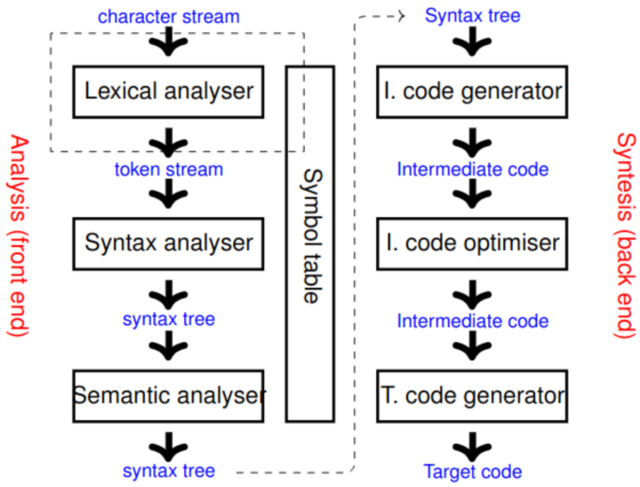
Compiladores - a partir do Cap. 5

José Mendes 107188

2023

1 Linguagens, Expressões e Gramáticas Regulares

1.1 Papel da análise lexical



Converte a sequência de caracteres numa sequência de **tokens**.
Um **token** é um tuplo: <**token-name, attrinute-value**>.

- token-name é um símbolo (abstrato), que representa um tipo de entrada;
- attribute-value representa o valor corrente desse símbolo;

Exemplo:

$$pos = pos + vel * 5$$

é convertido em:

$$< ID, "pos" ><=>< ID, "pos" >< + >< ID, "vel" >< * >< INT, 5 >$$

Tipicamente, alguns símbolos são descartados pelo analisador lexical.
O conjunto dos tokens corresponde a uma linguagem regular

- os tokens são descritos usando expressões regulares e/ou gramáticas regulares;
- são reconhecidos usando autómatos finitos;

1.2 Linguagem Regular

A classe das **linguagens regulares** sobre o alfabeto A define-se indutivamente da seguinte forma:

- O conjunto vazio, $\emptyset$, é uma linguagem regular (LR).
- Qualquer que seja o $a \in A$, o conjunto $\{a\}$ é uma LR.

  Nota:

  - em $a \in A$, a é uma letra do alfabeto
  - em $\{a\}$, a é uma palavra com apenas uma letra
  - Numa analogia Java, o primeiro é um 'a' e o segundo um "a"

- Se L_1 e L_2 são LR, então a sua reunião $(L_1 \cup L_2)$ é uma LR.

  Exemplo:

  - Seja $L_1 = \{ab, c\}$, uma LR sobre o alfabeto $A = \{a, b, c\}$
  - e $L_2 = \{bb, c\}$, outra LR sobre o mesmo alfabeto A
  - então, $L_3 = L_1 \cup L_2 = \{ab, bb, c\}$ é uma LR sobre o mesmo alfabeto A

- Se L_1 e L_2 são LR, então a sua concatenação $(L_1 \cdot L_2)$ é uma LR.

  Exemplo:

  - Seja $L_1 = \{ab, c\}$, uma LR sobre o alfabeto $A = \{a, b, c\}$
  - e $L_2 = \{bb, c\}$, outra LR sobre o mesmo alfabeto A
  - então, $L_3 = L_1 \cdot L_2 = \{abbb, abc, cbb, cc\}$ é uma LR sobre o mesmo alfabeto A

- Se L_1 é uma LR, então o seu fecho de Kleene $(L_1)^*$ é uma LR.

  Exemplo:

  - Seja $L_1 = \{ab, c\}$, uma LR sobre o alfabeto $A = \{a, b, c\}$
  - então, $L_2 = L_1^* = \{\varepsilon, ab, c, abab, abc, cab, cc, \dots\}$ é uma LR sobre o mesmo alfabeto A

- Nada mais é LR.

  Nota:

  - $\{\varepsilon\}$ é uma LR, uma vez que $\{\varepsilon\} = \emptyset$

1.3 Definição de linguagem regular

$\mathcal{Q}$ Mostre que a linguagem L, constituída pelo conjunto dos números binários começados em 1 e terminados em 0 é uma LR sobre o alfabeto $A = \{0, 1\}$

$\mathcal{R}$

- pela regra 2 (elementos primitivos), $\{0\}$ e $\{1\}$ são LR
- pela regra 3 (união), $\{0, 1\} = \{0\} \cup \{1\}$ é uma LR
- pela regra 5 (fecho), $\{0, 1\}^*$ é uma LR
- pela regra 4 (concatenação), $\{1\} \cdot \{0, 1\}^*$ é uma LR
- pela regra 4, $(\{1\} \cdot \{0, 1\}^*) \cdot \{0\}$ é uma LR
- logo, $L = \{1\} \cdot \{0, 1\}^* \cdot \{0\}$ é uma LR

1.4 Expressões regulares

O conjunto das **expressões regulares** sobre o alfabeto A define-se indutivamente da seguinte forma:

- $\emptyset$ é uma expressão regular (ER) que representa LR $\{\}$.
- Qualquer que seja o $\alpha \in A$, α é uma ER que representa a LR $\{\alpha\}$
- Se e_1 e e_2 são ER representando respetivamente as LR L_1 e L_2, então $(e_1|e_2)$ é uma ER representando a LR $L_1 \cup L_2$.
- Se e_1 e e_2 são ER representando respetivamente as LR L_1 e L_2, então (e_1e_2) é uma ER representando a LR $L_1 \cdot L_2$.
- Se e_1 é uma ER representando a LR L_1, então $(e_1)^*$ é uma ER representando a LR $(L_1)^*$
- Nada mais é uma expressão regular.

Nota: É habitual representar-se por ε a ER $\emptyset^*$. Representa a linguagem $\{\varepsilon\}$

Na escrita de expressões regulares assume-se a seguinte precedência dos operadores:

- fecho (*)
- concatenação
- escolha (|)

O uso destas precedências permite a queda de alguns parêntesis e consequentemente uma notação simplificada.

Exemplo: $e_1|e_2e_3^*$ recorre a precedência para representar a expressão regular: $(e_1)|(e_2((e_3)^*))$

Exemplos:

> $\mathcal{Q}$ Determine uma ER que represente o conjunto dos números binários começados em `1` e terminados em `0`.
>
> $\mathcal{R}$ `1(0|1)*0`

> $\mathcal{Q}$ Determine uma ER que represente as sequências definidas sobre o alfabeto $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ que satisfazem o requisito de qualquer `b` ter um `a` imediatamente à sua esquerda e um `c` imediatamente à sua direita.
>
> $\mathcal{R}$ O `a` pode aparecer sozinho; o `c` também; o `b`, se aparecer, tem de ter um `a` à sua esquerda e um `c` à sua direita. Ou seja, pode considerar-se que as palavras da linguagem são sequências de 0 ou mais `a`, `c` ou `abc`.
>
> `(a|abc|c)*`

> $\mathcal{Q}$ Determine uma ER que represente as sequências binárias com um número par de zeros.
>
> $\mathcal{R}$ `(1*01*01*)*|1*` $=$ `1*(01*01*)*`

A operação de escolha goza das propriedades:

- comutativa: $e_1|e_2 = e_2|e_1$
- associativa: $e_1|(e_2|e_3) = (e_1|e_2)|e_3 = e_1|e_2|e_3$
- idempotência: $e_1|e_1 = e_1$
- existência de elemento neutro: $e_1|\emptyset = \emptyset|e_1 = e_1$

A operação de concatenação goza das propriedades:

- associativa: $e_1(e_2e_3) = (e_1e_2)e_3 = e_1e_2e_3$
- existência de elemento neutro: $e_1\varepsilon = \varepsilon e_1 = e_1$
- existência de elemento absorvente $e_1\emptyset = \emptyset e_1 = \emptyset$
- **não goza da propriedade comutativa**

A combinação das operações de concatenação e escolha gozam das propriedades:

- distributiva à esquerda da concatenação em relação à escolha:

$$e_1|(e_2|e_3) = e_1e_2|e_1e_3$$

- distributiva à direita da concatenação em relação à escolha:

$$(e_1|e_2)e_3 = e_1e_3|e_2e_3$$

A operação de fecho goza das propriedades:

- $(e^*)^* = e^*$
- $(e_1^*|e_2^*)^* = (e_1|e_2)^*$
- $(e_1|e_2^*)^* = (e_1|e_2)^*$
- $(e_1^*|e_2)^* = (e_1|e_2)^*$

Mas **atenção**:

- $(e_1|e_2)^* \neq e_1^*|e_2^*$
- $(e_1 e_2)^* \neq e_1^* e_2^*$

Exemplos:

$\mathcal{Q}$ Sobre o alfabeto $A = \{0, 1\}$ construa uma expressão regular que represente a linguagem

$$L = \{\omega \in A^* \ : \ \#(0, \omega) = 2\}$$

$\mathcal{R}$ $1^*01^*01^*$

$\mathcal{Q}$ Sobre o alfabeto $A = \{a, b, \cdots, z\}$ construa uma expressão regular que represente a linguagem

$$L = \{\omega \in A^* \ : \ \#(a, \omega) = 3\}$$

$\mathcal{R}$ $(b|c|\cdots|z)^*a(b|c|\cdots|z)^*a(b|c|\cdots|z)^*a(b|c|\cdots|z)^*$

Na última resposta, onde estão as reticências $(\ldots)$ deveriam estar todas as letras entre d e y. Parece claro que faz falta uma forma de simplificar este tipo de expressões

1.4.1 Extensões notacionais comuns

$\rightarrow$ Uma ou mais ocorrências: $e^+ = e \cdot e^*$

$\rightarrow$ Uma ou nenhuma ocorrência: $e? = (e|\varepsilon)$

$\rightarrow$ Um símbolo do sub-alfabeto dado: $[a_1 a_2 a_3 \ldots a_n] = (a_1|a_2|a_3|\ldots|a_n)$

$\rightarrow$ Um símbolo do sub-alfabeto dado: $[a_1 - a_n] = (a_1|\ldots|a_n)$

$\rightarrow$ Um símbolo do alfabeto fora do conjunto dado: $[\hat{\ } a_1 a_2 a_3 \ldots a_n]$, $[\hat{\ } a_1 - a_n]$

$\rightarrow$ n ocorrência de: $e\{n\} = \underbrace{e.e.\ldots.e}_{n}$

$\rightarrow$ de n_1 a n_2 ocorrências: $e\{n_1, n_2\} = \underbrace{e.e.\ldots.e}_{n_1, n_2}$

$\rightarrow$ n ou mais ocorrências: $e\{n,\} = \underbrace{e.e.\ldots.e}_{n,}$

$\rightarrow$ "." representa um símbolo Qualquer

→ ”^” representa palavra vazia no início de linha

→ ”$” representa palavra vazia no fim de linha

→ ”\ <” representa palavra vazia no ínicio de palavra

→ ”\ >” representa palavra vazia no fim de palavra

Exemplo:

$\mathcal{Q}$ Sobre o alfabeto $A = \{0, 1\}$ construa uma expressão regular que reconheça a linguagem

$$L = \{\omega \in A^* \ : \ \#(0, \omega) = 2\}$$

$\mathcal{R}$ $1^*01^*01^* = (1^*0)\,(1^*0)\,1^* = (1^*0)\{2\}\,1^*$

$\mathcal{Q}$ Sobre o alfabeto $A = \{a, b, \cdots, z\}$ construa uma expressão regular que reconheça a linguagem

$$L = \{\omega \in A^* \ : \ \#(a, \omega) = 3\}$$

$\mathcal{R}$ $(b|c|\cdots|z)^*a(b|c|\cdots|z)^*a(b|c|\cdots|z)^*a(b|c|\cdots|z)^*$

$= ([b\text{-}z]^*a)\,([b\text{-}z]^*a)\,([b\text{-}z]^*a)\,[b\text{-}z]^*$

$= ([b\text{-}z]^*a)\{3\}[b\text{-}z]^*$

1.5 Gramáticas regulares

Exemplo de gramática regular:

$$\begin{aligned} S &\to a\ X \\ X &\to a\ X \\ &\mid b\ X \\ &\mid \varepsilon \end{aligned}$$

Exemplo de gramática <u>não</u> regular:

$$\begin{aligned} S &\to a\ S\ a \\ X &\to b\ S\ b \\ &\mid a \end{aligned}$$

Letras minúsculas representam símbolos terminais e letras maísculas representam símbolos não terminais (o contrário do ANTLR).

Nas gramáticas regulares os símbolos não terminais apenas podem aparecer no fim.

Uma **gramática regular** é um quádruplo $G = (T, N, P, S)$ onde:

- T é um conjunto finito não vazio de símbolos terminais;
- N, sendo $N \cap T = \emptyset$, é um conjunto finito não vazio de símbolos não terminais;
- P é um conjunto de produções (ou regra de rescrita), cada uma da forma $\alpha \to \beta$, onde;
  - $\alpha \in N$
  - $\beta \in T^* \cup T^* N$
- $S \in N$ é o símbolo inicial.
- A linguagem gerada por uma gramática regular é regular.

  Logo, é possível converter-se uma gramática regular numa expressão regular que represente a mesma linguagem e vice-versa.

1.5.1 Operações sobre gramáticas regulares

As gramáticas regulares são fechadas sob as operações de:

- reunião
- concatenação
- fecho
- interseção
- complementação

As operações de interseção e complementação serão abordadas mais adiante através de autómatos finitos

1.5.2 Reunião de gramáticas regulares

Exemplo:

$\mathcal{Q}$ Sobre o conjunto de terminais $T = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, determine uma gramática regular que represente a linguagem

$$L = L_1 \cup L_2$$

sabendo que

$$L_1 = \{\omega\mathtt{a} \;:\; \omega \in T^*\} \qquad L_2 = \{\mathtt{a}\omega \;:\; \omega \in T^*\}$$

$\mathcal{R}$

$$\begin{array}{lll} S_1 & \rightarrow & \mathtt{a}\ S_1 \\ & | & \mathtt{b}\ S_1 \\ & | & \mathtt{c}\ S_1 \\ & | & \mathtt{a} \end{array} \qquad \begin{array}{lll} S_2 & \rightarrow & \mathtt{a}\ X_2 \\ X_2 & \rightarrow & \mathtt{a}\ X_2 \\ & | & \mathtt{b}\ X_2 \\ & | & \mathtt{c}\ X_2 \\ & | & \varepsilon \end{array}$$

Comece-se por obter as gramáticas regulares que representam L_1 e L_2.

$\mathcal{Q}$ Sobre o conjunto de terminais $T = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, determine uma gramática regular que represente a linguagem

$$L = L_1 \cup L_2$$

sabendo que

$$L_1 = \{\omega\mathtt{a} \;:\; \omega \in T^*\} \qquad L_2 = \{\mathtt{a}\omega \;:\; \omega \in T^*\}$$

$\mathcal{R}$

$$\begin{array}{lll} S_1 & \rightarrow & \mathtt{a}\ S_1 \\ & | & \mathtt{b}\ S_1 \\ & | & \mathtt{c}\ S_1 \\ & | & \mathtt{a} \end{array} \qquad \begin{array}{lll} S_2 & \rightarrow & \mathtt{a}\ X_2 \\ X_2 & \rightarrow & \mathtt{a}\ X_2 \\ & | & \mathtt{b}\ X_2 \\ & | & \mathtt{c}\ X_2 \\ & | & \varepsilon \end{array} \qquad \begin{array}{lll} S & \rightarrow & S_1 \ | \ S_2 \\ S_1 & \rightarrow & \mathtt{a}\ S_1 \ | \ \mathtt{b}\ S_1 \ | \ \mathtt{c}\ S_1 \\ & | & \mathtt{a} \\ S_2 & \rightarrow & \mathtt{a}\ X_2 \\ X_2 & \rightarrow & \mathtt{a}\ X_2 \ | \ \mathtt{b}\ X_2 \ | \ \mathtt{c}\ X_2 \\ & | & \varepsilon \end{array}$$

E acrescenta-se as transições $S \rightarrow S_1$ $S \rightarrow S_2$ que permitem escolher as palavras de L_1 e de L_2, sendo S o novo símbolo inicial.

Algoritmo:

$\mathcal{D}$ Sejam $G_1 = (T_1, N_1, P_1, S_1)$ e $G_2 = (T_2, N_2, P_2, S_2)$ duas gramáticas regulares quaisquer, com $N_1 \cap N_2 = \emptyset$. A gramática $G = (T, N, P, S)$ onde

$$\begin{array}{lll} T & = & T_1 \cup T_2 \\ N & = & N_1 \cup N_2 \cup \{S\} \quad \text{com} \quad S \notin (N_1 \cup N_2) \\ P & = & \{S \rightarrow S_1, S \rightarrow S_2\} \cup P_1 \cup P_2 \end{array}$$

é regular e gera a linguagem $L = L(G_1) \cup L(G_2)$.

Para $i = 1, 2$, a nova produção $S \rightarrow S_i$ permite que G gere a linguagem $L(G_i)$

1.5.3 Concatenação de gramáticas regulares

Exemplo:

$\mathcal{Q}$ Sobre o conjunto de terminais $T = \{a, b, c\}$, determine uma gramática regular que represente a linguagem

$L = L_1 \cdot L_2$

sabendo que

$L_1 = \{\omega a \ : \ \omega \in T^*\} \qquad L_2 = \{a\omega \ : \ \omega \in T^*\}$

$\mathcal{R}$

$$\begin{array}{llll} S_1 \to a\ S_1 & & S_2 \to a\ X_2 & \\ \quad \mid b\ S_1 & & X_2 \to a\ X_2 & \\ \quad \mid c\ S_1 & & \quad \mid b\ X_2 & \\ \quad \mid a & & \quad \mid c\ X_2 & \\ & & \quad \mid \varepsilon & \end{array}$$

Comece-se por obter que representam L_1 e L_2.

$\mathcal{Q}$ Sobre o conjunto de terminais $T = \{a, b, c\}$, determine uma gramática regular que represente a linguagem

$L = L_1 \cdot L_2$

sabendo que

$L_1 = \{\omega a \ : \ \omega \in T^*\} \qquad L_2 = \{a\omega \ : \ \omega \in T^*\}$

$\mathcal{R}$

$$\begin{array}{lll} S_1 \to a\ S_1 & S_2 \to a\ X_2 & S_1 \to a\ S_1 \mid b\ S_1 \mid c\ S_1 \\ \quad \mid b\ S_1 & X_2 \to a\ X_2 & \quad \mid a\ S_2 \\ \quad \mid c\ S_1 & \quad \mid b\ X_2 & S_2 \to a\ X_2 \\ \quad \mid a & \quad \mid c\ X_2 & X_2 \to a\ X_2 \mid b\ X_2 \mid c\ X_2 \\ & \quad \mid \varepsilon & \end{array}$$

A seguir substitui-se $S_1 \to a$ por $S_1 \to a\ S_2$, de modo a impor que a segunda parte das palavras têm de pertemcer a L_2.

Algoritmo:

$\mathcal{D}$ Sejam $G_1 = (T_1, N_1, P_1, S_1)$ e $G_2 = (T_2, N_2, P_2, S_2)$ duas gramáticas regulares quaisquer, com $N_1 \cap N_2 = \emptyset$. A gramática $G = (T, N, P, S)$ onde

$$\begin{array}{rcl} T & = & T_1 \cup T_2 \\ N & = & N_1 \cup N_2 \\ P & = & \{A \to \omega S_2 \ : \ (A \to \omega) \in P_1 \ \wedge \ \omega \in {T_1}^*\} \\ & & \cup \{A \to \omega \ : \ (A \to \omega) \in P_1 \ \wedge \ \omega \in {T_1}^* N_1\} \\ & & \cup P_2 \\ S & = & S_1 \end{array}$$

é regular e gera a linguagem $L = L(G_1) \cdot L(G_2)$.

As produções da primeira gramática do tipo $\beta \in T^*$ ganham o símbolo inicial da segunda gramática no fim.

As produções da primeira gramática do tipo $\beta \in T^* N$ mantêm-se inalteradas.

As produções da segunda gramática mantêm-se inalteradas.

1.5.4 Fecho de gramáticas regulares

$\mathcal{Q}$ Sobre o conjunto de terminais $T = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, determine uma gramática regular que represente a linguagem

$$L = {L_1}^*$$

sabendo que

$$L_1 = \{\omega\mathtt{a} \,:\, \omega \in T^*\}$$

$\mathcal{R}$

$$\begin{aligned} S_1 &\rightarrow \mathtt{a}\ S_1 \\ &\mid \mathtt{b}\ S_1 \\ &\mid \mathtt{c}\ S_1 \\ &\mid \mathtt{a} \end{aligned}$$

Começa-se pela obtenção da gramática regular que representa L_1.

$\mathcal{Q}$ Sobre o conjunto de terminais $T = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, determine uma gramática regular que represente a linguagem

$$L = {L_1}^*$$

sabendo que

$$L_1 = \{\omega\mathtt{a} \,:\, \omega \in T^*\}$$

$\mathcal{R}$

$$\begin{aligned} S_1 &\rightarrow \mathtt{a}\ S_1 \\ &\mid \mathtt{b}\ S_1 \\ &\mid \mathtt{c}\ S_1 \\ &\mid \mathtt{a} \end{aligned} \qquad \begin{aligned} S &\rightarrow \varepsilon \mid S_1 \\ S_1 &\rightarrow \mathtt{a}\ S_1 \mid \mathtt{b}\ S_1 \mid \mathtt{c}\ S_1 \\ &\mid \mathtt{a}\ S \end{aligned}$$

Acrescentando-se a transição $S \rightarrow S_1$ e substituindo-se $S_1 \rightarrow a$ por $S_1 \rightarrow a\ S$, permite-se iterações sobre S_1.

Acrescentando-se $S \rightarrow \varepsilon$, permite-se 0 ou mais iterações.

Algoritmo:

$\mathcal{D}$ Seja $G_1 = (T_1, N_1, P_1, S_1)$ uma gramática regular qualquer. A gramática $G = (T, N, P, S)$ onde

$$\begin{aligned} T &= T_1 \\ N &= N_1 \cup \{S\} \quad \text{com} \quad S \notin N_1 \\ P &= \{S \rightarrow \varepsilon, S \rightarrow S_1\} \\ &\quad \cup \{A \rightarrow \omega S \,:\, (A \rightarrow \omega) \in P_1 \wedge \omega \in {T_1}^*\} \\ &\quad \cup \{A \rightarrow \omega \,:\, (A \rightarrow \omega) \in P_1 \wedge \omega \in {T_1}^* N_1\} \end{aligned}$$

é regular e gera a linguagem $L = (L(G_1))^*$.

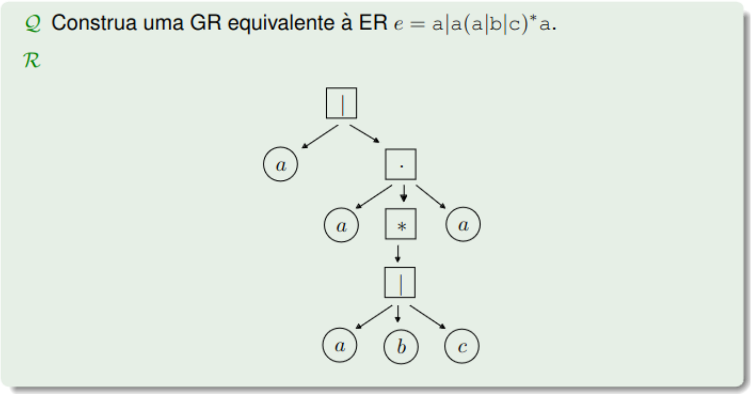
As novas produções $S \rightarrow \varepsilon$ e $S \rightarrow S_1$ garantem que $(L(G_1))^* \subseteq L(G)$, para qualquer $n \geq 0$.

As produções que só têm terminais ganham o novo símbolo inicial no fim.

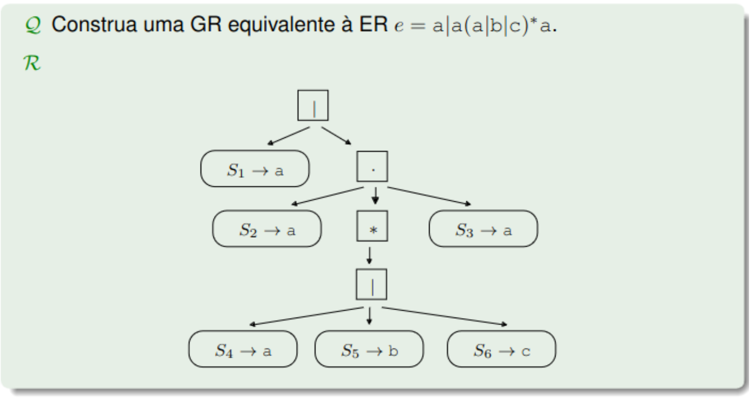
As produções que terminam num não terminal mantêm-se inalteradas.

1.6 Conversão de uma ER em um GR

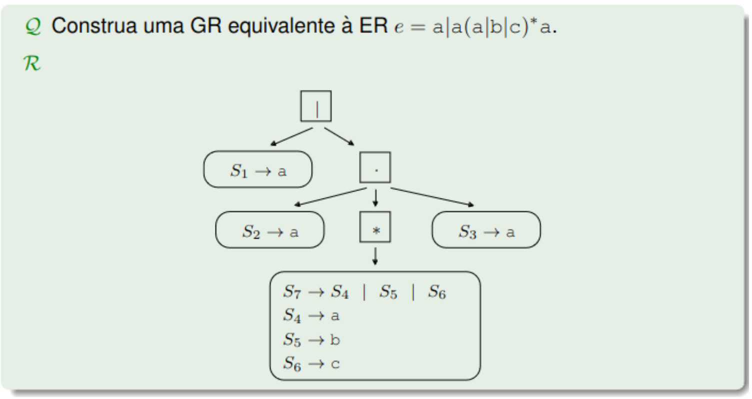
1.6.1 Exemplo



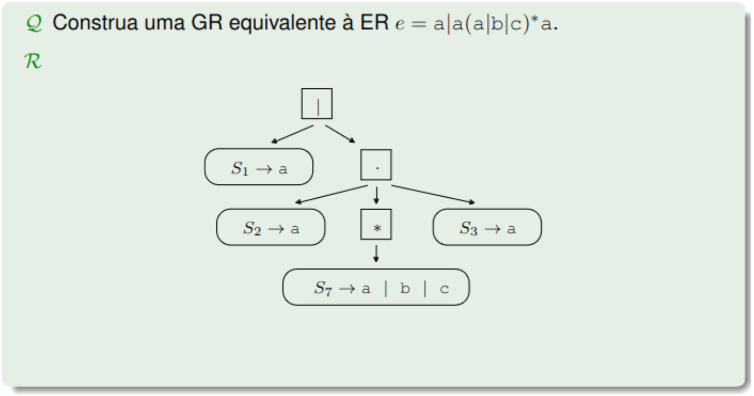
Coloque-se de forma arbórea.



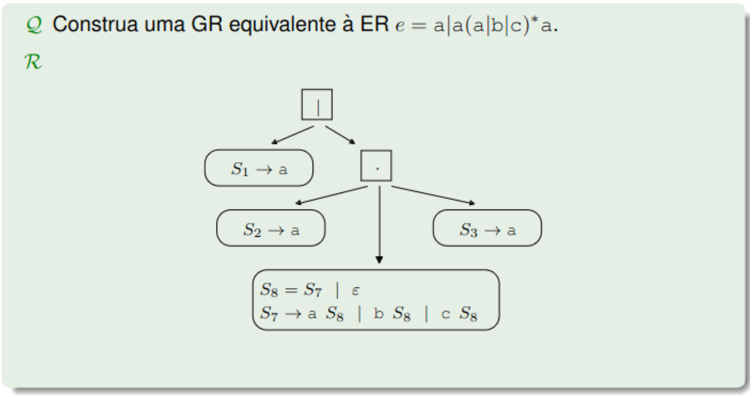
Após converter as folhas (elementos primitivos) em GR.



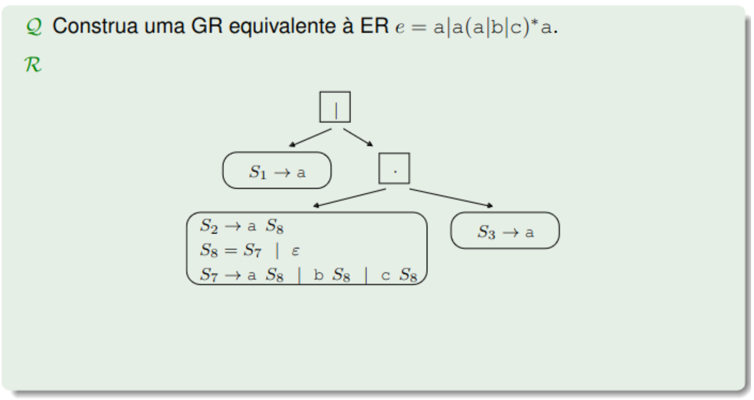
Após aplicar a escolha (reunião) de baixo.



Simplificando.



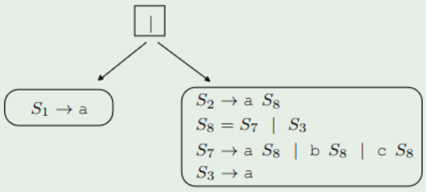
Após aplicar o fecho.



Após aplicar a concatenação.

$\mathcal{Q}$ Construa uma GR equivalente à ER $e = \mathsf{a}|\mathsf{a}(\mathsf{a}|\mathsf{b}|\mathsf{c})^*\mathsf{a}$.

$\mathcal{R}$



Após aplicar a concatenação da direita.

$\mathcal{Q}$ Construa uma GR equivalente à ER $e = \mathsf{a}|\mathsf{a}(\mathsf{a}|\mathsf{b}|\mathsf{c})^*\mathsf{a}$.

$\mathcal{R}$

$$
\begin{aligned}
&S \to S_1 \mid S_2 \\
&S_1 \to \mathsf{a} \\
&S_2 \to \mathsf{a}\ S_8 \\
&S_8 \to S_7 \mid S_3 \\
&S_7 \to \mathsf{a}\ S_8 \mid \mathsf{b}\ S_8 \mid \mathsf{c}\ S_8 \\
&S_3 \to \mathsf{a}
\end{aligned}
$$

e simplificando

$$
\begin{aligned}
&S \to \mathsf{a} \mid \mathsf{a}\ S_8 \\
&S_8 \to \mathsf{a}\ S_8 \mid \mathsf{b}\ S_8 \mid \mathsf{c}\ S_8 \mid \mathsf{a}
\end{aligned}
$$

Finalmente após aplicar escolha (reunião) de cima.

1.6.2 Abordagem

Dada uma expressão regular qualquer ela é:

- ou um elemento primitivo;
- ou uma expressão do tipo e^*, sendo e uma expressão regular qualquer;
- ou uma expressão do tipo $e_1 \cdot e_2$, sendo e_1 e e_2 duas expressões regulares quaisquer;
- ou uma expressão do tipo $e_1 \mid$, sendo e_1 e e_2 duas expressões regulares quaisquer.

Identificando-se as GR equivalentes às ER primitivas, tem-se o problema resolvido, visto que se sabe como fazer a reunião, a concatenação e o fecho de GR.

expressão regular	gramática regular
ε	$S \to \varepsilon$
a	$S \to \mathsf{a}$

1.6.3 Algoritmo de conversão

1. Se ER é do tipo primitivo, a GR correspondente pode ser obtido da tabela anterior.
2. Se é do tipo e^*, aplica-se este mesmo algoritmo na obtenção de uma GR equivalente à expressão regular e e, de seguida,, aplica-se o fecho de GR.
3. Se é do tipo $e_1 \cdot e_2$, aplica-se este mesmo algoritmo na obtenção de uma GR para as expressões e_1 e e_2, de seguida, aplica-se a concatenação de GR.
4. Finalmente, se é do tipo $e_1 \mid e_2$, aplica-se este mesmo algoritmo na obtenção de GR para as expressões e_1 e e_2, de seguida, aplica-se a reunião de GR.

(Na realidade, o algoritmo corresponde a um processo de decomposição arbórea a partir da raiz seguido de um processo de construção arbórea a partir das folhas).

1.7 Conversão de uma GR em uma ER

1.7.1 Exemplo

$\mathcal{Q}$ Obtenha uma ER equivalente à gramática regular seguinte

$S \to \mathrm{a}\ S \mid \mathrm{c}\ S \mid \mathrm{aba}\ X$
$X \to \mathrm{a}\ X \mid \mathrm{c}\ X \mid \varepsilon$

$\mathcal{R}$ Abordagem admitindo expressões regulares nas produções das gramáticas

$E \to \varepsilon\ S$
$S \to \mathrm{a}\ S \mid \mathrm{c}\ S \mid (\mathrm{aba})\ X$
$X \to \mathrm{a}\ X \mid \mathrm{c}\ X \mid \varepsilon\ \varepsilon$

- *acrescentou-se um novo símbolo inicial de forma a garantir que não aparece do lado direito*

$E \to \varepsilon\ S$
$S \to (\mathrm{a|c})\ S \mid (\mathrm{aba})\ X$
$X \to (\mathrm{a|c})\ X \mid \varepsilon\ \varepsilon$

- *transformou-se* $S \to \mathrm{a}\ S$ *e* $S \to \mathrm{c}\ S$ *em* $S \to (\mathrm{a|c})\ S$
- *fez-se algo similar com o X*

$E \to \varepsilon\ (\mathrm{a|c})^*\ (\mathrm{aba})\ X$
$X \to (\mathrm{a|c})\ X \mid \varepsilon\ \varepsilon$

- *transformaram-se as produções* $E \to \varepsilon\ S$, $S \to (\mathrm{a|c})\ S$ *e* $S \to \mathrm{aba}\ X$ *em* $E \to (\mathrm{a|c})^*\mathrm{aba}\ X$
- *Note que o* $(\mathrm{a|c})$ *passou a* $(\mathrm{a|c})^*$

$E \to \varepsilon\ (\mathrm{a|c})^*\ (\mathrm{aba})\ (\mathrm{a|c})^*\ \varepsilon$

- *repetiu-se com o* X, *obtendo-se a ER desejada:* $(\mathrm{a|c})^*\mathrm{aba}(\mathrm{a|c})^*$

$\mathcal{Q}$ Obtenha uma ER equivalente à gramática regular seguinte

$S \to \mathrm{a}\ S \mid \mathrm{c}\ S \mid \mathrm{aba}\ X$
$X \to \mathrm{a}\ X \mid \mathrm{c}\ X \mid \varepsilon$

$\mathcal{R}$ Abordagem transformando a gramática num conjunto e triplos

$\{(E,\varepsilon,S),$
$(S,\mathrm{a},S),(S,\mathrm{c},S),(S,\mathrm{aba},X),$
$(X,\mathrm{a},X),(X,\mathrm{c},X),(X,\varepsilon,\varepsilon)\}$

- *converte-se a gramática num conjunto de triplos, acrescentando um inicial*

$\{(E,\varepsilon,S),(S,(\mathrm{a|c}),S),(S,\mathrm{aba},X),$
$(X,(\mathrm{a|c}),X),(X,\varepsilon,\varepsilon)\}$

- *transformou-se* $(S,\mathrm{a},S),(S,\mathrm{c},S)$ *em* $(S,(\mathrm{a|c}),S)$
- *fez-se algo similar com o X*

$\{(E,(\mathrm{a|c})^*\mathrm{aba},X),$
$(X,(\mathrm{a|c}),X),(X,\varepsilon,\varepsilon)\}$

- *transformou-se o triplo de triplos* $(E,\varepsilon,S),(S,(\mathrm{a|c}),S),(S,\mathrm{aba},X)$ *em* $(E,(\mathrm{a|c})^*\mathrm{aba},X)$
- *Note que o* $(\mathrm{a|c})$ *passou a* $(\mathrm{a|c})^*$

$\{(E,(\mathrm{a|c})^*\mathrm{aba}(\mathrm{a|c})^*,\varepsilon)\}$

- *repetiu-se com o* X, *obtendo-se a ER desejada:* $(\mathrm{a|c})^*\mathrm{aba}(\mathrm{a|c})^*$

1.7.2 Algoritmo

- Uma expressão regular e que represente a mesma linguagem que a gramática regular G pode ser obtida por um processo de transformações de equivalência.
- Primeiro, converte-se a gramática $G = (T, N, P, S)$ no conjunto de triplos seguinte:

$$\begin{aligned} \mathcal{E} &= \{(E, \varepsilon, S)\} \\ &\cup \{(A, \omega, B) : (A \to \omega B) \in P \wedge B \in N\} \\ &\cup \{(A, \omega, \varepsilon) : (A \to \omega) \in P \wedge \omega \in T^*\} \end{aligned}$$

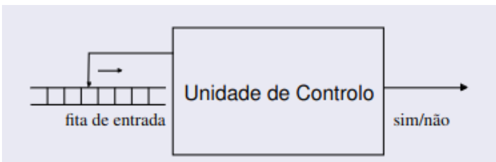
  com $E \notin N$.
- A seguir, removem-se, por transformações de equivalência, um a um, todos os símbolos de N, até se obter um único triplo da forma (E, e, ε).
- O valor de e é a expressão regular pretendida.

1. Substituir todos os triplos da forma (A, α_i, A), com $A \in N$, por um único (A, ω_2, A), onde $\omega_2 = \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_m$
2. Substituir todos os triplos da forma (A, β_i, B), com $A, B \in N$, por um único (A, ω_1, B), onde $\omega_1 = \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$
3. Substituir cada triplo de triplos da forma $(A, \omega_1, B), (B, \omega_2, B), (B, \omega_3, C)$, com $A, B, C \in N$, pelo triplo $(A, \omega_1 \omega_2{}^* \omega_3, C)$
4. Repetir os passos anteriores enquanto houver símbolos intermédios

Note que, se não existir qualquer triplo do tipo (A, α_i, A), w_2 representa o consjunto vazio e consequentemente $w_2^* = \varepsilon$

2 Atómatos Finitos

Autómato Finito - é um mecanismo reonhecedor das palavras de uma linguagem regular.

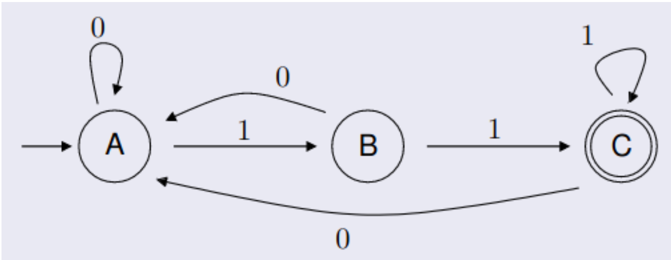


A unidade de controlo é baseada nas noções de estado e de transição entre estados (número finito de estados).

A fita de entrada é só de leitura, com acesso sequencial.

Os autómatos finitos podem ser **deterministas**, **não deterministas** ou **generalizados**.

2.1 Autómato finito determinista

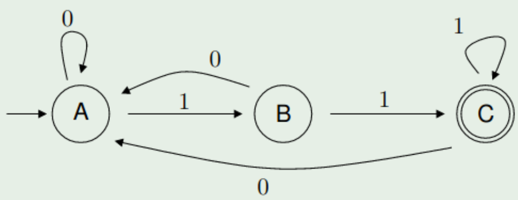


Um **autómato finito determinista** é um autómato finito onde:

- as transições estão associadas a símbolos individuais do alfabeto;
- de cada estado sai **uma e uma só** transição por cada símbolo do alfabeto;
- há um estado inicial;
- há 0 ou mais estados de aceitação, que determinam as palavras aceites;
- os caminhos que começam no estado inicial e terminam num estado de aceitação representam as palavras aceites (reconhecidas) pelo autómato.

2.1.1 Exemplos

$\mathcal{Q}$ Que palavras binárias são reconhecidas pelo autómato seguinte?

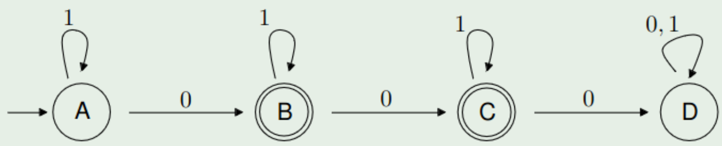


$\mathcal{R}$ Todas as palavras terminadas em 11.

A expressão seria:

$$(0 \mid 1)^* \; 11$$

$\mathcal{Q}$ Que palavras binárias são reconhecidas pelo autómato seguinte?

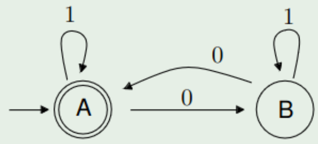


$\mathcal{R}$ Todas as palavras com apenas 1 ou 2 zeros.

A expressão seria (acho eu):

$$1^* \; 0 \; 1^* \; 0 \; 1^*$$

Q Que palavras binárias são reconhecidas pelo autómato seguinte?



R as sequências binárias com um número par de zeros.

A expressão seria (acho eu):

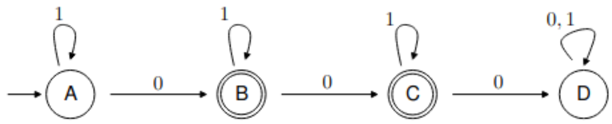
$$(1^*\ 0\ 1^*\ 0\ 1^*)^*$$

2.1.2 Definição de um autómato finito determinista

Um autómato finito determinista (AFD) é um quíntuplo $M = (A, Q, q_0, \delta, F)$, em que:

- A é o alfabeto de entrada;
- Q é um conjunto finito não vazio de estados;
- $q_0 \in Q$ é o estado inicial;
- $\delta: \ Q \times A \to Q$ é uma função que determina a transição entre estados;
- $F \subseteq Q$ é o conjunto dos estados de aceitação.

Exemplo:

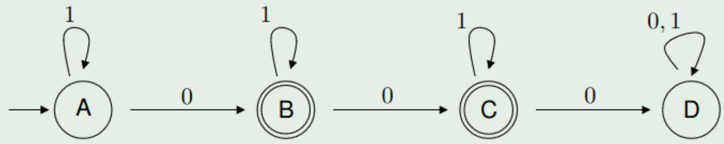


$$\begin{aligned}
&A = \{0, 1\}\\
&Q = \{A, B, C, D\}\\
&q_0 = A\\
&F\{B, C\}
\end{aligned}$$

Q Como representar a função δ ?

- Matriz de $|Q|$ linhas por $|A|$ colunas. As células contêm elementos de Q.
- Conjunto de pares $((q, a), q) \in (Q \times A) \times Q$
  - ou equivalentemente conjunto de triplos $(q, a, q) \in Q \times A \times Q$

$\mathcal{Q}$ Represente textualmente o AFD seguinte.



$\mathcal{R}$

$M = (A, Q, q_0, \delta, F)$ com

- $A = \{0, 1\}$
- $Q = \{A, B, C, D\}$
- $q_0 = A$
- $F = \{B, C\}$

- $\delta = \{$
  $(A, 0, B), (A, 1, A),$
  $(B, 0, C), (B, 1, B),$
  $(C, 0, D), (C, 1, C),$
  $(D, 0, D), (D, 1, D)\}$

- $\delta =$

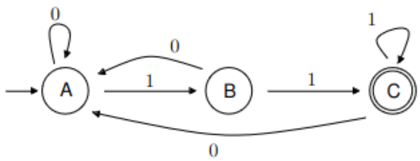
	0	1
A	B	A
B	C	B
C	D	C
D	D	D

2.1.3 Linguagem reconhecida por um AFD

Diz-se que um AFD $M = (A, Q, q_0, \delta, F)$, **aceita** uma palavra $u \in A^*$ se u se puder escrever na forma $u = u_1 u_2 \dots u_n$ e existir uma sequência de estados $s_0, s_1, \dots, s_n$, que satisfaça as seguintes condições:

- $s_0 = q_0^*$;
- qualquer que seja o $i = 1, \dots, n$, $s_i = \delta(s_{i-1}, u_i)$;
- $s_n \in F$.

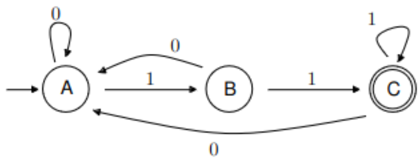
Caso contrário diz-se que M **rejeita** a sequência de entrada.



- A palavra $\omega_1 =$ `0101` faz o caminho $A \xrightarrow{0} A \xrightarrow{1} B \xrightarrow{0} A \xrightarrow{1} B$
  - como B não é de aceitação, ω_1 não pertence à linguagem
- A palavra $\omega_2 =$ `0011` faz o caminho $A \xrightarrow{0} A \xrightarrow{0} A \xrightarrow{1} B \xrightarrow{1} C$
  - como C é de aceitação, ω_2 pertence à linguagem

- Seja $\delta^* : Q \times A^* \to Q$ a extensão de δ definida indutivamente por
  1. $\delta^*(q, \varepsilon) = q$
  2. $\delta^*(q, av) = \delta^*(\delta(q, a), v)$, com $a \in A \wedge v \in A^*$
- M aceita u se $\delta^*(q_0, u) \in F$.
- $L(M) = \{u \in A^* \; : \; M \text{ aceita } u\} = \{u \in A^* \; : \; \delta^*(q_0, u) \in F\}$

- $\delta^*(A, \mathtt{0101}) = \delta^*(\delta(A, \mathtt{0}), \mathtt{101}) = \delta^*(A, \mathtt{101})$
  $= \delta^*(\delta(A, \mathtt{1}), \mathtt{01}) = \delta^*(B, \mathtt{01})$
  $= \delta^*(\delta(B, \mathtt{0}), \mathtt{1}) = \delta^*(A, \mathtt{1}) = \delta^*(B, \varepsilon) = B$
- $\delta^*(A, \mathtt{0011}) = \delta^*(\delta(A, \mathtt{0}), \mathtt{011}) = \delta^*(A, \mathtt{011})$
  $= \delta^*(\delta(A, \mathtt{0}), \mathtt{11}) = \delta^*(A, \mathtt{11})$
  $= \delta^*(\delta(A, \mathtt{1}), \mathtt{1}) = \delta^*(B, \mathtt{1}) = \delta^*(C, \varepsilon) = C$



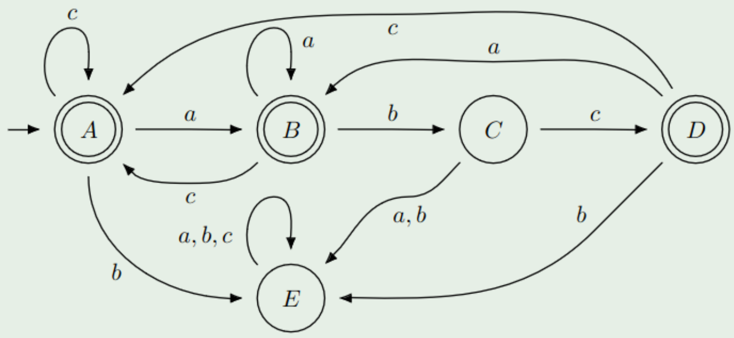
2.1.4 Linguagem para Autómato

$\mathcal{Q}$ Sobre o alfabeto $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ considere a linguagem

$$L = \{\omega \in A^* \; : \; (\omega_i = \mathtt{b}) \Rightarrow ((\omega_{i-1} = \mathtt{a}) \wedge (\omega_{i+1} = \mathtt{c}))\}$$

Projecte um autómato que reconheça L.

$\mathcal{R}$



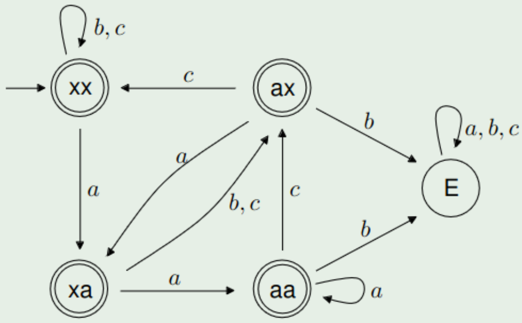
** Este aqui **

Q Sobre o alfabeto $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ considere a linguagem

$$L = \{\omega \in A^* \ : \ (\omega_i = \mathtt{a}) \ \Rightarrow \ (\omega_{i+2} \neq \mathtt{b})\}$$

Projecte um autómato que reconheça L.

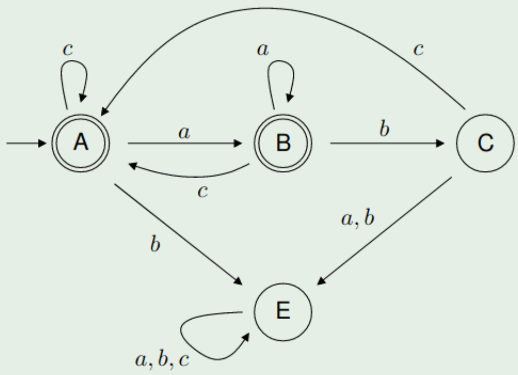
R



2.1.5 Redução de um AFD

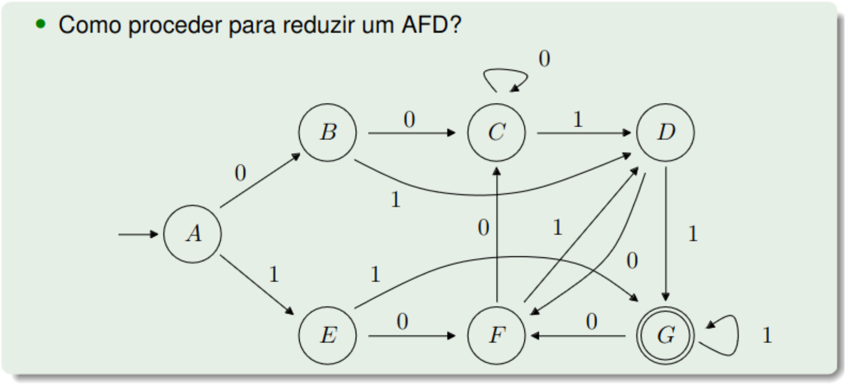
No exemplo a duas imagens a cima, comparando os estados A e C concluímos que as suas transições são iguais (para os mesmos estados), podemos dizer que são **equivalentes**. Por consequente podem ser fundidos.

- O que resulta em

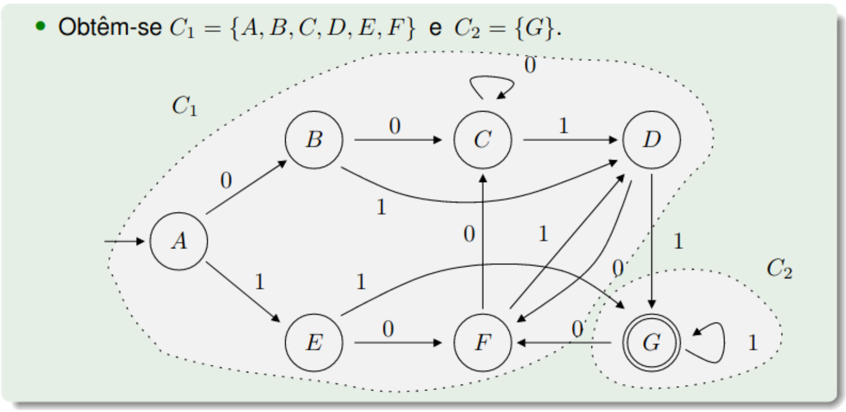


Este pode provar-se, no tem estados redundantes. Está no estado **reduzido**.

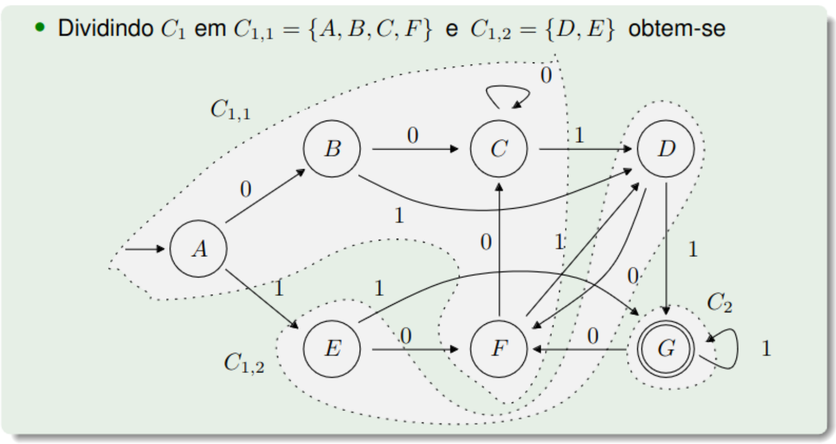
2.1.6 Algoritmo de redução de AFD



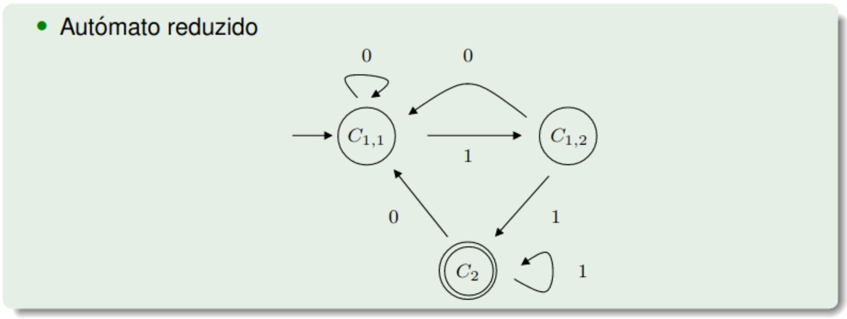
Primeiro, divide-se os estados em dois conjuntos, um contendo os estados de aceitação e outro os de não-aceitação.



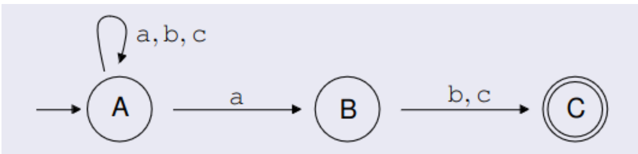
Em C_1, as transições em 0 são todas internas, mas em 1 podem ser internas ou provocar uma ida para C_2. Logo, não representa uma classe de equivalência e tem de ser dividido.



Pode verificar-se que $C_{1,1}$, $C_{1,2}$ e C_2 são classes de equivalência, pelo que se chegou à versão reduzida do autómato.



2.2 Autómato finito não determinista

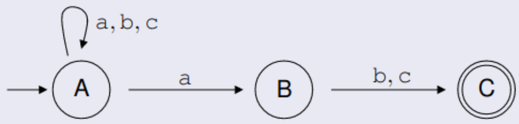


Um **autómato finito não determinista** é um autómato finito onde:

- as transições estão associadas a símbolos individuais do alfabeto **ou à palavra vazia** (ε);
- de cada estado saem **zero ou mais** transições por cada símbolo do **alfabeto ou** ε;
- há um estado inicial;
- há 0 ou mais estados de aceitação, que determinam as palavras aceites;
- os caminhos que começam no estado inicial e terminam num estado de aceitação representam as palavras aceites (reconhecidas) pelo autómato;
- As transições múltiplas ou com ε permitem alternativas de reconhecimento;
- As transições ausentes representam quedas num estado **morte** (estado não representado).

2.2.1 AFND: caminhos alternativos

- Analise o processo de reconhecimento da palavra `abab` ?

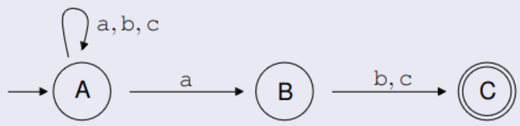


- Há 3 caminhos alternativos

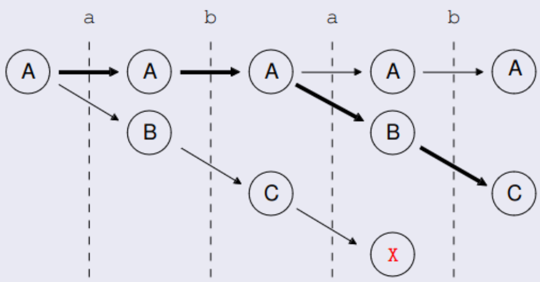
1. $A \xrightarrow{a} B \xrightarrow{b} C \xrightarrow{a}$ X
2. $A \xrightarrow{a} A \xrightarrow{b} A \xrightarrow{a} A \xrightarrow{b} A$
3. $A \xrightarrow{a} A \xrightarrow{b} A \xrightarrow{a} B \xrightarrow{b} C$

- Como há um caminho que conduz a um estado de aceitação a palavra é reconhecida pelo autómato

- Analise o processo de reconhecimento da palavra `abab` ?

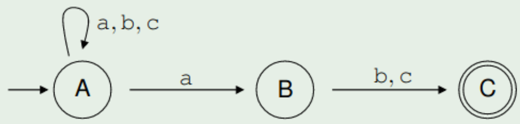


- Que se podem representar de forma arbórea



2.2.2 Exemplo

$\mathcal{Q}$ Que palavras são reconhecidas pelo autómato seguinte?

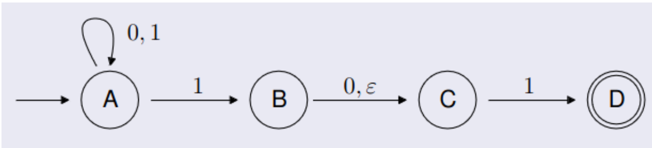


$\mathcal{R}$ Todas as palavras que terminarem em ab ou ac

$L = \{\omega \mathrm{a} x \, : \, \omega \in A^* \, \wedge \, x \in \{\mathrm{b}, \mathrm{c}\}\}.$

Percebe-se uma grande analogia entre este autómato e a expressão regular: $(a|b|c)^* a(b|c)$

2.2.3 AFND com transições - ε



A palavra **101** é reconhecida pelo autómato através do caminho:

$$A \xrightarrow{1} B \xrightarrow{0} C \xrightarrow{1} D$$

A palavra **11** é reconhecida pelo autómato através do caminho:

$$A \xrightarrow{1} B \xrightarrow{\varepsilon} C \xrightarrow{1} D$$

$$\text{porque } 11 = 1\varepsilon 1$$

Este autómato reconhece as palavras terminadas em **11** ou **101**.

$$L = \{w_1 w_2 : w_1 \in A^* \wedge w_2 \in \{11, 101\}\}$$

2.2.4 Definição

Um autómato finito não determinista (AFND) é um quíntuplo $M = (A, Q, q_0, \delta, F)$, em que:

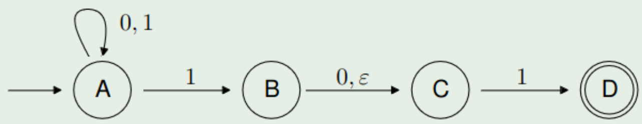
- A é o alfabeto de entrada;
- Q é um conjunto finito não vazio de estados;
- $q_0 \in Q$ é o estado inicial;
- $\delta \subseteq (Q \times A_\varepsilon \times Q)$ é a relação de transição entre estados, com $A_\varepsilon = A \cup \{\varepsilon\}$;
- $S \subseteq Q$ é o conjunto dos estados de aceitação.

Apenas a definição de δ difere em relação aos AFD.

Se se representa δ na forma de uma tabela, as células são preenchidas com elementos de $\wp(Q)$, ou seja, sub-conjuntos de Q.

2.2.5 Exemplo

$\mathcal{Q}$ Represente textualmente o AFND



$\mathcal{R}$ $M = (A, Q, q_0, \delta, F)$ com

- $A = \{0, 1\}$
- $Q = \{A, B, C, D\}$
- $q_0 = A$
- $F = \{D\}$

- $\delta = \{$
  $(A, 0, A), (A, 1, A),$
  $(A, 1, B), (B, 0, C),$
  $(B, \varepsilon, C), (C, 1, D)$
  $\}$

- $\delta =$

	0	1	ε
A	$\{A\}$	$\{A, B\}$	$\{\}$
B	$\{C\}$	$\{\}$	$\{C\}$
C	$\{\}$	$\{D\}$	$\{\}$
D	$\{\}$	$\{\}$	$\{\}$

O par $(A, 1, A)$, $(A, 1, B)$ faz com que δ não seja uma função.

2.2.6 AFND: linguagem reconhecida

Diz-se que um AFND $M = (A, Q, q_0, \delta, F)$, **aceita** uma palavra $u \in A^*$ se u se puder escrever na forma $u = u_1 u_2 \ldots u_n$, com $u_i \in A_\varepsilon$ e existir uma sequência de estados $s_0, s_1, \ldots, s_n$, satisfaça as seguintes condições:

- $s_0 = q_0$;
- qualquer que seja o $i = 1, \ldots, n$, $(s_{i-1}, u_i, s_i) \in \delta$;
- $s_n \in F$.

Caso contrário diz-se que M **rejeita** a entrada.

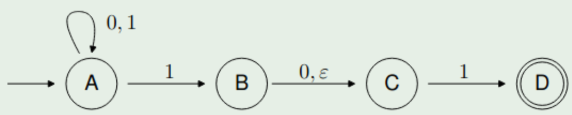
Note que n pode ser maior que $|u|$, porque alguns u_i podem ser ε.

Usar-se-á a notação $q_i \xrightarrow{u} q_j$ para indicar que a palavra u permite ir do estado q_i ao estado q_j.

Usando esta notação tem-se $L(M) = \{u : q_0 \xrightarrow{u} q_f \wedge q_f \in F\}$.

2.2.7 Exemplo

$\mathcal{Q}$ Sobre o alfabeto $A = \{0, 1\}$, considere o AFND M seguinte



e a linguagem $L = \{\omega \in A^* : \omega = (01)^n, n > 1\}$. Mostre que $L \subset L(M)$.

$\mathcal{R}$

2.2.8 Equivalência entre AFD e AFND

A classe das linguagens cobertas por um AFD é a mesma que a classe das linguagens cobertas por um AFND.

Isto significa que:

- Se M é um AFD, então $\exists_{M' \in AFND} : L(M') = L(M)$
- Se M é um AFND, então $\exists_{M' \in AFD} : L(M') = L(M)$

Como determinar um AFND equivalente a um AFD dado?

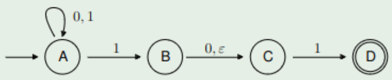
Pelas definições de AFD e AFND, um AFD é um AFND. Porquê?

- Q, q_0 e F têm a mesma definição.
- Nos AFD, $\delta : Q \times A \longrightarrow Q$.
- Nos AFND, $\delta \subset Q \times A_\varepsilon \longrightarrow Q$.
- Mas, se $\delta : Q \times A \longrightarrow Q$ então $\delta \subseteq Q \times A \times Q \subset Q \times A_\varepsilon \times Q$.
- Logo, um AFD é um AFND.

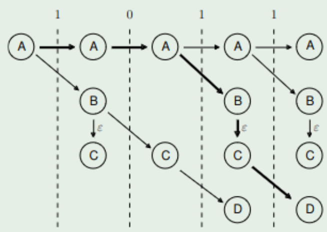
2.2.9 Equivalente AFD de um AFND

Como determinar um AFD equivalente a um AFND dado?

- No AFND



a árvore de reconhecimento da palavra `1011` sugere que a evolução se faz de sub-conjunto em sub-conjunto de estados



- Dado um AFND $M = (A, Q, q_0, \delta, F)$, considere o AFD $M' = (A, Q', q'_0, \delta', F')$ onde:
  - $Q' = \wp(Q)$
  - $q'_0 = \varepsilon\text{-closure}(q_0)$
  - $F' = \{f' \in \wp(Q) \; : \; f' \cap F \neq \emptyset\}$
  - $\delta' = \wp(Q) \times A \rightarrow \wp(Q)$,

    com $\delta'(q', a) = \bigcup_{q \in q'} \{s \; : \; s \in \varepsilon\text{-closure}(s') \wedge (q, a, s') \in \delta\}$
- M e M' reconhecem a mesma linguagem.

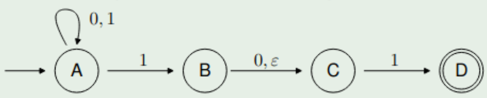
ε-closure(q) é o conjunto de estados constituído por q mais todos os direta ou indiretamente alcançáveis a partir de q apelas por transições-ε

Note que:

- O estado inicial (q'_0) pode conter 1 ou mais elementos de Q;
- Cada elemento do conjunto de chegada ($g' \in F'$) por conter elementos de F e $Q - F$.

Exemplo

$\mathcal{Q}$ Determinar um AFD equivalente ao AFND seguinte ?



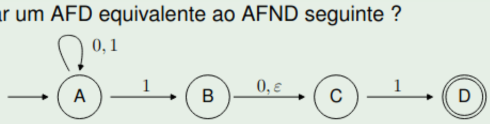
$\mathcal{R}$

- $Q' = \{X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, x_{10}, X_{11}, X_{12}, X_{13}, X_{14}, X_{15}\}$, com

$X_0 = \{\}$	$X_1 = \{A\}$	$X_2 = \{B\}$	$X_3 = \{A, B\}$
$X_4 = \{C\}$	$X_5 = \{A, C\}$	$X_6 = \{B, C\}$	$X_7 = \{A, B, C\}$
$X_8 = \{D\}$	$X_9 = \{A, D\}$	$X_{10} = \{B, D\}$	$X_{11} = \{A, B, D\}$
$X_{12} = \{C, D\}$	$X_{13} = \{A, C, D\}$	$X_{14} = \{B, C, D\}$	$X_{15} = \{A, B, C, D\}$

- $q'_0 = \varepsilon$-closure$(A) = \{A\} = X_1$
- $F' = \{X_8, X_9, X_{10}, X_{11}, X_{12}, X_{13}, X_{14}, X_{15}\}$

$\mathcal{Q}$ Determinar um AFD equivalente ao AFND seguinte ?

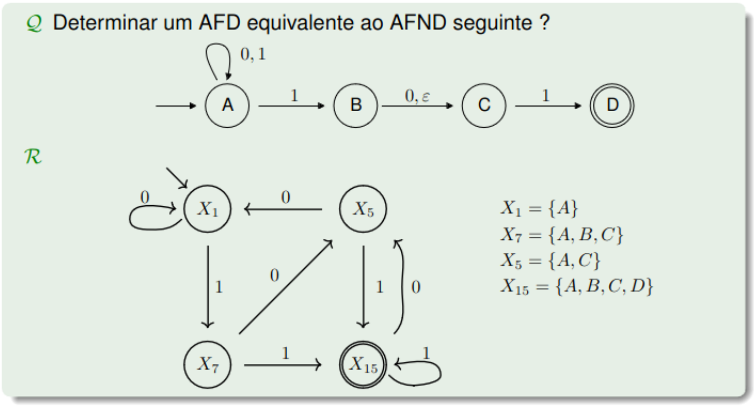


$\mathcal{R}$

- $\delta' =$

estado	0	1	estado	0	1
$X_0 = \{\}$	X_0	X_0	$X_1 = \{A\}$	X_1	X_7
$X_2 = \{B\}$	X_4	X_0	$X_3 = \{A, B\}$	X_5	X_7
$X_4 = \{C\}$	X_0	X_8	$X_5 = \{A, C\}$	X_1	X_{15}
$X_6 = \{B, C\}$	X_4	X_8	$X_7 = \{A, B, C\}$	X_5	X_{15}
$X_8 = \{D\}$	X_0	X_0	$X_9 = \{A, D\}$	X_1	X_7
$X_{10} = \{B, D\}$	X_4	X_0	$X_{11} = \{A, B, D\}$	X_5	X_7
$X_{12} = \{C, D\}$	X_0	X_8	$X_{13} = \{A, C, D\}$	X_1	X_{15}
$X_{14} = \{B, C, D\}$	X_4	X_8	$X_{15} = \{A, B, C, D\}$	X_5	X_{15}

Mais um exemplo (passos nos slides)



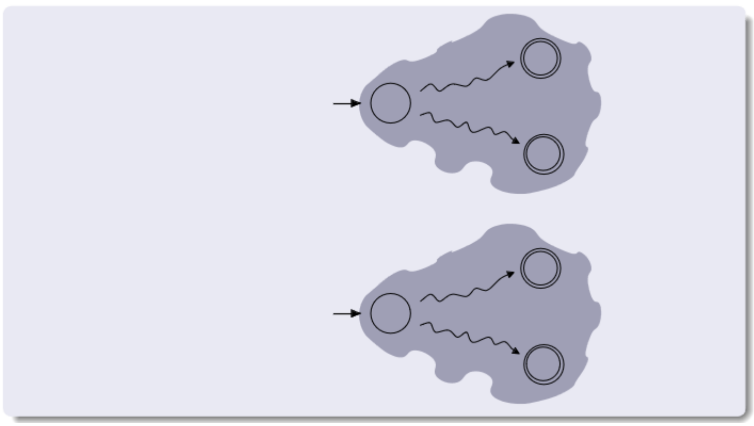
2.2.10 Operações sobre AFD e AFND

Os automátos finitos (AF) são fechados sobre as operações de:

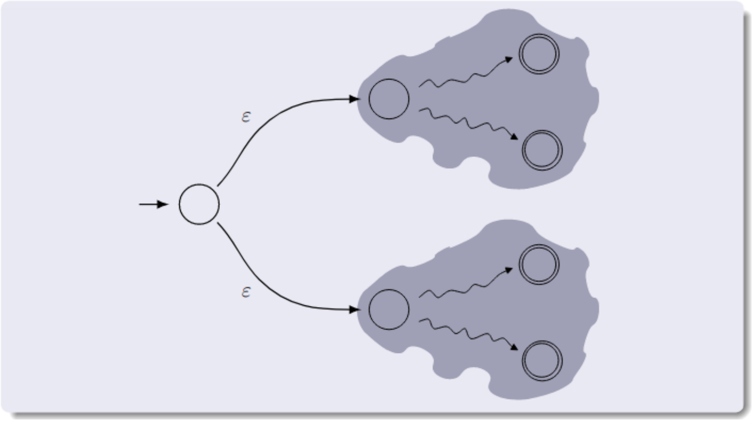
- Reunião
- Concatenação
- Fecho
- Interseção
- Complementação

Reunião de AF

Q - Como criar um AF que represente a reunião destes dois AF?



Acrescenta-se um novo estado que passa a ser o inicial, e acrescentam-se transições-ε deste novo estado para os estados iniciais originais.



Reunião de AF: definição

$\mathcal{D}$ Seja $M_1 = (A, Q_1, q_1, \delta_1, F_1)$ e $M_2 = (A, Q_2, q_2, \delta_2, F_2)$ dois autómatos (AFD ou AFND) quaisquer.
O AFND $M = (A, Q, q_0, \delta, F)$, onde

$$Q = Q_1 \cup Q_2 \cup \{q_0\}, \quad \text{com } q_0 \notin Q_1 \wedge q_0 \notin Q_2$$
$$F = F_1 \cup F_2$$
$$\delta = \delta_1 \cup \delta_2 \cup \{(q_0, \varepsilon, q_1), (q_0, \varepsilon, q_2)\}$$

implementa a reunião de M_1 e M_2, ou seja, $L(M) = L(M_1) \cup L(M_2)$.

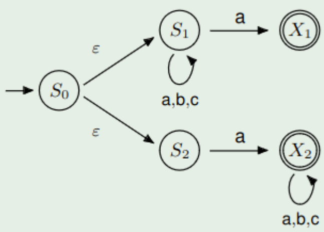
Reunião de AF: exemplo

$\mathcal{Q}$ Sobre o alfabeto $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, sejam L_1 e L_2 as duas linguagens seguintes:

$$L_1 = \{\omega\mathtt{a} \mid \omega \in A^*\} \qquad L_2 = \{\mathtt{a}\omega \mid \omega \in A^*\}$$

Determine um AF que reconheça $L = L_1 \cup L_2$.

$\mathcal{R}$



$\mathcal{Q}$ Sobre o alfabeto $A = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, sejam L_1 e L_2 as duas linguagens seguintes:

$$L_1 = \{\omega\mathrm{a} \mid \omega \in A^*\} \qquad L_2 = \{\mathrm{a}\omega \mid \omega \in A^*\}$$

Determine um AF que reconheça $L = L_1 \cup L_2$.

$\mathcal{R}$

$M_1 = (A, Q_1, q_1, \delta_1, F_1)$ com

$Q_1 = \{S_1, X_1\}, \quad q_1 = S_1, \quad F_1 = \{X_1\}$

$\delta_1 = \{(S_1, \mathrm{a}, S_1), (S_1, \mathrm{b}, S_1), (S_1, \mathrm{c}, S_1), (S_1, \mathrm{a}, X_1)$

$M_2 = (A, Q_2, q_2, \delta_2, F_2)$ com

$Q_2 = \{S_2, X_2\}, \quad q_2 = S_2, \quad F_2 = \{X_2\}$

$\delta_2 = \{(S_2, \mathrm{a}, X_2), (X_2, \mathrm{a}, X_2), (X_2, \mathrm{b}, X_2), (X_2, \mathrm{c}, X_2)$

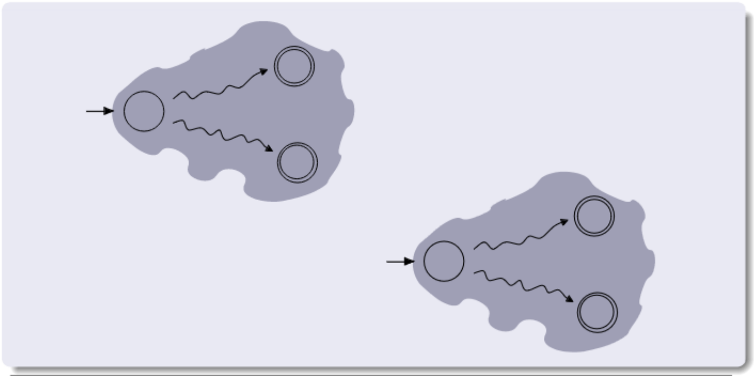
$M = M_1 \cup M_2 = (A, Q, q_0, \delta, F)$ com

$Q = \{S_0, S_1, X_1, S_2, X_2\}, \quad q_0 = S_0, \quad F = \{X_1, X_2\},$

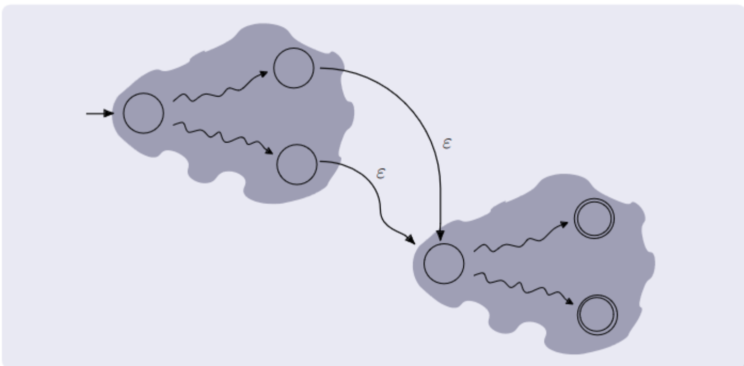
$\delta = \{(S_0, \varepsilon, S_1), (S_0, \varepsilon, S_2), (S_1, \mathrm{a}, S_1), (S_1, \mathrm{b}, S_1), (S_1, \mathrm{c}, S_1),$
$(S_1, \mathrm{a}, X_1), (S_2, \mathrm{a}, X_2), (X_2, \mathrm{a}, X_2), (X_2, \mathrm{b}, X_2), (X_2, \mathrm{c}, X_2)\}$

Concatenação de AF

Q - Como criar um AF que represente a concatenação destes dois AF?



O estado inicial passa a ser o estado inicial do AF da esquerda.
Os estados de aceitação são apenas os estados de aceitação do AF da direita.
Acrescentam-se transições-ε dos (antigos) estados de aceitação do AF da esquerda para o estado inicial do AF da direita.



Concatenação de AF: definição

$\mathcal{D}$ Seja $M_1 = (A, Q_1, q_1, \delta_1, F_1)$ e $M_2 = (A, Q_2, q_2, \delta_2, F_2)$ dois autómatos (AFD ou AFND) quaisquer.
O AFND $M = (A, Q, q_0, \delta, F)$, onde

$$Q = Q_1 \cup Q_2$$
$$q_0 = q_1$$
$$F = F_2$$
$$\delta = \delta_1 \cup \delta_2 \cup (F_1 \times \{\varepsilon\} \times \{q_2\})$$

implementa a concatenação de M_1 e M_2, ou seja,
$L(M) = L(M_1) \cdot L(M_2)$.

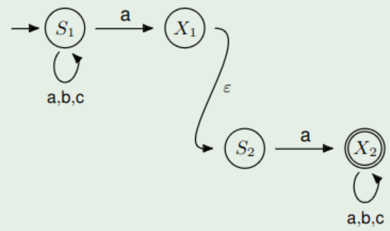
Concatenação de AF: exemplo

$\mathcal{Q}$ Sobre o alfabeto $A = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, sejam L_1 e L_2 as duas linguagens seguintes:

$$L_1 = \{\omega\mathrm{a} \mid \omega \in A^*\} \qquad L_2 = \{\mathrm{a}\omega \mid \omega \in A^*\}$$

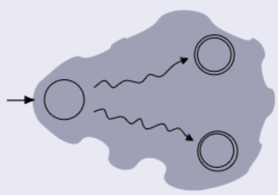
Determine um AF que reconheça $L = L_1 \cdot L_2$.

$\mathcal{R}$

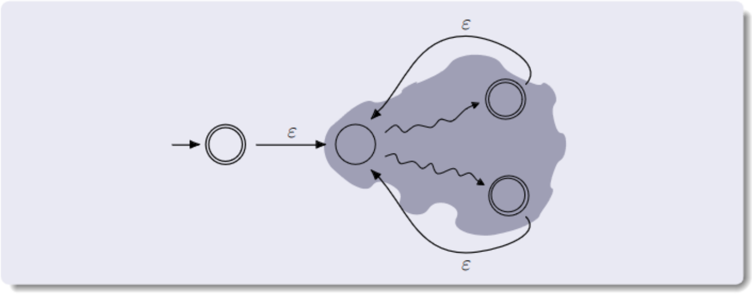


Fecho de AF

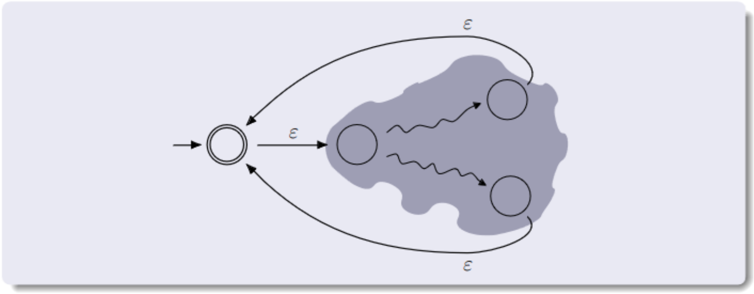
Q - Como criar um AF que represente o fecho deste AF?



Acrescenta-se um novo estado que passa a ser o inicial.
O novo estado inicial é de aceitação.
Acrescenta-se transições-ε dos estados de aceitação do AF para o estado inicial original.



Ou acrescenta-se transições-ε dos estados de aceitação do AF para o novo estado inicial (caso em que antigos estados de aceitação podem deixar de o ser).
Note que em geral não se pode fundir o novo estado inicial com o antigo.



Fecho de AF: definição

$\mathcal{D}$ Seja $M_1 = (A, Q_1, q_1, \delta_1, F_1)$ um autómato (AFD ou AFND) qualquer. O AFND $M = (A, Q, q_0, \delta, F)$, onde

$$Q = Q_1 \cup \{q_0\}$$
$$F = \{q_0\}$$
$$\delta = \delta_1 \cup (F_1 \times \{\varepsilon\} \times \{q_0\}) \cup \{(q_0, \varepsilon, q_1)\}$$

implementa o fecho de M_1, ou seja, $L(M) = L(M_1)^*$.

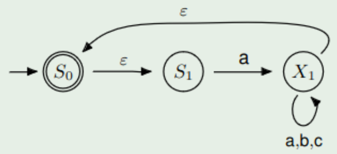
Fecho de AF: exemplo

$\mathcal{Q}$ Sobre o alfabeto $A = \{a, b, c\}$, seja

$$L_1 = \{a\omega \mid \omega \in A^*\}$$

Determine o AFND que reconhece a linguagem ${L_1}^*$.

$\mathcal{R}$



Interseção de AF

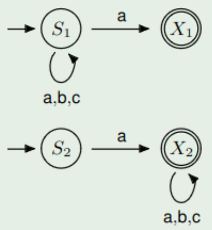
Como criar um AF que represente a interseção de L_1 e L_2?
Controi-se AF para as linguagens e definir os estados que resultam do produto cartesiano $\{S_1, X_1\} \times \{S_2, X_2\}$. No entanto, alguns podem ser inalcançáveis.

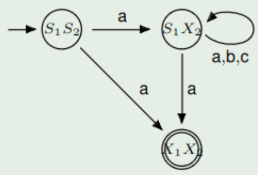
$\mathcal{Q}$ Sobre o alfabeto $A = \{a, b, c\}$, sejam L_1 e L_2 as duas linguagens seguintes:

$$L_1 = \{\omega a \mid \omega \in A^*\} \qquad L_2 = \{a\omega \mid \omega \in A^*\}$$

Determine um AFD ou AFND que reconheça $L = L_1 \cap L_2$.

$\mathcal{R}$





Interseção de AF: definição

$\mathcal{D}$ Seja $M_1 = (A, Q_1, q_1, \delta_1, F_1)$ e $M_2 = (A, Q_2, q_2, \delta_2, F_2)$ dois autómatos (AFD ou AFND) quaisquer.
O AFND $M = (A, Q, q_0, \delta, F)$, onde

$$Q = Q_1 \times Q_2$$
$$q_0 = (q_1, q_2)$$
$$F = F_1 \times F_2$$
$$\delta \subseteq (Q_1 \times Q_2) \times A_\varepsilon \times (Q_1 \times Q_2)$$

sendo δ definido de modo que
$((q_i, q_j), a, (q_i', q_j')) \in \delta$ se e só se $(q_i, a, q_i') \in \delta_1$ e $(q_j, a, q_j') \in \delta_2$,
implementa intersecção de M_1 e M_2, ie., $L(M) = L(M_1) \cap L(M_2)$.

Concatenação de AF

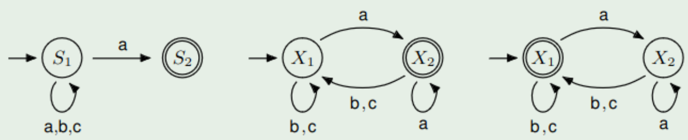
Como criar um AF que represente o complemento de L_1?
Considerando L_1 como um AFND, obtém-se um determinista equivalente e completa-se com os estados de aceitação.

$\mathcal{Q}$ Sobre o alfabeto $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, sejam L_1 e L_2 as duas linguagens seguintes:

$$L_1 = \{\omega\mathtt{a} \mid \omega \in A^*\}$$

Determine um AFD ou AFND que reconheça $L = \overline{L_1}$.

$\mathcal{R}$



2.2.11 Equivalência entre ER e AF

A classe das linguagens cobertas por expressões regulares (ER) é a mesma que a classe das linguagens cobertas por autómatos finitos (AF).

Logo:

- Se e é uma ER, então $\exists_{M \in AF} : L(M) = L(e)$;
- Se M é um AF, então $\exists_{e \in ER} : L(e) = L(M)$.

Isto indroduz duas operações:

- Como converter uma ER num AF equivalente;
- Como converter um AF numa AF equivalente.

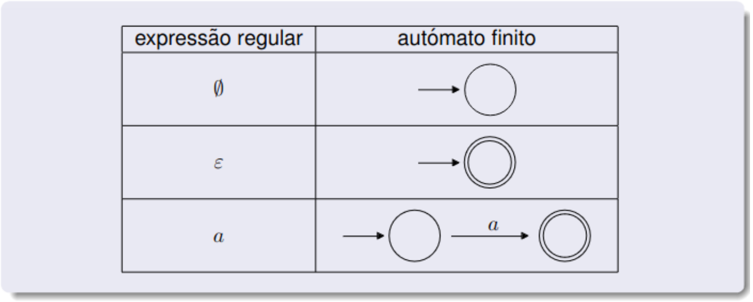
Abordagem

Já se viu anteriormente que uma ER qualquer é:

- ou um elemento primitivo;
- ou uma expressão do tipo $e_1|e_2$, sendo e_1 e e_2 duas expressões regulares quaisquer;
- ou uma expressão do tipo e_1e_2, sendo e_1 e e_2 duas expressões regulares quaisquer;
- ou uma expressão do tipo e^*, sendo e uma expressão regular qualquer.

Já se viu anteriormente como realizar a **reunião**, **concatenação** e **fecho** de autómatos finitos.

Então, se se identificar autómatos finitos equivalentes às expressões regulares primitivas, tem-se o problema da conversão de uma expressão regular para uma autómato finito resolvido.



Na realidade, o autómato referente a ε pode ser obtido aplicando o fecho ao autómato de $\emptyset$.

Algoritmo de conversão

- Se a expressão regular é do tipo primitivo, o autómato correspondente pode ser obtido na tabela anterior;
- Se é do tipo e^*, aplica-se este mesmo algoritmo na obtenção de um autómato equivalente à expressão regular e e, de seguida, aplica-se o fecho de autómatos;
- Se é do tipo $e_1 e_2$, aplica-se este mesmo algoritmo na obtenção de autómatos para as expressões e_1 e e_2 e, de seguida, aplica-se a concatenação de autómatos;
- Finalmente, se é do tipo $e_1|e_2$, aplica-se este mesmo algoritmo na obtenção de autómatos para as expressões e_1 e e_2 e, de seguida, aplica-se a reunião de autómatos.

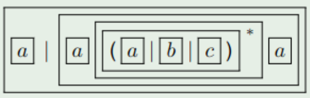
Na realidade, o algoritmo corresponde a um processo de decomposição arbórea a partir da raiz seguido de um processo de construção arbórea a partir das folhas.

Exemplo

$\mathcal{Q}$ Construa um autómato equivalente à expressão regular $e = a|a(a|b|c)^*a$

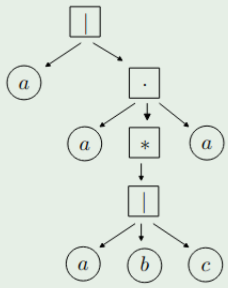
$\mathcal{R}$

❶ Decomposição

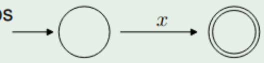


$\mathcal{R}$

❶ Decomposição

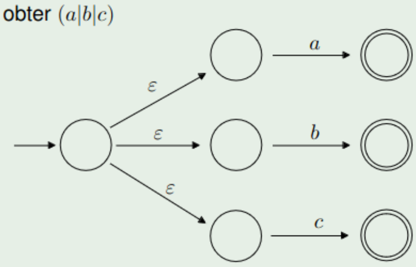


❷ Autómatos primitivos

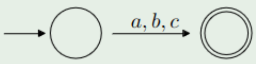


com $x = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$

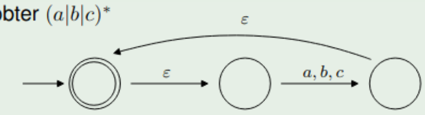
❸ Reunião para obter $(a|b|c)$



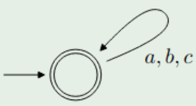
❹ Simplificando



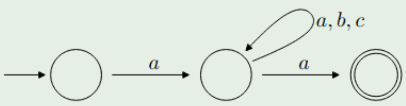
❺ Fecho para obter $(a|b|c)^*$

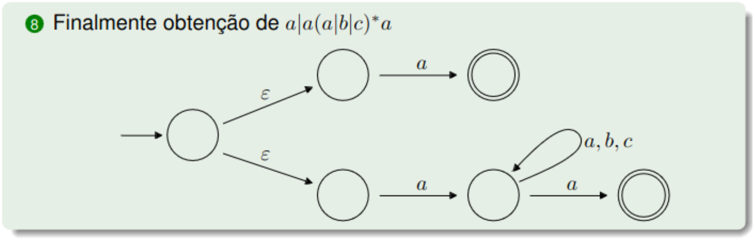


❻ Simplificando

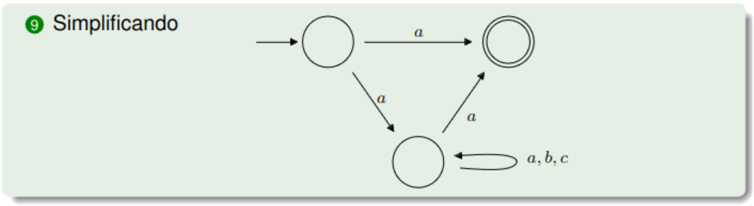


❼ Concatenação (já com simplificação) para obter $a(a|b|c)^*a$





8 Finalmente obtenção de $a|a(a|b|c)^*a$



9 Simplificando

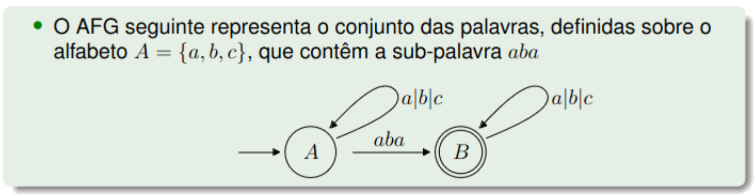
2.3 Autómato finito generalizado (AFG)

Um **autómato finito generalizado (AFG)** é um quíntuplo $M = (A, Q, q_0, \delta, F)$, em que:

- A é o alfabeto de entrada;
- Q é um conjunto finito não vazio de estados;
- $q_0 \in Q$ é o estado inicial;
- $\delta \subseteq (Q \times E \times Q)$ é a relação de transição entre estados, sendo E o conjunto das expressões regulares definidas sobre A;
- $F \subseteq Q$ é o conjunto dos estados de aceitação.

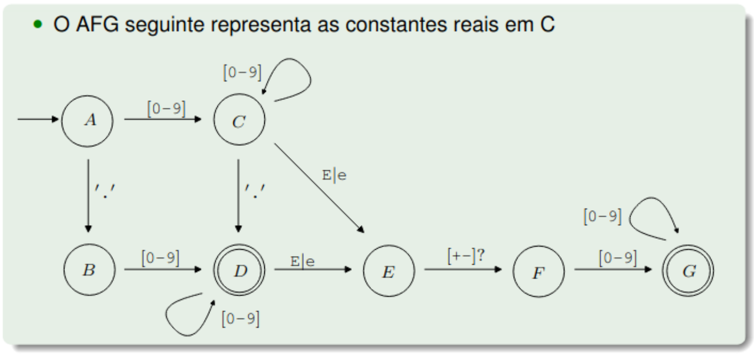
A diferença em relação ao AFD e AFND está na definição da relação δ. Neste caso as etiquetas são expressões regulares.

Com base nesta definição os AFD e os AFND são autómatos finitos generalizados.



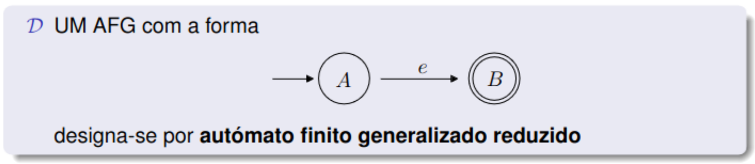
- O AFG seguinte representa o conjunto das palavras, definidas sobre o alfabeto $A = \{a, b, c\}$, que contêm a sub-palavra aba

Note que a etiqueta das transições $A \longrightarrow A$ e $B \longrightarrow B$ é $a|b|c$ (uma expressão regular) e não a, b, c (que representa 3 transições, uma em a, uma em b e uma em c).



Note que se usou '.' e não . , porque o último é uma expressão regular que representa qualquer letra do alfabeto.

2.3.1 Conversão de um AFG numa ER



Note que:

- O estado A não é de aceitação e não tem transições a chegar;
- O estado B é de aceitação e não tem transições a sair.

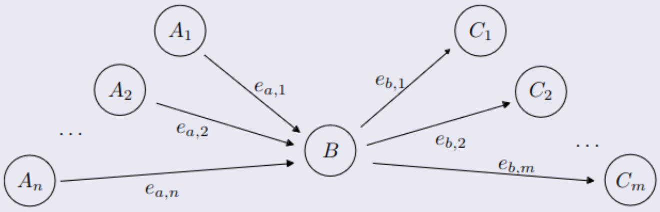
Se se reduzir um AFG à forma anterior, e é uma expressão regular equivalente ao autómato

O processo de conversão resume-se assim à conversão de AFG à forma reduzida.
Algoritmo de conversão

1. transformação de um AFG noutro cujo estado inicial **não tenha transições a chegar**
   - Se necessário, acrescenta-se um novo estado inicial com uma transição em ε para o antigo
2. transformação de um AFG noutro com **um único estado de aceitação, sem transições de saída**
   - Se necessário, acrescenta-se um novo estado, que passa a ser o único de aceitação, que recebe transições em ε dos anteriores estados de aceitação, que deixam de o ser
3. Eliminação dos estados intermédios
   - Os estados são eliminados um a um, em processos de transformação que mantêm a equivalência

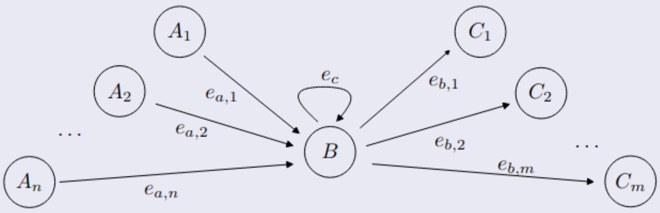
Algoritmo de eliminação de um estado

- Caso em que o estado a eliminar (B) **não tem** transições de si para si



- Pode acontecer que haja $A_i = C_j$
- Para ir de A_i para C_j através de B, para $i = 1, 2, \cdots, n$ e $j = 1, 2, \cdots, m$, é preciso uma palavra que encaixe na expressão regular $(e_{a,i})(e_{b,j})$
- Então, se se retirar B, é preciso acrescentar uma transição de A_i para C_j que contemple essas palavras, ou seja, com a etiqueta $(e_{a,i})(e_{b,j})$
- Esta transição fica em paralelo com uma que já exista

- Caso em que o estado a eliminar (B) **tem** transições de si para si



- Pode acontecer que haja $A_i = C_j$
- Para ir de A_i para C_j através de B, para $i = 1, 2, \cdots, n$ e $j = 1, 2, \cdots, m$, é preciso uma palavra que encaixe na expressão regular $(e_{a,i})(e_c)^*(e_{b,j})$
- Então, se se retirar B, é preciso acrescentar uma transição de A_i para C_j que contemple essas palavras, ou seja com etiqueta $(e_{a,i})(e_c)^*(e_{b,j})$
- Esta transição fica em paralelo com uma que já exista

2.4 Equivalência entre GR e AF

A classe das linguagens cobertas por gramáticas regulares (ER) é a mesma que a classe das linguagens cobertas por autómatos finitos (AF).

Logo:

- Se G é uma ER, então $\exists_{M \in AF} : L(M) = L(G)$
- Se M é uma ER, então $\exists_{G \in ER} : L(G) = L(M)$

Isto introduz duas operações:

- Como converter um AF numa GR equivalente
- Como converter uma GR num AF equivalente

2.4.1 Conversão de um AF numa GR

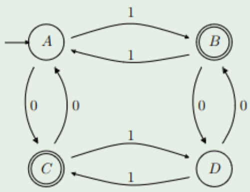
$\mathcal{A}$ Seja $M = (A, Q, q_0, \delta, F)$ um autómato finito qualquer.
A GR $G = (T, N, P, S)$, onde

$$T = A$$
$$N = Q$$
$$S = q_0$$
$$P = \{p \to a\,q \;:\; p, q \in Q \,\wedge\, a \in T \,\wedge\, (p, a, q) \in \delta\} \cup \{p \to \varepsilon \;:\; p \in F\}$$

representa a mesma linguagem que M, isto é, $L(G) = L(M)$.

Exemplo

$\mathcal{Q}$ Determine uma GR equivalente ao AF



$\mathcal{R}$

$$A \to 0\ C \mid 1\ B$$
$$B \to 0\ D \mid 1\ A \mid \varepsilon$$
$$C \to 0\ A \mid 1\ D \mid \varepsilon$$
$$D \to 0\ B \mid 1\ C$$

2.4.2 Conversão de uma GR num AFG

$\mathcal{A}$ Seja $G = (T, N, P, S)$ uma gramática regular qualquer.
O AF $M = (A, Q, q_0, \delta, F)$, onde

$$\begin{aligned}
&A = T \\
&Q = N \cup \{q_f\}, \quad \text{com } q_f \notin N \\
&q_0 = S \\
&F = \{q_f\} \\
&\delta = \{(q_i, e, q_j) \,:\, q_i, q_j \in N \;\wedge\; e \in T^* \;\wedge\; q_i \to e\, q_j \in P\} \\
&\quad\;\; \cup \{(q, e, q_f) \,:\, q \in N \;\wedge\; e \in T^* \;\wedge\; q \to e \in P\}
\end{aligned}$$

representa a mesma linguagem que G, isto é, $L(M) = L(G)$.

Exemplo

$\mathcal{Q}$ Determine um AFG equivalente à GR

$$\begin{aligned}
&S \to \mathtt{a}\ S \mid \mathtt{b}\ S \mid \mathtt{c}\ S \mid \mathtt{a\,b\,a}\ X \\
&X \to \mathtt{a}\ X \mid \mathtt{b}\ X \mid \mathtt{c}\ X \mid \varepsilon
\end{aligned}$$

$\mathcal{R}$

Sendo $M = (A, Q, q_0, \delta, F)$ o AFG equivalente, tem-se

$$\begin{aligned}
&A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\} \\
&Q = \{S, X, q_f\} \\
&q_0 = S \\
&\delta = \{(S, \mathtt{a}, S), (S, \mathtt{b}, S), (S, \mathtt{c}, S), (S, \mathtt{aba}, X), \\
&\quad (X, \mathtt{a}, X), (X, \mathtt{b}, X), (X, \mathtt{c}, X), (X, \varepsilon, q_f)\} \\
&F = \{q_f\}
\end{aligned}$$