Capítulo III

Autômatos Finitos e Conjuntos Regulares

Geradores X Reconhecedores

Gramáticas Tipo 0	→	Máquinas de Turing
G. Sensíveis ao Contexto	→	Aut. Lim. Lineares
G. Livres de Contexto	→	Autômatos de Pilha
Gramáticas Regulares	→	Autômatos Finitos

Autômatos Finitos

- **São reconhecedores de linguagens regulares**

- **Tipos de Autômatos Finitos:**
  - **Autômato Finito Determinístico (AFD)**
  - **Autômato Finito Não Determinístico(AFND)**

III.1 – Autômatos Finitos Determinísticos (AFD)

Definição formal : **$M = (K, \Sigma, \delta, q_o, F)$,** onde:

- **K** → É um conjunto finito não vazio de **Estados**;
- **Σ** → É um **Alfabeto**, finito, de entrada;
- **δ** → **Função de Mapeamento** (ou de transição) definida em: $K \times \Sigma \rightarrow K$
- **q_o** → $\in K$, é o **Estado Inicial**
- **F** → $\subseteq K$, é o conjunto de **Estados Finais**

Exemplo: Seja **$M = (K, \Sigma, \delta, q_o, F)$,** onde:

$K = \{q_0, q_1\}$

$\Sigma = \{ a, b\}$

$\delta = \{\delta(q_0,a)=q_0, \delta(q_0, b)=q_1, \delta(q_1, b)= q_1, \delta(q_1,a)= - \}$

$q_o = q_0$

$F = \{q_1\}$

- **Que sentenças são aceitas (reconhecidas) por M?**
- **Qual a Linguagem aceita (reconhecida) por M?**

Representações de AF

- **Alem da representação formal, um AF pode também ser representado por:**
  - **Diagrama de Transição**
  - **Tabela de Transições**

Sentenças Aceitas (reconhecidas) por um A.F. M:

$$\delta(q_o, x) = p \mid p \in F$$

Linguagem Aceita por M:

$$T(M) = \{ x \mid \delta(q_o, x) = p \wedge p \in F \}$$

III.2 - A.F.N.D.

Definição: M = (K, Σ, δ, qo, F), onde:

K, Σ, q_o, F → mesma definição dos A.F.D.

$\delta \rightarrow K \times \Sigma = \rho(K)$, onde $\rho(K) \subseteq$ K

	Vantagem	Desvantagem
AFD	Implementação Trivial / eficiência	Representação menos natural de algumas L.R.
AFND	Representação mais natural de algumas LR	Implementação complexa / ineficiência

Exemplos: Construa um AFND M |

a) T(M) = { (a, b)*abb }

b) T(M) = { (0, 1)* (00 | 11) (0, 1)*}

c) Construa AFD ≡ AFND dos itens a) e b)

III.3 – Transformação de AFND para AFD

Teorema 3.1: "Se L é um conjunto aceito por um A.F.N.D., então ∃ um A.F.D. que aceita L"

Para provar o Teorema 3.1, precisamos:

1 - Construir um AFD M´ a partir de um dado AFND M

2 - Mostrar que M´ ≡ M

Prova:

1 – Dado um AFND M = (K, Σ, δ, qo, F), construir um A.F.D. M' = (K', Σ, δ', qo', F') como segue:

1 – K' = {ρ(k)}

2 – qo' = [qo]

3 – F' = {ρ(K) | ρ(K) ∩ F ≠ φ}

4 – Para cada ρ(K) ⊂ K'

faça δ'(ρ(K),a) = ρ'(K), onde

ρ'(K) = {p | para algum q ∈ ρ(K), δ(q, a) = p};

2 – Para mostrar que M´ ≡ M , basta mostrar que T(M´) = T(M).

Exemplo: Seja M um A.F.N.D. definido por:

δ	a	b
→qo	qo,q1	qo
q1	---	q2
q2	---	q3
*q3	---	---

III.4 - Relação entre GR e AF

Teorema 3.2: "Se G = (V_n, V_T, P, S) é uma G.R., então $\exists$ um A.F. M = (K, Σ, δ, qo, F) | T(M) = L(G)".

Prova: a – Mostrar que M existe
b – Mostrar que T(M) = L(G)

a) Defina M, como segue:

1 – $K = Vn \cup \{A\}$, onde A é um símbolo novo

2 – $\Sigma = V_T$

3 – $qo = S$

4 – $F = \{A, S\}$ se $S \rightarrow \varepsilon \in P$
$\{A\}$ se $S \rightarrow \varepsilon \notin P$

5 – Construa δ de acordo com as regras a, b e c.

a) Se $B \rightarrow a \in P \Rightarrow \delta(B, a) = A$

b) Se $B \rightarrow a\,C \in P \Rightarrow \delta(B, a) = C$

c) Para todo $a \in VT$, $\delta(A, a) = -$ (indefinido)

b) Para mostrar que T(M)=L(G), deve-se mostrar:

1 – $L(G) \subseteq T(M)$
2 – $T(M) \subseteq L(G)$

Exemplos:

1) $S \rightarrow a\,S \mid b\,B$ $B \rightarrow b\,B \mid c$	2) $S \rightarrow b\,A \mid a\,B \mid b \mid \varepsilon$ $A \rightarrow b\,A \mid a\,B \mid b$ $B \rightarrow b\,B \mid a\,C$ $C \rightarrow b\,C \mid a\,A \mid a$

Teorema 3.3: **"Se M = (K, Σ, δ, qo, F) é um A. F., então ∃ uma G.R. G = (Vn, Vt, P, S) | L(G) = T(M)"**

Prova: **a – Mostrar que G existe**
b – Mostrar que L(G) = T(M)

a) Seja M = (K, Σ, δ, qo, F) um A.F.D..
Construa uma G.R. G=(Vn, V_T, P, S), como segue:

1 – Vn = K
2 – Vt = Σ
3 – S = qo
4 – Defina P, como segue:

- **a) Se δ(B, a) = C então adicione B → aC em P**
- **b) Se δ(B, a) = C ∧ C ∈ F, adicione B → a em P**
- **c) Se qo ∈ F,**
  então ε ∈ T(M).
  Neste caso, L(G) = T(M) – {ε}, portanto, construa uma GR G_1 | L(G_1) = L(G) U {ε}
  Senão ε ∉ T(M) e L(G) = T(M)

b) Para mostrar que L(G) = T(M), devemos mostrar que:

1 – T(M) ⊆ L(G)
2 – L(G) ⊆ T(M)

Exemplos :

δ	a	b
*→S	A	B
A	S	C
B	C	S
C	B	A

δ	a	b	b
→S	S	B	-
B	-	B	A
*A	-	-	-

III.5 - Minimização de Autômatos Finitos

Definição: Um AFD M = (K, Σ, δ, qo, F) é mínimo se:

1 – Não possui estados inacessíveis (inalcançáveis);

2 – Não possui estados mortos;

3 – Não possui estados equivalentes.

Algoritmo para construção do A.F. Mínimo

Entrada: Um A.F.D. M = (K, Σ, δ, qo, F);
Saída: Um AFD Mínimo M' = (K', Σ, δ', qo', F') | M' $\equiv$ M;
Método:

1 – Elimine os estados Inacessíveis;
2 – Elimine os estados Mortos;
3 – Construa todas as CE de M como segue:

3.1 – Crie, um estado ϕ para representar as indefinições;

3.2 – Divida K em duas CE : F e K-F;

3.3 – Aplique a lei de formação de CE, até que nenhuma nova CE seja formada :

$$q_1 \equiv q_2 \Leftrightarrow \delta(q_1, a) \equiv \delta(q_2, a), \text{ p/todo } a \in \Sigma$$

4 – Construa M', como segue:

a) K' = { CE }
b) qo' = CE que contiver qo;
c) F' = { [q] | $\exists$ p $\in$ F em [q] }
d) $\delta' = \delta'([p], a) = [q] \Leftrightarrow \delta(p_1, a) = q_1$ está em M $\wedge$ $p_1 \in [p]$ $\wedge$ $q_1 \in [q]$

Exemplo: Minimize o seguinte A.F.D.

δ	a	b
* → A	G	B
B	F	E
C	C	G
* D	A	H
E	E	A
F	B	C
* G	G	F
H	H	D

Exercícios:

1)

δ	a	b	c
* → S	A	B,F	S,F
A	S,F	C	A
B	A	-	B,S,F
C	S,F	-	A,C
*F	-	-	-

2)

δ	0	1
→ S	A , D	E
A	A , B	C , E
B	B	-
C	A , B	-
D	B , D	C
*E	E	E

3)

δ	a	b
→ q0	q1	q2
q1	q3	-
q2	-	q4
*q3	q3	q3
*q4	q4	q4

4)

δ	a	b	c
*→ S	A , C	A , D	B , C
*A	A	A	B
*B	A	A	-
*C	C	D	C
*D	C	-	C

III.6 – Conjuntos e Expressões Regulares

Conjuntos Regulares (C.R.)

1 – (* definição matemática (primitiva) *)

Seja Σ um alfabeto qualquer.
Definimos um C.R. sobre Σ, como segue:

a – ϕ é um C.R. sobre Σ;

b – {ε} é um C.R. sobre Σ;

c – {a}, para todo a ∈ Σ, é um C.R. sobre Σ;

d – Se P e Q são C.R. sobre Σ, então:

1 – P ∪ Q	**(união),**
2 – P.Q (ou PQ)	**(concatenação),**
3 – P^*	**(fechamento).**

Também são C.R. sobre ∑;

e – Nada mais é C.R.

2 – Linguagens geradas por Gramáticas Regulares.

3 – Linguagens reconhecidas por Autômatos Finitos.

4 – Linguagens denotados por Expressões Regulares.

Expressões Regulares (E.R.)

Definição:

1 – ϕ é uma E.R. e denota o C.R. ϕ

2 – ε é uma E.R. e denota o C.R. $\{\varepsilon\}$

3 – $\underline{a} \in \Sigma$, é uma E.R. e denota o C.R. $\{ \underline{a} \}$

4 – Se $\underline{p}$ e $\underline{q}$ são E.R. denotando P e Q, então:

a – (p | q) é uma E.R. denotando $P \cup Q$

b – (p.q) ou (pq) é uma E.R. denotando PQ

c – $(p)^*$ é uma E.R. denotando P^*

5 – Nada mais é E.R..

Observações:

1 – ordem de precedência: 1) * 2) . 3) |

2 – abreviaturas usuais:

$$p^+ = pp^*$$

$$p^? = p \mid \varepsilon$$

Relação entre E.R. e C.R.

1 – Para todo C.R. $\exists$ uma E.R. que o denota

2 – Para toda E.R. é possível construir seu C.R.

3 – E1 $\equiv$ E2 $\Leftrightarrow$ elas denotam o mesmo C.R..

III.6.1 – Autômatos Finitos com ε-transições

AFND-ε : M = (K, Σ, δ, qo, F), onde:

K, Σ, qo, F → mesma definição dos A.F.D.

δ → K x Σ ∪ { ε } = ρ(K) , onde ρ(K) ⊆ K

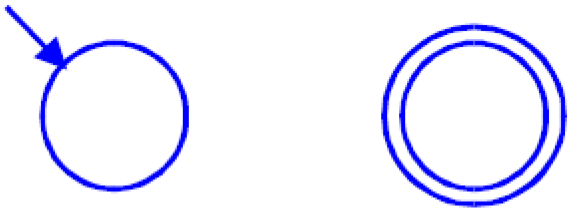
Observações:

- **ε-transições permitem movimentos independentes da entrada;**
- **O uso de ε-transições não incrementa a expressividade dos AF;**
- **Todo AFND-ε possui um AFND equivalente;**

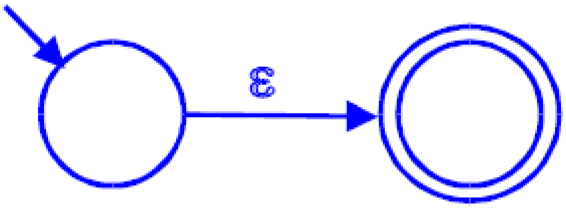
III.6.2 – Correspondência entre ER e AF

Para mostrarmos que toda ER possui um AF correspondente, é suficiente mostrarmos que toda ER básica (Φ, ε, a, (α | β), (α . β) e α* - onde α, β são ERs quaisquer) possui um AF correspndente:

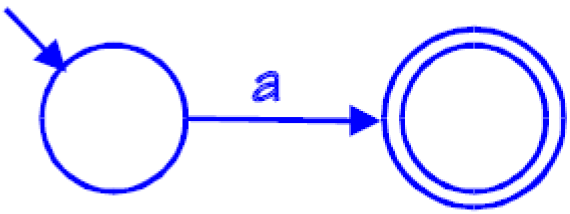
1 – AF representando a ER "ϕ" (M|T(M) = ϕ)



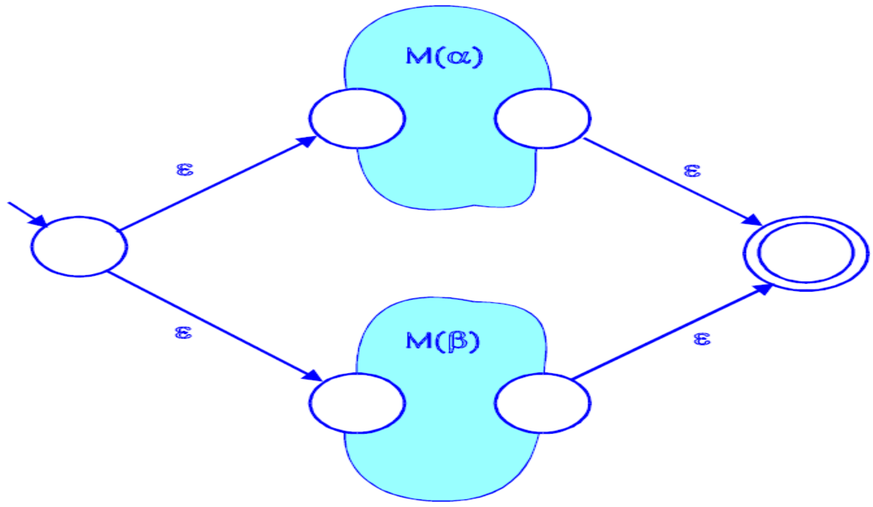
2 – AF representando a ER "ε" (M|T(M) = {ε})



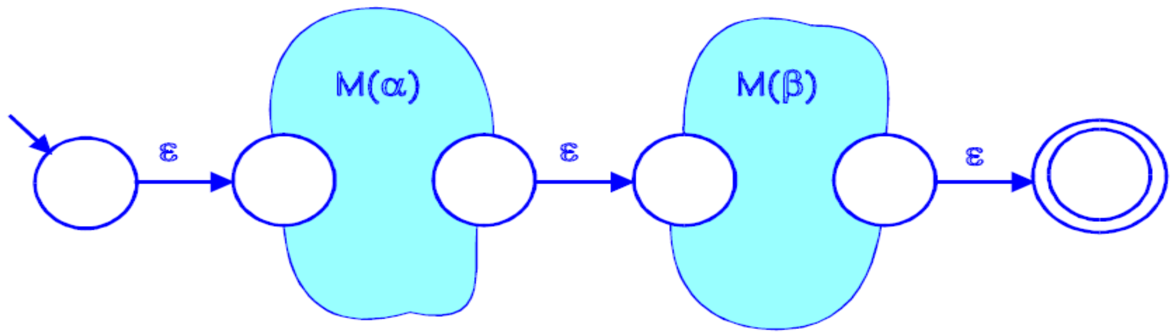
3 – AF representando a ER "a" (M|T(M) = {a})



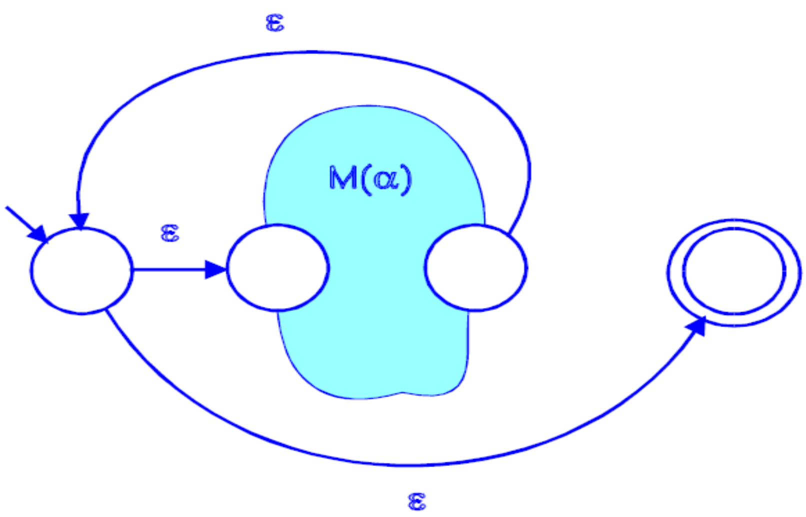
4 – AF representando a ER "α | β" (M|T(M) = {α|β})



5 – AF representando a ER "α.β" (M|T(M) = {α.β})



5 – AF representando a ER "α *" (M|T(M) = {α*})



OBS. Figuras extraídas de J.L.M.Rangel Neto (COPPE/UFRJ-PUC/RJ)

III.6.3 - Transformação de ER para AF

Principais métodos (estratégias):

- **Método de Thompson**
  - **Variação de Thompson sem ε-transições**
- **Método De Simone (Adap. de De Remer/AHO)**

III.6.3.1 - Método de Thompson

- **Consiste em:**

  1 - Decompor uma ER em sub-expressões básicas;

  2 - Construir o AFNDε de cada subexpressão;

  3 - Compor o AFNDε final (usando ε-transições)

- **Exemplo método de Thompson**

- **Exemplo variação de Thompson**

III.6.3.2 - Método de De Simone: ER → AFD

1 - Construa uma árvore binária costurada correspondente a ER, onde os nodos internos representam os operadores e os nodos folhas representam os operandos;

2 - Numere os nodos folha de 1 a n;

3 – Defina o Estado Inicial do AFD M, como sendo o estado composto pelos números dos nodos folhas alcançados no percurso da Árvore, de acordo com as rotinas "Descer" e "Subir", a partir do nodo raiz;

OBS.: Caso o percurso atinja o final da Árvore, inclua "λ" na composição do estado;

4 - Defina as transições de cada estado "q" de M, como segue:
4.1 - Se "a" é label de algum nodo que compõe "q", crie a transição

$$\delta(q, a) = p$$

onde "p" é o estado composto pelos nodos alcançados percorrendo a árvore (de acordo com as rotinas Descer e Subir) a partir da costura dos nodos com label "a" que compõe "q".
Obs.: 1 - Caso o percurso atinja o final da Árvore, inclua "λ" na composição do estado "p";
2 - Estados com a mesma composição devem ser considerados estados equivalentes;

4.2 - Se "a" não é label de nenhum nodo que compõe "q", crie a transição

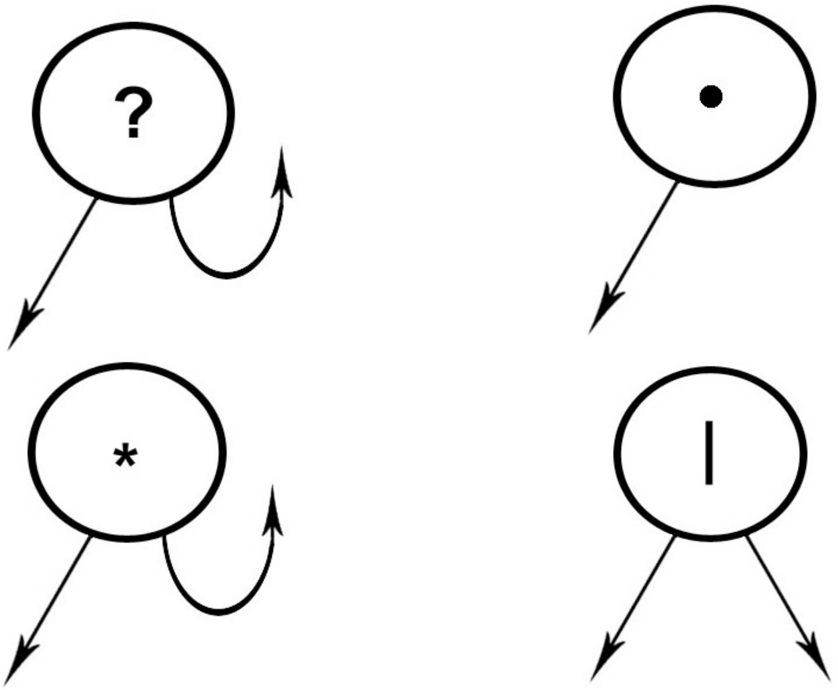
$$\delta(q, a) = -$$

5 – Repita o passo 4 para todos os estados novos criados na definição das transições;

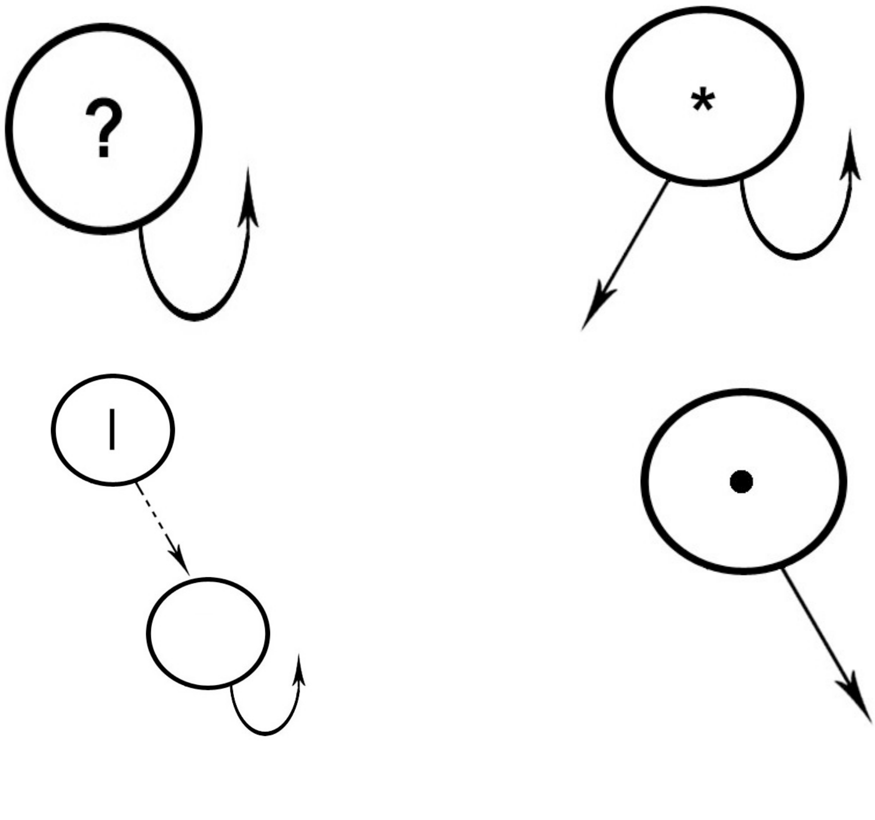
6 – Defina como Finais, os estados que contenham "λ" em sua composição.

- **Exemplos:**

Rotinas Descer:



Rotinas Subir:



III.7 – Lema do Bombeamento para Linguagens Regulares - "Pumping Lemma"

Objetivo: demonstrar que algumas linguagens não são regulares.

Lema do Bombeamento : Se L é uma LR, então existe uma constante **$n \geq 1$** | para todo **$w \in L$, $|w| \geq n$** , podemos escrever **w** como **x y z** onde:

- **$|xy| \leq n$**
- **$y \neq \varepsilon$**
- **$xy^iz \in L$ para qualquer $i \geq 0$**

Idéia geral: O "**Lema do Bombeamento**", ou "Pumping Lemma", nos diz que qualquer sentença **w** de uma **linguagem regular** pode ser decomposta em três partes: **w = xyz**, de maneira que a repetição (o *bombeamento*) de **y**, qualquer número de vezes, resulta em sentenças **xy^iz** que também pertencem à linguagem; ou seja, as sequências **xz, xyz, xyyz, ...** , também serão sentenças da linguagem em questão.

Para mostrar que uma linguagem **não é regular**, basta encontrar uma sentença w qualquer pertencente à linguagem, que não satisfaça o lema do bombeamento – isto é, não possa ser decomposta em xyz de forma que seja possível *bombear* **y** e continuar na linguagem.

Exemplos:

III.8 – Propriedades e Prob. de Decisão de CR

Propriedades Básicas de C.R.

1 – União
2 – Concatenação
3 - Fechamento
4 – Complemento: Se $L_1 \subseteq \Sigma^*$ é CR $\Rightarrow$ Σ^* - L_1 também é CR
5 – Intersecção: Se L1 e L2 são CR $\Rightarrow$ L1 $\cap$ L2 também é CR

Problemas de Decisão sobre C.R.

1 – Membership : $x \in T(M)$?
2 – Emptiness : $T(M) = \varphi$?
3 – Finiteness : T(M) é finita?
4 – Containment : $T(M1) \subseteq T(M2)$?
5 – Equivalencia : $T(M1) = T(M2)$?
6 – Intersecção Vazia : $T(M1) \cap T(M2) = \varphi$?

III.9 – Implementação de Autômatos Finitos

Formas básicas para implementação de A.F.:

- Implementação Específica
- Implementação Geral (ou genérica);

III.10 – AF com saída

A funcionalidade dos AF pode ser estendida (sem alterar a classe de linguagens reconhecida), atribuindo-se ações (significados):

- Às Transições (Máquinas de Mealy);
- Aos estados (Máquinas de Moore)

III.11 – Aplicações de A.F. e E.R.

1 – Compiladores – Análise Léxica
2 – Editores de texto – busca/substituição
3 – Reconhecimento de padrões
4 – Outras: S.O, Redes, Hipertexto, Robótica, Criptografia, ...