

1 Linguagens regulares

1.1 Alfabetos

Um alfabeto é denotado por Σ . Exemplos:

$$\Sigma = \{0, 1\} \quad \Sigma = \{a, b, c, d, e\} \quad \Sigma = \{\triangle, O, \square, X\}$$

1.2 Palavras

Uma palavra (ou cadeia) é uma sequência de zero ou mais símbolos do alfabeto.

Notação:

$$\lambda = \emptyset$$

$$0^4 = 0000$$

$$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

$$\Sigma^* = \bigcup_{i \in \mathbb{N}} \Sigma^i \quad \text{conjunto de todas as possíveis palavras deste alfabeto.}$$

Concatenação:

$$x = 00 \quad y = 11$$

$$xy = 0011$$

Reverso:

$$(xy)^R = 1100$$

Observação: uma palavra w é um palíndromo se, e somente se $w^R = w$.

1.3 Linguagens

Uma linguagem é um conjunto de palavras $L \subseteq \Sigma^*$.

Operações:

$$L_1 L_2 = \{xy \mid x \in L_1, y \in L_2\}$$

$$L^0 = \{\lambda\}$$

$$L^1 = L$$

$$L^2 = LL$$

$$L^* = \bigcup_{i \in \mathbb{N}} L^i \quad \text{Fecho de Kleene}$$

$$L^+ = \bigcup_{i \in \mathbb{N}^*} L^i$$

$$\emptyset^* = \{\lambda\}$$

$$\emptyset^+ = \emptyset$$

Teorema: As linguagens regulares são fechadas sob as seguintes operações

- União
- Interseção
- Complemento
- Concatenação
- Fecho de Kleene

2 Autômatos finitos

2.1 Determinísticos

Um autômato finito determinístico é definido por:

Q	Um conjunto finito de estados.
Σ	Um alfabeto finito.
$\delta : Q \times \Sigma \rightarrow Q$	Uma função de transição.
$q_o \in Q$	Um estado inicial.
$F \subseteq Q$	Um conjunto de estados finais.

Notação:

$L(M) = A$ A linguagem reconhecida pelo autômato M .

$L(M : F = \emptyset) = \emptyset$

$\hat{\delta} : Q \times \Sigma^* \rightarrow Q$

$\hat{\delta}(e, w)$: aplicação sucessiva de δ aos símbolos de w .

Ainda, nos autômatos existe um estado especial, denominado \emptyset , que aprisiona todas as transições omitidas.

2.1.1 Computação

Seja $M = (Q, \Sigma, \delta, q_0, F)$ um autômato finito determinístico, e $w \in \Sigma^*$.

Dizemos que M aceita w se existe uma **sequência** de estados $r_1, \dots, r_n \in Q$ satisfazendo:

1. $r_0 = q_0$
2. $\forall i \in [0, n) : \delta(r_i, w_{i+1}) = r_{i+1}$
3. $r_n \in F$

Um autômato M reconhece uma linguagem L se $\forall w \in L : M$ aceita w .

Uma linguagem é regular se existe um autômato finito que a reconhece.

2.1.2 Minimização de estados

Dois estados e e e' são **equivalentes** se

$$\hat{\delta}(e, w) \in F \iff \hat{\delta}(e', w) \in F$$

O algoritmo de minimização, então, é:

1. Produza uma partição $P_0 = \{F, Q - F\}$ de Q , separando os estados finais dos não finais.
2. Para cada bloco de estados B na partição P_i , cada símbolo s do alfabeto Σ , e cada par de estados (e, e') contidos no bloco B :
 - (a) Sejam $d = \delta(e, s)$ e $d' = \delta(e', s)$ os estados para os quais o AFD transita quando lê o símbolo s a partir dos estados e e e' , respectivamente.
 - (b) Se d e d' pertencem a blocos diferentes na partição P_i , então os estados e e e' não são equivalentes, e devem ser separados na partição P_{i+1} .
3. Se a partição P_{i+1} for diferente da partição P_i , repita o passo 2.
4. O autômato mínimo é construído de tal forma que seus estados são os blocos da última partição P produzida.

2.2 Não determinísticos

Um autômato finito não determinístico é definido por:

Q	Um conjunto finito de estados.
Σ	Um alfabeto finito.
$\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$	Uma função de transição.
$I \subseteq Q$	Um conjunto de estados iniciais.
$F \subseteq Q$	Um conjunto de estados finais.

Sendo $a \in \Sigma$ um símbolo, e $w \in \Sigma^*$ uma palavra, define-se a função de transição estendida:

$$\begin{aligned}\hat{\delta} : Q \times \Sigma^* &\rightarrow \mathcal{P}(Q) \\ \hat{\delta}(\emptyset, w) &= \{\emptyset\} \\ \hat{\delta}(X, \lambda) &= X \\ \hat{\delta}(X, aw) &= \hat{\delta}\left(\bigcup_{l \in X} \delta(l, a), w\right)\end{aligned}$$

Teorema: Todo AFN possui um AFD equivalente.

Por construção:

$$\begin{aligned}Q &= \mathcal{P}(Q_{\text{afn}}) \\ \Sigma &= \Sigma_{\text{afn}} \\ \delta(X, a) &= \bigcup_{l \in X} \delta_{\text{afn}}(l, a) \\ q_o &= I_{\text{afn}} \\ F &= \{X \subseteq Q_{\text{afn}} \mid X \cap F \neq \emptyset\}\end{aligned}$$

2.2.1 Transições λ

Um autômato finito não determinístico com transições λ introduz a possibilidade de transições sem a consumação de símbolos.

$$\begin{aligned}Q &= Q_{\text{afn}} \\ \Sigma &= \Sigma_{\text{afn}} \\ \delta : Q \times \Sigma_\lambda &\rightarrow \mathcal{P}(Q) \\ I &= I_{\text{afn}} \\ F &= F_{\text{afn}}\end{aligned}$$

Onde $\Sigma_\lambda = \Sigma \cup \{\lambda\}$.

Os estados para os quais se transita sem consumir símbolos é definido pelo fecho λ :

$$\mathcal{F}_\lambda : \mathcal{P}(Q) \rightarrow \mathcal{P}(Q)$$

Teorema: O fecho lambda de um estado é pelo menos o próprio estado.

$$\forall X \in Q : X \in \mathcal{F}_\lambda(\{X\})$$

Assim, define-se a função de transição estendida:

$$\begin{aligned}\hat{\delta} &: Q \times \Sigma_{\lambda}^* \rightarrow \mathcal{P}(Q) \\ \hat{\delta}(\emptyset, w) &= \emptyset \\ \hat{\delta}(X, \lambda) &= \mathcal{F}_{\lambda}(X) \\ \hat{\delta}(X, ay) &= \hat{\delta}\left(\bigcup_{Y \in \mathcal{F}_{\lambda}(X)} \delta(Y, a), y\right)\end{aligned}$$

Teorema: Todo AFN λ possui um AFN equivalente.

Por construção:

$$\begin{aligned}Q &= Q_{\text{afn}\lambda} \\ \Sigma &= \Sigma_{\text{afn}\lambda} \\ \delta &= \mathcal{F}_{\lambda} \circ \delta_{\text{afn}\lambda} \\ I &= \mathcal{F}_{\lambda}(I_{\text{afn}\lambda}) \\ F &= F_{\text{afn}\lambda}\end{aligned}$$

2.3 Com pilha

Um autômato finito com pilha não determinístico é definido por:

Q	Um conjunto finito de estados.
Σ	Um alfabeto finito.
Γ	Um alfabeto de pilha finito.
$\delta : Q \times \Sigma_{\lambda} \times \Gamma_{\lambda} \rightarrow \mathcal{P}(\Gamma^* \times Q)$	Uma função de transição.
$I \subseteq Q$	Um conjunto de estados iniciais.
$F \subseteq Q$	Um conjunto de estados finais.

Em cada transição, o elemento do topo da pilha é retirado para a função de transição, que por sua vez devolve uma sequência de elementos a serem empilhados, além do estado transitado.

Um AFPN aceita uma palavra se ao consumi-la, encerra-se em um estado final **com a pilha vazia**.

3 Expressões regulares

Uma expressão regular pode ser uma das seguintes formas, cada qual com a linguagem correspondente:

λ	$\{\lambda\}$
\emptyset	\emptyset
a	$\{a\}$
$R_1 + R_2$	$L(R_1) \cup L(R_2)$
$R_1 R_2$	$L(R_1) \cdot L(R_2)$
R^*	$L(R)^*$

Operações:

$$\begin{aligned}R^+ &= RR^* \\ R^0 &= \lambda \\ R^n &= RR^{(n-1)}\end{aligned}$$

4 Linguagens irregulares

Nas linguagens regulares, têm-se o **lema do bombardeamento**:

Se L é uma linguagem regular, então

$$\begin{aligned} \exists k \in \mathbb{N}^* : \\ \forall z \in L, |z| \geq k : \\ \exists u, v, w : \\ 1. z = uvw \\ 2. |uv| \leq k \\ 3. v \neq \lambda \\ 4. \forall i \in \mathbb{N}^* : (uv^i w) \in L \end{aligned}$$

O lema pode ser utilizado para provar que uma dada linguagem não é regular.

5 Linguagens livres de contexto

Uma linguagem livre de contexto é uma linguagem que pode ser denotada por uma gramática livre de contexto.

5.1 Gramáticas livres de contexto

Uma gramática livre de contexto é definida por:

V	Um conjunto finito de variáveis.
Σ	Um alfabeto finito.
R	Um conjunto de regras.
$S \in V$	Uma variável inicial.

As regras são constituídas da seguinte forma:

1. O lado esquerdo de uma regra é constituído por **uma única variável**.
2. O lado direito é constituído por uma combinação de terminais e variáveis.

Por convenção a variável inicial é a variável alvo da primeira regra.

Exemplo:

$$\begin{aligned} G = (\{A, B\}, \{0, 1, 5\}, R, A) \\ R : \quad A \rightarrow 0A1 \\ \quad \quad A \rightarrow B \\ \quad \quad B \rightarrow 5 \end{aligned}$$

Lema: Para toda GLC, existe um AFPN que a reconhece.

$$\begin{aligned} G = (V, \Sigma, R, S) \\ M = (\{i, f\}, \Sigma, (V \cup \Sigma), \delta, \{i\}, \{f\}) \end{aligned}$$

$$\begin{aligned} \delta(i, \lambda, \lambda) &= \{[f, S]\} \\ \delta(f, \lambda, X) &= \{[f, \beta] \mid (X \rightarrow \beta) \in R\} \\ \delta(f, a, a) &= \{[f, \lambda]\} \end{aligned}$$

Lema: Para todo AFPN, existe uma GLC equivalente.

Nas linguagens livres de contexto, têm-se o **lema do bombardeamento**:

Se L é uma linguagem livre de contexto, então

$$\begin{aligned} \exists k \in \mathbb{N}^* : \\ \forall w \in L, |w| \geq k : \\ \exists u, v, x, y, z : \\ 1. w = uvxyz \\ 2. |vxy| \leq k \\ 3. vy \neq \lambda \\ 4. \forall i \in \mathbb{N}^* : (uv^i xy^i z) \in L \end{aligned}$$

O lema pode ser utilizado para provar que uma dada linguagem não é livre de contexto.

5.2 Propriedade de fechamento

A classe das linguagens livre de contexto são fechadas sob as seguintes operações

- União
- Concatenação
- Fecho de Kleene

6 Máquinas de Turing

Uma máquina de Turing é definida por:

Q	Um conjunto finito de estados.
Σ	Um alfabeto de linguagem finito.
$\Gamma \supset \Sigma$	Um alfabeto de fita finito, no qual Σ está contido.
$\langle \in \Gamma$	Um demarcador do começo da fita.
$\phi \in \Gamma$	Um símbolo nulo.
$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{E, D\}$	Uma função de transição.
$i \in Q$	Um estado inicial.
$F \subseteq Q$	Um conjunto de estados finais.

Esta máquina possui uma fita, que inicialmente é

$$[\langle \Sigma^* \phi^\infty]$$

A função de transição percorre a fita, consumindo símbolos e produzindo um estado alvo, o símbolo a substituir o símbolo atual, e a direção do próximo caminhamento.

O demarcador limita o caminhamento à esquerda, ao contrário do caminhamento à direita que é ilimitado:

$$\forall \delta, e \in Q : \delta(e, \langle) = [e', \langle, D]$$

A máquina aceita uma palavra caso a transição estendida sobre esta palavra **encerra** em um estado final. Ao contrário dos autômatos, a máquina não transita implicitamente para um estado de erro quando a transição é indefinida. Neste caso, a máquina encerra a execução no estado corrente.

7 Linguagens recursivas e recursivamente enumeráveis

Linguagens recursivamente enumeráveis são as linguagens reconhecidas por uma máquina de turing. Linguagens recursivas são as linguagens reconhecidas por uma máquina de turing que **sempre encerra**.

Portanto, as linguagens recursivas são um subconjunto das linguagens recursivamente enumeráveis.

7.1 Gramáticas Irrestritas

Uma gramática irrestrita é dada por

V	Um conjunto finito de variáveis.
Σ	Um alfabeto finito.
$R = \{\alpha \rightarrow \beta \mid \alpha \in (V \cup \Sigma)^+, \beta \in (V \cup \Sigma)^*\}$	Um conjunto de regras.
$S \in V$	Uma variável inicial.

As regras são constituídas da seguinte forma: As linguagens geradas por gramáticas irrestritas são linguagens recursivamente enumeráveis.

8 Linguagens Sensíveis ao Contexto

8.1 Gramáticas Sensíveis ao Contexto

Uma gramática sensível ao contexto é dada por

V	Um conjunto finito de variáveis.
Σ	Um alfabeto finito.
$R = \{\alpha \rightarrow \beta \mid \alpha, \beta \in (V \cup \Sigma)^+, \alpha \leq \beta \}$	Um conjunto de regras.
$S \in V$	Uma variável inicial.

As gramáticas sensíveis ao contexto são **mais restritas** que as gramáticas irrestritas.

9 Autômato linearmente limitado

Um autômato linearmente limitado é uma máquina de turing não determinística com uma fita limitada:

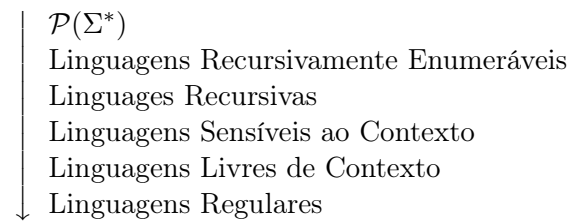
Q	Um conjunto finito de estados.
Σ	Um alfabeto de linguagem finito.
$\Gamma \supset \Sigma$	Um alfabeto de fita finito, no qual Σ está contido.
$\langle \in \Gamma$	Um demarcador do começo da fita.
$\rangle \in \Gamma$	Um demarcador do fim da fita.
$\phi \in \Gamma$	Um símbolo nulo.
$\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{E, D\})$	Uma função de transição não determinística.
$i \in Q$	Um estado inicial.
$F \subseteq Q$	Um conjunto de estados finais.

Esta máquina possui uma fita, que inicialmente é

$$[\langle \Sigma^* \rangle]$$

10 Hierarquia de Chomsky

Chomsky definiu a seguinte hierarquia das linguagens:



Tese de Church-Turing: Se uma função é computável, ela pode ser computada pela máquina de Turing.