AP × LLC

- **A classe das linguagens reconhecidas pelos AP é igual à classe das LLC**
- **Outras conclusões**
  - para qualquer GLC, existe um AP que reconhece a linguagem gerada e sempre pára
  - construção de um AP a partir de uma GLC
    - simples e imediata
  - qualquer LLC pode ser reconhecida por um AP com somente um estado de controle lógico
    - a facilidade de memorização de informações através de estados (como nos autômatos finitos) não aumenta o poder computacional

- ***Teorema*. Se L é uma LLC, então existe M, AP tq ACEITA(M) = L**

- **Prova**

  - suponha que a palavra vazia não pertence à L
  - AP a partir de uma gramática na FNG
    - produções da forma $A \rightarrow a\alpha$, α palavra de variáveis
    - o AP simula a derivação mais à esquerda

      lê o símbolo a da fita

      lê o símbolo A da pilha

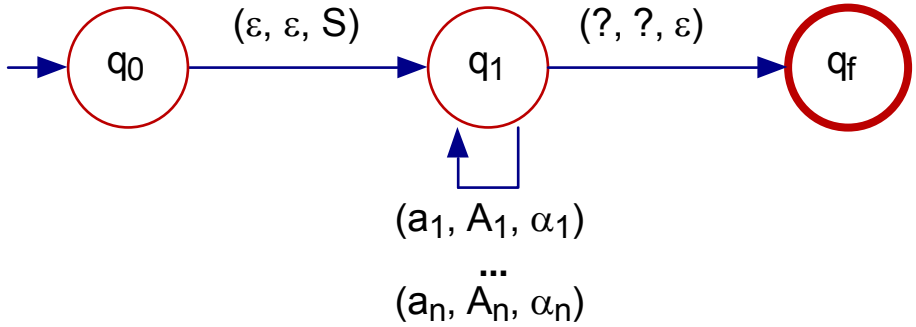
      empilha a palavra de variáveis α

- AP M a partir da gramática G = (V, T, P, S)
  - G' = (V', T', P',S), é G na FNG
  - M = (T', {q_0, q_1, q_f}, δ, q_0, {q_f}, V')

    $\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, S)\}$

    $\delta(q1,a,A) = \{(q_1, \alpha) \mid A \rightarrow a\alpha \in P'\}$

    $\delta(q_1, ?, ?) = \{(q_f, \varepsilon)\}$



- demonstração de ACEITA(M) = GERA(G')

* indução no número de movimentos de M (ou derivações de G')
* sugerida como exercício
* como o autômato pode ser modificado para tratar a palavra vazia?

♦ Exemplo $L = \{a^n b^n \mid n \geq 1\}$

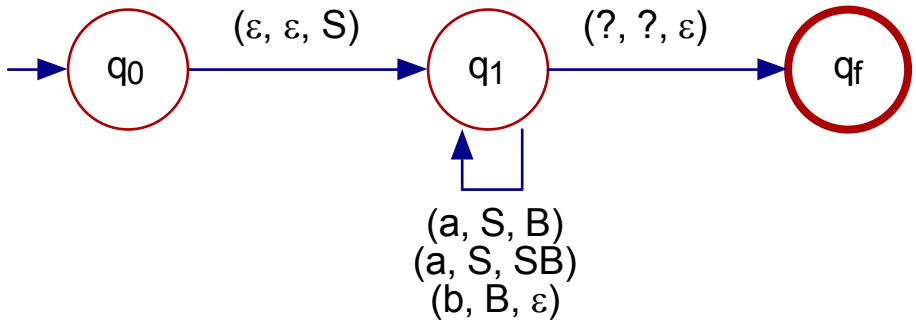
- FNG

  G = ({S, B}, {a, b}, P, S), onde

  P = {S → aB | aSB, B → b}

- AP

  M = ({a, b}, {q_0, q, q_f}, δ, q_0, {q_f}, {S, B})



- **os 2 teoremas que seguem são corolários do anterior**

- ***Teorema*. Se L é uma LLC, então:**

  a) existe M, AP que aceita por estado final, com somente 3 estados tq
  ACEITA(M) = L

  b) existe M, AP que aceita por pilha vazia, com somente um estado tq
  ACEITA(M) = L

- **Portanto**

  - o uso dos estados como "memória" não aumenta o poder de reconhecimento dos AP relativamente às LLC

- ***Teorema*. Se L é uma LLC, então existe M, AP tal que**
  - ACEITA(M) = L
  - REJEITA(M) = $\sum^*$ - L
  - LOOP(M) = $\varnothing$.
- **Ou seja**
  - para qualquer LLC existe um AP que sempre pára para qualquer entrada (por que?)
- ***Teorema*. Se L é aceita por um AP, então L é LLC**
  - não será demonstrado

de Pilhas × Poder Computacional

- **Autômato com Pilha**
  - modelo adequado para estudos
    - aplicados
    - formais
  - estrutura de pilha
    - adequada para implementação em computadores
  - poucas modificações sobre a definição do AP
    - determinam significativas alterações no seu poder computacional
  - assim, os principais estudos de linguagens e computabilidade
    - podem ser desenvolvidos usando exclusivamente o AP
    - variando o número de pilhas
    - com ou sem a facilidade de não-determinismo

♦ Principais variações

- Autômato com Pilha, *sem* usar a estrutura de *pilha*
- Autômato com Pilha *Determinístico*
- Autômato com Pilha *Não-Determinístico*
- Autômato com *Duas Pilhas*
- Autômato com *Mais de Duas Pilhas*

♦ Autômato com Pilha, sem usar a estrutura de pilha

- estados
  - ∗ única forma de memorizar informações passadas
- AP sem usar a pilha
  - ∗ muito semelhante ao Autômato Finito
- Classe das Linguagens aceitas por AP sem pilha

* com ou sem não-determinismo
* é igual a Classe das Linguagens Regulares
* exercício

♦ AP Determinístico - APD

- aceita um subconjunto próprio das LLC
  - *Linguagens Livres do Contexto Determinísticas* - *LLCD*
- LLCD inclui muitas das linguagens aplicadas em informática, com destaque para as de programação
- a implementação de um APD
  - simples e eficiente
  - facilita o desenvolvimento de processadores de linguagens
- é possível definir um tipo de gramática que gera exatamente a Classe das LLCD
  - não são restrições simples sobre a definição geral de gramática
- é fechada para a operação de complemento

- *não* é fechada para as operações de
  * união
  * intersecção
  * concatenação

♦ AP (Não-Determinístico)

- a classe das linguagens reconhecida pelo AP é exatamente a Livre do Contexto

♦ Autômato com Duas Pilhas - A2P

- é equivalente, em termos de poder computacional, à Máquina de Turing
- assim, se existe um algoritmo para resolver um problema (por exemplo, reconhecer uma determinada linguagem), então este algoritmo pode ser expresso como um A2P

- a facilidade de não-determinismo não aumenta o poder computacional do A2P

♦ Aut. com Mais de Duas Pilhas - AnP

- o poder computacional é equivalente ao do
- ou seja
  - se um problema é solucionado por um AnP
  - então o mesmo problema pode ser solucionado por um A2P

Propriedades das LLC

- **As LLC são mais gerais que as LR**
  - mas ainda são relativamente restritas
  - é fácil definir linguagens que *não* são LLC
    - $\{ww \mid w \text{ pertence a } \{a, b\}^*\}$
    - $\{a^n b^n c^n \mid n \geq 0\}$.
- **Assim**
  - como determinar se uma linguagem é LLC?
  - a Classe das LLC é fechada para operações como
    - união?
    - intersecção?

  - concatenação?
  - complemento ?

- como verificar se uma LLC é
  - infinita?
  - finita (ou até mesmo vazia)?

Investigação se é LLC

♦ **Prova de que uma linguagem é LLC**

- é suficiente expressá-la usando os formalismos
  - Gramática Livre do Contexto
  - Autômato com Pilha

♦ Prova de que uma linguagem *não* é LLC

- necessita ser realizada caso a caso
- "lema do bombeamento" para as LLC

Operações sobre LLC

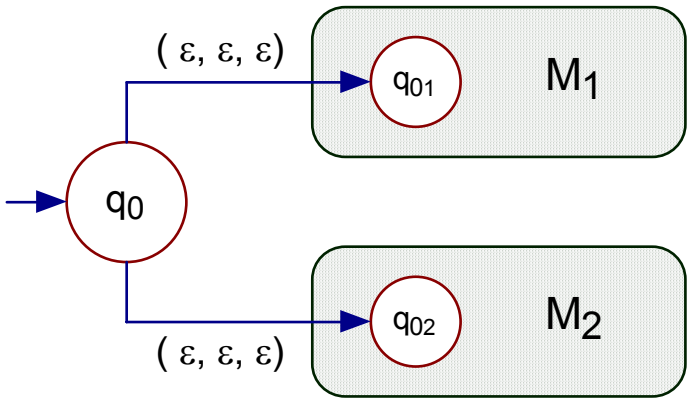
- ***Teorema*: As LLC são fechadas p/**
  - união
  - concatenação
- **Prova: *União***
  - demonstração é baseada em AP (GLC: exercício)
  - suponha L_1 e L_2, LLC. Então, existem

    $M_1 = (\sum_1, Q_1, \delta_1, q0_1, F_1, V_1)$ e

    $M_2 = (\sum_2, Q_2, \delta_2, q0_2, F_2, V_2)$

    tq $ACEITA(M_1) = L_1$ e $ACEITA(M_2) = L_2$
  - seja

$M_3 = (\sum_1 \cup \sum_2, Q_1 \cup Q_2 \cup \{q_0\}, \delta_3, q_0, F_1 \cup F_2, V_1 \cup V_2)$



- claramente, M_3 reconhece $L_1 \cup L_2$

♦ Prova: *Concatenação*

- demonstração é baseada em GLC (AP: exercício)
- suponha L_1 e L_2, LLC. Então, existem

  $G_1 = (V_1, T_1, P_1, S_1)$ e

  $G_2 = (V_2, T_2, P_2, S_2)$

  tq $GERA(G_1) = L_1$ e $GERA(G_2) = L_2$

- seja

$G_3 = (V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \rightarrow S_1S_2\}, S)$

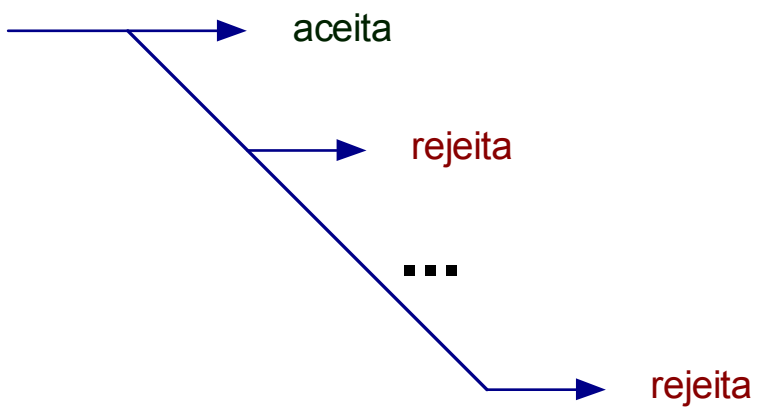
- a única produção de S é $S \rightarrow S_1S_2$
- qq palavra terá, como
  * prefixo, uma palavra de L_1
  * sufixo, uma palavra de L_2

- **O próximo teorema mostra que a Classe das LLC *não* é fechada para**
  - intersecção
  - complemento
- **Aparentemente, é uma contradição**
  - já foi verificado que
    - se L é LLC, então existe M, AP tal que ACEITA(M) = L e REJEITA(M) = L'
    - ou seja, M é capaz de *rejeitar* qualquer palavra que não pertença à L
  - o próximo teorema mostra que
    - se L é LLC, não implica que L' também é LLC
    - ou seja, *não* se pode afirmar que existe um AP que *aceite* L'

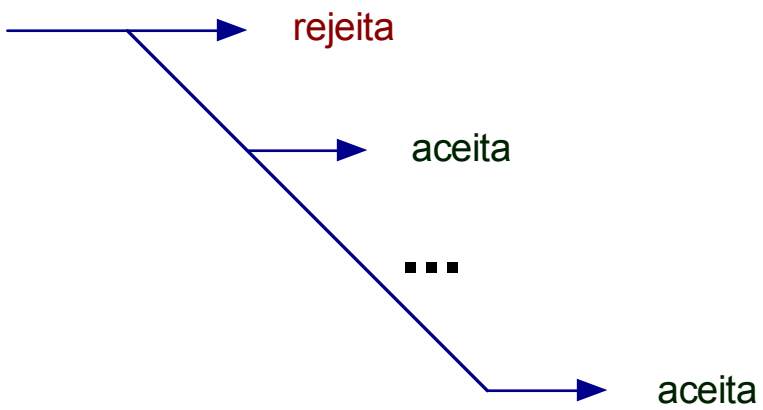
- **Assim**
  - é perfeitamente possível *rejeitar* o complemento de uma LLC
  - embora nem sempre seja possível *aceitar* o complemento

♦ Uma explicação intuitiva

- um AP não-determinista
  - aceita se pelo menos um dos caminhos alternativos aceita



- inverção de aceita por rejeita e vice-versa
  - a condição continua sendo de aceitação



- **Portanto**
  - considerando a facilidade de não-determinismo
    - o fato de existir um AP capaz de rejeitar o complemento de uma linguagem
    - não implica que existe um AP capaz de aceitar o mesmo complemento
- ***Teorema*: A Classe das LLC *não* é fechada para as operações**
  - intersecção
  - complemento

♦ Prova: *Intersecção*

- contra- exemplo
- sejam
  - $L_1 = \{a^n b^n c^m \mid n \geq 0 \text{ e } m \geq 0\}$ e
  - $L_2 = \{a^m b^n c^n \mid n \geq 0 \text{ e } m \geq 0\}$
- é fácil mostrar que L_1 e L_2 são LLC
- entretanto
  - $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$
  - não é LLC

♦ Prova: *Complemento*

- considerando que
  - *não* é fechada para a intersecção

  * intersecção pode ser representada em termos de união e complemento
  * é fechada para a união

- *não* pode-se afirmar que é fechada para o complemento

Investigação se uma LLC é Vazia, Finita ou Infinita

- **Teorema. Se L é LLC, então é possível determinar se L é**
  - vazia
  - finita
  - infinita
- **Prova: *Vazia***
  - seja G = (V, T, P, S), GLC tq GERA(G) = L
  - seja G'= (V', T', P', S) equivalente a G, eliminando os símbolos inúteis
  - se P' for vazio, então L é vazia

♦ Prova: *Finita e Infinita*

- seja G = (V, T, P, S) uma GLC tq GERA(G) = L
- seja G' = (V', T', P', S) equivalente a G
  - Forma Normal de Chomsky
  - (A $\rightarrow$a ou A $\rightarrow$ BC)
- considere somente as produções da forma A $\rightarrow$ BC
- se existe A tq
  - A $\rightarrow$ BC (A no lado esquerdo)
  - X $\rightarrow$ YA ou X $\rightarrow$ AY (A no lado direito)

  e se existe um ciclo em A do tipo A $\Rightarrow^{+}$ αAβ então
  - A é capaz de gerar palavras de qq tamanho
  - a linguagem é infinita

    - caso *não* exista tal A, então a linguagem e finita