Ensinando Teoria da Computação com Jupyter Notebook

Davi R. Vasconcelos[1] **, Paulo T. Guerra**[1]

[1]Universidade Federal do Ceará (UFC) – Campus de Quixadá
Av. José de Freitas Queiroz, 5003 – Cedro – 63902-580 – Quixadá – CE – Brazil

{daviromero,paulodetarso}@ufc.br

Abstract. *Theory of Computation is part of the core program of several Computer Science graduation courses. This work aims to present interactive documents, Jupyter Notebooks, which were developed and shared via Google Colab for teaching the subject of Theory of Computation. We present the context in which they were used, as well as the result of the evaluation of their use as an auxiliary tool in the teaching-learning process in the Theory of Computation courses offered for the undergraduate and master's programs of Computer Science courses at the Federal University of Ceará - Campus Quixadá.*

Resumo. *Teoria da Computação faz parte do conjunto de conteúdos curriculares de formação básica de cursos de Computação. Este trabalho tem como objetivo apresentar os documentos interativos, Jupyter Notebooks, que foram desenvolvidos e compartilhados pelo Google Colab para o ensino da disciplina de Teoria da Computação. Apresentamos o contexto em que eles foram utilizados, bem como o resultado da avaliação de seu uso como ferramenta auxiliar no processo de ensino-aprendizagem nas disciplinas de Teoria da Computação ofertadas para os cursos de graduação e de mestrado em Ciência da Computação do Campus de Quixadá da Universidade Federal do Ceará.*

1. Introdução

A Teoria da Computação é um área da ciência da computação e matemática que busca determinar quais problemas podem ser computados em um dado modelo de computação abstrata [Diverio and Menezes 2009, Sipser and de Queiroz 2007, Hopcroft et al. 2007, Carnielli and Epstein 2005, Martin 2010]. A computação pode ser definida como a solução de um problema ou, formalmente, o cálculo de uma função por meio de um algoritmo.

Teoria da Computação faz parte do conjunto de conteúdos curriculares de formação básica e tecnológica previstos nas Diretrizes Curriculares Nacionais dos cursos de bacharelado e de licenciatura em Computação, como os bacharelados em Ciência da Computação, Sistemas de Informação e em Engenharia de Computação e a licenciatura em Computação [BRASIL 2012]. Usualmente ofertada com um curso introdutório, a disciplina de Teoria da Computação traz os conceitos fundamentais da Computação que permitem o raciocínio sobre modelos de computação abstratos para investigar quais problemas tem solução e quais não tem.

Em geral, os conteúdos da disciplina de Teoria da Computação são densos e teóricos, sendo fundamental que os estudantes realizem diversos exercícios para melhor

assimilar os conteúdos. Nesse contexto, como forma de auxiliar o aprendizado e autonomia dos alunos, propomos a organização do material didático da disciplina de Teoria da Computação em documentos interativos que permitam, além da exposição de conteúdo, a interação direta no mesmo ambiente com exercícios práticos de fixação dos conteúdos.

Existem diversas ferramentas para auxiliar o ensino de Linguagens Formais e Autômatos. *Language Emulator* [Vieira et al. 2003] é uma ferramenta para manipulação de formas mais usuais de descrição de linguagens regulares. *LFA Virtual* [Oliveira and Uchôa 2009] é uma aplicação desenvolvida para o aprendizado de autômatos finitos e de linguagens formais, como linguagens regulares e linguagens livres-do-contexto. *FAdo* [Almeida et al. 2009] é um projeto de código aberto que permite manipulação simbólica de linguagens formais por meio de programação em alto nível, fácil prototipagem e portabilidade.

Entre as ferramenta que permitem que também abordam os conceitos de Máquinas de Turing, destaca-se o JFLAP (*Java Formal Languages and Automata Package*)[1] [Rodger and Finley 2006], ferramenta de código aberto, desenvolvida em Java e amplamente utilizada no estudo de Teoria da Computação devido aos seus diversos recursos. Contudo essas ferramentas não apresentam uma estrutura didática que permita a mescle a apresentação de conceitos, autoestudo e práticas guiadas.

O *Executable Book Project*[2] é uma colaboração internacional entre várias universidades e projetos de código aberto para compartilhar "livros executáveis". Livros como [VanderPlas 2016], por exemplo, disponibiliza uma versão digital[3] implementada por meio de *Jupyter Notebook*, onde o leitor pode, além acompanhar o texto principal do livro, executar e alterar os exemplos fornecidos pelos autores.

O *Jupyter Notebook* é um software de código aberto que disponibiliza um ambiente online para a edição e execução de documentos interativos, chamados de *notebooks*. Os *notebooks* permitem combinar em um único documento linguagens de marcação (HTML, LaTeX, *Rich Text*, *Markdown*) e código executável (Python, R, Julia, Scala, entre outros), que podem ser visualizados e executados em um navegador *web*. Serviços como o *Colab*[4] (ou *Collaboratory*) permitem a criação e execução de *notebooks* em Python em um ambiente em nuvem, sem que o usuário precise realizar qualquer tipo de instalação local de *software* e ainda contando com a integração com sua conta do *Google* para compartilhamento de *notebooks* entre os usuários do serviço. Por conta dessas diversas características, foi a escolha realizada nesse trabalho.

Este trabalho tem como objetivo apresentar ferramentas de auxílio ao ensino-aprendizagem de Teoria da Computação, implementadas por meio de *notebooks* que foram desenvolvidos e compartilhados pelo *Colab* para o ensino da disciplina de Teoria da Computação em português. Discutimos neste trabalho o contexto em que eles foram utilizados, bem como os resultados obtidos com a avaliação de seu uso em duas disciplinas de Teoria da Computação ofertadas para os cursos de graduação e de mestrado em Ciência da Computação do Campus da Universidade Federal do Ceará em Quixadá. Essas ferra-

[1]Disponível em `https://www.jflap.org/`
[2]Disponível em `https://executablebooks.org`
[3]Disponível em `https://jakevdp.github.io/PythonDataScienceHandbook/`
[4]Disponível em `https://colab.research.google.com/`.

mentas foram especialmente cruciais para o ensino de Teoria da Computação quando das restrições a aulas presenciais em decorrência da pandemia de Covid-19.

O restante deste trabalho está organizado como segue. A Seção 2 apresenta trabalhos relacionados à abordagem proposta. A Seção 3 apresenta o referencial teórico e os conteúdos práticos que foram disponibilizados em *notebooks*. A Seção 4 aponta os resultados da avaliação da experiência do uso do material didático desenvolvido. E, por fim, na Seção 5 apresentaremos as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Em [Fouh et al. 2014], os autores propõem um arcabouço chamado OpenDSA para criação de livros textos eletrônicos para diferentes temas de ciência da computação, como algoritmos, estrutura de dados e pensamento computacional. O OpenDSA busca permitir a criação de livros-textos virtuais integrados com elementos de visualização de conceitos bem como exercícios com correção automática. O arcabouço é baseado em HTML5 e utiliza a biblioteca JSAV (JavaScript Algorithm Visualization).

O uso de livros interativos também é defendido em [Mohammed et al. 2019] como forma de contornar o fato que livros texto de disciplinas teóricas de Ciência da Computação tendem a ser pesados em prosa e em sua apresentação matemática. Baseado no OpenDSA, os autores desenvolveram um material baseado no paradigma pedagógico de Instrução Programada [Skinner 1986] para ser utilizado nos estudos de Linguagens Formais e Autômatos na Virginia Tech. O material proposto busca reduzir problemas encontrados na utilização de livros clássicos, onde os alunos por vezes não se envolvem com o material ou avançam no conteúdo sem ter entendido conceitos-chave.

Ao longo da última década, cadernos computacionais como o Jupyter Notebook tornaram-se ambientes populares de programação, principalmente em áreas como ciência de dados e aprendizado de máquina. Assim como o OpenDSA, cadernos computacionais permitem intercalar texto expositivo e elementos de interação, com o diferencial de poder também embutir código executável e gerar saídas complexas como tabelas de dados, imagens e *widgets* interativos. Outra vantagem é a existência de ambientes computacionais, como o Google Colab, que permitem a execução e compartilhamento de *notebooks* sem a necessidade de instalação local.

Em [Gopalakrishnan and Neff 2021] os autores apresentam o arcabouço Jove[5] para o ensino de Autômatos e Computabilidade através de Jupyter Notebooks. Os estudantes podem criar autômatos em linguagem *markdown* e estes então são traduzidos em diagramas e animações. Jove inclui demonstrações interativas e controles de animação para ilustrar conceitos como autômatos finitos determinísticos, autômatos finitos não-determinísticos, Máquinas de Turing e entre outros. Utilizando a linguagem Python os estudantes podem realizar testes e implementar algoritmos próprios baseado nas implementações disponíveis. Por estar disponível no *Github*, seus notebooks podem ser executados via Google Colab sem que seja necessário instalação local.

O presente trabalho segue a linha dos trabalhos aqui apresentados. A escolha pelo uso de Jupyter Notebook é baseada na facilidade de criação e execução de *notebooks* em um ambiente em nuvem, sem que o usuário precise realizar qualquer tipo de instalação

[5]Disponível em `https://github.com/ganeshutah/Jove.git`

local, facilitando assim o compartilhamento com estudantes. Ao mesmo tempo, nossa abordagem diverge por ter o foco no público-alvo nacional, produzindo material didático em português, auto-explicativas, com exercícios alinhados com livros-textos populares e com implementações que necessitam de pouco conhecimento de programação.

3. Notebooks para a Disciplina de Teoria da Computação

A disciplina de Teoria da Computação foi dividida em módulos e, para cada módulo, foi disponibilizado um *notebook*, contendo a apresentação do conteúdo teórico (texto com definições, teoremas, exemplos e videoaulas), e o conteúdo prático (exercícios) que são executados e interativos. A Figura 1 ilustra um desses *notebooks*.

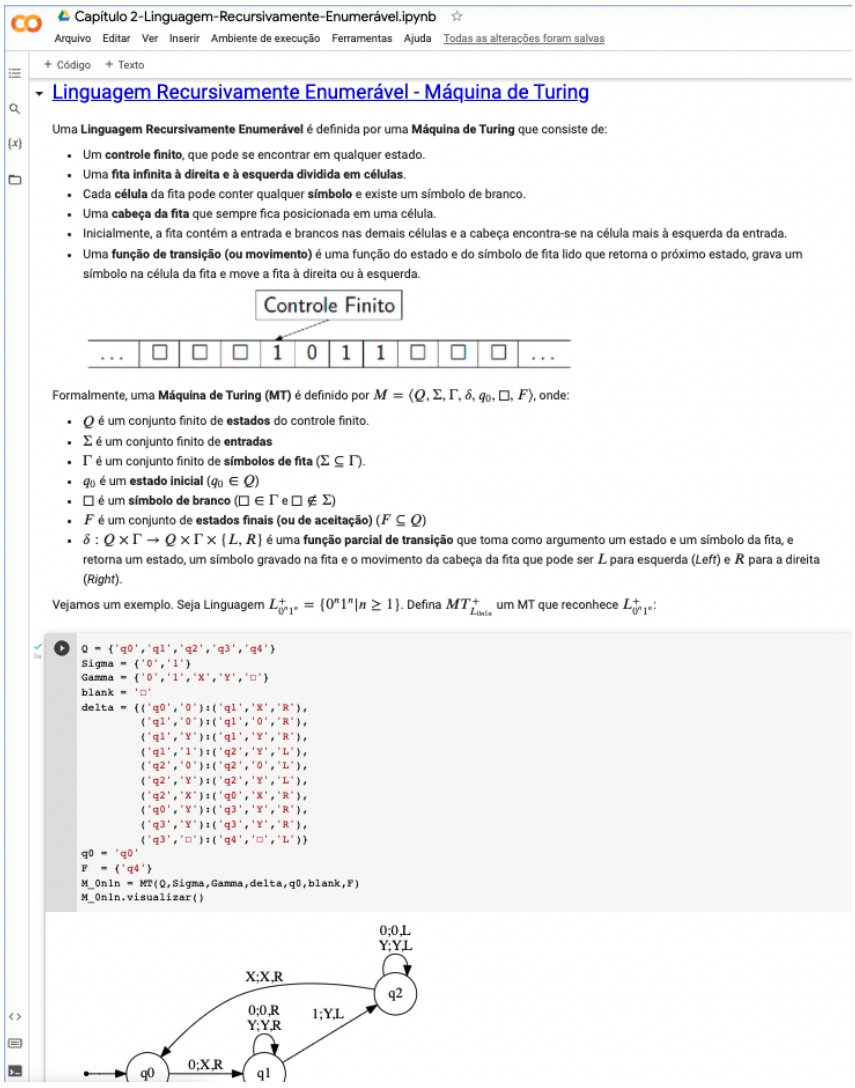

Figura 1. Exemplo de *notebook*.

O conteúdo prático foi implementado por meio de código em Python, contendo a implementação dos conceitos abordados em cada conteúdo, bem como ferramentas que auxiliem na sua visualização e interação. Em geral, os estudantes não necessitam conhecer o código implementado ou mesmo ter um conhecimento profundo da linguagem

Python, bastando acompanhar os exemplos e experimentando construir seus próprios modelos.

Os conteúdos abordados na disciplina foram organizados nos seguintes capítulos:

- **Capítulo 1 - Revisão sobre Conjuntos, Relações, Funções e Linguagens** que aborda os tópicos: definição de conjuntos, subconjuntos e conjunto potência; operações e propriedades sobre conjuntos; produto cartesiano, relação e auto-relação; fechos de relações; função, função injetora, sobrejetora e bijetora; definição de conjuntos finitos e infinitos; provas por indução sobre os naturais; o princípio da diagonalização; alfabeto, palavras e linguagens; e conversão de palavras em binário em números.
- **Capítulo 2 - As Linguagens Recursivamente Enumeráveis** que aborda os tópicos: Máquinas de Turing; Descrição Instantânea e Dedução; Linguagens Recursivamente Enumerável e Recursiva; Técnicas de Programação (várias trilhas e sub-rotinas) em Máquina de Turing; e, Extensões de Máquinas de Turing (movimento vazio, várias fitas e não-determinismo).
- **Capítulo 3 - As Funções Recursivas de Kleene** que aborda os tópicos: operações básicas; substituição composicional; função recursiva; exemplos de funções (soma, multiplicação, fatorial, antecessor, subtração própria, predicados lógicos); definição por casos; somatório e produtório; quantificação limitada existencial e universal; número de Gödel; e Minimização.
- **Capítulo 4 - O Cálculo Lambda** que aborda os tópicos: a Linguagem Lambda; substituição, redução e igualdade; representação de valores Booleanos como funções lambdas; predicados lógicas (negação, conjunção, disjunção e implicação); representando números inteiros como numerais de Church; funções sobre os números (sucessor, soma, multiplicação, exponenciação, predecessor, subtração própria, menor ou igual); recursão e ponto-fixo (função fatorial, número de Fibonacci).
- **Capítulo 5 - Indecidibilidade** que aborda os tópicos: hierarquia das linguagens, complementos de linguagens recursivamente enumerável e recursiva; codificação de palavras em binário como números; provas como strings binárias; máquinas de Turing como strings binárias; linguagens não recursivamente enumeráveis (linguagem da diagonalização); linguagem recursivamente enumerável e não recursiva; linguagem universal e máquina universal; reduções; Problema da Correspondência de Post (PCP) e indecidibilidade do PCP.

Em virtude de limitação de espaço, neste trabalho iremos abordar apenas conteúdos referentes ao Capítulos 2, mais especificamente quanto a introdução aos conceitos de máquinas de Turing. O conjunto completo de notebooks pode ser encontrado no GitHub[6].

3.1. Notebook: Introdução às Máquinas de Turing

Um ponto central do estudo de Teoria da Computação está na abordagem proposta pela tese de Church-Turing, que formaliza o conceito de algoritmo e daquilo que é computável por meio em um modelo computacional denominado de **Máquina de Turing**. Uma Máquina de Turing é um dispositivo abstrato capaz de realizar operações de leitura e

[6]Disponível em `https://github.com/daviromero/teocomp/`

escrita em uma fita de trabalho. As operações são guiadas por um mecanismo de controle que opera uma cabeça de leitura e escrita e realiza movimentos simples na fita para direita ou esquerda. A Figura 2 ilustra uma Máquina de Turing.

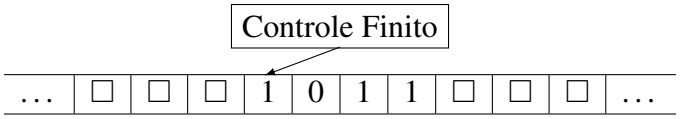

Figura 2. Ilustração de uma máquina de Turing

Em um curso de Teoria da Computação, a apresentação do conceito de Máquina de Turing passa não só pela exposição intuitiva de seu funcionamento, mas também por sua definição matemática formal, conforme apresenta a Definição 1.

Definição 1 (Máquina de Turing) *Uma **Máquina de Turing (MT)** é definido por* $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \square, F \rangle$*, onde:*

- Q *é um conjunto finito de **estados** do controle finito.*
- Σ *é um conjunto finito de **entradas***
- Γ *é um conjunto finito de **símbolos de fita** (*$\Sigma \subseteq \Gamma$*).*
- q_0 *é um **estado inicial** (*$q_0 \in Q$*)*
- $\square$ *é um **símbolo de branco** (*$\square \in \Gamma$ *e* $\square \notin \Sigma$*)*
- F *é um conjunto de **estados finais (ou de aceitação)***
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ *é uma **função parcial de transição** que toma como argumento um estado e um símbolo da fita, e retorna um estado, um símbolo gravado na fita e o movimento da cabeça da fita que pode ser* L *para esquerda (Left) e* R *para a direita (Right).*

Para fins didáticos, essas máquinas costumam ser representadas graficamente por figuras que ilustram as relações de transições entre estados descritas por sua função de transição. Uma Máquina de Turing formalmente definida como $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, \square\}, \delta, q_0, \square, \{q_4\})$, onde $\delta(q_0, 0) = \langle q_1, X, R\rangle, \delta(q_1, 0) = \langle q_1, 0, R\rangle, \delta(q_1, Y) = \langle q_1, Y, R\rangle, \delta(q_1, 1) = \langle q_2, Y, L\rangle, \delta(q_2, 0) = \langle q_2, 0, L\rangle, \delta(q_2, Y) = \langle q_2, Y, L\rangle, \delta(q_2, X) = \langle q_0, X, R\rangle, \delta(q_0, Y) = \langle q_3, Y, R\rangle, \delta(q_3, Y) = \langle q_3, Y, R\rangle, \delta(q_3, \square) = \langle q_4, \square, L\rangle$, pode ser representada graficamente como na Figura 3.

Nos notebooks que construímos, este conceitos introdutórios sobre Máquinas de Turing vistos até aqui são apresentados na porção inicial. A flexibilidade do uso de linguagens de marcação, como *Markdown*, Latex e HTML, permite que tais conteúdos possam ser apresentados com qualidade semelhante aquela presente nos livros didáticos tradicionais (ver Figura 1). Contudo, aproveitando as possibilidades dessa mídia, também incorporamos nesta porção do *notebook* videoaulas na qual estes conceitos são apresentados, permitindo que o aluno possa ter uma apresentação complementar deste conteúdo em um outro formato.

Todavia, o que de fato diferencia o uso de *notebook* das abordagens tradicionais é a possibilidade de utilizar células executáveis. Nesse sentido, implementamos por meio da linguagem Python o modelo de Máquina de Turing conforme a Definição 1. Esta implementação foi incorporada ao *notebook*, estando inicialmente oculta ao leitor, de modo que para exercitar esse conteúdo o estudante poderá apenas criar uma célula

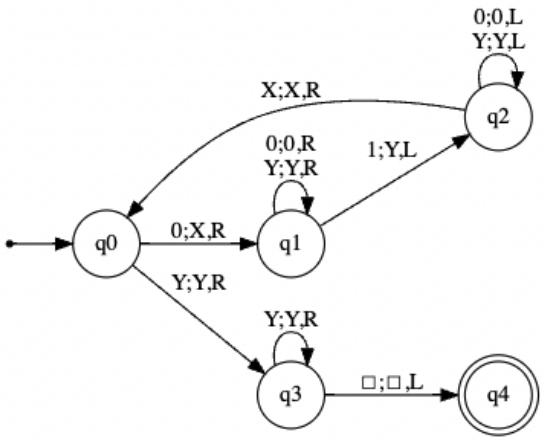

Figura 3. Representação gráfica de uma Máquina de Turing.

de código definindo a máquina de Turing semelhante aquela dada pela definição formal. A Figura 4 ilustra a definição da Máquina de Turing da Figura 3 utilizando nossa implementação.

O *notebook* desenvolvido neste trabalho também permite a definição de Máquinas de Turing por meio de XML, seguindo a sintaxe do JFLAP, permitindo assim o uso combinado com essa ferramenta. A Figura 5 apresenta a instanciação da Máquina de Turing da Figura 3 a partir do código em XML.

```
Q = {'q0','q1','q2','q3','q4'}
Sigma = {'0','1'}
Gamma = {'0','1','X','Y','□'}
blank = '□'
delta = {('q0','0'):('q1','X','R'),
         ('q1','0'):('q1','0','R'),
         ('q1','Y'):('q1','Y','R'),
         ('q1','1'):('q2','Y','L'),
         ('q2','0'):('q2','0','L'),
         ('q2','Y'):('q2','Y','L'),
         ('q2','X'):('q0','X','R'),
         ('q0','Y'):('q3','Y','R'),
         ('q3','Y'):('q3','Y','R'),
         ('q3','□'):('q4','□','L')}
q0 = 'q0'
F  = {'q4'}
M = MT(Q,Sigma,Gamma,delta,q0,blank,F)
```

Figura 4. Definição direta

```
input = '''<structure><type>turing</type><automaton>
    <state id="0" name="q0"><initial/></state>
    <state id="1" name="q1"></state>
    <state id="2" name="q2"></state>
    <state id="3" name="q3"></state>
    <state id="4" name="q4"><final/></state>
    <transition><from>2</from><to>0</to><read>X</read>
      <write>X</write><move>R</move></transition>
    <transition><from>1</from><to>1</to><read>0</read>
      <write>0</write><move>R</move></transition>
    <transition><from>1</from><to>1</to><read>Y</read>
      <write>Y</write><move>R</move></transition>
    <transition><from>3</from><to>3</to><read>Y</read>
      <write>Y</write><move>R</move></transition>
    <transition><from>3</from><to>4</to><read/>
      <write/><move>L</move></transition>
    <transition><from>0</from><to>1</to><read>0</read>
      <write>X</write><move>R</move></transition>
    <transition><from>0</from><to>3</to><read>Y</read>
      <write>Y</write><move>R</move></transition>
    <transition><from>1</from><to>2</to><read>1</read>
      <write>Y</write><move>L</move></transition>
  </automaton></structure>'''
M = MT(input_jff=input)
```

Figura 5. Definição via XML

A computação de uma Máquina de Turing é definida através das mudanças de estados, posição da cabeça de leitura e do conteúdo da fita indicados por sua função de transição a partir de uma dada informação de entrada. Essas informações compõem o que chamamos de **descrição instantânea** de uma Máquina de Turing.

Definição 2 (Descrição Instantânea) *Representamos a descrição instantânea (ID) de uma Máquina de Turing como* $X_1X_2 \ldots X_{i-1}qX_i \ldots X_n$*, onde* q *é o estado da máquina e* $X_1X_2 \ldots X_{i-1}X_i \ldots X_n$ *é a parte da fita entre o não-branco mais à esquerda e o não-*

branco mais à direita, e indica que a cabeça da fita está posicionada sobre o i-ésimo símbolo a partir da esquerda.

Um passo de computação, formalmente denominado de *dedução*, constitui a transição entre duas descrições instantâneas de acordo com o que é definido pela função de transição de uma Máquina de Turing.

Definição 3 *Uma **dedução** $\vdash$ entre duas descrições instantâneas é definida por:*

- $X_1X_2 \ldots X_{i-1}qX_iX_{i+1} \ldots X_n \vdash X_1X_2 \ldots X_{i-1}YpX_{i+1} \ldots X_n$, *se* $\delta(q, X_i) = \langle p, Y, R\rangle$,
- $X_1X_2 \ldots X_{i-1}qX_iX_{i+1} \ldots X_n \vdash X_1X_2 \ldots pX_{i-1}YX_{i+1} \ldots X_n$, *se* $\delta(q, X_i) = \langle p, Y, L\rangle$,

Denotamos por $\vdash^*$ a sequência de zero ou mais deduções tal que $I \vdash^* J$ se $I = J$, $I \vdash J$ ou $I \vdash K$ para algum $K \vdash^* J$.

Dada uma Máquina de Turing $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \Box, F\rangle$ e uma cadeia w sobre o alfabeto Σ, se $q_0w \vdash^* \alpha p\beta$ e $p \in F$, dizemos que M *aceita* w. A seguinte sequência de deduções indica que Máquina de Turing M da Figura 3 aceita a cadeia 0011.

$$q_00011 \vdash Xq_1011 \vdash X0q_111 \vdash Xq_20Y1 \vdash q_2X0Y1 \vdash Xq_00Y1 \vdash XXq_1Y1 \vdash XXYq_11$$
$$\vdash XXq_2YY \vdash Xq_2XYY \vdash XXq_0YY \vdash XXYq_3Y \vdash XXYYq_3\Box \vdash XXYq_4Y\Box$$

Para facilitar a compreensão do conceito de computação, o *notebook* contém um ambiente de simulação de modo a permitir que o usuário visualize se uma dada palavra de entrada é aceita em uma Máquina de Turing, exibindo graficamente a Máquina de Turing, o controle e o conteúdo da fita. A Figura 6 apresenta a descrição instantânea inicial com o conteúdo da fita a palavra 0011 e o controle está no q_0, representado também pelo estado q_0 em cinza na visualização da MT. O usuário pode clicar no botão "Passo-a-Passo" e realizar uma transição ou pode ainda clicar no botão "Simular" e visualizar a animação da computação. É possível configurar o intervalo entre as transições (pausa) e o número máximo de passos que a MT pode realizar antes de encerrar a computação. Após a computação da palavra 0011, o ambiente de simulação exibirá a descrição instantânea final apresentada graficamente na Figura 7.

É possível ainda ilustrar o caráter composicional das Máquinas de Turing através da implementação realizada, por meio do uso de Máquinas de Turing como sub-rotinas para facilitar a definição de outras Máquinas de Turing. A Figura 9 apresenta a definição da MT que multiplica dois números unários e que utiliza a máquina de cópia como sub-rotina, descrita na Figura 8. Para tanto, basta utilizar um identificador da sub-rotina (no exemplo `copy`) na função de transição da nova máquina de Turing e incluir a sub-rotina como parâmetro adicional na definição.

O *notebook* abordado nesta seção apresenta ainda os conceitos e definições de extensões de Máquina de Turing, como Máquinas de Turing com várias fitas e Máquinas de Turing Não-Determinística. Os códigos e mecanismos de simulação destas estão implementados de modo semelhante aqueles apresentados anteriormente para o modelo padrão, permitindo assim que o aluno possa verificar seus funcionamentos e construções que mostram a equivalência entre estes modelos.

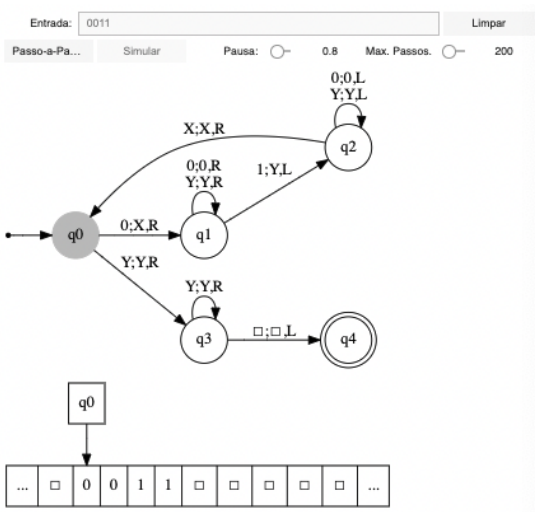

Figura 6. Descrição Instantânea inicial

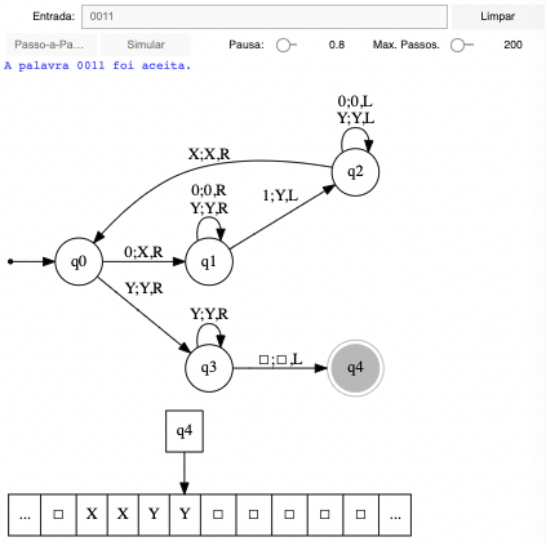

Figura 7. Descrição Instantânea final

```
Q = {'q0','q1','q2','q3','q4'}
Sigma = {'0','1'}
Gamma = {'0','1','X','Y','□'}
q0 = 'q0'
F = set()
blank = '□'
delta ={('q0','0'):('q1','X','R'),
        ('q1','0'):('q1','0','R'),
        ('q1','1'):('q1','1','R'),
        ('q1','□'):('q2','0','L'),
        ('q2','0'):('q2','0','L'),
        ('q2','1'):('q2','1','L'),
        ('q2','X'):('q0','X','R'),
        ('q0','1'):('q3','1','L'),
        ('q3','X'):('q3','0','L'),
        ('q3','1'):('q4','1','R')
}
M_copy = MT(Q,Sigma,Gamma,delta,q0,blank,F)
```

Figura 8. Máquina de Turing `Copy`

```
Q = {'q0','q1','copy','q3','q4','q5','q6','q7','q8'}
Sigma = {'0','1'}
Gamma = {'0','1','X','Y','□'}
q0 = 'q0'
blank = '□'
F = {'q7'}
delta = { ('q0','0'):('q1','□','R'),
          ('q1','0'):('q1','0','R'),
          ('q1','1'):('copy','1','R'),
          ('copy','0'):('q3','0','L'),
          ('q3','1'):('q4','1','L'),
          ('q4','0'):('q8','0','L'),
          ('q4','□'):('q5','□','R'),
          ('q5','1'):('q6','□','R'),
          ('q6','0'):('q6','□','R'),
          ('q6','1'):('q7','□','R'),
          ('q8','0'):('q8','0','L'),
          ('q8','□'):('q0','□','R')
        }
MTs = {"copy":M_copy}
M_mult = MT(Q,Sigma,Gamma,delta,q0,blank,F,MTs)
```

Figura 9. Máquina de Turing `Mult`

4. Avaliação da experiência do estudante no uso dos *notebooks*

A avaliação da experiência de uso dos *notebooks* foi realizada por meio de formulário disponibilizado na ferramenta *Google Forms*, com alunos das turmas de Teoria da Computação da graduação e do mestrado em Ciência da Computação no semestre 2021.2 e 2022.2 do Campus da UFC em Quixadá. Em 2021.1, a turma do mestrado contava com 7 alunos matriculados dos quais 5 responderam ao instrumento, e a turma da graduação contava com 38 alunos, dos quais 15 responderam, totalizando 45 alunos matriculados e 20 respondentes. Em 2022.2, a turma do mestrado contava com 4 alunos matriculados dos quais 4 responderam ao instrumento, e a turma da graduação contava com 31 alunos, dos quais 22 responderam, totalizando 35 alunos matriculados e 26 respondentes.

O questionário incluiu as seguintes perguntas:

1. "*Caso tenha utilizado o Colab, em quais momentos a ferramenta serviu como suporte às suas atividades?*"
2. "*Com que frequência você utilizou o Colab no período em que o conteúdo de Teoria da Computação estava sendo ministrado?*"

3. "*Você considera que o Colab ..... ajudou a exercitar o conteúdo de Teoria da Computação?*", sendo realizada uma pergunta para cada assunto.

A seguir apresentamos os resultados das perguntas que constavam no instrumento.

- Da pergunta "*Caso tenha utilizado o Colab, em quais momentos a ferramenta serviu como suporte às suas atividades?*", 100% dos alunos afirmaram que usaram para resolver exercícios das atividades assíncronas propostas, 95% para resolver exercícios das atividades síncronas propostas e 45% para resolver outros exercícios, além dos propostos.
- Da pergunta "*Com que frequência você utilizou o Colab no período em que o conteúdo de Teoria da Computação estava sendo ministrado?*", 39% responderam que usaram três ou mais vezes por semana, 39% duas vezes por semana, 15% uma vez por semana, e 7% não usaram de forma regular.
- Quanto ao auxílio fornecido pelos *notebooks*, 100% afirmaram que o *notebook* sobre o conteúdo Indecidibilidade ajudou a exercitar o conteúdo de Teoria da Computação, 94% afirmaram que o *notebook* sobre o conteúdo Máquinas de Turing ajudou a exercitar o conteúdo de Teoria da Computação, e 95% afirmaram que os demais notebooks ajudaram a exercitar os seus respectivos conteúdos.

De uma maneira geral, 69% e 17% dos alunos avaliaram, respectivamente, como excelente e muito boa a utilização dos *notebooks* na disciplina Teoria da Computação.

5. Conclusão e Trabalhos Futuros

Este trabalho apresentou os documentos interativos, *Jupyter Notebooks*, que foram desenvolvidos e compartilhados pelo *Google Colab*, para auxiliar o ensino da disciplina de Teoria da Computação. Os notebooks deste trabalho mesclam elementos textuais, visuais e interativos, nos quais os estudantes podem utilizar de modo complementar aos materiais didáticos tradicionais disponíveis na disciplina. Além disso, os notebooks contêm exercícios de fixação do conteúdo com correção automática, sendo ponto importante no processo autônomo de aprendizado. Como vantagem adicional, os estudantes podem experimentar modificações nos notebooks, incluindo implementações próprias, testando novas situações e consolidando a compreensão dos conceitos.

Por limitações de espaço, neste trabalho foi detalhado apenas o notebook com o conteúdo introdutório sobre Máquinas de Turing. O conjunto completo de notebooks está disponível no GitHub[7].

Os *notebooks* foram utilizados nas disciplinas de Teoria da Computação da graduação e do mestrado em Ciência da Computação do campus de Quixadá da Universidade Federal do Ceará, durante os semestres letivo de 2021.2 e 2022.2. Aproximadamente 57% dos alunos matriculados avaliaram os conteúdos e a grande maioria mencionou: utilizar os *notebooks* duas vezes ou mais vezes por semana; para resolver exercícios das atividades assíncronas e síncronas propostas; que os documentos ajudaram a exercitar os conteúdos abordados; por fim, que avaliaram o material como excelente ou muito bom.

Como trabalho futuro, pretendemos aprimorar o material, incluir mais exercícios nos *notebooks*, incluir novos conteúdos e formatar os conteúdos no modelo de um livro "executável".

[7]Disponível em `https://github.com/daviromero/teocomp/`

Referências

Almeida, A., Almeida, M., Alves, J., Moreira, N., and Reis, R. (2009). Fado and guitar: tools for automata manipulation and visualization. In *International Conference on Implementation and Application of Automata*, pages 65–74. Springer.

BRASIL, Ministério da Educação, C. N. E. (2012). Parecer CNE/CES nº 136/2012. *Diário Oficial da União*.

Carnielli, W. and Epstein, R. (2005). *Computabilidade, funções computáveis, lógica e os fundamentos da matemática*. Editora da UNESP.

Diverio, T. and Menezes, P. (2009). *Teoria da Computação - 3.Ed. - UFRGS: Máquinas Universais e Computabilidade*. Grupo A - Bookman.

Fouh, E., Karavirta, V., Breakiron, D. A., Hamouda, S., Hall, S., Naps, T. L., and Shaffer, C. A. (2014). Design and architecture of an interactive etextbook – the opendsa system. *Science of Computer Programming*, 88:22–40. Software Development Concerns in the e-Learning Domain.

Gopalakrishnan, G. and Neff, R. (2021). Automata and computability education via jove. SIGCSE '21, page 1374, New York, NY, USA. Association for Computing Machinery.

Hopcroft, J., Motwani, R., and Ullman, J. (2007). *Introduction to Automata Theory, Languages, and Computation*. Introduction to Automata Theory, Languages, and Computation. Pearson/Addison Wesley.

Martin, J. (2010). *Introduction to Languages and the Theory of Computation*. McGraw-Hill Education.

Mohammed, M., Rodger, S. H., and Shaffer, C. A. (2019). Using programmed instruction to help students engage with etextbook content. In Sosnovsky, S. A., Brusilovsky, P., Baraniuk, R. G., Agrawal, R., and Lan, A. S., editors, *Proceedings of the First Workshop on Intelligent Textbooks co-located with 20th International Conference on Artificial Intelligence in Education (AIED 2019), Chicago, IL, USA, June 25, 2019*, volume 2384 of *CEUR Workshop Proceedings*, pages 135–145. CEUR-WS.org.

Oliveira, P. V. N. and Uchôa, J. Q. (2009). *LFA Virtual – Uma ferramenta de ensino para a disciplina de linguagens formais e autômatos*. PhD thesis, Universidade Federal de Lavras.

Rodger, S. H. and Finley, T. W. (2006). *JFLAP: an interactive formal languages and automata package*. Jones & Bartlett Learning.

Sipser, M. and de Queiroz, R. (2007). *Introdução à teoria da computação*. Thomson Learning.

Skinner, B. (1986). Programmed instruction revisited. *Phi Delta Kappan*, 68(2):103–10.

VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data*. O'Reilly Media, Inc.

Vieira, L. F. M., Vieira, M. A. M., and Vieira, N. J. (2003). Language emulator, uma ferramenta de auxílio no ensino de teoria da computação. In *XIII Workshop sobre Educação em Computação–XXV Congresso da Sociedade Brasileira de Computação*.