Expressões Regulares

Linguagens Formais e Autômatos

Prof. Daniel Saad Nogueira Nunes

INSTITUTO FEDERAL
Brasília
Campus
Taguatinga

Sumário

Introdução

- Expressões regulares são mecanismos que conseguem denotar linguagens regulares.
- Possibilitam uma série de aplicações relacionadas a casamento de padrões e construção de tradutores.
- Possuem uma proximidade grande com os ε-NFA.
- Por uma notação compacta, similar as expressões algébricas, conseguimos expressar qualquer linguagem regular.

Introdução

- Até o momento, vimos como capturar linguagens por formalismos que lembram máquinas, como os nossos autômatos finitos.
- Expressões regulares, por sua vez, são um mecanismo declarativo. Ao montar uma expressão regular, estamos indicando quais palavras queremos aceitar.

Introdução

- Diversas aplicações como o `grep` ou editores de texto possibilitam formas de buscar padrões utilizando expressões regulares.
- Analisadores léxicos, como o `Flex` ou o `Lex`, utilizam expressões regulares para classificar o código-fonte em *tokens*.

Sumário

Expressões regulares

- Expressões regulares **denotam** linguagens.
- Por exemplo: a expressão $01^* + 10^*$ denota a linguagem que consiste das palavras que começam com um 0 e são seguidas por qualquer número de 1s, ou que começam com um 1 e são seguidas por qualquer número de 0s.
- Falaremos agora de algumas operações que podem ser realizadas em linguagens para então explicar como os operadores das expressões regulares as representam.

Operações em Linguagens

União de Linguagens

A união de duas linguagens L e M é denotada por $L \cup M$. Por exemplo se $L = \{001, 10, 111\}$ e $M = \{\varepsilon, 001\}$, então $L \cup M = \{\varepsilon, 10, 001, 111\}$.

Operações em Linguagens

Concatenação de Linguagens

A concatenação das linguagens L e M, denotada por LM é o conjunto de palavras que pode ser formado concatenando uma palavra qualquer de L com outra palavra qualquer de M.

Por exemplo, se $L = \{001, 10, 111\}$ e $M = \{\varepsilon, 001\}$, então, $LM = \{001, 10, 111, 001001, 10001, 111001\}$.

Caso uma linguagem L seja concatenada com ela mesma i vezes, temos

$$L^i = \underbrace{LL \dots L}_{i \text{ vezes}}$$

Em especial, $L^0 = \{\varepsilon\}$.

Operações em Linguagens

Fecho Kleene, ou estrela

Se L é uma linguagem, então L^* é formada pela concatenação de zero ou mais strings de L. Inclusive a mesma string pode ser utilizada várias vezes. Formalmente temos:

$$L^* = L^0 \cup L^1 \cup L^2 \ldots = \bigcup_{i \geq 0} L_i$$

Construção de Expressões Regulares

- Expressões são formadas a partir da composição de expressões elementares com operadores.
- Descreveremos as expressões regulares recursivamente utilizando operadores representando cada uma das operações em linguagens apresentadas.
- Para cada expressão regular E, $L(E)$ denotará a linguagem representada pela expressão E.

Expressões regulares

Casos base

As expressões regulares elementares podem ser categorizadas em três:

- **Constantes**: ε e $\emptyset$ são expressões regulares que denotam, respectivamente as linguagens $L(\varepsilon) = \{\varepsilon\}$ e $L(\emptyset) = \emptyset$.
- **Símbolos**: se a é um símbolo qualquer, a denota a linguagem $\{a\}$.
- **Variáveis**: uma variável L representa uma linguagem L qualquer.

Expressões regulares

Indução

1. **Operador +**: se E e F são expressões regulares, então $E + F$ é uma expressão regular denotando $L(E)$ ou $L(F)$. Em outras palavras: $L(E + F) = L(E) \cup L(F)$.
2. **Concatenação**: se E e F são expressões regulares, então EF representa a concatenação de $L(E)$ com $L(F)$. Isto é, $L(EF) = L(E)L(F)$.
3. **Estrela**: se E é uma expressão regular, então E^* é uma expressão regular que indica o fecho Kleene de E. Em outras palavras, $L(E^*) = (L(E))^*$.

Expressões regulares

Indução

1. **Operador + sobrescrito**: Se E é uma expressão regular, então E^+ denota EE^*, isto é, todas as palavras que podem ser formadas por uma ou mais concatenações das palavras que estão em $L(E)$. $L(E^+) = L(EE^*)$.
2. **Parênteses**: Se E é uma expressão regular, (E) também é, e $L(E) = L((E))$.

Expressões Regulares

Precedência dos operadores

estrela $>$ concatenação $>$ união

Sumário

Exemplos

Considerando o alfabeto $\Sigma = \{a, b\}$.

E	$L(E)$
aa	{aa}
ba^*	Palavras que começam com b e são seguidas por zero ou mais a's
$\mathrm{a}^*\mathrm{ba}^*$	Palavras que contém um único b
$(\mathrm{a}+\mathrm{b})^*\mathrm{b}(\mathrm{a}+\mathrm{b})^*$	Palavras que tem, **pelo menos**, um b
$(\mathrm{a}+\mathrm{b})^*\mathrm{aba}(\mathrm{a}+\mathrm{b})^*$	Palavras que contém a subpalavra aba
$\mathrm{b}^*(\mathrm{ab}^+)^*$	Palavras em que todo a é seguido por, **pelo menos**, um b
$((\mathrm{a}+\mathrm{b})(\mathrm{a}+\mathrm{b}))^*$	Palavras de comprimento par
$\mathrm{ab}+\mathrm{ba}$	Somente as palavras $\{ab, ba\}$
$(\mathrm{a}+\mathrm{b})^*$	Σ^*
$(\mathrm{a}+\mathrm{b})^*\mathrm{aa}(\mathrm{a}+\mathrm{b})^*$	Todas as palavras que contém a subpalavra aa
$(\mathrm{a}+\mathrm{b})^*(\mathrm{aa}+\mathrm{bb})$	Palavras que terminam com aa ou bb

Sumário

Sumário

Equivalência

- REGEX $\Rightarrow$ ε-NFA
- ε-NFA $\Rightarrow$ REGEX

Equivalência com ε-NFA

- Toda expressão regular pode ser convertida em um ε-NFA que reconhece a mesma linguagem.
- A construção do ε-NFA é realizada recursivamente.
- Utilizaremos um argumento baseado em **indução estrutural**.

Equivalência com ε-NFA

Caso base: $\emptyset$

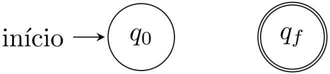


Figura: ε-NFA para a expressão regular $\emptyset$

Equivalência com ε-NFA

Caso base: ε

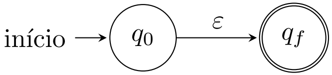


Figura: ε-NFA para a expressão regular ε

Equivalência com ε-NFA

Caso base: a

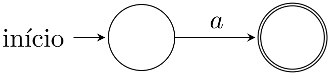


Figura: ε-NFA para a expressão regular a

Equivalência com ε-NFA

Passo indutivo: $R + S$

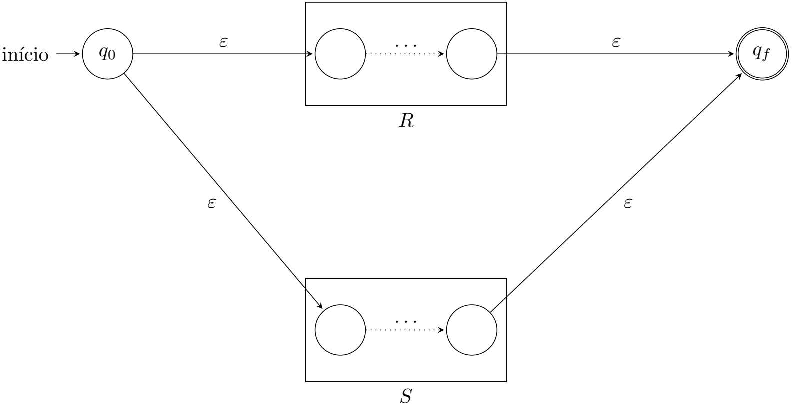


Figura: ε-NFA para a expressão regular $\mathrm{R} + \mathrm{S}$

Equivalência com ε-NFA

Passo indutivo: RS

Figura: ε-NFA para a expressão regular RS

Equivalência com ε-NFA

Passo indutivo: R^*

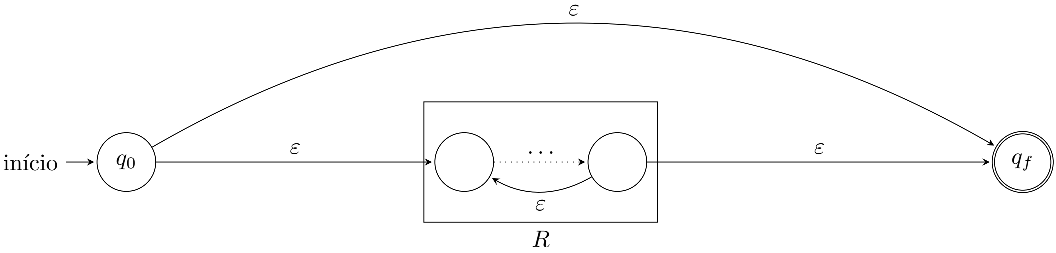


Figura: ε-NFA para a expressão regular R^*

Sumário

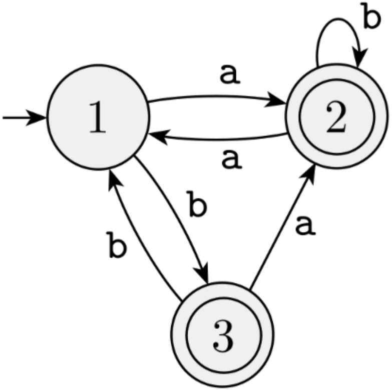
Equivalência

- REGEX $\Rightarrow$ ε-NFA
- ε-NFA $\Rightarrow$ REGEX

Equivalência com ε-NFA

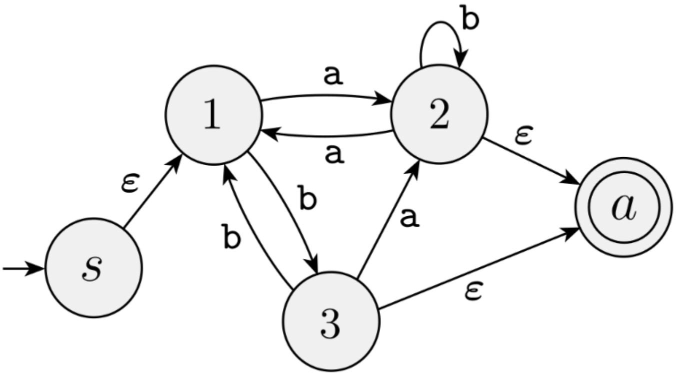
- Todo ε-NFA é convertível em uma expressão regular.
- A ideia é usar um ε-NFA generalizado. Com um novo estado de aceitação e rejeição.
- Cada transição nesse ε-NFA generalizado é rotulada com uma expressão regular.
- A cada passo, um estado intermediário é removido, até que reste apenas o estado inicial e o estado de aceitação.
- Quando um estado intermediário é removido, as transições que passavam por ele são atualizadas com novas expressões regulares.
- No final, a expressão regular que rotula a transição do estado inicial para o estado final é a expressão regular que denota a linguagem reconhecida pelo ε-NFA original.

Equivalência com ε-NFA



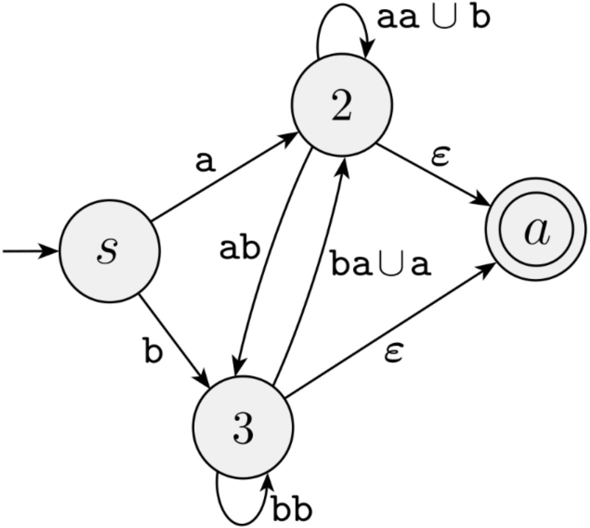
(a)

Equivalência com ε-NFA

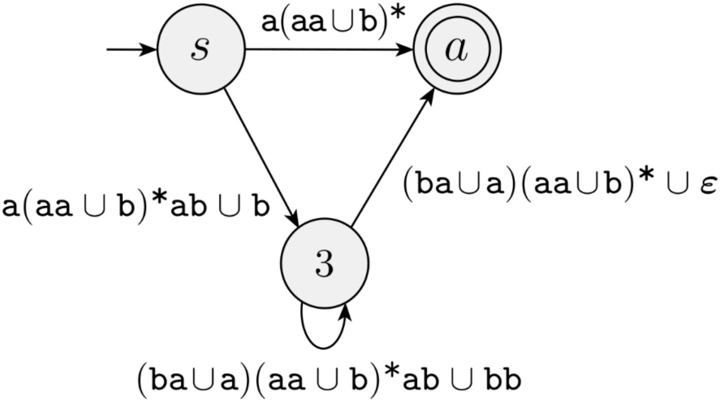


(b)

Equivalência com ε-NFA

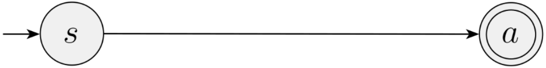


Equivalência com ε-NFA



(d)

Equivalência com ε-NFA



$$(\mathtt{a}(\mathtt{aa}\cup\mathtt{b})^*\mathtt{ab}\cup\mathtt{b})((\mathtt{ba}\cup\mathtt{a})(\mathtt{aa}\cup\mathtt{b})^*\mathtt{ab}\cup\mathtt{bb})^*((\mathtt{ba}\cup\mathtt{a})(\mathtt{aa}\cup\mathtt{b})^*\cup\varepsilon)\cup\mathtt{a}(\mathtt{aa}\cup\mathtt{b})^*$$

Sumário

Equivalência de formalismos

$$
\begin{tikzcd}
\text{DFA} \arrow[d, leftrightarrow] \\
\text{NFA} \arrow[d, leftrightarrow] \\
\varepsilon\text{-NFA} \arrow[d, leftrightarrow] \\
\text{REGEX}
\end{tikzcd}
$$