4. Gramáticas e Expressões Regulares

Copiar link

Vimos até agora duas formas de definição de linguagens:

- Definir o conjunto de palavras da linguagem, por extensão e compreensão, utilizando ainda as habituais operações e outras operações definidas sobre conjuntos

- A partir de um autómato A, definir uma linguagem como sendo o conjunto de palavras aceites por A

Existe uma terceira forma, muito importante, que veremos agora.

Uma *gramática* é um tuplo (N, Σ, S, P) em que:
- N é um conjunto de variáveis ou símbolos *não-terminais*
- Σ é um conjunto de símbolos *terminais*
- $S \in N$ é uma variável especial a que chamaremos *Símbolo Inicial*
- P é um conjunto de regras de produção, da forma $\alpha \rightarrow \beta$ com $\alpha, \beta \in (N \cup \Sigma)^*$, palavras formadas por símbolos terminais e não terminais, sendo pelo menos um símbolo de α não-terminal

Exemplo
$G = (\{S, A\}, \{0,1\}, S, \{S \rightarrow 0A1, 0A \rightarrow 00A1, A \rightarrow \varepsilon\})$

Esta gramática sobre o alfabeto $\{0, 1\}$ utiliza, além de um símbolo inicial obrigatório, um outro símbolo não-terminal A, e possui 3 regras de produção.

As regras de produção permitem *derivar* palavras utilizando um mecanismo recursivo baseado em *reescrita:*

> em cada palavra, já derivada, onde **ocorra o lado esquerdo de uma regra de produção**, podemos substituir esse lado esquerdo pelo lado direito da mesma regra, derivando assim uma nova palavra.
> Podemos esquematizar este processo da seguinte forma:
> *Se $S \rightarrow s \in P$ e $s \in \Sigma^*$ (S não contém variáveis), então a palavra s é derivada pela gramática, que escreveremos $S \rightarrow s$*
> *Se $S \rightarrow s$ e $\alpha \rightarrow \beta \in P$ e $s = x\alpha y$, então $S \rightarrow x\beta y$*

A linguagem induzida por esta gramática é definida como o conjunto de palavras derivadas a partir do símbolo inicial: $L_G = \{w \mid w \in \Sigma^* \wedge S \rightarrow w\}$

Assim, nesta gramática temos por exemplo as seguintes derivações:

- $S \rightarrow 0A1 \rightarrow 0\varepsilon 1$, logo $01 \in L_G$
- $S \rightarrow 0A1 \rightarrow 00A11 \rightarrow 00\varepsilon 11$, logo $0011 \in L_G$
- $S \rightarrow 0A1 \rightarrow 00A11 \rightarrow 000A111 \rightarrow 000\varepsilon 111$, logo $000111 \in L_G$

É fácil ver que a gramática G define a linguagem $L_G = \{0^n 1^n \mid n \geq 1\}$, que, como vimos antes, é um exemplo clássico de uma linguagem que *não é regular*.

Daqui se conclui que as gramáticas são em geral mais expressivas do que os autómatos DFA ou NFA, uma vez que, ao contrário daqueles, podem definir linguagens não regulares.

Exercício Resolvido
Considere a linguagem $L = \{a^i b^j \mid i \geq 0 \wedge j > 0\}$. Defina uma gramática para esta linguagem.

Note-se que as palavras de L têm um segmento possivelmente vazio de as seguido de um segmento não-vazio de bs. Introduziremos um símbolo não-terminal B correspondente ao segmento de bs. Assim, para o símbolo inicial S teremos duas produções: $S \rightarrow aS$, que corresponde à aceitação de um número arbitrário de as, e $S \rightarrow B$, que marca o final do primeiro segmento (tendo sido lidos possivelmento zero as). O segmento seguinte pode ser construído por apenas um b, ou então por um b seguido de mais bs. Para o não-terminal B teremos então as produções $B \rightarrow bB$ e $B \rightarrow b$.

Daqui resulta a gramática $(\{S, B\}, \{a, b\}, S, \{S \rightarrow aS, S \rightarrow B, B \rightarrow bB, B \rightarrow b\})$

Em alternativa poderíamos introduzir um símbolo não-terminal A para o segmento inicial, e teríamos
- $S \rightarrow AB$
- $A \rightarrow aA$
- $A \rightarrow \varepsilon$
- $B \rightarrow bB$
- $B \rightarrow b$

Gramáticas Regulares

É no entanto possível restringir a definição de gramática por forma a que as linguagens geradas sejam exactamente as linguagens regulares.

Uma gramática regular é uma gramática em que todas as regras de produção $\alpha \rightarrow \beta \in P$:
- α é apenas um símbolo não-terminal, e
- β é apenas um símbolo terminal, OU um símbolo terminal seguido de um símbolo não terminal, OU ε

Exemplo
$G_{01Rep} = (\{S, A\}, \{0, 1\}, S, \{S \rightarrow 0A, A \rightarrow 1S, S \rightarrow \varepsilon\})$

Exemplos de palavras derivadas por G incluem $01, 0101, 010101$. De facto a linguagem definida é-nos familiar de +1. Linguagens :

$L_{01Rep} = \{(01)^i \mid i \geq 0\}$

Expressões Regulares

No caso concreto das linguagens regulares, é habitual utilizar-se um outro mecanismo, em alternativas às gramáticas regulares. As expressões regulares (ERs) sobre um alfabeto Σ definem-se recursivamente da seguinte forma:

- ε é uma ER
- a é uma ER, com $a \in \Sigma$

- se r é uma ER, então (r) e r^* são ERs
- se r_1, r_2 são ERs, então $r_1 r_2$ e $r_1 + r_2$ são ERs

A noção de linguagem $L(r)$ definida pela expressão regular r define-se recursivamente da seguinte forma:

- $L(\varepsilon) = \{\varepsilon\}$
- $L(a) = \{a\}$
- $L((r)) = L(r)$
- $L(r^*) = (L(r))^*$ — note-se diferentes interpretações de *, como construtor de ERs ou de linguagens
- $L(r_1 r_2) = L(r_1)L(r_2)$ — concatenação de linguagens
- $L(r_1 + r_2) = L(r_1) \cup L(r_2)$

Exemplos

- $(0 + \varepsilon)(1 + \varepsilon)$ corresponde à linguagem $\{\varepsilon, 0, 1, 01\}$
- $0 + 10^*$ corresponde à linguagem $\{0, 1, 10, 100, 1000, 10000, \ldots\}$
- (0^*10^*) corresponde à linguagem $\{0^i 10^j \mid i, j \geq 0\}$
- $(a + b)^*$ corresponde à linguagem de todas as palavras constituídas por as e bs, incluindo ε
- $(aa)^*$ corresponde à linguagem $\{a^{2i} \mid i \geq 0\}$
- $(0^*10^*10^*)^*$ linguagem de todas as palavras constituídas por 0s e 1s, com um número par de 1s

Exercício

Escreva expressões regulares que descrevam cada uma das seguintes linguagens.

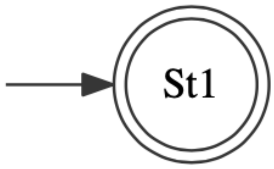
1. Palavras constituídas pelos símbolos ‘a’ e ‘b’, em qualquer número e ordem, terminando com ‘aaa’.
2. Palavras constituídas por um número par de 0s seguido de um número ímpar de 1s.
3. Palavras constituídas por um número par de símbolos 0 ou 1, em qualquer ordem.
4. Palavras sobre o alfabeto $\{0, 1\}$ que têm um número ímpar de 1s, **ou** exactamente três 0s

Conversão de ERs em NFAs

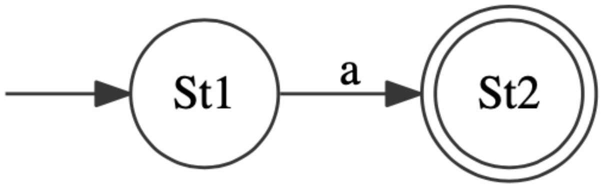
Foi já referido que o poder descritivo das expressões regulares e dos autómatos DFA/NFA é o mesmo. É pois possível, a partir de uma ER, obter um autómato que reconhece exactamente a mesma linguagem. Veremos agora como pode ser construído um autómato NFA.

Previsivelmente, a construção é recursiva. Relembre que os autómatos não-determinísticos podem ter mais do que um estado inicial, e podem também ter transições espontâneas, que associamos à leitura da palavra vazia

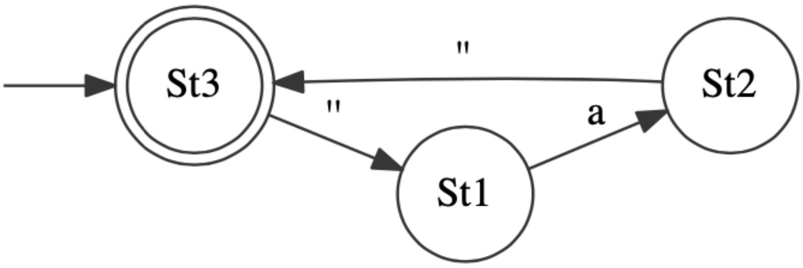
- $N(\varepsilon)$ define-se como o autómato:



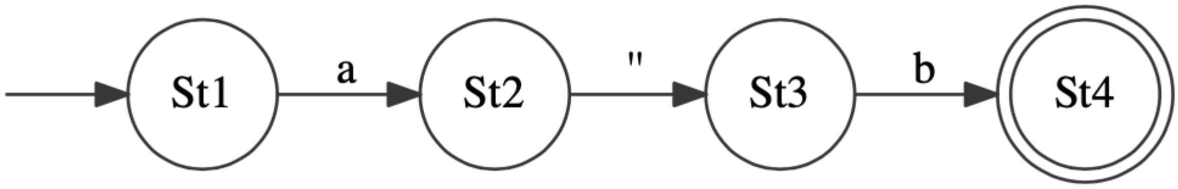
- $N(a)$, com $a \in \Sigma$, define-se como



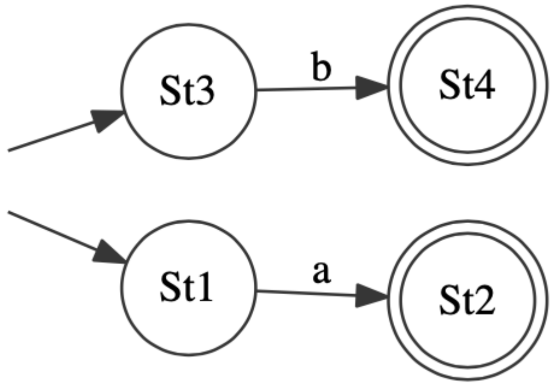
- $N((a))$ define-se como $N(a)$
- $N(a^*)$ define-se como o autómato seguinte, em que $\mathrm{St1}$ e $\mathrm{St2}$ são os estados inicial e final do autómato $N(a)$. Note-se que $N(a)$ pode ter mais do que estado iniciais / finais, e nesse caso deverão ser incluídas mais transições ε (").



- $N(ab)$ define-se como o autómato seguinte, em que $\mathrm{St1}$ e $\mathrm{St2}$ são os estados inicial e final do autómato $N(a)$, e $\mathrm{St3}$ e $\mathrm{St4}$ são os estados inicial e final do autómato $N(b)$. Note-se que
  - $N(a)$ pode ter mais do que estado final, e $N(b)$ pode ter mais do que um estado inicial. Deverão ser incluídas mais transições ε ("), de todos os estados finais de $N(a)$ para todos os estados iniciais de $N(b)$.
  - $\mathrm{St2}$ não é estado final de $N(ab)$, apesar de ser final em $N(a)$!

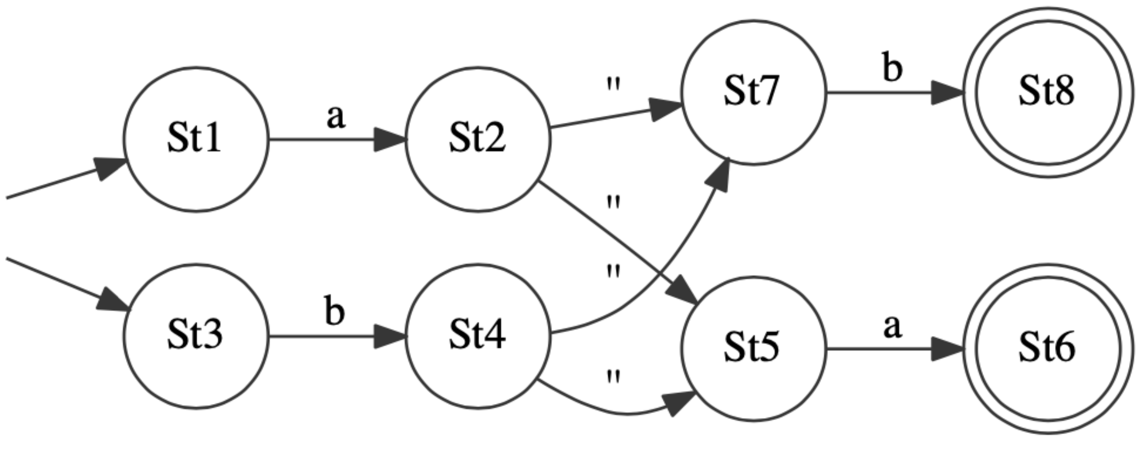


- $N(a+b)$ define-se como o autómato seguinte, em que $\mathrm{St1}$ e $\mathrm{St2}$ são os estados inicial e final do autómato $N(a)$, e $\mathrm{St3}$ e $\mathrm{St4}$ são os estados inicial e final do autómato $N(b)$.

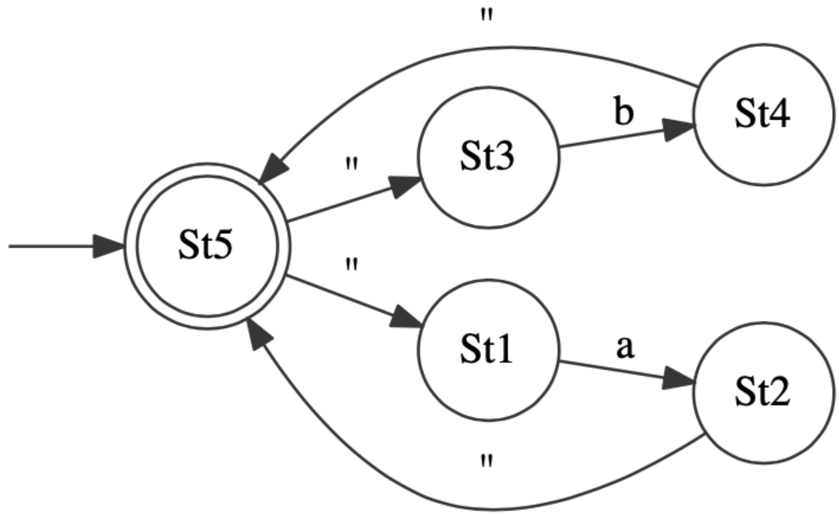


Exemplos

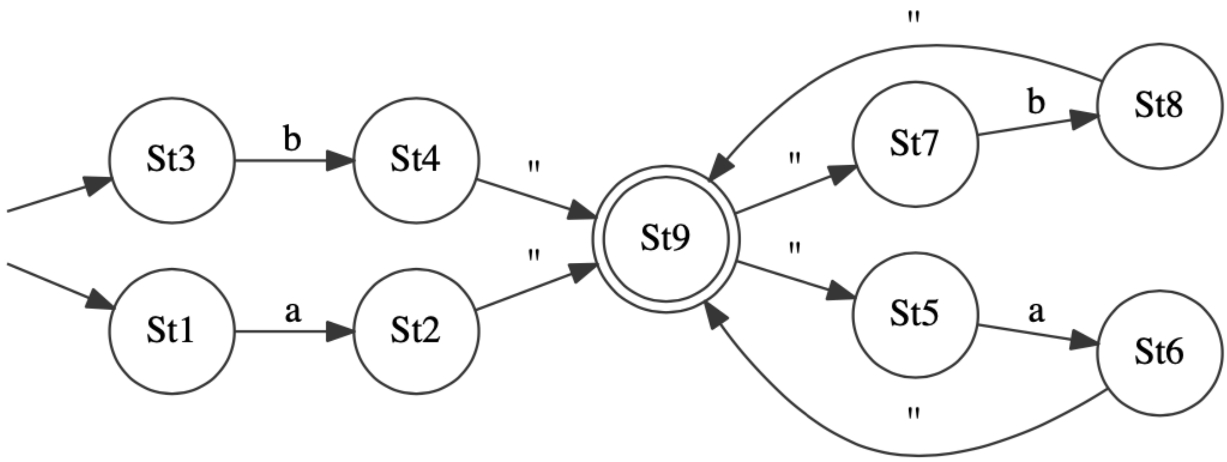
1. Autómato correspondente à expressão $(a+b)(a+b)$



2. Autómato correspondente a $(a+b)^*$



3. Autómato correspondente a $(a+b)(a+b)^*$



Exercício
Construa autómatos NFA para algumas das expressões regulares dos exemplos e exercícios anteriores.

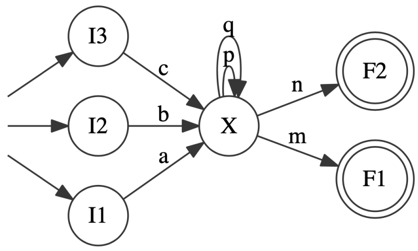
Conversão de NFAs em ERs

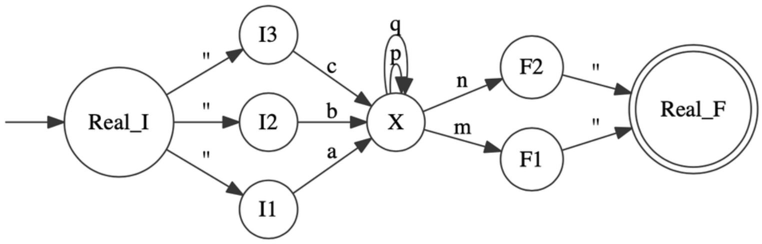
O processo inverso passa pela introdução de uma noção generalizada de autómato não determinístico. Um Autómato Finito Não-determinístico Generalizado (GNFA) é em tudo idêntico a um NF, sendo no entanto os símbolos associados às transições de estados *substituídos por expressões regulares.*

O processo de conversão consiste no seguinte:

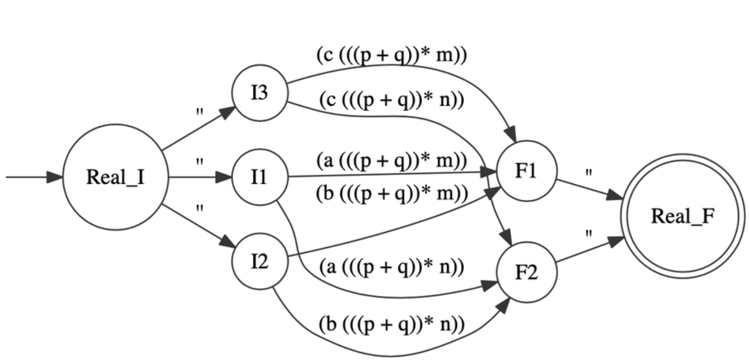
- introdução de um único estado inicial $\mathrm{Real\_I}$ e um único estado final $\mathrm{Real\_F}$; os estados iniciais e finais do autómato inicial são ligadas a estes por transições ε, deixando de ser iniciais / finais no GNFA
- Repete-se depois iterativamente um passo de remoção de um qualquer estado, até restarem apenas os estados $\mathrm{Real\_I}$ e $\mathrm{Real\_F}$
    - A remoção de um estado A obriga à consideração de todos os “caminhos” existentes no autómato passando por esse estado, que deverão ser introduzidos como novas transições no autómato sem A.
    - Se o estado A tiver m entradas e n saídas haverá em em geral mn caminhos que poderão dar origem a novas transições
    - Uma entrada etiquetada com a expressão regular P, um anel etiquetado com Q, e uma saída com etiqueta R corresponde a um caminho que deverá ser introduzido no autómato com etiqueta $PQ^{*}R$
    - Se existirem dois caminhos, com etiquetas x e y, deverão ser representados no autómato por uma única transição com etiqueta $x + y$

Considerando como exemplo este grafo com 3 estados iniciais e dois estados finais, começamos por introduzir os estados $\mathrm{Real\_I}$ e $\mathrm{Real\_F}$



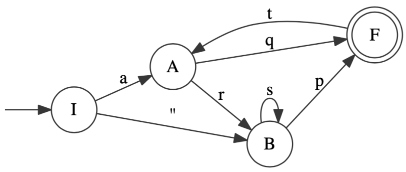


Neste autómato existem 3.2 = 6 caminhos que passam pelo estado X. A eliminação deste estado resulta no seguinte:

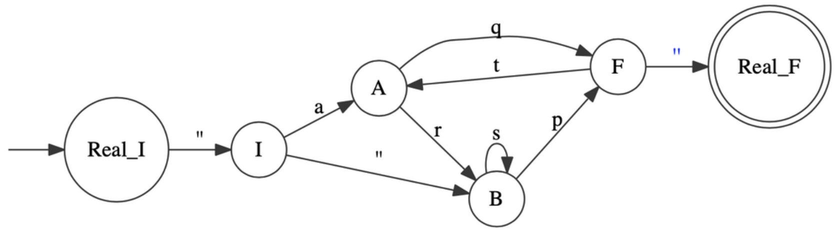


Exemplo Completo

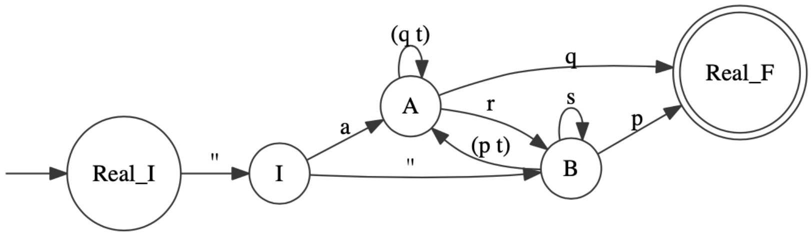
1. NFA inicial



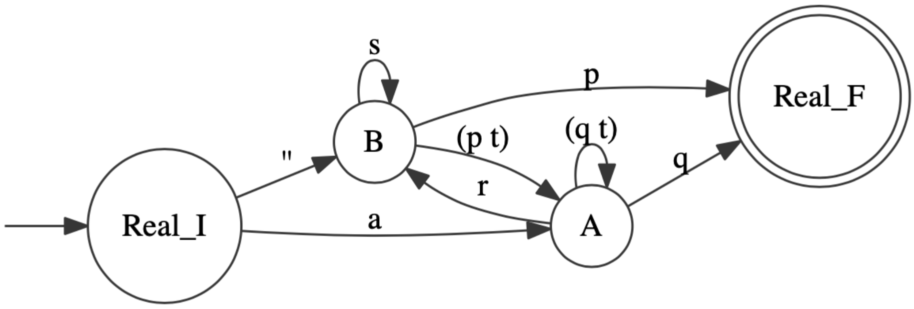
2. GNFA inicial



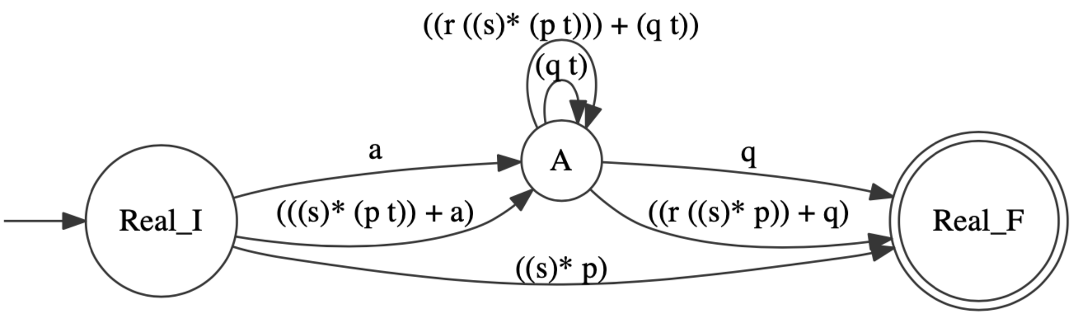
3. Eliminação de F. Há lugar a introdução de “transições” correspondentes a caminhos de A para B e de B para A passando por F, e também a um anel em A



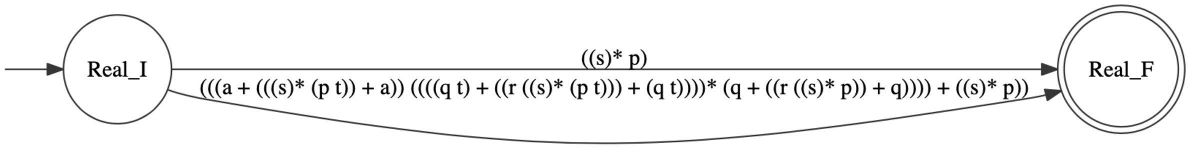
4. Eliminação de I



5. Eliminação de B



6. Eliminação de A



7. Leitura da ER final:
$$(((s)^*p) + (((a + (((s)^*(pt)) + a))((((qt) + ((r((s)^*(pt))) + (qt))))^*(q + ((r((s)^*p)) + q)) + ((s)^*p)))))$$

Conclusão

Apesar de não apresentarmos uma prova formal, é suficientemente claro que estes dois processos constroem um(a) NFA / ER que reconhecem a mesma linguagem que a(o) ER / NFA de que se partiu.

Sendo assim, prova-se que de facto as Expressões Regulares e os Autómatos Finitos (DFA ou NFA) têm o mesmo poder expressivo: ambos reconhecem as chamadas Linguagens Regulares.