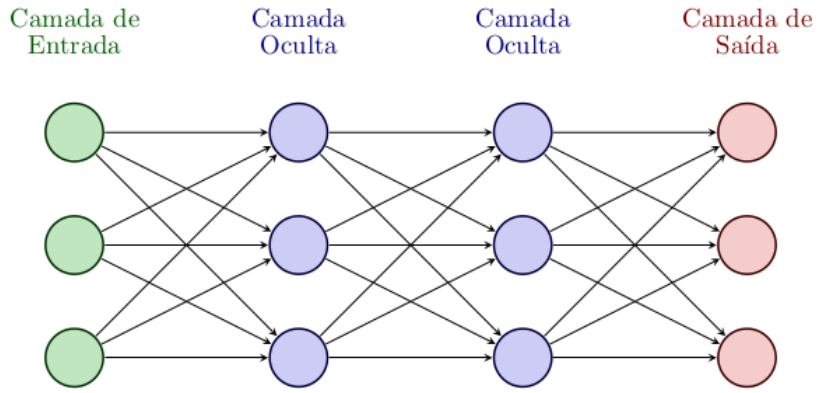
A arquitetura da rede neural está intimamente relacionada ao tipo de problema que se pretende resolver: certas arquiteturas são mais adequadas para certos tipos de dados e tarefas. Como exemplos de arquiteturas de redes neurais temos: *Multilayer Perceptrons* (MLPs), CNNs (Redes Neurais Convolucionais), RNNs (Redes Neurais Recorrentes) e *Transformers*.

2.5 Redes Neurais Transformacionais

As Redes Neurais Transformacionais (ou simplesmente *Transformers*) (Vaswani *et al.*, 2017) são uma arquitetura de redes neurais que têm desempenhado um papel fundamental no Processamento de Linguagem Natural (PLN). Os transformers utilizam mecanismos de *atenção* para permitir que cada elemento de entrada se comunique diretamente com todos os

[2] https://sistemas.quixada.ufc.br/nadia/index.jsp

Figura 18 – Arquitetura de uma RNA com múltiplas camadas.



Fonte: Próprio autor (2025).

outros elementos da rede, o que fornece a capacidade de capturar relações de longo alcance nas sequências e lidar com contextos complexos. Um mecanismo de atenção calcula a importância de cada parte de uma sequência de entrada, em relação as demais, para o uso em questão. Os *Tranformers* são a arquitetura dos principais *Large Language Models* atuais, tais como o *GPT-4.1-mini, GPT-4o e GPT-3.5-turbo*.

O ChatGPT (OpenAI, 2021) é um chatbot desenvolvido pela OpenAI que utiliza a rede neural transformacional *Generative Pre-trained Transformer* (GPT) (Brown *et al.*, 2020). Ele foi treinado com bilhões de palavras e trechos de texto de várias fontes na internet, permitindo que a ferramenta aprendesse a gramática, o estilo e o contexto de diferentes formas de comunicação humana. Embora seja capaz de produzir respostas incrivelmente impressionantes em determinadas situações, ele também pode gerar respostas incorretas em outras. A confiabilidade das respostas do ChatGPT depende da qualidade e da precisão dos dados com os quais o modelo foi treinado.

2.5.1 Fine-tuning

Apesar de os LLMs serem treinados em grandes quantidades de texto e apresentarem um bom desempenho em diversas tarefas, em contextos específicos - como a resolução de problemas de dedução natural, que exige uso de regras e formatos rígidos - podem produzir resultados insatisfatórios, conforme observado em nossos trabalhos anteriores (Martins *et al.*, 2023; Martins *et al.*, 2025). Para mitigar essas limitações, podem ser empregadas diferentes estratégias para melhorar o desempenho de determinado modelo em uma tarefa específica como: melhorar a qualidade das mensagens de entrada e o *Fine-tuning* do modelo. O *Fine-tuning* permite treinar um modelo, previamente treinado, com exemplos para seu caso de uso específico

e criar um modelo personalizado que produz de forma mais confiável o estilo e o conteúdo desejados. O ajuste fino supervisionado tem quatro partes principais[3]:

- Crie seu conjunto de dados de treinamento.
- Carregue um conjunto de dados de treinamento contendo exemplos de entrada e saída desejada.
- Crie uma tarefa de fine-tuning para um modelo base usando seus dados de treinamento.
- Avalie seus resultados usando o modelo ajustado.

Inicialmente se deve criar um conjunto de dados de treinamento, no qual se utiliza exemplos de dados do domínio escolhido, depois se carrega esses dados na plataforma no formato requerido (JSONL) como podemos observar na Figura 19. Em seguida se deve criar a tarefa de *Fine-tuning* propriamente dita ajustando os hiperparâmetros disponibilizados, são eles: número de épocas, *batch size* e *learning rate multiplier*. O número de épocas define a quantidade de vezes que o conjunto de treino passa pelo modelo completo, o batch size se refere a quantas amostras são processadas em paralelo antes de atualizar os pesos e por fim learning rate multiplier é o fator de ajuste da taxa de aprendizado padrão do modelo. Ao final do treino o modelo ajustado é disponibilizado para avaliação com novos dados de entrada.

Figura 19 – Exemplo de mensagens em JSONL para *fine-tuning*.

```
{"messages": [
{"role": "user", "content": "Qual é a capital da França?"},
{"role": "assistant", "content": "A capital da França é Paris."}
]}
{"messages": [
{"role": "user", "content": "Qual é a capital da Inglaterra?"},
{"role": "assistant", "content": "A capital da Inglaterra é Londres."}
]}
```

Fonte: Adaptado da documentação da OpenAI (2025).

2.5.2 Comunicação com os Modelos

Para a comunicação com os modelo foi utilizada a API (do inglês, *Application Programming Interface*) disponibilizada pela *OpenAI*[4], que foi acessada por meio de sua biblioteca na linguagem de programação *Python*. O modelo recebe uma conversa, que consiste em uma lista de mensagens como entrada e retorna uma mensagem gerada como saída.

[3] https://platform.openai.com/docs/guides/supervised-fine-tuning
[4] https://platform.openai.com/docs/api-reference

Na Figura 20, podemos observar um código simples na linguagem *Python* que estabelece uma comunicação com a API do ChatGPT utilizando o modelo `gpt-4.1-mini`. No exemplo, cria-se a variável `client`, que recebe uma instância do cliente autenticada com a chave de API (linha 3). O método `client.chat.completions.create()` envia ao modelo uma lista de mensagens (`messages`), onde cada mensagem é composta por um papel ("role") e o conteúdo da mensagem ("content"). O papel indica quem está falando, nesse caso usuário (`user`) ou modelo (`assistant`). No exemplo, a conversa possui uma mensagem do usuário, seguida por uma resposta do assistente e, depois, outra mensagem do usuário, simulando uma conversa. O resultado dessa chamada (a resposta da última mensagem) é armazenado na variável `response`. Para exibir apenas o texto da resposta, utiliza-se `print(response.choices[0].message["content"])` (linha 15).

Figura 20 – Exemplo simples de código Python.

```
1  from openai import OpenAI
2
3  client = OpenAI(api_key="SUA_CHAVE_API_AQUI")
4
5  # Envio de mensagem para o modelo
6  response = client.chat.completions.create(
7      model="gpt-4.1-mini",
8      messages=[
9      {"role": "user", "content": primeira_mensagem_usuario},
10     {"role": "assistant", "content":resposta_assistente},
11     {"role": "user", "content":segunda_mensagem_usuario}]
12 )
13
14 # resposta
15 print(response.choices[0].message["content"])
```

Fonte: Próprio autor (2025)

3 TRABALHOS RELACIONADOS

Neste capítulo serão apresentadas algumas abordagens que utilizam LLMs para resolução de exercícios de Lógica Proposicional e Lógica de Predicados. Os artigos foram encontrados por meio da busca no mecanismo de pesquisa *Google Scholar*[1], utilizando as seguintes *strings* de busca: *"natural deduction"* + *"large language models"*.

3.1 LeanDojo: Theorem Proving with Retrieval-Augmented Language Models

Em (Yang *et al.*, 2024) é apresentado o LeanDojo[2] que consiste em um conjunto de ferramentas, modelos e *benchmarks* de código aberto para prova de teoremas usando o assistente de provas *Lean* (Moura *et al.*, 2015). Também é apresentado o *ReProver*, um provador baseado em *LLM*, com baixa demanda de recursos computacionais e que necessita de pouco tempo de treinamento (alguns dias de GPUs). Os autores apresentam o conceito de Provadores de Teorema Iterativos (do inglês ITP - *Interactive Theorem Proving*), nos quais as provas são construídas por especialistas humanos interagindo com ferramentas tais como Coq (Coq, 1996), Isabelle (Nipkow *et al.*, 2002) e Lean (Moura *et al.*, 2015). Os autores afirmam que os algoritmos de aprendizado de máquina podem automatizar essa interação, abrindo novas perspectivas para área de provadores automáticos de teorema (Yang; Deng, 2019): o modelo pode aprender a interagir com os assistentes de prova, a partir dos dados provenientes das provas escritas pelos especialistas humanos. Os autores evidenciam, porém, que a área de prova formal de teoremas impõe desafios para os algoritmos de aprendizado de máquina uma vez que há meios de se checar formalmente e automaticamente a correção das provas e, assim, não há espaço para que o modelo alucine. Assim, acoplar LLMs com assistentes de provas pode melhorar a capacidade dos LLMs de realizar provas com raciocínio de vários passos (Schick *et al.*, 2024). Todavia, os autores afirmam que a pesquisa de utilizar LLMs para prova de teoremas tem enfrentado desafios pelos seguintes fatos: (i) não há LLMs-baseados em provadores que sejam código aberto (Polu; Sutskever, 2020; Jiang *et al.*, 2021; Han *et al.*, 2022; Lample *et al.*, 2022; Jiang *et al.*, 2022; Polu *et al.*, 2022; First *et al.*, 2023; Wang *et al.*, 2023), não sendo possível reproduzir o treinamento e iterações com os assistentes de prova; (ii) eles usam dados pré-treinados privados e; (iii) necessitam de complexos recursos computacionais que podem chegar a milhares de dias de uso de GPUs.

[1] https://scholar.google.com.br/
[2] https://leandojo.org/

O LeanDojo extrai dados do assistente de provas Lean e permite a interação com o ambiente de provas. Ele contém anotações de premissas nas provas, fornecendo dados para a tarefa de seleção de premissas, que é um gargalo na área de provadores de teorema. Usando estes dados, os autores construíram um novo *benchmark* com 98.734 teoremas e o usaram para treinar e avaliar o modelo do ReProver. Segundo os autores, o treinamento levou apenas cinco dias em uma única GPU. O ReProver foi capaz de provar $51,2\%$ dos teoremas no *benchmark* do LeanDojo, **superando um baseline que utiliza o GPT-4** $(29,0\%)$. Além disso, o *ReProver* foi capaz de provar 65 teoremas que atualmente não possuem provas no *Lean*, demonstrando sua capacidade de contribuir para a expansão das bibliotecas matemáticas existentes.

As semelhanças do trabalho aqui proposto e do trabalho de (Yang; Deng, 2019) é que ambos estão avaliando o uso de LLMs na tarefa de *provas de teoremas*. No entanto, o trabalho de (Yang; Deng, 2019) utiliza provas matemáticas no geral enquanto esse trabalho foca na resolução de um tipo específico de prova que é a dedução natural. Os autores do trabalho (Yang; Deng, 2019) também constroem um novo provador de teoremas, treinado com os dados disponíveis no Lean, e, pelo mostrado nos experimentos, esta nova ferramenta melhorou o desempenho de generalização de provas de vários passos.

3.2 Evaluating Large Language Models with NeuBAROCO: Syllogistic Reasoning Ability and Human-like Biases

O trabalho de (Ando *et al.*, 2023) investiga se os LLMs atuais apresentam vieses de raciocínio lógico semelhantes a vieses apresentados por seres humanos. A avaliação é efetuada em raciocínio sobre silogismos, que é uma forma de inferência bem estudada em Lógica, consistindo em quatro tipos de sentenças categóricas como mostrado no Quadro 2.

Quadro 2 – Quatro Tipos de Sentenças Categóricas.

Tipo	Forma	Descrição
A	Todo S é P	Afirmação Universal
E	Nenhum S é P	Negação Universal
I	Algum S é P	Afirmação Particular
O	Algum S não é P	Negação Particular

Fonte: Tradução Livre de (Ando *et al.*, 2023).

Cada silogismo possui duas premissas (P1 e P2) e uma conclusão (C). Assim, podemos identificar um silogismo de acordo com o tipo das premissas que o formam. O Silogismo 3.1 apresentado a seguir é do tipo **EIO**.

$P1$:Nenhum B é C. (tipo **E**)

$P2$:Algum A é B. (tipo **I**) (3.1)

C :Algum A não é C. (tipo **O**)

Os vieses examinados foram de três tipos: *viés de crença*, *erros de conversão* e *efeitos de atmosfera*. Os *vieses de crença* ocorrem quando há dificuldade de determinar se uma inferência é válida quando a mesma possui uma *frase que é contrária ao senso comum*. No Silogismo 3.2, apresentado a seguir, podemos observar que *mesmo que o silogismo seja logicamente válido* o fato da conclusão ser contra o senso comum *pode levar algumas pessoas a julga-lá como contraditória*.

$P1$:Alguns animais são seres humanos.

$P2$: Todos os animais são tomates. (3.2)

C : Alguns humanos são tomates.

Os *erros de conversão* são erros causados pela interpretação incorreta de termos. Podemos observar no Silogismo 3.3 uma conclusão considerada neutra, que não podemos afirmar ou negar logicamente sua veracidade, entretanto as pessoas podem ter a tendência de julga-lá como válida ao considerar que a sentença "*Todos A são B*" é equivalente a "*Todos B são A*".

$P1$: Todos B são A.

$P2$: Todos B são C. (3.3)

C : Todos A são C.

Os *efeitos atmosféricos* podem ser interpretados a partir de dois princípios. O princípio da qualidade, no qual se uma ou ambas as premissas forem negativas (tipo E ou tipo O), a conclusão deve ser negativa; caso contrário, é positivo (tipo A ou tipo I). E, o princípio da quantidade: se uma ou ambas as premissas forem particulares (tipo I ou tipo O), então a conclusão será particular; caso contrário, é universal (tipo A ou tipo E). Podemos observar no Silogismo 3.4 uma conclusão considerada neutra, isto é, não se pode afirmar ou negar logicamente a sua veracidade.

$$\begin{array}{l} P1 : \text{Todos os animais são seres vivos.} \\ P2 : \text{Alguns humanos não são seres vivos.} \\ \hline C : \text{Nenhum dos animais é humano.} \end{array} \tag{3.4}$$

Para avaliação se os LLMs cometiam tais vieses descritos anteriormente, os autores criaram uma base de dados chamada *NeuBaroco*, construída sobre a base de dados *Baroco* (Shikishima *et al.*, 2009), porém com questões adicionadas e rótulos atribuídos manualmente. Para cada inferência foi atribuído um rótulo de implicação, contradição ou neutro, sendo 375 inferências no total: 122 rotulados como implicação; 71 rotulados como contradição e; 182 como neutras. Foram avaliados os modelos RoBERTa (Liu *et al.*, 2019) e BART (Lewis *et al.*, 2020) e GPT-3.5 (GPT-3.5-turbo[3]) e os resultados mostram que os modelos apresentam muitas falhas causadas por vieses semelhantes aos seres humanos.

As semelhanças entre o trabalho de (Ando *et al.*, 2023) e o nosso trabalho residem na avaliação de LLM do tipo GPT em tarefas relacionadas ao raciocínio lógico. Ambos os estudos empregam inferências ligadas à Lógica de Predicados. No entanto, enquanto os autores de (Ando *et al.*, 2023) utilizam textos em linguagem natural como entrada, esse artigo utiliza fórmulas nas linguagens da lógica proposicional e de predicados. O tipo de prova também difere do nosso trabalho: enquanto (Ando *et al.*, 2023) avalia a aplicação de regras de silogismos este trabalho avalia a aplicação de regras de dedução natural.

3.3 Testing the General Deductive Reasoning Capacity of Large Language Models Using OOD Examples

Em (Saparov *et al.*, 2023) foi avaliada a capacidade de raciocínio dedutivo geral de LLMs, medindo quão bem eles são capazes de aprender com provas simples e generalizar esse aprendizado para realizar provas mais complexas. O Quadro 3 mostra os tipos de generalização pretendidas pelos autores.

De modo geral, cada exemplo de treinamento é um exemplo de demonstração de sequência de raciocínio (do inglês, *CoT - Chain of Thoughts*), uma sequência lógica de ideias ou conceitos interconectados que formam um raciocínio ou argumento coerente, e cada exemplo de teste é um exemplo de prova que o modelo deve produzir. A complexidade das provas foram

[3] https://platform.openai.com/docs/model-index-for-researchers

Quadro 3 – Tipos de generalização avaliadas por (Saparov *et al.*, 2023).

Exemplo de treino	Tipo de teste	Exemplo de teste
"Alex é um cachorro. Todos os cães são mamíferos. Alex é um mamífero."	Teste sobre regras de dedução não vistas em treinamento	"Alex não é um mamífero. Todos os cães são mamíferos. Suponha que Alex seja um cachorro. Alex é um mamífero. Isso contradiz que Alex não é um mamífero. Alex não é um cachorro."
"Alex é um cachorro. Todos os cães são mamíferos. Alex é um mamífero."	Teste em provas mais profundas	"Alex é um cachorro. Todos os cães são mamíferos. Alex é um mamífero. Todos os mamíferos são vertebrados. Alex é um vertebrado."
"Alex é um cachorro. Alex é macio. Alex é um cachorro e macio".	Teste em provas mais amplas	"Alex é um cachorro. Alex é macio. Alex é gentil. Alex é um cachorro, macio e gentil."
"Alex é um cachorro. Todos os cães são mamíferos. Alex é um mamífero. Fae é um gato. Fae é agradável. Fae é agradável e um gato".	Teste em provas de composição	"Alex é um cachorro. Todos os cães são mamíferos. Alex é um mamífero. Alex não é mau. Alex é um mamífero e não é mau."

Fonte: Tradução Livre de (Saparov *et al.*, 2023).

descritas em três grupos: (i) quantidade de regras de dedução envolvidas; (ii) profundidade da prova, i.e., o comprimento de uma cadeia sequencial de passos de uma prova e; (iii) largura da prova, i.e., o número de premissas de cada passo de prova.

Para medir a capacidade geral de raciocínio dedutivo dos LLMs, os autores: determinaram se os modelos aprenderam um conjunto completo de **regras de dedução natural**; avaliaram se os modelos conseguem raciocinar sobre provas mais longas do que as fornecidas como exemplos (generalização em profundidade e largura) e; avaliaram se os modelos são capazes de utilizar múltiplas regras de dedução diferentes em uma única prova (generalização composicional).

O conjunto de dados utilizado foi o PRONTOQA-OOD [4] no qual cada exemplo contém um conjunto de premissas, uma consulta (o fato alvo a ser provado ou reprovado) e uma cadeia de pensamento contendo a prova da consulta. Para avaliar o raciocínio usando diferentes regras de dedução, foram geradas provas com regras de dedução para todos os conectivos na lógica proposicional (conjunção, disjunção, implicação e negação). Para evitar que os LLMs explorem o conhecimento do pré-treinamento para resolver os problemas sem raciocínio, os autores utilizaram nomes fictícios para todos os conceitos (por exemplo, "*wumpus*" no lugar

[4] https://github.com/asaparov/prontoqa

de “gato”). O conjunto de dados utilizado exige que os LLMs gerem provas completas, no lugar de respostas como verdadeiro e falso. Os experimentos foram realizados utilizando quatro LLMs, *FLAN-T5* (Chung *et al.*, 2024), *LLaMA* (Touvron *et al.*, 2023), *GPT-3.5* [5] e *PaLM* (Chowdhery *et al.*, 2023), de tamanhos e objetivos de treinamento variados. Para os experimentos, foram utilizadas oito demonstrações, sendo comparados os desempenhos em dois ambientes. No primeiro ambiente, os exemplos de contexto e os exemplos de teste são semelhantes aos utilizados nos treinamentos (chamados de exemplos de mesma distribuição). No segundo ambiente, o exemplo de teste possui regra de dedução específica, mas os exemplos no contexto são distribuídos uniformemente por todas as outras regras de dedução - exemplos chamados de fora de distribuição (do inglês, OOD - *Out of Distribution*). A Figura 21 apresenta um exemplo de questão do conjunto de dados contendo as premissas, o comando da prova (que deve ser provado ou rejeitado), a resposta fornecida pelo modelo (resposta prevista) e, logo após, a resposta esperada.

Figura 21 – Exemplo de uma prova incorreta gerada pelo GPT-3.5. As premissas utilizadas na prova estão escritas em azul e os passos inválidos em vermelho.

Premissas: Lempuses são gorpuses. Rompuses são wumpuses. Tumpuses são wumpuses. Dumpuses são gorpuses. Todo zumpus é um wumpus. Todo brimpus é um gorpus. Yumpuses são gorpuses. Lempuses são wumpuses. Rompuses são gorpuses. Tumpuses são gorpuses. Max é um tumpus ou um rompus ou um lempus. Max é um zumpus, um tumpus, ou um rompus.
Prove: Max é um gorpus.
Resposta prevista: Max é um tumpus ou um rompus ou um lempus. Max é um tumpus. Tumpuses são wumpuses. Max é um wumpus. Rompuses são gorpuses. Max é um gorpus. Max é um gorpus.
Resposta esperada: Suponha que Max é um tumpus. Tumpuses são gorpuses. Max é um gorpus. Suponha que Max é um rompus. Rompuses são gorpuses. Max é um gorpus. Suponha que Max é um lempus. Lempuses são gorpuses. Max é um gorpus. Visto que Max é um tumpus ou um rompus ou um lempus, Max é um gorpus.

Fonte: Tradução Livre de (Saparov *et al.*, 2023).

De modo geral, não foi observada uma forte correlação entre tamanho do modelo e desempenho. Além disso, a maioria dos modelos provaram que eram capazes de usar cada regra de dedução razoavelmente bem, com o GPT-3.5 apresentando o melhor desempenho. Os modelos foram capazes de usar quatro regras de dedução natural, apesar de não ser mostrado um exemplo no contexto com essas regras: introdução de disjunção, eliminação de implicações, conjunção introdução e conjunção eliminação. Por fim, os modelos foram capazes de gerar provas composicionais até certo ponto e tiveram dificuldade de gerar provas mais longas, além

[5] https://platform.openai.com/docs/model-index-for-researchers

de requerem demonstração explícita para produzir sub provas hipotéticas, principalmente em provas por casos e contradição.

Assim como o nosso trabalho, os autores em (Saparov *et al.*, 2023) buscam avaliar o desempenho de LLMs, incluindo o GPT 3.5 e LLaMA, **na tarefa de realizar provas complexas de dedução natural**. Entretanto no conjunto de dados utilizado em (Saparov *et al.*, 2023), as questões são descritas em linguagem natural (que poderia ser traduzida para a linguagem da lógica de predicados) e o conjunto de dados que usamos neste trabalho utiliza as questões descritas como fórmulas nas linguagens das lógicas proposicional e de predicados. Os autores em (Saparov *et al.*, 2023) realizam um treinamento para que os LLMs aprendam sobre as regras de dedução natural para só depois avaliar se eles são capazes de aplicá-las em diferentes contextos de provas. Neste trabalho, os *LLM*s serão avaliados antes e depois de serem treinados.

3.4 Comparativo das abordagens

O Quadro 4 fornece uma comparação entre nosso trabalho e os trabalhos apresentados nesse capítulo. Destacamos para cada trabalho: os LLMs utilizados; a linguagem na qual os dados são apresentados aos modelos e tipo de provas realizadas. O trabalho de (Saparov *et al.*, 2023) é o mais relacionado a esta pesquisa. No entanto, além de avaliar LLMs diferentes, utiliza um *dataset* em linguagem natural, enquanto esse trabalho utiliza como *dataset* exercícios já formalizados nas linguagens da Lógica Proposicional e de Predicados.

Quadro 4 – Comparação entre os trabalhos relacionados e esta proposta.

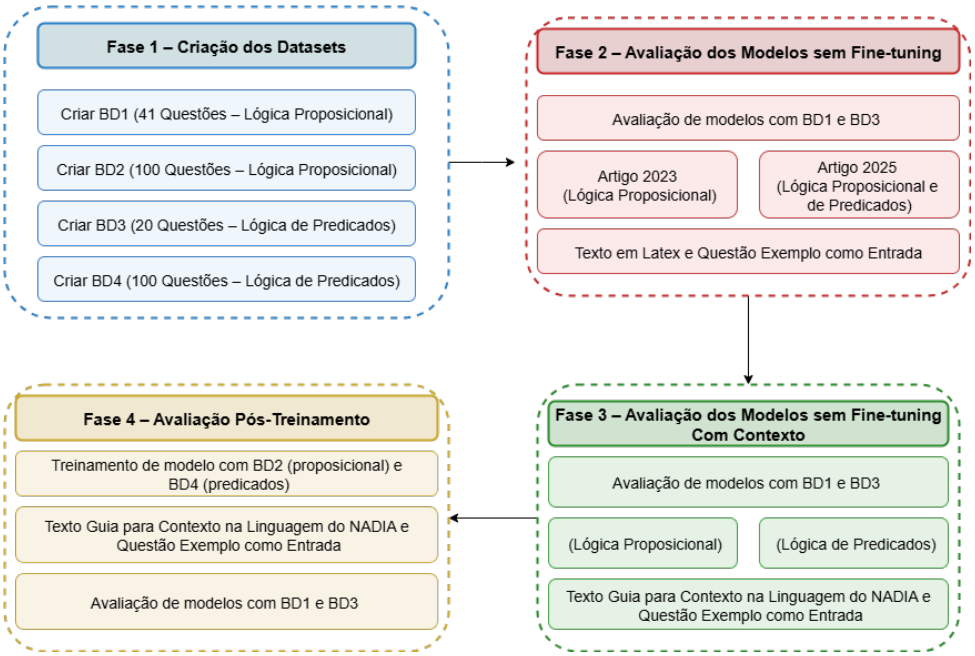
Trabalho	**LLMs utilizadas**	**Tipo de Linguagem**	**Tipo de Prova**
(Yang *et al.*, 2024)	GPT-4	Linguagem Formal (LEAN)	Provas matemáticas no geral
(Ando *et al.*, 2023)	RoBERTa, BART e GPT-3.5	Linguagem Natural que pode ser formalizada em Lógica de Predicados	Silogismos em Lógica de Predicados
(Saparov *et al.*, 2023)	FLAN-T5, LLaMA, GPT-3.5 e PaLM	Linguagem Natural que pode ser formalizada em Lógica de Predicados	Dedução Natural em Lógica de Predicados
Este trabalho	**GPT-4.1-mini, GPT 4o e GPT 3.5-turbo**	**Lógica Proposicional e Predicados**	**Dedução Natural em Lógica Proposicional e de Predicados**

Fonte: Próprio autor (2025).

4 METODOLOGIA

Com objetivo de avaliar a habilidade de LLMs na resolução de exercícios de Dedução Natural em Lógica Proposicional e Lógica de Predicados, foram realizadas uma série de atividades, divididas em quatro fases, as quais serão descritas a seguir e são ilustradas na Figura 22.

Figura 22 – Atividades Metodológicas do Trabalho.



Fonte: Próprio autor (2025).

4.1 Fase 1: Construção das bases de dados

Foram construídas 4 bases de dados de exercícios de Dedução Natural em Lógica Proposicional e Lógica de Predicados com suas respectivas respostas corretas, as quais denominamos: BD1, BD2, BD3 e BD4. Os exercícios são textos escritos no formato aceito pela ferramenta *NADIA* (e.g, `A -> B, ~B |- ~A`) correspondentes a teoremas com premissas e conclusão, a saber: $A \to B, \neg B \vdash \neg A$ (com premissas $A \to B$ e $\neg B$ e conclusão $\neg A$).

As respostas são provas de dedução natural no estilo *Fitch* no mesmo formato. Para a validação das respostas utilizamos a ferramenta NADIA (*Natural Deduction Proof Assistant*), que é um assistente de provas para o sistema de Dedução Natural em Lógica Proposicional e Lógica de Predicados, no estilo de Fitch (Vasconcelos; Menezes, 2024). As bases de dados foram divididas da seguinte maneira:

- **BD1:** 41 exercícios de Dedução Natural em Lógica Proposicional (enunciados e suas respectivas respostas corretas)
- **BD2:** 100 exercícios de Dedução Natural em Lógica Proposicional (enunciados e suas respectivas respostas corretas).
- **BD3:** 20 exercícios de Dedução Natural em Lógica de Predicados (enunciados e suas respectivas respostas corretas).
- **BD4:** 100 exercícios de Dedução Natural em Lógica de Predicados (enunciados e suas respectivas respostas corretas).

As bases de dados **BD1** e **BD3** tiveram suas questões coletadas das listas de exercícios na disciplina de Lógica para Computação para alunos de graduação em cursos da área de Tecnologia da Informação da Universidade Federal do Ceará - Campus Quixadá[1] e foram utilizadas nesse trabalho para testes, assim como em (Martins *et al.*, 2023; Martins *et al.*, 2025). As bases de dados **BD2** e **BD4** são *datasets* criados para serem utilizados para o treino, suas questões foram em partes coletadas nas listas de exercícios na disciplina de Lógica para Computação para alunos de graduação em cursos da área de Tecnologia da Informação da Universidade Federal do Ceará - Campus Quixadá (assim como as questões das bases **BD1** e **BD3**), na literatura (Huth; Ryan, 2004) bem como em repositórios online, ou geradas utilizando a ferramenta *NotebookLM* [2]. Essa ferramenta possui uma *LLM* na qual se pode utilizar fontes como contexto, a ferramenta foi utilizada para gerar questões novas utilizando como fontes outras questões já coletadas e validadas. As bases de dados estão disponíveis em diversos formatos de forma gratuita em um repositório online[3].

4.2 Fase 2: Avaliação dos Modelos sem Treinamento

Um artigo com experimentos referentes à Lógica Proposicional foi publicado na conferência nacional Simpósio Brasileiro de Informática na Educação (SBIE 2023 - Qualis A3): (Martins *et al.*, 2023). Este artigo recebeu o prêmio de melhor artigo da conferência. Assim, fomos convidados a submeter uma versão estendida à RBIE (Revista Brasileira de Informática na Educação - Qualis *B1*), com conteúdo adicional sobre Lógica de Predicados (Martins *et al.*, 2025). A seguir, serão apresentadas as metodologias para os experimentos do artigo (Martins *et al.*, 2023) na Seção 4.2.1, e as metodologias para os experimentos do artigo (Martins *et al.*,

[1] https://github.com/daviromero/logic4py/blob/main/src/book/Index.ipynb
[2] https://notebooklm.google.com/
[3] https://github.com/leonardomartins777/Deducao-Natural-com-LLMs

2025) nas Seções 4.2.2 (Lógica Proposicional) e 4.2.3 (Lógica de Predicados).

4.2.1 Metodologia utilizada em (Martins et al., 2023) - Contexto fornecido para resolução de exercícios em Lógica Proposicional

Inicialmente, um exercício exemplo foi submetido ao modelo (GPT 3.5-turbo), cujo enunciado do exercício é um texto com um teorema em LaTeX (conforme mostrado na Figura 23) e a saída é um código LaTeX (conforme mostrado na Figura 24), no formato adotado pelo livro texto "*Lógica em Ciência da Computação: Modelagem e Raciocínio sobre sistemas*" (Huth; Ryan, 2004), o qual utiliza o pacote `logicproof`[4] de escrita de provas de dedução natural no estilo *Fitch*. Em seguida, os exercícios do *BD1* são submetidos ao modelo para que ele construa a sua solução, como o exemplo apresentado na Figura 25.

Figura 23 – Pergunta exemplo fornecida ao modelo.

"escreva a prova `$A \rightarrow B, B \rightarrow C \vdash A \rightarrow C$` usando o sistema de dedução natural no estilo fitch e o pacote latex logicproof."

Fonte: Próprio autor (2025).

Figura 24 – Resposta exemplo fornecida ao modelo.

```
\begin{logicproof}{6}
    A \rightarrow B & premissa \\
    B \rightarrow C & premissa \\
    \begin{subproof}
        A & hipótese \\
        B & $\rightarrow e$ 3,1 \\
        C & $\rightarrow e$ 4,2
    \end{subproof}
    A \rightarrow C & $\rightarrow i$ 3-5
\end{logicproof}
```

Fonte: Próprio autor (2025).

Figura 25 – Pergunta fornecida para que o modelo construa sua demonstração.

"escreva a prova `\vdash A \rightarrow (B \rightarrow A)` usando o sistema de dedução natural no estilo fitch e o pacote latex logicproof"

Fonte: Próprio autor (2025).

Avaliamos se nas respostas fornecidas *as regras de Dedução Natural eram aplicadas corretamente* e se as provas foram escritas de acordo com a sintaxe do pacote `logicproof`. Os

[4] https://ctan.org/pkg/logicproof

erros apresentados eram classificados em dois principais erros: *erros lógicos*, quando ocorrem erros na aplicação das regras de Dedução Natural e; *erros de referências nas regras*, quando as respostas, apesar das regras serem utilizadas adequadamente, apresentavam erros nas referências durante sua aplicação.

4.2.2 Metodologia utilizada em (Martins et al., 2025) - Contexto fornecido para resolução de exercícios em Lógica Proposicional

Um exercício de lógica proposicional foi fornecido como contexto ao modelo *GPT 4o* cujo enunciado é um texto com um teorema em LaTeX (conforme mostrado na Figura 26) e a saída é um código LaTeX (conforme mostrado na Figura 27), no formato adotado pelo livro texto "*Lógica em Ciência da Computação: Modelagem e Raciocínio sobre sistemas*" ((Huth; Ryan, 2004)), o qual utiliza o pacote `logicproof`[5] de escrita de provas de dedução natural no estilo *Fitch*.

Figura 26 – Pergunta exemplo fornecida ao modelo.

```
"'escreva a prova do teorema a seguir, usando o sistema de dedução natural no estilo fitch e o
pacote latex logicproof.
Teorema:"
$A \rightarrow B, B \rightarrow C \vdash A \rightarrow C$
"
"'
```

Fonte: Próprio autor (2025).

Figura 27 – Resposta exemplo fornecida ao modelo.

```
"'
Prova:"
\begin{logicproof}{6}
     A \rightarrow B & premissa \\
     B \rightarrow C & premissa \\
     \begin{subproof}
          A & hipótese \\
          B & $\rightarrow e$ 3,1 \\
          C & $\rightarrow e$ 4,2
     \end{subproof}
     A \rightarrow C & $\rightarrow i$ 3-5
\end{logicproof}
"
"'
```

Fonte: Próprio autor (2025).

[5] https://ctan.org/pkg/logicproof

4.2.3 Metodologia utilizada em (Martins et al., 2025) - Contexto fornecido para resolução de exercícios em Lógica de Predicados

Assim como em logica proposicional, um exercício exemplo de lógica de predicados foi submetido à ferramenta com o enunciado sendo um texto com o seguinte teorema em LaTeX: $\forall x(H(x) \rightarrow M(x)), \forall x \neg M(x) \vdash \neg \exists x H(x)$ (conforme mostrado na Figura 28) e a saída um código LaTeX (conforme mostrado na Figura 29).

Figura 28 – Pergunta exemplo fornecida ao modelo.

```
“‘escreva a prova do teorema a seguir, usando o sistema de dedução natural no estilo
fitch e o pacote latex logicproof.
Teorema:“
$ \forall x (H(x) \rightarrow M(x)),\forall x \lnot M(x) \vdash
\lnot \exists x H(x)$
”
”’
```

Fonte: Próprio autor (2025).

Figura 29 – Resposta exemplo fornecida ao modelo.

```
“‘
Prova:“
\begin{logicproof}{6}
     \forall x (H(x)\rightarrow M(x)) & premissa \\
     \forall x \lnot M(x) & premissa \\
     \begin{subproof}
          \exists x H(x) & hipótese \\
          \begin{subproof}
               \llap{a\quad}H(a) & hipótese \\
               H(a)\rightarrow M(a) & \forall e 1 \\
                M(a) & \rightarrow e 4, 5 \\
                \lnot M(a) & \forall e 2 \\
                \bot & \lnot e 6, 7
          \end{subproof}
          \bot & ∃e 3,4-8
     \end{subproof}
     \lnot \exists x H(x) & \lnot i 3-9
\end{logicproof}
”
”’
```

Fonte: Próprio autor (2025).

4.2.4 Resolução de exercícios das bases de dados pelo LLM

Após fornecido o exercício exemplo de contexto ao LLM, procedemos com o fornecimento de perguntas disponíveis nas bases de dados *BD1* e *BD3*. O LLM recebe uma pergunta da base de dados e fornece uma resposta como saída. Cada resposta é posteriormente avaliada. As Figuras 30 e 31 ilustram, respectivamente, exemplos de perguntas da base de dados de lógica proposicional e lógica de predicados que foram submetidas ao LLM.

Figura 30 – Pergunta fornecida para que o modelo construa sua demonstração.

```
“‘escreva a prova do teorema a seguir, usando o sistema de dedução natural no estilo
fitch e o pacote latex logicproof.
Teorema:“
\vdash A \rightarrow (B \rightarrow A)
”
”’
```

Fonte: Próprio autor (2025).

Figura 31 – Pergunta fornecida para que o modelo construa sua demonstração.

```
“‘escreva a prova do teorema a seguir, usando o sistema de dedução natural no estilo
fitch e o pacote latex logicproof.
Teorema:“
\forall x P(x) \vdash \lnot \exists x \lnot P(x)
”
”’
```

Fonte: Próprio autor (2025).

Para cada resposta emitida pelo LLM, avaliamos se as regras de Dedução Natural eram aplicadas corretamente e se as provas foram escritas de acordo com a sintaxe do pacote `logicproof`, verificando assim os seguintes tipos de erros:

- *Erros lógicos* que são erros na aplicação das regras de Dedução Natural;
- *Erros de referências* nas regras que são erros relacionados a apresentação de referências incorretas durante a aplicação das regras ou erros de escrita das regras em geral.

Algumas questões possuíam erros que não podiam ser classificados como erros de lógica ou de formatação. Esses eram erros sintáticos de uso do LaTeX tais como: Quebras abruptas de linha em alguns casos, ausências de quebra de linha em outros e adição de caracteres inválidos etc. Como esses erros eram erros facilmente corrigíveis e não comprometiam a avaliação dos exercícios, optamos por fazer a correção manual desses erros.

Na Figura 32(a), podemos observar uma resposta para o enunciado $A \vee B, \neg B \vdash A$

onde existem duas linhas, 5 e 8, apenas a numeração da linha e com os demais espaços em branco, isso ocorre por uma de quebra de linha extra após o fechamento da caixa. Na Figura 32(b), é apresentado uma resposta após aplicar as correções de formatação: as quebras de linhas foram removidas. É importante ressaltar que a resposta continua incorreta, porém o erro de formatação foi corrigido.

Figura 32 – Exemplos de demonstrações para $A \vee B, \neg B \vdash A$, resultado da compilação do código LaTeX escrito pelo modelo.

1.	$A \vee B$	premissa
2.	$\neg B$	premissa
3.	A	hipótese
4.	A	Reiteração 3
5.		
6.	B	hipótese
7.	$\bot$	$\neg e$ 2, 5
8.		
9.	A	$\vee e$ 1, 3-4, 5-6

(a) Antes da Correção na Formatação

1.	$A \vee B$	premissa
2.	$\neg B$	premissa
3.	A	hipótese
4.	A	Reiteração 3
5.	B	hipótese
6.	$\bot$	$\neg e$ 2, 5
7.	A	$\vee e$ 1, 3-4, 5-6

(b) Depois da Correção na Formatação

Fonte: Próprio autor (2025).

Cada um dos resultados foi avaliado manualmente por dois autores e revisados pelos outros dois. Optamos por uma avaliação manual onde todo o processo da prova é avaliado, pois as questões não são objetivas o que dificulta o processo de avaliação automatizado. As questões, os gabaritos e as respostas dos modelos estão disponibilizadas como corpus para teste[6].

4.3 Fase 3: Avaliação do Modelo sem Treinamento e Com Contexto

Para os experimentos dessa seção foram utilizadas as bases de dados **BD1** (lógica proposicional) e **BD3** (lógica de predicados), descritas na Seção 4.1. Os enunciados dessas questões foram submetidos ao Modelo GPT 4.1-mini, que foi escolhido para o experimento pelo seu desempenho em várias tarefas [7] bem como seu preço acessível[8] e por seus modelos antecessores terem sido utilizados nos trabalhos relacionados (Yang *et al.*, 2024; Ando *et al.*, 2023; Saparov *et al.*, 2023) bem como nossos artigos já publicados (Martins *et al.*, 2023; Martins *et al.*, 2025) que apresentamos na Seção 4.2. A seguir, serão apresentadas a metodologias para

[6] https://github.com/leonardomartins777/ChatGPT_Naturalded
[7] https://openai.com/index/gpt-4-1/
[8] https://platform.openai.com/docs/pricing

os experimentos nas Seções 4.3.1 e 4.3.2.

4.3.1 Contexto fornecido para resolução de exercícios em Lógica Proposicional e de Predicados

Foi criado um texto padrão onde se explica de maneira didática como é feito o processo de conversão de um texto em LaTeX para a linguagem aceita pela ferramenta NADIA, bem como, a explicação de como se constrói provas de dedução natural nessa linguagem. O texto é apresentado no Capítulo 7 e foi adaptado do material preparado para a disciplina de Lógica para Computação da Universidade Federal do Ceará - Campus de Quixadá[9], o texto está escrito em markdown [10]. Esse texto que no decorrer desse trabalho chamaremos de *Texto Padrão*, foi enviado como contexto em cada interação com os modelos. Além do texto padrão um exercício exemplo de lógica de predicados foi submetido à ferramenta com o enunciado sendo um texto com o seguinte teorema na linguagem aceita pelo NADIA: `Ax (H(x)->M(x)), Ax ~M(x) |- ~Ex H(x)` (conforme mostrado na Figura 33) junto com uma mensagem contendo a resposta da questão de exemplo, simulando uma resposta do sistema, como podemos observar (na Figura 34).

Figura 33 – Pergunta exemplo fornecida ao modelo.

```
‘‘‘
Questão:
escreva a prova do teorema a seguir, usando o sistema de dedução natural no estilo fitch e de
acordo com o formato aceito pelo NADIA.
Teorema:“
Ax (H(x)->M(x)), Ax ˜ M(x) |- ˜ Ex H(x)
”
’’’
```

Fonte: Próprio autor (2025).

4.3.2 Resolução de exercícios das bases de dados pelo LLM

Após fornecido o exercício exemplo com contexto ao LLM junto com a resposta, procedemos com o fornecimento do texto padrão junto com as perguntas disponíveis nas bases de dados. O LLM recebe o texto padrão junto com uma pergunta da base de dados e fornece uma

[9] https://github.com/daviromero/logic4py/blob/main/src/book/Index.ipynb
[10] https://daringfireball.net/projects/markdown/

Figura 34 – Resposta exemplo fornecida ao modelo.

```
‘‘‘
Prova:“
1. Ax(H(x) -> M(x)) pre
2. Ax ˜ M(x) pre
3. { Ex H(x) hip
4. { a H(a) hip
5. H(a) -> M(a) Ae 1
6. M(a) ->e 4, 5
7. ˜ M(a) Ae 2
8. @ ˜ e 6, 7
}
9. @ Ee 3, 4-8
}
10. ˜ Ex H(x) ˜ i 3-9
”
’’’
```

Fonte: Próprio autor (2025).

resposta como saída. Cada resposta é posteriormente avaliada. A Figura 35 ilustra um exemplo de pergunta da base de dados que foi submetida ao LLM (junto com o texto padrão).

Figura 35 – Pergunta exemplo fornecida ao modelo.

```
‘‘‘
Questão:
escreva a prova do teorema a seguir, usando o sistema de dedução natural no estilo
fitch e de acordo com o formato aceito pelo NADIA.
Teorema:“
|- A -> (B -> A)
”
’’’
```

Fonte: Próprio autor (2025).

4.4 Fase 4: Avaliação do Modelo Treinado

Após a resolução dos exercícios pelos LLMs sem treinamento, os modelos foram treinados utilizando as bases de dados *BD2* e *BD4*. Para o treinamento do modelo GPT 4.1-mini foi utilizada a ferramenta de *fine-tuning* disponibilizada pela empresa OpenAI[11]. O tipo de fine-tuning utilizado é o supervisionado, onde exemplos de entradas são apresentadas juntamente com saídas esperadas, o que permite um treino voltado para casos de usos específicos. Os

[11] https://platform.openai.com/finetune/

dados foram adaptados ao formato JSONL[12] para serem utilizados de acordo com a formatação exigida. A partir de então as bases de dados *BD2* e *BD4* foram submetidas à ferramenta padrão da OpenAI, com todos os parâmetros definidos com os valores padrão.

Depois do treinamento, o modelo foram submetidos aos mesmos experimentos descritos na Seção 4.3: Os enunciados dos exercícios das bases de dados BD1 e BD3 foram submetidos aos modelos. As respostas fornecidas foram armazenadas no seguinte repositório: https://github.com/leonardomartins777/ChatGPT_Naturalded.

[12] https://platform.openai.com/docs/guides/supervised-fine-tuning

5 RESULTADOS

A seguir, serão apresentados os resultados de cada fase apresentada no Capítulo 4. A Seção 5.1 apresenta os *datasets* criados. A Seção 5.2 apresenta os resultados dos experimentos do artigo (Martins *et al.*, 2023) e dos experimentos do artigo (Martins *et al.*, 2025). Na Seção 5.3 são apresentados os resultados dos experimentos utilizando a linguagem aceita pelo *NADIA* e o texto padrão, porém sem treinamento. Na Seção 5.4 é apresentado o resultado dos experimentos utilizando o modelo treinado a linguagem aceita pela ferramenta *NADIA* e o texto padrão, e por fim, na Seção 5.5 é apresentado um resumo dos resultados do trabalho.

5.1 Resultados para a Fase 1: Construção das bases de dados

Nesta fase, construímos as bases de dados descritas na Seção 4.1. Estas questões estão disponíveis em um repositório online[1]. No caso dos *datasets* utilizados para o treino (BD2 e BD4) as questões estão estão escritas na linguagem aceita pela ferramenta NADIA, assim como as respostas. No caso dos *datasets* utilizados para teste, os enunciados são escritos na linguagem aceita pela ferramenta *NADIA* bem como em LaTeX com o pacote `logicproof`. A Figura 36 mostra o enunciado de uma questão de lógica proposicional (base de dados BD1) escrito em LaTeX e na linguagem aceita pela ferramenta *NADIA* respectivamente. As respostas escritas na linguagem aceita pela ferramenta *NADIA* podem ser convertidas para o LaTeX utilizando a implementação da ferramenta *NADIA* disponível de forma online[2].

Figura 36 – Exemplo de Enunciados de Questões Escritos em Formatos Diferentes.

$\vdash (\neg A \rightarrow B) \rightarrow ((\neg A \rightarrow \neg B) \rightarrow A)$

(a) LaTeX *LogicProof*.

```
|- (~A->B)->((~A->~B)->A)
```

(b) Linguagem aceita pela ferramenta *NADIA*.

Fonte: Próprio autor (2025).

5.2 Resultados para a Fase 2: Avaliação dos Modelos sem Treinamento

5.2.1 Resultados de (Martins et al., 2023) - Resolução de exercícios em Lógica Proposicional

Nesta seção iremos apresentar os resultados dos experimentos do artigo (Martins *et al.*, 2023). Avaliamos se nas respostas fornecidas *as regras de Dedução Natural eram aplicadas*

[1] Exercícios disponíveis em https://github.com/leonardomartins777/Deducao-Natural-com-LLMs
[2] https://sistemas.quixada.ufc.br/nadia/index.jsp

corretamente e se as provas foram escritas de acordo com a sintaxe do pacote `logicproof`.

A Figura 37 mostra um exemplo de demonstração correta para $\vdash (A \to (B \to C)) \to ((A \to B) \to (A \to C))$. Nesta resposta, o modelo escreveu um código LaTeX, o qual foi compilado e corresponde a demonstração mostrada na figura. Veja em mais detalhes que o modelo: fez a suposição correta das hipóteses nas linhas 1, 2 e 3; aplicou corretamente a regra da implicação eliminação ($\to e$) e referenciou corretamente as linhas com a fórmula contendo o condicional e a veracidade do antecedente nas linhas 4, 5 e 6; aplicou corretamente a regra da implicação introdução ($\to i$) e referenciou corretamente os intervalos das caixas nas linhas 7, 8 e 9.

Figura 37 – Prova correta para $\vdash (A \to (B \to C)) \to ((A \to B) \to (A \to C))$, resultado da compilação do código LaTeX escrito pelo modelo.

1.	$A \to (B \to C)$	hipótese
2.	$A \to B$	hipótese
3.	A	hipótese
4.	B	$\to e$ 3, 2
5.	$B \to C$	$\to e$ 1, 3
6.	C	$\to e$ 4, 5
7.	$A \to C$	$\to i$ 3-6
8.	$(A \to B) \to (A \to C)$	$\to i$ 2-7
9.	$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$	$\to i$ 1-8

Fonte: Próprio autor (2025).

No entanto, a maior parte das respostas obtidas continham algum tipo de erro, os quais foram classificados em: *erros lógicos*, quando ocorrem erros na aplicação das regras de Dedução Natural e; *erros de referências nas regras*, quando as respostas, apesar das regras serem utilizadas adequadamente, apresentavam erros nas referências durante sua aplicação.

Nos *erros lógicos* avaliamos todos os erros decorrentes do mal uso das regras de Dedução Natural e incoerências lógicas em geral, que são os erros mais críticos para o objetivo do experimento. A partir dos resultados podemos observar que a maioria das respostas apresentam erros lógicos, um total de 31 (75,61%). Os erros de escrita de prova foram bem frequentes nestes experimentos. Destacamos aqui as questões que tiveram exclusivamente erros de referência de linhas, no qual são feitas referências incorretas às linhas durante a aplicação de determinadas regras. Esses erros totalizaram apenas 04 questões (9,76%). Dessa forma, podemos observar uma

grande imprecisão do modelo na tarefa de escrever as demonstrações, totalizando 35 questões erradas (85,37%). Em algumas respostas produzidas pelo modelo, há vários erros dessa categoria. A Figura 38 ilustra uma resposta incorreta produzida pelo modelo na qual há erros de escrita de prova (referência de linhas) e também erros lógicos. Nesta demonstração, o modelo: referenciou incorretamente nas linhas 3 e 4, as linhas $1,1$, uma vez que a aplicação da regra $\vee e$ deveria referenciar apenas a linha 2; referenciou incorretamente na linha 6, a linha 2 na aplicação da regra $\wedge e$, uma vez que deveria ter mencionado a linha 5 (o mesmo ocorre na linha 8); referenciou incorretamente na linha 7, as linhas $3,2$, uma vez que a aplicação da regra $\vee$ deveria referenciar apenas a linha 6; referenciou incorretamente na linha 9, as linhas $3,4$, uma vez que a aplicação da regra $\vee$ deveria referenciar apenas a linha 8 e; **cometeu um erro lógico** na linha 10 no uso da regra $\wedge$ i, a qual não poderia ser aplicada nesse contexto. Conforme a prova foi escrita, o correto seria utilizar a regra $\vee$ e para obter $A \vee C$. No entanto, não seria possível obter $(A \vee B)$.

Figura 38 – Demonstração incorreta para $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$, resultado da compilação do código LaTeX escrito pelo modelo.

1.	$A \vee (B \wedge C)$	premissa
2.	A	hipótese
3.	$A \vee B$	$\vee i$ 1, 1
4.	$A \vee C$	$\vee i$ 1, 1
5.	$B \wedge C$	hipótese
6.	B	$\wedge e_1$ 2
7.	$A \vee B$	$\vee i$ 3, 2
8.	C	$\wedge e_2$ 2
9.	$A \vee C$	$\vee i$ 3, 4
10.	$(A \vee B) \wedge (A \vee C)$	$\wedge i$ 2-6

Fonte: Próprio autor (2025).

A Tabela 1 resume o desempenho do modelo *GPT 3.5-turbo* na tarefa de Dedução Natural, onde ele teve apenas 06 acertos completos num total de 41 exercícios (14,63%). Além disso, podemos observar os tipos de erros mais comuns apresentados pelo modelo, sendo que para cada resposta mais de um erro pode ter ocorrido.

Tabela 1 – Desempenho apresentado pelo modelo na tarefa de dedução natural

Modelo	Acertos	Erros Apenas de Referência	Erros Lógicos
GPT 3.5-turbo	6	4	31
Total (%)	14,63%	9,76%	75,61%

Fonte: Próprio autor (2025).

5.2.2 Resultados de (Martins et al., 2025) - Resolução de exercícios em Lógica Proposicional e de Predicados

5.2.2.1 Resultados para os Exercícios de Lógica Proposicional

A Figura 39 mostra um exemplo de demonstração realizada corretamente pelo ChatGPT para o enunciado $\vdash (A \to (A \to B)) \to (A \to B)$. Nesta resposta, o modelo escreveu um código LaTeX, o qual foi compilado e corresponde a demonstração mostrada na figura. Veja em mais detalhes que o modelo: fez a suposição correta das hipóteses nas linhas 1 e 2; aplicou corretamente a regra da implicação eliminação ($\to e$) nas linhas 3 e 4 com as referencias corretas; e por fim, aplicou corretamente a regra da implicação introdução ($\to i$) e referenciou corretamente os intervalos das caixas nas linhas 5 e 6.

Figura 39 – Prova correta para $\vdash (A \to (A \to B)) \to (A \to B)$, resultado da compilação do código LaTeX escrito pelo modelo.

1.	$(A \to (A \to B))$	hipótese
2.	A	hipótese
3.	$A \to B$	$\to e$ 1, 2
4.	B	$\to e$ 3, 2
5.	$A \to B$	$\to i$ 2-4
6.	$((A \to (A \to B)) \to (A \to B))$	$\to i$ 1-5

Fonte: Próprio autor (2025).

A Figura 40(a) ilustra uma resposta incorreta produzida pelo modelo na qual há erros de referência e também erros lógicos. Nesta demonstração, o modelo:

- referenciou incorretamente na linha 8, as linhas $3,6$, uma vez que a linha 6, que foi referenciada, não possui formula correspondente;
- referenciou incorretamente na linha 10, o intervalo $6-7$ na aplicação da regra $\vee e$, uma vez que deveria ter mencionado o intervalo $7-8$;
- as linhas 6 e 9 não possuem formula (afirmação). Além disso, na linha 9 existe a referência

a um intervalo onde não existe caixa (*6-7*).

A Figura 40(b) apresenta uma prova correta na qual é usada a regra $\vee e$ apenas uma vez na linha 8, com todas as referências correspondentes aos intervalos apresentados. Além disso, podemos notar que as referências nas linhas 5 e 7 também estão corretas. Em comparação com a Figura 40(a), podemos observar também que não existem linhas sem fórmulas.

Figura 40 – Exemplos de demonstrações para $A \to C, B \to C, A \vee B \vdash C$, resultado da compilação do código LaTeX escrito pelo modelo. Erros destacados em vermelho e correções em azul.

1.	$A \vee B$	premissa
2.	$A \to C$	premissa
3.	$B \to C$	premissa
4.	A	hipótese
5.	C	$\to e$ 2, 4
6.	$\vee e$ 1, 4 – 5	
7.	B	hipótese
8.	C	$\to e$ 3, 6
9.	$\vee e$ 1, 6 – 7	
10.	C	$\vee e$ 1, 4-5, 6-7

(a) Demonstração incorreta realizada pelo modelo.

1.	$A \to C$	premissa
2.	$B \to C$	premissa
3.	$A \vee B$	premissa
4.	A	hipótese
5.	C	$\to e$ 4, 1
6.	B	hipótese
7.	C	$\to e$ 6, 2
8.	C	$\vee e$ 3, 4-5, 6-7

(b) Exemplo de demonstração correta.

Fonte: Próprio autor (2025).

A Tabela 2 resume o desempenho do modelo ChatGPT na tarefa de resolução de exercícios de dedução natural em lógica proposicional, onde ele teve 10 acertos num total de 41 exercícios (24,39%). Das questões que possuíam erros, 24 possuíam pelo menos um erro lógico, 16 possuíam pelo menos um erro de referencia e 9 questões possuíam ambos (pelo menos um de cada). Durante a avaliação, 23 questões tiveram revisões na formatação, incluindo 5 das 10 questões avaliadas como corretas. Podemos ressaltar que em todos os casos em que seria necessária a aplicação da regra de redução ao absurdo, a regra não foi aplicada e houve ao menos um erro lógico.

Tabela 2 – Desempenho apresentado pelo modelo na tarefa de dedução natural

Modelo	Acertos	Erros Lógicos	Erros de Referência
GPT 4o	10	24	16
Total (%)	24,39%	58,53%	39,02%

Fonte: Próprio autor (2025).

5.2.2.2 Resultados para os Exercícios em Lógica de Predicados

O desempenho do modelo na resolução de exercícios de lógica de predicados foi pior que os apresentados para a lógica proposicional. Na Figura 41, apresentamos um exemplo de resposta correta produzida pelo modelo na solução do seguinte enunciado: $\exists x(P \to Q(x)) \vdash (P \to \exists x Q(x))$. Podemos observar na resposta a declaração correta da premissa (linha 1), a suposição correta de hipóteses (linhas 2 e 3) e aplicação correta das regras eliminação da implicação ($\to e$), introdução do quantificador existencial ($\exists e$), eliminação do quantificador existencial ($\exists e$) e introdução da implicação ($\to i$) nas linhas 4, 5, 6 e 7 respectivamente. Além disso, as duas caixas (subprovas) foram bem construídas e os intervalos das linhas nas referências foram descritos corretamente.

Figura 41 – Prova correta para $\exists x(P \to Q(x)) \vdash (P \to \exists x Q(x))$, resultado da compilação do código LaTeX escrito pelo modelo.

1.		$\exists x(P \to Q(x))$	premissa
2.		P	hipótese
3.	a	$P \to Q(a)$	hipótese
4.		$Q(a)$	$\to e$ 2, 3
5.		$\exists x Q(x)$	$\exists i$ 4
6.		$\exists x Q(x)$	$\exists e$ 1, 3-5
7.		$P \to \exists x Q(x)$	$\to i$ 2-6

Fonte: Próprio autor (2025).

No entanto. na Figura 42(a), podemos observar um exemplo onde o modelo respondeu incorretamente o exercício $\exists x(P(x) \vee Q(x)) \vdash (\exists x P(x) \vee \exists x Q(x))$. A prova possui o seguinte **erro lógico**:

– Na linha 2, se declara uma variável que é usada em seguida na fórmula $P(a) \vee Q(a)$ de forma correta, porem adiciona a justificativa com a regra de existencial eliminação ($\exists e$);

Como **erros de referência** podemos observar na linha 10 existe uma regra de eliminação do existencial ($\exists e$) e se utiliza uma referência a uma caixa, inexistente na prova, que começaria da linha 1 até a linha 9.

A Figura 42(b) apresenta um exemplo de demostração correta para o exercício $\exists x(P(x) \vee Q(x)) \vdash (\exists x P(x) \vee \exists x Q(x))$: na linha 2, podemos observar a justificativa que é declarada corretamente como hipótese e na linha 10, a justificativa apresenta de forma correta a regra

de eliminação do existencial ($\exists e$) referenciando em seguida a linha 1 e a caixa que vai da linha 2 até a linha 9 (*2-9*).

Figura 42 – Exemplos de demonstrações para $\exists x(P(x) \vee Q(x)) \vdash (\exists x P(x) \vee \exists x Q(x))$, resultado da compilação do código LaTeX escrito pelo modelo. Erros destacados em vermelho e correções em azul.

1.		$\exists x(P(x) \vee Q(x))$	premissa
2.	a	$P(a) \vee Q(a)$	$\exists e$ 1
3.		$P(a)$	hipótese
4.		$\exists x\, P(x)$	$\exists i$ 3
5.		$\exists x P(x) \vee \exists x Q(x)$	$\vee i$ 4
6.		$Q(a)$	hipótese
7.		$\exists x\, Q(x)$	$\exists i$ 6
8.		$\exists x P(x) \vee \exists x Q(x)$	$\vee i$ 7
9.		$\exists x P(x) \vee \exists x Q(x)$	$\vee e$ 2, $3-5$, $6-8$
10.		$\exists x P(x) \vee \exists x Q(x)$	$\exists e$ $1-9$

(a) Exemplo de demonstração incorreta.

1.		$\exists x(P(x) \vee Q(x))$	premissa
2.	a	$P(a) \vee Q(a)$	hipótese
3.		$P(a)$	hipótese
4.		$\exists x P(x)$	$\exists i$ 3
5.		$\exists x P(x) \vee \exists x Q(x)$	$\vee i$ 4
6.		$Q(a)$	hipótese
7.		$\exists x Q(x)$	$\exists i$ 6
8.		$\exists x P(x) \vee \exists x Q(x)$	$\vee i$ 7
9.		$\exists x P(x) \vee \exists x Q(x)$	$\vee e$ 2, 3-5, 6-8
10.		$\exists x P(x) \vee \exists x Q(x)$	$\exists e$ 1,2-9

(b) Exemplo de demonstração correta.

Fonte: Próprio autor (2025).

A Tabela 3 apresenta o desempenho do modelo *GPT 4o* nas 20 questões de Dedução Natural em Lógica de Predicados. Apenas 2 questões estavam corretas, 14 possuíam pelo menos um erro lógico, 8 possuíam pelo menos um erro de referência e 4 questões possuíam ambos (pelo menos um de cada). Durante a avaliação 10 Questões tiveram revisões na formatação, incluindo uma das questões corretas. Assim como em lógica proposicional, em todos os casos em que seria necessário a aplicação da regra de redução ao absurdo a regra não foi aplicada e houve ao menos um erro lógico.

Tabela 3 – Desempenho apresentado pelo modelo na tarefa de dedução natural

Modelo	Acertos	Erros Lógicos	Erros de Referência
GPT 4o	2	14	8
Total (%)	10,0%	70,0%	40,0%

Fonte: Próprio autor (2025).

5.3 Resultados para a Fase 3: Avaliação do Modelo sem Treinamento e Com Contexto

A Figura 43 apresenta um exemplo de resposta correta feita pelo modelo GPT 4.1 mini, em um código aceito pela ferramenta NADIA, para o enunciado $\forall x \neg P(x) \vdash \neg \exists x P(x)$. Podemos observar que o modelo listou de forma correta a única premissa na linha 1, bem como

a suposição correta de duas hipóteses nas linhas 2 e 3, além disso, aplicou diversas regras de forma correta, são essas Eliminação Universal ($\forall e$) linha 4, Negação Eliminação ($\neg e$) linha 5, Eliminação do Existencial ($\exists$ e) linha 6 e Negação Introdução ($\neg i$ linha 7) e referenciou corretamente os intervalos das caixas nas linhas 6 e 7.

Figura 43 – Exemplo Resposta Correta do Modelo GPT 4.1 sem treino.

```
‘‘‘
1. Ax ∼P(x)          pre
2. { Ex P(x)         hip
3. { a P(a)          hip
4. ∼P(a)             Ae 1
5. @                 ∼e 3,4
}
6. @                 Ee 2, 3-5
}
7. ∼Ex P(x)          ∼i 2-6
’’’
```

Fonte: Próprio autor (2025).

A Figura 44 apresenta um exemplo de resposta incorreta feita pelo modelo GPT 4.1 mini, em um código aceito pela Ferramenta NADIA, para o enunciado $\vdash (\neg A \to B) \to ((\neg A \to \neg B) \to A)$. O modelo fez 3 suposições de forma correta nas três primeiras linhas, aplicou corretamente a regra de Implicação Eliminação ($\to$e) nas linhas 4 e 5, e também acertou na negação eliminação ($\neg$e), entretanto, na linha 7 aplica a regra de negação introdução ($\neg$i), o correto seria utilizar a regra de redução ao absurdo (raa), além disso, na linha 8 chega em uma fórmula incorreta ao aplicar a regra de implicação introdução ($\to$i) .

Na Tabela 4, podemos observar o desempenho do modelo *GPT 4.1-mini*, que apresenta um total de 22 acertos em lógica proposicional e apenas 2 acertos em lógica de predicados. O que demonstra uma séria dificuldade em lógica de predicados, assim como, o modelo *GPT 4o* como podemos ver em (Martins *et al.*, 2025).

Tabela 4 – Desempenho do modelo GPT-4.1 mini antes do fine-tuning

Modelo	**Acertos BD1 (41)**	**Acertos em BD3 (20)**
GPT-4.1 mini (sem fine-tuning)	22	2
Total (%)	53.6%	10%

Fonte: Próprio autor (2025).

Figura 44 – Exemplo Resposta Correta do Modelo GPT 4.1 sem treino.

```
‘‘‘
1. { ∼A -> B       hip
2. { ∼A -> ∼B      hip
3. { ∼A       hip
4. B      ->e 1,3
5. ∼B       ->e 2,3
6. @       ∼e 4,5
}
7. A       ∼i 3-6
}
8. A -> ∼B       ->i 2-7
}
9. (∼A->B) -> ((∼A->∼B) -> A)     ->i 1-8
’’’
```

(a) Resposta Incorreta.

```
‘‘‘
1. { ∼A -> B       hip
2. { ∼A -> ∼B      hip
3. { ∼A       hip
4. B      ->e 3,1
5. ∼B       ->e 3,2
6. @       ∼e 4,5
}
7. A       raa 3-6
}
8. (∼A -> ∼B) -> A     ->i 2-7
}
9. (∼A -> B) -> ((∼A -> ∼B) -> A) ->i 1-8
’’’
```

(b) Correção da Resposta.

Fonte: Próprio autor (2025).

5.4 Resultados para a Fase 4 : Avaliação do Modelo Treinado

A Figura 45 apresenta um exemplo de resposta correta feita pelo modelo GPT 4.1 mini treinado, em um código aceito pela Ferramenta NADIA para o enunciado $\forall x(P \rightarrow Q(x)) \vdash P \rightarrow \forall x Q(x)$. Podemos observar que o modelo listou de forma correta a única premissa na linha 1, bem como a suposição correta da hipótese na linha 2 e também supoe uma variável "a"na linha 3, além disso, aplicou as seguintes regras de forma correta: Eliminação do Universal ($\forall e$) linha 4, Implicação Eliminação ($\rightarrow e$) linha 5, Eliminação do Universal ($\forall e$) linha 6 e Implicação Eliminação ($\rightarrow e$) linha 7 e referenciou corretamente os intervalos das caixas nas linhas 6 e 7.

Figura 45 – Exemplo Resposta Correta do Modelo GPT 4.1 com treino.

```
‘‘‘
1. Ax(P->Q(x))          pre
2. { P                  hip
3. { a
4. P -> Q(a)          Ae 1
5. Q(a)          ->e 2,4
}
6. Ax Q(x)          Ai 3-5
}
7. P -> Ax Q(x)          ->i 2-6
’’’
```

Fonte: Próprio autor (2025).

A Figura 46 apresenta um exemplo de resposta incorreta feita pelo modelo GPT 4.1 mini treinado, em um código aceito pela Ferramenta NADIA para o enunciado $\vdash (A \vee (A \wedge B)) \rightarrow A$. O modelo fez duas suposições corretas nas linhas 1,2 e 4, usou a regra do copie corretamente na linha 3, a regra de conjunção eliminação corretamente na linha 5, entretanto, na linha 6 utilizou a regra Implicação Eliminação ($\rightarrow$e) quando deveria ter utilizado a regra Disjunção Eliminação ($\vee$e).

Figura 46 – Exemplo Resposta Incorreta do Modelo GPT 4.1 com treino.

```
‘‘‘
1. { A | (A & B)          hip
2. { A          hip
3. A          copie 2
}
4. { A & B          hip
5. A          &e 4
}
6. A | (A & B)          ->e 1, 2-3, 4-5
}
7. (A | (A & B)) -> A          ->i 1-6
’’’
```

(a) Resposta Incorreta.

```
‘‘‘
1. { A | (A & B)          hip
2. { A          hip
3. A          copie 2
}
4. { A & B          hip
5. A          &e 4
}
6. A          le 1, 2-3, 4-5
}
7. (A | (A & B)) -> A          ->i 1-6
’’’
```

(b) Correção da Resposta.

Fonte: Próprio autor (2025).

Na tabela 5, podemos observar o desempenho do modelo *GPT 4.1-mini* treinado, que apresenta um total de 32 acertos em lógica proposicional e 15 acertos em lógica de predicados. Diferente dos demais modelos bem como sua versão sem treino, esse modelo apresenta uma diferença mínima do percentual de acertos em lógica proposicional (78.0%) e lógica de predicados (75%).

Tabela 5 – Desempenho do modelo GPT-4.1 mini depois do fine-tuning

Modelo	**Acertos BD1(41)**	**Acertos BD3 (20)**
GPT-4.1 mini (com fine-tuning)	32	15
Total (%)	78.0%	75%

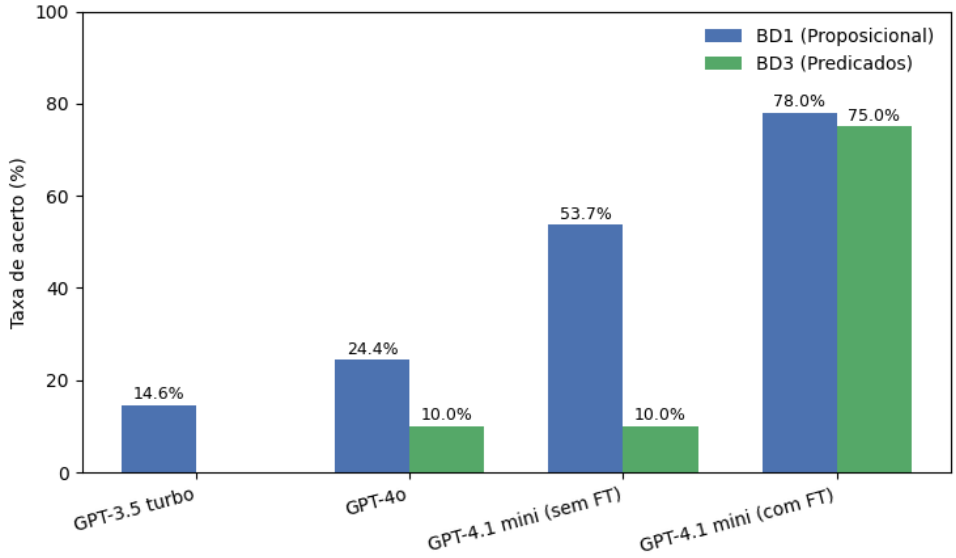
Fonte: Próprio autor (2025).

5.5 Resumo dos Resultados

Nesta seção, reportamos os resultados gerais da resolução dos exercícios das bases de dados BD1 e BD3 utilizando o *GPT 4.1-mini*. Além de compararmos com os resultados

publicados em (Martins *et al.*, 2023) e os resultados publicados em (Martins *et al.*, 2025). Na Figura 47, podemos observar o percentual de acertos de cada modelo em cada base de dados. Podemos observar que houve uma melhora significativa no desempenho em lógica proposicional nos modelos mesmo sem treinamento, e que o treinamento no modelo *GPT 4.1-mini* fez com que a taxa de acerto melhorasse ainda mais. No caso de lógica de predicados não houve melhoras do *GPT 4.1-mini* em relação ao Modelo *GPT 4o*, porém depois do treinamento é possível observar uma melhora significativa no resultado final, dessa forma, podemos avaliar que o treinamento se mostrou como um caminho viável para esse domínio.

Figura 47 – Comparação de Desempenho entre os Modelos LLM.



Fonte: Próprio autor (2025).

Na Tabela 6, podemos observar de forma mais detalhada o desempenho de cada modelo em cada base de dados. No caso das questões de lógica proposicional (BD1), o desempenho inicial foi de 6 acertos depois passou a ser 10 e depois 22, no caso dos modelos sem fine-tuning e no caso do modelo GPT-4.1 mini com fine-tuning o desempenho do modelo foi pra 32. No caso das questões de lógica de predicados (BD3), o desempenho dos modelos GPT-4o e GPT-4.1 mini são iguais (2 acertos), e no caso do modelo GPT-4.1 mini com fine-tuning o desempenho do modelo foi pra 15, o modelo GPT-3.5 turbo não foi testado em lógica de predicados.

Tabela 6 – Comparação de desempenho entre diferentes modelos LLM em tarefas de lógica

Modelo	**Acertos BD1 (41)**	**Acertos BD3 (20)**
GPT-3.5 turbo ((Martins *et al.*, 2023))	6 (14,63%)	-
GPT-4o ((Martins *et al.*, 2025))	10 (24,39%)	2 (10,00%)
GPT-4.1 mini (Atual sem fine-tuning)	22 (53,65%)	2 (10,00%)
GPT-4.1 mini (Atual com fine-tuning)	32 (78,04%)	15 (75,00%)

Fonte: Próprio autor (2025).

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, avaliamos os LLMs *GPT 3.5-turbo, GPT 4o e GPT 4.1-mini* na tarefa de construir provas em Dedução Natural em Lógica Proposicional e Lógica de Predicados sem treinamento e com treinamento no caso do modelo *GPT 4.1-mini*. Após a análise dos resultados, fica evidente que existe uma melhora no desempenho dos modelos nesse domínio, pois, mesmo sem treinamento é possível observar uma melhora nos resultados no caso de Lógica Proposicional, entretanto, essa melhora não se reflete nos resultados de Lógica de Predicados.

No caso de comparação com o modelo treinado, podemos observar que houve uma melhora significativa em todos os casos, o que ressalta a eficácia de um treinamento especifico em comparação com um treinamento apenas em dados com uma ampla variedade de textos.

Como trabalhos futuros, pretendemos: treinar outros LLMs estado da arte - tais como *Deepseek-r1, Gemma3 e Qwen3* - para verificar se eles são capazes de aprender as regras de dedução natural em lógica proposicional e de predicados e avaliar se há melhora no desempenho; utilizar bases de dados diferentes com um conjuntos de dados maiores e; verificar o desempenho do modelo na tarefa de resolução de exercícios de outras temáticas envolvendo lógica proposicional e lógica de predicados tais como resolução de exercícios de argumentação em linguagem natural e exercícios de *tableaux* analíticos.

REFERÊNCIAS

ANDO, R.; MORISHITA, T.; ABE, H.; MINESHIMA, K.; OKADA, M. Evaluating large language models with neubaroco: Syllogistic reasoning ability and human-like biases. **arXiv preprint arXiv:2306.12567**, 2023.

BROWN, T.; MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J. D.; DHARIWAL, P.; NEELAKANTAN, A.; SHYAM, P.; SASTRY, G.; ASKELL, A. *et al.* Language models are few-shot learners. **Advances in neural information processing systems**, v. 33, p. 1877–1901, 2020.

CHOWDHERY, A.; NARANG, S.; DEVLIN, J.; BOSMA, M.; MISHRA, G.; ROBERTS, A.; BARHAM, P.; CHUNG, H. W.; SUTTON, C.; GEHRMANN, S.; SCHUH, P.; SHI, K.; TSVYASHCHENKO, S.; MAYNEZ, J.; RAO, A.; BARNES, P.; TAY, Y.; SHAZEER, N.; PRABHAKARAN, V.; REIF, E.; DU, N.; HUTCHINSON, B.; POPE, R.; BRADBURY, J.; AUSTIN, J.; ISARD, M.; GUR-ARI, G.; YIN, P.; DUKE, T.; LEVSKAYA, A.; GHEMAWAT, S.; DEV, S.; MICHALEWSKI, H.; GARCIA, X.; MISRA, V.; ROBINSON, K.; FEDUS, L.; ZHOU, D.; IPPOLITO, D.; LUAN, D.; LIM, H.; ZOPH, B.; SPIRIDONOV, A.; SEPASSI, R.; DOHAN, D.; AGRAWAL, S.; OMERNICK, M.; DAI, A. M.; PILLAI, T. S.; PELLAT, M.; LEWKOWYCZ, A.; MOREIRA, E.; CHILD, R.; POLOZOV, O.; LEE, K.; ZHOU, Z.; WANG, X.; SAETA, B.; DIAZ, M.; FIRAT, O.; CATASTA, M.; WEI, J.; MEIER-HELLSTERN, K.; ECK, D.; DEAN, J.; PETROV, S.; FIEDEL, N. Palm: Scaling language modeling with pathways. **Journal of Machine Learning Research**, v. 24, n. 240, p. 1–113, 2023. Disponível em: http://jmlr.org/papers/v24/22-1144.html. Acesso em: 10 jan. 2025.

CHUNG, H. W.; HOU, L.; LONGPRE, S.; ZOPH, B.; TAY, Y.; FEDUS, W.; LI, Y.; WANG, X.; DEHGHANI, M.; BRAHMA, S.; WEBSON, A.; GU, S. S.; DAI, Z.; SUZGUN, M.; CHEN, X.; CHOWDHERY, A.; CASTRO-ROS, A.; PELLAT, M.; ROBINSON, K.; VALTER, D.; NARANG, S.; MISHRA, G.; YU, A.; ZHAO, V.; HUANG, Y.; DAI, A.; YU, H.; PETROV, S.; CHI, E. H.; DEAN, J.; DEVLIN, J.; ROBERTS, A.; ZHOU, D.; LE, Q. V.; WEI, J. Scaling instruction-finetuned language models. **Journal of Machine Learning Research**, v. 25, n. 70, p. 1–53, 2024. Disponível em: http://jmlr.org/papers/v25/23-0870.html. Acesso em: 10 jan. 2025.

COQ, P. The coq proof assistant-reference manual. **INRIA Rocquencourt and ENS Lyon, version**, v. 5, 1996.

ENDERTON, H. B. **A mathematical introduction to logic**. [*S. l.*]: Academic Press, 1972. ISBN 978-0-12-238450-9.

FERREIRÓS, J. The road to modern logic—an interpretation. **Bulletin of Symbolic Logic**, Cambridge University Press, v. 7, n. 4, p. 441–484, 2001.

FIRST, E.; RABE, M.; RINGER, T.; BRUN, Y. Baldur: Whole-proof generation and repair with large language models. In: ACM JOINT EUROPEAN SOFTWARE ENGINEERING CONFERENCE AND SYMPOSIUM ON THE FOUNDATIONS OF SOFTWARE ENGINEERING, 31. **Proceedings [...]**. [*S. l.*], 2023. p. 1229–1241.

HAMMER, E. M. Semantics for existential graphs. **Journal of Philosophical Logic**, Springer, v. 27, p. 489–503, 1998.

HAN, J. M.; RUTE, J.; WU, Y.; AYERS, E. W.; POLU, S. Proof artifact co-training for theorem proving with language models. In: **International Conference on Learning Representations**. [*S. l.*: *s. n.*], 2022.

HUTH, M.; RYAN, M. **Logic in Computer Science**: Modelling and reasoning about systems (2nd ed.). [*S. l.*]: Cambridge University Press, 2004.

JIANG, A. Q.; LI, W.; HAN, J. M.; WU, Y. Lisa: Language models of isabelle proofs. In: **6th Conference on Artificial Intelligence and Theorem Proving**. [*S. l.*: *s. n.*], 2021. p. 378–392.

JIANG, A. Q.; LI, W.; TWORKOWSKI, S.; CZECHOWSKI, K.; ODRZYGÓŹDŹ, T.; MIŁOŚ, P.; WU, Y.; JAMNIK, M. Thor: Wielding hammers to integrate language models and automated theorem provers. **Advances in Neural Information Processing Systems**, v. 35, p. 8360–8373, 2022.

KASNECI, E.; SESSLER, K.; KÜCHEMANN, S.; BANNERT, M.; DEMENTIEVA, D.; FISCHER, F.; GASSER, U.; GROH, G.; GÜNNEMANN, S.; HÜLLERMEIER, E. *et al.* Chatgpt for good? on opportunities and challenges of large language models for education. **Learning and Individual Differences**, Elsevier, v. 103, p. 102274, 2023.

LAMPLE, G.; LACROIX, T.; LACHAUX, M.-A.; RODRIGUEZ, A.; HAYAT, A.; LAVRIL, T.; EBNER, G.; MARTINET, X. Hypertree proof search for neural theorem proving. **Advances in neural information processing systems**, v. 35, p. 26337–26349, 2022.

LEWIS, M.; LIU, Y.; GOYAL, N.; GHAZVININEJAD, M.; MOHAMED, A.; LEVY, O.; STOYANOV, V.; ZETTLEMOYER, L. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: JURAFSKY, D.; CHAI, J.; SCHLUTER, N.; TETREAULT, J. (Ed.). **Proceedings [...]**. Online: Association for Computational Linguistics, 2020. p. 7871–7880. Disponível em: https://aclanthology.org/2020.acl-main.703. Acesso em: 10 jan. 2025.

LIU, H.; NING, R.; TENG, Z.; LIU, J.; ZHOU, Q.; ZHANG, Y. Evaluating the logical reasoning ability of chatgpt and gpt-4. **arXiv preprint arXiv:2304.03439**, 2023.

LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L.; STOYANOV, V. Roberta: A robustly optimized bert pretraining approach. **arXiv preprint arXiv:1907.11692**, 2019.

MARTINS, F. L. B.; OLIVEIRA, A. C. A. de; VASCONCELOS, D. R. de; MENEZES, M. V. de. Avaliando a habilidade do chatgpt de realizar provas de deduçao natural em lógica proposicional. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 34. **Anais [...]**. [*S. l.*]: SBC, 2023. p. 1282–1292.

MARTINS, F. L. B.; OLIVEIRA, A. C. A. de; VASCONCELOS, D. R. de; MENEZES, M. V. de. Avaliando a habilidade do chatgpt de realizar provas de dedução natural em lógica proposicional e lógica de predicados. **Revista Brasileira de Informática na Educação**, v. 33, p. 244–278, 2025.

MOURA, L. D.; KONG, S.; AVIGAD, J.; DOORN, F. V.; RAUMER, J. von. The lean theorem prover (system description). In: SPRINGER. **Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25**. [*S. l.*], 2015. p. 378–388.

NIPKOW, T.; WENZEL, M.; PAULSON, L. C. **Isabelle/HOL**: a proof assistant for higher-order logic. [*S. l.*]: Springer, 2002.

OpenAI. **ChatGPT**. 2021. https://openai.com/research/chatgpt. Acesso em: 13 jun. 2023.

PELLETIER, F. J. A brief history of natural deduction. **History and Philosophy of Logic**, Taylor & Francis, v. 20, n. 1, p. 1–31, 1999.

PELLETIER, F. J. A history of natural deduction and elementary logic textbooks. **Logical consequence: Rival approaches**, Hermes Science Publications Oxford, v. 1, p. 105–138, 2000.

POLU, S.; HAN, J. M.; ZHENG, K.; BAKSYS, M.; BABUSCHKIN, I.; SUTSKEVER, I. Formal mathematics statement curriculum learning. **arXiv preprint arXiv:2202.01344**, 2022.

POLU, S.; SUTSKEVER, I. Generative language modeling for automated theorem proving. **arXiv preprint arXiv:2009.03393**, 2020.

RUSSELL, S. J. **Artificial intelligence a modern approach**. [*S. l.*]: Pearson Education, Inc., 2010.

SAPAROV, A.; PANG, R. Y.; PADMAKUMAR, V.; JOSHI, N.; KAZEMI, S. M.; KIM, N.; HE, H. Testing the general deductive reasoning capacity of large language models using ood examples. **arXiv preprint arXiv:2305.15269**, 2023.

SCHICK, T.; DWIVEDI-YU, J.; DESSÌ, R.; RAILEANU, R.; LOMELI, M.; HAMBRO, E.; ZETTLEMOYER, L.; CANCEDDA, N.; SCIALOM, T. Toolformer: Language models can teach themselves to use tools. **Advances in Neural Information Processing Systems**, v. 36, 2024.

SHIKISHIMA, C.; HIRAISHI, K.; YAMAGATA, S.; SUGIMOTO, Y.; TAKEMURA, R.; OZAKI, K.; OKADA, M.; TODA, T.; ANDO, J. Is g an entity? a japanese twin study using syllogisms and intelligence tests. **Intelligence**, Elsevier, v. 37, n. 3, p. 256–267, 2009.

TLILI, A.; SHEHATA, B.; ADARKWAH, M. A.; BOZKURT, A.; HICKEY, D. T.; HUANG, R.; AGYEMANG, B. What if the devil is my guardian angel: Chatgpt as a case study of using chatbots in education. **Smart Learning Environments**, Springer, v. 10, n. 1, p. 15, 2023.

TOUVRON, H.; LAVRIL, T.; IZACARD, G.; MARTINET, X.; LACHAUX, M.-A.; LACROIX, T.; ROZIèRE, B.; GOYAL, N.; HAMBRO, E.; AZHAR, F.; RODRIGUEZ, A.; JOULIN, A.; GRAVE, E.; LAMPLE, G. **LLaMA**: Open and efficient foundation language models. 2023.

VASCONCELOS, D. R. de; MENEZES, M. V. Nadia-natural deduction proof assistant: Um assistente de provas de dedução natural para lógica proposicional e lógica de predicados. **Revista Brasileira de Informática na Educação**, v. 32, p. 842–870, 2024.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017.

WANG, H.; YUAN, Y.; LIU, Z.; SHEN, J.; YIN, Y.; XIONG, J.; XIE, E.; SHI, H.; LI, Y.; LI, L. *et al.* Dt-solver: Automated theorem proving with dynamic-tree sampling guided by proof-level value function. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (VOLUME 1: LONG PAPERS), 61. **Proceedings [...]**. [*S. l.*], 2023. p. 12632–12646.

YANG, K.; DENG, J. Learning to prove theorems via interacting with proof assistants. In: PMLR. **International Conference on Machine Learning**. [*S. l.*], 2019. p. 6984–6994.

YANG, K.; SWOPE, A.; GU, A.; CHALAMALA, R.; SONG, P.; YU, S.; GODIL, S.; PRENGER, R. J.; ANANDKUMAR, A. Leandojo: Theorem proving with retrieval-augmented language models. **Advances in Neural Information Processing Systems**, v. 36, 2024.

ZHAO, W. X.; ZHOU, K.; LI, J.; TANG, T.; WANG, X.; HOU, Y.; MIN, Y.; ZHANG, B.; ZHANG, J.; DONG, Z. *et al.* A survey of large language models. **arXiv preprint arXiv:2303.18223**, 2023.

7 APÊNDICE TEXTO DE REFERÊNCIA SOBRE DEDUÇÃO NATURAL

A seguir apresentamos o o texto complementar utilizado juntamente com as questões de lógica para o experimentos apresentados nesse Trabalho.

Introdução

O guia a seguir detalha o formato aceito pelo NADIA para a construção de provas de Dedução Natural em Lógica Proposicional e Lógica de Primeira Ordem no estilo Fitch:

Importante:

- Os átomos e os predicados são escritos em letras maiúsculas (e.g. `A`, `B`, `H(x)`).

- As variáveis são escritas com a primeira letra em minúsculo, podendo ser seguida de letras e números (e.g. `x`, `x0`, `xP0`).

- Os símbolos de $\bot$,$\vdash$ e os conectivos $\neg$,$\wedge$,$\vee$,$\rightarrow$ são escritos por `@`, `|-`, `~`, `&`, `|`, `->` respectivamente.

- As fórmulas com o $\forall$x e $\exists$x serão representadas por `Ax` e `Ex` (`A` e `E` seguidos da variável `x`). Por exemplo, `Ax (H(x)->M(x))` representa $\forall$x (H(x)$\rightarrow$M(x)).

- A ordem de precedência dos quantificadores e dos conectivos lógicos é definida por $\neg$,$\forall$,$\exists$,$\wedge$,$\vee$,$\rightarrow$ com alinhamento à direita. Por exemplo:

  - A fórmula `~A&B->C` representa a fórmula (((¬A)$\wedge$B)$\rightarrow$C).

  - O teorema `~Ax P(x) |- Ex ~P(x)` representa o teorema (¬(($\forall$x) P(x)))$\vdash$(($\exists$x) (¬P(x))).

- As palavras *Premissa* e *Hipótese* são representadas por `pre` e `hip`, respectivamente.

Dedução Natural no Estilo de Fitch

O sistema de Dedução Natural é um mecanismo que permite a construção de uma prova formal, estabelecendo uma conclusão φ a partir de um conjunto de premissas $\Gamma = \{\varphi_1, \varphi_2, \cdots, \varphi_n\}$, denotado por $\Gamma \vdash \varphi$, aplicando-se sucessivamente **regras de demonstração**.

Em Dedução Natural no estilo de Fitch, as demonstrações são apresentadas de forma linear e sequencial, na qual cada uma das linhas da prova é numerada, tem uma afirmação e uma justificativa. As justificativas são definidas por serem *premissas* da prova ou pela aplicação de uma das *regras do sistema dedutível*. As subprovas dentro de uma prova maior têm *caixas* ao redor e servem para delimitar o escopo de hipóteses temporárias. Provas podem ter caixas dentro de caixas, ou pode-se abrir outras caixas depois de fechar outras, obedecendo as regras de demonstração. Uma fórmula só pode ser utilizada em uma prova em um determinado ponto se essa fórmula aconteceu anteriormente e se nenhuma caixa que contenha essa ocorrência da fórmula tenha sido fechada.

A seguir iremos apresentar todas as regras do sistema de Dedução Natural no Estilo de Fitch.

Explicar cada regra:

Regra das Premissas

O primeiro passo em uma demonstração em Dedução Natural no estilo *Fitch* é enumerar as premissas da prova. Abaixo exibe-se a estrutura geral da regra das premissas, na qual $X1, X2, \ldots, Xn$ são representadas em uma linha cada, seguindo uma numeração sequencial e como justificativa "premissa" (pre).

```
1. X1  pre

2. X2  pre

⋮

n. Xn  pre

⋮
```

Regras da Conjunção

A regra da introdução da conjunção (&i) permite concluir a fórmula A&B em uma linha p se A e B foram demonstradas nas linhas m (ou n) e n (ou m), respectivamente, anteriores a linha p e que não foram descartadas. Abaixo exibe-se a aplicação &i da fórmula A&B na linha 3 a partir das fórmulas A e B, definidas nas linhas 1 e 2, respectivamente, que são anteriores a linha 3.

NADIA - Exemplo: A,B|-A&B

```
1. A     pre

2. B     pre

3. A&B   &i 1,2
```

A regra da **eliminação da conjunção (&e)** permite concluir a fórmula A (ou B) na linha m a partir da eliminação à esquerda (ou à direita) da conjunção da fórmula A&B da linha n (anterior a p e não foi descartada). Abaixo exibe-se uma aplicação da regra na qual A é obtida na linha 3 pela eliminação da conjunção à esquerda da fórmula A&B da linha 1.

NADIA - Exemplo: A&B,C|-A&C

```
1. A&B    pre

2. C      pre

3. A      &e 1

4. A&C    &i 3,2
```

Regras da Implicação

A regra da **eliminação da implicação (->e)**, também conhecida como *Modus Ponens*, permite concluir a fórmula B na linha p a partir da eliminação da implicação da fórmula A->B da linha m (ou n) e A da linha n (ou m), anteriores a p e não descartadas. Abaixo exibe-se uma aplicação da regra na qual a fórmula C é obtida na linha 4 pela eliminação da implicação das fórmulas B e B->C das linha 3 e 2, respectivamente.

NADIA - Exemplo: A&B,B->C|-C

```
1. A&B    pre

2. B->C   pre

3. B      &e 1

4. C      ->e 3,2
```

A regra da **introdução da implicação (->i)** constrói condicionais que não ocorrem como premissas. Para construção de um condicional é necessário realizar *raciocínio hipotético*, isto é, supor *temporariamente* que uma dada fórmula é verdadeira. Chamamos esta fórmula de *hipótese*. Assim, utilizamos *caixas de demonstração*, que servem para delimitar o *escopo da hipótese temporária*. Nesta regra, para provar o condicional A->C na linha n+1, colocamos A como **hipótese** no topo de uma caixa (linha m), aplicamos um número finito de regras de forma a obter C na linha n. Todo

o raciocínio para obter C depende da veracidade de A e, por isso, as fórmulas resultante deste raciocínio ficam delimitadas na caixa. Na linha seguinte (n+1) podemos aplicar a regra ->i para obter A→C, sendo que este condicional não mais depende da hipótese A. Na justificativa da linha n+1 utilizamos o nome da regra seguido das linhas inicial e final da caixa (->i m-n). Abaixo exibe-se uma aplicação da regra na qual a fórmula A->C é obtida na linha 6 a partir da caixa das linhas 3 a 5 em que A é a hipótese.

Observe que para a obtenção de C é possível utilizar quaisquer outras fórmulas como premissas e conclusões provisórias feitas até então. Demonstrações podem ter caixas dentro de caixas, ou pode-se abrir novas caixas depois de fechar outras. No entanto, existem regras sobre quais fórmulas podem ser utilizadas em que ponto na demonstração. Em geral, só podemos usar uma fórmula em um determinado ponto se esta fórmula ocorre *antes* desse ponto e se nenhuma caixa que contenha a ocorrência desta fórmula tenha sido fechada.

NADIA - Exemplo: A->B,B->C|-A->C

```
1. A->B      pre

2. B->C      pre

3. {   A     hip

4.     B     ->e 3,1

5.     C     ->e 4,2

   }

6. A->C      ->i 3-5
```

Regras da Disjunção

A regra da **introdução da disjunção (|i)** permite concluir a fórmula A|B em uma linha p se A (ou B) ocorre em uma linha m anterior a p e que não foi descartada. Abaixo exibe-se a aplicação da introdução da disjunção na linha 3 com a introdução de A|B a partir da fórmula A definida na linha 2.

NADIA - Exemplo: (A∨B)→C⊢A→C

```
1. (A|B)->C      pre
2. {   A         hip
3.     A|B       |i 2
4.     C         ->e 3,1
   }
5. A->C          ->i 2-4
```

A regra da **disjunção eliminação (|e)**, permite concluir uma fórmula X na linha p+1 se eliminarmos a disjunção da fórmula A|B na linha m e se fizermos a suposição de A em uma caixa, na linha m, e a suposição de B em outra caixa, na linha n+1, tal que ambas as caixas tenham como conclusão X, nas linhas n e p, respectivamente, por meio de uma sequência finita de passos (regras). Note que essa regra se assemelha a introdução da implicação no sentido de que fazemos raciocínio hipotético, nas caixas de (m+1)−n e (n+1)−p. Abaixo exibe-se a aplicação da eliminação da disjunção na linha 8, na qual concluímos C, a partir da disjunção de A|B na linha 3 e das caixas 4−5 e 6−7 onde supomos A na linha 4 e concluímos C na linha 5 e supomos B na linha 6 e concluímos C na linha 7.

NADIA - Exemplo: A->C,B->C,(A|B)|-C

```
1. A->C       pre
2. B->C       pre
3. A|B        pre
4. {   A      hip
5.     C      ->e 4,1
   }
6. {   B      hip
7.     C      ->e 6,2
```

```
   }

8. C          |e 3, 4-5, 6-7
```

Regras da Negação

A regra da **negação eliminação (~e)** envolve a noção de **contradição**. Contradições são expressões da forma X&~X ou ~X∧X onde X é qualquer fórmula da lógica proposicional. A fórmula @ é utilizada para denotar uma contradição e este fato é expresso na regra ~e. Nesta regra temos que uma fórmula A na linha m (ou n) e a sua negação ¬A na linha n (ou m) podem ser combinadas para aparecimento da contradição @ na linha p com a aplicação da regra ~e. Abaixo exibe-se uma aplicação da regra na qual a contradição @ é obtida na linha 3 devido às fórmulas A na linha 1 e ¬A na linha 2.

NADIA - Exemplo: A,¬A⊢⊥

```
1. A     pre

2. ~A    pre

3. @     ~e 1,2
```

A regra da **negação introdução (¬i)**, é uma regra que envolve raciocínio hipotético e contradição. Se tomarmos A como hipótese (linha m) e, após a aplicação de um número finito de regras, chegarmos a uma contradição @ na linha n, significa que a hipótese não pode ser verdadeira. Desse modo, finalizamos nosso raciocínio hipotético introduzindo a negação na hipótese e obtendo ~A na linha n+1. Abaixo exibe-se um exemplo da aplicação da regra ~i, para provamos ~A na linha 6, assumimos A como hipótese no topo da caixa na linha 3 e chegamos a uma contradição no final da caixa na linha 5.

NADIA - Exemplo: A->B,~B|-~A

```
1. A->B      pre

2. ~B        pre
```

```
3. {   A     hip

4.     B     ->e 1,3

5.     @     ~e 2,4

   }

6. ~A        ~i 3-5
```

Regra da Contradição

A **regra da contradição eliminação** permite concluir uma fórmula qualquer B na linha n se demonstramos em uma linha m, anterior a n, a contradição. Abaixo exibe-se a demonstração de B na linha 4 a partir da eliminação da contradição @ da linha 3.

NADIA - Exemplo: |- A->(~A->B)

```
1. {   A         hip

2.     {   ~A    hip

3.         @     ~e 1,2

4.         B     @e 3

       }

5.     ~A->B     ->i 2-4

   }

6. A->(~A->B)    ->i 1-5
```

Regra de Redução ao Absurdo

A regra de **redução ao absurdo** é uma regra na qual para provarmos uma fórmula X em uma linha n+1, iremos supor temporariamente a negação da fórmula, ~X, em uma caixa que inicia na linha m e que conclui a contradição, @, na linha n, após uma sequência de aplicações de regras. Abaixo exibe-se a demonstração de A|~A, também conhecido como terceiro-excluído. Para provarmos A|~A na linha 8, fazemos a suposição de ~(A|~A), na linha 1 (início da caixa) e concluímos a contradição @, na linha 7 (fim da caixa).

NADIA - Exemplo: ⊢A∨¬A

```
1. {   ~(A|~A)     hip

2.     {   ~A      hip

3.         A|~A    |i 2

4.         @       ~e 3,1

       }

5.     A           raa 2-4

6.     A|~A        |i 5

7.     @           ~e 6,1

   }

8. A|~A          raa 1-7
```

Regra do Copie

A regra do **copie**, apresentada, é necessária, no estilo de Fitch, para permitir concluir uma caixa com uma fórmula que já apareceu anteriormente na demonstração. Abaixo exibe-se que, para demonstrar A->B, na linha 10, é preciso que a caixa que justifica a introdução da implicação inicie com A, linha 8, e termine com B, linha 9. Ocorre que a justificativa de B já havia sido realizada e, portanto, a justificativa da linha 9 é a cópia da fórmula da linhas 7.

NADIA - Exemplo: ~A|B ⊢A->B

```
1. ~A|B      pre

2. { ~A      hip

3.   { A     hip

4.     @     ~e 2,3

5.     B     @e 4

      }

6.   A->B    ->i 3-5

    }

7. { B       hip

8.   { A     hip

9.     B     copie 7

      }

10.  A -> B  ->i 8-9

    }

11. A -> B |e 1,2-6,7-10
```

Regras do Universal

A regra da **eliminação do universal (Ae)** permite concluir a fórmula Fxt em uma linha p se ∀xF foi demonstrada na linha m, desde que o termo t seja substituível para a variável x em F. Abaixo exibe-se a aplicação de Ae da fórmula H(s) ->M(s) na linha 3 a partir da fórmula Ax(H(x)->M(x)), definida na linha 1.

NADIA - Exemplo: Ax(H(x)->M(x)),H(s)|-M(s)

```
1. Ax(H(x)->M(x))   pre

2. H(s)             pre

3. H(s)->M(s)       Ae 1

4. M(s)             ->e 2,3
```

A regra da **introdução do universal (Ai)**, é a regra na qual para provarmos AxF na linha n+1, iremos supor que para uma variável "a" qualquer (arbitrária) em uma caixa que inicia na linha m e que conclui Fxa, na linha n. Dizemos que a variável "a" é qualquer se ela é uma variável nova na linha m, ou seja, a não ocorre como variável livre de qualquer fórmula que aconteça anteriormente a linha m que não esteja em uma caixa fechada. Abaixo exibe-se a introdução do universal na linha 7 para provar AxM(x) a partir da suposição de a, no início da caixa, na linha 3, e demonstramos M(a) ao final da caixa, na linha 6. Note que a variável a escolhida não é uma variável livre das fórmulas das linhas 1 e 2.

NADIA - Exemplo: Ax(H(x)->M(x)),AxH(x)|-∀xM(x)

```
1. Ax(H(x)->M(x))        pre

2. Ax H(x)               pre

3. { a

4.      H(a)->M(a)       Ae 1

5.      H(a)             Ae 2

6.      M(a)             ->e 4,5

   }

7. Ax M(x)               Ai 3-6
```

Regras do Existencial

A regra da **introdução do existencial (Ei)**, é a regra na qual a fórmula ExF pode ser concluída em uma linha p se Fxt foi demonstrada na linha m, desde que o termo t seja substituível para a variável x em F. Abaixo exibe-se a aplicação Ei da fórmula ExP(x) na linha 3 a partir da fórmula P(a), definida na linha 1.

NADIA - Exemplo: P(a),ExP(x)->B|-B

```
1. P(a)              pre

2. Ex P(x) -> B      pre

3. Ex P(x)           Ei 1

4. B                 ->e 3,2
```

A regra da **eliminação do existencial (Ee)**, é a regra na qual para provarmos uma fórmula α, na linha p+1, iremos eliminar o existencial da fórmula ExF, na linha m, supondo a fórmula Fxa para alguma variável a em uma caixa que inicia na linha n e que concluiu com uma fórmula α ao final da caixa, na linha p, desde que "a" não ocorra na conclusão α. Nesta regra sabemos que a fórmula F vale para algum elemento. Entretanto, não podemos assumir nenhuma propriedade específica para esta variável. Assim, a variável a deve ser uma variável nova na linha n, ou seja, a não ocorre como variável livre de qualquer fórmula que aconteça anteriormente a linha n que não esteja em uma caixa fechada e nem pode estar na conclusão α da regra.

Abaixo exibe-se a eliminação do existencial para concluir @ na linha 9, a partir da fórmula ExH(x), na linha 3, e pela caixa que inicia na linha 4 com a suposição da fórmula H(a) com a (nova) variável a e que termina a caixa na linha 8 com a conclusão @.

NADIA - Exemplo: Ax(H(x)->M(x)),Ax~M(x)|-~AxH(x)

```
1. Ax(H(x)->M(x))         pre

2. Ax ~M(x)               pre

3. {  Ex H(x)             hip

4.    { a   H(a)          hip
```

```
5.          H(a)->M(a)    Ae 1

6.          M(a)          ->e 4,5

7.          ~M(a)         Ae 2

8.          @             ~e 6,7

      }

9.    @                   Ee 3, 4-8

   }

10.~Ex H(x)              ~i 3-9
```

Erros nas Regras do Universal e do Existencial

Importante ressaltar que as restrições impostas as regras da introdução do universal e da eliminação do existencial são fundamentais para que todas as demonstrações sejam corretas. Na sequência, iremos apresentar alguns exemplos de demonstrações incorretas que não observam as restrições podem nos conduzir a conclusões erradas a partir de um conjunto de premissas. Os exemplos serão definidos no NADIA e os respectivos erros serão apontados pela ferramenta.

NADIA - Exemplo de Demonstração incorreta: P(a)|-AxP(x)

Erro: Na introdução do universal, a variável escolhida não pode ocorrer anteriormente.

```
1. P(a)           pre

2. { a

3.      P(a)      copie 1

   }

4. Ax P(x)        Ai 2-3
```

NADIA - Exemplo de Demonstração incorreta: ExP(x)|-P(a)

Erro: Na conclusão da eliminação do existencial, a variável escolhida na eliminação não pode ocorrer.

```
1. Ex P(x)   pre

2. { a  P(a) hip

   }

3. P(a)      Ee 1, 2-2
```

NADIA - Exemplo de Demonstração incorreta: Ex Par(x),Ex Impar(x)|-Ex(Par(x)&Impar(x))

Erro: Na eliminação do existencial, a variável escolhida na eliminação já ocorria anteriormente.

```
1. Ex PAR(x)                      pre

2. Ex IMPAR(x)                    pre

3. { a  PAR(a)                    hip

4.      { a   IMPAR(a)            hip

5.            PAR(a)&IMPAR(a)     &i 3,4

6.            Ex(PAR(x)&IMPAR(x)) Ei 5

        }

7.      Ex(PAR(x)&IMPAR(x))       Ee 2, 4-6

   }

8. Ex (PAR(x)&IMPAR(x))           Ee 1, 3-7
```

NADIA - Exemplo de Demonstração incorreta: AyEx MENOR(y,x)|-Ex MENOR(x,x)

Erro: Na eliminação do existencial, a variável escolhida na eliminação já ocorria anteriormente.

```
1. Ay Ex MENOR(y,x)       pre

2. Ex MENOR(a,x)          Ae 1

3. { a  MENOR(a,a)        hip

4.      Ex MENOR(x,x)     Ei 3

   }

5. Ex MENOR(x,x)          Ee 2, 3-4
```
