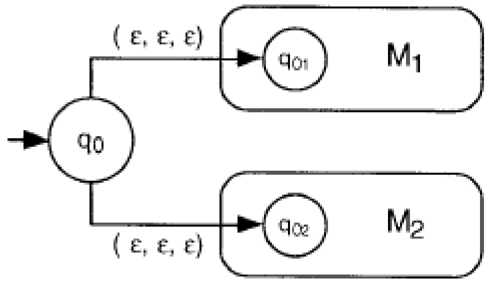
Propriedades das LLC

Operações sobre LLC

- Operações fechadas sobre LLC:
  - União
  - Concatenação



Operações sobre LLC

- Operações não-fechadas sobre LLC:
- Intersecção:
  - L1 = $\{a^n b^n c^m \mid n>=0 \text{ e } m>=0\}$ é LLC
  - L2 = $\{a^m b^n c^n \mid n>=0 \text{ e } m>=0\}$ é LLC
  - L3 = $L1 \cap L2$ = $\{a^n b^n c^n \mid n>=0\}$ é **LDC.**
- Complemento:
  - A operação de intersecção pode ser representada em termos de união e complemento, como a intersecção não é fechada não se pode afirmar nada sobre o complemento.

LLC vazia, finita ou infinita

- Se L é LLC, então L pode ser:
  - Vazia
  - Infinita/Finita
- Vazia: Seja G = (V,T,P,S) uma GLC tal que GERA(G) = L. Seja G' = (V',T',P',S) equivalente a G, eliminando os símbolos inúteis. Se P' for vazio, L é vazia.

LLC vazia, finita ou infinita

- Se L é LLC, então L pode ser:
  - Vazia
  - Infinita/Finita
- Finita/Infinita: Seja G = (V,T,P,S) uma GLC tal que GERA(G) = L. Seja G' = (V',T',P',S) equivalente a G na FNC. Se existe A, variável de V' tal que:
  - $A \rightarrow BC$, ou seja, A é referenciada no lado esquerdo de uma produção que não gera diretamente terminais;

LLC vazia, finita ou infinita

- Se L é LLC, então L pode ser:
  - Vazia
  - Infinita/Finita
- Finita/Infinita: Seja G = (V,T,P,S) uma GLC tal que GERA(G) = L. Seja G' = (V',T',P',S) equivalente a G na FNC. Se existe A, variável de V' tal que:
  - $X \rightarrow YA$ ou $X \rightarrow AY$, ou seja, se A é referenciada no lado direito de alguma produção;

LLC vazia, finita ou infinita

- Se L é LLC, então L pode ser:
  - Vazia
  - Infinita/Finita
- Finita/Infinita: Seja G = (V,T,P,S) uma GLC tal que GERA(G) = L. Seja G' = (V',T',P',S) equivalente a G na FNC. Se existe A, variável de V' tal que:
  - Existe um ciclo em A do tipo $A \stackrel{+}{\Rightarrow} \alpha A \beta$

LLC vazia, finita ou infinita

- Se L é LLC, então L pode ser:
  - Vazia
  - Infinita/Finita
- Finita/Infinita: Seja G = (V,T,P,S) uma GLC tal que GERA(G) = L. Seja G' = (V',T',P',S) equivalente a G na FNC. Se existe A, variável de V' tal que:
  - Existe um ciclo em A do tipo $A \stackrel{+}{\Rightarrow} \alpha A \beta$, então A é capaz de gerar palavras de qualquer tamanho (linguagem infinita). Se A não existe a linguagem é finita.

Algoritmo reconhecedor

- Verifica se uma palavra pertence ou não a uma linguagem.
- São contruídos a partir de uma gramática que define a linguagem .
- Reconhecedores gerados por AP são muito simples e pouco eficientes.
  - Não recomendado para entradas de tamanho considerável.
  - Top-down (preditivo): da raiz para folhas.
  - Bottom-up: das folhas para a raiz.

AP como reconhecedor

- AP Descendente
  - Gerado a partir de uma gramática sem recursão à esquerda e simula a derivação mais a direita.
  - Inicialmente, empilha o símbolo inicial.
  - Sempre que existir uma variável no topo da pilha, substitui (de forma não determinística) por todas as produções da variável.
  - Se o topo da pilha for um terminal, verifica se é igual ao próximo símbolo da entrada.

AP como reconhecedor

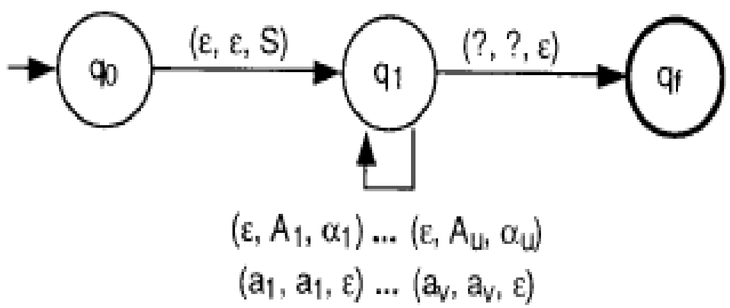
- AP Descendente
  - Tome M = (T, {q0,q1,qf}, P, q0,{qf}, V U T) onde

    P(q0,ε,ε) = {(q1,S)}

    P(q1,ε,A) = {(q1,α) | A $\rightarrow \alpha \in P$}

    P(q1,a,a) = {(q1,ε)}, para todo terminal de T

    P(q1,?,?) = {(qf, ε)}



AP como reconhecedor

- AP Descendente
  - Exemplo: Tome L5 = $\{a^n b^n \mid n >= 1\}$, representada pela gramática (sem recursão à esquerda): G5' = ({S},{a,b},P5' = {S → aSb | ab},S) é reconhecida pelo APD:

AP como reconhecedor

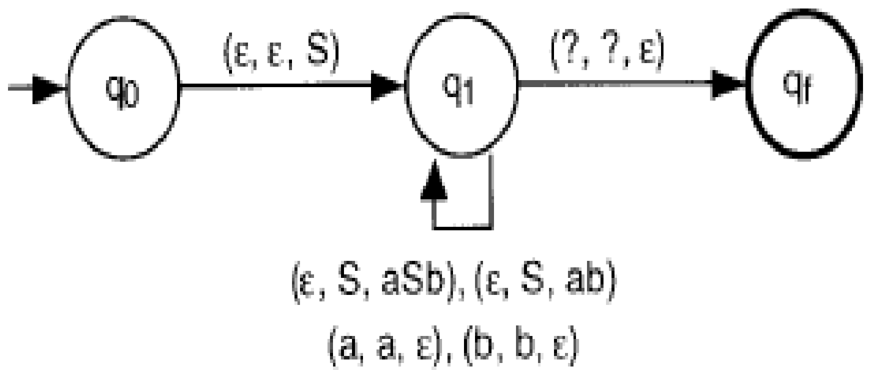
- AP Descendente
  - Exemplo: Tome L5 = $\{a^n b^n \mid n >= 1\}$, representada pela gramática (sem recursão à esquerda): G5' = ({S},{a,b},P5' = {S → aSb | ab},S) é reconhecida pelo APD:

  M5' = ({a,b},{q0,q1,qf},R5',q0,{qf},{S,a,b})

AP como reconhecedor

- AP Descendente
  - Exemplo: Tome L5 = $\{a^n b^n \mid n >= 1\}$, representada pela gramática (sem recursão à esquerda): G5' = ({S},{a,b},P5' = {S → aSb | ab},S) é reconhecida pelo APD:

  M5' = ({a,b},{q0,q1,qf},R5',q0,{qf},{S,a,b})

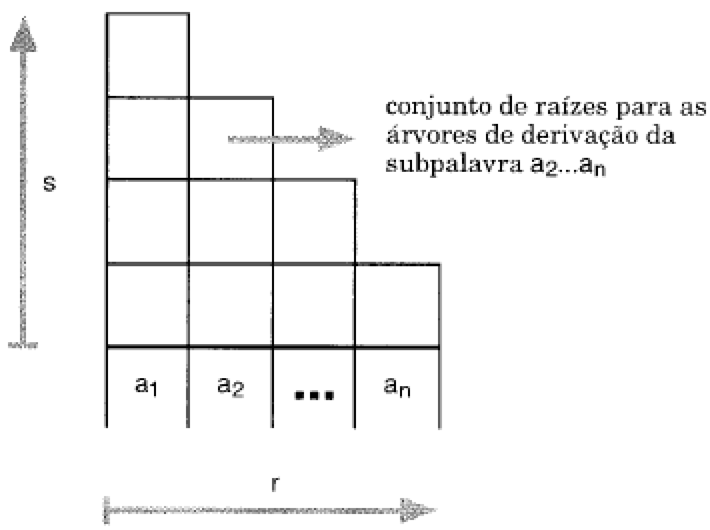


Algoritmo de Cocke-Younger-Kasami (CYK)

- Construído sobre a FN de Chomsky.
- Gera bottom-up todas as árvores de derivação da entrada em um tempo proporcional a $|w|^3$.
- Preencher a tabela triangular de derivação:
    1. Variável que geram diretamente terminais.
    2. Produções que geram duas variáveis.
    3. Condição de aceitação da entrada. Se o símbolo inicial da gramática pertencente ao vértice, então a entrada é aceita.

Algoritmo CYK

- Considere a gramática: G = ({S,A},{a,b},P,S), onde P = {S → AA | AS | b, A → SA | AS | a}.
  - Tome a palavra abaab.



Algoritmo CYK

- Considere a gramática: G = ({S,A},{a,b},P,S), onde P = {S → AA | AS | b, A → SA | AS | a}.
  - Tome a palavra abaab.
  - Etapa 1:

A	S	A	A	S
a	b	a	a	b

para r variando de 1 até n
faça $V_{r1} = \{ A \mid A \to a_r \in P \}$

Algoritmo CYK

- Considere a gramática: G = ({S,A},{a,b},P,S), onde P = {S → AA | AS | b, A → SA | AS | a}.
  - Tome a palavra abaab.
  - Etapa 2:

para s variando de 2 até n
faça para r variando de 1 até $n-s+1$
  faça $V_{rs} = \emptyset$
    para k variando de 1 até $s-1$
    faça $V_{rs} = V_{rs} \cup \{ A \mid A \to BC \in P,\ B \in V_{rk} \text{ e } C \in V_{(r+k)(s-k)} \}$

S,A	A	S	S,A	
A	S	A	A	S
a	b	a	a	b

Algoritmo CYK

- Considere a gramática: G = ({S,A},{a,b},P,S), onde P = {S → AA | AS | b, A → SA | AS | a}.
  - Tome a palavra abaab.
  - Etapa 2:

para s variando de 2 até n
faça **para** r variando de 1 até $n-s+1$
  faça $V_{rs} = \emptyset$
    para k variando de 1 até $s-1$
    faça $V_{rs} = V_{rs} \cup \{ A \mid A \to BC \in P,\ B \in V_{rk} \text{ e } C \in V_{(r+k)(s-k)} \}$

S,A	S	S,A		
S,A	A	S	S,A	
A	S	A	A	S
a	b	a	a	b

Algoritmo CYK

- Considere a gramática: G = ({S,A},{a,b},P,S), onde P = {S → AA | AS | b, A → SA | AS | a}.
  - Tome a palavra abaab.
  - Etapa 2:

para s variando de 2 até n
faça para r variando de 1 até n-s+1
    faça $V_{rs} = \emptyset$
        para k variando de 1 até s-1
        faça $V_{rs} = V_{rs} \cup \{ A \mid A \rightarrow BC \in P,\ B \in V_{rk} \text{ e } C \in V_{(r+k)(s-k)} \}$

S,A	S,A			
S,A	S	S,A		
S,A	A	S	S,A	
A	S	A	A	S
a	b	a	a	b

Algoritmo CYK

- Considere a gramática: $G = (\{S,A\},\{a,b\},P,S)$, onde $P = \{S \rightarrow AA \mid AS \mid b,\ A \rightarrow SA \mid AS \mid a\}$.
  - Tome a palavra abaab.
  - Etapa 2:

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

$$V_{rs} = V_{rs} \cup \{ A \mid A \rightarrow BC \in P,\ B \in V_{rk} \text{ e } C \in V_{(r+k)(s-k)} \}$$

S,A				
S,A	S,A			
S,A	S	S,A		
S,A	A	S	S,A	
A	S	A	A	S
a	b	a	a	b

Como S é raiz da árvore de derivação, a entrada é aceita

Exercício

- As linguagens geradas pelas gramáticas cujas produções estão representadas abaixo são vazias, finitas ou infinitas?

  - $S \rightarrow AB \mid CA$
    $A \rightarrow a$
    $B \rightarrow BC$ ε
    $C \rightarrow AB \mid$

  - $S \rightarrow As \mid AsBs \mid X$
    $X \rightarrow SS$