Processamento de Linguagens

Trabalho Prático 01

Linguagens Regulares

Realizado por:

Tiago Bastos 12851

Marco Simão 23132

João Teixeira 23013

Licenciatura em Engenharia de Sistemas Informáticos

2023/24

Índice

[Introdução 1](#_Toc164106691)

[Recursos 2](#_Toc164106692)

[Objetivos 3](#_Toc164106693)

[Enunciado 4](#_Toc164106694)

[Autómatos Finitos Deterministas (AFD) 5](#_Toc164106695)

[ Resultados Obtidos [AFD]: 7](#_Toc164106696)

[Expressão Regular para AFND 9](#_Toc164106697)

[Resultados Obtidos [Expressão Regular para AFND]: 12](#_Toc164106698)

[Conversão de AFND para AFD 13](#_Toc164106699)

[ Resultados Obtidos [AFND-AFD]: 16](#_Toc164106700)

[Dificuldades 18](#_Toc164106701)

[Conclusão 19](#_Toc164106702)

[Bibliografia 20](#_Toc164106703)

Índice de Figuras

[Figura 1 - Figura Ilustrativa Python 2](#_Toc164106704)

[Figura 2 - Figura Ilustrativa VSCode 2](#_Toc164106705)

[Figura 3 - Função para reconhecer uma palavra inserida pelo utilizador no ficheiro AFD[A] 5](#_Toc164106706)

[Figura 4 - Função para imprimir o ficheiro .gv e criar o arquivo .json [A] 5](#_Toc164106707)

[Figura 5 – Código para verificar o argumento inserido [A] 6](#_Toc164106708)

[Figura 6 - Função para criar o graphviz de um AFD [A] 6](#_Toc164106709)

[Figura 7 - Output do comando -rec [A] 7](#_Toc164106710)

[Figura 8 - Output do comando -graphviz [A] 7](#_Toc164106711)

[Figura 9 - Ficheiro .gv [A] 7](#_Toc164106712)

[Figura 10 – Grafo do output do comando graphviz [A] 8](#_Toc164106713)

[Figura 11 - Função que inicia a conversão de ER para AFND [A] 10](#_Toc164106714)

[Figura 12 – Função invocada pela a função de converter um ER para AFND [A] 10](#_Toc164106715)

[Figura 13 – Funções invocadas pela função de converter um ER [A] 11](#_Toc164106716)

[Figura 14 - Ficheiro ER [B] 12](#_Toc164106717)

[Figura 15 – Comando para transformar um ficheiro ER em AFND [B] 12](#_Toc164106718)

[Figura 16 - Ficheiro AFND [B] 12](#_Toc164106719)

[Figura 17 - Função para ler o ficheiro .json inserido [C] 13](#_Toc164106720)

[Figura 18 - Função para guardar o ficheiro AFD em Json e o resultado do graphviz num ficheiro .gv [C] 13](#_Toc164106721)

[Figura 19 - Função para converter um ficheiro AFND.json em um AFD [C] 14](#_Toc164106722)

[Figura 20 - Função para criar o graphviz do AFND e guardar o output num ficheiro[C] 15](#_Toc164106723)

[Figura 21 - Código de verificação dos argumentos “graphviz” e o “output” [C] 15](#_Toc164106724)

[Figura 22 - Output do comando -graphviz [C] 16](#_Toc164106725)

[Figura 23 - Ficheiro AFND.gv [C] 16](#_Toc164106726)

[Figura 24 - Grafo criado ao usar o comando graphviz [C] 16](#_Toc164106727)

[Figura 25 – Execução do comando -output para converter o AFND em AFD [C] 16](#_Toc164106728)

[Figura 26 – Output da conversão de AFND para AFD [C] 17](#_Toc164106729)

[Figura 27 – Execução do comando graphviz para um AFD [C] 17](#_Toc164106730)

[Figura 28 – Output do comando graphviz [C] 17](#_Toc164106731)

[Figura 29 - Grafo criado ao usar o comando graphviz [C] 17](#_Toc164106732)

## Introdução

Este trabalho é desenvolvido no contexto da unidade curricular de Processamento de Linguagem Natural. O principal objetivo é projetar e implementar identificadores de linguagem regular utilizando a linguagem de programação Python.

Este relatório destaca não apenas a evolução do projeto, mas também os desafios enfrentados e as soluções adotadas ao longo do percurso. Além disso, ressalta a importância das contribuições individuais e coletivas no contexto do desenvolvimento do projeto.

## Recursos

Python é uma linguagem de programação de alto nível, conhecida pela sua legibilidade e eficiência. Foi criada por Guido van Rossum e lançada pela primeira vez em 1991. Python suporta múltiplos paradigmas de programação, incluindo programação orientada a objetos, imperativa e, em menor grau, funcional. É usada em diversas áreas, desde desenvolvimento web e automação até ciência de dados e inteligência artificial, graças à sua extensa biblioteca padrão e à vasta gama de frameworks e ferramentas de terceiros.



Figura 1 - Figura Ilustrativa Python

O Visual Studio Code, tambem conhecido como VSCode, é um editor de código-fonte desenvolvido pela Microsoft. Disponível para Windows, macOS e Linux, é gratuito e de código aberto. O VSCode suporta várias linguagens de programação e é amplamente utilizado para desenvolvimento de software devido à sua leveza, personalização extensiva por meio de extensões e integração com ferramentas como controle de versão Git.



Figura 2 - Figura Ilustrativa VSCode

## Objetivos

Os objetivos impostos pelo docente neste trabalho são os seguintes:

* Demonstrar a relevância do uso de expressões regulares;
* Definir expressões regulares para o reconhecimento de elementos simples;
* Implementar analisadores de expressões regulares baseados em autómatos finitos deterministas;
* Conhecer o processo de desenvolvimento de ferramentas reconhecedoras de expressões regulares;
* Implementar os autómatos finitos, adequando a representação conforme se trate de um autómato determinista, ou não determinista;
* Conhecer o processo de conversão de um autómato não determinista num determinista, especificando funções que realizem a implementação.

## Enunciado

As expressões regulares permitem representar de uma forma compacta uma linguagem.

O seu reconhecimento pode ser realizado recorrendo a um autómato finito.

Neste primeiro trabalho prático pretende-se ter por base a definição formal de uma expressão regular, e a partir dessa representação, suportar a execução do algoritmo de reconhecimento das palavras da linguagem gerada.

Sendo o principal objetivo deste trabalho prático, o estudo do algoritmo de reconhecimento de linguagens regulares, não devem ser utilizadas bibliotecas externas de suporte a expressões regulares.

## Autómatos Finitos Deterministas (AFD)

Implementar o algoritmo de reconhecimento de linguagens baseado num autómato finito determinista (AFD).

Considerando a representação de um AFD estudado nas aulas deve ser estruturada uma solução para facilitar a utilização desta estrutura de dados, nomeadamente as seguintes funcionalidades:

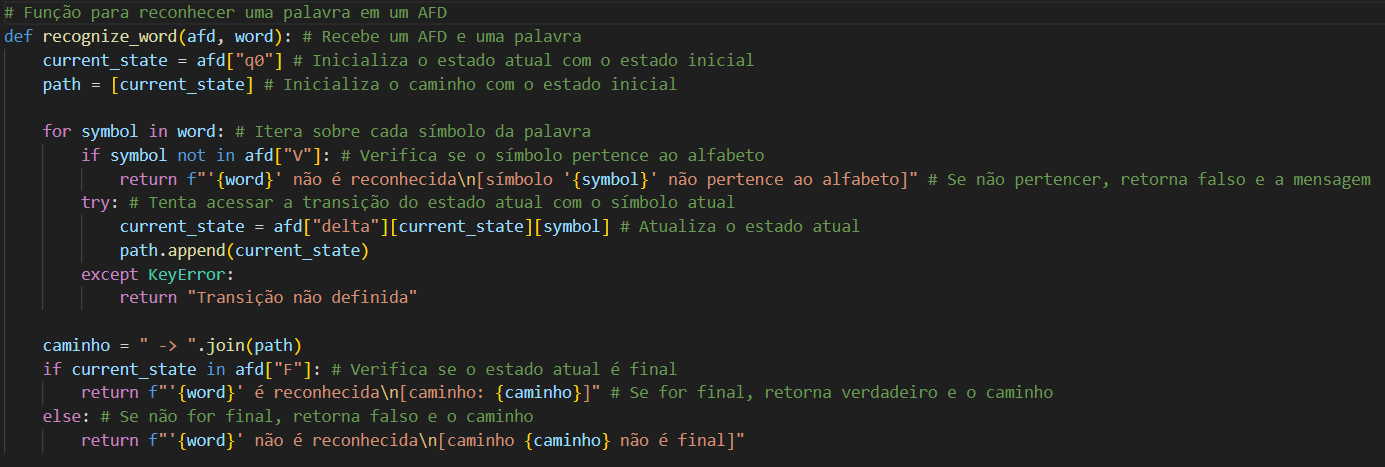


Figura 3 - Função para reconhecer uma palavra inserida pelo utilizador no ficheiro AFD[A]

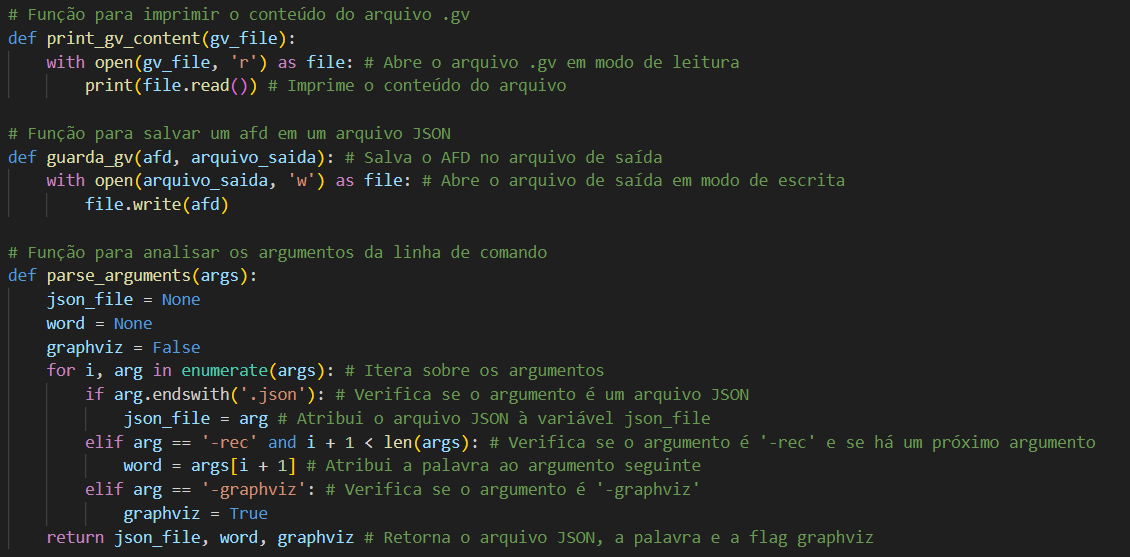
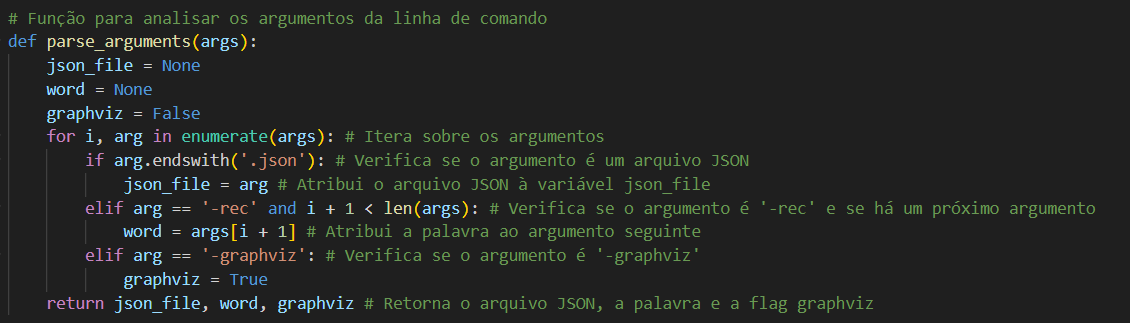


Figura 4 - Função para imprimir o ficheiro .gv e criar o arquivo .json [A]



Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 5 – Código para verificar o argumento inserido [A]

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 6 - Função para criar o graphviz de um AFD [A]

## Resultados Obtidos [AFD]:

Uma imagem com texto, Tipo de letra, software, captura de ecrã

Descrição gerada automaticamente

Figura 7 - Output do comando -rec [A]

Uma imagem com texto, software, Software de multimédia, captura de ecrã

Descrição gerada automaticamente

Figura 8 - Output do comando -graphviz [A]

Uma imagem com texto, captura de ecrã, software, multimédia

Descrição gerada automaticamente

Figura 9 - Ficheiro .gv [A]

Uma imagem com captura de ecrã, Software de multimédia, Software gráfico, Edição

Descrição gerada automaticamente

Figura 10 – Grafo do output do comando graphviz [A]

## Expressão Regular para AFND

Pretende-se implementar as regras de conversão de uma expressão regular para um autómato finito não determinista (AFND).

Uma imagem com texto, captura de ecrã, Tipo de letra, design

Descrição gerada automaticamente

Desenvolva um programa que comece por ler uma expressão regular especificada num ficheiro json de acordo com os exemplos acima, e gere um AFND equivalente (com o formato json estudado no ponto anterior). A entrada do programa deverá ser já o json correspondente à expressão regular.

Exemplos de utilização:



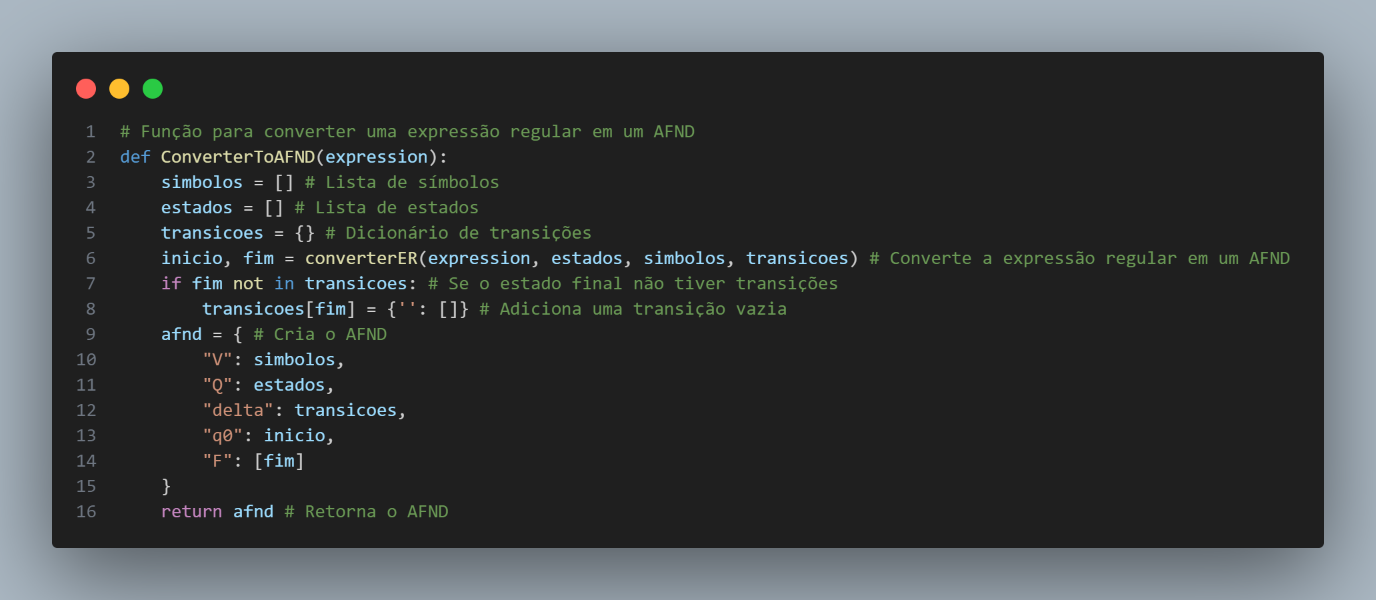


Figura 11 - Função que inicia a conversão de ER para AFND [A]



Figura 12 – Função invocada pela a função de converter um ER para AFND [A]



Figura 13 – Funções invocadas pela função de converter um ER [A]

## Resultados Obtidos [Expressão Regular para AFND]:



Figura 14 - Ficheiro ER [B]



Figura 15 – Comando para transformar um ficheiro ER em AFND [B]



Figura 16 - Ficheiro AFND [B]

## Conversão de AFND para AFD

Algoritmo que converte de AFND para AFD e que imprime e gere o output, tambem que imprime o digraph AFND.

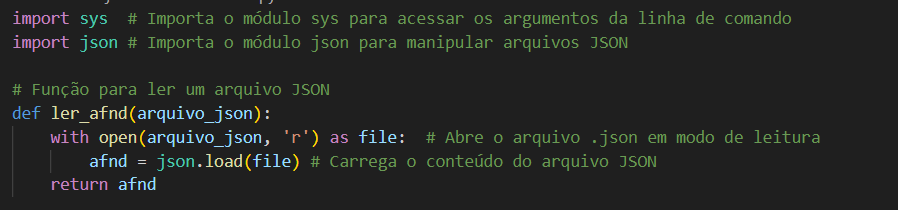


Figura 17 - Função para ler o ficheiro .json inserido [C]

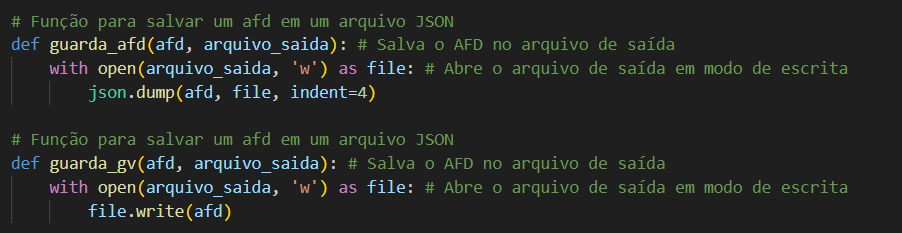


Figura 18 - Função para guardar o ficheiro AFD em Json e o resultado do graphviz num ficheiro .gv [C]



Figura 19 - Função para converter um ficheiro AFND.json em um AFD [C]

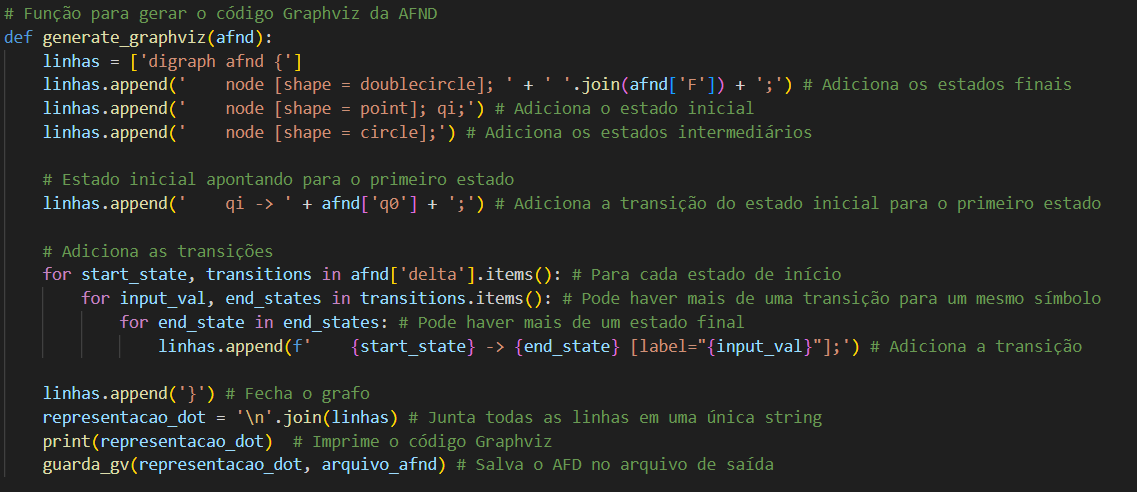


Figura 20 - Função para criar o graphviz do AFND e guardar o output num ficheiro[C]

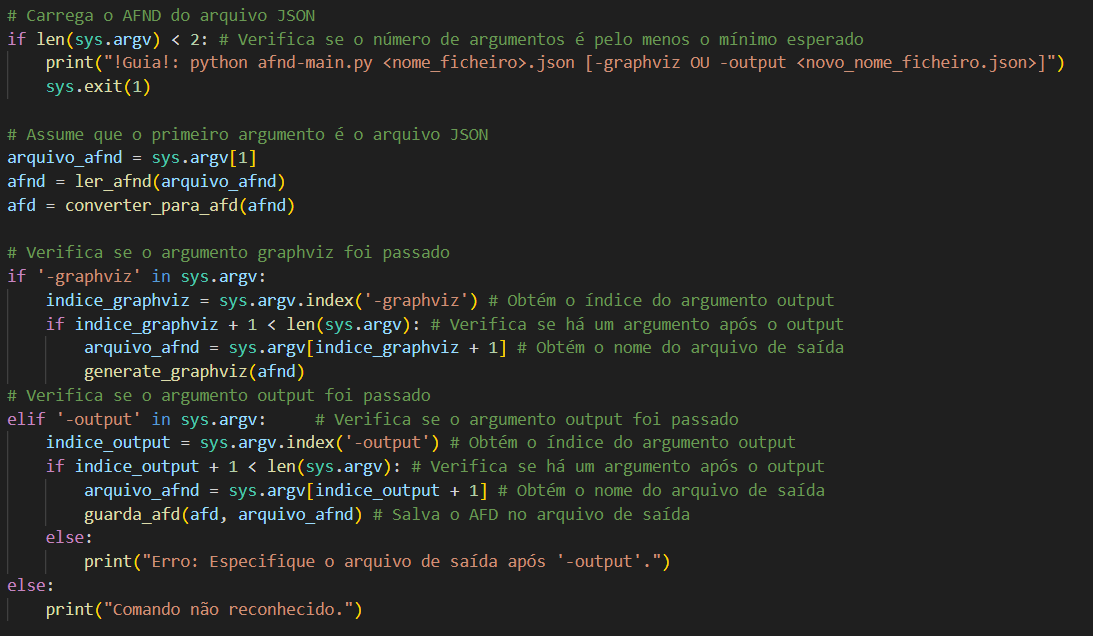


Figura 21 - Código de verificação dos argumentos “graphviz” e o “output” [C]

## Resultados Obtidos [AFND-AFD]:

Uma imagem com texto, captura de ecrã, Software de multimédia, software

Descrição gerada automaticamente

Figura 22 - Output do comando -graphviz [C]

Uma imagem com texto, captura de ecrã, software, Sistema operativo

Descrição gerada automaticamente

Figura 23 - Ficheiro AFND.gv [C]

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 24 - Grafo criado ao usar o comando graphviz [C]



Figura 25 – Execução do comando -output para converter o AFND em AFD [C]



Figura 26 – Output da conversão de AFND para AFD [C]

Uma imagem com texto, software, Software de multimédia, captura de ecrã

Descrição gerada automaticamente

Figura 27 – Execução do comando graphviz para um AFD [C]

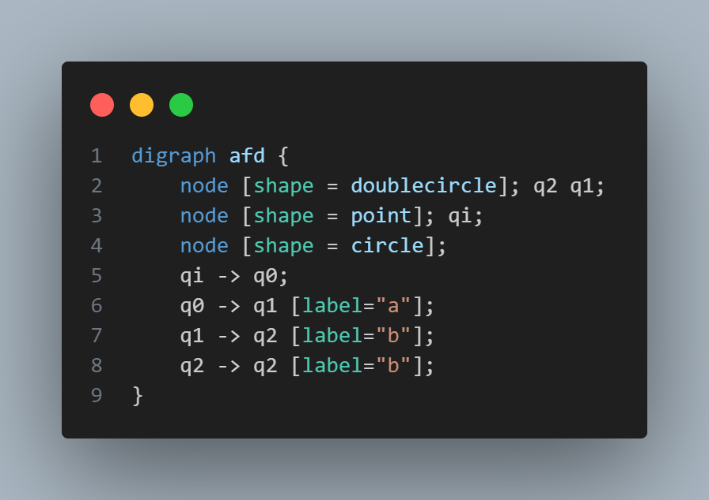


Figura 28 – Output do comando graphviz [C]

Uma imagem com captura de ecrã, texto, Software de multimédia, software

Descrição gerada automaticamente

Figura 29 - Grafo criado ao usar o comando graphviz [C]

## Dificuldades

Durante o desenvolvimento deste projeto na UC de Processamento de Linguagem, enfrentamos desafios significativos, especialmente na conversão de expressões regulares para a implementação desejada. A complexidade inerente às expressões regulares, juntamente com a necessidade de traduzi-las eficazmente para código em Python, apresentou-se como um obstáculo substancial inicialmente.

No entanto, as dificuldades foram sendo superadas à medida que nos aprofundávamos nos conteúdos adicionais fornecidos pelo professor. As aulas teóricas e práticas mostraram-se cruciais para entender melhor os conceitos envolvidos e as estratégias de aplicação desses conceitos na programação. Além disso, os recursos e exemplos compartilhados pelo professor ajudaram a esclarecer as dúvidas.

## Conclusão

Ao concluir este trabalho prático da UC de Processamento de Linguagens, foi possível atingir todos os objetivos propostos no enunciado, contribuindo significativamente para o nosso entendimento e habilidade em implementar reconhecedores de linguagens regulares usando Python. A implementação desde autômatos finitos deterministas (**AFD**) até a conversão de autômatos finitos não deterministas (**AFND**) para (**AFD**), proporcionou uma compreensão profunda sobre o funcionamento e a utilidade das expressões regulares em contextos computacionais.

Superamos desafios técnicos, como a conversão correta das expressões regulares em estruturas de autômatos e a interpretação dessas estruturas para validar cadeias de caracteres dentro da linguagem especificada. A prática de ler e escrever representações de autômatos em JSON, assim como gerar gráficos e mostrar o ficheiro utilizando a extensão Graphviz, foi particularmente enriquecedora, melhorando a nossa capacidade de trabalhar com representações de dados complexas e ferramentas de visualização.

## Bibliografia

<https://cyberzhg.github.io/toolbox/nfa2dfa>

<https://www.geeksforgeeks.org/converting-epsilon-nfa-to-dfa-using-python-and-graphviz/amp/>

<https://medium.com/pyladiesbh/regex-básico-em-python-31dcb7fac046>