[Aula 16] Propriedades e reconhecimento das LLC

Prof. João F. Mari
joaof.mari@ufv.br

BIBLIOGRAFIA

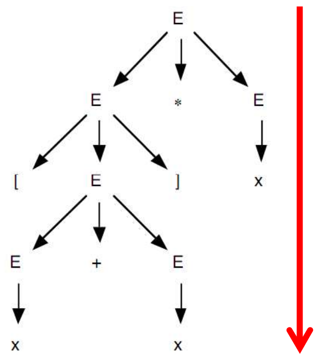
- MENEZES, P. B. **Linguagens formais e autômatos**, 6. ed., Bookman, 2011.
  - Capítulo 7.
  - + Slides disponibilizados pelo autor do livro.

ROTEIRO

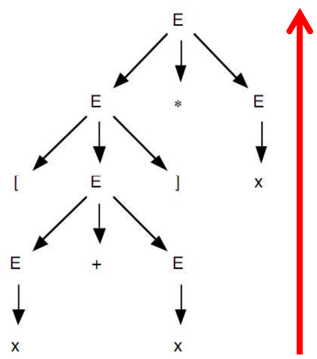
- Reconhecimento das LLC
- Autômato com pilha como reconhecedor
- AP a partir de GLC na FNG
- Autômato com pilha descendente
- **[EX]** Autômato com pilha descendente
- Algoritmo de Cocke-Younger-Kasami (CYK)
- **[EX]** Algoritmo de Cocke-Younger-Kasami (CYK)

Reconhecimento das LLC

- Algoritmos de reconhecimento podem ser:
  - *Top-down* (preditivos):
    - Construir uma árvore de derivação para a palavra a ser reconhecida (palavra de entrada);
    - Gerar os ramos, partindo da raiz (símbolo inicial da gramática), em direção às folhas (palavra de símbolos terminais).



  - *Botton-up*:
    - A partir das folhas construir a árvore de derivação em direção à raiz.



Autômato com pilha como reconhecedor

- Reconhecedores usando AP:
  - A construção é simples e imediata;
  - Existe uma relação quase direta entre produções e transições.
  - Algoritmos *top-down*:
    - Simulam a derivação mais à esquerda;
    - São não determinísticos.

AP a partir de GLC na FNG

- (visto na aula sobre "Autômato com Pilha")
- Partindo de uma GLC na Forma Normal de Greibach;
  - Em que cada produção gera exatamente um terminal;
  - A geração da palavra w leva |w| etapas de derivação.

- Como cada variável pode ter diversas produções associadas:
  - O AP testa as diversas alternativas;
  - O número de passos para reconhecer w:
    - $k^{|w|}$
    - Sendo k a metade das média de produções nas variáveis.

- Ou seja, o AP pode ser muito **ineficiente** para reconhecer entradas muito longas.

Autômato com pilha descendente

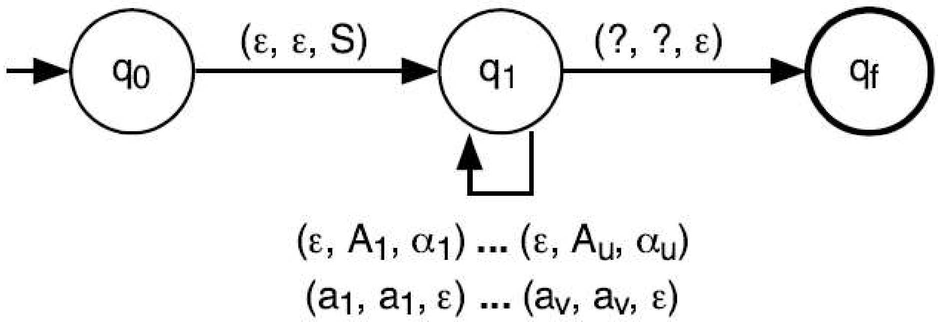
- Uma forma alternativa de construir um AP.
  - Construir um AP a partir de uma GLC sem recursão à esquerda.
    - **[OBS]** Para gerar uma GLC sem recursão à esquerda basta executar o algoritmo da FNG até a etapa 4
      - *ver aula sobre "Forma Normal de Greibach".*
  - Consiste em simular a derivação mais à esquerda.

- Ideia do algoritmo:
  - Empilhar o símbolo inicial;
  - Se topo possuí variável:
    - Substituir por todas as produções dessa variável.
  - Se top possuí terminal:
    - Testar se é igual ao próximo símbolo da entrada.

Autômato com pilha descendente

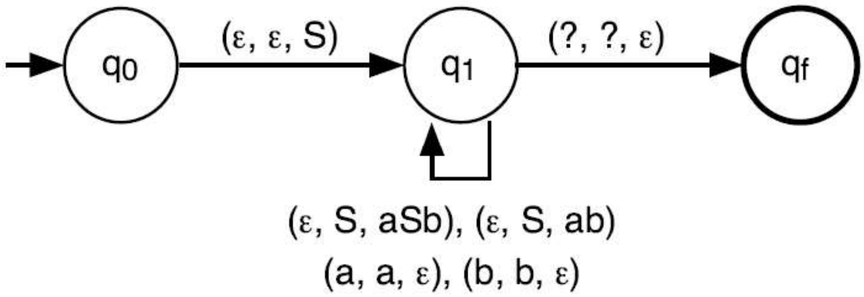
- Considere G uma GLC sem recursão à esquerda:
  - $G = (V, T, P, S)$

- O AP M reconhece a linguagem gerada por G:
  - $M = (T, \{q_0, q_1, q_f\}, \delta, q_0, \{q_f\}, V \cup T)$
    - $\delta(q_0, \varepsilon, \varepsilon) = \{ (q_1, S) \}$
    - $\delta(q_1, \varepsilon, A) = \{ (q_1, \alpha) \mid A \rightarrow \alpha \in P\}$, para toda $A \in V$
    - $\delta(q_1, a, a) = \{ (q_1, \varepsilon) \}$, para toda $a \in T$
    - $\delta(q_1, ?, ?) = \{ (q_f, \varepsilon) \}$



[EX] Autômato com pilha descendente

- Seja G uma GLC sem recursão à esquerda que reconhece a LLC L:
  - $L = \{ a^n b^n \mid n >= 1\}$
  - G = ({ S }, {a, b}, P, S)
    - P = { S → aSb | ab }
- O AP descendente que reconhece L:
  - M = ({a,b}, {q_0, q_1, q_f}, δ, q_0, {S, a, b})

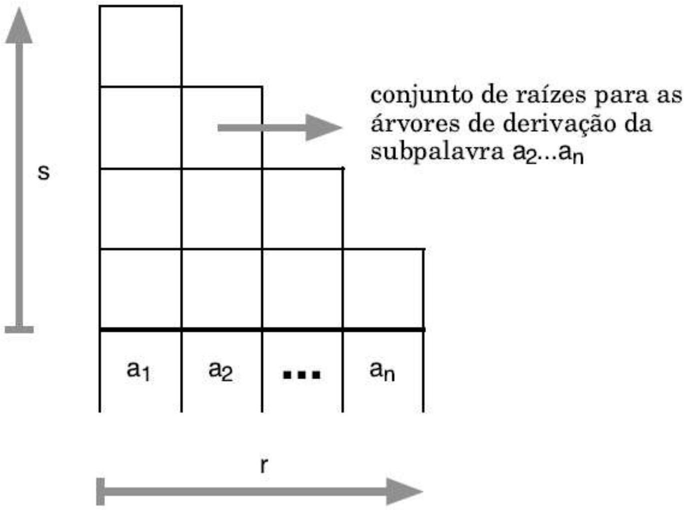


Algoritmo de Cocke-Younger-Kasami (CYK)

- Desenvolvido independentemente por **J. Cocke**, **D. H. Younger** e **T. Kasami** em 1965;
- Reconhece uma palavra a partir de uma GLC na **Forma Normal de Chomsky**:
  - Gera *botton-up* todas as árvores de derivação da entrada w.
  - O tempo de processamento é proporcional a $|w|^3$.
- Ideia do algoritmo:
  - Consiste de uma tabela triangular de derivação;
  - Célula: raízes que podem gerar a correspondente sub-árvore.

Algoritmo de Cocke-Younger-Kasami (CYK)

- Construção da tabela triangular:
  - Seja G uma GLC na FNC:
    - G = (V, T, P, S)
  - E w é a palavra a ser reconhecida
    - w = $a_1a_2...a_n$
  - V_{rs} são as células da tabela



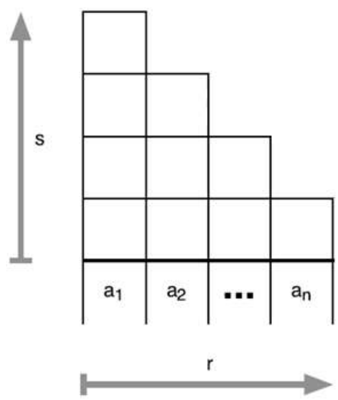
Algoritmo de Cocke-Younger-Kasami (CYK)

- Etapa 1: Variáveis que geram diretamente terminais (A → a)

```
– para r variando de 1 até n faça
–     V_{r,1} = {A | A → a_r ∈ P }
```

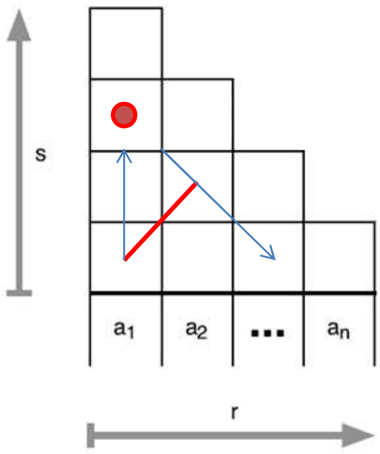
- Etapa 2: Produções que geram duas variáveis (A → BC)

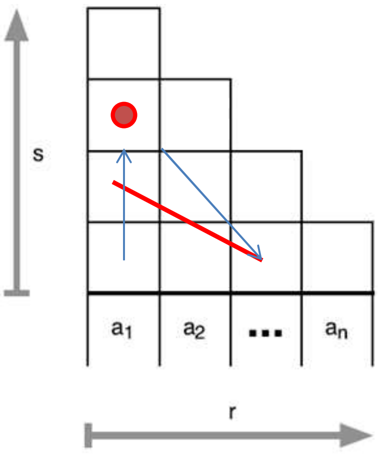
```
– para s variando de 2 até n faça
–   para r variando de 1 até (n – s + 1) faça
–      V_{r,s} = Ø
–      para k variando de 1 até (s – 1) faça
–         V_{r,s} = V_{r,s} U {A | A → BC ∈ P,
–                          B ∈ V_{rk} e
–                          C ∈ V_{(r+k),(s-k)}  }
```



Algoritmo de Cocke-Younger-Kasami (CYK)

- Interpretação das células $V_{r,k}$ e $V_{(r+k),(s-k)}$
  - $V_{r,s} = V_{1,3}$ (n=4)
  - Para s=3, r=1,2 (n-s+1)
    - Para r=1, k=1,2 (s-1)
      - k=1 → $v_{1,1}$ e $v_{2,2}$
      - k=2 → $v_{1,2}$ e $v_{3,1}$





- Etapa 3: condições de aceitação da entrada:
  - Se o símbolo inicial pertence à $V_{1,n}$ (raiz de toda palavra):
    - A palavra é aceita.

[EX] Algoritmo de Cocke-Younger-Kasami (CYK)

- G = ({S,A}, {a,b}, P, S} é uma GLC na FNC
- P = { S → AA | AS | b
- A → SA | AS | a }

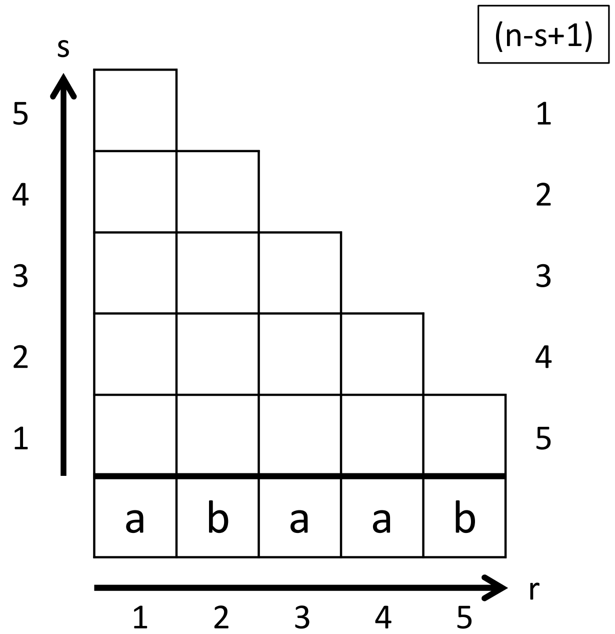
- w = abaab, n = |w| = 5

- Etapa 1:

```
para r variando de 1 até n faça
    V_r,1 = {A | A → a_r ∈ P }
```

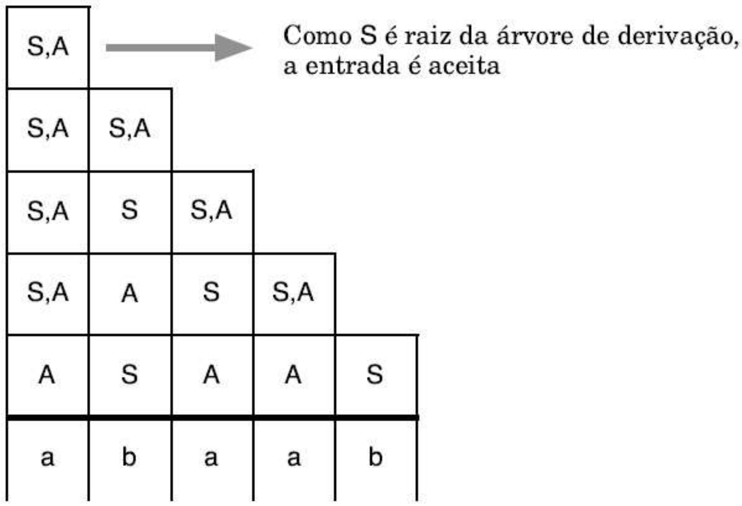
- Etapa 2:

```
para s variando de 2 até n faça
  para r variando de 1 até (n - s + 1) faça
     V_r,s = ∅
     para k variando de 1 até (s - 1) faça
        V_r,s = V_r,s U {A | A → BC ∈ P,
                         B ∈ V_rk e
                         C ∈ V_(r+k),(s-k) }
```



[EX] Algoritmo de Cocke-Younger-Kasami (CYK)

- G = ({S,A}, {a,b}, P, S} é uma GLC na FNC
- P = { S → AA | AS | b
- A → SA | AS | a }



[FIM]

- FIM:
  - **[Aula 16]** Propriedades e reconhecimento das LLC
- Próxima aula:
  - **[Aula 17]** Propriedades e reconhecimento das LLC – Algoritmo de Early