Explorando a teoria dos autômatos com uma atividade educacional usando a gramática gráfica para o ensino fundamental e médio

Julia Veiga da Silva[1], Braz Araujo da Silva Junior[1], Simone' Andre' da Costa Cavalheiro[1], Luciana Foss[1]

[1]Centro de Desenvolvimento Tecnológico - Universidade Federal de Pelotas (UFPel) CEP 96.010-610 - Pelotas - RS - Brasil

{jvsilva,badsjunior,simone.costa,lfoss}@inf.ufpel.edu.br

Resumo. *Este artigo propõe uma atividade educacional para o ensino fundamental e médio, alinhada com a Base Nacional Comum Curricular, que explora a Teoria dos Autômatos usando a Gramática de Grafos. Embora várias áreas da computação estejam cada vez mais integradas ao contexto educacional, a área teórica ainda é negligenciada. Devido à escassez de abordagens diretas no ensino fundamental e médio, este projeto busca preencher essa lacuna. A atividade proposta não apenas desenvolve uma habilidade específica da Base Nacional Comum Curricular, mas também aprimora indiretamente as habilidades de Pensamento Computacional.*

1. Introdução

Impulsionados pelo impacto da Ciência da Computação (CS) na vida cotidiana, surgiram iniciativas e esforços para tornar a educação nessa área disponível para todos. Influenciada pelo importante trabalho de Wing sobre Pensamento Computacional (CT) [Wing 2006], a percepção do ensino de CS passou a entender a computação não apenas como codificação ou programação, mas como a capacidade de resolver problemas. Os alunos de hoje vivem em um mundo fortemente influenciado pela computação, e muitos trabalharão em áreas que envolvem ou são influenciadas por ela. Portanto, não é mais suficiente esperar até que os alunos estejam na faculdade para introduzir conceitos relacionados à computação. Eles devem começar a trabalhar com resolução de problemas algorítmicos e métodos e ferramentas computacionais no ensino fundamental e médio [Barr e Stephenson 2011]. Trabalhos recentes mostram a aplicação de diferentes práticas aliadas à CT no ensino fundamental e médio, como computação desconectada [Chen et al. 2023], robótica [Gonzalez et al. 2024, Ching and Hsu 2023] e até mesmo inteligência artificial [Lee and Kwon 2024, Yim and Su 2024, Sanusi et al. 2023].

Em 2022, o Conselho Nacional de Educação do Brasil aprovou as Normas de Computação no ensino fundamental e médio, complementando a Base Nacional Comum Curricular (BNCC)[1] [Brasil 2022]. A CT é um dos três eixos principais descritos neste apêndice. Apesar do recente crescimento da TC no Brasil [Farias et al. 2023], desafios como o desenvolvimento profissional, a disponibilidade de materiais didáticos e a aplicação de metodologias eficazes devem ser enfrentados. Além disso, há necessidade de criar atividades que promovam o desenvolvimento dos objetos de conhecimento delineados pela BNCC para cada nível do ensino fundamental e médio.
12. Esclarecendo, os objetos de conhecimento são os assuntos abordados em cada componente curricular, representando os meios para o desenvolvimento de habilidades.

[1] Documento que define os conhecimentos, as habilidades e as competências essenciais que todos os alunos do ensino fundamental e médio no Brasil devem desenvolver ao longo de sua escolaridade.

DOI: 10.5753/sbie.2024.242502

Embora a educação em computação tenha sido reacendida com o advento da TC e tenha se expandido para o ensino fundamental e médio, a Ciência da Computação Teórica (TCS) recebe relativamente menos atenção nesse contexto. Uma revisão sistemática da literatura sobre CTS no ensino fundamental e médio revelou que tópicos como expressões regulares, linguagens formais e teoria de autômatos são muito menos populares [Silva Junior et al. 2021]. Os trabalhos abrangeram várias abordagens, incluindo ferramentas digitais, aulas tradicionais e atividades práticas não relacionadas a computadores, enfatizando a resolução de problemas como elemento central de suas propostas educacionais. No entanto, a TCS é subestimada em comparação com abordagens mais populares.

Com o objetivo de disponibilizar recursos nessa área negligenciada, este artigo apresenta uma atividade educacional baseada na teoria dos autômatos, um subtópico da TCS, para desenvolver uma habilidade descrita no apêndice de computação da BNCC para o ensino fundamental e médio. A atividade foi desenvolvida usando o GrameStation, um mecanismo de jogo baseado na Graph Grammar (GG), uma linguagem formal e visual para descrever sistemas e verificar propriedades. O restante deste artigo está organizado da seguinte forma: A Seção 2 apresenta alguns trabalhos relacionados e diferencia nossa proposta das existentes; a Seção 3 aprofunda a fundamentação teórica, explorando conceitos relativos à teoria de autômatos e à GG, além de apresentar o mecanismo de jogo GrameStation; a Seção 4 introduz nossa abordagem, detalhando a atividade no GrameStation; a Seção 5 apresenta uma discussão sobre a proposta; e a Seção 6 conclui o artigo, apresentando as observações finais e delineando possíveis caminhos para pesquisas futuras.

2. Trabalho relacionado

Embora não sejam tão populares quanto as linguagens de programação visual, alguns trabalhos propõem a introdução de autômatos na educação. Por exemplo, um jogo de quebra-cabeça de autômatos foi proposto para introduzir a teoria dos autômatos usando gamificação para alunos do ensino fundamental e médio (9 a 12 anos de idade) de 36 escolas de ensino fundamental no Japão [Isayama et al. 2016]. Em sua pesquisa, os autores se concentraram apenas no conceito de Autômato Finito Determinístico (DFA). O jogo tem várias etapas, divididas em *perguntas de rotulagem*, em que os alunos devem definir as transições de um determinado DFA para reconhecer as entradas fornecidas, e *perguntas de reconhecimento*, em que os alunos devem identificar se um determinado DFA reconhece ou não uma determinada entrada.

Um total de 90 crianças jogaram o jogo; quinze eram alunos da 4ª série, quarenta e um da 5ª série e trinta e quatro da 6ª série. Suas ações foram registradas em logs, que consistiam em informações sobre quando e quais botões foram selecionados e os valores definidos para esses botões nesses momentos. Uma análise dos registros mostra que, ao combinar os dados de rotulagem e reconhecimento, aproximadamente 60% das crianças tiveram taxas de respostas corretas de 70% ou mais das perguntas. Assim, muitas delas compreenderam os conceitos de autômatos suficientemente bem para concluir as fases do jogo. Entretanto, a análise também mostra que, embora algumas crianças do estudo (não muitas) pudessem compreender os conceitos apresentados, a maioria delas não os compreendia totalmente, pois 80% ou mais não conseguiram descobrir as características dos autômatos nas perguntas de reconhecimento.

No contexto brasileiro, não foram encontradas atividades que trabalhem a teoria dos autômatos no ensino fundamental e médio, portanto, apresentaremos alguns exemplos de atividades que trabalham o tema com alunos de graduação. Alguns trabalhos abordam o escopo deste trabalho [Carvalho et al. 2021, Tomizawa e Junior 2021, Vieira e Sarinho 2019, Silva et al. 2010, Leite et al. 2014], e a seguir detalhamos aqueles que abordam com mais profundidade os conceitos considerados neste trabalho.

"Máquina de Senhas", em português "Password Machine" [Vieira e Sarinho 2019], é um jogo baseado na readaptação de aspectos de jogabilidade do jogo Mastermind. Ele segue o seguinte formato: um exemplo de autômato é apresentado ao jogador, e uma sequência de símbolos aceitos pelo autômato é gerada. Essa sequência é invisível para o jogador. O jogador deve então descobrir a sequência oculta do autômato dado. Após cada movimento, o jogador recebe informações sobre a tentativa atual, incluindo se há símbolos em comum entre a sequência jogada e a sequência oculta, se algum símbolo está na posição correta e se o autômato aceita a sequência sugerida. O jogo apresenta um nível crescente de dificuldade nos estágios em relação à complexidade dos desafios, aumentando o número de caracteres na sequência a ser descoberta e o número de símbolos no alfabeto do autômato.

Outro jogo, denominado "Automata Defense 2.0" [Silva et al. 2010], é um jogo educativo projetado como complemento pedagógico do curso de Linguagens Formais e Autômatos. A versão 1.0 do jogo se concentra apenas no conceito de DFA, enquanto a versão 2.0 expande o conteúdo para incluir também tópicos sobre Autômatos Finitos Não Determinísticos (NFA) e Autômatos Determinísticos Pushdown (PDA). O jogo consiste em um jogo de estratégia de defesa de torres, desafiando os jogadores a projetar autômatos para serem bem-sucedidos. O jogo apresenta diversos tipos de criaturas, e os jogadores devem eliminar os monstros para não perder a pontuação. Para isso, eles podem criar torres com autômatos para atacar criaturas que tenham palavras reconhecidas pelos autômatos. O jogo também exige raciocínio estratégico dos jogadores, pois cada estado ou transição adicionada incorre em um custo a ser pago com dinheiro virtual. A pesquisa envolveu um teste de usabilidade e uma avaliação preliminar de sua eficácia pedagógica. Um total de 26 alunos concluiu todas as etapas da pesquisa. Os testes pré e pós-sessão abrangeram os tópicos de DFA, NFA e questões teóricas gerais. O jogo foi considerado útil como complemento pedagógico, mas foram identificadas áreas de aprimoramento na interface, como o fornecimento de regras dentro do jogo e maior diferenciação entre os personagens da atividade.

Por fim, o jogo "Chomsky's Mountain" [Leite et al. 2014] é baseado na Hierarquia de Chomsky, que classifica as linguagens formais em quatro níveis. O jogo consiste em diferentes ambientes ou estágios, cada um representando um nível da hierarquia. O jogador enfrenta problemas relacionados às linguagens de cada nível e deve criar modelos formais, como autômatos, expressões regulares ou gramáticas, para representar ou reconhecer essas linguagens. À medida que o jogador resolve os problemas, novos estágios são desbloqueados, tornando-se cada vez desafiadores. O objetivo é chegar ao topo da montanha, o que representa a solução de todos os problemas em todos os níveis da hierarquia. O estudo envolveu a aplicação de um questionário eletrônico aos alunos após a interação com o jogo. O teste incluiu 39 alunos voluntários, dos quais 19 estavam matriculados no curso de Teoria da Computação e 10 já haviam concluído o curso. Os resultados mostraram que a maioria dos alunos avaliou positivamente sua experiência com o jogo (41% deram nota 10, 26% deram nota 9 e 18% deram nota 8), considerando-o útil para o aprendizado da matéria. Eles também relataram que o jogo os ajudou a entender melhor os conceitos e a resolver problemas relacionados à Teoria da Computação. Com relação às dificuldades, 38% dos alunos enfrentaram algum tipo de dificuldade durante o jogo, principalmente relacionada ao acesso pela primeira vez e à interpretação de algumas perguntas. No entanto, a maioria dos alunos disse que preferiria resolver problemas de Teoria da Computação por meio do jogo, destacando a importância do feedback imediato fornecido pelo jogo.

Em comparação com trabalhos relacionados, a atividade apresentada neste artigo se destaca por sua integração de autômatos com uma linguagem formal, exigindo que os alunos manipulem um GG explicitamente. Essa atividade não apenas desenvolve uma habilidade de BNCC, mas também contribui indiretamente para o desenvolvimento de habilidades de CT. A definição e a execução de um autômato por meio de um GG promovem várias habilidades de CT [Silva Junior et al. 2019], incluindo abstração, em que os alunos devem entender um autômato que sintetiza o comportamento de um robô e manipular um GG que não define um algoritmo como uma sequência de etapas, mas sim como um conjunto de ações (sem uma ordem predefinida) que podem ser executadas com base no contexto atual do sistema (o estado do autômato e a célula atual da fita); representação de dados, em que os alunos devem interpretar a função de transição, a fita e o robô por meio de vértices e arestas (gráficos); coleta de dados, em que os alunos devem definir a sequência de informações (criação da fita) necessária para que o robô cumpra sua tarefa; análise, em que os alunos devem avaliar a fita de entrada e identificar o estado final do robô; simulação, em que os alunos devem executar regras gramaticais que emulam o comportamento do robô em diferentes situações; e reconhecimento de padrões, em que os alunos devem definir a correspondência para aplicar as regras de GG procurando o lado esquerdo da regra a ser aplicada no gráfico que representa o estado do sistema.

3. Contexto teórico

Esta seção apresenta o contexto teórico no qual nosso estudo está situado. Ela discute as definições de teoria de autômatos (enfatizando DFA) e GG, além de apresentar a ferramenta usada para criar a atividade.

3.1. Teoria dos autômatos

Um autômato é um modelo abstrato de um sistema que pode seguir um conjunto específico de instruções para executar uma determinada tarefa. Ele pode ser considerado uma máquina com um conjunto finito de estados que pode ler símbolos de entrada, fazer a transição entre estados com base nos símbolos lidos e, em alguns casos, produzir uma saída. Os autômatos são categorizados em diferentes tipos, como autômatos finitos ou autômatos pushdown. No entanto, este trabalho se concentra no DFA, uma subcategoria de autômatos finitos. Formalmente, um DFA é definido da seguinte forma [Mogensen 2024].

Um DFA é uma tupla $M = (Q, \Sigma, \delta, q0, F)$, em que: Q é um conjunto finito de estados; Σ é um conjunto finito de símbolos de entrada, chamado de alfabeto de entrada; $\delta : Q \times \Sigma \rightarrow Q$ é a função de transição, que mapeia um estado e um símbolo de entrada para um novo estado, mas não necessariamente definido para cada combinação possível de estado e símbolo de entrada; $q0 \in Q$ é o estado inicial; $F \subseteq Q$ é o conjunto de estados de aceitação (ou final).

Assim, o autômato lê símbolos de um alfabeto de entrada e faz transições entre estados com base nesses símbolos e em sua função de transição, δ. A função de transição, δ, mapeia um estado atual e um símbolo de entrada para um novo estado. Esse processo de transição ocorre sequencialmente para cada símbolo na entrada, começando com o estado inicial, $q0$. Depois de ler toda a cadeia de entrada, se o DFA estiver aceitando o estado (pertencente ao conjunto F), a cadeia de entrada será aceita pelo DFA, indicando que a entrada faz parte da linguagem reconhecida pelo DFA. Caso contrário, se o DFA estiver em um estado de não aceitação ou não puder terminar de ler a string de entrada, a string de entrada não será reconhecida pelo DFA.

3.2. Gramática gráfica

Um GG descreve um sistema modelando seus estados como gráficos (compostos de vértices e bordas) e eventos que podem alterar seu estado atual como um conjunto de regras de transformação de gráficos

[Ehrig et al. 1997]. Um GG deve definir como seu gráfico de estado começa, o que é chamado de **gráfico inicial**. Além disso, um GG pode diferenciar e restringir seus elementos, declarando-os em um **gráfico de tipos**. Por exemplo, a Figura 1 ilustra os gráficos de tipo (esquerda) e inicial (direita) do jogo Pac-Man como um GG. O gráfico de tipo define os elementos que compõem o jogo, enquanto o gráfico inicial mostra um Pac-Man, um fantasma e frutas em uma grade 3x4 de lugares, além de um contador (triângulo rosa) relacionado ao Pac-Man para contar quantas frutas foram comidas. Em particular, o jogo Pac-Man tem quatro regras (Figura 2): *PacMove*, *GhostMove*, *PacEat* e *GhostKill*, cada uma representada por um par de gráficos ligados por um homomorfismo de gráfico.

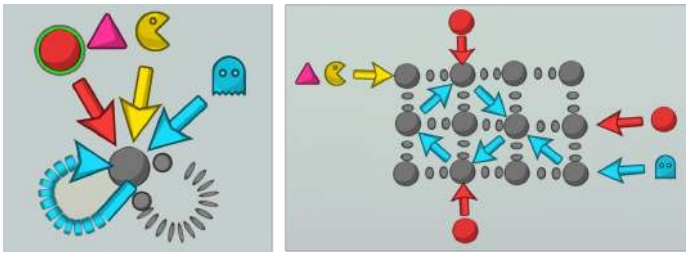

Figura 1. Gráfico de tipo (esquerda) e gráfico inicial (direita) do jogo Pac-Man como gramática gráfica

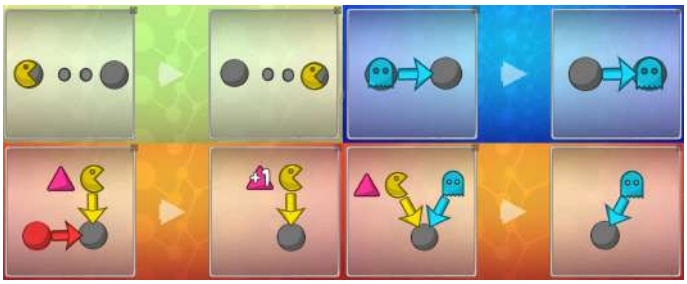

Figura 2. *PacMove* (em cima, à esquerda), *GhostMove* (em cima, à direita), *PacEat* (embaixo, à esquerda) e *GhostKill* (embaixo, à direita)

O par de gráficos que representa as regras é o lado esquerdo (LHS), que expressa uma condição para aplicar a regra, e o lado direito (RHS), que expressa a consequência da aplicação da regra. No *PacMove* (Figura 2), por exemplo, o LHS define a condição de ter um Pac-Man em um local que tenha um caminho para outro, e o RHS define a consequência de remover o Pac-Man do local inicial e colocá-lo em outro. Essa representação também implica mapeamentos elemento a elemento entre os gráficos (morfismos), portanto, para cada elemento em um gráfico, é preciso dizer qual elemento (se houver) no outro gráfico corresponde a ele. Se um elemento for mapeado com êxito, isso significa que a regra o preserva, como o Pac-Man do *PacMove* (o Pac-Man é preservado e seu mapeamento para o local é excluído e criado). Se um elemento não for mapeado e estiver no LHSa regra o excluirá, como o Pac-Man de *GhostKill*. Se ele não estiver mapeado e estiver no RHS, a regra o criará. Por fim, para aplicar uma regra (ou seja, alterar o estado atual do sistema), é necessário encontrar uma correspondência mapeando os elementos do LHS em seus elementos correspondentes no gráfico de estado.

3.3. GrameStation

O GrameStation [Junior et al. 2021] é uma ferramenta baseada em GG usada para criar e executar jogos modelados de acordo com essa linguagem formal. Como os jogos são representados como GG, ela também promove o desenvolvimento de habilidades relacionadas à CT. Essas habilidades são desenvolvidas tanto pela pessoa que cria um jogo (especifica um GG) quanto pela pessoa que executa um jogo (simula um GG) [da Silva et al. 2021]. O GrameStation está organizado em três módulos: Grame Explorer, Grame Builder e Grame Player (Figura 3). Esses módulos permitem que os usuários encontrem, criem e executem jogos, respectivamente.

Na ferramenta, o gráfico de tipos corresponde a uma área de declaração, o gráfico inicial refere-se à organização do jogo quando ele é iniciado e as regras representam as possíveis ações a serem executadas pelo jogador. Finalmente, para jogar um jogo (usando o módulo Grame Player), o usuário pode selecionar, mapear e aplicar as regras especificadas durante a execução. Quando uma regra é selecionada, os gráficos LHS e RHS são exibidos, e o usuário deve encontrar uma correspondência clicando nos elementos LHS e, em seguida, nos elementos correspondentes no gráfico de estado - o GrameStation sinaliza quando uma correspondência está correta ou incorreta.

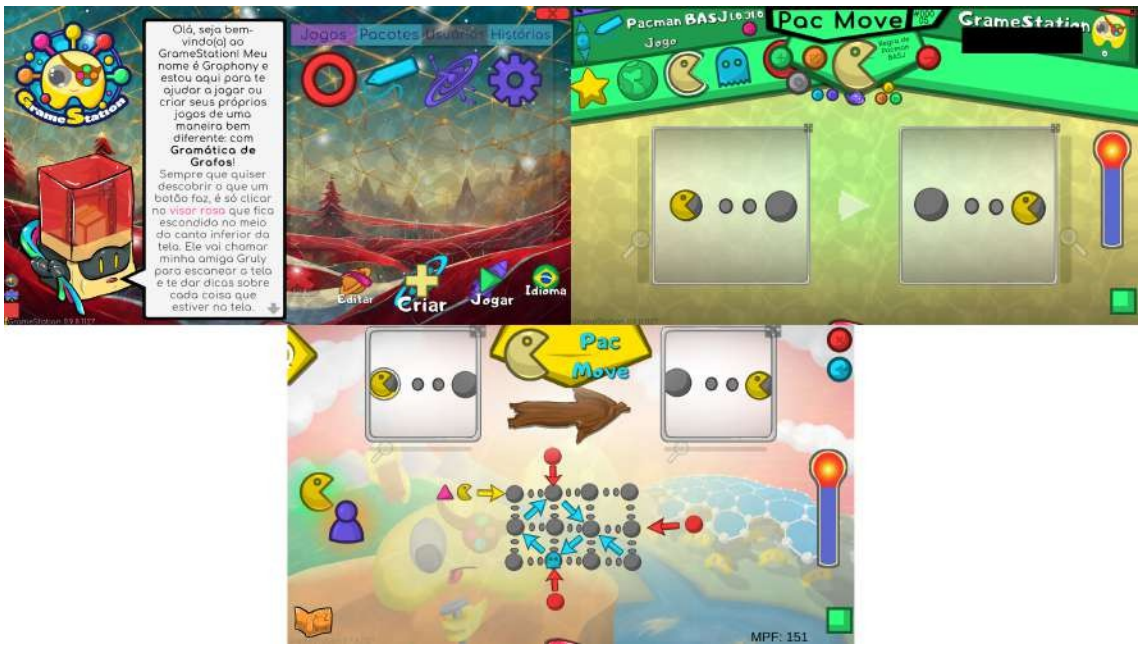

Figura 3. Módulos Explorer (em cima, à esquerda), Builder (em cima, à direita) e Player (embaixo) do GrameStation

4. Nossa abordagem

Nesta seção, apresentamos a atividade proposta. Na subseção 4.1, descrevemos as etapas metodológicas envolvidas na concepção da proposta e, na subseção 4.2, detalhamos as tarefas propostas.

4.1. Metodologia

O estudo de autômatos finitos envolve vários conceitos, como estados, transições e fita de entrada. Como o público-alvo desta proposta são alunos do ensino fundamental, uma introdução gradual desses conceitos é mais adequada. De acordo com Resnick (2017), é

É importante que uma atividade seja inicialmente projetada com objetivos e ferramentas simples (piso baixo) e, ao mesmo tempo, permita a expansão para incluir conceitos mais complexos (teto alto). Nesse contexto, as etapas metodológicas seguidas nesta proposta foram:

1. **Escolha do tema**: selecionamos um tema que engajaria o público-alvo. Nosso objetivo é desenvolver uma habilidade que faz parte do eixo CT da BNCC e é destinada a alunos do 9º ano do ensino fundamental. Portanto, a atividade educacional "A Missão Intergaláctica" é um jogo com tema espacial criado para ensinar a teoria dos autômatos a esse público (alunos do ensino fundamental e médio). O jogo se passa em um futuro em que a humanidade colonizou vários planetas. O jogador assume o papel do Robô Explorador, encarregado de ajudar a colonizar um planeta recém-descoberto.
2. **Definição dos conceitos abordados na atividade**: identificar os principais conceitos a serem introduzidos. A atividade é baseada na habilidade **EF09C003** da BNCC, definida como a capacidade de utilizar autômatos para descrever comportamentos de forma abstrata, automatizando-os por meio de uma linguagem de programação baseada em eventos. Nesse contexto, a atividade será desenvolvida em etapas, cada uma abordando diferentes modelos de autômatos. O primeiro estágio abrangerá conceitos relacionados ao DFA, incluindo estados, transições (indefinidas), estados inicial e final, aceitação e rejeição, fita de entrada e linguagem reconhecida. Os estágios subseqüentes abordarão os conceitos de NFA e PDA. Para cada estágio, serão seguidas as seguintes etapas:
    (a) **Projeto da tarefa inicial**: essa tarefa abrangerá os conceitos fundamentais necessários para entender o modelo em questão.
    (b) **Projeto de uma tarefa que englobe todos os conceitos**: projetar uma tarefa abrangente que integre os principais conceitos do modelo em consideração.
    (c) **Implementação das tarefas propostas nos itens anteriores**: colocar em prática as tarefas previamente projetadas, usando o GrameStation.
    (d) **Projeto de níveis intermediários**: desenvolvimento de tarefas intermediárias para unir conceitos básicos e avançados.
    (e) **Projeto de níveis mais avançados**: criação de tarefas avançadas para desafiar e aprofundar a compreensão.
    (f) **Estudo de caso piloto**: aplicação das tarefas implementadas a um grupo de alunos do 9º ano para identificar possíveis problemas e desafios, bem como para avaliar a compreensão dos conceitos introduzidos.
    (g) **Redesenho e implementação de todas as tarefas**: Com base nos resultados obtidos no estudo de caso piloto, serão analisados os ajustes para todas as tarefas. Posteriormente, a implementação de todas as tarefas prosseguirá.
    (h) **Estudo de caso**: será realizado um estudo de caso com uma turma do 9º ano do ensino fundamental para avaliar a eficácia de toda a etapa relacionada ao modelo em questão.

Neste trabalho, nos concentramos apenas na primeira etapa relacionada ao DFA. Uma descrição das etapas (a), (b) e (c) é apresentada na subseção 4.2, enquanto as etapas (d) e (e) são descritas na seção 5. As etapas (f) a (h) são trabalhos futuros.

4.2. Atividade: Tarefas projetadas e implementadas

O primeiro estágio é estruturado em fases, cada uma representada por um DFA correspondente a uma tarefa específica que o robô explorador deve concluir. A fase inicial se concentra na exploração do terreno do planeta, utilizando sensores para identificar zonas seguras e os principais recursos naturais, como

como água e minerais. Este artigo se aprofundará nessa fase introdutória, examinando o primeiro autômato da fase inicial.

A primeira fase é uma introdução, começando com um autômato simples de um pequeno número de estados. O DFA é composto pelos seguintes estados: *Descansando-Terra*; *Viajando-Destino*; *Descansando-Destino*; *Explorando-Mina*; e *Viajando-Terra*. O estado inicial do DFA é *Resting-Earth*, em que o robô inicia a atividade em seu planeta natal, pronto para começar a tarefa. O estado *Traveling-Destination* indica que o robô está a caminho do planeta de destino para exploração. Ao chegar, o robô entra no estado *Rest- ing-Destination*, preparando-se para começar a explorar o novo planeta, especificamente uma mina, neste caso. Após a exploração, o robô faz a transição para o estado *Traveling-Earth (Viajando para a Terra*), o que significa sua jornada de retorno ao planeta de origem. As transições entre os estados do robô são acionadas pelos comandos *To Leave*, *To Land* e *To Explore* (*"Partir"*, *"Aterrissar"* e *"Explo- rar"*, respectivamente). O objetivo é guiar o robô pelo DFA corretamente, permitindo que ele conclua a tarefa determinada.

Na primeira tarefa, o aluno recebe uma fita completa e deve, usando as regras apropriadas, identificar o estado em que o robô deve parar ao executar a sequência de comandos na fita. Em seguida, o aluno processa e verifica a fita para checar se ela está correta. Se o robô atingir o estado especificado pelo aluno, ele poderá avançar no jogo; caso contrário, não poderá. A fita nunca contém uma sequência de comandos inválida. Na segunda tarefa, o aluno recebe o autômato completo, incluindo o(s) estado(s) final(is), e uma fita em branco para construir a sequência de comandos que levará o robô do estado inicial a um dos estados finais. Nesse cenário, o jogador pode criar uma sequência inválida, resultando em uma transição indefinida e uma "perda" da fase. Como alternativa, se o robô não atingir um estado final até o fim da fita, isso também resultará em uma "derrota". O aluno só passa de nível se concluir a leitura da fita inteira e terminar em um estado final.

A Figura 4 ilustra o gráfico de tipos da atividade. Há cinco estados possíveis para o robô, com transições rotuladas entre esses estados. O vértice do sino amarelo demarca o estado inicial, o robô é o personagem principal, o vértice esférico com transparência indica um estado não final e o vértice da bandeira representa o estado final. Há também um vértice correspondente às células de fita, que podem conter símbolos de comando ou "brancos", juntamente com um ponteiro vermelho que indica a posição atual da célula de fita. Além disso, há um vértice que indica um erro, que será explicado mais adiante nas regras.

A Figura 5 apresenta o gráfico inicial, mostrando todos os estados como não finais e com o robô no primeiro estado (*Repouso-Terra*). O ícone do símbolo atual da fita indica que a fita é lida da esquerda para a direita. O vértice do estado de parada (vértice da bandeira) também está incluído para que, de acordo com regras específicas, o aluno possa determinar em que estado a tarefa terminará. A Figura 6 apresenta exemplos de regras que permitem que o jogador designe o estado de parada em um dos estados. Essas regras declaram que, se um estado for ininterrupto (conectado ao vértice transparente), ele poderá ser . A borda que conecta o estado ao vértice não final é então removida, e uma nova borda é criada conectando o estado à bandeira do estado final. Como há cinco estados nesse estágio, são necessárias cinco regras desse tipo.

Na primeira tarefa, o aluno recebe a fita concluída e deve, além de marcar o estado final, mover o robô pelo autômato. Para isso, o transi

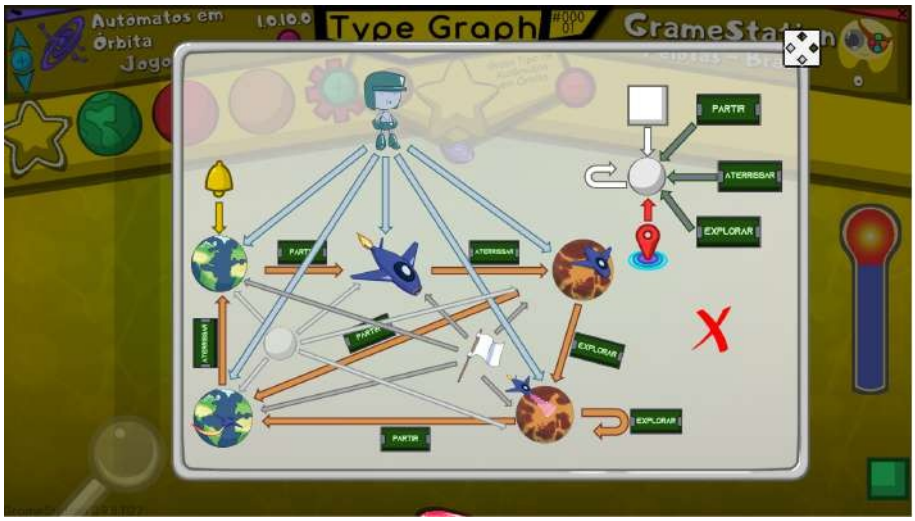

Figura 4. Gráfico do tipo de atividade no GrameStation

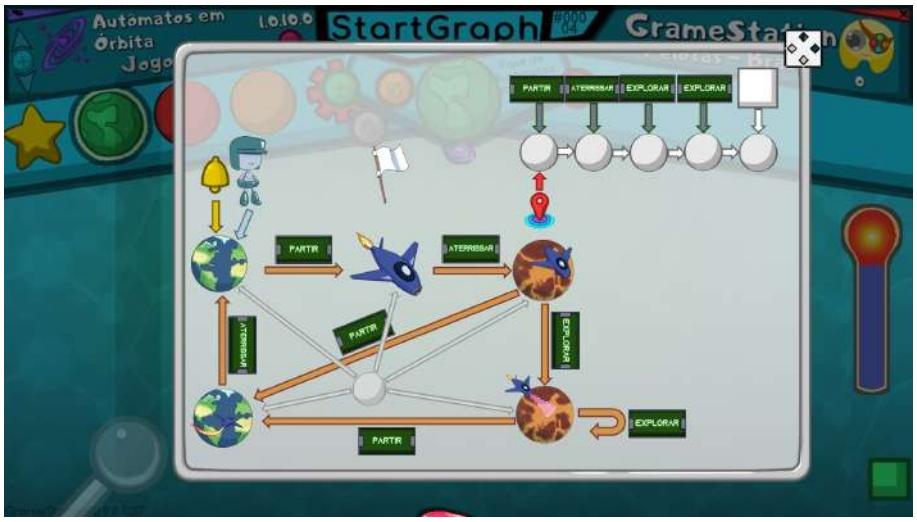

Figura 5. Gráfico inicial da atividade no GrameStation

Figura 6. Exemplos de regras *"FinalState*

são definidas. Exemplos dessas regras podem ser vistos na Figura 7. Se a transição entre os estados do autômato corresponder ao símbolo atual na fita, a transição poderá ser executada. Como há sete transições no autômato apresentado, sete transições

regras são necessárias. A Figura 7 ilustra a transição do estado *de Repouso-Terra* para o estado *de Viagem-Destino* (esquerda) e do estado *de Viagem-Destino* para o estado *de Repouso-Destino* (direita).

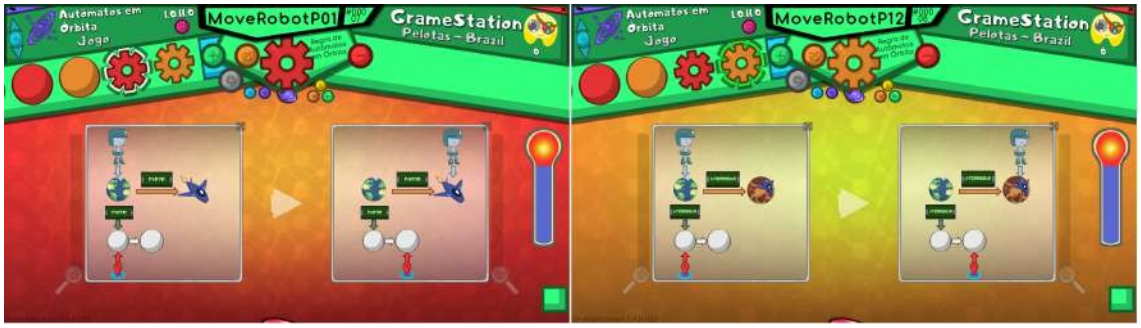

Figura 7. Exemplos de regras *do "MoveRobot*

Na segunda tarefa, o aluno está ciente do estado final do autômato e deve construir a sequência de comandos na fita para guiar o robô até esse estado. Nesse caso, o gráfico inicial contém o vértice de bandeira relacionado a um ou mais estados e a fita tem apenas vértices "brancos". Portanto, o jogador deve substituir os símbolos "brancos" na fita pelos rótulos de ação correspondentes. A Figura 8 ilustra exemplos de regras para essa tarefa. Como há três tipos de rótulos (*To Leave*, *To Land* e *To Explore*; *"Partir"*, *"Aterrissar"* e *"Explorar"* em português), são necessárias três regras. A Figura 8 mostra duas dessas regras. Depois que o estado final é marcado e a fita é preenchida, o jogador pode continuar a guiar o robô pelo autômato com as regras de transição.

Figura 8. Exemplos de regras *"WriteOnTape*

Além disso, são implementadas regras para lidar com os erros cometidos pelo jogador. primeira tarefa, por exemplo, o erro é terminar a leitura da fita em um estado que não é final. Ou seja, quando o ponteiro está marcando a última célula da fita, mas o estado em que o robô se encontra não tem relação com o vértice da bandeira (Figura 9 à esquerda). Na segunda tarefa, por exemplo, selecionar um estado em que a transição não corresponde ao rótulo atual na fita é um erro. Nesse caso, o robô é desconectado do estado atual e um vértice de erro é exibido na tela. A Figura 9 (à direita) ilustra exemplos de regras para esse cenário. Por , na regra *Error2*, o robô está no estado *Resting-Earth*, mas o símbolo atual fita é *To Explore* (*"Explorar"*). Como a única transição válida do estado *Repouso-Terra* é *Partir* (*"Partir"*), essa regra resulta em um comando inválido. Na segunda tarefa, além das transições inválidas, também pode acontecer de o robô não atingir o estado final, o que também representa "derrota" na fase.

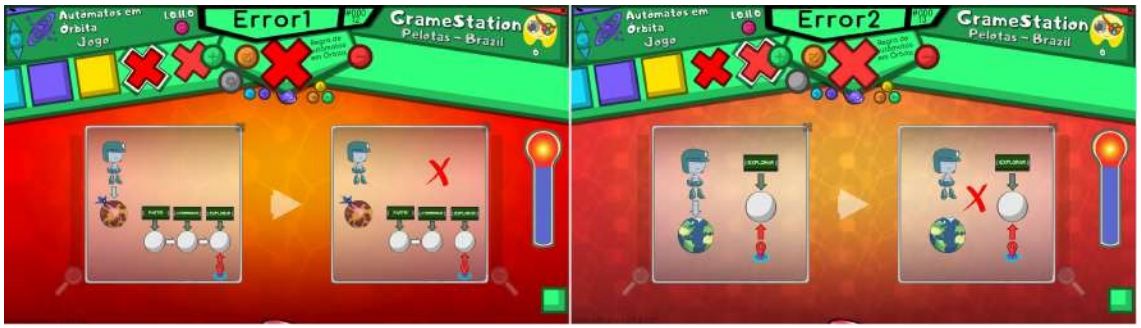

Figura 9. Exemplos de regras de "Erro"

5. Discussão

A Tabela 1 apresenta uma breve descrição das abordagens propostas, listando os conceitos abordados e destacando os principais pontos a serem considerados. As tarefas já implementadas no GrameStation correspondem às abordagens 1 e 5. As abordagens 6 e 7 darão suporte aos níveis avançados.

Dentro das abordagens propostas, além dos conceitos computacionais, elas também habilidades de TC. Além das relações já estabelecidas por Silva Junior et al. (2019) usando GG, é importante observar que a manipulação e a especificação de autômatos também aprimoram essas habilidades. O uso de autômatos para representar o comportamento do robô durante a missão exercita *a abstração*, especificamente por meio de estados que representam sua condição atual e transições que representam ações que podem modificar essa condição. Ao especificar os autômatos, os alunos definem *algoritmos* usando uma linguagem orientada por eventos para resolver problemas. Ao definir a fita para que o robô execute uma missão específica, eles estão delineando a sequência de ações (*algoritmo*) para que o robô cumpra a missão. Além disso, para identificar a linguagem reconhecida por um autômato, é necessário discernir os *padrões* em várias fitas de entrada e *generalizar* as regras de formação para todas as palavras aceitas. As habilidades *de depuração* também podem ser desenvolvidas quando os alunos recebem a tarefa de definir uma fita de entrada que orienta o robô a concluir uma missão. Isso permite que os alunos simulem o processamento da fita para verificar se a missão foi cumprida.

6. Conclusão

Este artigo apresenta uma proposta de atividade que utiliza a teoria dos autômatos no ensino fundamental e médio, especificamente usando a linguagem formal GG. A atividade foi projetada para ser envolvente e educativa, com o objetivo de ajudar os alunos a compreender os conceitos abstratos da teoria dos autômatos por meio da aplicação prática de forma interativa. Ao concluir as fases da atividade, os alunos não apenas aprendem sobre autômatos, mas também desenvolvem habilidades de CT, como abstração, representação de dados, coleta de dados, análise, simulação e reconhecimento de padrões.

Em trabalhos futuros, pretendemos expandir a atividade com a inclusão de fases adicionais, aumentando gradualmente sua complexidade, além de explorar outros tipos de autômatos, como os não determinísticos. Além disso, pretendemos testar a atividade com o público-alvo: crianças do ensino fundamental.

Abordagem	Descrição	Comentários	Conceitos de autômatos
1	Os alunos recebem uma fita completa e precisam identificar o estado de parada do robô antes de simular o autômato. Depois disso, eles devem processar a sequência de rótulos/comandos na fita para verificar se identificaram o estado de parada do robô. especificou o estado corretamente.	A fita nunca contém uma sequência inválida de comandos.	Estados e transições.
2	Os alunos recebem repetidamente a mesma fita e devem especificar um estado inicial para o autômato. A cada iteração, eles o autômato para observar o estado em que a tarefa termina.	Como a fita é a Da mesma forma, a tarefa pode ou não ser concluída, dependendo da entrada. O resultado pode variar de acordo com o estado inicial escolhido.	Estado inicial e transição ções.
3	Os alunos devem determinar qual estado que o autômato alcançará após processar uma sequência de símbolos e indicar se a tarefa será concluída (ou seja, alcançará o estado final) ou não.	Há dois casos: (i) a palavra é lida e o autômato para em um estado final, e (ii) a palavra é lida e o autômato não para em um estado final.	Aceitação ou rejeição de uma palavra.
4	Dada uma sequência de instruções Em um dos testes, o aluno deve indicar se o robô pode concluir a tarefa (ou seja, se é possível ler a fita inteira).	Porque a transição é parcial, talvez não seja possível concluir todas as instruções na fita devido a transições indefinidas.	Transições indefinidas.
5	Os alunos recebem uma O robô é um autômato, incluindo o(s) estado(s) final(is), e uma fita em branco para construir a sequência de comandos que levarão o robô do estado inicial a um dos estados finais.	É possível criar uma sequência de comandos inválida, resultando em uma transição indefinida.	Estados inicial e final, transições, aceitação ou rejeição de uma palavra e transições indefinidas.
6	Os alunos devem determinar o re instruções necessárias para concluir a tarefa testando diferentes opções de fitas. Por fim, eles devem identificar de maneira geral o que é necessário para concluir a tarefa.	Os alunos podem identificar apenas um subconjunto do idioma.	Idioma reconhecido.
7	Com um idioma, os alunos deve construir um autômato que aceite essa linguagem.	O idioma fornecido será regular, garantindo que sempre seja possível criar um autômato que o reconheça.	Especificação de autômatos.

Tabela 1. Design de tarefas em diferentes níveis

Referências

Barr, V. e Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *ACM Inroads*, 2(1):48-54.

Brasil (2022). Normas sobre Computação na Educação Básica. http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=182481-texto-referencia-normas-sobre-computacao-na-educacao-basica&category_slug=abril-2021-pdf&Itemid=30192. Online. Acessado em março de 2024.

Carvalho, F., Junior, M. C., e Costa, Y. (2021). Jogos Educativos no Ensino de Autômato Finito Determinístico: Um Estudo de Caso com o Jogo A Factory Disas- ter. In *Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*, páginas 472-478, Porto Alegre, RS, Brasil. SBC.

Gómez, I. U., Martinez-Ríos, E. A., Bustamante-Bello, R., Ramírez-Mendoza, R., e Ramírez-Montoya, M. S. (2024). Experimentando a robótica leve na educação: A Systematic Literature Review from 2006 to 2022. *IEEE Transactions on Learning Technologies*, páginas 1-18.

Chen, P., Yang, D., Metwally, A. H. S., Lavonen, J. e Wang, X. (2023). Fostering Computational Thinking Through Unplugged Activities (Promovendo o pensamento computacional por meio de atividades desconectadas): A Systematic Literature Review and Meta-Analysis. *Revista Internacional de Educação STEM*, 10(1):47.

Ching, Y.-H. e Hsu, Y.-C. (2023). Robótica educacional para desenvolver o pensamento computacional em jovens aprendizes: A Systematic Review. *TechTrends*, páginas 1-12.

da Silva, J. V., Junior, B. S., Foss, L., e Cavalheiro, S. (2021). Adaptação do processo engajado para o desenvolvimento de conteúdos curriculares em uma plataforma de jogos baseada em Gramática de Grafos. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 316-327, Porto Alegre, RS, Brasil. SBC.

Ehrig, H., Heckel, R., Korff, M., Löwe, M., Ribeiro, L., Wagner, A., e Corradini, A. (1997). Algebraic Approaches to Graph Transformation (Abordagens algébricas para transformação de gráficos). Parte II: Abordagem Single Pushout e comparação com a abordagem Double Pushout. Em *Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations (Fundamentos)*. World Scientific Publishing Co., Inc.

Farias, E., Lopes, P., Carvalho, W., e Porfírio, E. (2023). Análise da Adoção de Pensa- mento Computacional no Contexto Escolar Brasileiro: Um Mapeamento Sistemático da Literatura. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 1625-1636, Porto Alegre, RS, Brasil. SBC.

Isayama, D., Ishiyama, M., Relator, R., e Yamazaki, K. (2016). Educação em ciência da computação para alunos do ensino fundamental e médio: Teaching the Concept of Automata (Ensinando o conceito de autômatos). *ACM Trans. Comput. Educ.*, 17(1).

Junior, B. S., Cavalheiro, S., e Foss, L. (2021). Gramestação: Especificando jogos com grafos. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 499-511, Porto Alegre, RS, Brasil. SBC.

Lee, S. J. e Kwon, K. (2024). Uma revisão sistemática da educação sobre IA em salas de aula do ensino fundamental e médio de 2018 a 2023: Topics, Strategies, and Learning Outcomes (Tópicos, estratégias e resultados de aprendizagem). *Computadores e educação: Artificial Intelligence*, 6:100211.

Leite, L., Sibaldo, M. A., de Carvalho, T., e de Souza, R. (2014). Montanha de Chom- sky: Jogo Tutor para Aux'ılio no Ensino de Teoria da Computação. In *Anais do XXII Workshop sobre Educação em Computação*, pages 110-119, Porto Alegre, RS, Brasil. SBC.

Mogensen, T.. (2024). *Introduction to Compiler Design*. Springer Nature.

Resnick, M. (2017). *Lifelong Kindergarten (Jardim de infância para toda a vida): Cultivating Creativity Through Projects, Passion, Peers, and Play [Cultivando a criatividade por meio de projetos, paixão, colegas e brincadeiras*]. MIT Press.

Sanusi, I. T., Oyelere, S. S., Vartiainen, H., Suhonen, J., e Tukiainen, M. (2023). A Systematic Review of Teaching and Learning Machine Learning in K-12 Education (Uma revisão sistemática do ensino e aprendizagem de aprendizado de máquina no ensino fundamental e médio). *Educação e Tecnologias da Informação*, 28(5):5967-5997.

Silva, R. C., Binsfeld, R. L., Carelli, I. M., e Watanabe, R. (2010). Automata De- fense 2.0: Proposição de um Jogo Educacional para Apoio em Linguagens Formais e Autoˆmatos. No *Simpósio Brasileiro de Informática na Educação (Simpósio Brasileiro de Informática na Educação ao-SBIE)*, volume 1.

Silva Junior, B., Cavalheiro, S., e Foss, L. (2019). Revisitando um Jogo Educacional para Desenvolver o Pensamento Computacional com Gramática de Grafos. No Simpósio *Brasileiro de Informática na Educação (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 30, página 863.

Silva Junior, B., Cavalheiro, S., e Foss, L. (2021). Ciência da Computação na Educação Básica: Uma Revisão Sistemática. In *Anais do VI Workshop-Escola de Informática*, páginas 133-140, Porto Alegre, RS, Brasil. SBC.

Silva Junior, B. A. d. (2020). Ggasct: trazendo métodos formais para o pensamento computacional. Dissertação de mestrado, Universidade Federal de Pelotas.

Tomizawa, M. e Junior, M. C. (2021). Fábrica de Brinquedos Autômatos: Um Jogo Educativo para Ensino de Autoˆmato com Pilha. In *Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 389-397, Porto Alegre, RS, Brasil. SBC.

Vieira, M. e Sarinho, V. (2019). Ma quina de Senhas: Um Jogo Digital para o Apren- dizado da Teoria dos Autoˆmatos. In *Anais da XIX Escola Regional de Computação Bahia, Alagoas e Sergipe*, pages 54-59, Porto Alegre, RS, Brasil. SBC.

Wing, J. M. (2006). Computational Thinking (Pensamento computacional). *Communications of the ACM*, 49(3):33-35. Yim, I. H. Y. e Su, J. (2024). Artificial Intelligence (AI) Learning Tools in k-12 Edu-
cação: A Scoping Review. *Journal of Computers in Education*, páginas 1-39.