



Compiladores

Linguagens Regulares, Expressões Regulares e Gramáticas Regulares

Artur Pereira <artur@ua.pt>,
Miguel Oliveira e Silva <mos@ua.pt>

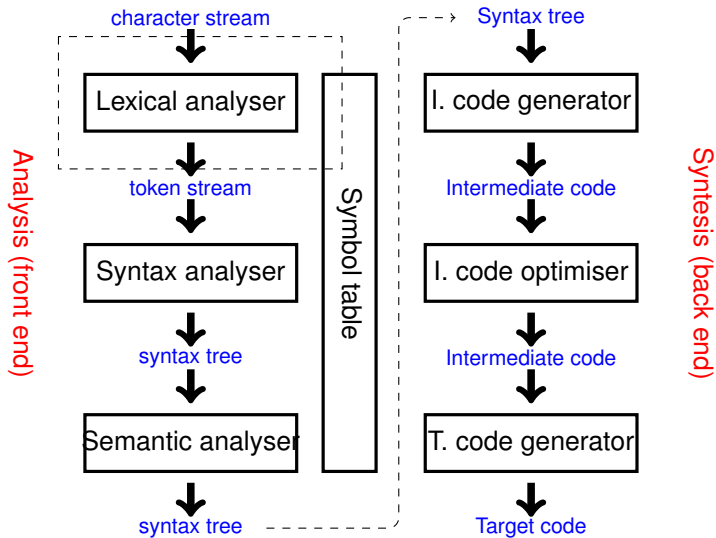
DETI, Universidade de Aveiro

Ano letivo de 2021-2022

Sumário

- 1 Análise lexical revisitada
- 2 Linguagens regulares
- 3 Expressões regulares
- 4 Gramáticas regulares
- 5 Equivalência entre expressões regulares e gramáticas regulares

Papel da análise lexical



Papel da análise lexical

- Converte a sequência de caracteres numa sequência de *tokens*
- Um *token* é um tuplo `<token-name, attribute-value>`
 - `token-name` é um símbolo (abstrato) representando 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

Linguagem regular

Definição

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

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

Note que:

- 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"

Linguagem regular

Definição

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

- 1 O conjunto vazio, \emptyset , é uma linguagem regular (LR).
- 2 Qualquer que seja o $a \in A$, o conjunto $\{a\}$ é uma LR.
- 3 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

Linguagem regular

Definição

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

- 1 O conjunto vazio, \emptyset , é uma linguagem regular (LR).
- 2 Qualquer que seja o $a \in A$, o conjunto $\{a\}$ é uma LR.
- 3 Se L_1 e L_2 são LR, então a sua reunião ($L_1 \cup L_2$) é uma LR.
- 4 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

Linguagem regular

Definição

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

- 1 O conjunto vazio, \emptyset , é uma linguagem regular (LR).
- 2 Qualquer que seja o $a \in A$, o conjunto $\{a\}$ é uma LR.
- 3 Se L_1 e L_2 são LR, então a sua reunião ($L_1 \cup L_2$) é uma LR.
- 4 Se L_1 e L_2 são LR, então a sua concatenação ($L_1 \cdot L_2$) é uma LR.
- 5 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

Linguagem regular

Definição

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

- 1 O conjunto vazio, \emptyset , é uma linguagem regular (LR).
- 2 Qualquer que seja o $a \in A$, o conjunto $\{a\}$ é uma LR.
- 3 Se L_1 e L_2 são LR, então a sua reunião ($L_1 \cup L_2$) é uma LR.
- 4 Se L_1 e L_2 são LR, então a sua concatenação ($L_1 \cdot L_2$) é uma LR.
- 5 Se L_1 é uma LR, então o seu fecho de Kleene $(L_1)^*$ é uma LR.
- 6 Nada mais é LR.

Note que

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

Definição de linguagem regular

exemplo #1

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 $A = \{0, 1\}$

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

Expressões regulares

Definição

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

- 1 \emptyset é uma expressão regular (ER) que representa a LR $\{\}$.
- 2 Qualquer que seja o $a \in A$, a é uma ER que representa a LR $\{a\}$.
- 3 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 $L_1 \cup L_2$.
- 4 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 $L_1.L_2$.
- 5 Se e_1 é uma ER representando a LR L_1 , então $(e_1)^*$ é uma ER representando a LR $(L_1)^*$.
- 6 Nada mais é expressão regular.

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

Expressões regulares

Precedência dos operadores regulares

- 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: a expressão regular

$e_1 | e_2 e_3^*$

recorre a esta precedência para representar a expressão regular

$(e_1) | (e_2 ((e_3)^*))$

Expressões regulares

Exemplos

Q Determine uma ER que represente o conjunto dos números binários começados em 1 e terminados em 0.

R $1(0|1)^*0$

Q Determine uma ER que represente as sequências definidas sobre o alfabeto $A = \{a, b, c\}$ que satisfazem o requisito de qualquer b ter um a imediatamente à sua esquerda e um c imediatamente à sua direita.

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 .

$(a|abc|c)^*$

Q Determine uma ER que represente as sequências binárias com um número par de zeros.

R $(1^*01^*01^*)^*|1^* = 1^*(01^*01^*)^*$

Expressões regulares

Propriedades da operação de escolha

- A operação de escolha goza das propriedades:
 - comutativa: $e_1 \mid e_2 = e_2 \mid e_1$
 - associativa: $e_1 \mid (e_2 \mid e_3) = (e_1 \mid e_2) \mid e_3 = e_1 \mid e_2 \mid e_3$
 - idempotência: $e_1 \mid e_1 = e_1$
 - existência de elemento neutro: $e_1 \mid () = () \mid e_1 = e_1$

-
- Exemplo:
 - comutativa: $a \mid ab = ab \mid a$
 - associativa: $a \mid (b \mid ca) = (a \mid b) \mid ca = a \mid b \mid ca$
 - idempotência: $ab \mid ab = ab$
 - não há interesse prático em fazer uma união com o conjunto vazio
 - assim, em algumas ferramentas (por exemplo o ANTLR), $()$ representa a palavra vazia, não o conjunto vazio

Expressões regulares

Propriedades da operação de concatenação

- 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: $e_1() = ()e_1 = ()$
 - **não goza da propriedade comutativa**

-
- Exemplo: seja $e_1 = a$, $e_2 = bc$, e $e_3 = c$
 - associativa: $a(bc\ c) = (a\ bc)c = a\ bc\ c$

Expressões regulares

Propriedades distributivas

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

- distributiva à esquerda da concatenação em relação à escolha:

$$e_1(e_2 \mid e_3) = e_1e_2 \mid e_1e_3$$

- distributiva à direita da concatenação em relação à escolha:

$$(e_1 \mid e_2)e_3 = e_1e_3 \mid e_2e_3$$

-
- Exemplo:

- distributiva à esquerda da concatenação em relação à escolha:

$$ab(a \mid cc) = aba \mid abcc$$

- distributiva à direita da concatenação em relação à escolha:

$$(ab \mid a)cc = abcc \mid acc$$

Expressões regulares

Propriedades da operação de fecho de Kleene

- A operação de fecho goza das propriedades:

- $(e^*)^* = e^*$
- $(e_1^* \mid e_2^*)^* = (e_1 \mid e_2)^*$
- $(e_1 \mid e_2^*)^* = (e_1 \mid e_2)^*$
- $(e_1^* \mid e_2)^* = (e_1 \mid e_2)^*$

- Mas atenção:

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

-
- Exemplo:

- $b(a^*)^* = ba^*$
- $(a^* \mid b^*)^* = (a \mid b)^*$
- $(a \mid b^*)^* = (a \mid b)^*$
- $(a^* \mid b)^* = (a \mid b)^*$
- $(a \mid b)^* \neq a^* \mid b^*$
- $(ab)^* \neq a^* b^*$

Expressões regulares

Exemplos

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

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

R $1^*01^*01^*$

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

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

R $(b|c|\dots|z)^*a(b|c|\dots|z)^*a(b|c|\dots|z)^*a(b|c|\dots|z)^*$

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

Expressões regulares

Extensões notacionais comuns

- uma ou mais ocorrências:

$$e^+ = e.e^*$$

- uma ou nenhuma ocorrência:

$$e? = (e|\varepsilon)$$

- um símbolo do sub-alfabeto dado:

$$[a_1 a_2 a_3 \cdots a_n] = (a_1 \mid a_2 \mid a_3 \mid \cdots \mid a_n)$$

- um símbolo do sub-alfabeto dado:

$$[a_1 - a_n] = (a_1 \mid \cdots \mid a_n)$$

- um símbolo do alfabeto fora do conjunto dado:

$$[\hat{a}_1 a_2 a_3 \cdots a_n], \quad [\hat{a}_1 - a_n]$$

Em ANTLR:

- $x..y$ é equivalente a $[x-y]$
- $\sim[abc]$ é equivalente a $[\hat{a}bc]$

Expressões regulares

Outras extensões notacionais

- n ocorrências de:

$$e\{n\} = \underbrace{e.e.\cdots.e}_n$$

- de n_1 a n_2 ocorrências:

$$e\{n_1, n_2\} = \underbrace{e.e.\cdots.e}_{n_1, n_2}$$

- n ou mais ocorrências:

$$e\{n, \} = \underbrace{e.e.\cdots.e}_{n,}$$

- $.$ representa um símbolo qualquer
- $^$ representa palavra vazia no início de linha
- $\$$ representa palavra vazia no fim de linha
- $\backslash <$ representa palavra vazia no início de palavra
- $\backslash >$ representa palavra vazia no fim de palavra

Em ANTLR:

- Pode ser feito através de predicados semânticos

Expressões regulares

Exemplos de extensões notacionais

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

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

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

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

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

$$\begin{aligned}\mathcal{R} \quad & (b|c|\dots|z)^*a(b|c|\dots|z)^*a(b|c|\dots|z)^*a(b|c|\dots|z)^* \\ &= ([b-z]^*a)([b-z]^*a)([b-z]^*a)[b-z]^* \\ &= ([b-z]^*a)\{3\}[b-z]^*\end{aligned}$$

Gramáticas regulares

Introdução

- Exemplo de gramática regular

$$\begin{array}{l} S \rightarrow a X \\ X \rightarrow a X \\ \quad | b X \\ \quad | \epsilon \end{array}$$

- Exemplo de gramática **não** regular

$$\begin{array}{l} S \rightarrow a S a \\ \quad | b S b \\ \quad | a \end{array}$$

-
- Letras minúsculas representam símbolos terminais e letras maiúsculas representam símbolos não terminais (o contrário do ANTLR)
 - Nas gramáticas regulares os símbolos não terminais apenas podem aparecer no fim

Gramáticas regulares

Definição

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 regras de rescrita), cada uma da forma $\alpha \rightarrow \beta$, 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

Gramáticas regulares

Operações sobre gramáticas regulares

- As gramáticas regulares são fechadas sob as operações de
 - reunião
 - concatenação
 - fecho
 - intersecção
 - complementação
- As operações de intersecção e complementação serão abordadas mais adiante através de autómatos finitos

Reunião de gramáticas regulares

Exemplo

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

$$L = L_1 \cup L_2$$

sabendo que

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

R

$S_1 \rightarrow a S_1$	$S_2 \rightarrow a X_2$
$b S_1$	$X_2 \rightarrow a X_2$
$c S_1$	$b X_2$
a	$c X_2$
	ϵ

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

Reunião de gramáticas regulares

Exemplo

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

$$L = L_1 \cup L_2$$

sabendo que

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

R

$$\begin{array}{l} S_1 \rightarrow a S_1 \\ \quad | \quad b S_1 \\ \quad | \quad c S_1 \\ \quad | \quad a \end{array}$$

$$\begin{array}{l} S_2 \rightarrow a X_2 \\ X_2 \rightarrow a X_2 \\ \quad | \quad b X_2 \\ \quad | \quad c X_2 \\ \quad | \quad \varepsilon \end{array}$$

$$\begin{array}{l} S \rightarrow S_1 \mid S_2 \\ S_1 \rightarrow a S_1 \mid b S_1 \mid c S_1 \\ \quad | \quad a \\ S_2 \rightarrow a X_2 \\ X_2 \rightarrow a X_2 \mid b X_2 \mid c X_2 \\ \quad | \quad \varepsilon \end{array}$$

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

Reunião de gramáticas regulares

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

$$T = T_1 \cup T_2$$

$$N = N_1 \cup N_2 \cup \{S\} \quad \text{com} \quad S \notin (N_1 \cup N_2)$$

$$P = \{S \rightarrow S_1, S \rightarrow S_2\} \cup P_1 \cup P_2$$

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 $L(G_i)$

Concatenação de gramáticas regulares

Exemplo

- 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

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

R

$S_1 \rightarrow a S_1$	$S_2 \rightarrow a X_2$
$b S_1$	$X_2 \rightarrow a X_2$
$c S_1$	$b X_2$
a	$c X_2$
	ϵ

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

Concatenação de gramáticas regulares

Exemplo

- 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

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

R

$S_1 \rightarrow a S_1$	$S_2 \rightarrow a X_2$	$S_1 \rightarrow a S_1 \mid b S_1 \mid c S_1$
$\mid b S_1$	$X_2 \rightarrow a X_2$	$\mid a S_2$
$\mid c S_1$	$\mid b X_2$	$S_2 \rightarrow a X_2$
$\mid a$	$\mid c X_2$	$X_2 \rightarrow a X_2 \mid b X_2 \mid c X_2$
	$\mid \epsilon$	

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

Concatenação de gramáticas regulares

Algoritmo

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

$$T = T_1 \cup T_2$$

$$N = N_1 \cup N_2$$

$$P = \{A \rightarrow \omega S_2 : (A \rightarrow \omega) \in P_1 \wedge \omega \in T_1^*\} \\ \cup \{A \rightarrow \omega : (A \rightarrow \omega) \in P_1 \wedge \omega \in T_1^* N_1\} \\ \cup P_2$$

$$S = S_1$$

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

Fecho de gramáticas regulares

Exemplo

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

$$L = L_1^*$$

sabendo que

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

R

$$\begin{array}{l} S_1 \rightarrow a S_1 \\ \quad | b S_1 \\ \quad | c S_1 \\ \quad | a \end{array}$$

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

Fecho de gramáticas regulares

Exemplo

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

$$L = L_1^*$$

sabendo que

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

R

$$\begin{array}{l} S_1 \rightarrow a S_1 \\ \quad | \quad b S_1 \\ \quad | \quad c S_1 \\ \quad | \quad a \end{array}$$

$$\begin{array}{l} S \rightarrow \varepsilon \quad | \quad S_1 \\ S_1 \rightarrow a S_1 \quad | \quad b S_1 \quad | \quad c S_1 \\ \quad \quad | \quad a S \end{array}$$

- 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

Fecho de gramáticas regulares

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

$$T = T_1$$

$$N = N_1 \cup \{S\} \quad \text{com} \quad S \notin N_1$$

$$P = \{S \rightarrow \varepsilon, S \rightarrow S_1\} \\ \cup \{A \rightarrow \omega S : (A \rightarrow \omega) \in P_1 \wedge \omega \in T_1^*\} \\ \cup \{A \rightarrow \omega : (A \rightarrow \omega) \in P_1 \wedge \omega \in T_1^* N_1\}$$

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

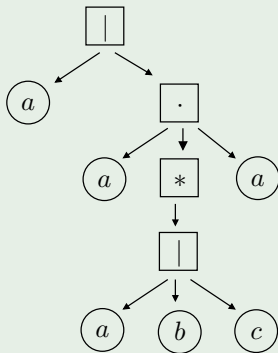
-
- As novas produções $S \rightarrow \varepsilon$ e $S \rightarrow S_1$ garantem que $(L(G_1))^n \subseteq L(G)$, para qualquer $n \geq 0$
 - As produções que só têm terminais ganham o novo símbolo inicial no fim
 - As produções que terminam num não terminal mantêm-se inalteradas

Conversão de uma ER em uma GR

exemplo

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

\mathcal{R}



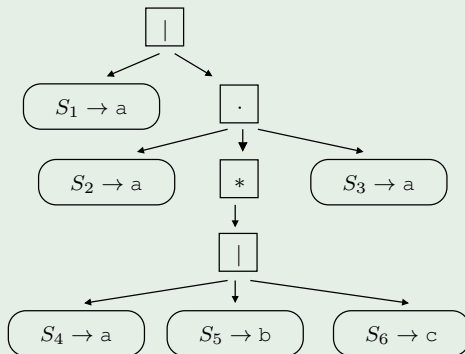
- Coloque-se de forma arbórea

Conversão de uma ER em uma GR

exemplo

Q Construa uma GR equivalente à ER $e = a|a(a|b|c)^*a$.

R



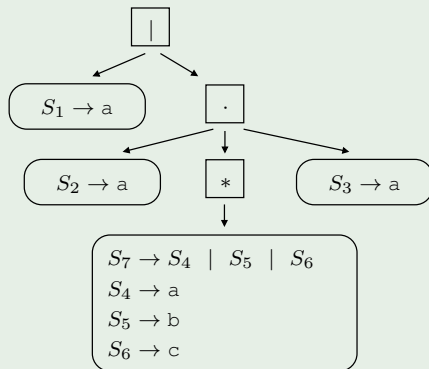
- Após converter as folhas (elementos primitivos) em GR

Conversão de uma ER em uma GR

exemplo

Q Construa uma GR equivalente à ER $e = a|a(a|b|c)^*a$.

R



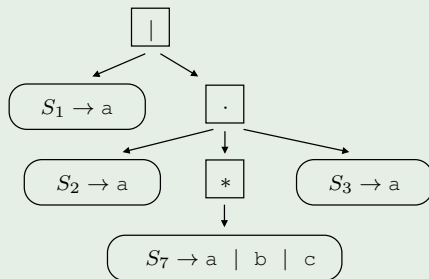
- Após aplicar a escolha (reunião) de baixo

Conversão de uma ER em uma GR

exemplo

Q Construa uma GR equivalente à ER $e = a|a(a|b|c)^*a$.

R



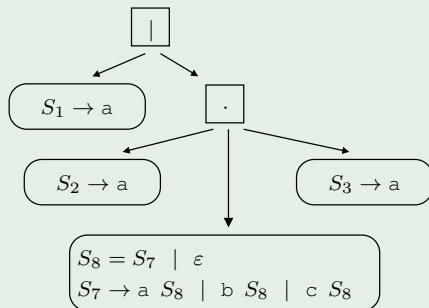
- Simplificando

Conversão de uma ER em uma GR

exemplo

Q Construa uma GR equivalente à ER $e = a|a(a|b|c)^*a$.

R



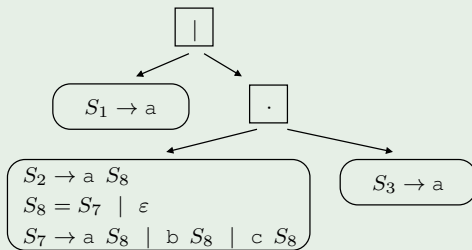
- Após aplicar o fecho

Conversão de uma ER em uma GR

exemplo

Q Construa uma GR equivalente à ER $e = a|a(a|b|c)^*a$.

R



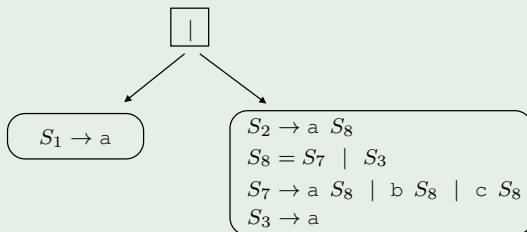
- Após aplicar a concatenação da esquerda

Conversão de uma ER em uma GR

exemplo

Q Construa uma GR equivalente à ER $e = a|a(a|b|c)^*a$.

R



- Após aplicar a concatenação da direita

Conversão de uma ER em uma GR

exemplo

Q Construa uma GR equivalente à ER $e = a|a(a|b|c)^*a$.

R

$$S \rightarrow S_1 \mid S_2$$

$$S_1 \rightarrow a$$

$$S_2 \rightarrow a S_8$$

$$S_8 \rightarrow S_7 \mid S_3$$

$$S_7 \rightarrow a S_8 \mid b S_8 \mid c S_8$$

$$S_3 \rightarrow a$$

e simplificando

$$S \rightarrow a \mid a S_8$$

$$S_8 \rightarrow a S_8 \mid b S_8 \mid c S_8 \mid a$$

- Finalmente após aplicar escolha (reunião) de cima

Conversão de uma ER em uma GR

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 $e_1.e_2$, sendo e_1 e e_2 duas expressões regulares quaisquer;
 - ou uma expressão do tipo $e_1|e_2$, 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	$S \rightarrow \epsilon$
a	$S \rightarrow a$

Conversão de uma ER em uma GR

Algoritmo de conversão

- 1 Se a 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 $e_1.e_2$, aplica-se este mesmo algoritmo na obtenção de GR para as expressões e_1 e e_2 e, de seguida, aplica-se a concatenação de GR.
- 4 Finalmente, se é do tipo $e_1|e_2$, aplica-se este mesmo algoritmo na obtenção de GR para as expressões e_1 e e_2 e, 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.

Conversão de uma GR em uma ER

Exemplo

Q Obtenha uma ER equivalente à gramática regular seguinte

$$S \rightarrow a S \mid c S \mid aba X$$

$$X \rightarrow a X \mid c X \mid \varepsilon$$

R Abordagem admitindo expressões regulares nas produções das gramáticas

$$E \rightarrow \varepsilon S$$

$$S \rightarrow a S \mid c S \mid (aba) X$$

$$X \rightarrow a X \mid c X \mid \varepsilon \varepsilon$$

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

$$E \rightarrow \varepsilon S$$

$$S \rightarrow (a|c) S \mid (aba) X$$

$$X \rightarrow (a|c) X \mid \varepsilon \varepsilon$$

- transformou-se $S \rightarrow a S$ e $S \rightarrow c S$ em $S \rightarrow (a|c) S$
- fez-se algo similar com o X

$$E \rightarrow \varepsilon (a|c)^* (aba) X$$

$$X \rightarrow (a|c) X \mid \varepsilon \varepsilon$$

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

$$E \rightarrow \varepsilon (a|c)^* (aba) (a|c)^* \varepsilon$$

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

Conversão de uma GR em uma ER

Exemplo

Q Obtenha uma ER equivalente à gramática regular seguinte

$$S \rightarrow a S \mid c S \mid aba X$$

$$X \rightarrow a X \mid c X \mid \varepsilon$$

R Abordagem transformando a gramática num conjunto e triplos

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

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

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

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

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

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

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

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

Conversão de uma GR em uma ER

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:

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

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.

Conversão de uma GR em uma ER

Algoritmo de remoção dos símbolos de N

- 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) , ω_2 representa o conjunto vazio e consequentemente $\omega_2^* = \varepsilon$



Compiladores

Autómatos finitos

Artur Pereira <artur@ua.pt>,
Miguel Oliveira e Silva <mos@ua.pt>

DETI, Universidade de Aveiro

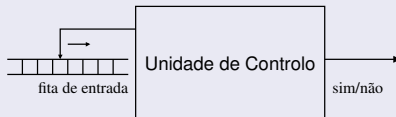
Ano letivo de 2021-2022

Sumário

- 1 Autômato finito determinista (AFD)
- 2 Redução de autômato finito determinista
- 3 Autômato finito não determinista (AFND)
- 4 Equivalência entre AFD e AFND
- 5 Operações sobre autômatos finitos (AF)
- 6 Equivalência entre ER e AF
- 7 Equivalência entre GR e AF

Autómato finito

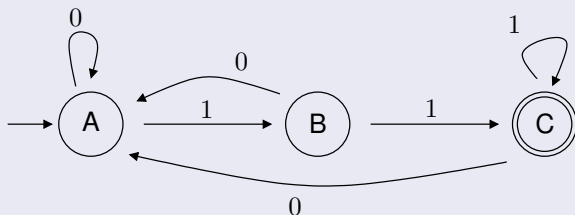
Um **autómato finito** é um mecanismo reconhecedor das palavras de uma linguagem regular



- A unidade de controlo é baseada na noção de estado e na de transição entre estados
 - número finito de estados
- A fita de entrada é só de leitura, com acesso sequencial
- A saída indica se a palavra é ou não aceite (reconhecida)
- Os autómatos finitos podem ser **deterministas**, **não deterministas** ou **generalizados**

Autômato finito determinista

Um **autômato finito determinista** é um autômato finito

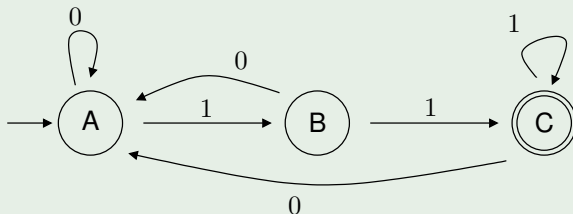


onde

- 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 autômato.

Autômato finito determinista: exemplo (1)

Q Que palavras binárias são reconhecidas pelo autômato seguinte?

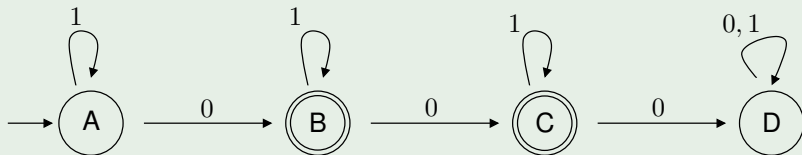


R Todas as palavras terminadas em 11.

ℰ Obtenha uma expressão regular que represente a mesma linguagem.

Autômato finito determinista: exemplo (2)

Q Que palavras binárias são reconhecidas pelo autômato seguinte?

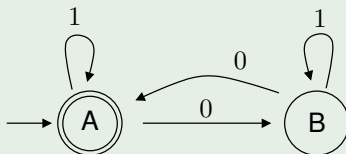


R Todas as palavras com apenas 1 ou 2 zeros.

ε Obtenha uma expressão regular que represente a mesma linguagem.

Autômato finito determinista: exemplo (3)

Q Que palavras binárias são reconhecidas pelo autômato seguinte?



R as sequências binárias com um número par de zeros.

Definição de autômato finito determinista

D Um **autômato finito determinista** (AFD) é um quintuplo $M = (A, Q, q_0, \delta, F)$, 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;
- $\delta : Q \times A \rightarrow Q$ é uma função que determina a transição entre estados; e
- $F \subseteq Q$ é o conjunto dos estados de aceitação.

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



Definição de autômato finito determinista

D Um **autômato finito determinista** (AFD) é um quintuplo $M = (A, Q, q_0, \delta, F)$, em que:

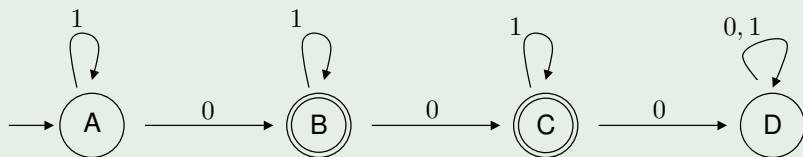
- A é o alfabeto de entrada;
- Q é um conjunto finito não vazio de estados;
- $q_0 \in Q$ é o estado inicial;
- $\delta : Q \times A \rightarrow Q$ é uma função que determina a transição entre estados; e
- $F \subseteq Q$ é o conjunto dos estados de aceitação.

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 $(q, a, q) \in Q \times A \times Q$

Autômato finito determinista: exemplo (4)

Q Represente textualmente o AFD seguinte.



R

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

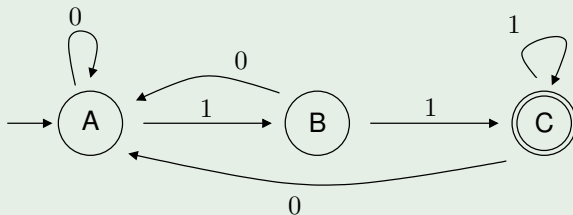
- $A = \{0, 1\}$
- $Q = \{A, B, C, D\}$
- $q_0 = A$
- $F = \{B, C\}$
- $\delta = \{$
 - $(A, 0, B), (A, 1, A),$
 - $(B, 0, C), (B, 1, B),$
 - $(C, 0, D), (C, 1, C),$
 - $(D, 0, D), (D, 1, D)\}$

• $\delta =$

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

Autômato finito determinista: exemplo (5)

Q Represente textualmente o AFD seguinte.



R

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

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

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

- $\delta =$

	0	1
A	A	B
B	A	C
C	A	C

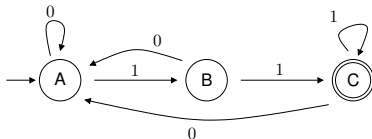
Linguagem reconhecida por um AFD (1)

- 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 $u = u_1 u_2 \cdots u_n$ e existir uma sequência de estados s_0, s_1, \cdots, s_n , que satisfaça as seguintes condições:

- 1 $s_0 = q_0$;
- 2 qualquer que seja o $i = 1, \cdots, n$, $s_i = \delta(s_{i-1}, u_i)$;
- 3 $s_n \in F$.

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

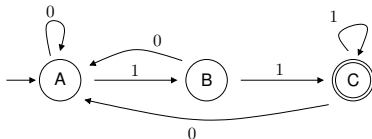
- A palavra $\omega_1 = 0101$ faz o caminho $A \xrightarrow{0} A \xrightarrow{1} B \xrightarrow{0} A \xrightarrow{1} B$
 - como B não é de aceitação, ω_1 não pertence à linguagem
- A palavra $\omega_2 = 0011$ faz o caminho $A \xrightarrow{0} A \xrightarrow{0} A \xrightarrow{1} B \xrightarrow{1} C$
 -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



Linguagem reconhecida por um AFD (2)

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

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



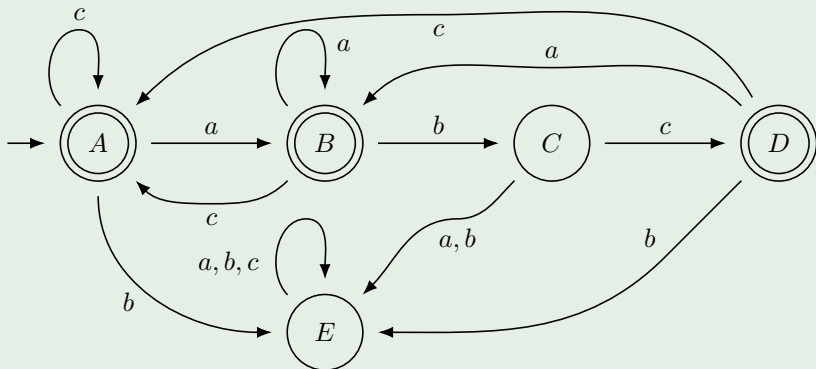
Autómato finito determinista: exemplo (6)

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

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

Projecte um autómato que reconheça L .

R



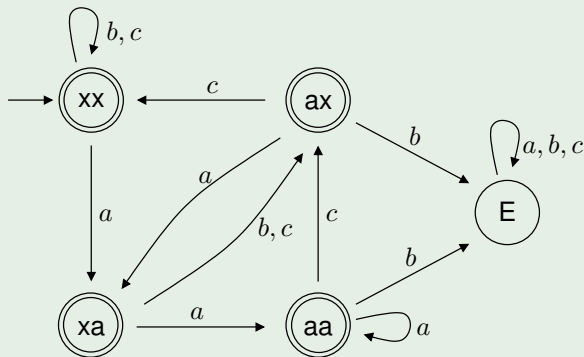
Autómato finito determinista: exemplo (7)

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

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

Projecte um autómato que reconheça L .

R



Autómato finito determinista: exemplo (8)

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

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

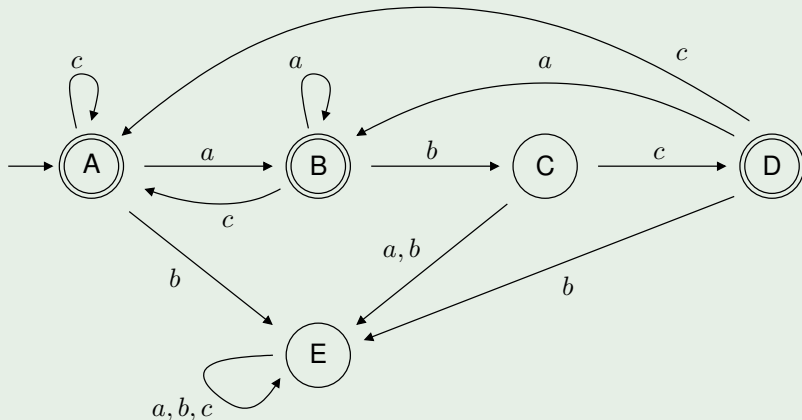
Projecte um autómato que reconheça L .

R

???

Redução de autômato finito determinista (1)

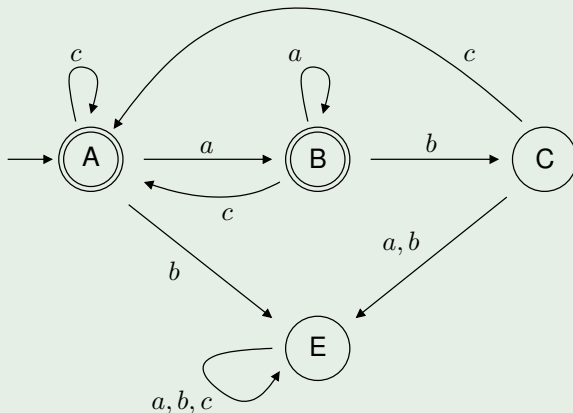
Q Considere o autômato seguinte (o do exemplo 6) e compare os estados A e D. Que pode concluir ?



- São equivalentes. Por conseguinte, podem ser fundidos

Redução de autômato finito determinista (2)

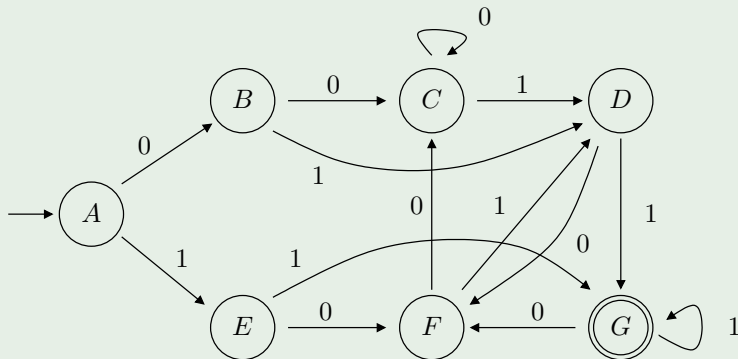
- O que resulta em



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

Algoritmo de Redução de AFD (1)

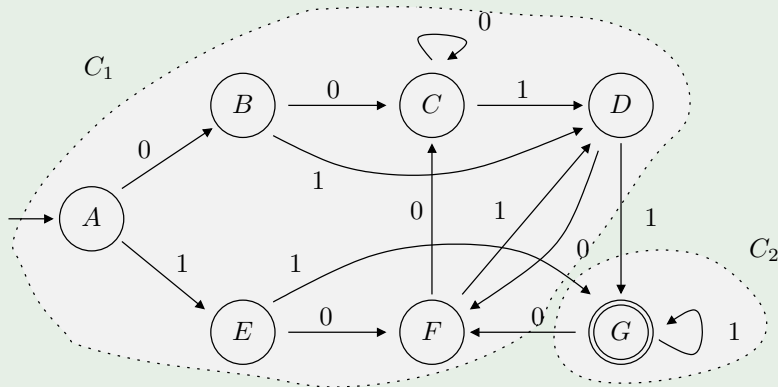
- Como proceder para reduzir um AFD?



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

Algoritmo de Redução de AFD (2)

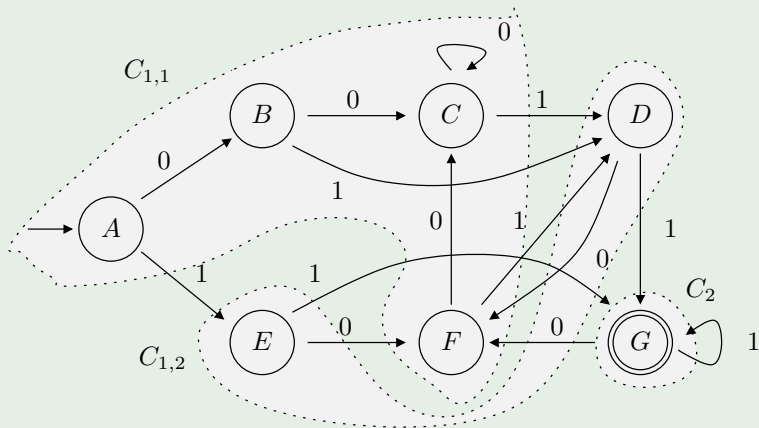
- Obtêm-se $C_1 = \{A, B, C, D, E, F\}$ e $C_2 = \{G\}$.



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

Algoritmo de Redução de AFD (3)

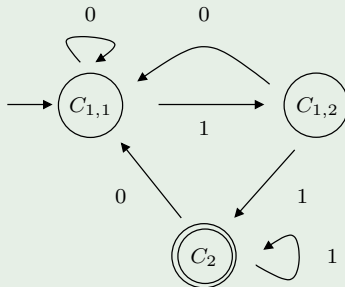
- Dividindo C_1 em $C_{1,1} = \{A, B, C, F\}$ e $C_{1,2} = \{D, E\}$ obtém-se



- 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 autômato.

Algoritmo de Redução de AFD (4)

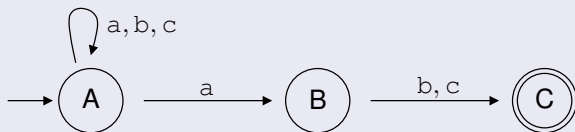
- Autômato reduzido



- Nos apontamentos encontra uma versão não gráfica do algoritmo.

Autômato finito não determinista

Um **autômato finito não determinista** é um autômato finito

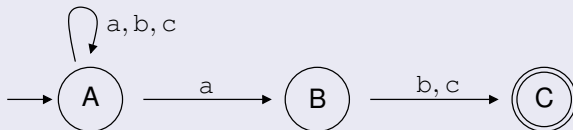


onde

- 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 autômato.
-
- As transições múltiplas ou com ϵ permitem alternativas de reconhecimento.
 - As transições ausentes representam quedas num estado de **morte** (estado não representado).

AFND: caminhos alternativos

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



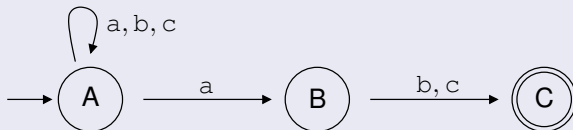
- Há 3 caminhos alternativos

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

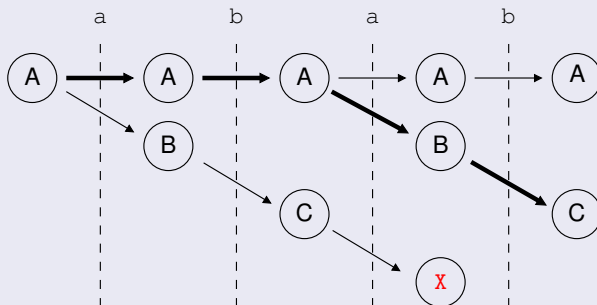
- Como há um caminho que conduz a um estado de aceitação a palavra é reconhecida pelo autômato

AFND: caminhos alternativos

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

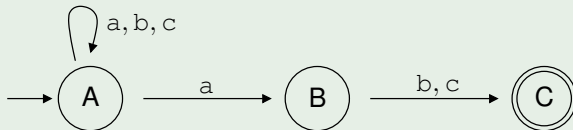


- Que se podem representar de forma arbórea



AFND: exemplo

Q Que palavras são reconhecidas pelo autômato seguinte?



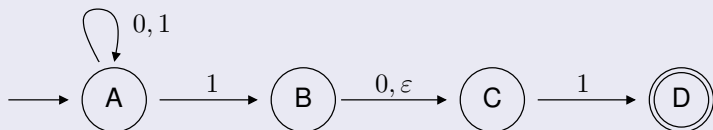
R Todas as palavras que terminarem em ab ou ac

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

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

AFND com transições- ϵ

- Considere o AFND seguinte que contém uma transição- ϵ .



- A palavra 101 é reconhecida pelo autômato através do caminho

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

- A palavra 11 é reconhecida pelo autômato através do caminho

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

porque $11 = 1\epsilon 1$

- Este autômato reconhece todas as palavras terminadas em 11 ou 101

$$L = \{\omega_1\omega_2 : \omega_1 \in A^* \wedge \omega_2 \in \{11, 101\}\}.$$

AFND: definição

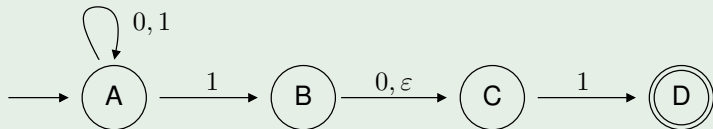
\mathcal{D} Um **autômato finito não determinista** (AFND) é um quintuplo $M = (A, Q, q_0, \delta, F)$, em que:

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

-
- Apenas a definição de δ difere em relação aos AFD.
 - Se se representar δ na forma de uma tabela, as células são preenchidas com elementos de $\wp(Q)$, ou seja, sub-conjuntos de Q .

AFND: Exemplo (2)

Q Represente textualmente o AFND



R

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

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

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 $u = u_1 u_2 \cdots u_n$, com $u_i \in A_\varepsilon$, e existir uma sequência de estados s_0, s_1, \cdots, s_n , que satisfaça as seguintes condições:
 - 1 $s_0 = q_0$;
 - 2 qualquer que seja o $i = 1, \cdots, n$, $(s_{i-1}, u_i, s_i) \in \delta$;
 - 3 $s_n \in F$.
- Caso contrário diz-se que M **rejeita** a entrada.
- Note que n pode ser maior que $|u|$, porque alguns dos 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 $L(M) = \{u : q_0 \xrightarrow{u} q_f \wedge q_f \in F\}$.

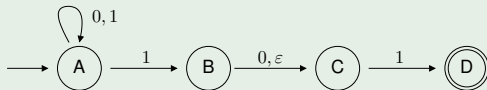
Equivalência entre AFD e AFND

- 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 $\exists_{M' \in \text{AFND}} : L(M') = L(M)$.
 - Se M é um AFND, então $\exists_{M' \in \text{AFD}} : L(M') = L(M)$.
- 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 $\delta : Q \times A \rightarrow Q$.
 - Nos AFND $\delta \subset Q \times A_\epsilon \times Q$
 - Mas, se $\delta : Q \times A \rightarrow Q$ então $\delta \subseteq Q \times A \times Q \subset Q \times A_\epsilon \times Q$
 - Logo, um AFD é um AFND

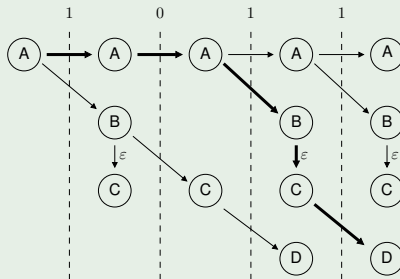
Equivalente AFD de um AFND (1)

- 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



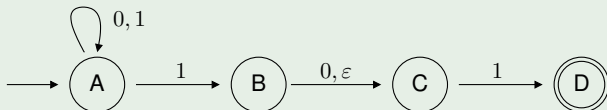
Equivalente AFD de um AFND (2)

- 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.

-
- $\varepsilon\text{-closure}(q)$ é o conjunto de estados constituído por q mais todos os direta ou indiretamente alcançáveis a partir de q apenas por transições- ε
 - Note que:
 - O estado inicial (q'_0) pode conter 1 ou mais elementos de Q
 - Cada elemento do conjunto de chegada ($f' \in F'$) por conter elementos de F e $Q - F$

Equivalente AFD de um AFND: exemplo

Q Determinar um AFD equivalente ao AFND seguinte ?



R

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

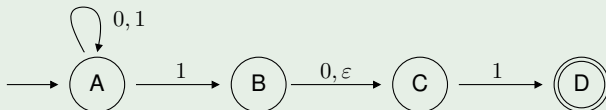
com

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

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

Equivalente AFD de um AFND: exemplo

Q Determinar um AFD equivalente ao AFND seguinte ?



R

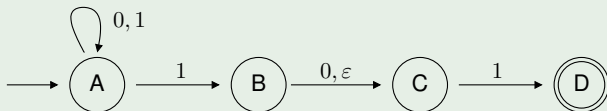
• $\delta' =$

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}

• Serão todos estes estados necessários?

Equivalente AFD de um AFND: exemplo

Q Determinar um AFD equivalente ao AFND seguinte ?

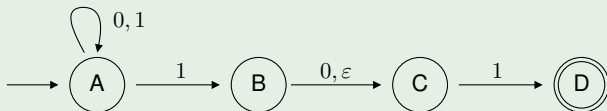


\mathcal{R}

- Consegue-se o mesmo resultado através de um processo construtivo.

Equivalente AFD de um AFND: exemplo

Q Determinar um AFD equivalente ao AFND seguinte ?



\mathcal{R}

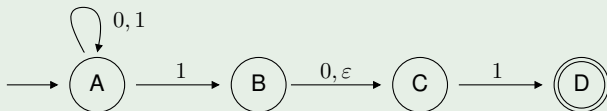


$$X_1 = \{A\}$$

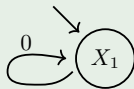
- Comece-se com o estado inicial ($X_1 = \{A\}$)

Equivalente AFD de um AFND: exemplo

Q Determinar um AFD equivalente ao AFND seguinte ?



R

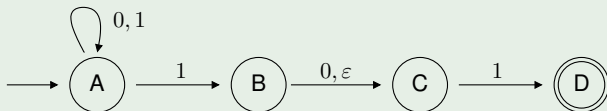


$$X_1 = \{A\}$$

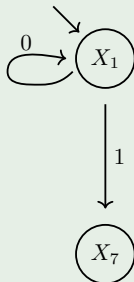
- $\delta'(X_1, 0) = \varepsilon\text{-closure}(A) = \{A\}$

Equivalente AFD de um AFND: exemplo

Q Determinar um AFD equivalente ao AFND seguinte ?



R



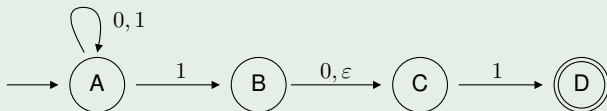
$$X_1 = \{A\}$$

$$X_7 = \{A, B, C\}$$

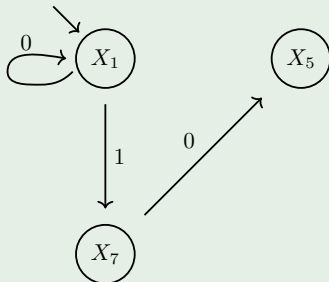
- $\delta'(X_1, 1) = \varepsilon\text{-closure}(A) \cup \varepsilon\text{-closure}(B) = \{A\} \cup \{B, C\} = \{A, B, C\}$

Equivalente AFD de um AFND: exemplo

Q Determinar um AFD equivalente ao AFND seguinte ?



\mathcal{R}



$$X_1 = \{A\}$$

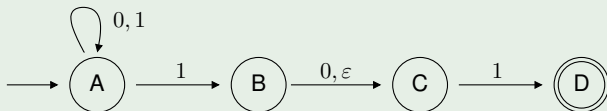
$$X_7 = \{A, B, C\}$$

$$X_5 = \{A, C\}$$

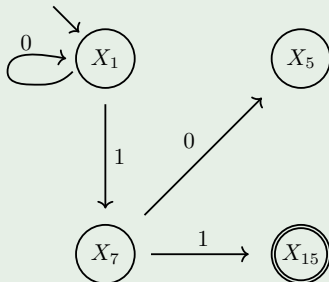
- $\delta'(X_7, 0) = \varepsilon\text{-closure}(A) \cup \varepsilon\text{-closure}(C) = \{A\} \cup \{C\} = \{A, C\}$

Equivalente AFD de um AFND: exemplo

Q Determinar um AFD equivalente ao AFND seguinte ?



\mathcal{R}



$$X_1 = \{A\}$$

$$X_7 = \{A, B, C\}$$

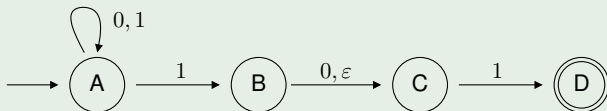
$$X_5 = \{A, C\}$$

$$X_{15} = \{A, B, C, D\}$$

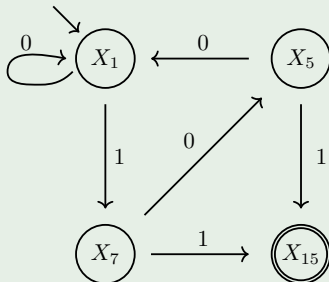
- $\delta'(X_7, 1) = \varepsilon\text{-closure}(A) \cup \varepsilon\text{-closure}(B) \cup \varepsilon\text{-closure}(D) = \{A\} \cup \{B, C\} \cup \{D\} = \{A, B, C, D\}$
- É de aceitação porque $\{A, B, C, D\} \cap \{D\} \neq \emptyset$

Equivalente AFD de um AFND: exemplo

Q Determinar um AFD equivalente ao AFND seguinte ?



R



$$X_1 = \{A\}$$

$$X_7 = \{A, B, C\}$$

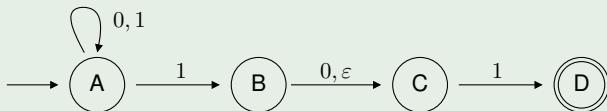
$$X_5 = \{A, C\}$$

$$X_{15} = \{A, B, C, D\}$$

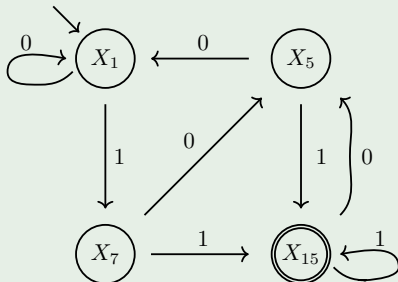
- $\delta'(X_5, 0) = \varepsilon\text{-closure}(A) = \{A\}$
- $\delta'(X_5, 1) = \varepsilon\text{-closure}(A) \cup \varepsilon\text{-closure}(B) \cup \varepsilon\text{-closure}(D) = \{A\} \cup \{B, C\} \cup \{D\} = \{A, B, C, D\}$

Equivalente AFD de um AFND: exemplo

Q Determinar um AFD equivalente ao AFND seguinte ?



R



$$X_1 = \{A\}$$

$$X_7 = \{A, B, C\}$$

$$X_5 = \{A, C\}$$

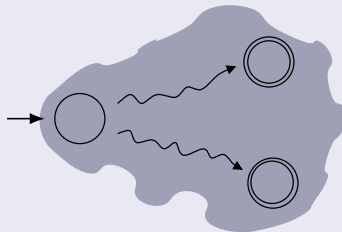
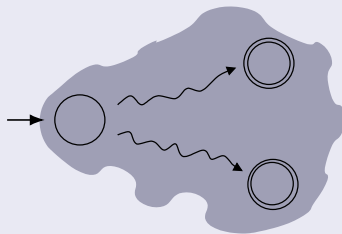
$$X_{15} = \{A, B, C, D\}$$

- $\delta'(X_{15}, 0) = \varepsilon\text{-closure}(A) \cup \varepsilon\text{-closure}(C) = \{A\} \cup \{C\} = \{A, C\}$
- $\delta'(X_{15}, 1) = \varepsilon\text{-closure}(A) \cup \varepsilon\text{-closure}(B) \cup \varepsilon\text{-closure}(D) = \{A\} \cup \{B, C\} \cup \{D\} = \{A, B, C, D\}$

Operações sobre AFD e AFND

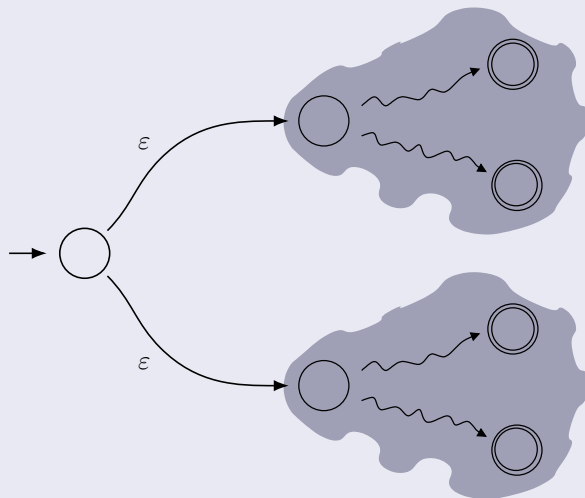
- Os automátos finitos (AF) são fechados sobre as operações de:
 - Reunião
 - Concatenação
 - Fecho
 - Intersecção
 - Complementação

Reunião de AF



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

Reunião de 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 (1)

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

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

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

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

R

- Como criar um AF que represente a reunião de L_1 e L_2 ?

Reunião de AF: exemplo (1)

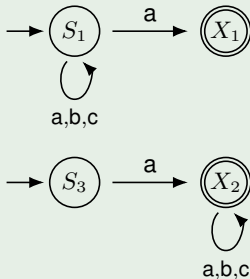
Q Sobre o alfabeto $A = \{a, b, c\}$, sejam L_1 e L_2 as duas linguagens seguintes:

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

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

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

R



- Constroi-se um AF para a linguagem L_1
- Constroi-se um AF para a linguagem L_2

Reunião de AF: exemplo (1)

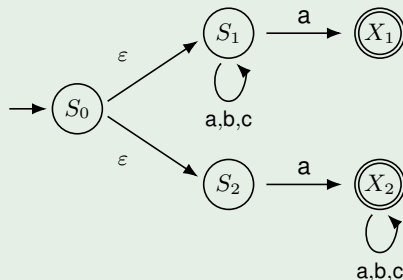
- Q Sobre o alfabeto $A = \{a, b, c\}$, sejam L_1 e L_2 as duas linguagens seguintes:

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

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

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

R



- Acrescenta-se um novo estado (S_0), que passa a ser o inicial
- E acrescentam-se transições- ϵ de S_0 (novo estado inicial) para S_1 e S_2 (os estados iniciais originais)

Reunião de AF: exemplo (1)

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

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

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

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

R

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

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

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

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

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

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

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

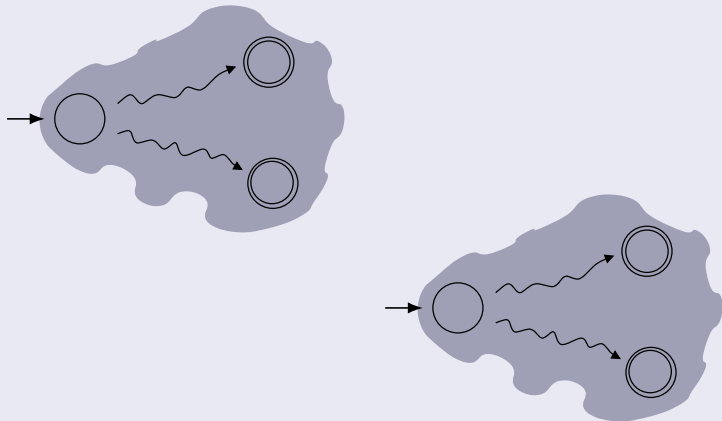
$$Q = \{S_0, S_1, X_1, S_2, X_2\}, \quad q_0 = S_0, \quad F = \{X_1, X_2\},$$

$$\delta = \{(S_0, \varepsilon, S_1), (S_0, \varepsilon, S_2), (S_1, a, S_1), (S_1, b, S_1), (S_1, c, S_1),$$

$$(S_1, a, X_1), (S_2, a, X_2), (X_2, a, X_2), (X_2, b, X_2), (X_2, c, X_2)\}$$

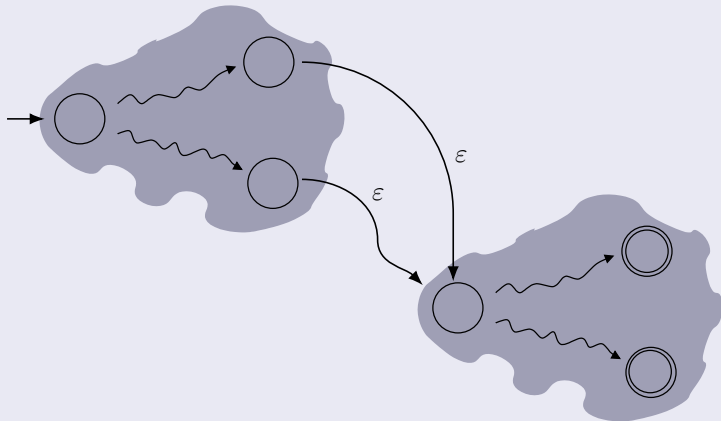
- Alternativamente, pode ser escrito de forma textual

Concatenação de AF



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

Concatenação de 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

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

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

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

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

R

-
- Como criar um AF que represente a concatenação de L_1 com L_2 ?

Concatenação de AF: exemplo

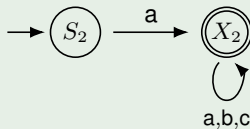
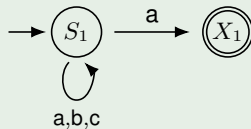
Q Sobre o alfabeto $A = \{a, b, c\}$, sejam L_1 e L_2 as duas linguagens seguintes:

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

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

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

R



- Constroi-se AF para as linguagens L_1 e L_2

Concatenação de AF: exemplo

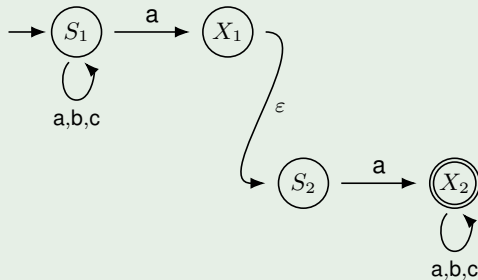
Q Sobre o alfabeto $A = \{a, b, c\}$, sejam L_1 e L_2 as duas linguagens seguintes:

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

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

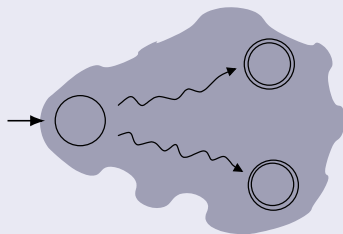
Determine um AF que reconheça $L = L_1 \cdot L_2$.

R



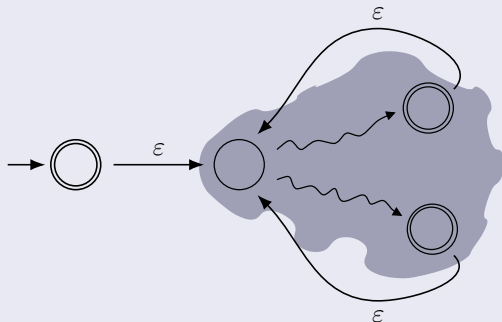
- X_1 deixa de ser de aceitação; S_2 deixa de ser de entrada
- acrescenta-se uma transição- ϵ de X_1 para S_2

Fecho de AF



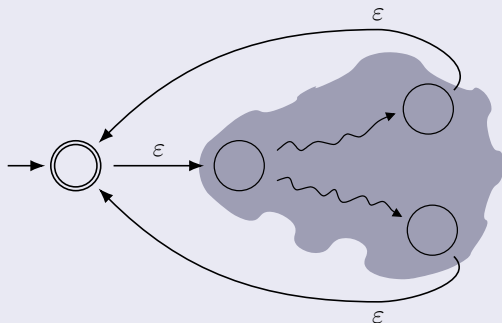
- Como criar um AF que represente o fecho deste AF?

Fecho de AF



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

Fecho de AF



- acrescenta-se um novo estado que passa a ser o inicial
- o novo estado inicial é de aceitação
- ou acrescentam-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, $L(M) = L(M_1)^*$.

- Em alternativa poder-se-á considerar que $F = F_1 \cup \{q_0\}$ e que de F_1 as novas transições- ε se dirigem a q_1

Fecho de AF: exemplo

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^* .

R

-
- Como criar um AF que represente o fecho de L_1 ?

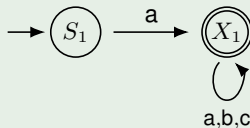
Fecho de AF: exemplo

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^* .

R



- Constroi-se um AF para L_1

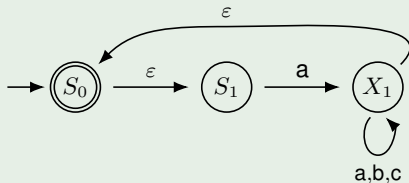
Fecho de AF: exemplo

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^* .

R



- acrescenta-se um novo estado (S_0), que passa a ser o inicial e é de aceitação
- liga-se este estado ao S_1 (inicial anterior) por uma transição- ϵ
- liga-se o estado X_1 (aceitação anterior) ao S_0 (novo inicial)
- X_1 deixa (pode deixar) de ser de aceitação

Intersecção de AF: exemplo

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

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

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

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

R

-
- Como criar um AF que represente a intersecção de L_1 e L_2 ?

Intersecção de AF: exemplo

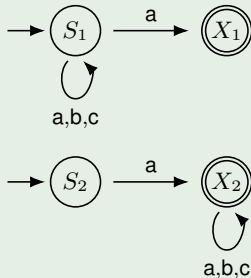
Q Sobre o alfabeto $A = \{a, b, c\}$, sejam L_1 e L_2 as duas linguagens seguintes:

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

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

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

R



- Constroi-se AF para as linguagens L_1 e L_2

Intersecção de AF: exemplo

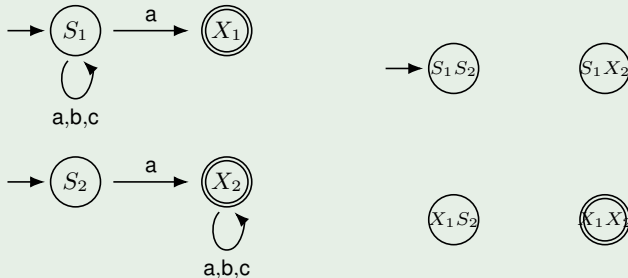
Q Sobre o alfabeto $A = \{a, b, c\}$, sejam L_1 e L_2 as duas linguagens seguintes:

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

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

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

R



- Definem-se os estados que resultam do produto cartesiano $\{S_1, X_1\} \times \{S_2, X_2\}$
- Mas, alguns podem não ser alcançáveis

Intersecção de AF: exemplo

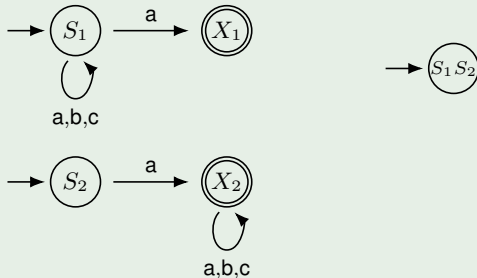
Q Sobre o alfabeto $A = \{a, b, c\}$, sejam L_1 e L_2 as duas linguagens seguintes:

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

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

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

R



- Pelo que começemos apenas pelo $S_1 S_2$, que corresponde ao estado inicial

Intersecção de AF: exemplo

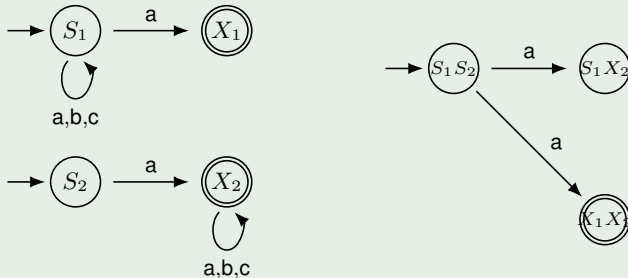
Q Sobre o alfabeto $A = \{a, b, c\}$, sejam L_1 e L_2 as duas linguagens seguintes:

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

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

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

R



- de $S_1 \xrightarrow{a} S_1$ e $S_2 \xrightarrow{a} X_2$ aparece $S_1S_2 \xrightarrow{a} S_1X_2$
- de $S_1 \xrightarrow{a} X_1$ e $S_2 \xrightarrow{a} X_2$ aparece $S_1S_2 \xrightarrow{a} X_1X_2$

Intersecção de AF: exemplo

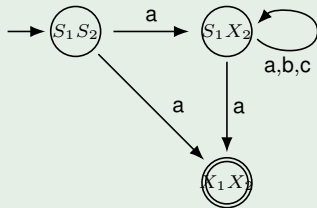
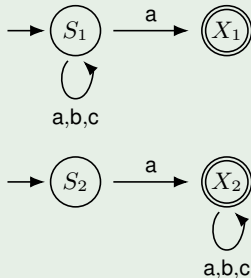
Q Sobre o alfabeto $A = \{a, b, c\}$, sejam L_1 e L_2 as duas linguagens seguintes:

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

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

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

R



- de $S_1 \xrightarrow{x} S_1$ e $X_2 \xrightarrow{x} X_2$ aparece $S_1X_2 \xrightarrow{x} S_1X_2$, para $x \in \{a, b, c\}$
- de $S_1 \xrightarrow{a} X_1$ e $X_2 \xrightarrow{a} X_2$ aparece $S_1X_2 \xrightarrow{a} X_1X_2$

Intersecção de AF: definição

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

$$Q = Q_1 \times Q_2$$

$$q_0 = (q_1, q_2)$$

$$F = F_1 \times F_2$$

$$\delta \subseteq (Q_1 \times Q_2) \times A_\epsilon \times (Q_1 \times Q_2)$$

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., $L(M) = L(M_1) \cap L(M_2)$.

Complementação de AF

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

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

Determine um AF que reconheça a linguagem $\overline{L_1}$.

R

- Para se obter o complementar de um autômato finito determinista (em sentido estrito, ie. com todos os estados representados) basta complementar o conjunto de aceitação
- Para o caso de um autômato finito não determinista **é preciso** calcular o determinista equivalente e complementá-lo.

Complementação de AF: exemplo

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

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

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

R

-
- Como criar um AF que represente a intersecção de L_1 e L_2 ?

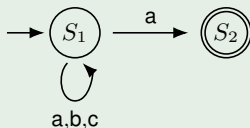
Complementação de AF: exemplo

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

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

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

R



- Considere-se um AFND para a linguagem L_1

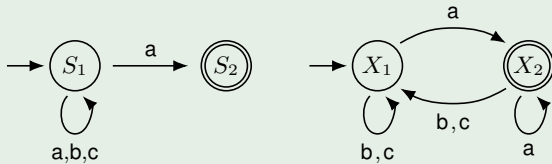
Complementação de AF: exemplo

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

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

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

R



- Obtenha-se um determinista equivalente

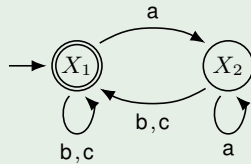
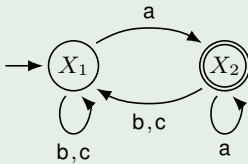
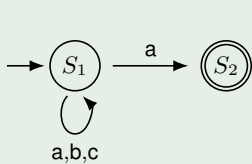
Complementação de AF: exemplo

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

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

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

R



- Complemente-se os estados de aceitação

Operações sobre AF: exercício

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

$$L_1 = \{v\omega \mid v \in \{a, b\} \wedge \omega \in A^*\} \quad (\text{palavras começadas por } a \text{ ou } b)$$

$$L_2 = \{\omega \in A^* \mid \#(a, \omega) \bmod 2 = 0\} \quad (\text{palavras com um número par de } a)$$

Determine AF que reconheça a linguagem

- L_1
- L_2
- $L_3 = L_1 \cup L_2$
- $L_4 = L_1 \cdot L_2$
- $L_6 = L_1 \cap L_2$
- $L_7 = \overline{L_2}$
- $L_8 = (L_4 \cup L_3)^*$

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 introduz duas operações:
 - Como converter uma ER num AF equivalente
 - Como converter um AF numa ER equivalente

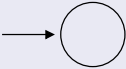
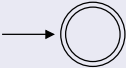
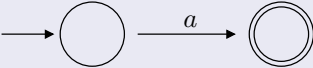
Conversão de uma ER num AF

Abordagem

- Já se viu anteriormente que uma expressão regular 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**, a **concatenação** e o **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 um autômato finito resolvido

Conversão de uma ER num AF

Autômatos dos elementos primitivos

expressão regular	autômato finito
\emptyset	
ε	
a	

- Na realidade, o autômato referente a ε pode ser obtido aplicando o fecho ao autômato de \emptyset

Conversão de uma ER num AF

Algoritmo de conversão

- Se a expressão regular é do tipo primitivo, o autômato correspondente pode ser obtido da tabela anterior
- Se é do tipo e^* , aplica-se este mesmo algoritmo na obtenção de um autômato equivalente à expressão regular e e, de seguida, aplica-se o fecho de autômatos
- Se é do tipo e_1e_2 , aplica-se este mesmo algoritmo na obtenção de autômatos para as expressões e_1 e e_2 e, de seguida, aplica-se a concatenação de autômatos
- Finalmente, se é do tipo $e_1|e_2$, aplica-se este mesmo algoritmo na obtenção de autômatos para as expressões e_1 e e_2 e, de seguida, aplica-se a reunião de autômatos

-
- 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

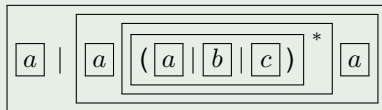
Conversão de uma ER num AF

Exemplo

Q Construa um autômato equivalente à expressão regular $e = a|a(a|b|c)^*a$

R

1 Decomposição



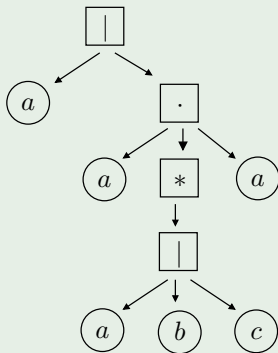
Conversão de uma ER num AF

Exemplo

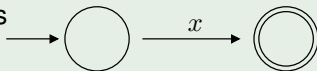
Q Construa um autômato equivalente à expressão regular $e = a|a(a|b|c)^*a$

R

1 Decomposição



2 Autômatos primitivos

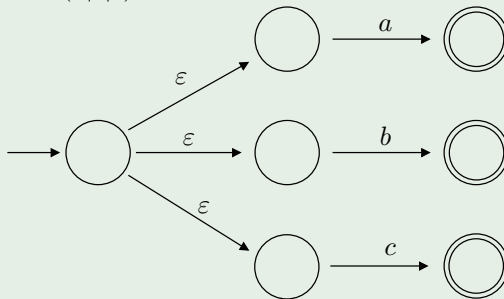


com $x = \{a, b, c\}$

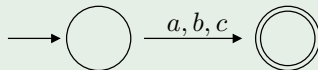
Conversão de uma ER num AF

Exemplo

3 Reunião para obter $(a|b|c)$



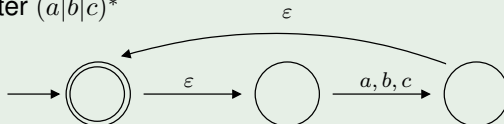
4 Simplificando



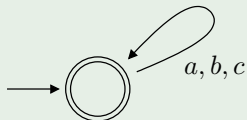
Conversão de uma ER num AF

Exemplo

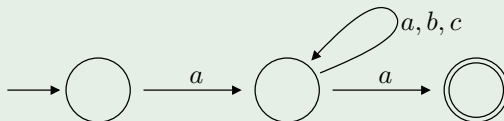
5 Fecho para obter $(a|b|c)^*$



6 Simplificando



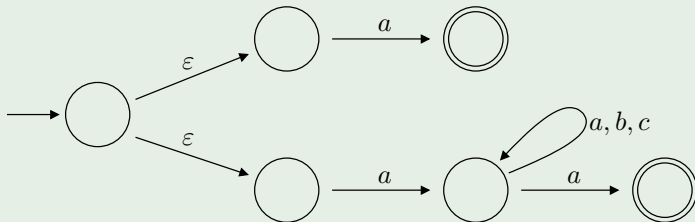
7 Concatenação (já com simplificação) para obter $a(a|b|c)^*a$



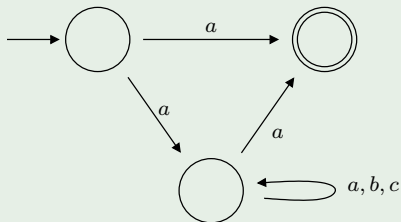
Conversão de uma ER num AF

Exemplo

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



9 Simplificando



Autômato finito generalizado (AFG)

Definição

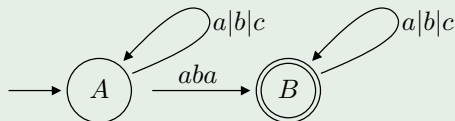
- \mathcal{D} Um **autômato finito generalizado** (AFG) é um quintuplo $M = (A, Q, q_0, \delta, F)$, em que:
- A é o alfabeto de entrada
 - Q é um conjunto finito não vazio de estados
 - $q_0 \in Q$ é o estado inicial
 - $\delta \subseteq (Q \times E \times Q)$ é a relação de transição entre estados, 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 autômatos finitos generalizados

Autômato finito generalizado (AFG)

Exemplo

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

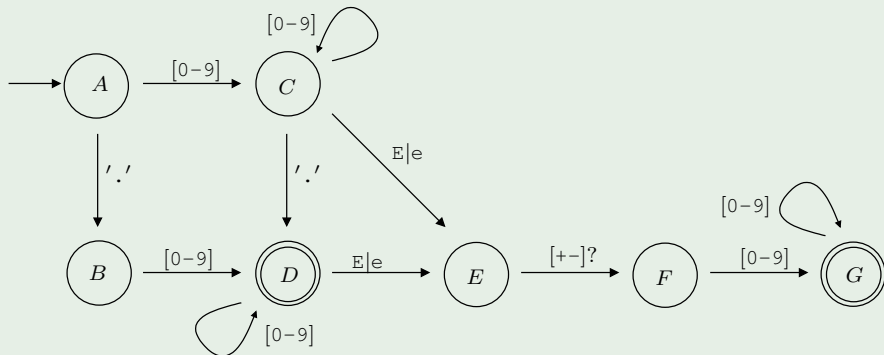


- Note que a etiqueta das transições $A \rightarrow A$ e $B \rightarrow B$ é $a|b|c$ (uma expressão regular) e não a, b, c (que representa 3 transições, uma em a , uma em b e uma em c)

Autômato finito generalizado (AFG)

Exemplo

- O AFG seguinte representa as constantes reais em C

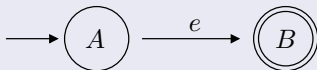


- Note que se usou $'.'$ e não $.$, porque o último é uma expressão regular que representa qualquer letra do alfabeto

Conversão de um AFG numa ER

Abordagem

\mathcal{D} UM AFG com a forma



designa-se por **autômato finito generalizado reduzido**

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

Conversão de um AFG numa ER

Algoritmo de conversão

- ① transformação de um AFG noutra 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 noutra 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

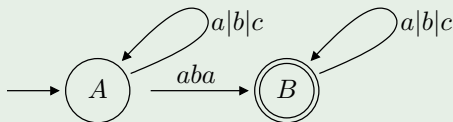
Conversão de um AFG numa ER

Ilustração com um exemplo

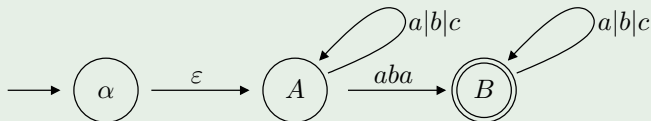
- 1 transformação de um AFG noutra cujo estado inicial **não tenha transições a chegar**

- Se necessário, acrescenta-se um novo estado inicial com uma transição em ε para o antigo

antes



depois



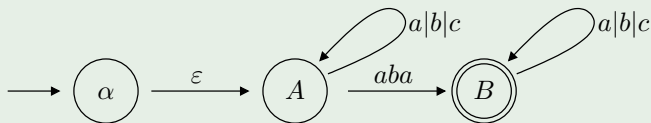
Conversão de um AFG numa ER

Ilustração com um exemplo

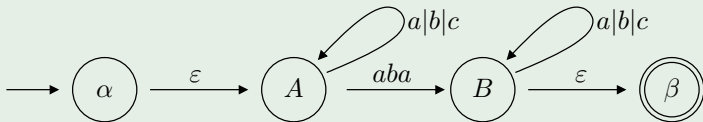
② transformação de um AFG noutro com **um único estado de aceitação e sem transições de saída**

- 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

antes



depois



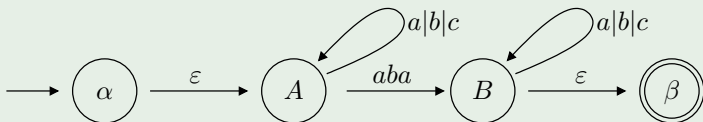
Conversão de um AFG numa ER

Ilustração com um exemplo

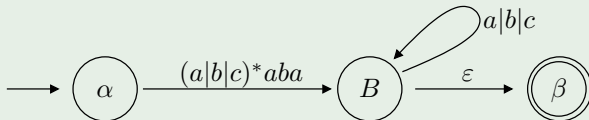
3 Eliminação dos restantes estados

- Os estados são eliminados um a um, em processos de transformação que mantêm a equivalência
- Comece-se pelo estado A

antes



depois da eliminação de A



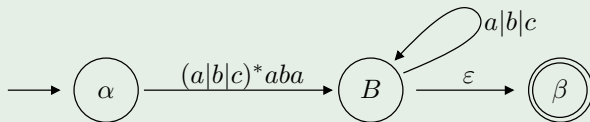
Conversão de um AFG numa ER

Ilustração com um exemplo

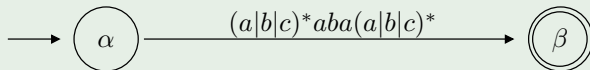
3 Eliminação dos restantes estados

- Os estados são eliminados um a um, em processos de transformação que mantêm a equivalência
- Remove-se agora o estado B

depois da eliminação de A



depois da eliminação de A , seguido da eliminação de B

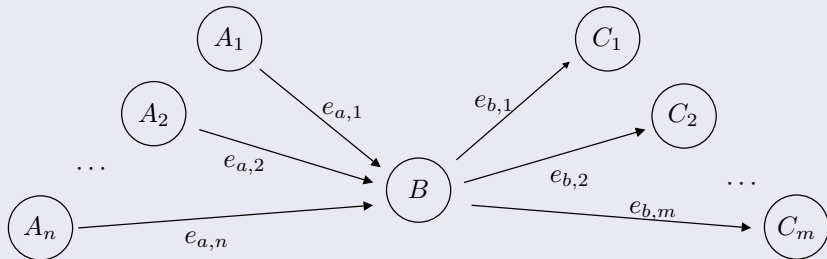


- Sendo $(a|b|c)^*aba(a|b|c)^*$ a expressão regular pretendida

Conversão de um AFG numa ER

Algoritmo de eliminação de um estado

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

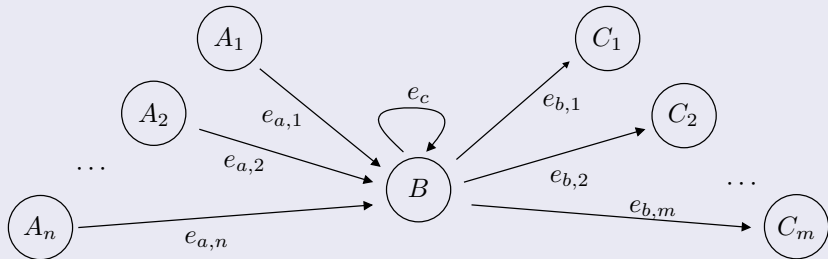


- Pode acontecer que haja $A_i = C_j$
- Para ir de A_i para C_j através de B , para $i = 1, 2, \dots, n$ e $j = 1, 2, \dots, m$, é preciso uma palavra que encaixe na expressão regular $(e_{a,i})(e_{b,j})$
- Então, se se retirar B , é preciso acrescentar uma transição de A_i para C_j que contemple essas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

Conversão de um AFG numa ER

Algoritmo de eliminação de um estado

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

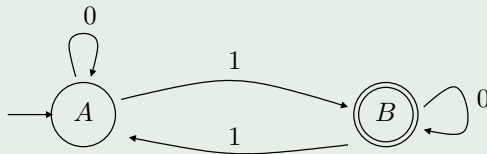


- Pode acontecer que haja $A_i = C_j$
- Para ir de A_i para C_j através de B , para $i = 1, 2, \dots, n$ e $j = 1, 2, \dots, m$, é preciso uma palavra que encaixe na expressão regular $(e_{a,i})(e_c)^*(e_{b,j})$
- Então, se se retirar B , é preciso acrescentar uma transição de A_i para C_j que contemple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

Conversão de um AFG numa ER

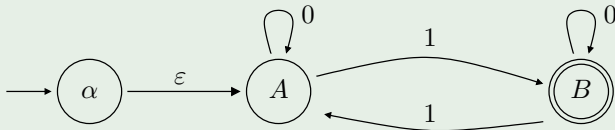
Exercício

Q Obtenha uma ER equivalente ao AF seguinte



R Aplique-se passo a passo o algoritmo de conversão

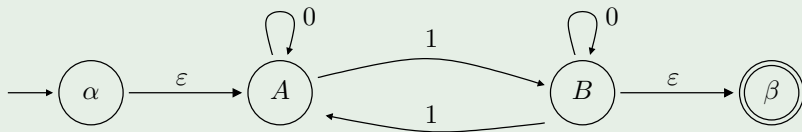
- Porque o estado inicial possui uma transição a entrar, deve substituir-se o estado inicial, de acordo com o passo 1 do algoritmo



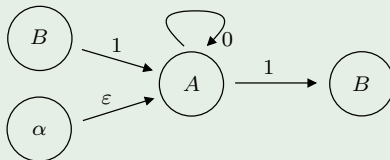
Exemplo de conversão de um AFG numa ER

Exercício

- Porque o estado de aceitação possui uma transição a sair, deve-se aplicar o passo 2 do algoritmo de conversão



- Elimine-se o estado A . Para isso é preciso ver os segmentos de caminhos que passam por A .

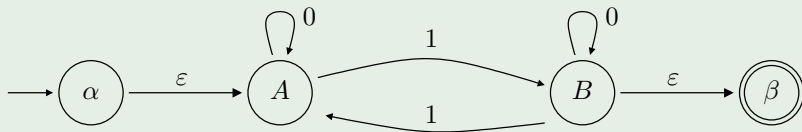


- Note que B aparece à esquerda e à direita

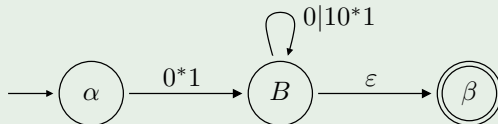
Exemplo de conversão de um AFG numa ER

Exercício

- Porque o estado de aceitação possui uma transição a sair, deve-se aplicar o passo 2 do algoritmo de conversão



- Eliminando o estado A obtém-se



- Finalmente, eliminando o estado B obtém-se a ER $0^*1(0|10^*1)^*$

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 um AF, 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

Conversão de um AF numa GR

Procedimento de conversão

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

$$T = A$$

$$N = Q$$

$$S = q_0$$

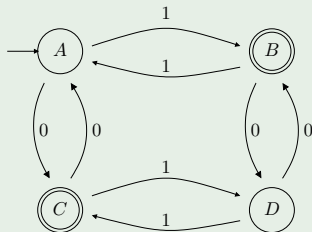
$$P = \{p \rightarrow a q : p, q \in Q \wedge a \in T \wedge (p, a, q) \in \delta\} \\ \cup \{p \rightarrow \varepsilon : p \in F\}$$

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

Conversão de um AF numa GR

Exemplo

Q Determine uma GR equivalente ao AF



R

$A \rightarrow 0 C \mid 1 B$

$B \rightarrow 0 D \mid 1 A \mid \varepsilon$

$C \rightarrow 0 A \mid 1 D \mid \varepsilon$

$D \rightarrow 0 B \mid 1 C$

Conversão de uma GR num AFG

Procedimento de conversão

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

$$A = T$$

$$Q = N \cup \{q_f\}, \quad \text{com } q_f \notin N$$

$$q_0 = S$$

$$F = \{q_f\}$$

$$\delta = \{(q_i, e, q_j) : q_i, q_j \in N \wedge e \in T^* \wedge q_i \rightarrow e q_j \in P\} \\ \cup \{(q, e, q_f) : q \in N \wedge e \in T^* \wedge q \rightarrow e \in P\}$$

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

Conversão de uma GR num AFG

Exemplo

Q Determine um AFG equivalente à GR

$$S \rightarrow a S \mid b S \mid c S \mid a b a X$$

$$X \rightarrow a X \mid b X \mid c X \mid \varepsilon$$

R

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

$$A = \{a, b, c\}$$

$$Q = \{S, X, q_f\}$$

$$q_0 = S$$

$$\delta = \{(S, a, S), (S, b, S), (S, c, S), (S, aba, X), \\ (X, a, X), (X, b, X), (X, c, X), (X, \varepsilon, q_f)\}$$

$$F = \{q_f\}$$

