O que é Linguagem Regular

Linguagem regular

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

Dada uma linguagem L:

- É possível determinar se ela pertence ou não à classe das linguagens regulares?
- É possível facilitar a obtenção de um AF para L?

Um teorema sobre linguagens regulares

O Lema do Bombeamento

Seja L uma linguagem regular. Então existe uma constante $k > 0$ tal que para qualquer palavra $z \in L$ com $|z| \geq k$ existem u, v e w que satisfazem as seguintes condições:

- $z = uvw$;
- $|uv| \leq k$;
- $v \neq \lambda$; e
- $uv^iw \in L$ para todo $i \geq 0$.

Uma aplicação do Lema do Bombeamento

O LB pode ser usado para provar que uma linguagem infinita, L, não é regular da seguinte forma:

1. supõe-se que L seja linguagem regular;
2. supõe-se $k > 0$, a "constante do LB";
3. escolhe-se $\boxed{\text{uma palavra } z \text{ tal que } |z| \geq k}$;
4. mostra-se que, para toda decomposição de z em u v e w tal que $|uv| \leq k$, e $v \neq \lambda$, $\boxed{\text{existe } i \text{ tal que } uv^iw \notin L}$.

Exemplo de uso do lema do bombeamento

Demonstração que $L = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbf{N}\}$ não é regular

Suponha que L seja uma linguagem regular. Seja k a constante do LB e $z = \mathtt{a}^k\mathtt{b}^k$. Como $|z| > k$, o lema diz que existem u, v e w tais que:

- $z = uvw$;
- $|uv| \leq k$;
- $v \neq \lambda$; e
- $uv^iw \in L$ para todo $i \geq 0$.

Neste caso, v só tem as, pois $uvw = \mathtt{a}^k\mathtt{b}^k$ e $|uv| \leq k$, e v tem pelo menos um a, porque $v \neq \lambda$. Isso implica que $uv^2w = \mathtt{a}^{k+|v|}\mathtt{b}^k \notin L$, o que contradiz o LB. Portanto, L não é linguagem regular.

Outro exemplo de uso do lema do bombeamento

Demonstração que $L = \{0^m 1^n \mid m > n\}$ não é regular

Suponha que L seja uma linguagem regular. Seja k a constante do LB, e seja $z = 0^{k+1}1^k$. Como $|z| > k$, o lema diz que existem u, v e w tais que:

- $z = uvw$;
- $|uv| \leq k$;
- $v \neq \lambda$; e
- $uv^iw \in L$ para todo $i \geq 0$.

Como $uvw = 0^{k+1}1^k$ e $0 < |v| \leq k$, v só tem 0s e possui no mínimo um 0. Logo, $uv^0w = 0^{k+1-|v|}1^k \notin L$, contradizendo o LB. Portanto,L não é regular.

Mais um exemplo de uso do lema do bombeamento

Demonstração que $L = \{0^n \mid n \text{ é primo}\}$ não é regular

Suponha que L seja regular. Seja k a constante do LB, e seja $z = 0^n$, em que n é um número primo maior que k. Como $|z| > k$, para provar que L não é regular, basta mostrar um i tal que $uv^iw \notin L$ supondo que $z = uvw$, $|uv| \leq k$ e $v \neq \lambda$. Como $z = 0^n$, $uv^iw = 0^{n+(i-1)|v|}$. Assim, i deve ser tal que $n + (i-1)|v|$ não seja um número primo. Ora, para isso, basta fazer $i = n+1$, obtendo-se $n + (i-1)|v| = n + n|v| = n(1+|v|)$, que não é primo (pois $|v| > 0$). Desse modo, $uv^{n+1}w \notin L$, contradizendo o LB. Logo, L não é linguagem regular.

Algumas propriedades de fechamento

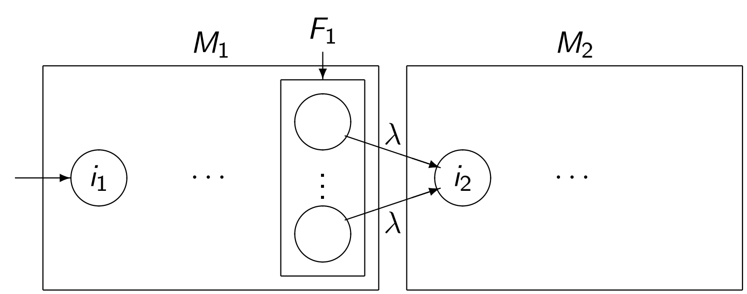
O que é fechamento

Seja uma classe de linguagens, $\mathcal{L}$, e uma operação sobre linguagens, O. Diz-se que $\mathcal{L}$ é *fechada* sob O se a aplicação de O a linguagens de $\mathcal{L}$ resulta sempre em uma linguagem de $\mathcal{L}$.

A classe das linguagens regulares é fechada sob:

1. complementação;
2. união;
3. interseção;
4. concatenação;
5. fecho de Kleene.

Fechamento sob concatenação/esquema



Fechamento sob concatenação

Sejam dois AFDS:

$$M_1 = (E_1, \Sigma_1, \delta_1, i_1, F_1) \text{ e } M_2 = (E_2, \Sigma_2, \delta_2, i_2, F_2),\ E_1 \cap E_2 = \emptyset.$$

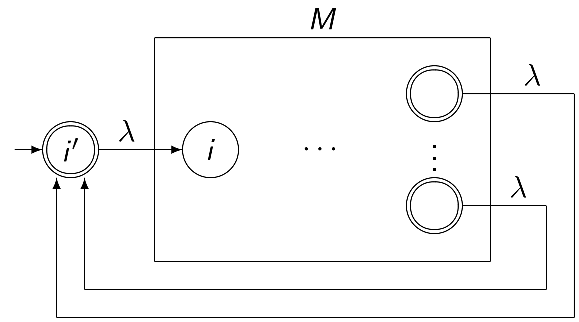
O AFNλ M_3 reconhece $L(M_1)L(M_2)$:

$$M_3 = (E_1 \cup E_2, \Sigma_1 \cup \Sigma_2, \delta_3, \{i_1\}, F_2)$$

em que δ_3 é dada por:

- $\delta_3(e, a) = \{\delta_1(e, a)\}$ para todo $e \in E_1$ e $a \in \Sigma_1$;
- $\delta_3(e, a) = \{\delta_2(e, a)\}$ para todo $e \in E_2$ e $a \in \Sigma_2$;
- $\delta_3(e, \lambda) = \{i_2\}$ para todo $e \in F_1$, e
  $\delta_3(e, \lambda) = \emptyset$ para $e \in (E_1 \cup E_2) - F_1$.

Fechamento sob fecho de kleene/esquema



Fechamento sob fecho de kleene

Seja um AFD $M = (E, \Sigma, \delta, i, F)$.

O AFNλ M' reconhece $L(M)^*$:

$$M' = (E \cup \{i'\}, \Sigma, \delta', \{i'\}, F \cup \{i'\})$$

em que $i' \notin E$, e δ' é dada por:

- $\delta'(i', \lambda) = \{i\}$;
- $\delta'(e, a) = \{\delta(e, a)\}$ para todo $e \in E$ e $a \in \Sigma$;
- $\delta'(e, \lambda) = \{i'\}$ para todo $e \in F$, e
  $\delta'(e, \lambda) = \emptyset$ para $e \in E - F$.

Aplicações das propriedades de fechamento

Três aplicações das propriedades de fecho das linguagens regulares:

1. provar que uma linguagem é regular;
2. provar que uma linguagem não é regular;
3. facilitar a obtenção de AF para uma linguagem regular.

Exemplo de aplicação do tipo 1

Sejam:

- $L_1 = \{w \in \{0,1\}^* \mid w$ representa número divisível por $6\}$; e
- $L_2 = \{w \in \{0,1\}^* \mid$ o terceiro dígito de w, da direita para a esquerda, é $1\}$.

$L_1 - L_2$ é regular?

Exemplo de aplicação do tipo 2

Seja $L = \{\mathtt{a}^k\mathtt{b}^m\mathtt{c}^n \mid k = m + n\}$. Prova-se, a seguir, que L não é regular.
Suponha que L seja uma linguagem regular. Como $\{\mathtt{a}\}^*\{\mathtt{b}\}^*$ é linguagem regular e a classe das linguagens regulares é fechada sob interseção, segue-se que $L \cap \{\mathtt{a}\}^*\{\mathtt{b}\}^*$ deve ser uma linguagem regular. Mas, $L \cap \{\mathtt{a}\}^*\{\mathtt{b}\}^* = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$, que não é linguagem regular. Logo, L não é linguagem regular.

Exemplo de aplicação do tipo 3

Sejam:

- $L_1 = \{w \in \{0,1\}^* \mid w$ representa número divisível por $6\}$; e
- $L_2 = \{w \in \{0,1\}^* \mid$ o terceiro dígito de w, da direita para a esquerda, é $1\}$.

Como construir um AFD para $L_1 - L_2$?

Associando saída aos estados

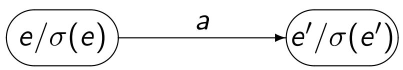
Máquina de Moore

Uma máquina de Moore é uma sêxtupla $(E, \Sigma, \Delta, \delta, \sigma, i)$, em que:

- E (o conjunto de estados), Σ (o alfabeto de entrada), δ (a função de transição) e i (o estado inicial) são como em AFDs;
- Δ é o alfabeto de saída; e
- $\sigma : E \to \Delta$ é a função de saída, uma função total.

Diagrama de estados de uma máquina de Moore

Em um diagrma de estados, a transição $\delta(e, a) = e'$ é representada assim, juntamente com $\sigma(e)$ e $\sigma(e')$:



A saída computada por uma máquina de Moore

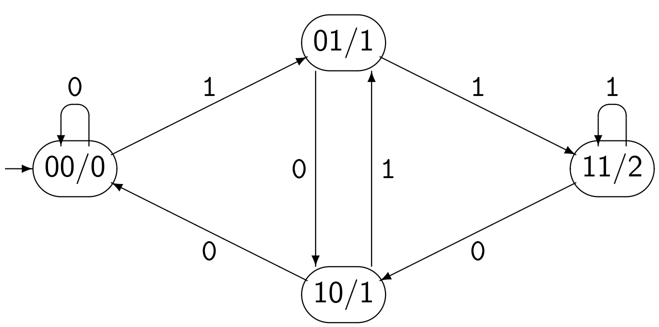
Seja uma máquina de Moore $M = (E, \Sigma, \Delta, \delta, \sigma, i)$. A **função de saída estendida** para M, $r : E \times \Sigma^* \rightarrow \Delta^*$ é definida recursivamente como segue:

a) $r(e, \lambda) = \sigma(e)$;
b) $r(e, ay) = \sigma(e)r(\delta(e, a), y)$, para todo $a \in \Sigma$ e $y \in \Sigma^*$.

A **saída computada** por uma máquina de Moore $M = (E, \Sigma, \Delta, \delta, \sigma, i)$ para a palavra $w \in \Sigma^*$ é $r(i, w)$.

Exemplo de máquina de Moore

Último símbolo de $r(00, w)$: número de 1s nos dois últimos símbolos de w.



Mealy

Associando saída às transições

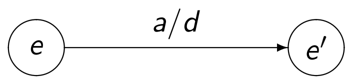
Máquina de Mealy

Uma máquina de Mealy é uma sêxtupla $(E, \Sigma, \Delta, \delta, \sigma, i)$, em que:

- E *(*o conjunto de estados*)*, Σ *(*o alfabeto de entrada*)*, δ *(*a função de transição*)* e i *(*o estado inicial*)* são como em AFDs;
- Δ é o alfabeto de saída;
- $\sigma : E \times \Sigma \rightarrow \Delta$ é a função de saída, uma função total.

Diagrama de estados de uma máquina de Mealy

Uma transição $\delta(e, a) = e'$ com a saída $\sigma(e, a) = d$ é representada em um diagrama de estados assim:



A saída computada por uma máquina de Mealy

Seja uma máquina de Mealy $M = (E, \Sigma, \Delta, \delta, \sigma, i)$. A **função de saída estendida** para M, $s : E \times \Sigma^* \to \Delta^*$, é definida recursivamente como segue:

a) $s(e, \lambda) = \lambda$;

b) $s(e, ay) = \sigma(e, a)s(\delta(e, a), y)$, para todo $a \in \Sigma$ e $y \in \Sigma^*$.

A **saída computada** por uma máquina de Mealy $M = (E, \Sigma, \Delta, \delta, \sigma, i)$ para a palavra $w \in \Sigma^*$ é $s(i, w)$.

Quociente da divisão por 6 de número em binário

Sejam $\eta(x) = 6q_1 + r_1$ $(0 \leq r_1 < 6)$,
$\eta(xa) = 6q_2 + r_2$ $(0 \leq r_2 < 6)$.

Dois casos:

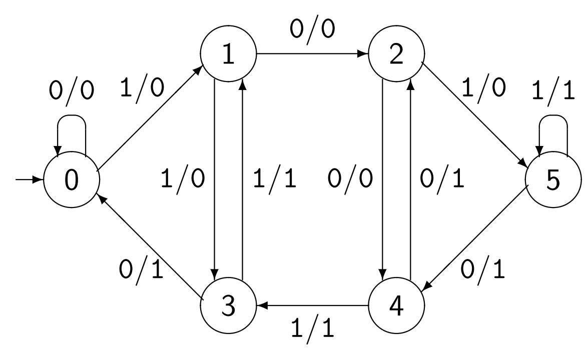
$\underline{a = 0}$: Como $\eta(x0) = 2\eta(x)$, $6q_2 + r_2 = 2(6q_1 + r_1)$.
Logo, $q_2 = 2q_1 + (2r_1 - r_2)/6$.

$\underline{a = 1}$: Como $\eta(x1) = 2\eta(x)+1$, $6q_2+r_2 = 2(6q_1+r_1)+1$.
Logo, $q_2 = 2q_1 + (2r_1 + 1 - r_2)/6$.

Portanto, se o próximo símbolo for:

- 0: o próximo dígito do quociente é $(2r_1 - r_2)/6$.
- 1: o próximo dígito do quociente é $(2r_1 - r_2 + 1)/6$.

Máquina Mealy para quociente da divisão por 6



Equivalência de máquinas de Moore e de Mealy

Máquinas equivalentes

Uma máquina de Moore $(E_1, \Sigma, \Delta, \delta_1, \sigma_1, i_1)$ e uma máquina de Mealy $(E_2, \Sigma, \Delta, \delta_2, \sigma_2, i_2)$ são ditas equivalentes se, para todo $w \in \Sigma^*$, $r(i_1, w) = \sigma_1(i_1)s(i_2, w)$.

Obtendo máquina de Mealy a partir de máquina de Moore

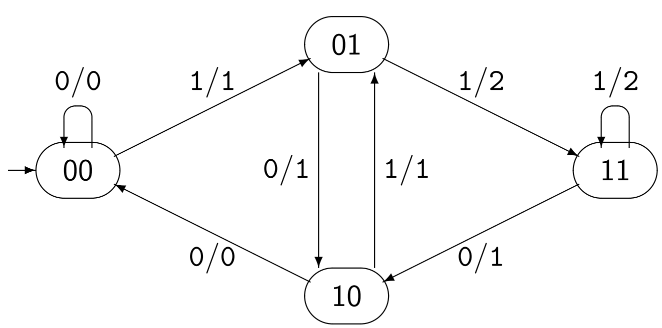
Seja uma máquina de Moore $M = (E, \Sigma, \Delta, \delta, \sigma, i)$.
Máquina de Mealy equivalente:

$M' = (E, \Sigma, \Delta, \delta, \sigma', i)$, em que:

$\sigma'(e, a) = \sigma(\delta(e, a))$, $\forall (e, a) \in E \times \Sigma$.

Um exemplo

Máquina de Moore ▸ Moore $\Rightarrow$ Máquina de Mealy:



Obtendo máquina de Moore a partir de máquina de Mealy

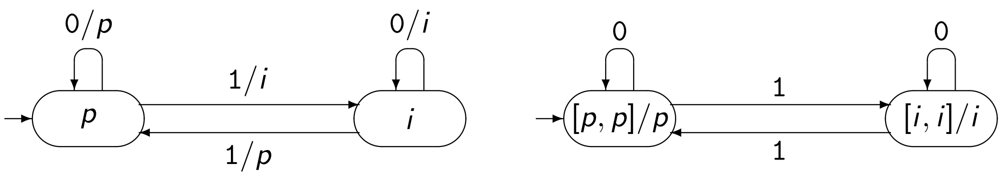
Seja uma máquina de Mealy $M = (E, \Sigma, \Delta, \delta, \sigma, i)$.
Máquina de Moore equivalente:

$M' = (E', \Sigma, \Delta, \delta', \sigma', i')$, em que:

- $i' = [i, d_0]$ para um certo $d_0 \in \Delta$ (qualquer um serve);
- $E' = \{[\delta(e, a), \sigma(e, a)] \mid e \in E \text{ e } a \in \Sigma\} \cup \{i'\}$;
- $\delta'([e, d], a) = [\delta(e, a), \sigma(e, a)]$ para cada $[e, d] \in E'$ e $a \in \Sigma$;
- $\sigma'([e, d]) = d$ para cada $e \in E$ e $d \in \Delta$.

Um exemplo

Máquina de Mealy $\Rightarrow$ Máquina de Moore:



Máquina de Mealy   Máquina de Moore

Denotação e geração de linguagens regulares

Duas novas formas de especificar as linguagens regulares: expressões regulares e gramáticas regulares.

- **Expressão Regular:** especifica uma linguagem por meio de uma expressão que a denota.
- **Gramática Regular:** especifica uma linguagem por meio de um conjunto de regras que a gera.

O que é expressão regular

Expressão regular

Uma expressão regular (ER) sobre um alfabeto Σ é definida recursivamente como segue:

1. $\emptyset$, λ, e a para qualquer $a \in \Sigma$ são expressões regulares; elas denotam $\emptyset$, $\{\lambda\}$ e $\{a\}$;
2. se r e s são expressões regulares, então são expressões regulares: $(r+s)$, (rs), e r^*; elas denotam $L(r) \cup L(s)$, $L(r)L(s)$ e $L(r)^*$.

Exemplos de expressões regulares

ERs sobre $\Sigma = \{0, 1\}$ e conjuntos regulares denotados por elas:

ER	*Linguagem denotada*
$\emptyset$	$\emptyset$
λ	$\{\lambda\}$
(01)	$\{0\}\{1\} = \{01\}$
$(0+1)$	$\{0\} \cup \{1\} = \{0, 1\}$
$((0+1)(01))$	$\{0, 1\}\{01\} = \{001, 101\}$
0^*	$\{0\}^* = \{0^n \mid n \geq 0\}$
$(0+1)^*$	$\{0, 1\}^* = \Sigma^*$
$(((0+1)^*1)(0+1))$	$\{0, 1\}^*\{1\}\{0, 1\}$

Prioridades dos operadores

Regras para omissão de parênteses:

a) Como a união é associativa, pode-se escrever $(r_1 + r_2 + \cdots + r_n)$, omitindo-se os parênteses internos.

b) Idem, para a concatenação.

c) Os parênteses externos podem ser omitidos.

d) Fecho de Kleene tem precedência sobre união e concatenação.

e) Concatenação tem precedência sobre união.

Algumas equivalências

1.	$r + s = s + r$	11.	$r^{**} = r^*$
2.	$r + \emptyset = r$	12.	$r^* = (rr)^*(\lambda + r)$
3.	$r + r = r$	13.	$\emptyset^* = \lambda$
4.	$r\lambda = \lambda r = r$	14.	$\lambda^* = \lambda$
5.	$r\emptyset = \emptyset r = \emptyset$	15.	$r^*r^* = r^*$
6.	$(r + s)t = rt + st$	16.	$rr^* = r^*r$
7.	$r(s + t) = rs + rt$	17.	$(r^* + s)^* = (r + s)^*$
8.	$(r + s)^* = (r^*s)^*r^*$	18.	$(r^*s^*)^* = (r + s)^*$
9.	$(r + s)^* = r^*(sr^*)^*$	19.	$r^*(r + s)^* = (r + s)^*$
10.	$(rs)^* = \lambda + r(sr)^*s$	20.	$(r + s)^*r^* = (r + s)^*$

Algumas observações sobre a tabela

- Qualquer equivalência que não envolva fecho de Kleene pode ser derivada a partir de 1 a 7 mais as propriedades de associatividade da união e da concatenação.
- Com o fecho de Kleene, não há um conjunto finito de equivalências a partir das quais se possa derivar qualquer outra.
- Algumas equivalências são redundantes. Por exemplo, a 13 pode ser obtida de 2, 5 e 10:

$$\begin{aligned} \emptyset^* &= (r\emptyset)^* && \text{por 5} \\ &= \lambda + r(\emptyset r)^*\emptyset && \text{por 10} \\ &= \lambda + \emptyset && \text{por 5} \\ &= \lambda && \text{por 2} \end{aligned}$$

Simplificação de expressões regulares

$$\begin{aligned} \underline{(00^* + 10^*)}0^*(1^* + 0)^* &= (0 + 1)\underline{0^*0^*}(1^* + 0)^* && \text{por 6} \\ &= (0 + 1)0^*\underline{(1^* + 0)^*} && \text{por 15} \\ &= (0 + 1)0^*\underline{(1 + 0)^*} && \text{por 17} \\ &= (0 + 1)\underline{0^*(0 + 1)^*} && \text{por 1} \\ &= (0 + 1)(0 + 1)^* && \text{por 19} \end{aligned}$$

Diga por que:

- $(r + rr + rrr + rrrr)^* = r^*$.
- $((0(0 + 1)1 + 11)0^*(00 + 11))^*(0 + 1)^* = (0 + 1)^*$.
- $r^*(r + s^*) = r^*s^*$.

Notações úteis

- r^+ significa (rr^*).
- r^n, $n \geq 0$ é assim definida, recursivamente:
  a) $r^0 = \lambda$;
  b) $r^n = rr^{n-1}$, para $n \geq 1$.

Exemplos:

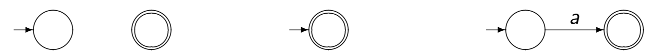
$(0+1)^{10}$.

$r^* = (r^n)^*(\lambda + r + r^2 + \cdots + r^{n-1})$, para $n > 1$.

Obtendo AF a partir de expressão regular

Toda expressão regular denota uma linguagem regular.

1. AFs que reconhecem $\emptyset$, $\{\lambda\}$ e $\{a\}$:



AF para $\emptyset$ **AF para** $\{\lambda\}$ **AF para** $\{a\}$

2. Dados AFs para L_1 e L_2, é possível construir AFs para $L_1 \cup L_2$, L_1L_2 e L_1^*.

Obtendo expressões regular a partir de AF

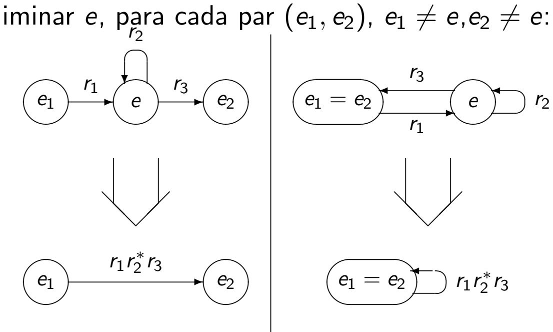
Toda linguagem regular é denotada por alguma expressão regular.

Seja um AFN $M = (E, \Sigma, \delta, I, F)$.

1. Obtenha AFNλ $M' = (E', \Sigma, \delta, i, \{f\})$ equivalente a M tal que:
   - $i \notin \delta(e, a)$ para todo par $(e, a) \in E' \times \Sigma$;
   - $\delta(f, a) = \emptyset$ para todo $a \in \Sigma$.
2. Obtenha diagrama ER inicial a partir de M': substitua transições de e para e' sob s_1, s_2, ..., s_m, por uma só transição de e para e' sob $s_1 + s_2 + \cdots + s_m$.

Obtendo expressões regular a partir de AF

3. Elimine um a um os estados do diagrama ER, exceto i e f.
   - Para eliminar e, para cada par (e_1, e_2), $e_1 \neq e$, $e_2 \neq e$:



(a) $\mathbf{e_1 \neq e_2}$ **(b)** $\mathbf{e_1 = e_2}$

   - Se havia transição de e_1 para e_2 sob s substitua-a por transição de e_1 para e_2 sob $s + r_1 r_2^* r_3$.
4. A ER resultante é o rótulo da transição de i para f.

Exemplo

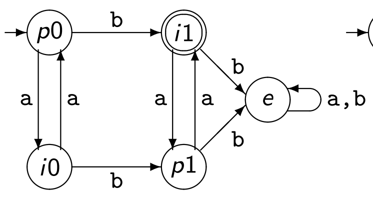


Diagrama de estados

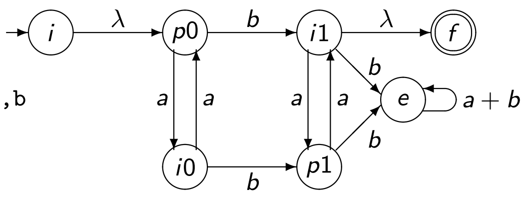
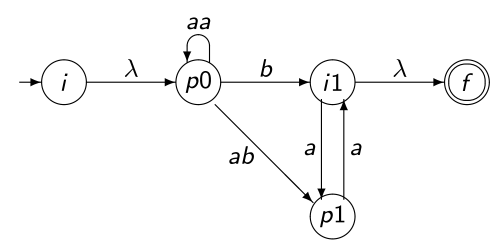


Diagrama ER

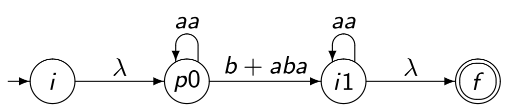
Exemplo (cont.)

1. **Eliminando** e. Como não existe transição de e para algum e_2 diferente de e, ele é simplesmente eliminado.
2. **Eliminando** $i0$:

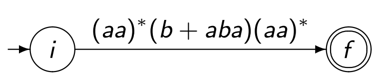


Exemplo (cont.)

3. **Eliminando** $p1$:



4. **Eliminando** $p0$ **e** i_1:



O que é uma gramática regular

Gramática regular

Uma gramática regular (GR) é uma gramática (V, Σ, R, P), em que cada regra tem uma das formas:

- $X \rightarrow a$;
- $X \rightarrow aY$;
- $X \rightarrow \lambda$;

$X, Y \in V$ e $a \in \Sigma$.

$\Rightarrow$ Formato das formas sentenciais: wA, $w \in \Sigma^+$, $A \in V$.

Exemplo

$L = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid w$ não contém $\mathtt{abc}\}$.

Uma GR que gera L: $(\{A, B, C\}, \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, R, A)$, em que R contém:

$$A \to \mathtt{a}B \mid \mathtt{b}A \mid \mathtt{c}A \mid \lambda$$
$$B \to \mathtt{a}B \mid \mathtt{b}C \mid \mathtt{c}A \mid \lambda$$
$$C \to \mathtt{a}B \mid \mathtt{b}A \mid \lambda$$

AF a partir de GR

Toda gramática regular gera uma linguagem regular.

Seja $G = (V, \Sigma, R, P)$ e $Z \notin V$.
Um AFN que reconhece L(G): $M = (E, \Sigma, \delta, \{P\}, F)$, em que

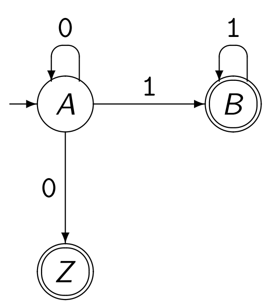
- $E = \begin{cases} V \cup \{Z\} & \text{se } R \text{ contém regra da forma } X \to a \\ V & \text{caso contrário.} \end{cases}$
- Para toda regra da forma:
  - $X \to aY$ faça $Y \in \delta(X, a)$,
  - $X \to a$ faça $Z \in \delta(X, a)$.
- $F = \begin{cases} \{X \mid X \to \lambda \in R\} \cup \{Z\} & \text{se } Z \in E \\ \{X \mid X \to \lambda \in R\} & \text{caso contrário.} \end{cases}$

Exemplo

G:

$$A \to 0A|1B|0$$
$$B \to 1B|\lambda$$

$L(G) = 0^*(0 + 1^+)$. AFN para $L(G)$:



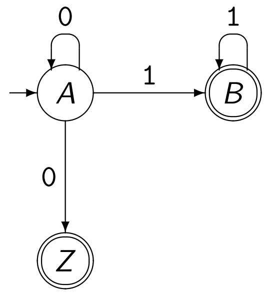
Gramáticas Regulares

Toda linguagem regular é gerada por gramática regular.

Seja um AFN $M = (E, \Sigma, \delta, \{i\}, F)$.
Uma GR que gera $L(M)$: $G = (E, \Sigma, R, i)$, em que:

$$R = \{e \to ae' \mid e' \in \delta(e, a)\} \cup \{e \to \lambda \mid e \in F\}.$$

Exemplo



GR:

$A \to 0A|0Z|1B$

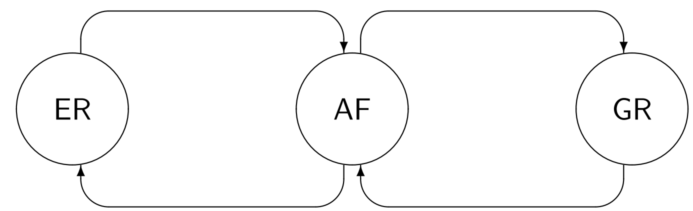
$B \to 1B|\lambda$

$Z \to \lambda$

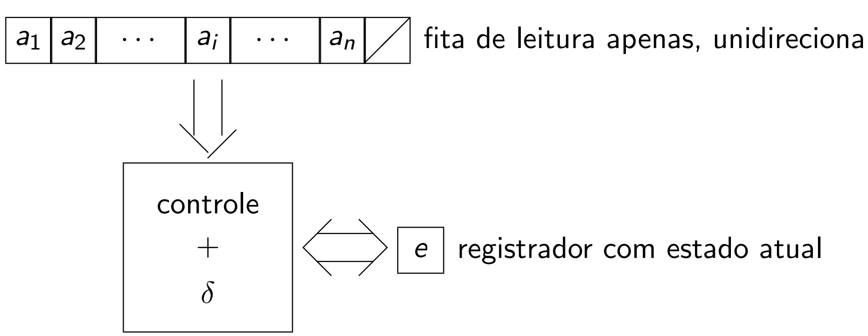
Uma síntese

$\Longrightarrow$ ERs, GRs e AFs são formalismos alternativos para linguagens regulares.

Transformações entre formalismos:

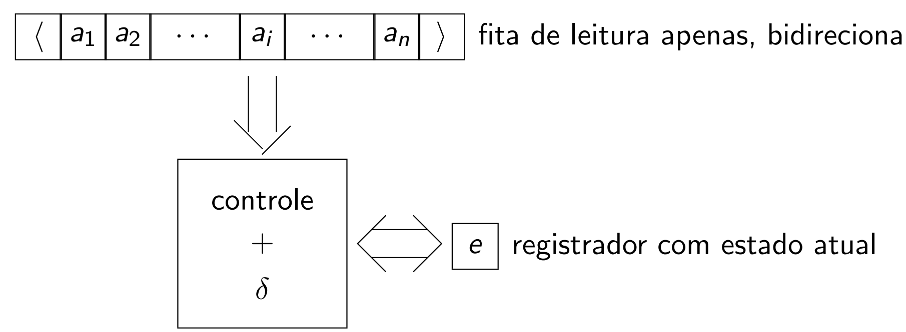


AFD visto como um computador



AFD com fita bidirecional

Função de transição **parcial** da forma $\delta : E \times \Sigma \to E \times \{\mathtt{e}, \mathtt{d}\}$.



$L(M)$: toda palavra $w \in \Sigma^*$ para a qual M para em um estado final (se M não parar para w, w não é aceita).

Expressividade de AFD bidirecional

- AFDs bidirecionais reconhecem exatamente as linguagens regulares.
- Mesmo com não determinismo, um AF bidirecional reconhece apenas linguagens regular.
- Que "incrementos" poderiam aumentar o poder computacional?