Expressões Regulares e Autómatos

José Carlos Ramalho
2022

Sintaxe	Descrição
.	Qualquer carácter excepto o '\n'
*	0 ou mais ocorrências da expressão anterior
+	1 ou mais ocorrências da expressão anterior
?	0 ou 1 ocorrência da expressão anterior
[...]	1 carácter qualquer definido no conjunto
[^]	1 carácter não especificado no conjunto
^	Início de linha
$	Fim de linha
{m,n}	Número de ocorrências mínimo e máximo

Sintaxe	Descrição
\w	Carácter alfanumérico: [a-zA-Z0-9_]
\d	Dígito: [0-9]
\s	Carácter branco: [\t\n\r\f\v]
\b	Fronteira de palavra
\W, \D, and \S	Não palavra, não dígito e não branco
( )	Grupo de captura
(?:)	Grupo não capturado
\1, \2, ..., \9	Referência a grupo previamente capturado
\	Retira o significado ao carácter seguinte

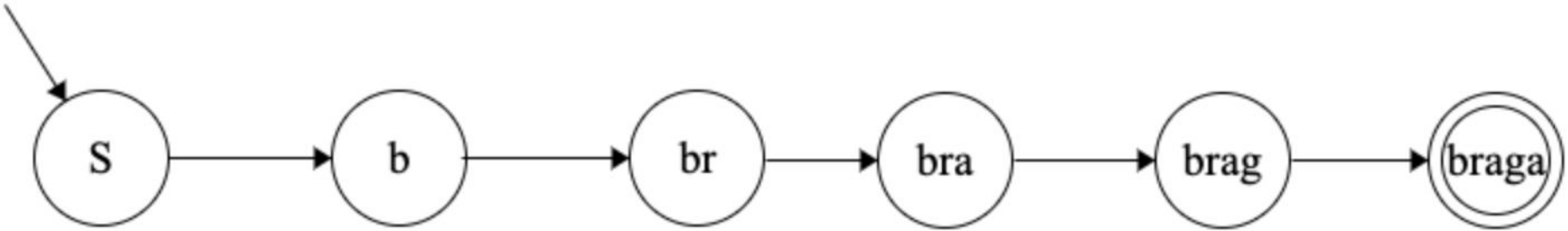
Contexto

`^foo(bar).*` - fácil de perceber?

`^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$` - e esta?

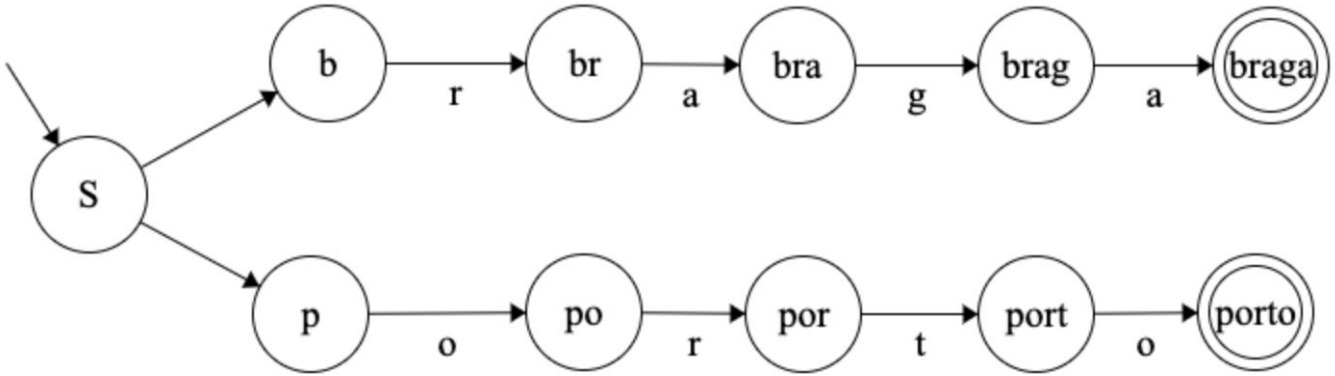
Reconhecer uma sequência

```
e = r'braga'
```



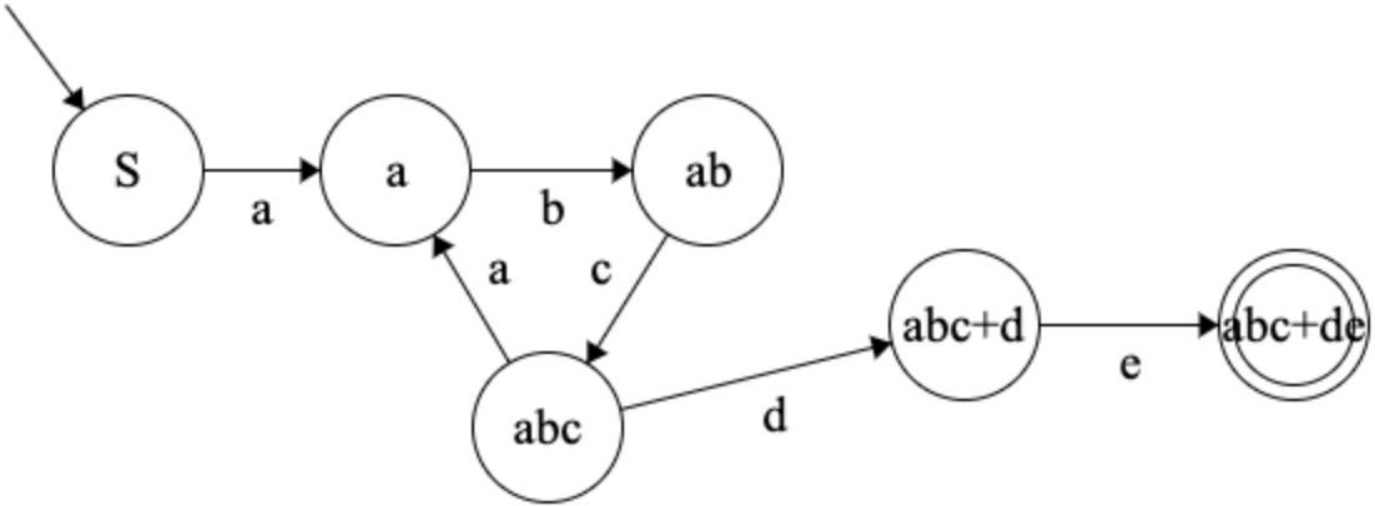
Reconhecer alternativas

```
e = r'braga|porto'
```



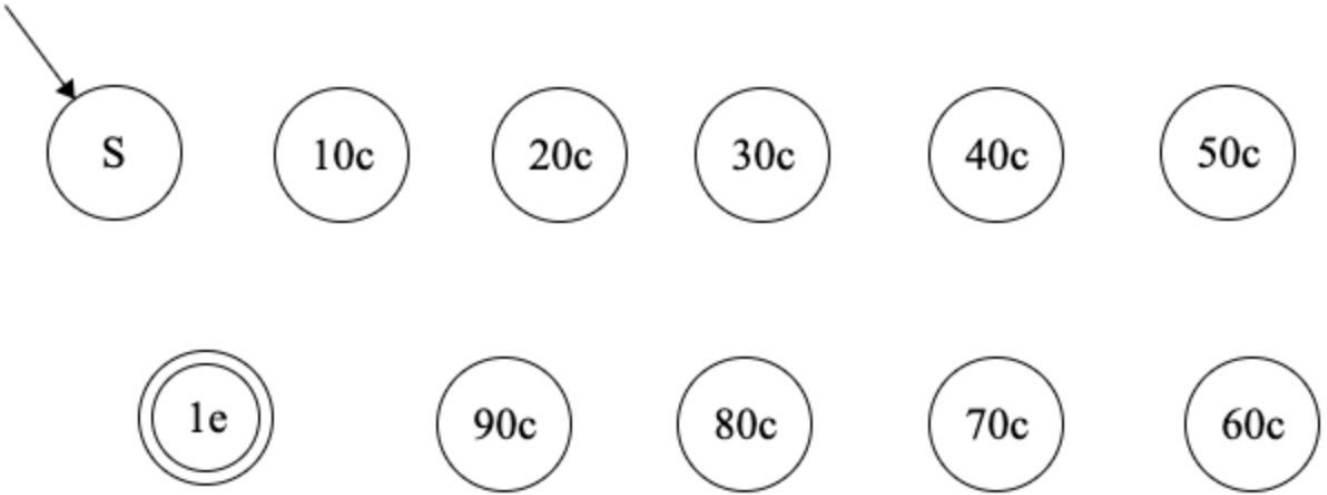
Iteradores

```
e = r'(abc)+de'
```



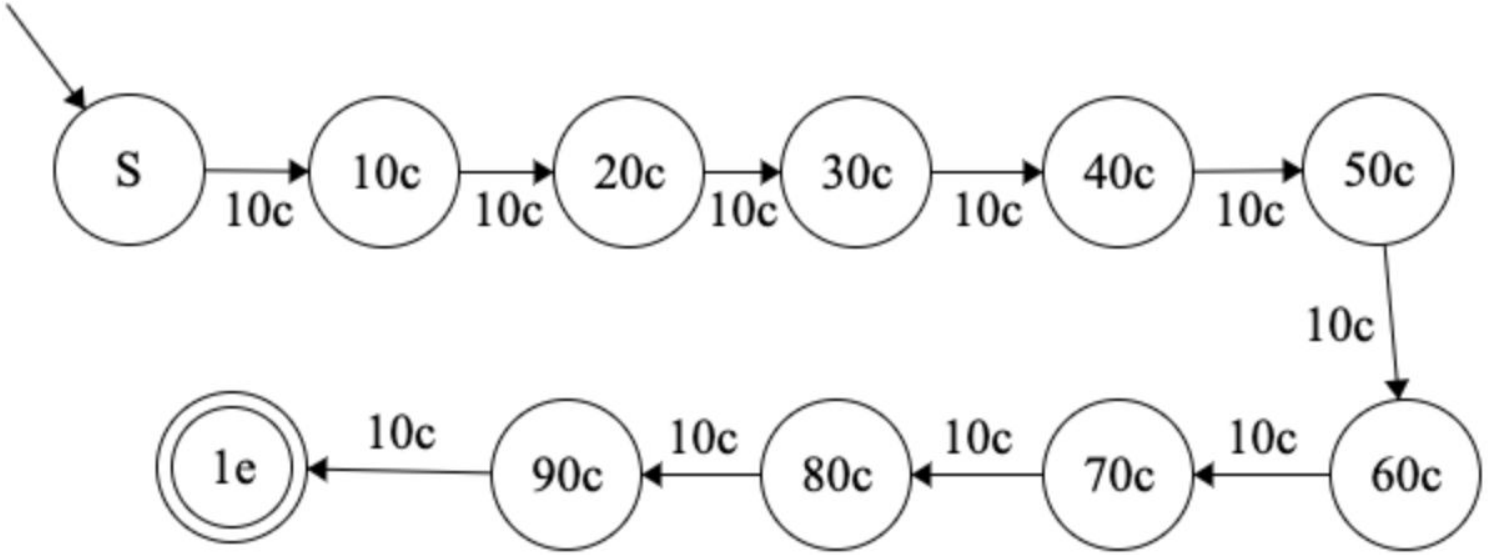
Exercício: Máquina de estados para reconhecer 1 euro em moedas (aceita: 1e, 50c, 20c, 10c)

O que representam os estados do autómato?



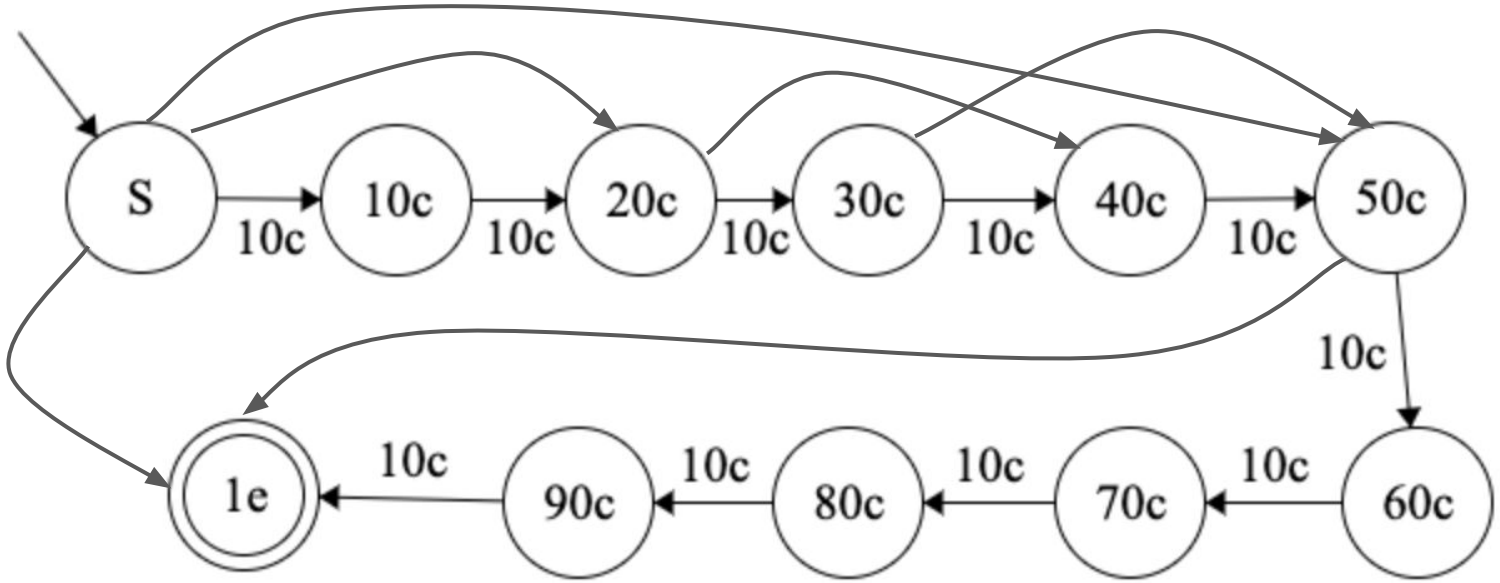
Exercício: Máquina de estados para reconhecer 1 euro em moedas (aceita: 1e, 50c, 20c, 10c)

Transições: 10c



Exercício: Máquina de estados para reconhecer 1 euro em moedas (aceita: 1e, 50c, 20c, 10c)

Transições: as restantes...



Desafio: strings binárias com pelo menos 3 0s

1. **Tenta escrever a expressão regular: ...**
2. **Tenta desenhar o autómato: o que representa cada estado? ...**
3. **Reescreve a expressão regular a partir do autómato.**

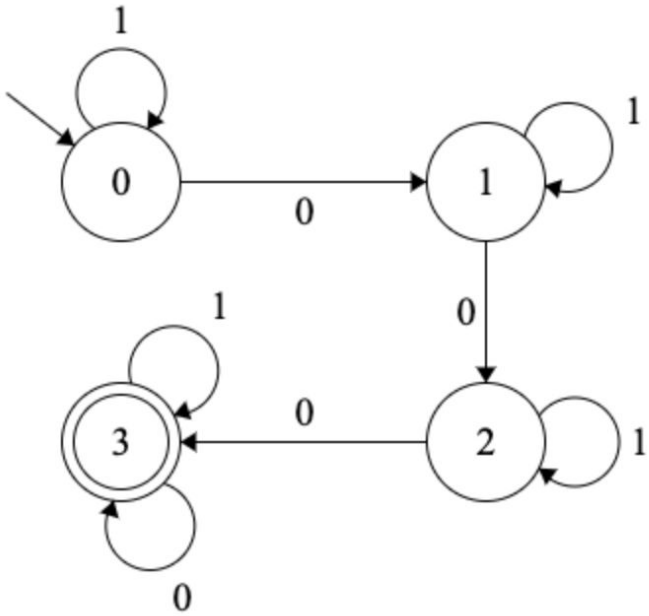
Desafio: strings binárias com pelo menos 3 0s

1. **Tenta escrever a expressão regular: ...**

```
[01]*0[01]*0[01]*0[01]*

([01]*0[01]*){3}

1*01*01*0[01]*

1*(01*){3,}
```

Desafio: strings binárias com pelo menos 3 0s

1. **Tenta escrever a expressão regular: ...**
2. **Tenta desenhar o autómato: o que representa cada estado? ...**



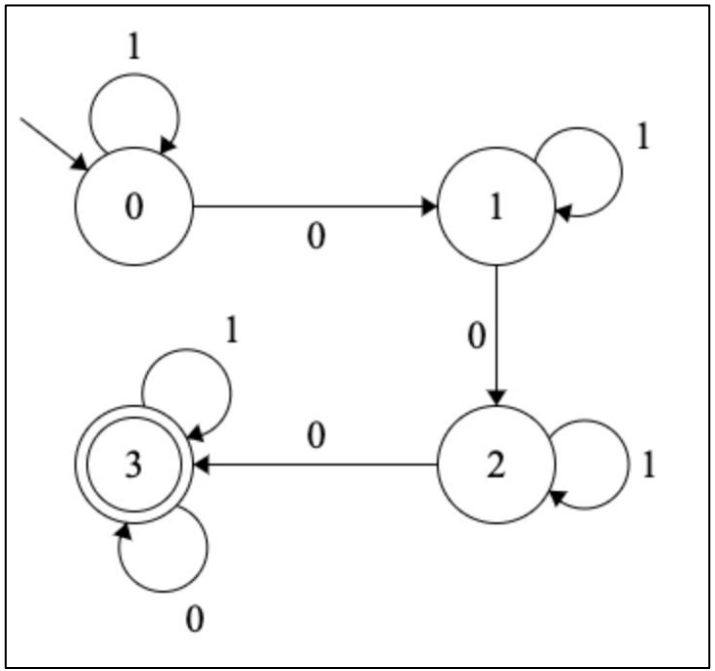
Desafio: strings binárias com pelo menos 3 0s

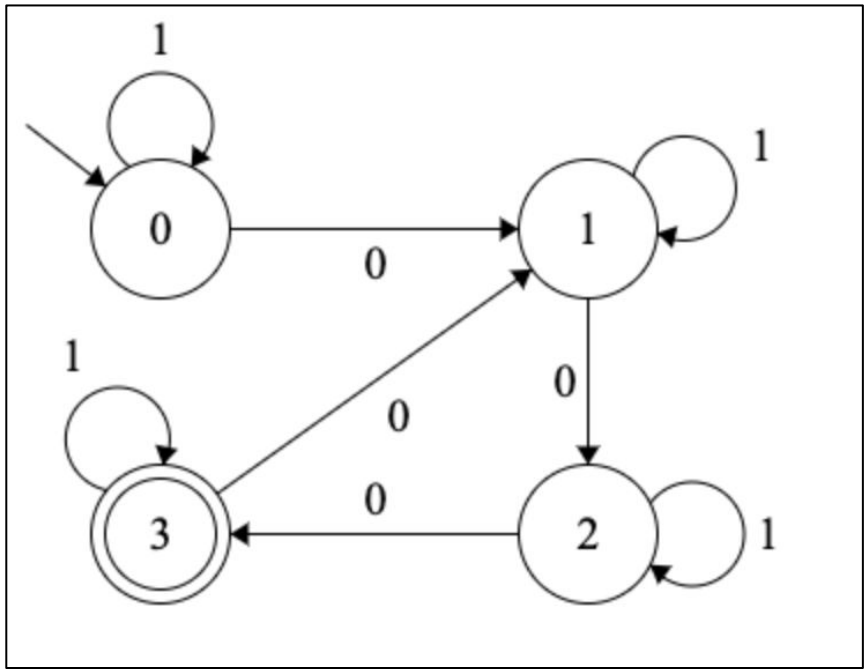
1. **Tenta escrever a expressão regular: ...**
2. **Tenta desenhar o autómato: o que representa cada estado? ...**
3. **Reescreve a expressão regular a partir do autómato.**

```
1*01*01*0[01]*

1*(01*01*01*)+
```

Desafio++: o número de 0s é múltiplo de 3

Que alteração fazemos ao autómato?

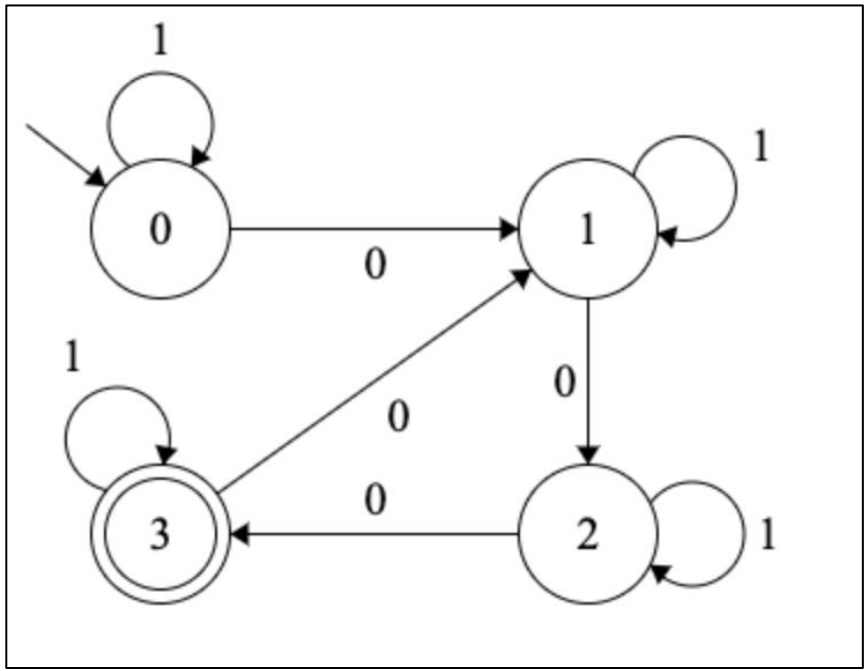




Desafio++: o número de 0s é múltiplo de 3

Como fica a expressão?

```
1*(01*01*01*)+
```

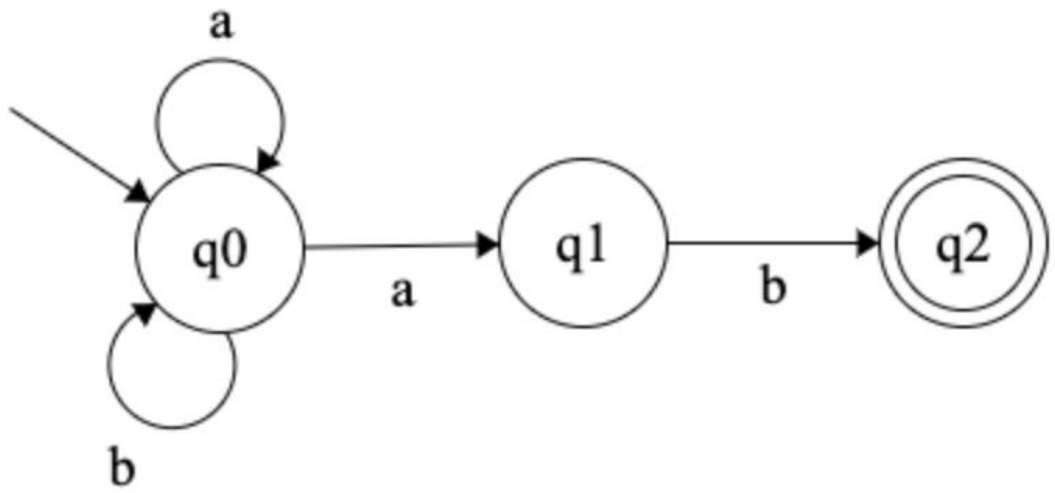


Desafio: para o teste...

Strings binárias com número par de 0s e ímpar de 1s

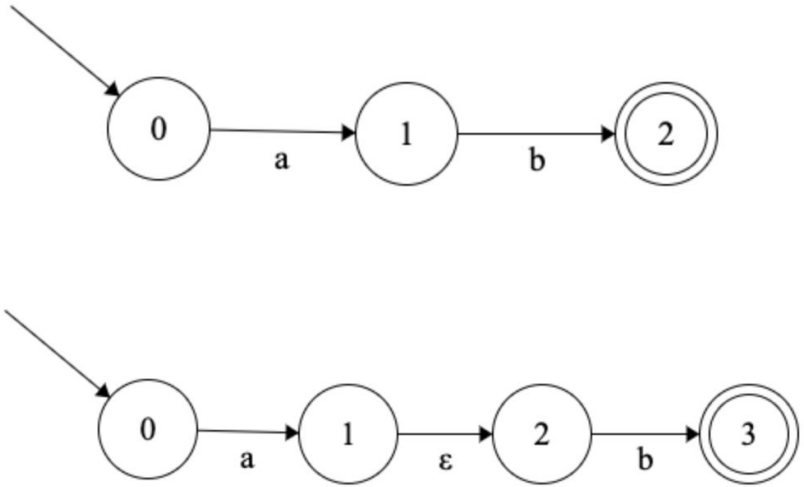
1. **Tenta escrever a expressão regular: ...**
2. **Tenta desenhar o autómato: o que representa cada estado?**
   ...
3. **Reescreve a expressão regular a partir do autómato.**

Autómato Finito Não Determinista



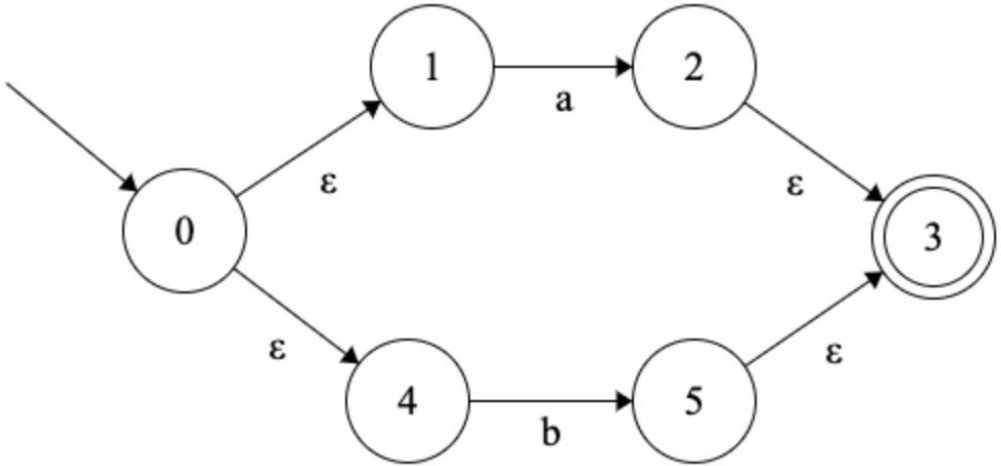
De ExpReg a AFND:

ab



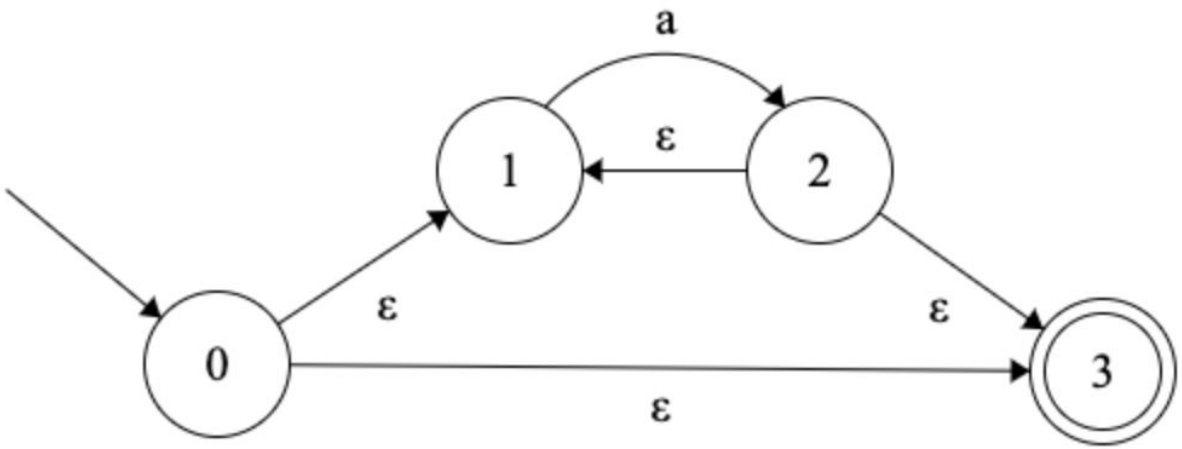
De ExpReg a AFND:

`a|b`



De ExpReg a AFND:

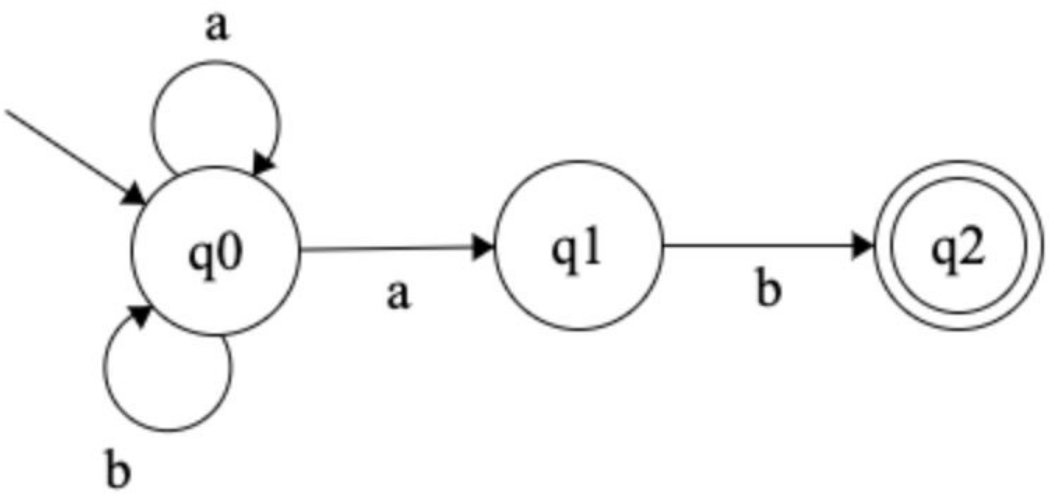
a*



Exercício: cria o AFND para ExpReg

```
(a|b)*a
```

Exercício: cria o AFD para o AFND



Exercício: abc

Strings formadas por V={a,b,c} em qualquer ordem e número mas em que os símbolos estão na sua ordem alfabética:

- bccccccc
- aaac
- aaabbbbbbbbbc
- abc
- c
- ac
- b
- a

Exercício: abc

Strings formadas por V={a,b,c} em qualquer ordem e número mas em que os símbolos estão na sua ordem alfabética:

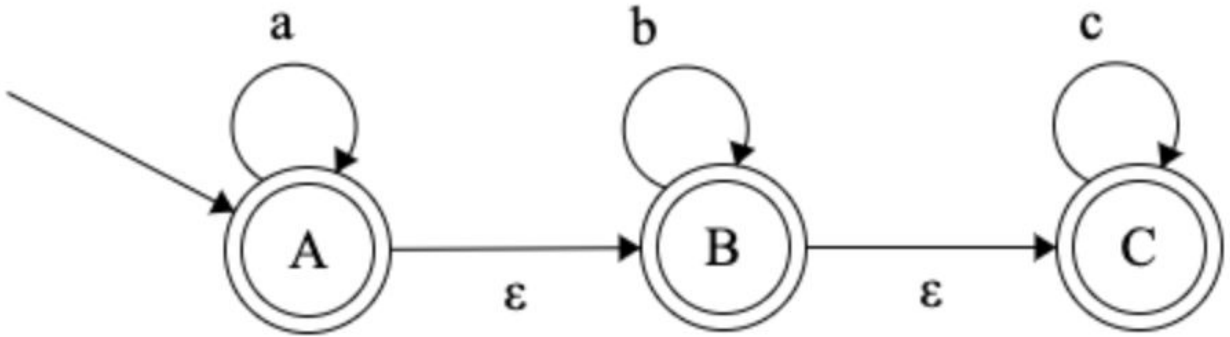
- bccccccc
- aaac
- aaabbbbbbbbbc
- abc
- c
- ac
- b
- a

```
a*b*c*
```

Exercício: abc - da expreg ao afnd

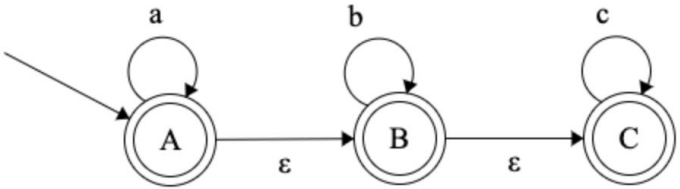
Strings formadas por V={a,b,c} em qualquer ordem e número mas em que os símbolos estão na sua ordem alfabética.

```
a*b*c*
```



Exercício: abc - do afnd ao afd

Strings formadas por V={a,b,c} em qualquer ordem e número mas em que os símbolos estão na sua ordem alfabética.



	a	**b**	**c**
>*ABC	ABC	BC	C
*BC	---	BC	C
*C	---	---	C

Exercício: abc - afd

Strings formadas por V={a,b,c} em qualquer ordem e número mas em que os símbolos estão na sua ordem alfabética.

	a	**b**	**c**
ABC	ABC	BC	C
BC	---	BC	C
C	---	---	C

Desafio final: constrói o Autómato para a linguagem L que tem Vocabulário = {0,1,2} e cujas frases corretas terminam num símbolo que tenha ocorrido antes

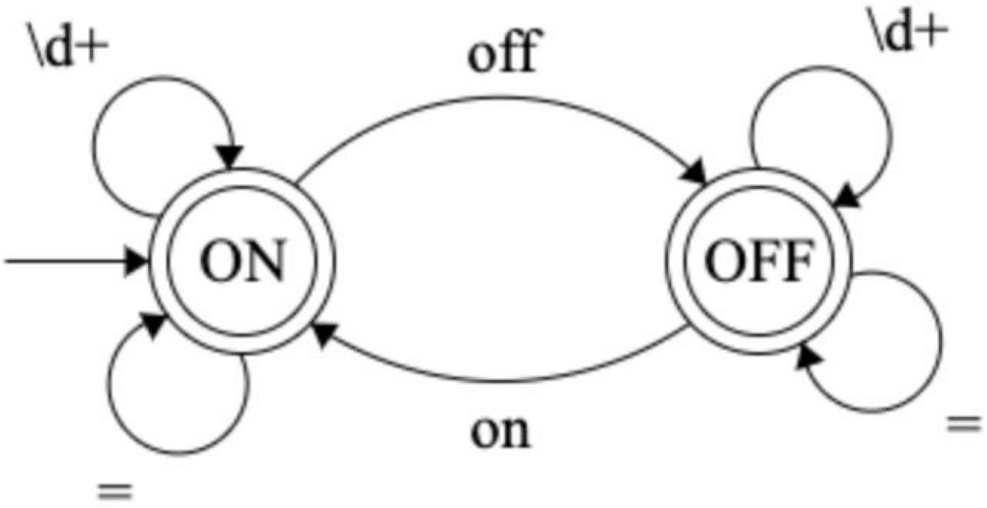
```
Input : 01101
Output : Accepted

Input : 012
Output : Not Accepted

Input : 2
Output : Not Accepted

Input : 0122
Output : Accepted
```

Somador on/off

1. Pretende-se um programa que some todas as sequências de dígitos que encontre num texto;
2. Sempre que encontrar a string “Off” em qualquer combinação de maiúsculas e minúsculas, esse comportamento é desligado;
3. Sempre que encontrar a string “On” em qualquer combinação de maiúsculas e minúsculas, esse comportamento é novamente ligado;
4. Sempre que encontrar o caráter “=”, o resultado da soma é colocado na saída.

Somador on/off: AFD

Como desambiguar as expressões regulares comuns?



Condições de contexto / Start Conditions

```
states = (
   ('on','exclusive'),
   ('off','inclusive'),
 )
```

```
t_on_NUMBER = r'\d+'

t_off_NUMBER = r'\d+

def t_off_newline(t):
    r'\n'
    t.lexer.lineno += 1
```

Condições de contexto / Start Conditions

```
states = (
   ('on','exclusive'),
   ('off','inclusive'),
 )
```

```
t_on_off_NUMBER = r'\d+'

# equivalente a:

# t_ANY_NUMBER = r'\d+'
```

Condições de contexto / Start Conditions

```
states = (
   ('on','exclusive'),
   ('off','inclusive'),
 )
```

```
# equivalentes:

t_NUMBER = r'\d+'

t_INITIAL_NUMBER = r'\d+'
```

Condições de contexto / Start Conditions

```
states = (
   ('on','exclusive'),
   ('off','inclusive'),
 )
```

```
# skip e erro:
t_on_ignore = r'\t '

def t_off_error(t):
    pass
```

Condições de contexto / Start Conditions

```
states = (
   ('on','exclusive'),
   ('off','inclusive'),
 )
```

```
# estado inicial e mudança de estado:

def t_begin_on(t):
    r'[Oo][Nn]'
    t.lexer.begin(on)
```

Somador on/off: criar o programa em Python

1. Pretende-se um programa que some todas as sequências de dígitos que encontre num texto;
2. Sempre que encontrar a string “Off” em qualquer combinação de maiúsculas e minúsculas, esse comportamento é desligado;
3. Sempre que encontrar a string “On” em qualquer combinação de maiúsculas e minúsculas, esse comportamento é novamente ligado;
4. Sempre que encontrar o caráter “=”, o resultado da soma é colocado na saída.