UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

CAMPUS CHAPECÓ

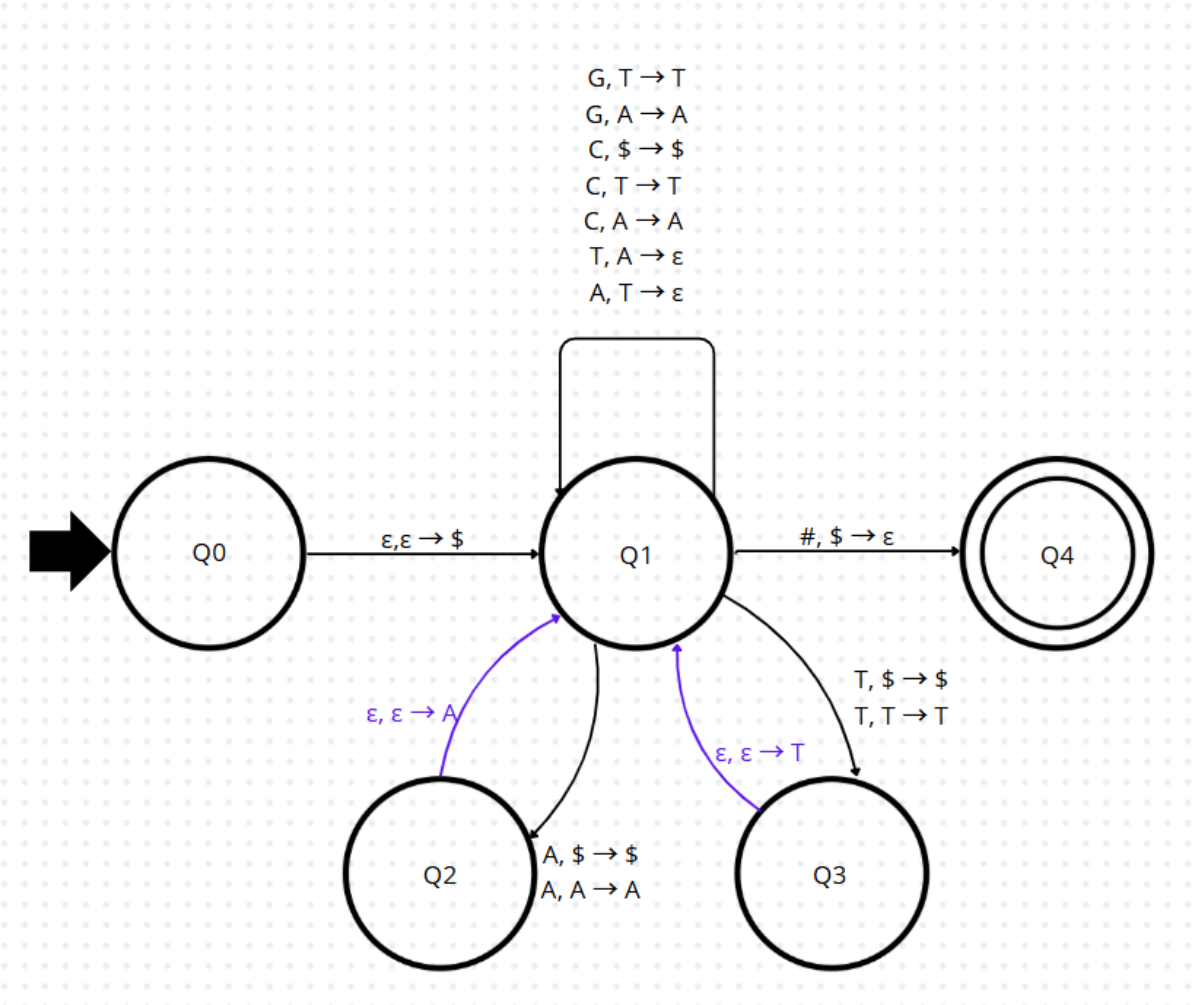
CURSO DE CIÊNCIA DA COMPUTAÇÃO
Linguagens Formais e Autômatos

Segundo Trabalho
Linguagens Formais e Autômatos

GUSTAVO BOTEZINI
LEONARDO DE OLIVEIRA KLITZKE

CHAPECÓ
2025

Autômato:



Q: {Q0, Q1, Q2, Q3, Q4}
Σ: {A, T, G, C, ε, #}
Γ: {A, T, $, ε }
q0: {Q0}
F: {Q4}
δ: {

```
('q0', EPSILON, EPSILON): ('q1', '$'),
('q1', 'G', '$'): ('q1', '$'),
('q1', 'G', 'T'): ('q1', 'T'),
('q1', 'G', 'A'): ('q1', 'A'),
('q1', 'C', '$'): ('q1', '$'),
('q1', 'C', 'T'): ('q1', 'T'),
('q1', 'C', 'A'): ('q1', 'A'),
('q1', 'A', 'T'): ('q1', EPSILON),
('q1', 'A', '$'): ('q2', '$'),
('q1', 'A', 'A'): ('q2', 'A'),
('q1', 'T', 'A'): ('q1', EPSILON),
('q1', 'T', '$'): ('q3', '$'),
('q1', 'T', 'T'): ('q3', 'T'),
('q1', '#', '$'): ('q4', EPSILON),
('q2', EPSILON, EPSILON): ('q1', 'A'),
('q3', EPSILON, EPSILON): ('q1', 'T'),
```

```
}
```

Funções:

def clean(self)

Limpa a pilha e coloca o autômato na posição inicial

def realizar_transicao(self, q, appendpilha, poppilha)

- Se poppilha não for **EPSILON**, remove o topo da pilha.
- Se appendpilha não for **EPSILON**, adicione um símbolo ao topo da pilha.
- Atualize o estado atual para que, se for válido.

def run(self, entrada)

- Reinicializa o autômato.
- Adiciona **EPSILON** à pilha.
- Para cada símbolo da entrada:
- Verifica se o símbolo pertence ao alfabeto.
- Executa todas as transições vazias possíveis antes da transição normal.
- Busca uma transição válida para o símbolo atual e o topo da pilha.
- Se existir, executa a transição.
- Se não houver transição possível, retorna **False**.
- Após processar toda a entrada, retorna **True** se o estado atual for final, senão **False**.

def fazer_transicao_vazia_se_existir(self)

- Busca uma transição do tipo (estado_atual, EPSILON, EPSILON).
- Se existir, executa a transição e retorna **True**.
- Caso contrário, retorna **False**.

Classe:

```
class AP:
    def __init__(self, Q, Sigma, gamma, delta, q0, F):
        self._Q = Q
        self._Sigma = Sigma
        self._gama = gama
        self._delta = delta
        self._q0 = q0
        self._F = F
        self.qA = q0
        self._stack = deque()
```

Exemplos:

```
entrada: AAACGCCTCATTAAAGTGGTTT#
esperado: True, obtido: True

entrada: AAACGCCTCATTAAAGTGGTTT
esperado: False, obtido: False

entrada: AAACGCCTCATTAAAGTGGTTT#A
esperado: False, obtido: False

entrada: AAACGCCTCATTAAAGTGGTTT#G
esperado: False, obtido: False

entrada: AAACGCCTCATTAAAGTGGTTT#C
esperado: False, obtido: False
```