



**UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
INE-DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
INE5421 - LINGUAGENS FORMAIS**

**PEDRO HENRIQUE TAGLIALENHA (22203674)
VITOR PRAXEDES CALEGARI (22200379)
ENRICO CALIOLO (23150562)**

Documentação do Gerador de Analisadores Léxicos

**FLORIANÓPOLIS
2025**

SUMÁRIO

1 INTRODUÇÃO.....	3
2 PRÉ-REQUISITOS.....	4
3 CONCEITOS FUNDAMENTAIS.....	5
4 INICIANDO A APLICAÇÃO.....	6
5 VISÃO GERAL DA INTERFACE.....	7
6 DEFININDO EXPRESSÕES REGULARES.....	8
6.1 Formato Básico.....	8
6.2 Operadores Suportados.....	8
6.3 Classes de Caracteres [...].	8
6.4 Agrupamento (...).	9
6.5 Caracteres de Escape \.	9
6.6 Épsilon (&).	10
6.7 Exemplos.....	11
7 UTILIZANDO OS MODOS DE OPERAÇÃO.....	13
7.1 Modo Manual (Thompson ou Followpos).....	13
7.2 Modo Teste Completo.....	14
8 INTERPRETANDO AS SAÍDAS.....	15
9 SALVANDO RESULTADOS.....	17

1 INTRODUÇÃO

Este software é um Gerador de Analisadores Léxicos que permite definir padrões de tokens usando expressões regulares, construir os autômatos finitos correspondentes (AFN e AFD) e, finalmente, utilizar o AFD otimizado para analisar um código fonte, gerando uma lista de tokens.

Ele oferece duas abordagens principais para a construção do AFD:

- **Método de Thompson:** Constrói AFNs individuais para cada ER, une-os, determina o AFN resultante para um AFD e, opcionalmente, minimiza este AFD.
- **Método da Árvore de Sintaxe (Followpos):** Constrói um AFD diretamente a partir de uma árvore sintática aumentada da expressão regular combinada, e opcionalmente minimiza este AFD.

2 PRÉ-REQUISITOS

- **Python 3.x:** A linguagem de programação utilizada.
- **Bibliotecas Python:**
 - customtkinter: Para a interface gráfica.
 - Pillow (PIL): Para manipulação de imagens.
 - graphviz: Para desenhar os grafos dos AFDs.
- **Graphviz (Software):** O software Graphviz (especificamente o comando dot) precisa estar instalado no seu sistema e acessível através do PATH do sistema para que a funcionalidade de desenhar AFDs funcione (Apenas necessário se o OS for windows, em linux apenas o pip install é o suficiente). Você pode baixá-lo em graphviz.org/download/. Ou em <https://community.chocolatey.org/packages/Graphviz>.

3 CONCEITOS FUNDAMENTAIS

- **Padrão (Expressão Regular (ER)):** Uma sequência de caracteres que define ou descreve um padrão de busca.
- **Token:** Símbolo abstrato que representa um tipo de unidade léxica (ex: palavra chave). Tem um padrão associado.
- **Lexema:** A sequência de caracteres do código fonte que corresponde ao padrão de um token (ex: if, minhaVariavel, 123, +).
- **Autômato Finito Não Determinístico (AFN):** Um modelo matemático que reconhece um conjunto de strings. Pode ter múltiplas transições para o mesmo símbolo ou transições épsilon.
- **Autômato Finito Determinístico (AFD):** Um tipo de AFN onde, para cada estado e símbolo de entrada, existe no máximo uma transição.
- **Tabela de Símbolos:** Uma estrutura de dados que armazena informações sobre os identificadores e outros símbolos encontrados no código.




4 INICIANDO A APLICAÇÃO

Navegue até a pasta raiz do projeto (AnalizadorLexico/) e execute:
`python main.py`

5 VISÃO GERAL DA INTERFACE


Tela Inicial

Ao iniciar, você verá a tela principal com as seguintes opções de modo:

-  **Modo Manual (Thompson):** Permite inserir manualmente ERs e código fonte, e executar passo a passo o processo de construção do lexer usando o algoritmo de Thompson.
-  **Modo Manual (Followpos):** Similar ao anterior, mas utiliza o método de construção direta do AFD via Followpos.
-  **Modo Teste Completo (Automático):** Permite selecionar casos de teste pré-definidos (de tests.py) e executar automaticamente todas as etapas de construção e análise, usando o método Thompson ou Followpos.

Modo Manual (Thompson / Followpos)

Esta tela é dividida em duas seções principais:

- **Painel de Controle e Definições (Esquerda):**
 - **1. Definições Regulares:** Caixa de texto para inserir suas ERs ou carregar de um arquivo.
 - **Botões de Processamento:**
 - Carregar Definições de Arquivo: Carrega ERs de um arquivo .txt ou .re.
 - A. Processar ERs: Inicia a primeira etapa (ER -> AFN individual ou ER -> Árvore/Followpos).
 - B. Unir Autômatos & Determinar (Thompson) / B. (AFD Direto da Etapa A) (Followpos - desabilitado, pois é integrado em A): Prossegue para a construção do AFD não minimizado.
 - C. Minimizar AFD: Minimiza o AFD gerado.
 -  Desenhar AFD Minimizado: Gera uma imagem do AFD final.
 - Salvar Tabela AFD Minimizada: Salva a tabela do AFD no formato do Anexo II e formatos legíveis.
 - **2. Texto Fonte para Análise:** Caixa de texto para inserir o código que você deseja analisar.
 - Analisar Texto Fonte (Gerar Tokens): Executa o analisador léxico no texto fonte.
 - Voltar à Tela Inicial.
- **Painel de Visualização (Direita, com Abas):**
 - Exibe os resultados de cada etapa do processo em abas separadas (detalhes abaixo).

Modo Teste Completo (Automático)

- **Painel de Controle (Esquerda):**
 - Seleção do método de construção (Thompson/Followpos).
 - Caixas de texto (desabilitadas) para exibir as ERs e o código fonte do teste selecionado.
 - Lista de botões para executar os casos de teste pré-definidos.
 - Voltar à Tela Inicial.
- **Painel de Visualização (Direita, com Abas):**
 - Similar ao Modo Manual, exibe os resultados das etapas.

6 DEFININDO EXPRESSÕES REGULARES

As ERs devem ser definidas no formato especificado no Anexo I do enunciado, que é:

NOME_DO_TOKEN: ExpressaoRegular

- NOME_DO_TOKEN: Um nome identificador para o tipo de token (ex: ID, NUM, IF, WHILE).
- ExpressaoRegular: A expressão regular que define o padrão para este token.

6.1 Formato Básico

Cada definição deve estar em uma nova linha. Linhas começando com # são comentários e são ignoradas.

```
# Exemplo de definições
ID: [a-zA-Z_][a-zA-Z0-9_]*
NUM: [0-9]+(\.[0-9]+)?
IF: if
WS: [\t\n]+ %ignore
```

6.2 Operadores Suportados

- **Concatenação (Implícita ou Explícita .):**
 - ab significa 'a' seguido de 'b'.
 - O sistema insere o operador de concatenação (.) automaticamente durante o pré-processamento onde necessário (ex: ab se torna a.b, a(bc) se torna a.(b.c)).
- **Alternativa (OU |):**
 - a|b significa 'a' OU 'b'.
- **Fecho de Kleene (Zero ou mais *):**
 - a* significa zero ou mais ocorrências de 'a'.
- **Fecho Positivo (Um ou mais +):**
 - a+ significa uma ou mais ocorrências de 'a'. O sistema trata a+ como aa*.
- **Opcional (Zero ou um ?):**
 - a? significa zero ou uma ocorrência de 'a'. O sistema trata a? como (a|&).

6.3 Classes de Caracteres [...]

Usadas para definir um conjunto de caracteres possíveis.

- **Literais:** [abc] é equivalente a (a|b|c).
- **Intervalos (Ranges):**
 - [a-z]: Qualquer letra minúscula de 'a' a 'z'.
 - [A-Z]: Qualquer letra maiúscula de 'A' a 'Z'.
 - [0-9]: Qualquer dígito de '0' a '9'.
 - Combinações: [a-zA-Z0-9_] inclui letras minúsculas, maiúsculas, dígitos e underscore.
 - **Importante:** Os intervalos a-z, A-Z, 0-9 funcionam como esperado. Outros tipos de intervalo (ex: [-.]) podem não ser interpretados como um intervalo de caracteres ASCII contínuo, mas sim como os literais dentro da classe. A

lógica em `expand_char_class` trata o hífen como literal se não estiver entre dois caracteres que formam um intervalo alfabético ou numérico válido.

- **Caracteres Especiais Dentro de Classes:**

- A maioria dos metacaracteres de ER (como `*`, `+`, `?`, `.`, `(`, `)`) perdem seu significado especial dentro de `[...]` e são tratados como literais.
- **Exceções Notáveis:**
 - `\` (barra invertida): Ainda usada para escape (ex: `[\]` para um literal de barra invertida, `[\[` para um literal de colchete de abertura).
 - `-` (hífen): Usado para definir intervalos (ex: `a-z`). Para usá-lo como literal, coloque-o no início ou no fim da classe (`[-abc]` ou `[abc-]`) ou escape-o (`[a\c]`). A implementação atual parece adicionar literais se a condição de range não for satisfeita.
 - `]` (colchete de fechamento): Para incluí-lo como literal, escape-o (`[ab\c]`) ou coloque-o como o primeiro caractere após `[` (ex: `[]abc]`, embora isso possa ser confuso, escapar é mais seguro).
- O código em `expand_char_class` irá escapar automaticamente caracteres que são metacaracteres fora de classes (como `*`, `+`) se eles forem adicionados como parte de uma expansão de classe. Exemplo: `[a*]` se torna `[a*]`.

6.4 Agrupamento (...)

Parênteses são usados para:

1. Agrupar subexpressões para aplicar operadores a elas como um todo.
Ex: `(ab)+` significa uma ou mais ocorrências da sequência "ab".
2. Controlar a precedência dos operadores.
Ex: `a(b|c)` é 'a' seguido de 'b' ou 'c', enquanto `ab|c` é 'ab' OU 'c'.

A precedência dos operadores é (do maior para o menor):

1. `*`, `+`, `?` (Fechos e opcional)
2. `.` (Concatenação)
3. `|` (Alternativa)

6.5 Caracteres de Escape \

Use a barra invertida para tratar um metacaractere como um literal.

- `\.` : Corresponde ao caractere literal `'.'`
- `*` : Corresponde ao caractere literal `'*'`
- `\+` : Corresponde ao caractere literal `'+'`
- `\?` : Corresponde ao caractere literal `'?'`
- `\|` : Corresponde ao caractere literal `'|'`
- `\(` : Corresponde ao caractere literal `'('`
- `\)` : Corresponde ao caractere literal `')'`
- `\[` : Corresponde ao caractere literal `'['`
- `\]` : Corresponde ao caractere literal `']'`
- `\\` : Corresponde ao caractere literal `'\'`

Prioridade das Definições

A ordem em que você define os tokens no arquivo de entrada é crucial. Quando o analisador léxico encontra uma sequência de caracteres que pode ser correspondida por múltiplas ERs, ele escolherá a ER que foi definida primeiro no arquivo. Isso é importante para casos como palavras-chave vs. identificadores:

```
IF: if      # Definido primeiro
          ELSE: else
ID: [a-zA-Z]+ # Definido depois
```

Neste caso, "if" será reconhecido como o token IF, não como um ID. Se ID fosse definido antes, "if" seria incorretamente tokenizado como ID.

Palavras Reservadas

O sistema possui uma heurística para identificar palavras reservadas:

Se o NOME_DO_TOKEN estiver em MAIÚSCULAS e a ExpressaoRegular correspondente for a forma em minúsculas desse nome, ele é tratado como uma palavra reservada.

Ex: IF: if, WHILE: while.

Palavras reservadas têm prioridade sobre padrões mais genéricos (como ID) que poderiam também corresponder a elas, independentemente da ordem de definição, devido à lógica no `Lexer.tokenize` que verifica `last_match_lexeme.lower()` in `self.reserved_words`.

Ignorando Padrões (%ignore)

Você pode instruir o lexer a reconhecer um padrão mas não gerar um token para ele (ou seja, ignorá-lo) adicionando `%ignore` ao final da linha de definição. Isso é comumente usado para espaços em branco e comentários.

```
WS: [ ]+ %ignore
```

6.6 Épsilon (&)

O caractere `&` é usado para representar a string vazia (épsilon) em suas expressões, principalmente quando se usa o operador `?` ou `*` implicitamente, ou para expressar união com a string vazia.

- `a?` é internamente tratado como `(a|&)`
- `a*` é internamente tratado como `(a+|&)` (ou de forma mais complexa com loops épsilon)

Você pode definir um token que seja explicitamente épsilon:

```
EPSILON_TOKEN: &
```

No entanto, o lexer atual, se um token épsilon não for `%ignore`, pode sinalizar um erro de "zero-length match". Usualmente, épsilon é um conceito interno para a

construção do autômato, não um token explicitamente emitido, a menos que seja para fins muito específicos e ignorados.

6.7 Exemplos

ID: [a-zA-Z_][a-zA-Z0-9_]*

NUM: [0-9]+(\.[0-9]+)?

WS: []+ %ignore

- ID: Começa com letra ou _, seguido por zero ou mais letras, dígitos ou _.
- NUM: Um ou mais dígitos, opcionalmente seguidos por um . e mais um ou mais dígitos (para números decimais).
- WS: Um ou mais espaços, ignorados.

IF: if

ID: [a-zA-Z_][a-zA-Z0-9_]*

PLUS: [+]

LPAREN: [(]

- IF: Palavra reservada.
- PLUS: [+]: O caractere +. Dentro de [], + é literal. Fora de [], seria \+.
- LPAREN: [(]: O caractere (. Dentro de [], (é literal. Fora de [], seria \(.

A_STAR: a*

B_PLUS: b+

C_OPT: c?

D_SEQ: d[ef]*g?h+

- A_STAR: Zero ou mais 'a'.
- B_PLUS: Um ou mais 'b'.
- C_OPT: 'c' opcional.
- D_SEQ: 'd', seguido por zero ou mais 'e' ou 'f', seguido por 'g' opcional, seguido por um ou mais 'h'.

ALT_GROUP: (ab|cd)+

EMAIL_LIKE: [a-zA-Z0-9._%+~]+@[a-zA-Z0-9.-]+\.[a-zA-Z][a-zA-Z][a-zA-Z]*

- ALT_GROUP: Uma ou mais ocorrências de "ab" OU "cd".
- EMAIL_LIKE: Um padrão simplificado para emails. Note \. para o literal ponto. [.] também funcionaria para um literal ponto.

LIT_STAR: *

LIT_PLUS: \+

LIT_DOT: \.

- Demonstra como corresponder aos caracteres que normalmente são operadores.

7 UTILIZANDO OS MODOS DE OPERAÇÃO

7.1 Modo Manual (Thompson ou Followpos)

1. **Inicie a Aplicação** e selecione o modo manual desejado na tela inicial.
2. **Defina as Expressões Regulares:**
 - Digite suas definições na caixa de texto "1. Definições Regulares:".
 - OU clique em "Carregar Definições de Arquivo" para carregar de um arquivo .txt ou .re.
3. **Insira o Código Fonte:**
 - Digite o código que você deseja analisar na caixa de texto "2. Texto Fonte para Análise:".
4. **Etapa A: Processar ERs**
 - Clique no botão "A. Processar ERs".
 - **Saída (Thompson):** Os AFNs individuais para cada ER (ou erros) serão exibidos na aba "ER → NFA Ind. / Árvore+Followpos".
 - **Saída (Followpos):** A Árvore Sintática Aumentada combinada, a Tabela Followpos e um "NFA Combinado Conceitual" serão exibidos na aba "ER → NFA Ind. / Árvore+Followpos". O AFD Direto (Não Minimizado) será exibido na aba "NFA Combinado (União ϵ) / AFD Direto (Não-Minim.)".
 - A "Tabela de Símbolos (Definições & Dinâmica)" mostrará as definições estáticas (padrões e palavras reservadas identificadas).
5. **Etapa B: Unir Autômatos & Determinar (Apenas para Thompson)**
 - Se estiver no modo Thompson e a Etapa A foi bem-sucedida, clique em "B. Unir NFAs & Determinar".
 - **Saída:** O AFN combinado global e o AFD Não Minimizado resultante da determinização serão exibidos na aba "NFA Combinado (União ϵ) / AFD Direto (Não-Minim.)".
 - (Para Followpos, esta etapa é conceitualmente parte da Etapa A, e o botão B estará desabilitado).
6. **Etapa C: Minimizar AFD**
 - Se um AFD Não Minimizado foi gerado (da Etapa B para Thompson, ou Etapa A para Followpos), clique em "C. Minimizar AFD".
 - **Saída:** O AFD Não Minimizado (entrada para minimização) e o AFD Minimizado (final) serão exibidos na aba "AFD Minimizado (Final)".
7. **Desenhar AFD Minimizado (Opcional)**
 - Clique em "🎨 Desenhar AFD Minimizado".
 - **Saída:** Uma imagem do AFD será exibida na aba "Desenho AFD Minimizado" e salva no subdiretório imagens/.
8. **Analisar Texto Fonte**
 - Clique em "Analisar Texto Fonte (Gerar Tokens)".
 - **Saída:**
 - A lista de tokens reconhecidos (ou erros) será exibida na aba "Saída do Analisador Léxico (Tokens)".
 - A Tabela de Símbolos Dinâmica (com os lexemas encontrados) será atualizada na aba "Tabela de Símbolos (Definições & Dinâmica)".
9. **Salvar Tabela AFD (Opcional)**

- Clique em "Salvar Tabela AFD Minimizada" para salvar a tabela de transição do AFD minimizado no formato do Anexo II e também em formatos mais legíveis.

7.2 Modo Teste Completo

1. **Inicie a Aplicação** e selecione "Modo Teste Completo (Automático)".
2. **Escolha o Método de Construção:** Selecione "Thompson" ou "Followpos" usando os botões de rádio.
3. **Selecione um Teste:** Clique em um dos botões "Executar: [Nome do Teste]".
4. **Resultados:**
 - As ERs e o código fonte do teste selecionado serão exibidos nas caixas de texto à esquerda.
 - Todas as etapas (A, B, C, Desenho, Análise) serão executadas automaticamente.
 - Os resultados de cada etapa preencherão as respectivas abas no painel de visualização à direita.
 - Uma mensagem indicará o início e a conclusão (ou erro) do teste.

8 INTERPRETANDO AS SAÍDAS

As abas no painel de visualização mostram:

- **Aba: "ER → NFA Ind. / Árvore+Followpos"**
 - **Modo Thompson:** Detalhes textuais de cada AFN construído para uma ER individual, incluindo estados, estado inicial, estado de aceitação e transições (ϵ para épsilon).
 - **Modo Followpos:**
 - A representação textual da Árvore Sintática Aumentada combinada para todas as ERs.
 - A Tabela Followpos, mostrando para cada posição (folha na árvore) o conjunto de posições que podem segui-la.
 - Um "NFA Combinado Conceitual": Uma representação de NFA baseada na união das ERs originais (sem os marcadores de fim), útil para fins pedagógicos e comparação.
- **Aba: "NFA Combinado (União ϵ) / AFD Direto (Não-Minim.)"**
 - **Modo Thompson:** Detalhes textuais do AFN global (após unir todos os AFNs individuais com um novo estado inicial e transições ϵ) e a tabela de transição do AFD Não Minimizado obtido após a determinização (algoritmo de subconjuntos).
 - **Modo Followpos:** A tabela de transição do AFD Direto (Não Minimizado) construído a partir da tabela Followpos.
- **Aba: "AFD Minimizado (Final)"**
 - Mostra a tabela de transição do AFD Não Minimizado (como entrada para o algoritmo de minimização).
 - Mostra a tabela de transição do AFD Minimizado final, que será usado pelo lexer.
 - Ambas tabelas incluem: número de estados, estado inicial, estados de aceitação (com o nome do token que aceitam) e o alfabeto.
- **Aba: "Desenho AFD Minimizado"**
 - Exibe uma imagem gráfica do AFD Minimizado.
 - Estados são círculos. Estados de aceitação são círculos duplos com o nome do token.
 - Setas indicam transições, rotuladas com os símbolos de entrada.
 - Uma seta sem origem aponta para o estado inicial.
- **Aba: "Tabela de Símbolos (Definições & Dinâmica)"**
 - **Parte Estática (após Etapa A):**
 - Lista das definições de padrões de tokens (ERs) na ordem de prioridade.
 - Lista das palavras reservadas identificadas.
 - Lista dos padrões a serem ignorados (%ignore).
 - **Parte Dinâmica (após Análise Léxica):**
 - Uma tabela mostrando os lexemas encontrados no código fonte, seu tipo de token e um índice. Formato: Índice | Lexema | Tipo.
- **Aba: "Saída do Analisador Léxico (Tokens)"**
 - Lista os tokens reconhecidos no texto fonte. Formato:

- <lexema, TIPO_TOKEN> (para palavras reservadas, operadores, etc.)
- <lexema, ID> (para identificadores, o lexema é mostrado)
- <lexema, NUM> (Valor: numero_convertido) (para números, o lexema e seu valor convertido são mostrados)
- <caractere_erro, ERRO!> (para caracteres não reconhecidos)

9 SALVANDO RESULTADOS

No Modo Manual, após gerar o AFD Minimizado (Etapa C), você pode:

- Clicar em "Salvar Tabela AFD Minimizada".
- Isso permitirá que você salve:
 1. Um arquivo (.dfa.txt) contendo a tabela de transição do AFD no formato do Anexo II (para avaliação automática, se aplicável).
 2. Um arquivo (_min_readable.txt) com a tabela de transição do AFD minimizado em formato legível por humanos.
 3. Um arquivo (_unmin_readable.txt) com a tabela de transição do AFD não minimizado em formato legível por humanos.
- A imagem do AFD também é salva automaticamente no diretório imagens/ quando você clica em "Desenhar AFD Minimizado".