



UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS DE CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GUSTAVO DA FONSECA ROZA & CAUAN MULINARI

AUTOMATO COM PILHA DETERMINÍSTICO
TRABALHO 2

CHAPECÓ
Junho de 2025

SUMÁRIO

1	INTRODUÇÃO	2
2	DESCRIÇÃO DO AUTÔMATO:	3
2.1	PROBLEMATIZAÇÃO:	3
2.2	SOLUÇÃO:	3
2.3	REPRESENTAÇÃO EM DIAGRAMA DE ESTADOS:	4
2.4	OBSERVAÇÕES:	4
3	TESTES DO AUTÔMATO:	5
4	CONCLUSÃO	7
	REFERÊNCIAS	8

1 INTRODUÇÃO

Este trabalho realizado pelos estudantes Gustavo da Fonseca Roza (2211100074) & Cauan Mulinari (2211100057), tem como objetivo, descrever um autômato com pilha determinístico (*pushdown automaton*), que reconhece uma sequência de ácido desoxirribonucleico (DNA).

2 DESCRIÇÃO DO AUTÔMATO:

2.1 PROBLEMATIZAÇÃO:

A atividade proposta, em resumo, era: descrever um 6-upla de um autômato com pilha, capaz de processar um DNA, composto por 4 bases nitrogenadas, sendo elas: adenina (A), guanina (G), citosina (C) e timina (T). Tal que, para ser válido, o DNA precisava conter a mesma quantidade de adenina e timina (A's e T's).

Além de descrever o autômato, foi necessário o desenvolvimento, de uma representação computacional através de uma linguagem de programação da preferência dos alunos.

Neste desenvolvimento, alguns requisitos deveriam ser cumpridos:

- O PDA deveria obrigatoriamente ter comportamento determinístico;
- A string à ser processada pelo autômato deveria terminar em '#';
- O código deve implementar uma função de transição, simulando efetivamente um autômato, e não, apenas uma representação superficial com *if & else*;

2.2 SOLUÇÃO:

A linguagem de programação escolhida foi **C++**. A escolha desta linguagem teve como principal argumento, as bibliotecas disponíveis nessa linguagem e a familiaridade que a disciplina de Grafos nos proporcionou com essa linguagem.

O autômato (M) idealizado e implementado, consiste na seguinte 6-upla:

$M = (Q, \Sigma, \Gamma, \Delta, q_0, F)$ tal que,

$\Sigma = \{q_0, q_1, q_2, q_3, q_4, q_5\};$

$\Gamma = \{A, T, Z\};$

$q_0 = q_0;$

$F = \{q_1\}$

As transições consistindo em: estadoOrigem \rightarrow estadoDestino : (caractere da String, Caractere à ser desempilhado; caractere à ser empilhado)

$\Delta = \{ q_0 \rightarrow q_0 : (T,A; @),$

$q_0 \rightarrow q_0 : (A,T; @),$

$q_0 \rightarrow q_0 : (G,@; @),$

$q_0 \rightarrow q_0 : (C,@; @),$

$q_0 \rightarrow q_3 : (A,Z; Z),$

$q_0 \rightarrow q_2 : (A,A; A),$

$q_0 \rightarrow q_5 : (T,Z; Z),$

$q_0 \rightarrow q_4 : (T,T; T),$

$q_0 \rightarrow q_1 : (\#, Z; @)$, (estado de aceitação),
 $q_3 \rightarrow q_0 : (@, @; A)$,
 $q_2 \rightarrow q_0 : (@, @; A)$,
 $q_5 \rightarrow q_0 : (@, @; T)$,
 $q_4 \rightarrow q_0 : (@, @; T)\}$.

Obs.: O símbolo '@' representa a transição vazia (epsilon, ε).

2.3 REPRESENTAÇÃO EM DIAGRAMA DE ESTADOS:

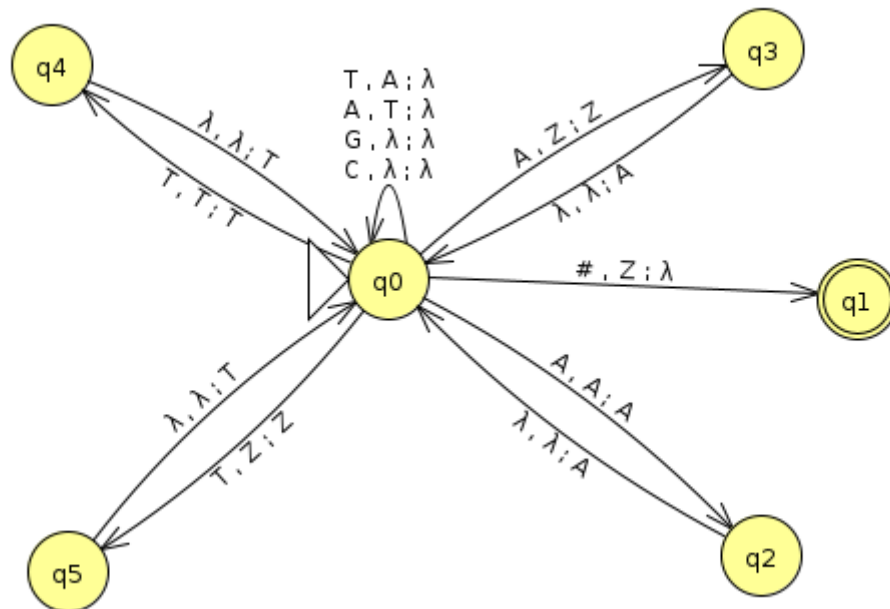


Figura 1 – Diagrama de estados do autômato representado no JFLAP

2.4 OBSERVAÇÕES:

Para a representação gráfica do autômato, foi utilizado o simulador **JFLAP** disponível em: (1).

Por convenção, o JFLAP utiliza (λ) para representar o caractere vazio, diferente da notação vista em aula (ε). Também por convenção, (Z) para demarcar o fundo da pilha, diferentemente do (\$) visto em aula.

3 TESTES DO AUTÔMATO:

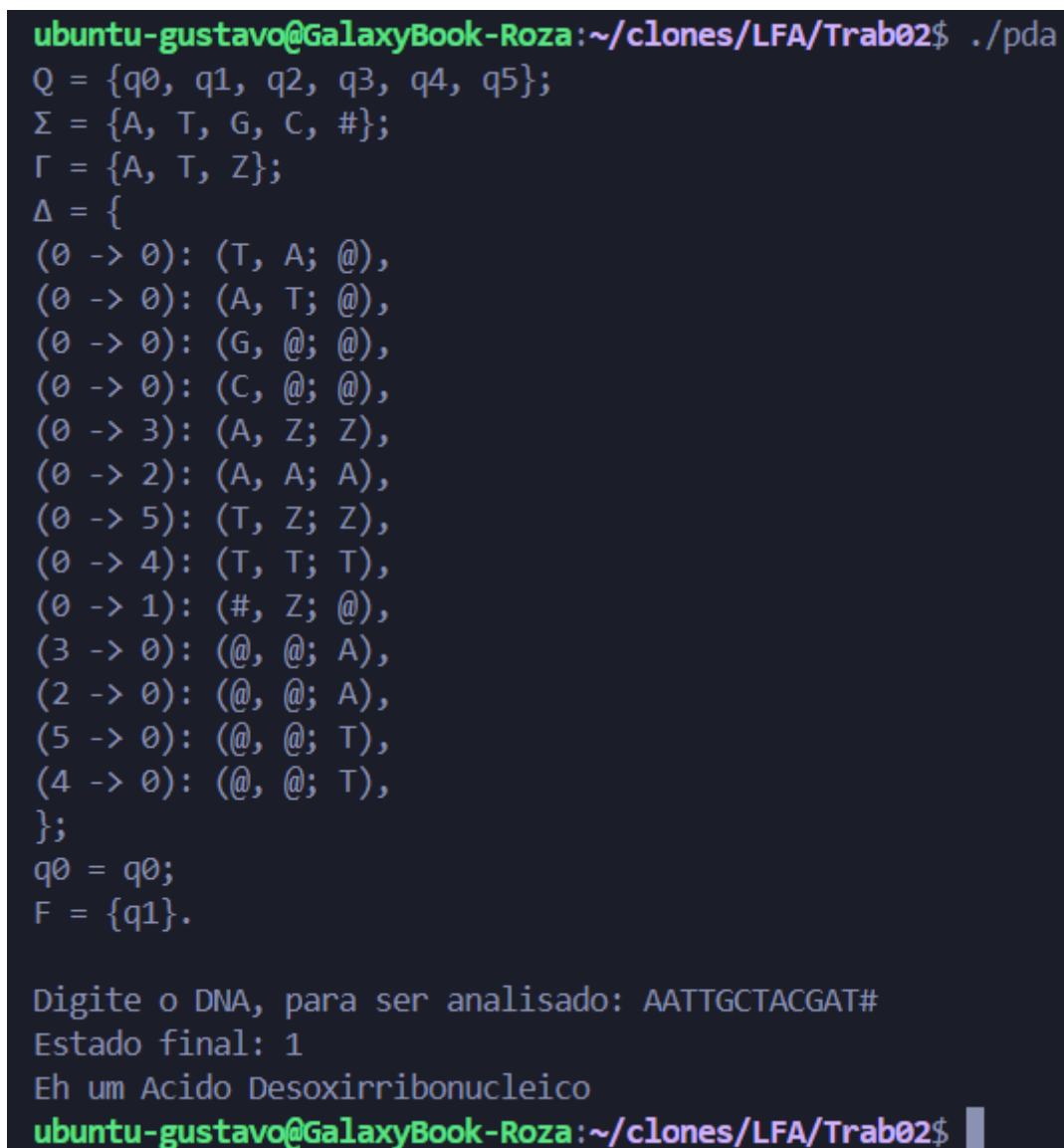
Para compilar o programa, basta ter o G++ instalado ao computador e compilar os arquivos com o seguinte comando:

```
g++ main.cpp AutomatoPilha.h AutomatoPilha.cpp -o pda
```

e depois executá-lo, com o comando:

```
./pda
```

O programa funciona em uma única execução, testando apenas uma *string* por vez, segue exemplos de execuções:



```
ubuntu-gustavo@GalaxyBook-Roza:~/clones/LFA/Trab02$ ./pda
Q = {q0, q1, q2, q3, q4, q5};
Σ = {A, T, G, C, #};
Γ = {A, T, Z};
Δ = {
  (0 -> 0): (T, A; @),
  (0 -> 0): (A, T; @),
  (0 -> 0): (G, @; @),
  (0 -> 0): (C, @; @),
  (0 -> 3): (A, Z; Z),
  (0 -> 2): (A, A; A),
  (0 -> 5): (T, Z; Z),
  (0 -> 4): (T, T; T),
  (0 -> 1): (#, Z; @),
  (3 -> 0): (@, @; A),
  (2 -> 0): (@, @; A),
  (5 -> 0): (@, @; T),
  (4 -> 0): (@, @; T),
};
q0 = q0;
F = {q1}.

Digite o DNA, para ser analisado: AATTGCTACGAT#
Estado final: 1
Eh um Acido Desoxirribonucleico
ubuntu-gustavo@GalaxyBook-Roza:~/clones/LFA/Trab02$
```

Figura 2 – Exemplo de execução com string válida.

```

ubuntu-gustavo@GalaxyBook-Roza:~/clones/LFA/Trab02$ ./pda
Q = {q0, q1, q2, q3, q4, q5};
Σ = {A, T, G, C, #};
Γ = {A, T, Z};
Δ = {
  (0 -> 0): (T, A; @),
  (0 -> 0): (A, T; @),
  (0 -> 0): (G, @; @),
  (0 -> 0): (C, @; @),
  (0 -> 3): (A, Z; Z),
  (0 -> 2): (A, A; A),
  (0 -> 5): (T, Z; Z),
  (0 -> 4): (T, T; T),
  (0 -> 1): (#, Z; @),
  (3 -> 0): (@, @; A),
  (2 -> 0): (@, @; A),
  (5 -> 0): (@, @; T),
  (4 -> 0): (@, @; T),
};
q0 = q0;
F = {q1}.

Digite o DNA, para ser analisado: AATCGTACCG#
Nao eh um Acido Desoxirribonucleico válido
ubuntu-gustavo@GalaxyBook-Roza:~/clones/LFA/Trab02$ █

```

Figura 3 – Exemplo de execução com string inválida.

4 CONCLUSÃO

Conclui-se, com os resultados obtidos através dos testes de execução, que incluíram tanto cadeias válidas quanto inválidas, comprovam o êxito da implementação. O autômato demonstrou ser capaz de analisar adequadamente as sequências de DNA fornecidas, confirmando o funcionamento de acordo com as especificações do trabalho.

Portanto, a realização deste trabalho foi uma forma eficaz de aplicar na prática a teoria da disciplina de Linguagens Formais e Autômatos. Desenvolver um autômato com pilha ajudou a consolidar o aprendizado e a concluir o projeto com sucesso.

REFERÊNCIAS

- 1 ORG, JFLAP. **Página Web do simulador de autômatos JFLAP**. 2018.
Disponível em: <<https://www.jflap.org/>>.