UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS* APUCARANA

João Vitor de Souza Ribeiro

RELATÓRIO TÉCNICO - LFA

APUCARANA – PR
2023

João Vitor de Souza Ribeiro

RELATÓRIO TÉCNICO – LFA

Trabalho apresentado à disciplina de Linguagens Formais, Autômatos e Computabilidade do curso de Bacharelado em Ciência da Computação.

Professor: Guilherme Henrique de Souza Nakahata;

APUCARANA – PR
2023

SUMÁRIO

INTRODUÇÃO

Como citado no último relatório técnico, Linguagens Formais, Autômatos e Computabilidade pode ser vista como uma disciplina mais prática que outras do curso de Ciência da Computação, devido as suas atividades práticas realizadas em sala, como as próprias montagens de autômatos, e aplicação destes em códigos funcionais. Novamente, mostramos que existem diversas correlações entre as disciplinas do curso, e, neste caso, a aplicabilidade de Gramáticas Regulares à algoritmos plenamente funcionais mostra um pouco dessa característica. Em seu trabalho sobre a real importância da gramática para o desenvolvimento moral e intelectual das pessoas [3], Gabriel da Silva Ramos diz que “Desde os primeiros passos da civilização da raça humana (ainda na infância de sua história), as linguagens tiveram papel fundamental para a eventual evolução lenta e progressiva da sociedade.”, e, de mesmo modo, na disciplina citada podemos ver que gramáticas são mecanismos geradores de uma linguagem formal, e, ainda, que a partir destas podemos gerar todas as palavras de uma linguagem formal, ou seja, a gramática é uma precursora para a disciplina e para as linguagens em si.

Ainda sobre o trabalho de Ramos, no que toca as concepções de gramática, o autor explicita que “[...] Almeida (1961, p. 23) [1] classifica a gramática como “a reunião ou exposição metódica dos fatos de uma língua”, enquanto Cegalla (2008, p. 16) [2] a constitui como “a história, registro e sistematização dos fatos de uma língua” [...]”. Ambos autores citados no trabalho são respeitados e entendem que a gramática é uma forma de sistematização e estruturação de determinada língua, como as Gramáticas tratadas dentro do curso. Em suma, a gramática se mostra como um ponto principal para o entendimento das linguagens e suas formações, sejam nas literaturas, nas disciplinas do curso ou em outras áreas, assim, a aplicação prática desta importante estrutura se mostra vantajosa e demasiadamente importante para a conceituação e aprendizado dos alunos, por meio da criação de um código fonte que represente seu funcionamento, como exposto no presente documento.

CAPÍTULO 1
OBJETIVOS

O objetivo principal do código criado pode ser interpretado como estabelecer o funcionamento prático de uma Gramática Regular, de modo que consiga realizar a execução de uma GLUD (Gramática Linear Unitária à direita) e GLUD (Gramática Linear Unitária à esquerda). De modo geral, o código fonte tem como objetivo receber uma descrição formal, ou seja, as características que definem a linguagem daquela gramática desejada, e as suas ordens de produções, com isso, poderão avaliar a aprovação ou não de palavras mediante a linguagem informada por meio das transições. O código também deve informar por quais ordens de produção a execução passou, além da impressão da palavra sendo formada por etapas. A Figura 1 traz um exemplo de uma GLUD com a sua descrição formal e ordens de produção. Já a Figura 2 um exemplo de uma GLUE com a sua descrição formal e ordens de produção.

- $L = \{a^n b^m / n \geq 0, m > 1\}$
- V = {S,A,B,X}
- T = {a,b}
- S = S
- P = {$S \rightarrow bX$

$S \rightarrow aA$
$A \rightarrow aA$
$A \rightarrow bX$
$B \rightarrow bB$
$B \rightarrow \lambda$
$X \rightarrow bB\}$

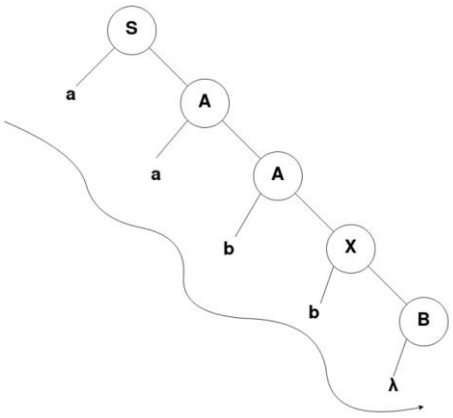

Figura 1

- $L = \{a^n b^m / n \geq 0, m > 1\}$
- V = {S,A,B,X}
- T = {a,b}
- S = S
- P = {$S \rightarrow Xb$

$A \rightarrow Ab$
$A \rightarrow Ba$
$A \rightarrow \lambda$
$B \rightarrow Ba$
$B \rightarrow \lambda$
$X \rightarrow Ab\}$

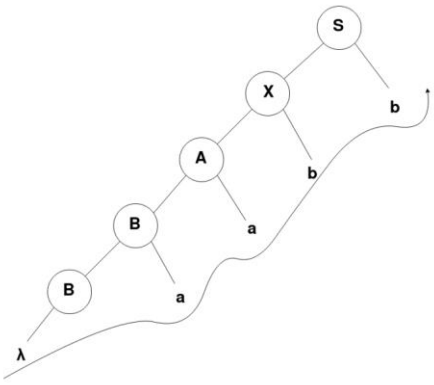

Figura 2

CAPÍTULO 2
MOTIVAÇÃO E RECURSOS UTILIZADOS

Baseando-se no exposto anteriormente, devemos explicitar os motivos para a realização do trabalho, ou seja, o objetivo final e os recursos utilizados para que isso seja cumprido.

2.1 Motivação

Como citado no capítulo que trata acerca dos objetivos do projeto em questão, a motivação também seria a realização de um código fonte funcional em que possamos demonstrar o pleno funcionamento de uma gramática regular, em seus dois formatos unitários (GLUD e GLUE), ao passo que recebemos as informações necessárias e poderemos determinar se as palavras requisitadas fazem ou não parte daquela linguagem, além de informar as ordens de produção utilizadas para que a execução seja de fato finalizada. Para tal, precisamos receber os dados, tais como a descrição formal e as ordens de produção, após, é necessário que se crie uma lógica de programação para atingir-se o esperado, de maneira eficaz. Na figura 3, por exemplo, podemos visualizar como as ordens de produção deverão ser recebidas, de maneira a imprimir o seu conteúdo formatado no modelo contido nesta.

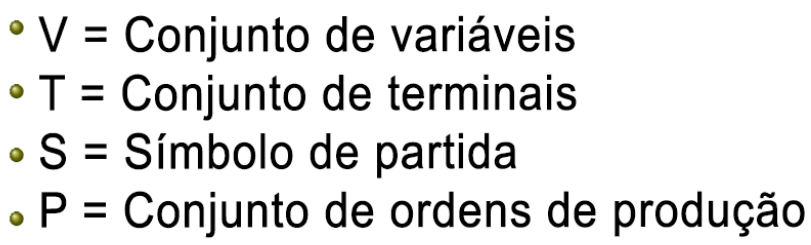

Figura 3

Assim, faz-se necessário, com as informações pertinentes acerca dos objetivos e motivações, esmiuçar-se àqueles dados relevantes à Estrutura de Dados, Linguagem de Programação e demais questões acerca da implementação do código em questão.

2.2 Estrutura de Dados

Resumidamente, o código fonte para a execução da atividade é composto por duas classes, sendo uma delas a principal, intitulada *Main*, e uma classe auxiliar para a configuração e utilização de um objeto, intitulada *Produções*. Na classe principal, inicialmente, pudemos receber o tipo de Gramática Linear Unitária, as variáveis, os terminais, o símbolo de partida e as ordens de produção, respectivamente. Após o recebimento a classe chama a execução, a verificação e a impressão dos resultados, em ordem. Cada uma dessas ações será especificada a seguir:

2.2.1 Recebimento e armazenamento dos dados

Para o recebimento da descrição formal e das ordens de produção, na classe Main, o usuário pode informar todos eles na ordem em que são requisitados, por meio do console, além, claro, da escolha entre GLUD e GLUE. Para o armazenamento destes, são utilizados vetores para as variáveis e para os terminais, e, um char para o símbolo de partida.

No que toca as ordens de produção, um objeto foi criado na classe Produções para facilitar o recebimento e desmembramento destas; de início, recebemos a ordem em formato String, completa no formato X>xX, passamos este formato para dentro do objeto criado, juntamente com a escolha entre os tipos de gramática. Dentro da classe Produções, a String é “desmembrada” de acordo com o tipo escolhido pelo usuário, de maneira que possa identificar corretamente quais posições dessa String são correspondentes aos campos “variável futura”, “variável atual” e “terminal”, que irão ser os principais elementos para a execução. Além destes atributos mencionados, a classe Produções conta com “pos” e “total”, a primeira utilizada para guardar a posição em que foi recebida e a segunda para guardar a ordem de produção inteira, da maneira como foi recebida. Após a criação deste objeto, colocamos ele dentro do ArrayList intitulado “ordens”, que, ao final do recebimento, conterá todas as ordens de produção em formato de objeto Produções, com todos os seus atributos. Ainda ao final do recebimento, o usuário digita a palavra que deseja conferir se pertence aquela linguagem fornecida para a gramática.

2.2.2 Execução

A função “Executa” é chamada logo ao final dos recebimentos, tendo como parâmetros a palavra a ser testada, o símbolo de partida e a escolha do tipo de gramática. A lista “ordens” é do tipo protected, o que garante acesso a ela em todas as funções do código. Outra lista é criada em tipo protected, intitulada “completos”, esta que irá, na função Executa, receber apenas aquelas ordens de produção que forem passadas, ou seja, aquelas que forem utilizadas para chegar a um resultado conclusivo.

Para cada uma das opções de gramática (GLUD/GLUE), uma execução diferente é realizada, na primeira – GLUD – uma StringBuilder é criada e recebe a palavra a ser testada adicionada de um *, representando que ali existe um vazio, ou seja, que a execução deve terminar. Uma variável do tipo char chamada ‘atual’ recebe o símbolo de início. A lógica de execução basicamente consiste em verificar se a variável atual é a mesma que a posição 0 na String correspondente as ordens de produção (‘atual’, no objeto), em caso positivo, conferimos se o caractere atual da palavra é igual ao terminal da ordem, se for, esta produção é necessária para a realização dessa gramática, assim, sua forma completa é adicionada a lista “completos”, e a variável atual recebe o campo ‘futuro’ de nosso objeto de ordens de produção. Após essa parte da execução, chamamos a função ‘Verifica’, que poderá devolver um valor true se a palavra teste for aceita na linguagem. Se a palavra for aceita podemos imprimir as ordens de produção contidas na lista ‘completos’, além da palavra sendo formada por cada ordem de produção.

Nos exemplos com GLUE, no entanto, a execução é alterada somente em alguns pontos, principalmente no que diz respeito ao meio em que acessamos a palavra teste para verificar as ordens que devem ser colocadas na lista ‘completos’, na Gramática Linear Unitária à direita, criamos uma repetição que percorre a palavra toda, iniciando do índice 0 e indo até seu último. Na GLUE, por outro lado, percorremos a palavra ao contrário, além de colocar o * no começo da palavra, não mais ao final dela. O restante da lógica de verificação continua o mesmo que citado anteriormente. Ainda ao final da execução o usuário pode escolher por testar outra palavra, para isso, recebemo-la e depois chamamos novamente a função ‘Executa’.

2.2.3 Verificação

A função ‘Verifica’ é do tipo boolean, ou seja, retorna um valor booleano quando a chamamos. A chamada desta função acontece dentro da função ‘Executa’, de modo que informe para a função citada se a palavra teste foi ou não aceita na linguagem. ‘Verifica’ recebe a palavra teste e a escolha entre GLUD e GLUE, para cada uma das opções existem tratamentos diferentes para verificar o aceite da palavra.

Em um primeiro momento, criamos a variável teste, do tipo boolean, que receberá true quando a palavra testada for aceita. Aqui utilizamos o campo do objeto Produções intitulado ‘pos’, que receberá, por meio de uma repetição, a sequência numérica na qual as ordens foram recebidas originalmente. Nessa etapa de verificação basicamente vamos percorrer as ordens de produção procurando por aquelas que foram utilizadas na execução, ou seja, percorremos a lista ‘completos’, colocando em uma StringBuilder auxiliar os caracteres que cada ordem forma, assim, ao final dessa repetição, obtendo a palavra formada por estas ordens, na ordem que foram colocadas na lista. Após a formação dessa palavra, por causa dos vazios (“*”), colocados nas ordens de produção, podemos ficar com uma palavra confusa, desse modo, percorremos essa palavra obtida retirando todas as aparições de *. Com isso, podemos conferir se a palavra formada é igual a palavra testada, se for, teste recebe true, indicando que a palavra foi aceita.

Essas etapas mencionadas acima são as mesmas para GLUD e GLUE, o que muda é a forma como adicionamos os caracteres na StringBuilder auxiliar, podemos explicitar cada uma delas:

GLUD: se a ordem que estou acessando estiver na posição 1 ou nunca tiver acessado nenhuma ordem antes, a variável ‘auxiliar’ recebe o terminal da ordem de produção em seu final e, após, o campo futuro em seu final. Em caso de a ordem não ser a primeira a ser acessada, substituímos a última posição existente na variável pelo terminal da ordem e adicionamos ao final da mesma o campo futuro.

GLUE: se a ordem que estou acessando estiver na posição 1 ou ainda não acessei nenhuma ordem, a variável ‘auxiliar’ recebe em seu início o terminal da ordem de produção, e, após, recebe, em seu início, o campo futuro. Em caso contrário, substituiremos a posição inicial da StringBuilder pelo

terminal da ordem e, na frente dele, colocaremos o campo futuro da mesma ordem.

Após fazer essas verificações, já com a variável ‘teste’ portando o resultado dessa, podemos retorná-lo a função ‘Executa’ que irá interpretar o conteúdo e mostrar ao usuário o aceite o não da palavra fornecida.

2.2.4 Impressão

A função ‘Imprime’ tem a função de, se a palavra testada foi aprovada, mostrar no console a passagem das ordens de produção que foram utilizadas para isso e a formação da palavra que, ao final, deve ser igual a recebida do usuário. A montagem desta função é basicamente a mesma da função ‘Verifica’, no entanto, ao passar pelas ordens de produção imprimimos o conteúdo de cada interação, como o conteúdo da StringBuilder em cada passagem da repetição (já sem os *) e, também, a posição original no recebimento e a ordem de produção responsável por aquela formação. As mesmas diferenças na lógica apresentadas no subcapítulo anterior são válidas para a impressão de GLUD e GLUE.

Além da impressão das ordens de produção, existe uma outra função intitulada ‘Titulo’ no código fonte, que, basicamente, imprime Títulos para organização da exibição no console. Um tamanho base (70) foi estabelecido como o número de caracteres totais dos títulos, visando formar uma visualização organizada. A função recebe uma String ‘texto’ e uma String ‘tipo’, a primeira se trata do texto que deverá ser impresso no centro do título, a segunda, no entanto, é a variável que representa os caracteres que formaram “a linha” do título, como as separações com ‘-‘ ou ‘=’.

2.3 Linguagem de Programação e demais informações

A linguagem escolhida para a implementação do código foi Java, de maneira que se utiliza-se a linguagem principalmente abordada no segundo ano de formação para implementar o ciclo de instruções. Java é uma linguagem de alto nível desenvolvida pela Sun Microsystems, orientada a objetos e amplamente utilizada em vários ramos da programação. Inclusive, no código é utilizado esta orientação a objetos; as bibliotecas da linguagem

utilizadas para o funcionamento do código foram Scanner e ArrayList, para recebimento dos dados pelo usuário, via console e para a criação das listas de ordens de produções e daquelas que passaram pela execução.

Além disso, foram utilizados, no código fonte, Unicodes para colorir partes específicas do código, como o título inicial, colorido de amarelo com o Unicode “\u001B[33m” e aplicando o negrito nesta e em outras partes da visualização com o Unicode “\u001B[1m”.

CAPÍTULO 3
RESULTADOS

Mediante os objetivos apresentados, o resultado esperado seria o pleno funcionamento de um código que demonstre como uma Gramática Linear Unitária (GLU) reconhece as palavras de uma determinada linguagem dado a sua descrição formal e ordens de produções. Para tal, foram implementados todas as funções e estruturas citadas anteriormente, garantindo que o funcionamento geral fosse aceitável e que surtisse os resultados esperados. As ordens de produção utilizadas para obter a aprovação da palavra, quando existe, é mostrada, juntamente com a formação dessa palavra, ordem a ordem, de maneira que possamos observar como cada uma delas edita a palavra, formando, em seu final, a mesma daquela recebida pelo usuário. Dessarte, com a implementação completa e revisada, os resultados foram atingidos, resultando em uma aplicação interativa e funcional, recebendo o tipo de GLU que deseja testar, podendo escolher entre GLUD (Gramática Linear Unitária à direita) e GLUE (Gramática Linear Unitária à esquerda), a quantidade de variáveis, as variáveis, a quantidade de terminais, os terminais, um símbolo de partida, a quantidade de ordens de produção, as ordens de produção e palavra a ser testada. Abaixo, nas Figuras 4,5,6,7,8 e 9 podemos ver o funcionamento de dois exemplos de Gramáticas, uma GLUD e uma GLUE, respectivamente

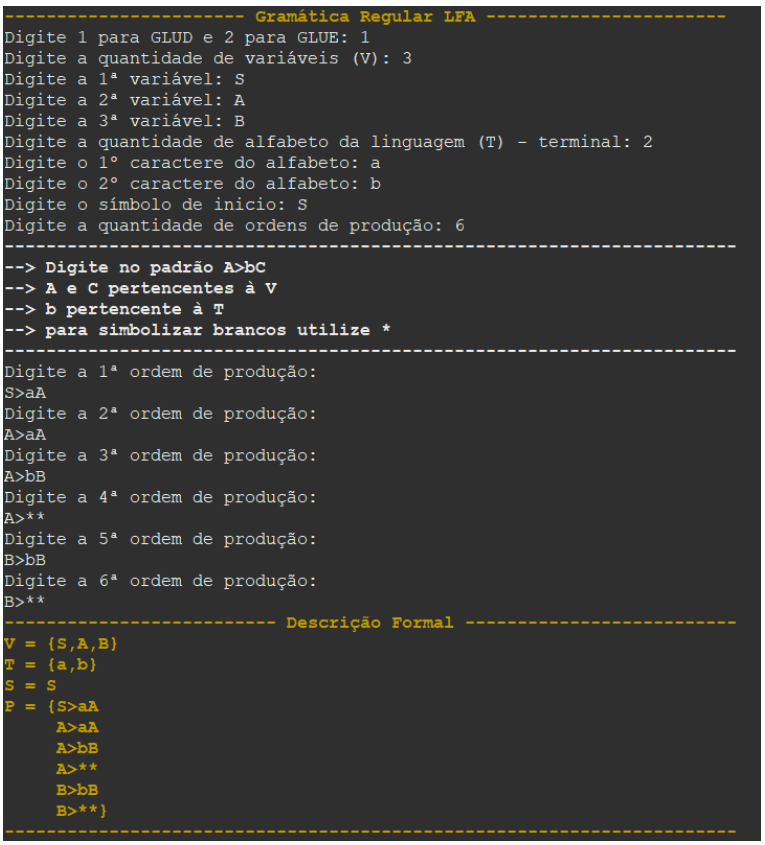

Figura 4

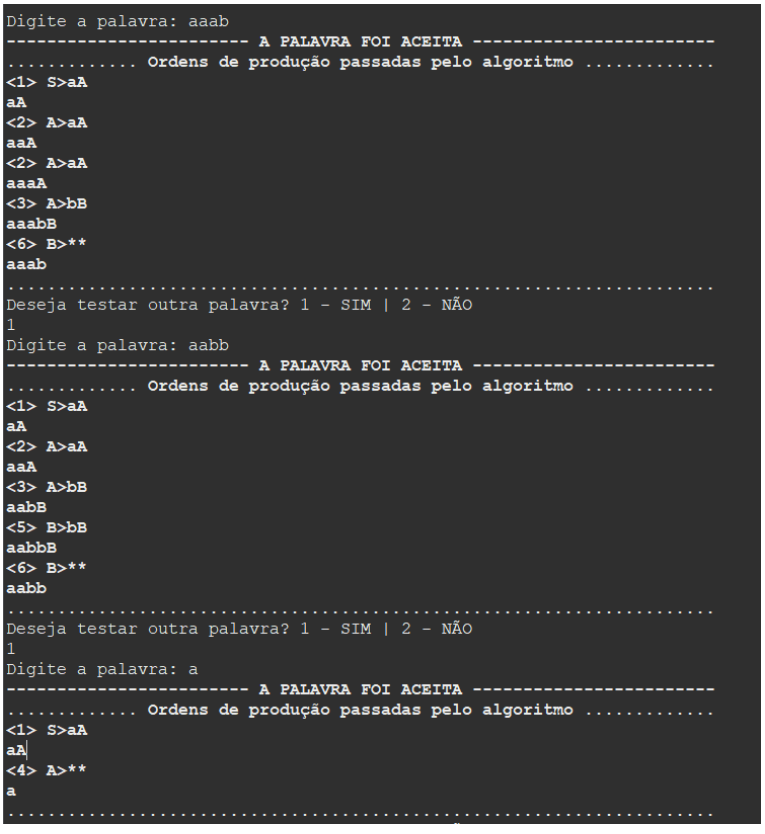

Figura 5

```
Deseja testar outra palavra? 1 - SIM | 2 - NÃO
1
Digite a palavra: abb
----------------------- A PALAVRA FOI ACEITA -----------------------
.............. Ordens de produção passadas pelo algoritmo ..............
<1> S>aA
aA
<3> A>bB
abB
<5> B>bB
abbB
<6> B>**
abb
.....................................................................
Deseja testar outra palavra? 1 - SIM | 2 - NÃO
1
Digite a palavra: aaabc
---------------------- A PALAVRA NÃO FOI ACEITA ----------------------
.....................................................................
Deseja testar outra palavra? 1 - SIM | 2 - NÃO
1
Digite a palavra: aba
---------------------- A PALAVRA NÃO FOI ACEITA ----------------------
.....................................................................
Deseja testar outra palavra? 1 - SIM | 2 - NÃO
1
Digite a palavra: ba
---------------------- A PALAVRA NÃO FOI ACEITA ----------------------
.....................................................................
Deseja testar outra palavra? 1 - SIM | 2 - NÃO
2
.............................. Obrigado ..............................
```

Figura 6

GLUE

```
---------------------- Gramática Regular LFA ----------------------
Digite 1 para GLUD e 2 para GLUE: 2
Digite a quantidade de variáveis (V): 3
Digite a 1ª variável: S
Digite a 2ª variável: A
Digite a 3ª variável: B
Digite a quantidade de alfabeto da linguagem (T) - terminal: 2
Digite o 1° caractere do alfabeto: a
Digite o 2° caractere do alfabeto: b
Digite o símbolo de inicio: S
Digite a quantidade de ordens de produção: 6
-------------------------------------------------------------------
--> Digite no padrão A>Bc
--> A e B pertencentes à V
--> c pertencente à T
--> para simbolizar brancos utilize *
-------------------------------------------------------------------
Digite a 1ª ordem de produção:
S>Bb
Digite a 2ª ordem de produção:
S>Aa
Digite a 3ª ordem de produção:
A>Aa
Digite a 4ª ordem de produção:
A>**
Digite a 5ª ordem de produção:
B>Bb
Digite a 6ª ordem de produção:
B>Aa
------------------------- Descrição Formal -------------------------
V = {S,A,B}
T = {a,b}
S = S
P = {S>Bb
     S>Aa
     A>Aa
     A>**
     B>Bb
     B>Aa}
-------------------------------------------------------------------
```

Figura 7

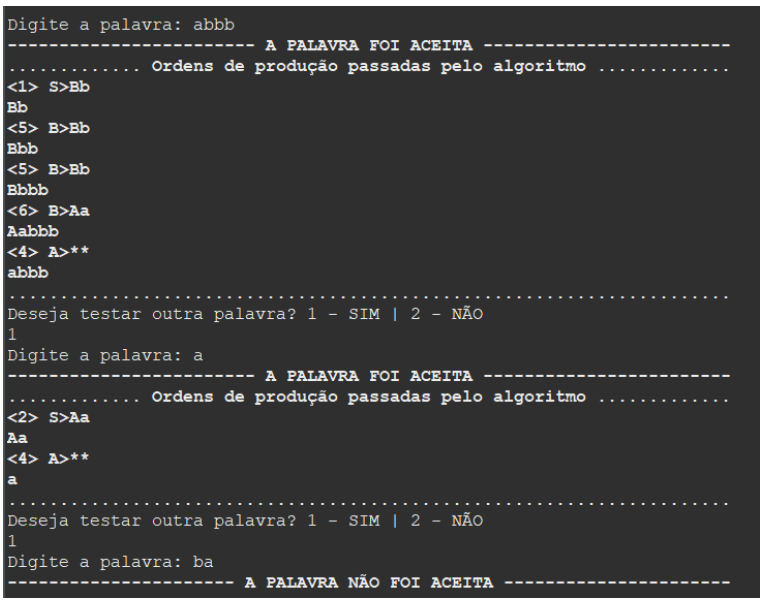

Figura 8

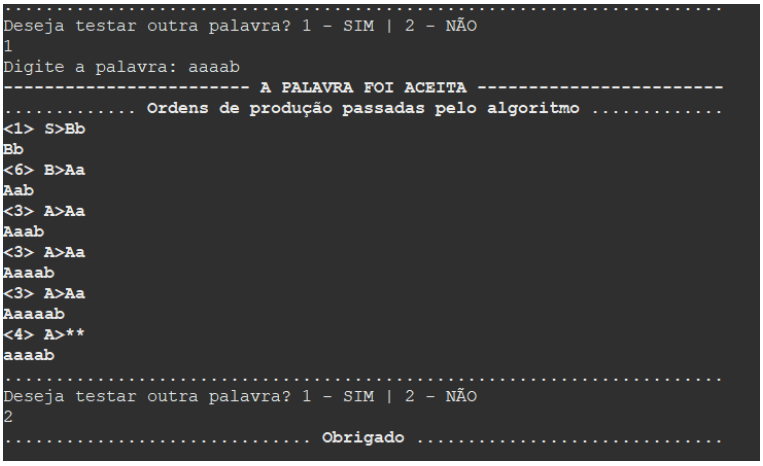

Figura 9

As execuções exemplificadas fornecem uma descrição formal e as ordens de produção relativas à sua linguagem, após, várias palavras são testadas, realizando o exposto no presente documento, demonstrando, quando a palavra é aceita e quando não. No primeiro caso, em relação a palavra ser aceita, imprimimos as ordens de produção na ordem que foram utilizadas e a formação da palavra em cada uma delas, no caso de não ser aceito apenas informamos isso ao usuário.

CONCLUSÃO

No presente documento evidenciamos, novamente, que a disciplina de LFA (Linguagens Formais, Autômatos e Computabilidade) pode ser mais prática que várias outras presentes na grade curricular de nosso curso, e, por isso, em atividades como a apresentada aqui, uma ligação interdisciplinar entre disciplinas de programação e LFA é possível. Ainda citando o trabalho de RAMOS [3], o autor evidencial em determinada parte do documento que aqueles que se comunicam, mesmo de forma natural e inconsciente, são dotados de uma capacidade cognitiva que pode, ao mesmo tempo, e em tempo real, escutar sons, decodifica-los, entendê-los e responde-los de maneira espontânea. Ele ainda diz que, mesmo portando toda esta capacidade, se ela não estiver sendo aplicada da melhor maneira possível, o uso da língua também não será da melhor maneira possível. Com isso podemos entender que, trazendo para o contexto tecnológico que vemos, as gramáticas, se não forem bem executadas, recebidas ou tratadas, podem resultar em uma linguagem incorreta, palavras externas a essa linguagem sendo aceitas ou outras variantes da melhor resolução.

A gramática aplicada à LFA é importante para a formação do discente, de modo com que entenda mais uma parte das formulações necessárias para se trabalhar com linguagens e suas vertentes, e, realizar aplicações práticas e funcionais em códigos relacionados ao seu funcionamento, somente garante mais interação e entendimento da disciplina/conteúdo. O presente documento visa expressar as principais estruturas utilizadas para a criação do código fonte e também contextualizar a sua necessidade e uso, objetivos, motivações, resultado e demais partes já explicitadas. Desse modo, novamente a abstração pode ser diminuída por meio de métodos definidos por MAZUR (2015) como metodologias ativas de ensino, aproximando, mais uma vez, estudantes da raiz dos problemas, aumentando a interação entre si e causando, provavelmente, um aprendizado maximizado. Em geral, o código apresentado desempenhou o papel e resultado esperado, tanto em funcionamento regular quanto em relação ao aprendizado ativo.

REFERÊNCIAS

[1] ALMEIDA, Napoleão Mendes de. **Gramática metódica da língua portuguesa (curso único e completo).** 13ª ed. São Paulo: Saraiva, 1961.

[2] CEGALLA, Domingos Paschoal. **Novíssima gramática da língua portuguesa**. 48ª ed. São Paulo: Companhia Editora Nacional, 2008.

[3] RAMOS, Gabriel da Silva. **A importância da gramática na vida humana: reflexões e ponderações.** 2022. 36 f. Monografia (Licenciatura em Letras) - Instituto de Ciências Humanas e Sociais, Universidade Federal de Mato Grosso, Barra do Garças, 2022.

[4] MAZUR, Eric. **Peer Instruction -A Revolução da Aprendizagem Ativa**.Editora Penso. Ano 2015.