Gerenciador de Conteúdo da Matéria LFA Baseado em Autômatos Finitos Determinísticos com Saída

Igor S. Froehner[1], **Matheus S. Geremias**[1]

[1]Universidade Estadual do Estado de Santa Catarina (UDESC)
Caixa Postal 631 – 89.219-710 – Joinville – SC – Brazil

igor.sf14@edu.udesc.br, suppersoppa@gmail.com

Abstract. *This article describes an application of Deterministic Finite Automata with Output, specifically Mealy machines, in the context of online content management for the UDESC "Linguagens Formais e Automatos"(LFA, in english Formal Languages and Automata) discipline. Initially, the formalisms used are dealt with, such as deterministic finite automata with output, and finally, it is applied to the online content manager where the final output is the content that has been studied.*

Resumo. *Este artigo descreve uma aplicação de Autômatos Finitos Determinísticos com Saída, especificamente máquinas de Mealy, ao contexto do gerenciamento de conteúdo online para a disciplina de Linguagens Formais e Autômatos(LFA) da UDESC. Inicialmente são abordados os formalismos usados, como autômatos finitos determinísticos com saída, e por fim aplica-se a o gerenciador de conteúdo online onde a saída final é o conteúdo que foi estudado.*

1. Introdução

Linguagens Formais são um tópico de fundamental importância pra computação, e tem várias aplicações tanto teóricas quanto práticas. Um dos principais temas abordados na área são os Autômatos Finitos (AFs), estes inicialmente com a tarefa de fazer o reconhecimento ou a rejeição de palavras, além de ser discutido uma extensão dos AFs, os Autômatos Finitos com Saída, que não somente aceitam ou rejeitam palavras, mas também geram uma saída de acordo com a palavra de entrada e, no caso de uma máquina de Mealy, relacionando as saídas com as transições [Menezes 1998].

Com o advento da evolução da computação nas mais variadas áreas, a sociedade tem se tornado cada vez mais digital, de modo que não é mais necessário estar comparecer presencialmente para participar dos mais diversos eventos. Movimento que foi extremamente acelerado pela pandemia de COVID-19, na qual não comparecer presencialmente aos lugares tornou-se essencial para o seu controle do vírus [Werneck and Carvalho 2020]. Dada essa conjectura, as salas de aula não podem parar e tem que se adaptar. As aulas se tornaram remotas e, com isso, o conteúdo não é mais necessariamente passado no quadro pelo professor, este pode ter se tornado em links que levam a páginas que tem o conteúdo. Então propomos uma ferramenta para gerenciar tal conteúdo, especificamente para a matéria Linguagens Formais e Autômatos (LFA) da UDESC, uma vez que bom gerenciamento desse conteúdo é de essencial importância para

o processo pedagógico [Morgado et al. 2020]. E para tal foi usado um modelo baseado em Autômatos Finitos.

Esse trabalho será estruturado da seguinte forma, na seção 2 se explora os conceitos teóricos usados no desenvolvimento; na seção 3 é analisado uma visão mais aprofundada ao problema a ser abordado; já a seção 4 elucida como os conceitos teóricos foram modelados e aplicados ao problema; a seção 5 ilustra como tais conceitos foram implementados usando a linguagem de programação Python; na seção 6 é exposto as conclusões da aplicação implementada; e por fim algumas ideias de trabalhos futuros são citadas na seção 7

2. Contexto

Com o intuito da compreensão deste trabalho, é necessária a introdução aos conceitos teóricos que serão abordados, portanto começa-se contextualizando o conteúdo. Os principais conceitos usados são: o autômato em si, envolvendo tanto o que é um autômato, o que são autômatos com saída e máquinas de Mealy;

2.1. Autômatos Finitos Determinístico

Um autômato finito pode ser entendido como uma máquina de estados composta de três componentes: fita, unidade de controle e função programa ou função de transição. A fita é a informação a ser processada pelo autômato, a unidade de controle é a responsável pela leitura da fita e pela movimentação, que movimenta-se somente para um lado, seja direita ou esquerda, e a função programa é a função que comanda as leituras e define o estado da máquina [Menezes 1998]. Os estados ditam comportamentos específicos da máquina, ou seja, eles ditam como a máquina se comporta no momento em que está no estado em questão. O determinismo, por sua vez, é dado pela característica que todo estado tem no máximo uma transição usando um símbolo do alfabeto, ou seja, não pode haver duas transições saindo de um mesmo estado com o mesmo símbolo. E é chamado finito pois o número de estados é finito.

2.2. Autômatos Finitos com Saída e Máquina de Mealy

AFs com Saída são uma extensão natural dos AFs (AFD ou AFN), em que além do autômato aceitar ou rejeitar uma palavra, ele também gera saídas relativas a palavra de entrada. Essas saídas estão associadas às transições na máquina de Mealy, e aos estados na máquina de Moore[Menezes 1998]. Como o modelo escolhido para o desenvolvimento do trabalho foi o da máquina de Mealy este vai ser mais desenvolvido nessa contextualização. Estes são definidos pela 6-tupla a seguir:

$$G = (\sum, Q, \delta, q_0, F, \Delta)$$

Onde:

$\sum$	alfabeto de símbolos de entrada;
Q	conjunto finito de estados possíveis do autômato;
δ	função programa;
q_0	estado inicial do autômato, $q_0 \in Q$;
F	conjunto de estados finais, $F \subset Q$;
Δ	alfabeto de símbolos de saída.

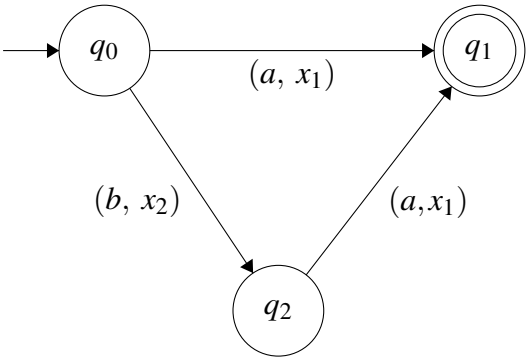

Figura 1. Máquina de Mealy

Os componentes $\sum$, Q, q_0 e F são comuns aos autômatos finitos, uma vez que Δ(alfabeto de saída) é adicionado a função programa é relacionada a esse alfabeto de modo que cada uma das saídas é dada pela transição em questão. Dessa forma pode-se representar a função programa por meio de um grafo direcionado, adicionando etiquetas às transições [Menezes 1998], um exemplo pode ser visto na figura 1.

3. Descrição do Problema

Como visto anteriormente, o momento atual levou o ensino a se tornar cada vez mais digital, assim, gerenciar esse conteúdo e o aprendizado do aluno é imprescindível para o processo pedagógico. Por conta disso, uma aplicação que ajudasse o estudante a se organizar e se orientar em cada matéria é de grande valia.

Mas com isso surgem questões a cerca do que deve ser estudado, em que ordem, se o aluno viu os conteúdos e como ele realizou este processo.

Além disso, partindo do princípio que cada disciplina tem seu próprio currículo, é necessário um sistema que seja de fácil alteração das etapas e seus conteúdos.

Dessa maneira, tendo como base um autômato de saída, é necessário definir os estados do sistema, as transições possíveis, o critério de finalização de estudo, uma maneira de realizar a visualização do autômato para o estudante reconhecer o caminho existente, como será feito a comunicação entre sistema-aluno e determinar o cálculo da porcentagem do curso que foi estudada.

4. Modelagem Computacional

Após analisar o problema e os fundamentos nos quais será baseado o sistema, a próxima etapa é modelar os conceitos de modo que possam ser implementados na linguagem escolhida, que, por conta de sua simplicidade e por algumas características de orientação a objetos que facilitam a implementação e a sua grande disponibilidade de bibliotecas, será Python[1], especificamente a versão 3.8.5.

4.1. Estados do sistema

Tendo como base a disciplina de LFA, foi decidido que cada um dos estados do autômato seria o equivalente a uma aula, de maneira que o estudante que se encontra na aula01 deve estudar o conteúdo referente a aula 01 para avançar para o estado aula02 e assim segue.

[1]https://www.python.org/

Desta maneira, existirão 18 (dezoito) estados de conteúdo, enumerados como visto acima.

4.2. Transições

Basicamente, cada transição representa o ato de estudar uma aula. Sabendo que dado uma entrada a transição deve ter como saída o conteúdo da aula do próximo estado, optou-se por ter uma saída mais genérica no console do sistema, de x01 até x18. Tal decisão foi tomada por conta do efeito estético que existia ao ter como saída os links inteiros, deixando assim de maneira mais sucinta.

Assim, após ocorrer a saída no console é aberto uma página web contendo a lista de links referentes aquela aula, compondo a saída de maneira concreta.

E além da transição para o próximo estado, cada estado (excetuando o inicial e o final) possui uma transição para si mesmo, que representa a revisão do conteúdo.

4.3. Estado final

O estado final foi uma discussão a parte e existiam várias possibilidades, como todo o estado ser final, pensando que o estudante poderia parar de estudar o momento que quiser, ou ainda criar um estado que serviria de poço, com base no mesmo pensamento anterior, mas por fim foi decidido que, sendo o objetivo do sistema auxiliar no estudo completo da disciplina, a finalização deveria ocorrer apenas após o último estado de aula.

Dessa maneira, a quantidade de estados totais do sistema é de 19 (vinte) estados.

5. Implementação Computacional

Após discutir sobre qual a linguagem seria utilizada para implementação do autômato, Python foi escolhido pois já haviam na internet vários exemplos de programas que implementaram autômatos, além de bibliotecas para a parte da visualização do AF.

Por fim, foi iniciado a implementação do modelo do autômato, que após as discussões da etapa anterior tem sua definição pela 6-tupla a seguir:

$$G = (\sum, Q, \delta, aula00, concl, \Delta)$$

Onde:

$\sum$	[a1, a2, ..., a18, a1c, a2c, ..., a18c]
Q	[aula00, aula01, ..., aula18, concl]
δ	função programa;
$aula00$	estado inicial do autômato, $aula00 \in Q$;
$concl$	conjunto de estados finais, $concl \subset Q$;
Δ	[x1, x2, ..., x18, c1, c2, ..., c18]

E δ:

δ	a1	a2	...	a18	ε
aula00	aula01				
aula01	aula01	aula02			
...					
aula18				aula18	concl
concl					

Para implementar esta 6-tupla na linguagem escolhida, a classe Estado possui um nome, uma lista com as transições existentes a partir dele e um indicador se ele é final ou não. Além disso, a classe Automato tem como atributos uma lista do alfabeto de entrada, uma lista do conjunto de estados, o estado inicial, uma lista com os estados finais, uma lista do alfabeto de saída e um dicionário com a relação entre um símbolo de saída e seus respectivos links.

E a classe Estudante possui um nome, o estado corrente, uma lista com a palavra de saída e uma instância de Automato.

Ao iniciar o programa, é escolhido quem é o aluno, após isso começa o loop principal, onde deve ser optado entre realizar uma transição ou adicionar um conteúdo personalizado, caso seja decidido por fazer uma transição o estudante deve entrar com um símbolo possível entre os que são disponíveis no momento, após isso é mostrado o autômato com o destaque para a transição e aberto um site com os links dos conteúdos, ou então o aluno deve digitar os links dos conteúdos personalizados.

5.1. Comunicação com o sistema

Inicialmente, a comunicação que o aluno poderia ter com o sistema era pelo prompt de comando, onde ao executar o programa era oferecida a escolha entre uma quantidade base de alunos ou a criação de um novo, indicando como escolher cada opção. Assim, a cada escolha necessária do sistema o aluno deveria escrever no console, como ilustra a figura a seguir.

```
Selecione o estudante que irá estudar:
1 - Igor
2 - Matheus
3 - Criar novo estudante
1
Estado atual: aula01
Porcentagem de conclusão do estudo: 5.0%
Transições possíveis:
Transicao com símbolo a1 para aula02 com saída x1

Selecione o que deseja:
1 - Realizar uma transição
2 - Adicionar um conteúdo personalizado
1
Próximo símbolo da palavra: a1

Saída até agora: x1
```

Figura 2. Console do programa.

Mas após a finalização da versão inicial, iniciou-se a criação de uma versão web, onde a interação ocorria por meio da seleção entre as opções existentes, como ilustrado abaixo na Figura 3.

Para o desenvolvimento dessa versão, foi utilizado o micro-framework Flask[2], por ser baseado em Python e por conta da sua simplicidade. A interação se dá pela exibição das possíveis transições, onde o usuário pode escolher o simbolo que quer dar como entrada e que é o próximo simbolo da palavra, dessa forma o a saída dada é os links da

[2]https://flask.palletsprojects.com/en/1.1.x/

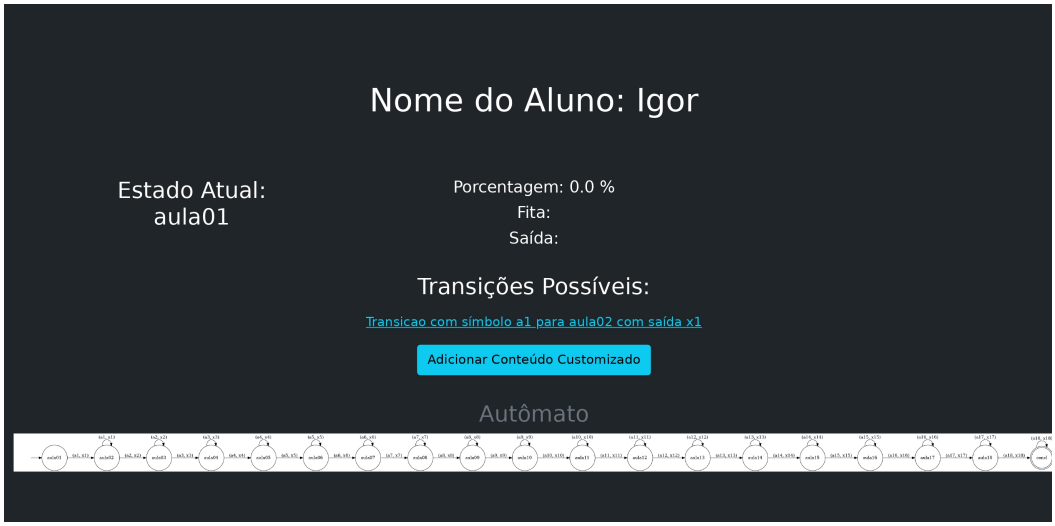

Figura 3. Screenshot Página Aluno.

transição que ele acabou de fazer. Quando uma transição é feita, a imagem do autômato é trocada destacando a ultima transição feita. Foi mantido a possibilidade do usuário criar mais transições para o estado em que ele está, representando que o estudante está estudando com um conteúdo customizado, para tal é requisitado os links que serão a saída dessa nova transição criada. Nessa implementação podem ser criados mais estudantes, uma vez que essa informação a informação de qual estudante está interagindo no momento é retirada da rota em que o sistema foi acessado, (i.g. /aluno/Igor). Apesar disso, ainda é uma versão inicial que deve ser aprimorada, tanto na interface e na implementação, como nas interações com o usuário.

5.2. Visualização do autômato

Para a visualização do autômato, foi implementado uma integração com a ferramente gráfica GraphViz[3], de maneira que a cada interação com o sistema é destacado a ação escolhida pelo estudante.

De maneira geral, foi criado uma classe Grafo, que tem como atributos um nome, o formato que será salvo cada imagem e uma instância de Digraph, classe da biblioteca do GraphViz necessária para a execução do programa.

Sabendo que cada aluno possui uma instância de Grafo, ao executar a função de desenhar é feito primeiramente o estado inicial do autômato, então é realizado uma iteração pela lista de estados finais, que no caso é apenas um, e finalmente uma iteração por todos os outros estados e suas transições. O destaque é executado ao pintar de azul a transição entre o estado antigo e o novo, caso a ação seja de finalizar o programa, nenhum destaque é realizado.

Por conta da abordagem utilizada, as transições referentes a conteúdos extras podem sobrepor o autômato base. Além disso, como o autômato possui muitos estados que estão em sequência, a ilustração acaba ficando um pouco pequena, pois é feito em uma linha horizontal.

[3]https://www.graphviz.org/

5.3. Porcentagem do curso concluída

Sabendo a quantidade total de estados referentes a aulas, o cálculo de porcentagem do curso concluída pode ser realizado ao, com base no estado atual do aluno, contar o número de estados diferentes que já foram visitados e efetuar uma regra de três com o total existe.

6. Conclusão

Por conta do momento atual, tecnologias que facilitem a organização e interação do aluno com o conteúdo são fundamentais para uma maior absorção do que foi estudado.

Pensando nisso, foi desenvolvido uma ferramenta que guiasse o estudante no decorrer de uma disciplina, apresentando os conteúdos de forma sequencial e permitindo a adição de links customizados em cada aula. O desenvolvimento de tal programa teve como base autômatos finitos, que deveriam primeiramente ser estudados.

Após ter compreendido os autômatos e seus funcionamentos, iniciou-se a etapa da modelagem. Foi neste momento que começaram a surgir dificuldades, por conta da escolha de qual autômato com saída escolher, a determinação do que representaria um estado, o que consistiria o estado final e os alfabetos do sistema.

Então veio a etapa de implementação em Python, que teve um início mais trabalhoso no momento de transformar o que seria um autômato em código, mas que depois fluiu rapidamente após o entendimento de como poderia ser feito.

Com este trabalho, foi possível perceber que ao desenvolver com base em autômatos finitos a implementação computacional ocorre com mais facilidade, pois o modelo já está pronto. Por conta disso, o campo de atuação dessa área é muito grande e o desenvolvimento desse sistema permitiu um vislumbre do potencial e as possibilidades do seu uso.

O código fonte desenvolvido e instruções para o uso da aplicação estão disponíveis em: https://github.com/IgorFroehner/cursoautomato

7. Trabalhos futuros

Entre algumas melhorias que podem ser feitas no sistema se encontram a parte de criação de exceções no código, pois caso o estudante entre com alguma coisa que não seja o que foi pedido, o programa dá erro e acaba fechando. Outra coisa que poderia ser feita é o aprimoramento da versão web existente, que ainda é muito básica e contendo bugs.

Além disso, incluir na versão web a possibilidade de criar ou editar o autômato, de forma a possibilitar o uso da ferramenta em outras disciplinas.

Referências

Menezes, P. B. (1998). *Linguagens formais e autômatos*. Sagra-Dcluzzato.

Morgado, J. C., Sousa, J. R. F., and Pacheco, J. A. (2020). Transformações educativas em tempos de pandemia: do confinamento social ao isolamento curricular.

Werneck, G. L. and Carvalho, M. S. (2020). A pandemia de covid-19 no brasil: crônica de uma crise sanitária anunciada.