

André Felipe Mireski

## **Prática Aula 02**

### **Autômatos Finitos e Determinísticos**

Relatório técnico de atividade prática solicitado pelo professor Rogério Aparecido Gonçalves na disciplina de Teoria da Computação do Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Universidade Tecnológica Federal do Paraná – UTFPR

Departamento Acadêmico de Computação – DACOM

Bacharelado em Ciência da Computação – BCC

Campo Mourão

Abril / 2025

# Resumo

Atividade prática com o intuito de fixar o conteúdo da aula sobre Autômatos Finitos e Determinísticos. Para tanto, foram realizados 7 exercícios, onde em cada um foi criado um AFD para reconhecer uma dada linguagem, primeiro, montando-o no JFLAP e depois criando um código python para testá-los usando a lib `automata-lib`.

**Palavras-chave:** Teoria da Computação. Autômatos Finitos e Determinísticos

# Sumário

1	Exercícios . . . . .	4
1.1	AFD $M_4$ que reconhece $L_4 = \{ba^nba \mid n \geq 0\}$ . . . . .	4
1.2	AFD $M_5$ que reconhece $L_5 = \{x \in \{a, b\}^* \mid  x  \bmod 3 = 0\}$ . . . . .	6
1.3	AFD $M_6$ que reconhece $L_6 = \{w \in \{a, b\}^* \mid ( w _a +  w _b) \bmod 2 = 0\}$ . . . . .	9
1.4	AFD $M_7$ que reconhece $L_7 = \{a^mb^n \mid m, n \geq 0 \wedge (m + n) \bmod 2 = 0\}$ . . . . .	11
1.5	AFD $M_8$ que reconhece $L_8 = \{x \in \Sigma^* \mid x \bmod 2 = 0\}$ . . . . .	13
1.6	AFD $M_9$ que reconhece $L_9 = \{x \in \Sigma^* \mid x \bmod 5 = 0\}$ . . . . .	16
1.7	AFD $M_{10}$ que reconhece $L = \{w \in \{a, b\}^* \mid  w _a \bmod 2 = 1 \wedge$ $ w _b \bmod 3 = 0\}$ . . . . .	18
2	Referências . . . . .	20

# 1 Exercícios

Nessa seção se encontram as imagens dos autômatos feitas no JFLAP e os testes feitos em python usando a lib `automata-lib`.

O código, com os resultados, se encontra disponível no repositório do github<sup>1</sup>.

## 1.1 AFD $M_4$ que reconhece $L_4 = \{ba^nba \mid n \geq 0\}$

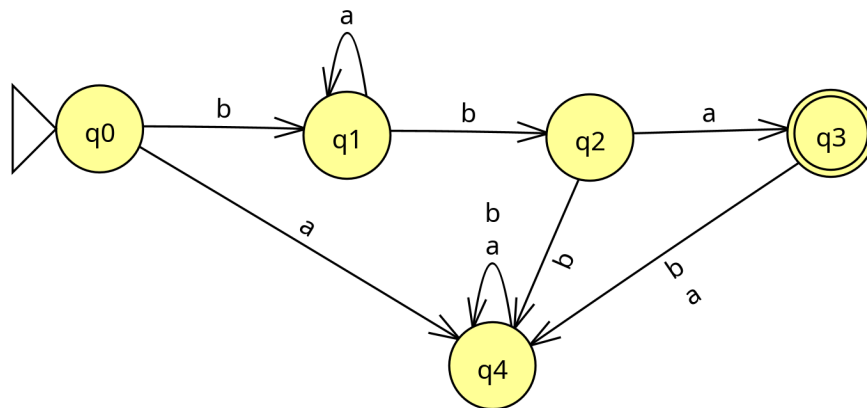


Figura 1 – AFD - Exercício 01

Listing 1 – AFD  $M_4$  em Python com a biblioteca `automata-lib`

```

1 from automata.fa.dfa import DFA
2
3 dfa01 = DFA(
4     states={"q0", "q1", "q2", "q3", "q4"},
5     input_symbols={"a", "b"},
6     transitions={
7         "q0": {"a": "q4", "b": "q1"},
8         "q1": {"a": "q1", "b": "q2"},
9         "q2": {"a": "q3", "b": "q4"},
10        "q3": {"a": "q4", "b": "q4"},
11        "q4": {"a": "q4", "b": "q4"},
12    },
13    initial_state="q0",

```

<sup>1</sup> [https://github.com/afmireski/BCC5003-atividades-praticas/tree/main/aula02\\_automatos\\_finitos\\_e\\_deterministicos](https://github.com/afmireski/BCC5003-atividades-praticas/tree/main/aula02_automatos_finitos_e_deterministicos)

```
14     final_states={"q3"},
15 )
16
17 test_inputs = [
18     "ba",
19     "baba",
20     "babab",
21     "bababab",
22     "babababa",
23     "abba",
24     "bbab",
25     "bab",
26     "baaa",
27     "bbbbaaa",
28     "bba",
29     "baaaba",
30     "baaaaaaaaaaba"
31 ]
32
33 for input_str in test_inputs:
34     if dfa01.accepts_input(input_str):
35         print(f"'{input_str}' - ACEITO.")
36     else:
37         print(f"'{input_str}' - REJEITADO.")
```

```

... 'ba' - REJEITADO.
    'baba' - ACEITO.
    'babab' - REJEITADO.
    'bababab' - REJEITADO.
    'babababa' - REJEITADO.
    'abba' - REJEITADO.
    'bbab' - REJEITADO.
    'bab' - REJEITADO.
    'baaa' - REJEITADO.
    'bbbbaaa' - REJEITADO.
    'bba' - ACEITO.
    'baaaba' - ACEITO.
    'baaaaaaaaaaaba' - ACEITO.

```

Figura 2 – AFD 01 - Resultados

1.2 AFD  $M_5$  que reconhece  $L_5 = \{x \in \{a, b\}^* \mid |x| \bmod 3 = 0\}$

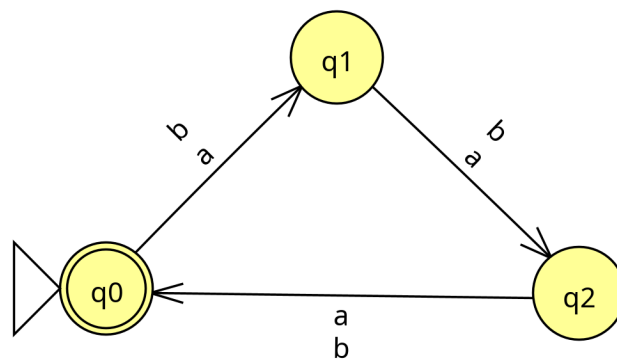


Figura 3 – AFD - Exercício 02

Listing 2 – AFD  $M_5$  em Python com a biblioteca automata-lib

```
1 from automata.fa.dfa import DFA
2
3 dfa02 = DFA(
4     states={"q0", "q1", "q2"},
5     input_symbols={"a", "b"},
6     transitions={
7         "q0": {"a": "q1", "b": "q1"},
8         "q1": {"a": "q2", "b": "q2"},
9         "q2": {"a": "q0", "b": "q0"},
10    },
11    initial_state="q0",
12    final_states={"q0"},
13 )
14
15 test_inputs = [
16     "aaa",
17     "bbb",
18     "ababab",
19     "babababbbb",
20     "aabaab",
21     "a",
22     "b",
23     "ab",
24     "ba",
25     "aabb",
26     "bbaa",
27     "bbaab",
28 ]
29
30 for input_str in test_inputs:
31     if dfa02.accepts_input(input_str):
32         print(f"'{input_str}' - ACEITO.")
33     else:
34         print(f"'{input_str}' - REJEITADO.")
```

```
... 'aaa' - ACEITO.  
    'bbb' - ACEITO.  
    'ababab' - ACEITO.  
    'babababbb' - ACEITO.  
    'aabaab' - ACEITO.  
    'a' - REJEITADO.  
    'b' - REJEITADO.  
    'ab' - REJEITADO.  
    'ba' - REJEITADO.  
    'aabb' - REJEITADO.  
    'bbaa' - REJEITADO.  
    'bbaab' - REJEITADO.
```

Figura 4 – AFD 02 - Resultados



### 1.3 AFD $M_6$ que reconhece $L_6 = \{w \in \{a, b\}^* \mid (|w|_a + |w|_b) \bmod 2 = 0\}$

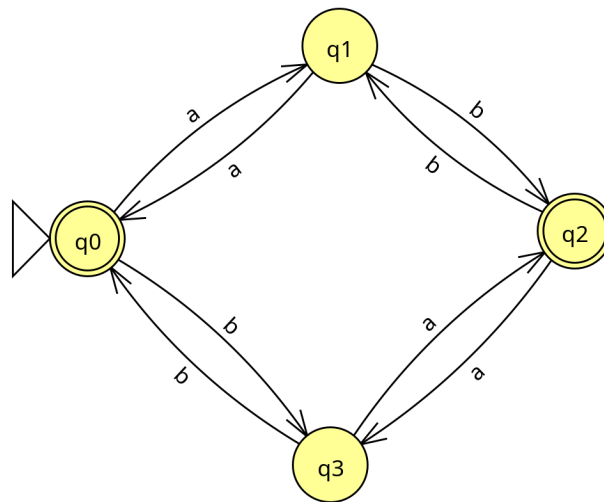


Figura 5 – AFD - Exercício 03

Listing 3 – AFD  $M_6$  em Python com a biblioteca automata-lib

```

1 from automata.fa.dfa import DFA
2
3 dfa03 = DFA(
4     states={"q0", "q1", "q2", "q3"},
5     input_symbols={"a", "b"},
6     transitions={
7         "q0": {"a": "q1", "b": "q3"},
8         "q1": {"a": "q0", "b": "q2"},
9         "q2": {"a": "q3", "b": "q1"},
10        "q3": {"a": "q2", "b": "q0"},
11    },
12    initial_state="q0",
13    final_states={"q0", "q2"},
14 )
15
16 test_inputs = [
17     "aaa", # R
18     "bbb", # R
19     "ababab", # A
20     "babababbbba", # A
21     "aabaab", # A
22     "a", # R

```

```
23     "b", # R
24     "ab", # A
25     "ba", # A
26     "aabb", # A
27     "bbaa", # A
28     "bbaab", # R
29 ]
30
31 for input_str in test_inputs:
32     if dfa03.accepts_input(input_str):
33         print(f"'{input_str}' - ACEITO.")
34     else:
35         print(f"'{input_str}' - REJEITADO.")
```

```
... 'aaa' - REJEITADO.
    'bbb' - REJEITADO.
    'ababab' - ACEITO.
    'babababbba' - ACEITO.
    'aabaab' - ACEITO.
    'a' - REJEITADO.
    'b' - REJEITADO.
    'ab' - ACEITO.
    'ba' - ACEITO.
    'aabb' - ACEITO.
    'bbaa' - ACEITO.
    'bbaab' - REJEITADO.
```

Figura 6 – AFD 03 - Resultados

### 1.4 AFD $M_7$ que reconhece $L_7 = \{a^m b^n \mid m, n \geq 0 \wedge (m + n) \bmod 2 = 0\}$

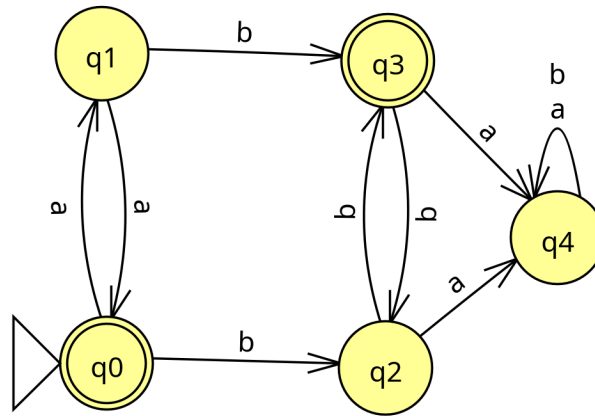


Figura 7 – AFD - Exercício 04

Listing 4 – AFD  $M_7$  em Python com a biblioteca automata-lib

```

1 from automata.fa.dfa import DFA
2
3 dfa04 = DFA(
4     states={"q0", "q1", "q2", "q3", "q4"},
5     input_symbols={"a", "b"},
6     transitions={
7         "q0": {"a": "q1", "b": "q2"},
8         "q1": {"a": "q0", "b": "q3"},
9         "q2": {"a": "q4", "b": "q3"},
10        "q3": {"a": "q4", "b": "q2"},
11        "q4": {"a": "q4", "b": "q4"},
12    },
13    initial_state="q0",
14    final_states={"q0", "q3"},
15 )
16
17 test_inputs = [
18     "aa", # A
19     "bb", # A
20     "aaab", # A
21     "bbba", # R
22     "ababab", # R

```

```
23     "a", # R
24     "b", # R
25     "ba", # R
26     "bbbb", # A
27     "aaaa", # A
28     "abab", # R
29     "aaaaabbb", # A
30 ]
31
32 for input_str in test_inputs:
33     if dfa04.accepts_input(input_str):
34         print(f"'{input_str}' - ACEITO.")
35     else:
36         print(f"'{input_str}' - REJEITADO.")
```

```
... 'aa' - ACEITO.
    'bb' - ACEITO.
    'aaab' - ACEITO.
    'bbba' - REJEITADO.
    'ababab' - REJEITADO.
    'a' - REJEITADO.
    'b' - REJEITADO.
    'ba' - REJEITADO.
    'bbbb' - ACEITO.
    'aaaa' - ACEITO.
    'abab' - REJEITADO.
    'aaaaabbb' - ACEITO.
```

Figura 8 – AFD 04 - Resultados

### 1.5 AFD $M_8$ que reconhece $L_8 = \{x \in \Sigma^* \mid x \bmod 2 = 0\}$

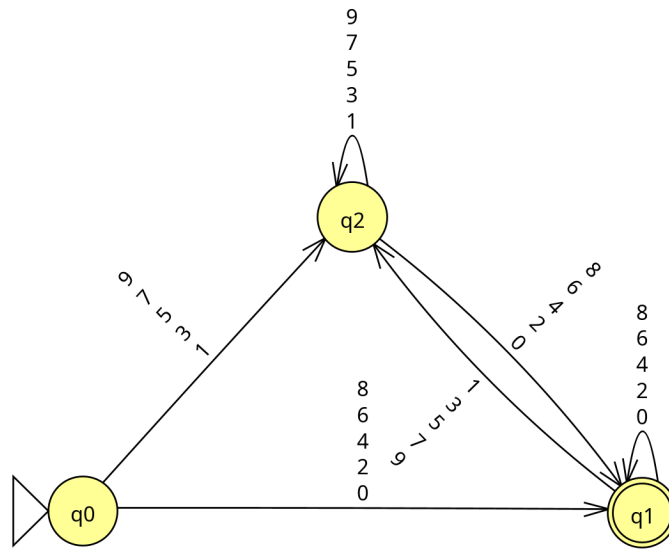


Figura 9 – AFD - Exercício 05

Listing 5 – AFD  $M_8$  em Python com a biblioteca automata-lib

```

1 from automata.fa.dfa import DFA
2
3 dfa05 = DFA(
4     states={"q0", "q1", "q2"},
5     input_symbols={"0", "1", "2", "3", "4", "5", "6", "7", "8", "9"},
6     transitions={
7         "q0": {"0": "q1", "2": "q1", "4": "q1", "6": "q1", "8": "q1", "1": "q2", "3": "q2", "5": "q2", "7": "q2", "9": "q2"},
8         "q1": {"0": "q1", "2": "q1", "4": "q1", "6": "q1", "8": "q1", "1": "q2", "3": "q2", "5": "q2", "7": "q2", "9": "q2"},
9         "q2": {"0": "q1", "2": "q1", "4": "q1", "6": "q1", "8": "q1", "1": "q2", "3": "q2", "5": "q2", "7": "q2", "9": "q2"},
10    },
11    initial_state="q0",
12    final_states={"q1"},
13 )
14
15

```

```
16 test_inputs = [  
17     "0", # A  
18     "024", # A  
19     "02134", # A  
20     "12345678", # A  
21     "975310", # A  
22     "012", # A  
23     "1", # R  
24     "1311", # R  
25     "021345", # R  
26     "1234567", # R  
27     "864201", # R  
28     "123", # R  
29 ]  
30  
31 for input_str in test_inputs:  
32     if dfa05.accepts_input(input_str):  
33         print(f"'{input_str}' - ACEITO.")  
34     else:  
35         print(f"'{input_str}' - REJEITADO.")
```

```
... '0' - ACEITO.  
    '024' - ACEITO.  
    '02134' - ACEITO.  
    '12345678' - ACEITO.  
    '975310' - ACEITO.  
    '012' - ACEITO.  
    '1' - REJEITADO.  
    '1311' - REJEITADO.  
    '021345' - REJEITADO.  
    '1234567' - REJEITADO.  
    '864201' - REJEITADO.  
    '123' - REJEITADO.
```

Figura 10 – AFD 05 - Resultados

## 1.6 AFD $M_9$ que reconhece $L_9 = \{x \in \Sigma^* \mid x \bmod 5 = 0\}$

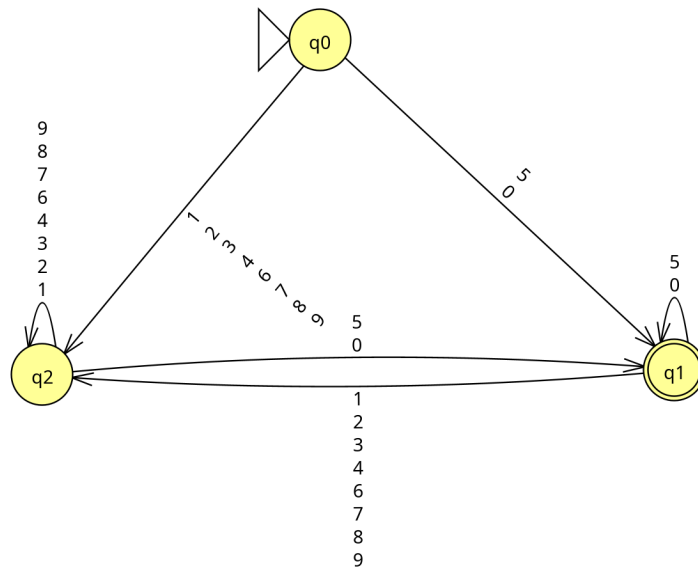


Figura 11 – AFD - Exercício 06

Listing 6 – AFD  $M_9$  em Python com a biblioteca automata-lib

```

1 from automata.fa.dfa import DFA
2
3 dfa06 = DFA(
4     states={"q0", "q1", "q2"},
5     input_symbols={"0", "1", "2", "3", "4", "5", "6", "7", "8", "9"},
6     transitions={
7         "q0": {"0": "q1", "5": "q1", "1": "q2", "2": "q2", "3": "q2", "4": "q2", "6": "q2", "7": "q2", "8": "q2", "9": "q2"},
8         "q1": {"0": "q1", "5": "q1", "1": "q2", "2": "q2", "3": "q2", "4": "q2", "6": "q2", "7": "q2", "8": "q2", "9": "q2"},
9         "q2": {"0": "q1", "5": "q1", "1": "q2", "2": "q2", "3": "q2", "4": "q2", "6": "q2", "7": "q2", "8": "q2", "9": "q2"},
10    },
11    initial_state="q0",
12    final_states={"q1"},
13 )
14

```



```
15
16 test_inputs = [
17     "0", # A
18     "05", # A
19     "5500055", # A
20     "0123467895", # A
21     "9876543210", # A
22     "000", # A
23     "1", # R
24     "0123456789", # R
25     "123456789", # R
26     "050501", # R
27     "551556", # R
28     "106", # R
29 ]
30
31 for input_str in test_inputs:
32     if dfa06.accepts_input(input_str):
33         print(f"'{input_str}' - ACEITO.")
34     else:
35         print(f"'{input_str}' - REJEITADO.")
```

```

... '0' - ACEITO.
    '05' - ACEITO.
    '5500055' - ACEITO.
    '0123467895' - ACEITO.
    '9876543210' - ACEITO.
    '000' - ACEITO.
    '1' - REJEITADO.
    '0123456789' - REJEITADO.
    '123456789' - REJEITADO.
    '050501' - REJEITADO.
    '551556' - REJEITADO.
    '106' - REJEITADO.

```

Figura 12 – AFD 06 - Resultados

1.7 AFD  $M_{10}$  que reconhece  $L = \{w \in \{a, b\}^* \mid |w|_a \bmod 2 = 1 \wedge |w|_b \bmod 3 = 0\}$

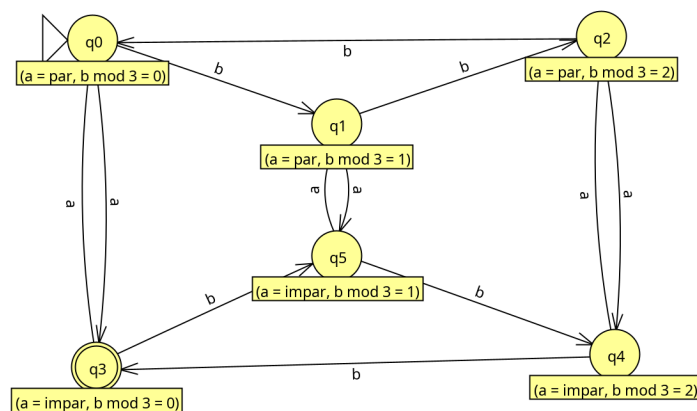
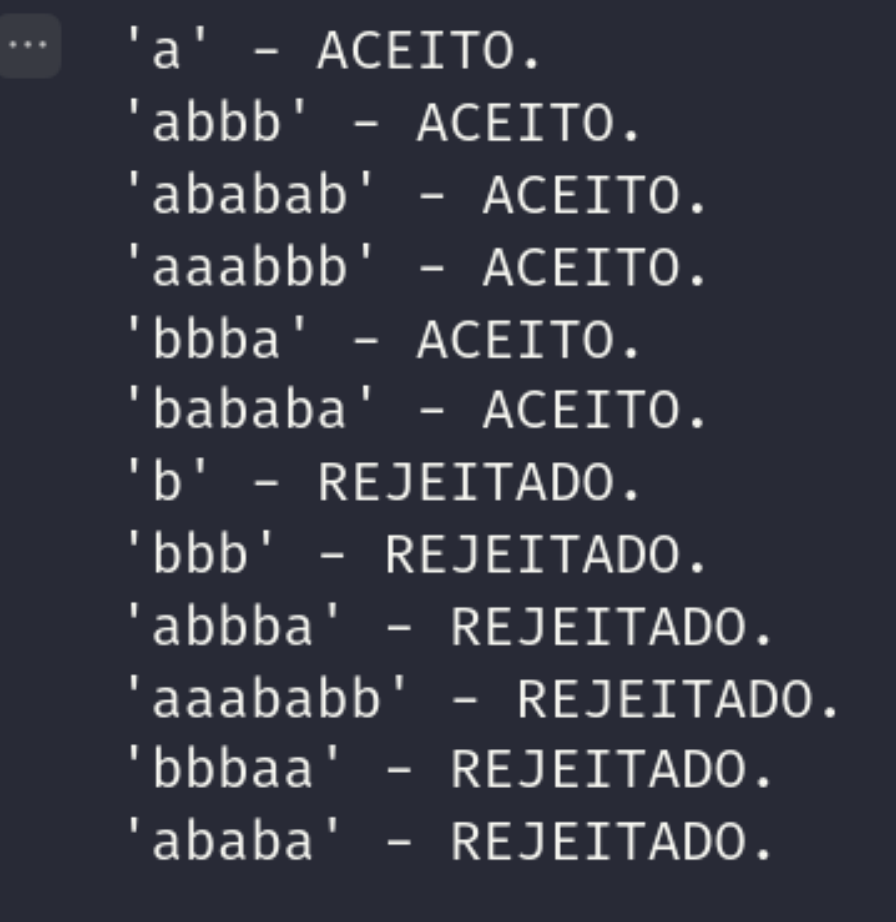


Figura 13 – AFD - Exercício 07

Listing 7 – AFD  $M_10$  em Python com a biblioteca automata-lib

```
1 from automata.fa.dfa import DFA
2
3 dfa07 = DFA(
4     states={'q0', 'q1', 'q2', 'q3', 'q4', 'q5'},
5     input_symbols={'a', 'b'},
6     transitions={
7         'q0': {'a': 'q3', 'b': 'q1'},
8         'q1': {'a': 'q5', 'b': 'q2'},
9         'q2': {'a': 'q4', 'b': 'q0'},
10        'q3': {'a': 'q0', 'b': 'q5'},
11        'q4': {'a': 'q2', 'b': 'q3'},
12        'q5': {'a': 'q1', 'b': 'q4'},
13    },
14    initial_state='q0',
15    final_states={'q3'}
16 )
17
18 test_inputs = [
19     "a", # A
20     "abbb", # A
21     "ababab", # A
22     "aaabbbb", # A
23     "bbba", # A
24     "bababa", # A
25     "b", # R
26     "bbb", # R
27     "abbba", # R
28     "aaababb", # R
29     "bbbbaa", # R
30     "ababa", # R
31 ]
32
33 for input_str in test_inputs:
34     if dfa07.accepts_input(input_str):
35         print(f"'{input_str}' - ACEITO.")
36     else:
37         print(f"'{input_str}' - REJEITADO.")
```

A terminal window with a dark background and light gray text. It shows a list of strings and their acceptance or rejection status. The strings are: 'a', 'abbb', 'ababab', 'aaabbb', 'bbba', 'bababa', 'b', 'bbb', 'abbba', 'aaababb', 'bbbaa', and 'ababa'. The first six are marked as 'ACEITO.' (Accepted) and the last six as 'REJEITADO.' (Rejected).

```
... 'a' - ACEITO.  
    'abbb' - ACEITO.  
    'ababab' - ACEITO.  
    'aaabbb' - ACEITO.  
    'bbba' - ACEITO.  
    'bababa' - ACEITO.  
    'b' - REJEITADO.  
    'bbb' - REJEITADO.  
    'abbba' - REJEITADO.  
    'aaababb' - REJEITADO.  
    'bbbaa' - REJEITADO.  
    'ababa' - REJEITADO.
```

Figura 14 – AFD 07 - Resultados

## 2 Referências