Teoria de Linguagens - Trabalho Prático 1

Carlos Magno Geraldo Barbosa

November 9, 2020

1 Introdução

O presente trabalho tem como intuito apresentar a implementação de um Autômato Finito Determinístico (AFD), para o expressão regular (ER) apresentada em 1. Para entendimento deste trabalho apresentamos juntamente com os resultados encontrados explicações condensadas sobre o capitulo de linguagens regulares e propriedades de linguagens regulares [1] vistos em sala de aula.

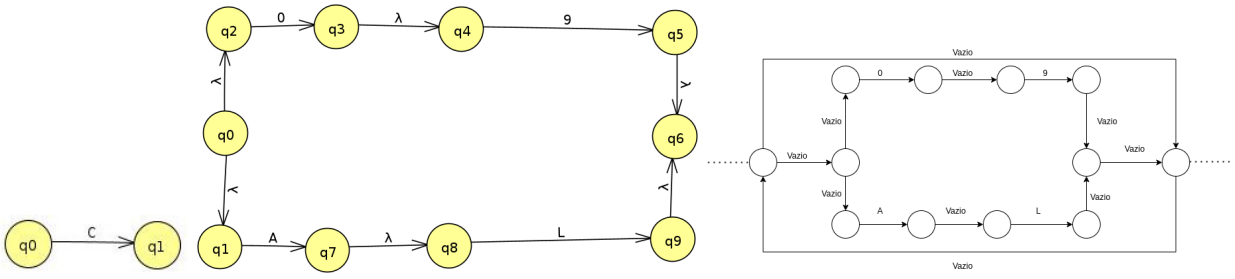
$$C1(09 + AL)^*3 \tag{1}$$

A **expressão regular** é um formalismo denotacional e gerador, pois se pode inferir como construir as palavras de uma linguagem. Uma expressão regular é definida a partir de conjuntos(linguagens) básicos e operações de concatenação e de união. Toda linguagem regular pode ser descrita por uma expressão denominada expressão regular.

Autômato é um formalismo reconhecedor, sendo basicamente um sistema de estados finitos. Sendo que este pode assumir um número finito e predefinido de estados. Um formalismo reconhecedor pode ser determinístico, não determinístico, e com movimentos vazios.

Autômato finito(determinístico) é uma máquina constituída por fita, que contém as informações a serem processadas. Unidade de Controle, que reflete o estado corrente da máquina, possui unidade de leitura e se desloca da esquerda para a direita acessando uma célula da fita de cada vez.

O processo de transformação de ER para automato finito não determinístico com movimentos vazios(AFN-ϵ) foi realizado de acordo com os passos apresentados na figura 1:



(a) ER C (b) ER (09+AL) (c) ER(09+AL)*

Figure 1: Geração do AFN com movimentos vazios

Nesse processo cada termo da expressão regular é transformado em um componente do (AFN-ϵ), resultando no autômato apresentado em 2.

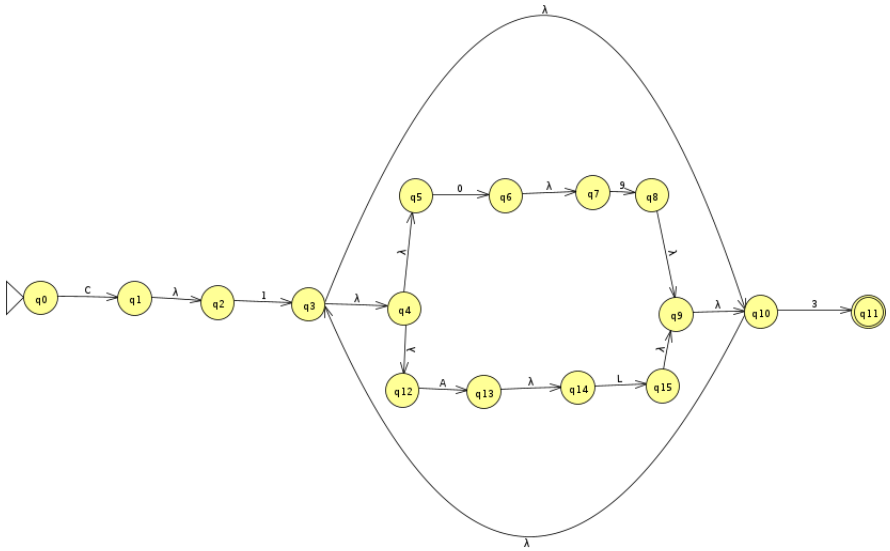


Figure 2: AFN-ϵ; gerado a partir da ER

A partir do AFN-ϵ é gerado um AFN sem movimentos vazios, apresentado na figura 3.

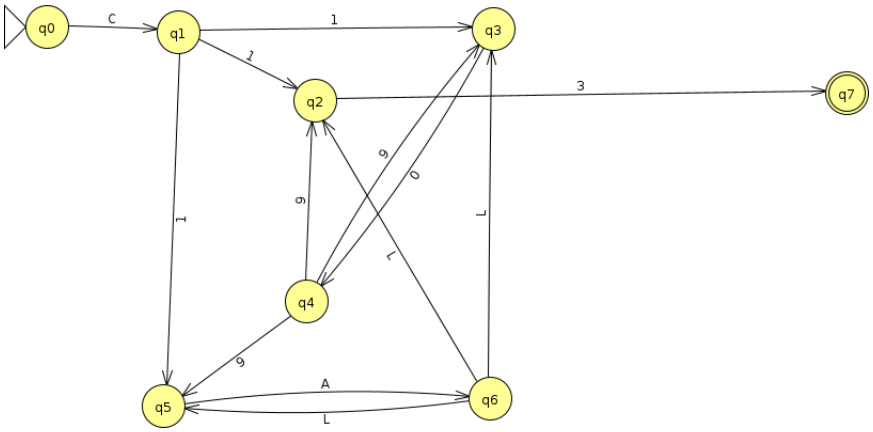


Figure 3: AFN sem movimentos vazios

O AFN é convertido para um autômato finito determinístico (AFD), apresentado na figura 4 e 5.

Estados	C	1	0	9	A	L	3	Renomeado
q0	q1	-	-	-	-	-	-	q0
q1	-	<q2, q3, q5>	-	-	-	-	-	q1
<q2, q3, q5>	-	-	q4	-	q6	-	qf	q2
q4	-	-	-	<q2, q3, q5>	-	-	-	q3
q6	-	-	-	-	-	<q2, q3, q5>	-	q4
qf	-	-	-	-	-	-	-	q5

Figure 4: Transição AFN para AFD

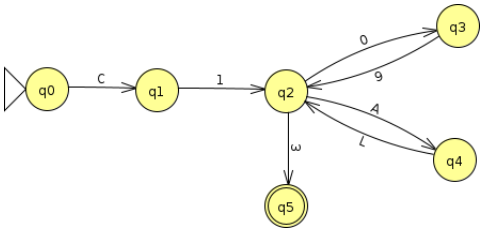


Figure 5: AFD gerado a partir do AFN sem movimentos vazios.

O presente AFD encontrado é mínimo, portanto, o mesmo já se encontra na sua forma minimizada não apresentando estados inúteis.

2 Implementação

O AFD proposto foi implementado na linguagem de programação python utilizando principalmente a estrutura de dados dicionário e lista. Dicionário e um estrutura de chave valor, similar a uma *hashmap*, no qual a partir de determinada chave é possível recuperar um valor armazenado. A fita contendo a palavra a ser verificada pelo automato é simulada através de uma estrutura de lista, no qual cada elemento da palavra é checado de maneira unitária.

O dicionário utilizado para manter os estados das transições validas e apresentado do modo da figura 6.

```
{'estado': {'termo_aceito': 'proximo_estado'}}
```

Figure 6: Estrutura dicionário

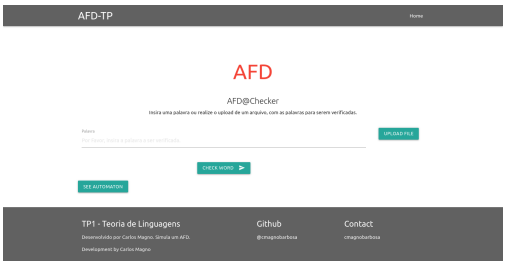
```
{'q0': {'C': 'q1'}}
```

Figure 7: Exemplo de Transição

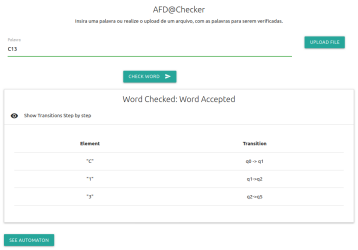
A figura 7, pode ser apresentada como lendo ‘C‘ no estado ‘q0‘ realize uma transição para o estado ‘q1‘.

3 Imagens da Interface

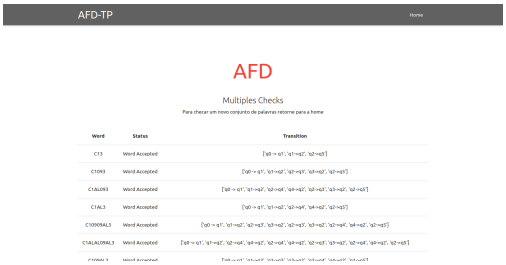
Foi desenvolvido uma interface WEB utilizando Flask e materialize css.



(a) Imagem 1



(b) Imagem 2



(c) Verificando palavras em arquivo.

Figure 8: Interface WEB

4 Manual

Segue em anexo um manual que apresenta um guia de como realizar a instalação e execução da implementação.

Palavra base aceita *C13*. Juntamente com o este trabalho segue um arquivo com palavras aceitas(palavras_aceitas.txt) e palavras negadas(palavras_negadas.txt).

References

[1] Paulo Blauth. Linguagens formais e autômatos. *Artmed Editora SA*, 2010.