Expressões Regulares e Linguagem Regular

Prof. Hamilton José Brumatto

Bacharelado em Ciência da Computação - UESC

25 de dezembro de 2024

Expressões Regulares

- As expressões regulares, costumam ser abreviadas como "regex" ou "regexp"
- São utilizadas em programação e processamento de texto para lidar com padrões de caracteres
- Ferramentas como AWK, GAWK, SED e outras são usadas para manipular texto buscando padrões definidos pelas expressões regulares.
- O uso das expressões regulares no contexto de linguagem regular é um pouco mais restrita do conceito geral.
- As expressões regulares são construídas com as três operações regulares: $\cup$, $\circ$ e *. Também ordem de precedência definida com parêntesis.

Um exemplo de Expressão Regular

- Vamos considerar a expressão matemática: $(3+8)\times 2$.
- Esta expressão pode ser realizada com o resultado 22.
- De forma equivalente podemos pensar em $(\{0\}\cup\{1\})\circ\{0\}^*$
- Esta última podemos descrever como um cadeia w tal que w comece com 0 ou 1 e é seguida por nenhum ou vários 0s: $\{0, 1, 00, 10, 000, 100, \ldots\}$
- Apesar de se aplicar a conjuntos representando as cadeias, há uma "liberdade" de expressão onde omite-se a indicação de conjunto, até mesmo a operação de concatenação pode ser omitida: $(0\cup 1)0^*$
- Se o alfabeto $\Sigma=\{0,1\}$ podemos indicar Σ como representando qualquer caracter: $(0\cup 1)$: $\Sigma^*=(0\cup 1)^*$.
- Esta última representa todas as cadeias construídas com o alfabeto, até mesmo a cadeia vazia.

Definição Formal

R é uma ***expressão regular*** se R for:

1. a para algum a no alfabeto Σ.
2. ε
3. $\emptyset$
4. $R_1 \cup R_2$, onde R_1 e R_2 são expressões regulares.
5. $R_1 \circ R_2$, onde R_1 e R_2 são expressões regulares.
6. R_1^*, onde R_1 é uma expressão regular.

Os dois primeiros representam as linguagens $\{a\}$ e $\{\varepsilon\}$ respectivamente. No terceiro, é a linguagem vazia. Nos demais temos as operações regulares.

Observe que $\{\varepsilon\}$ e $\emptyset$ são coisas distintas, a primeira representa a linguagem com a cadeia vazia, e a segunda uma linguagem sem nenhuma cadeia.

Construção de Expressões

Existe uma relação de precedência, primeiro a operação estrela: *, a segunda a operação de concatenação: $\circ$ e a terceira a de união: $\cup$.
Quando a ordem prevalece pode-se omitir parêntesis, caso contrário a precedência se dá por operações entre parêntesis.
A expressão a^* representa uma cadeia com nenhum, um ou mais caracteres a. Por conveniência para representar uma cadeia que aceita ao menos 1 caracter a: aa^* pode ser representada como: a^+.
A operação estrela: * sempre inclui a cadeia vazia, então $\emptyset^* = \{\varepsilon\}$
Situações especiais nas operações: $R \cup \emptyset = R$; $R \circ \emptyset = \emptyset$; $R \circ \varepsilon = R$; $R \cup \varepsilon$ é uma linguagem que passa a incluir a cadeia vazia caso $L(R)$ não a incluísse.

Algumas expressões

1. $0^*10^* = \{w | w \text{ contém um único } 1\}$
2. $\Sigma^*1\Sigma^* = \{w | w \text{ contém pelo menos um } 1\}$
3. $\Sigma^*001\Sigma^* = \{w | w \text{ contém a cadeia } 001 \text{ como subcadeia}\}$
4. $1^*(01^+)^* = \{w | \text{ todo } 0 \text{ em } w \text{ é seguido por pelo menos um } 1\}$
5. $(\Sigma\Sigma)^* = \{w | w \text{ tem comprimento par}\}$
6. $(01 \cup 10) = \{01, 10\}$
7. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w | w \text{ começa e termina com o mesmo símbolo}\}$
8. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
9. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$
10. Seja o alfabeto $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ e $\Sigma = \{+, -, .\} \cup D$, a expressão: $(+ \cup - \cup \varepsilon)(D^+ \cup D^+.D^* \cup D^*.D^+)$ representa qualquer cadeia conhecida por números de ponto flutuante ou inteiro, com ou sem sinal: 72, 3.14159, +7, -.01, 2.

Equivalência com Autômatos Finitos

Sabemos que um autômato finito representa uma linguagem, o mesmo acontece com expressões regulares.

Uma expressão regular representa uma linguagem

Com base nisto, um autômato finito pode ser transformado em uma expressão regular, e vice-versa.

Teorema

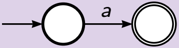
Uma linguagem é regular se e somente se alguma expressão regular a descreve

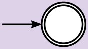
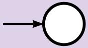
A ideia da prova é transformar expressões regulares em automatos finitos e vice-versa.

Uma linguagem é regular se uma expressão regular a descreve: $R : regexp \rightarrow L(R)$

Prova

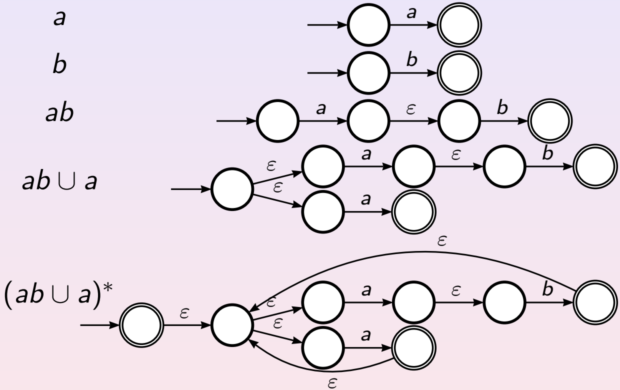
Vamos converter R em um AFN: R

1. $R = a$ *para algum* $a \in \Sigma$: $L(R) = \{a\}$:


$N = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$:
$\delta(q_1, a) = \{q_2\}, \delta r, b) = \emptyset | r \neq q_1 \vee b \neq a$
2. $R = \varepsilon$: $LR = \{\varepsilon\}$:

$N = (\{q\}, \Sigma, \delta, q, \{q\}), \delta(r, b) = \emptyset\ \forall r, b$
3. $R = \emptyset$: $L(R) = \emptyset$:

$N = (\{q\}, \Sigma, \delta, q, \emptyset), \delta(r, b) = \emptyset\ \forall r, b$
4. $R = R_1 \cup R_2$
5. $R = R_1 \circ R_2$
6. $R = R_1^*$

□

Exemplos



Desafio: Qual a linguagem regular que a expressão regular $(a \cup b)^* aba$ representa?

Uma linguagem é regular somente se uma expressão regular a descreve: $R : L(R) \rightarrow regexp$

Os principais passos para demonstrar este lema envolve o conceito de ***automato finito não-determinístico generalizado*** (AFNG), neste autômato a transição entre um estado e outro se dá não apenas por um símbolo do alfabeto, mas também por uma expressão regular. Então os passos da prova são:

Prova

1. *A partir da Linguagem criamos um AFD ou AFN.*
2. *Modificamos este para um AFNG incluindo um estado q_{ini} de início e um estado q_{ack} final diferente de todos os demais.*
3. *Transformamos o AFNG contraindo os estados intermediários um a um até que reste apenas q_{ini} e q_{ack}.*
4. *A expressão regular que realiza a transição entre q_{ini} e q_{ack} é a expressão regular que descreve a Linguagem.*

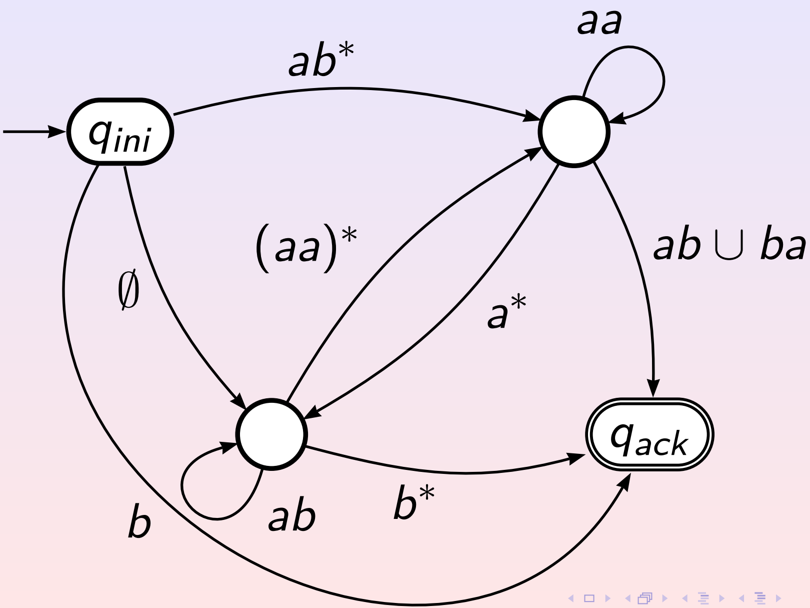
□

AFNG

Um AFNG possui as seguintes características:

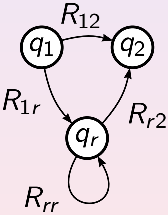
- Um estado de início: q_{ini} que só possui transições que saem deste estado.
- Um único estado de aceitação q_{ack} que só possui transições que chegam a este estado.
- Todos os demais estados intermediários tem transições que saem para todos os outros estados (exceto o q_{ini}) e transições que vem de todos os outros estados (exceto o q_{ack}).
- Cada estado intermediário tem também transições para si mesmo.

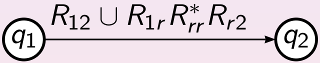
AFNG - Exemplo



Redução de estado

Entre 2 estados quaisquer q_1 e q_2 existe um q_r que será removido. A transição entre q_1 e q_2 será a transição direta, união com a transição que passe por q_r:





Formalização de AFNG e computabilidade

Um *autômato finito não-determinístico generalizado* é uma 5-upla, $(Q, \Sigma, \delta, q_{ini}, q_{ack})$, onde:

1. Q é o conjunto finito de estados.
2. Σ é o alfabeto de entrada.
3. $\delta : (Q \setminus q_{ini}) \times (Q \setminus q_{ack}) \to R$ é a função de transição.
4. q_{ini} é o estado inicial, e
5. q_{ack} é o estado de aceitação.

Um AFNG aceita uma cadeia w em Σ^* se $w = w_1 w_2, \cdots, w_k$, onde w_i está em Σ^*, e esiste uma sequência de estados $q_0, q_1, \cdots, q_k$ tal que:

1. $q_0 = q_{ini}$ é o estado inicial,
2. $q_k = q_{ack}$ é o estado de aceitação, e
3. para cada i, temos $w_i \in L(R_i)$, onde $R_i = \delta(q_{i-1}, q_i)$; em outras palavras, R_i é a expressão regular sobre a transição entre q_{i-1} e q_i.

Passo a passo da prova:

A partir da Linguagem, criamos um AFD ou AFN.

$$N = (Q, \Sigma, \delta, q_0, F)$$

Modificamos este para um AFNG incluindo um estado q_{ini} de início e um estado q_{ack} final diferente de todos os demais.

$$G = ((Q \cup \{q_{ini}, q_{ack}\}), \Sigma, \delta', q_{ini}, q_{ack})$$

$$\delta'(q_1, q_2) = \begin{cases} R : \biguplus_{\{w\}} & \delta(q_1, w) \rightarrow q_2 \ \forall \ q_1, q_2 \in Q \\ \{\varepsilon\} & q_1 = q_{ini} \wedge q_2 = q_0 \\ \emptyset & q_1 = q_{ini} \wedge q_2 \in Q \setminus q_0 \cup \{q_{ack}\} \\ \{\varepsilon\} & q_1 \in F \wedge q_2 = q_{ack} \\ \emptyset & q_1 \in Q - F \wedge q_2 = q_{ack} \end{cases}$$

Passo a passo da prova:

Transformamos o AFNG contraindo os estados intermediários um a um até que reste apenas q_{ini} e q_{ack}.
A expressão regular que realiza a transição entre q_{ini} e q_{ack} é a expressão regular que descreve a Linguagem.

$$\text{Dado } G = (Q, \Sigma, \delta, q_{ini}, q_{ack}) \text{ com } k = |Q|$$

1. Se $k = 2$ então $Q = \{q_{ini}, q_{ack}\}$ e $\delta(q_{ini}, q_{ack}) \to R$ e R é a expressão regular que reconhece a linguagem.
2. Se $k > 2$, seleciona-se $q_{rem} \in Q - \{q_{ini}, q_{ack}\}$, constrói-se G' o AFNG $(Q', \Sigma, \delta', q_{ini}, q_{ack})$, onde:
   - $Q' = Q \setminus q_{rem}$,
   - $\forall q_1, q_2, q_1 \in Q' \setminus q_{ini} \wedge q_2 \in Q' \setminus q_{ack} : \delta'(q_1, q_2) = \delta(q_1, q_2) \cup \delta(q_1, q_{rem}) \circ \delta^*(q_{rem}, q_{rem}) \circ \delta(q_{rem}, q_2)$

A demonstração desta prova se dá por indução

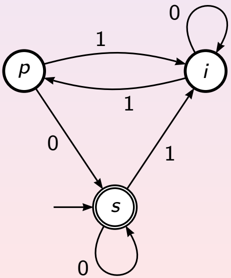
Exemplo: Definindo um AFN

Linguagem regular: $\Sigma = \{0, 1\}$

$L(R) = \{w | w$ é vazia ou contém um número par de 1s e termina com 0$\}$

Um AFN para a linguagem: $F = (\{s, p, i\}, \{0, 1\}, \delta, s, s)$ onde

δ	0	1
s	s	i
p	s	i
i	i	p

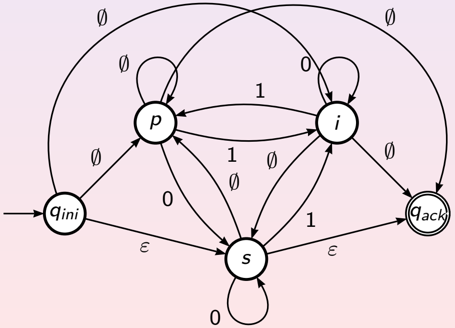


Exemplo: Gerando um AFNG

Linguagem regular: $\Sigma = \{0, 1\}$

$L(R) = \{w | w$ é vazia ou contém um número par de 1s e termina com 0$\}$

$G = (\{q_{ini}, s, p, i, q_{ack}\}, \{0, 1\}, \delta, s, s)$, onde



δ	s	p	i	q_{ack}
q_{ini}	ε	$\emptyset$	$\emptyset$	$\emptyset$
s	0	$\emptyset$	1	ε
p	0	$\emptyset$	1	$\emptyset$
i	$\emptyset$	1	0	$\emptyset$

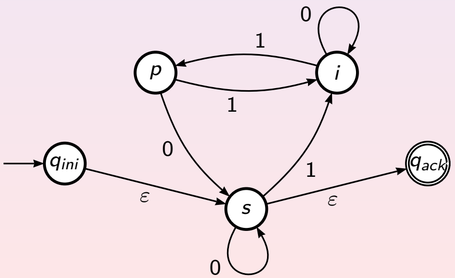
Exemplo: Gerando um AFNG (sem as $\emptyset$)

Linguagem regular: $\Sigma = \{0, 1\}$

$L(R) = \{w | w$ é vazia ou contém um número par de 1s e termina com $0\}$

$G = (\{q_{ini}, s, p, i, q_{ack}\}, \{0, 1\}, \delta, q_{ini}, q_{ack})$, onde

δ	s	p	i	q_{ack}
q_{ini}	ε	$\emptyset$	$\emptyset$	$\emptyset$
s	0	$\emptyset$	1	ε
p	0	$\emptyset$	1	$\emptyset$
i	$\emptyset$	1	0	$\emptyset$



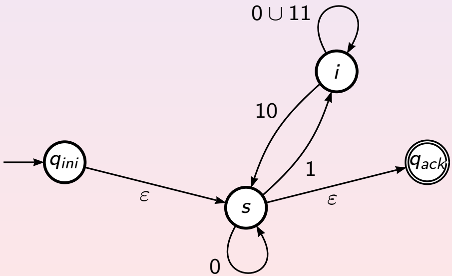
Exemplo: Gerando um AFNG (sem o estado p)

Linguagem regular: $\Sigma = \{0, 1\}$

$L(R) = \{w | w$ é vazia ou contém um número par de 1s e termina com 0$\}$

$G = (\{q_{ini}, s, i, q_{ack}\}, \{0, 1\}, \delta, q_{ini}, q_{ack})$, onde

δ	s	i	q_{ack}
q_{ini}	ε	$\emptyset$	$\emptyset$
s	0	1	ε
i	10	$0 \cup 11$	$\emptyset$



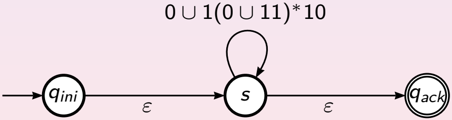
Exemplo: Gerando um AFNG (sem o estado i)

Linguagem regular: $\Sigma = \{0,1\}$

$L(R) = \{w|w$ é vazia ou contém um número par de 1s e termina com $0\}$

$G = (\{q_{ini}, s, q_{ack}\}, \{0,1\}, \delta, q_{ini}, q_{ack})$, onde

δ	s	q_{ack}
q_{ini}	ε	$\emptyset$
s	$0 \cup 1(0 \cup 11)^*10$	ε



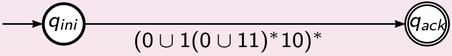
Exemplo: Gerando um AFNG (sem o estado s)

Linguagem regular: $\Sigma = \{0,1\}$

$L(R) = \{w|w$ é vazia ou contém um número par de 1s e termina com $0\}$

$G = (\{q_{ini}, s, q_{ack}\}, \{0,1\}, \delta, q_{ini}, q_{ack})$, onde

δ	q_{ack}
q_{ini}	$(0 \cup 1(0 \cup 11)^*10)^*$



$R = (0 \cup 1(0 \cup 11)^*10)^*$

Atividades baseada no Sipser

- Ler a seção 1.3
- Resolver os exercícios: 1.18, 1.19, 1.20, 1.21, 1.22, 1.23, 1.28,