Expressões regulares

Vimos nas aulas anteriores basicamente duas formas distintas de se expressar uma linguagem regular: (1) utilizando notação de conjuntos e suas operações; e (2) utilizando autômatos finitos. Embora esses formalismos sejam úteis em situações específicas, por exemplo, AFs são úteis sob a ótica de desenvolvimento de reconhecedores de linguagens, em outras situações eles não são tão úteis. Por exemplo, considere o problema de se especificar um padrão a ser buscado em um texto. Seria muito mais conveniente se houvesse um formalismo que permitisse a expressão linear de uma linguagem, ou seja, do padrão que ela representa. Hoje estudaremos um formalismo muito empregado nesses casos: as expressões regulares.

As expressões regulares são bastante populares por suas aplicações na especificação de padrões em sistemas operacionais e editores de texto. Ao contrário dos autômatos finitos que determinam um reconhecedor de uma linguagem, diz-se que as expressões regulares denotam uma linguagem.

★ Formalmente, uma **expressão regular** é recursivamente definida por:

- $\emptyset, \lambda, a \in \Sigma$ são expressões regulares, denotando, respectivamente, as linguagens $\emptyset, \{\lambda\}, \{a\}$
- Se r e s são expressões regulares, então $(r + s), rs, r^*$ também são expressões regulares que denotam as linguagens $L(r) \cup \mathrm{L}(s), L(r)L(s), L(r)^*$ respectivamente.

⚑ Exemplo: Considerando $\Sigma = \{0,1\}$, são expressões regulares denotando linguagens regulares:

- $\emptyset \rightarrow \emptyset$
- $\lambda \rightarrow \{\lambda\}$
- $(01) \rightarrow \{0\}\{1\} = \{01\}$
- $(((0 + 1)^*1)(0 + 1)) \rightarrow (\{0,1\}^*\{1\})\{0,1\}$

Podemos definir a precedência dos operadores nas expressões regulares para omitir alguns dos parênteses. Isso será bastante útil uma vez que, em expressões maiores, a quantidade de parênteses também é grande o que dificulta a escrita e leitura dessas expressões. A precedência dos operadores e regras para omissão de parênteses é a seguinte:

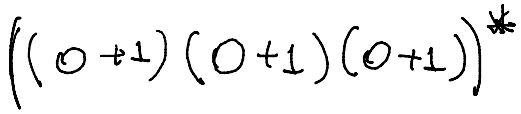
1. O fecho de Kleene tem a maior precedência entre os operadores, seguido da concatenação e, por último, a união.
2. Parênteses externos podem ser omitidos da expressão
3. Como a união e concatenação são associativas, parênteses delimitando sequências dessas operações podem ser omitidos. Por exemplo, $(r_1 + (r_2 + (r_3 + r_4)))$ pode ser expresso simplesmente por $r_1 + r_2 + r_3 + r_4$.

Assim, a última expressão do exemplo anterior pode ser escrita como $(0 + 1)^*1(0 + 1)$.

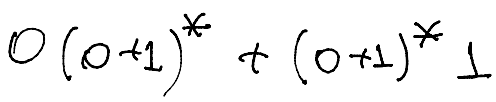
⚑ Exemplo: Considerando $\Sigma = \{0,1\}$, determine expressões regulares para as seguintes

linguagens:

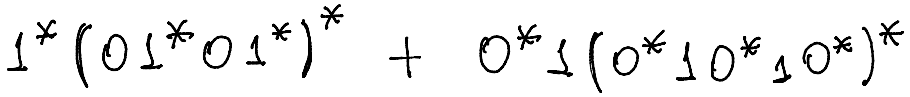
- O conjunto das palavras de tamanho múltiplo de 3



- Linguagem das palavras que começam com 0 ou terminam com 1



- Conjunto das palavras com um número par de 0s ou ímpar de 1s



- Conjunto das palavras que não contêm a substring 101:

Assim como nos autômatos, há diferentes formas de se especificar expressões regulares denotando uma mesma linguagem. Nesse caso, dizemos que essas expressões são equivalentes. Uma série de equivalências válidas com conjuntos também se aplicam a expressões regulares. Essas equivalências, exemplificadas na tabela abaixo, possibilitam a simplificação de algumas expressões.

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

Além dessas, duas simplificações úteis são:

1. $r^+ = rr^* = r^*r$
2. $r^k = rr \ldots r \ (k \ vezes)$