LINGUAGENS REGULARES - AUTOMATOS FINITOS NÃO-DETERMINÍSTICOS

Prof. Alexandre Agustini
alexandre.agustini@pucrs.br

Material original desenvolvido pelos profs. Júlio Machado, Renata Vieira, Alexandre Agustini e outros

Autômato Finito Não-Determinístico

Def: Autômato Finito Não-Determinístico (AFN)

M = (Σ, Q, δ, q0, F)

- Σ - alfabeto (de símbolos) de entrada
- Q - conjunto de estados possíveis (finito)
- δ - (função) programa ou função de transição (função parcial)

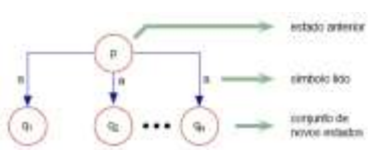
$$\delta: Q \times \Sigma \rightarrow 2^Q$$

transição: δ(q, a) = { q1, q2, …, qn }

- q0 é um elemento distinguido de Q: estado inicial
- F é um subconjunto de Q: conjunto de estados finais

Autômato como diagrama

δ(p, a) = { q1, q2, …, qn }



Computação de um autômato finito não-determinístico

- sucessiva aplicação da função programa
- para cada símbolo da entrada (da esquerda para a direita)
- até ocorrer uma condição de parada

Argumentos: computação/função programa estendida

- conjunto finito de estados e uma palavra

Parada do processamento

- Aceita a entrada

  após processar o último símbolo da fita, *existe* pelo menos um *estado final* pertencente ao conjunto de estados alternativos atingidos

- Rejeita a entrada - duas possibilidades

  após processar o último símbolo da fita, *todos* os estados alternativos atingidos são *não-finais*

  programa indefinido para o argumento (conjunto de estados e símbolo)

Def: Função Programa Estendida, Computação

M = (Σ, Q, δ, q0, F) autômato finito não-determinístico

$$\delta^*: 2^Q \times \Sigma^* \rightarrow 2^Q$$

recursivamente definida

- $\delta^*(E, \varepsilon) = E$
- $\delta^*(E, aw) = \delta^*(\cup_{q \in E}\ \delta(q, a), w)$

Transição estendida (a um conjunto de estados)

$$\delta^*(\{ q1, q2, \ldots, qn \}, a) = \delta(q1, a) \cup \delta(q2, a) \cup \ldots \cup \delta(qn, a)$$

Def: Linguagem Aceita, Linguagem Rejeitada

Seja M = (Σ, Q, δ, q_0, F) um autômato finito não-determinístico

Linguagem Aceita ou Linguagem Reconhecida por M

L(M) = ACEITA(M) = { w | δ*({ q_0 }, w) ∩ F ≠ ∅ }

Linguagem Rejeitada por M

REJEITA(M) = { w | δ*({ q_0 }, w) ∩ F = ∅ ou δ*({ q_0 }, w) é indefinida }

Exp: Autômato Finito Não-Determinístico:

aa ou bb como subpalavra

Linguagem:

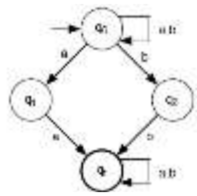
L5 = { w | w possui aa ou bb como subpalavra }

AFN:

M5 = ({ a, b }, { q_0, q_1, q_2, q_f }, $δ_5$, q_0, { q_f })

L5 = { w | w possui aa ou bb como subpalavra }

M5 = ({ a, b }, { q_0, q_1, q_2, q_f }, $δ_5$, q_0, { q_f })



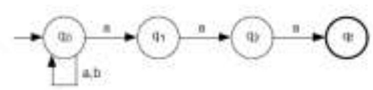
$δ_5$	a	b
q_0	{ q_0, q_1 }	{ q_0, q_2 }
q_1	{ q_f }	
q_2		{ q_f }
q_f	{ q_f }	{ q_f }

- o ciclo em q_0 realiza uma varredura em toda a entrada
- o caminho $q_0/q_1/q_f$ garante a ocorrência de aa
- o caminho $q_0/q_2/q_f$ garante a ocorrência de bb

Exp: AFN: aaa como sufixo

Linguagem:
L6 = { w | w possui aaa como sufixo }

AFN:
M6 = ({ a, b }, { q_0, q_1, q_2, q_f }, $δ_6$, q_0, { q_f })



Não-determinismo
- aparentemente, um significativo acréscimo ao poder computacional do autômato finito
- na realidade *não* aumenta seu poder computacional

Equivalência entre AFD **e** AFN
Classe dos Autômatos Finitos Determinísticos é equivalente à Classe dos Autômatos Finitos Não-Determinísticos

Equivalência entre AFD e AFN

Mostrar que
- a partir de um AFN M qualquer
- construir um AFD MD que realiza as mesmas computações
- MD simula M

AFN → AFD
- estados de MD simulam combinações de estados alternativos de M
- prova da simulação: por indução

AFD → AFN
- não necessita ser mostrado: decorre trivialmente das definições

$M = (\Sigma, Q, \delta, q_0, F)$ um AFN qualquer. AFD construído:
$MD = (\Sigma, QD, \delta D, \langle q_0 \rangle, FD)$

- QD - todas as combinações, sem repetições, de estados de Q
  - notação $\langle q_1 q_2 \ldots q_n \rangle$
  - ordem não distingue combinações: $\langle q_u q_v \rangle = \langle q_v q_u \rangle$
  - imagem de todos os estados alternativos de M
- $\delta D: QD \times \Sigma \rightarrow QD$
  $\delta D(\langle q_1 \ldots q_n \rangle, a) = \langle p_1 \ldots p_m \rangle$ sse $\delta^*(\{ q_1, \ldots, q_n \}, a) = \{ p_1, \ldots, p_m \}$
- $\langle q0 \rangle$ - estado inicial
- FD - conjunto de estados $\langle q_1 q_2 \ldots q_n \rangle$ pertencentes a QD
  - alguma componente qi pertence a F, para i em { 1, 2, …, n}

Prova: (*por indução*)

AFD MD simula as computações do AFN M ?
- indução no tamanho da palavra
- mostrar que
  $\delta D^*(\langle q_0 \rangle, w) = \langle q_1 \ldots q_u \rangle$ sse $\delta^*(\{ q_0 \}, w) = \{ q_1, \ldots, q_u \}$

Base de indução.
$| w | = 0$. Portanto $w = \varepsilon$:

$\delta D^*(\langle q_0 \rangle, \varepsilon) = \langle q_0 \rangle$ se e somente se $\delta^*(\{ q_0 \}, \varepsilon) = \{ q_0 \}$

-> verdadeiro, por definição de computação

Hipótese de indução. $| w | = n$ e $n \geq 1$.

Suponha que:
$\delta D^*(\langle q_0 \rangle, w) = \langle q_1 \ldots q_u \rangle$ sse $\delta^*(\{ q_0 \}, w) = \{ q_1, \ldots, q_u \}$

Passo de Indução. $| wa | = n + 1$ e $n \geq 1$

$\delta D^*(\langle q_0 \rangle, wa) = \langle p_1 \ldots p_v \rangle$ sse $\delta^*(\{ q_0 \}, wa) = \{ p_1, \ldots, p_v \}$
- equivale (hipótese de indução)

$\delta D(\langle q_1 \ldots q_u \rangle, a) = \langle p_1 \ldots p_v \rangle$ sse $\delta^*(\{ q_1, \ldots, q_u \}, a) = \{ p_1, \ldots, p_v \}$
- verdadeiro, por definição de δD

Logo, MD simula M para qualquer entrada w pertencente a Σ^*

Portanto, linguagem aceita por AFN
- é Linguagem Regular ou Tipo 3

Obs: Determinismo × Não-Determinismo
Muitas vezes é mais fácil desenvolver um AFN do que um AFD
- exemplo

{ w | o quinto símbolo da direita para a esquerda de w é a }
- solução determinista: não é trivial; número grande de estados
- solução não-determinista: bem simples; poucos estados

Alternativa para construir um AFD
- desenvolver inicialmente AFN
- aplicar o algoritmo apresentado na prova do teorema

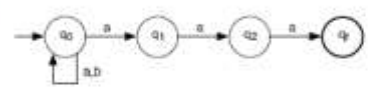
Conversão AFN → AFD
Algoritmo

1. O estado inicial do AFD é {e}, onde *e* é o estado inicial do AFN. Realize o passo 2 para este estado inicial do AFD.
2. Se $\{e_1, \ldots, e_n\}$ é o estado do AFD e $a \in \Sigma$, então construa o seguinte estado como uma entrada na tabela do AFD:

   $\delta D(\{e1, \ldots, en\}, a) = \delta N(e1, a) \cup \ldots \cup \delta N(en, a).$
3. Repita o passo 2 para cada novo estado do AFD construído dessa forma.
4. Um estado do AFD é final se um dos seus elementos é um estado final do AFN.

Construir um AFD de um AFN

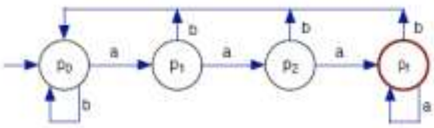
AFN



AFD

δ_D	a	b
$\langle q_0 \rangle$	$\langle q_0 q_1 \rangle$	$\langle q_0 \rangle$
$\langle q_0 q_1 \rangle$	$\langle q_0 q_1 q_2 \rangle$	$\langle q_0 \rangle$
$\langle q_0 q_1 q_2 \rangle$	$\langle q_0 q_1 q_2 q_f \rangle$	$\langle q_0 \rangle$
$\langle q_0 q_1 q_2 q_f \rangle$	$\langle q_0 q_1 q_2 q_f \rangle$	$\langle q_0 \rangle$

δ_{SD}	a	b
$p_0 = \langle q_0 \rangle$	$\langle q_0 q_1 \rangle$	$\langle q_0 \rangle$
$p_1 = \langle q_0 q_1 \rangle$	$\langle q_0 q_1 q_2 \rangle$	$\langle q_0 \rangle$
$p_2 = \langle q_0 q_1 q_2 \rangle$	$\langle q_0 q_1 q_2 q_f \rangle$	$\langle q_0 \rangle$
$p_f = \langle q_0 q_1 q_2 q_f \rangle$	$\langle q_0 q_1 q_2 q_f \rangle$	$\langle q_0 \rangle$



Autômato Finito com Movimentos Vazios

Movimentos vazios
- generalizam os movimentos não-determinísticos

Movimento vazio

transição sem leitura de símbolo algum da fita

interpretado como um não-determinismo interno ao autômato

transição encapsulada

excetuando-se por uma eventual mudança de estados

nada mais pode ser observado

Algumas vantagens
- facilita algumas construções e demonstrações

Poder computacional p/ autômatos finitos

não aumenta o poder de reconhecimento de linguagens

qualquer AFNε pode ser simulado por um AFD

Def: Autômato Finito com Movimentos Vazios – AFN-ε

$M=(\Sigma, Q, \delta, q_0, F)$

- Σ -alfabeto (de símbolos) de entrada
- Q -conjunto de estados possíveis
- δ -(função) programa ou função de transição (função parcial)
  δ:Q × ($\Sigma \cup$ { ε }) → 2^Q
  Movimento vazio ou transição vazia:
  - δ(p, ε) = { q1, q2, …, qn }
- q0 - elemento distinguido de Q: estado inicial
- F - subconjunto de Q: conjunto de estados finais

Fecho-ε

Seja N = $(\Sigma, Q, \delta, q_0, F)$ um AFN-ε e seja *e* ∈ Q um estado. Então o fecho-ε de *e*, denotado por Fε(e) é definido como segue:

- e ∈ Fε(e).
- Se p ∈ Fε(e) e existe uma transição ε de *p* para *q*, então q ∈ Fε(e).

- O fecho-ε de um estado corresponde aos estados que podem ser alcançaados com transições "ε" , isto é, sem o consumo de nenhum símbolo.

Fecho-ε de um conjunto de estados

Seja N = $(\Sigma, Q, \delta, q_0, F)$ um AFN-ε e A = {e_1, … , e_n} ⊆ Q um subconjunto dos estados de Q. Então o fecho-ε de A é definido como:

Fε({e_1, … , e_n}) = Fε(e_1) ∪ . . . ∪ Fε(e_n)

Transição estendida para AFN-ε

Seja N = $(\Sigma, Q, \delta, q_0, F)$ um AFN-ε. A função de transição estendida para N,

$\delta^*: 2^Q \times \Sigma^* \to 2^Q$, é definida recursivamente como segue:

- $\delta^*(\emptyset, w) = \emptyset$, para todo $w \in \Sigma^*$;
- $\delta^*(E, \varepsilon) = F\varepsilon(E)$, para todo $E \subseteq Q$;
- $\delta^*(E, aw) = \delta^*(\bigcup_{e \in F\varepsilon(E)} \delta(e, a), w)$, para $E \subseteq Q$, $a \in \Sigma$ e $w \in \Sigma^*$.

Conversão AFNε → AFD
Algoritmo

1. O estado inicial do AFD é $F_\varepsilon(\{e\})$, onde *e* é o estado inicial do AFN. Realize o passo 2 para este estado inicial.
2. Se $\{e_1, \dots, e_n\}$ é o estado do AFD e $a \in \Sigma$, então construa o seguinte estado como uma entrada na tabela do AFD:

$$\delta_D(\{e_1, \dots, e_n\}) = F_\varepsilon(\delta_N(e_1, a) \cup \dots \cup \delta_N(e_n, a))$$
$$= F_\varepsilon(\delta_N(e_1, a)) \cup \dots \cup F_\varepsilon(\delta_N(e_n, a))$$

Repita o passo 2 para cada novo estado do AFD construído dessa forma.

3. Um estado do AFD é final se um dos seus elementos é um estado final do AFN.