2.7 Propriedades das Linguagens Regulares

- **LR podem ser representadas por formalismos**
  - baixa complexidade
  - grande eficiência
  - fácil implementação
- **Entretanto, é uma classe de linguagens**
  - restrita
  - limitada
  - é fácil definir linguagens que *não* são regulares

♦ Algumas questões sobre LR necessitam ser analisadas

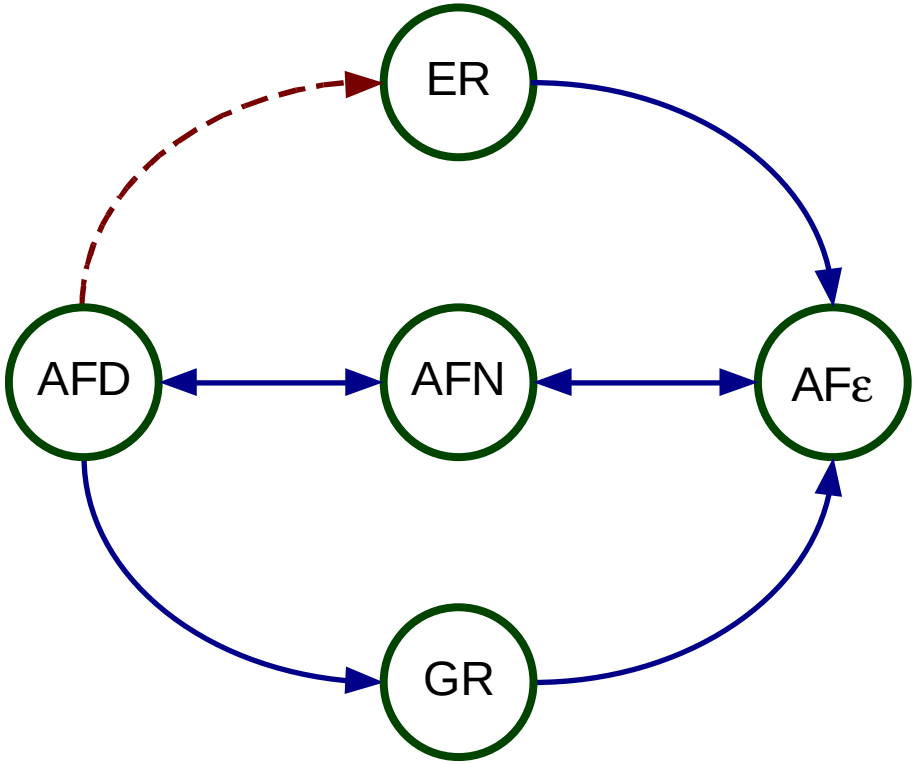
- como determinar se uma
  - linguagem *é regular*?
- como verificar se uma LR é
  - *infinita*?
  - *finita*?
  - *vazia*?
- é possível analisar duas LR e concluir se são
  - *iguais*?
  - *diferentes*?
- ou seja, é possível verificar
  - se duas especificações são equivalentes?
  - se dois compiladores são equivalentes?

  - se o compilador aceita exatamente a linguagem especificada?

- a classe das LR é fechada para operações de
  - *união*?
  - *concatenação*?
  - *intersecção*?

♦ A análise de cada propriedade

- é desenvolvida para *um* formalismo
- demais formalismos
  - é suficiente traduzir
  - AFD $\rightarrow$ ER não foi apresentada


ER
AFD
AFN
AFε
GR

♦ Bombeamento para as LR

- proposição útil no estudo das propriedades
- LR
  - é aceita por um AFD com n estados
- AFD reconhece w tq $|w| \geq n$
  - obrigatoriamente assume algum estado q mais de uma vez
  - existe um ciclo no programa que passa por q
- logo, w pode ser dividida em três subpalavras
  - $w = uvz$, onde $|uv| \leq n$ e $|v| \geq 1$
  - v é a parte reconhecida pelo ciclo
- claramente uv^iz para qq $i \geq 0$
  - é aceita pelo AFD executando o ciclo i vezes

- ***Teorema*: Se L é LR, então**
  - existe n tq,
  - para qq $w \in L$ onde $|w| \geq n$,
  - w pode ser definida como $w = uvz$
    - $|uv| \leq n$ e $|v| \geq 1$
  - para todo $i \geq 0$, $uv^iz \in L$

- **Prova**
  - L é LR
    - existe AFD $M = (\Sigma, Q, \delta, q_0, F)$ tq $L(M) = L$
  - seja n o cardinal de Q
  - seja $w = a_1a_2...a_m \in L$ tq $m \geq n$
  - suponha $\delta(q_0,a_1) = q_1$, $\delta(q_1,a_2) = q_2$,...,$\delta(q_{m-1},a_m) = q_m$

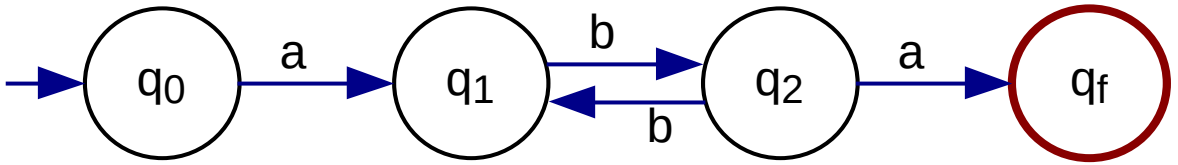
- como $m \geq n$ existem r, s onde $0 \leq r < s \leq n$ tq
  - $q_r = q_s$
  - $\underline{\delta}(q_0, a_1...a_r) = q_r$, $\underline{\delta}(q_r, a_{r+1}...a_s) = q_s$
  - $\underline{\delta}(q_s, a_{s+1}...a_m) = q_m$
- sejam $u = a_1...a_r$, $v = a_{r+1}...a_s$ e $z = a_{s+1}...a_m$
- como $r < s \leq n$, então $|v| \geq 1$ e $|uv| \leq n$
- como $q_r = q_s$, então
  - v é reconhecida em um ciclo
  - portanto, $uv^iz \in L$, para todo $i \geq 0$

♦ Lema do Bombeamento como um jogo de dois Adversários

1. O jogador 1 escolhe a linguagem L para provar que não é regular
2. O jogador 2 escolhe n , mas não revela para o jogador 1 que n foi escolhido; o jogador 1 deve inventar uma jogada para cada n possível.
3. O jogador 1 escolhe w, que deve depender do n e que deve ser no mínimo de tamanho n.
4. O jogador 2 divide w em u, v e z, obedecendo o lema do bombeamento ($v \neq \varepsilon$ e $|uv| \leq n$). Novamente, o jogador 2 não informa o jogador 1 quais são u, v e z**.**
5. O jogador 1 "ganha" escolhendo um k, que deve ser uma função de n, u, v e z**,** a qual $uv^k z$ não está em L.

- **Exemplo Provar que a linguagem L, que consiste em todas as palavras com um número igual de 0's e 1's (não tendo restrição na ordem), não é uma LR.**
- Em termos do lema do bombeamento como um jogo, somos o jogador 1 e devemos lidar com qualquer escolha que o jogador 2 faça. Se o jogador escolhe n, nós devemos escolher $w = 0^n1^n$, que certamente esta em L.
- O jogador 2 quebra w em uvz.
- Sabendo que $v \neq \varepsilon$ e $|uv| \leq n$, sabemos que uv só tem 0's. O lema do bombeamento diz que uz está em L se L é regular (k = 0). No entanto, uz tem n 1's, já que todos os 1's de w estão em z, e uz tem menos de n 0's já que perdemos os 0's de v. Como $v \neq \varepsilon$ então sabemos que pode ter até que n-1 0's entre x e z.
- Então, como uz não é palavra de L, L não é regular!

Exemplo. Bombeamento das LR



- n = 4
- no caso particular de w = abbba
  - $q_r = q_s = q_1$
  - u = a
  - v = bb
  - z = ba

E com n qualquer e $w = ab^n a$?

Linguagem Regular × Não-Regular

- ***É* regular**
  - é suficiente representar usando
    - autômato finito
    - expressão regular
    - gramática regular
- ***Não* é regular**
  - necessita ser desenvolvida para cada caso
  - algumas ferramentas específicas
    - ex: bombeamento
    - em geral, demonstração por absurdo

♦ *Exemplo*: Não é LR

L = {w | w possui o mesmo número de símbolos a e b, seguindo a ordem indicada}

♦ Prova

- bombeamento + absurdo
- suponha que L é LR
- então existe AFD M com n estados que aceita L
- seja $w = a^n b^n$
- pelo bombeamento
  - $w = uvz$ onde $|uv| \leq n$, $|v| \geq 1$
  - para todo $i \geq 0$, uv^iz é palavra de L
- ABSURDO!!!
  - $|uv| \leq n$ e uv é composta exclusivamente por símbolos a
  - então, por exemplo uv^2z, *não* pertence a L (não possui o mesmo número de símbolos a e b)

Operações Fechadas sobre LR

- **Obs**
  - para uma linguagem L sobre Σ*,
    L' denota o seu complemento em Σ*
- ***Teorema*: A classe das LR é fechada para**
  - união
  - concatenação
  - complemento
  - intersecção

♦ Prova

- *união* e *concatenação*
  - decorrem trivialmente da definição de ER
- *complemento*
  - a idéia consiste em inverter as condições de ACEITA/REJEITA do AF
- inversão das condições ACEITA/REJEITA
  - modifica o AF, garantindo que somente irá parar ao terminar de ler toda a entrada
  - transforma os estados finais em não-finais e vice-versa
- *intersecção*
  - tratar a intersecção em termos da união e complemento

♦ Complemento

- seja L uma LR sobre $\sum^*$
- seja M = ($\sum$, Q, F, q_0, δ) um AFD tq ACEITA(M) = L
- seja M' = ($\sum$, Q', F', q_0, δ') tq ACEITA(M') = L'
  - Q' = Q $\cup$ {d}
  - F' = Q' - F
  - δ' é como δ, excetuando-se

    $\delta'(q, a) = d$ se $\delta(q, a)$ não é definida

    $\delta'(d, a) = d$

♦ Intersecção

- Sejam L_1 e L_2 LR
- $L_1 \cap L_2 = (L_1' \cup L_2')'$
- como a classe das LR é fechada para
  - complemento
  - união

  então é fechada para a intersecção

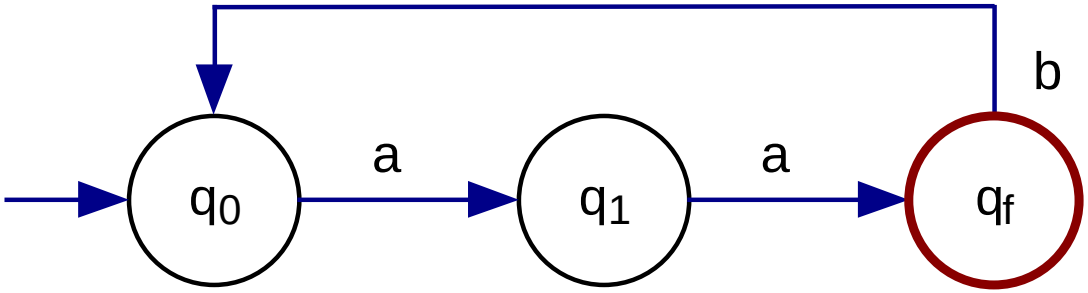
Investigação se uma LR é Vazia, Finita ou Infinita

- ***Teorema*: Se L é LR aceita por um AF M com *n* estados, então L é:**
  - **vazia** sse M *não* aceita qualquer w tq $|w| < n$
  - **finita** sse M *não* aceita w tq $n \leq |w| < 2n$
  - **infinita** sse M aceita w tq $n \leq |w| < 2n$
- **Ou seja, prova-se que**
  - *existe um algoritmo* para verificar se uma LR representada por um AF é *vazia*, *finita* ou *infinita*

♦ Pelo bombeamento

- se aceita w tq $|w| \geq n$
- então a linguagem é infinta

♦ *Exemplo*. LR Infinita



- infinita sse aceita w tq $n \leq |w| < 2n$
- menor palavra aceita de comprimento maior ou igual a 3
  - aabaa
  - possui comprimento 5
  - ou seja, $3 \leq |aabaa| < 6$

- *Aceita* w tq $n \leq |w| < 2n$, então é *Infinita*
  - seja $w \in L$ tq $n \leq |w| < 2n$
  - bombeamento
    - $w = uvz$ onde $|uv| \leq n$, $|v| \geq 1$
    - para todo $i \geq 0$, uv^iz é palavra de L
  - logo, L é *infinita*

- *Infinita*, então *aceita* w tq $n \leq |w| < 2n$
  - se é infinita, então existe w tq $|w| \geq n$
  - se $|w| < 2n$, então a prova esta completa
  - suponha que *não* existe w tq $|w| < 2n$
  - suponha $|w| \geq 2n$
    - mas de comprimento menor ou igual a qualquer outra t tq $|t| \geq 2n$
  - então $w = uvz$ onde $|uv| \leq n$, $|v| \geq 1$ e $uv^iz \in L$
  - em, particular, $1 \leq |v| \leq n$ e $uz \in L$
  - ABSURDO!!!, pois relativamente a uz tem-se que:
  - se $|uz| \geq 2n$
    - w *não* é a menor palavra tq $|w| \geq 2n$!!!
  - se $|uz| < 2n$

* como $|uvz| \geq 2n$ e $1 \leq |v| \leq n$
* então $n \leq |uz| < 2n$
* logo existe $w = uz$ tq $n \leq |w| < 2n$

- *Vazia* sse *não* aceita qualquer w tq $|w| < n$
  - processa M
    - para todo w tq $|w| < n$
  - se rejeita todas as palavras
    - a linguagem é *vazia*
  - prova
    - simples usando o bombeamento
    - exercício

- *Finita* sse *não* aceita w tq $n \leq |w| < 2n$
  - conseqüência direta do caso infinita
    - negação do caso *infinita*
  - processa M
    - para todo w tq $n \leq |w| < 2n$
  - se rejeita todas as palavras
    - a linguagem é *finita*

Igualdade de Linguagens Regulares

- ***Teorema.***
  - se M_1 e M_2 são AF
  - então existe algoritmo para determinar se
    - $ACEITA(M_1) = ACEITA(M_2)$
- **Portanto**
  - existe um algoritmo para verificar se
    - dois AF são equivalentes
    - reconhecem a mesma linguagem

♦ Prova

- sejam M_1 e M_2 AF tq
  - $ACEITA(M_1) = L_1$ e $ACEITA(M_2) = L_2$
- seja $L_3 = (L_1 \cap L_2') \cup (L_1' \cap L_2)$
- portanto, é possível construir um AF M_3 tq
  - $ACEITA(M_3) = L_3 = (L_1 \cap L_2') \cup (L_1' \cap L_2)$
- claramente, $L_1 = L_2$ sse L_3 é *vazia*
  - existe algoritmo para verificar se LR é vazia
  - existe algoritmo para $\cup$, $\cap$ e complemento

Eficiência de um AF como Algoritmo de Reconhecimento

- **Simulador de qualquer AFD**
  - fácil implementação
  - algoritmo que
    - controla a mudança de estado
    - a cada símbolo lido da entrada
- **Tempo de processamento**
  - para aceitar ou rejeitar
    - diretamente proporcional ao tamanho da entrada
  - em termos de Complexidade
    - pertence à mais rápida classe de algoritmos

- **Tempo de processamento**
  - *não* depende do AFD
  - qq AFD que reconheça a linguagem
    - terá a mesma eficiência
- **Otimização?**
  - redução do número de estados
  - existe um algoritmo para construir
    - AFD mínimo
    - AFD com o menor número de estados