Aplicação do Provador de Teoremas LEAN no Ensino de Lógica Matemática para Alunos de Sistemas de Informação: Um Estudo de Caso

Willian B. Vaneli[1], **Jefferson O. Andrade**[2]

[1] Campus Serra – Instituto Federal do Espírito Santo (Ifes)
Av. dos Sabiás, 330 – Morada de Laranjeiras, Serra – ES, 29166-630 – Brasil

[2]Programa de Pós-graduação em Computação Aplicada (PPComp)
Campus Serra – Instituto Federal do Espírito Santo (Ifes)
Av. dos Sabiás, 330 – Morada de Laranjeiras, Serra – ES, 29166-630 – Brasil

willianvaneli@gmail.com, jefferson.andrade@ifes.edu.br

Abstract. *This paper investigates the use of the LEAN theorem prover in teaching mathematical logic to Information Systems students. Although the quantitative evaluation of the impact of LEAN on student learning was postponed due to a strike, this study documents in detail the implementation of propositional logic problems using LEAN. The focus is on the application of LEAN to solve selected problems and on the analysis of how the tool can be used to explore logical concepts. This approach serves as a basis for future empirical evaluations of the effectiveness of this methodology in teaching mathematical logic.*
Keywords: *Mathematical Logic, LEAN Theorem Prover, Information Systems Education, Computational Tools, Educational Methodology.*

Resumo. *Este trabalho investiga o uso do provador de teoremas LEAN no ensino de lógica matemática para estudantes de Sistemas de Informação. Embora a avaliação quantitativa do impacto do LEAN no aprendizado dos alunos tenha sido adiada devido a uma greve, este estudo documenta detalhadamente a implementação de problemas de lógica proposicional usando o LEAN. O foco está na aplicação do LEAN para resolver problemas selecionados e na análise de como a ferramenta pode ser usada para explorar conceitos lógicos. Essa abordagem serve como base para futuras avaliações empíricas sobre a eficácia dessa metodologia no ensino de lógica matemática.*
Palavras-chave: *Lógica Matemática, Provador de Teoremas LEAN, Ensino de Sistemas de Informação, Ferramentas Computacionais, Metodologia Educacional.*

1. Introdução

A lógica matemática é fundamental no currículo dos cursos de Sistemas de Informação, fornecendo a base teórica para o desenvolvimento de algoritmos, estruturas de dados e sistemas de informação complexos [Huth and Ryan 2007, Gersting 2019]. Entretanto, a natureza abstrata desses conceitos pode dificultar a compreensão dos estudantes, impactando negativamente seu engajamento e desempenho acadêmico

[Gersting 2019]. Nesse cenário, o uso de ferramentas computacionais, como assistentes de provas, surge como uma abordagem promissora para melhorar o ensino de lógica [Feofiloff and Szwarcfiter 2009].

O provador de teoremas LEAN utiliza a teoria de tipos dependentes para formalizar e verificar teoremas matemáticos [De Moura et al. 2015]. Sua linguagem de programação funcional, combinada com um sistema de tipos avançado, e com um ambiente de programação iterativo, proporciona um ambiente de aprendizado interativo que facilita a visualização e compreensão dos conceitos lógicos, tornando o aprendizado mais acessível e envolvente [Avigad and Massot 2024].

Este trabalho investiga a aplicação do LEAN no ensino de lógica matemática para alunos de Sistemas de Informação, visando melhorar a compreensão e o engajamento dos estudantes. Problemas de lógica proposicional foram selecionados e resolvidos utilizando o LEAN, com os resultados documentados para analisar sua eficácia no aprendizado e envolvimento dos alunos.

2. O Provador de Teoremas LEAN

O LEAN é construído sobre um sistema de tipos poderoso, fundamentado na teoria de tipos dependentes, especificamente uma versão conhecida como o *Cálculo de Construções*. Este sistema de tipos permite que os tipos dependam de valores, possibilitando um framework rico e expressivo tanto para programação quanto para verificação formal [De Moura et al. 2015]. Projetado para ser conciso e legível, o LEAN facilita a escrita de provas que imitam o raciocínio matemático informal, mantendo a precisão necessária [Avigad et al. 2015].

O LEAN utiliza um sistema de tipos baseado na *teoria de tipos dependentes* (*Dependent Type Theory*), que é uma extensão dos sistemas de tipos tradicionais onde os tipos podem depender de valores [De Moura et al. 2015]. Esse sistema de tipos possibilita a construção de provas ao tratar proposições como tipos e provas como termos desses tipos. Uma proposição p é representada como um tipo $p : \mathsf{Prop}$, e uma prova de p é um termo α tal que $\alpha : p$. Em outras palavras, para construir uma prova para um determinado teorema , e.g., p, temos que escrever código que gere um valor do tipo correspondente, i.e., p. Essa abordagem unifica a programação e a prova de teoremas, permitindo que o LEAN verifique a correção lógica automaticamente.

Por exemplo, considere uma proposição simples como "para todo número natural n, existe um número natural m tal que $m = n + 1$". Em LEAN, isso pode ser expressado como:

$$\forall n : \mathbb{N}, \exists m : \mathbb{N}, m = n + 1$$

Aqui, $\forall$ (para todo) e $\exists$ (existe) são quantificadores que podem ser diretamente representados na teoria de tipos dependentes. A habilidade de modelar tal estrutura lógica complexa como tipos faz do LEAN uma poderosa ferramenta para a formalização e verificação de teoremas.

A teoria de tipos dependentes do LEAN suporta a verificação automática de provas, garantindo que cada etapa da prova seja lógica e consistente, minimizando er-

ros humanos e permitindo a exploração interativa de conceitos matemáticos complexos. A arquitetura modular e extensível do LEAN permite que os usuários personalizem o sistema conforme suas necessidades, enquanto sua comunidade ativa contribui com extensas bibliotecas que ampliam suas capacidades [Avigad et al. 2017]. Ao integrar o LEAN no currículo de Sistemas de Informação, não apenas se facilita o aprendizado de lógica matemática, mas também se contribui para a formação de profissionais mais preparados e competentes no campo da tecnologia da informação [Feofiloff and Szwarcfiter 2009, Cherniavsky and Heeren 2017]. O uso do LEAN exemplifica como ferramentas computacionais podem enriquecer a experiência de aprendizagem, permitindo que os alunos apliquem conceitos teóricos em situações práticas, promovendo uma educação mais dinâmica e eficaz [Charles Fadel 2015].

2.1. Visão Geral do Sistema de Tipos

Alguns dos principais conceitos que caracterizam o sistema de tipos do LEAN são:

- *Tipos Dependentes* – No LEAN, os tipos são cidadãos de primeira classe, o que significa que podem ser manipulados como outros valores. Isso permite a criação de tipos que podem expressar afirmações matemáticas e especificações complexas. Por exemplo, é possível definir um tipo que representa números naturais junto com uma propriedade que deve ser válida para esses números.
- *Tipos Indutivos* – LEAN oferece suporte a tipos indutivos, que são essenciais para definir estruturas de dados recursivas. Essa capacidade permite que os usuários construam tipos que podem ser definidos em termos de si mesmos, como listas e árvores, que são cruciais para representar conceitos matemáticos.
- *Universos* – LEAN emprega uma hierarquia de universos para evitar paradoxos associados a tipos autorreferenciais. Cada tipo pertence a um universo, e os próprios tipos podem habitar universos superiores. Essa estrutura possibilita a formulação de tipos mais complexos, mantendo a consistência.
- *Inferência e Elaboração de Tipos* – O sistema de tipos do Lean inclui mecanismos para inferência de tipos, permitindo que o sistema deduza automaticamente os tipos das expressões. A elaboração é um processo em que o Lean transforma especificações de alto nível em representações mais detalhadas que podem ser verificadas quanto à correção. Esse recurso ajuda a simplificar o processo de prova, reduzindo o ônus sobre o usuário.

2.2. Construção de Provas

2.2.1. Prova de Teoremas Interativa

LEAN facilita a prova de teoremas de forma interativa, onde os usuários podem construir provas passo a passo. O sistema fornece feedback sobre a validade de cada etapa, permitindo que os usuários refinem suas provas iterativamente. Essa interação é crucial para enfrentar provas complexas, ajudando os usuários a navegar por estruturas lógicas intrincadas.

2.2.2. Exemplo de Prova

A Fig. 1 apresenta um exemplo simples de um teorema e sua prova em LEAN. Neste exemplo, o teorema afirma que a operação de conjunção é comutativa.[1] A prova constrói os componentes necessários extraindo as proposições individuais da conjunção e, em seguida, reagrupando-as na ordem desejada.

```
theorem and_commutative (p q : Prop) : p ∧ q → q ∧ p :=
  fun hpq : p ∧ q ⇒
    have hp : p := And.left hpq
    have hq : q := And.right hpq
    show q ∧ p from And.intro hq hp
```

Figura 1. Exemplo de código LEAN para prova parcial da comutatividade da conjunção.

2.2.3. Automação e Táticas

LEAN inclui várias táticas e métodos automatizados que auxiliam na construção de provas. Essas ferramentas podem simplificar termos, realizar reescrita de termos e gerenciar tarefas de raciocínio complexas. Os usuários também podem definir táticas personalizadas para automatizar estratégias de prova repetitivas, aumentando a eficiência.

3. Metodologia

A metodologia deste estudo foi delineada em quatro etapas principais: (i) seleção de problemas, (ii) implementação das soluções no LEAN, e (iii) documentação do processo.

A *seleção dos problemas* foi conduzida com o objetivo de abranger tópicos fundamentais de lógica proposicional. Os problemas foram escolhidos com base em critérios de representatividade e complexidade, assegurando que fossem desafiadores, mas acessíveis para os alunos. Esta etapa foi crucial para garantir que os conceitos abordados fossem relevantes para o desenvolvimento do raciocínio lógico dos estudantes.

A etapa de *implementação* envolveu a utilização do provador de teoremas LEAN. Inicialmente, o ambiente de programação foi configurado, e os alunos foram introduzidos à sintaxe e funcionalidades do LEAN. Cada problema selecionado foi formalizado e verificado dentro do ambiente, permitindo uma exploração interativa e prática dos conceitos lógicos. A documentação detalhada das soluções incluiu as etapas do processo e as estratégias de resolução adotadas.

A *documentação do processo* incluiu um registro abrangente das soluções desenvolvidas, dificuldades enfrentadas e melhorias observadas no envolvimento e na compreensão dos conceitos pelos alunos. Essa documentação não apenas forneceu uma base sólida para a análise crítica da eficácia do método, mas também serviu como um recurso valioso para futuras pesquisas na área de ensino de lógica com ferramentas computacionais.

[1] A versão completa do teorema seria: $p \wedge q \leftrightarrow q \wedge p$. Entretanto, apresentamos, sem perda de generalidade, a prova de apenas uma das direções do bicondicional.

4. Resultados e Discussão

Este estudo focou na seleção de problemas de lógica matemática, na implementação de suas soluções utilizando o provador de teoremas LEAN, e na documentação detalhada do processo. Devido à greve de professores das universidades e institutos federais de 2024, a avaliação quantitativa do impacto do uso do LEAN no aprendizado dos alunos não pôde ser realizada. No entanto, foram observadas importantes implicações qualitativas durante as etapas realizadas.

Os problemas selecionados foram escolhidos com base em sua relevância e complexidade adequada ao nível dos alunos de Sistemas de Informação. Os critérios de seleção visaram garantir que os problemas abordassem conceitos fundamentais e desafiadores, servindo como uma base sólida para explorar a eficácia do LEAN. Os problemas selecionados foram:

1. $p \wedge q \to q \wedge p$ (comutatividade da conjunção);
2. $p \vee q \to q \vee p$ (comutatividade da disjunção);
3. $(p \wedge q) \wedge r \to p \wedge (q \wedge r)$ (associatividade da conjunção);
4. $(p \vee q) \vee r \to p \vee (q \vee r)$ (associatividade da disjunção);
5. $p \wedge (q \vee r) \to (p \wedge q) \vee (p \wedge r)$ (distributividade da conjunção);
6. $p \vee (q \wedge r) \to (p \vee q) \wedge (p \vee r)$ (distributividade da disjunção);
7. $(p \to (q \to r)) \to (p \wedge q \to r)$ (exportação);
8. $((p \vee q) \to r) \to (p \to r) \wedge (q \to r)$ (distributividade da implicação sobre disjunção);
9. $\neg(p \vee q) \leftrightarrow (\neg p \wedge \neg q)$ (lei de De Morgan 1);
10. $\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$ (lei de De Morgan 2);
11. $\neg(p \wedge \neg p)$ (lei da não contradição);
12. $(\neg p \vee q) \to (p \to q)$ (lei da implicação 1)
13. $p \wedge \neg q \to \neg(p \to q)$ (lei da implicação 2);
14. $\neg p \to (p \to q)$ (princípio da explosão);
15. $p \vee \bot \to p$ (identidade da disjunção);
16. $p \wedge \bot \to \bot$ (nulidade da conjunção);
17. $(p \to q) \leftrightarrow (\neg q \to \neg p)$ (equivalência contrapositiva);

Os problemas selecionados para o estudo foram escolhidos não apenas pela sua relevância lógica, mas também pelo seu potencial de engajamento. A interatividade proporcionada pelo LEAN, combinada com a natureza iterativa da construção de provas, oferece aos alunos uma forma mais prática e visual de entender conceitos abstratos, como comutatividade e associatividade. Embora a avaliação quantitativa não tenha sido possível, observamos que o processo de construção de provas com LEAN incentiva os alunos a participarem ativamente da aprendizagem, promovendo uma abordagem mais dinâmica e envolvente.

A implementação das soluções foi realizada no ambiente LEAN. Durante a implementação, observou-se que o LEAN oferece uma abordagem sistemática e rigorosa para a formalização e verificação de teoremas. A experiência prática permitiu identificar as vantagens e desafios do uso do LEAN, como a clareza na representação dos conceitos e a curva de aprendizado associada ao domínio da ferramenta.

A documentação detalhada foi criada para cada solução implementada, incluindo as etapas de desenvolvimento, estratégias utilizadas e desafios enfrentados. Este registro

abrangente serve como um recurso valioso para futuras análises e oferece insights sobre a aplicabilidade do LEAN em ambientes educacionais. A documentação também destaca as dificuldades técnicas e conceituais encontradas, fornecendo um panorama claro do processo. Abaixo damos dois exemplos das soluções criadas para os problemas propostos

4.1. Exemplo: Comutatividade da Disjunção

Este problema consiste em provar que a conjunção é comutativa. Ou seja, provar que $p \vee q \leftrightarrow q \vee p$.

```
1  variable (p q : Prop)
2
3  example : p ∨ q ↔ q ∨ p :=
4    Iff.intro
5      (fun h₁: p ∨ q ⇒
6        Or.elim h₁
7          (fun hp: p ⇒ Or.intro_right q hp)
8          (fun hq: q ⇒ Or.intro_left p hq))
9      (fun h₂: q ∨ p ⇒
10       Or.elim h₂
11         (fun hq: q ⇒ Or.intro_right p hq)
12         (fun hp: p ⇒ Or.intro_left q hp))
```

Figura 2. Prova em LEAN da comutatividade da disjunção.

Na Figura 2, a linha 1 declara as variáveis proposicionais p e q que serão usadas na prova. A linha 3 declara que o exemplo que stamos construindo deve ser do tipo $p \vee q \leftrightarrow q \vee p$. O que significa que precisamos construir, usando a linguagem do LEAN, um valor deste tipo. Este valor é a prova. Como primeiro passo, precisamos construir um bicondicional. Para isso, na linha 4, usamos a função `Iff.into`. Esta função tem a assinatura `Iff.intro` : $(a \to b) \to (b \to a) \to (a \leftrightarrow b)$, ou seja, ela recebe um valor do tipo $a \to b$ e um outro valor do tipo $b \to a$ e retorna um valor do tipo $a \leftrightarrow b$. Assim, para provar $p \vee q \leftrightarrow q \vee p$, precisamos de um valor do tipo $p \vee q \to q \vee p$ e de outro valor do tipo $q \vee p \to p \vee q$.

Em LEAN, para construir uma prova de que $a \to b$, precisamos construir uma função que receba um valor do tipo a e retorne um valor do tipo b. Então, para o primeiro argumento da função `Iff.intro`, precisamos construir uma função que receba um valor do tipo $p \vee q$ e retorne um valor do tipo $q \vee p$. O código nas linha 5 a 8 faz exatamente isso. Na linha 5 da Fig. 2 declaramos uma função que recebe um argumento chamado h_1 do tipo $p \vee q$. Para construir o valor de retorno, chamamos a função `Or.elim`, que tem a assinatura `Or.elim` : $(a \vee b) \to (a \to c) \to (b \to c) \to c$, ou seja, ela recebe três argumentos, um do tipo $a \vee b$, um do tipo $a \to c$ e um do tipo $b \to c$ e retorna um valor do tipo c. O primeiro argumento da função `Or.elim` é o próprio parâmetro h_1 que é do tipo $p \vee q$. Nós queremos que a função retorne um valor do tipo $q \vee p$, então os outros dois argumentos de `Or.elim` precisam ser dos tipos $p \to (q \vee p)$ e $q \to (q \vee p)$, respectivamente. Para cada um desses dois argumentos, será construída uma função.

Na linha 7 da Fig. 2 construímos uma função que recebe um argumento chamado hp do tipo p. Para construir o valor de retorno chamamos a função `Or.intro_right`,

com assinatura `Or.intro_right` : $a \to b \to (a \vee b)$ Que recebe um tipo a e um valor do tipo b e retorna um valor do tipo $a \vee b$. O tipo q e o próprio parâmetro `hp` são usados como argumentos o que garante que a função recebe um valor do tipo p e retorna um valor do tipo $q \vee p$. Na linha 8, construímos uma função análoga à da linha 7, mas que recebe um argumento chamado `hq` do tipo q, e usa a função `Or.intro_left` para construir um valor do tipo $q \vee p$. Deste modo, concluímos a construção do primeiro argumento da função `Iff.intro`. O segundo argumento da função `Iff.intro` é construído de modo análogo.

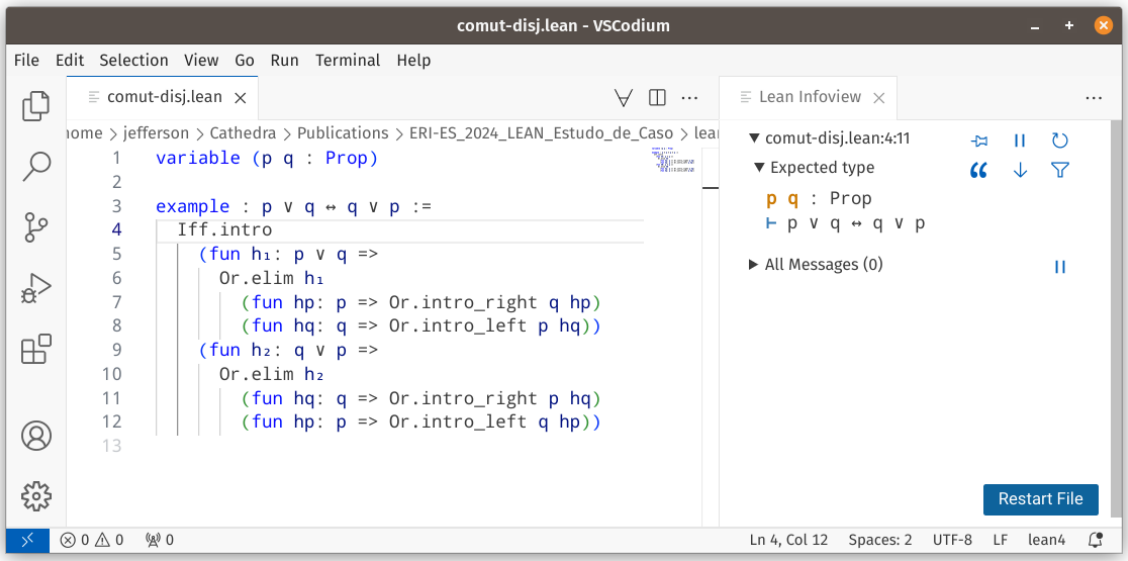

Figura 3. Ambiente de desenvolvimento LEAN no Visual Studio Code.

A Fig. 3 mostra o ambiente de desenvolvimento LEAN no editor *VSCodium* [Plainer 2021] configurado com a extensão pra LEAN 4[2]. Na parte esquerda da aba esquerda temos o programa LEAN sendo editado. Note que a linha 4 (função `Iff.intro`) está selecionada. Simultaneamente, na aba direita o ambiente de desenvolvimento mostra o tipo da expressão selecionada, que neste caso é indicado pela a mensagem "*Expected type* $\ll p\ q : Prop \vdash p \vee q \leftrightarrow q \vee p \gg$". Também pode-se ver que não há mensagens de erro.

5. Considerações Finais

Este estudo mostrou que o provador de teoremas LEAN pode enriquecer o ensino de lógica matemática ao oferecer um ambiente de aprendizagem interativo que aumenta a compreensão e o engajamento dos alunos. O LEAN, fundamentado na teoria de tipos dependentes, não apenas facilita a formalização de conceitos complexos, mas também permite que os alunos explorem esses conceitos de forma prática e intuitiva, promovendo uma transição eficaz dos métodos tradicionais para abordagens mais dinâmicas [Avigad et al. 2020].

Embora a greve tenha impactado a avaliação quantitativa do impacto do LEAN no aprendizado dos alunos, este estudo ofereceu uma visão valiosa sobre como o uso da ferramenta pode facilitar a compreensão de conceitos lógicos. A documentação detalhada do

[2] `https://github.com/leanprover/vscode-lean4`

processo fornece uma base sólida para futuras avaliações empíricas, que serão realizadas assim que as atividades forem normalizadas.

Futuros trabalhos devem incluir análises quantitativas e qualitativas para avaliar o impacto do LEAN no desempenho acadêmico dos alunos, coletando *feedback* por meio de questionários e comparando o desempenho antes e depois da introdução da ferramenta. Além disso, expandir o uso do LEAN para outras áreas da lógica e da computação pode abrir novas oportunidades de pesquisa e desenvolvimento educacional, destacando a importância de ferramentas computacionais no ensino moderno [Feofiloff and Szwarcfiter 2009].

A integração do LEAN no currículo de Sistemas de Informação não apenas pode melhorar a aprendizagem de lógica matemática, mas também pode preparar melhor os alunos para futuros desafios em suas carreiras, onde habilidades de raciocínio lógico e verificação formal são cada vez mais valorizadas no mercado de trabalho [Cherniavsky and Heeren 2017]. A aplicação contínua e expandida dessa abordagem pode servir como um modelo para outras instituições que buscam inovar no ensino de disciplinas complexas por meio de tecnologias interativas.

Referências

Avigad, J., De Moura, L., and Kong, S. (2015). Theorem proving in lean. *Release*, 3(0):1–4.

Avigad, J., Hölzl, J., and Serafin, L. (2017). A formally verified proof of the central limit theorem. *Journal of Automated Reasoning*, 59(4):389–423.

Avigad, J., Lewis, R. Y., and van Doorn, F. (2020). *Logic and Proof*. Lecture notes for the course Logic and Mathematical Inquiry.

Avigad, J. and Massot, P. (2024). Mathematics in lean. Technical Report Release 0.1, Lean Community.

Charles Fadel, Maya Bialik, e. B. T. (2015). *Four-Dimensional Education: The Competencies Learners Need to Succeed*. Solution Tree Press.

Cherniavsky, J. and Heeren, C. (2017). *Self-Adaptive Systems for Machine Intelligence*. IGI Global.

De Moura, L., Kong, S., Avigad, J., Van Doorn, F., and von Raumer, J. (2015). The LEAN theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pages 378–388. Springer.

Feofiloff, P. and Szwarcfiter, J. L. (2009). *Algoritmos e Estruturas de Dados*. LTC.

Gersting, J. L. (2019). *Fundamentos Matemáticos Para a Ciência Da Computação: Matemática Discreta e Suas Aplicações*. LTC, Rio de Janeiro, RJ, 7ª edição edition.

Huth, M. and Ryan, M. (2007). *Lógica Em Ciência Da Computação: Modelagem e Argumentação Sobre Sistemas*. LTC - Livros Tecnicos E Cientificos Editora Lda.

Plainer, M. (2021). Practical study of visual studio code. techreport, Technical University of Munich.