Teoria dos Autômatos e Linguagens Formais

aula 16: Linguagens não regulares

1 Introdução

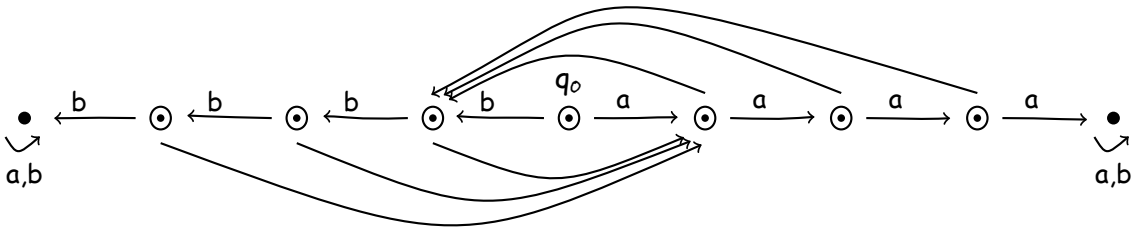
Nas últimas aula, nós vimos que os autômatos podem fazer muitas coisas.

Mas, os autômatos não podem fazer qualquer coisa.

Por exemplo, os autômatos não conseguem contar.

Quer dizer, os autômatos conseguem contar até um número fixo.

O autômato abaixo aceita as palavras que possuem no máximo 3 símbolos consecutivos iguais



E não é difícil aumentar esse número para 4, 5 ou mais.

Mas, os autômatos não conseguem fazer uma contagem sem limite.

Porque?

Bom, porque mais cedo ou mais tarde eles perdem a conta.

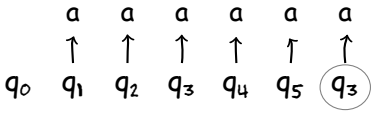
Porque?

Bom, porque contar significa ficar em um estado diferente a cada número que a gente conta.

É isso o que está acontecendo na sua cabeça quando você conta: *um, dois, três, ...* — (*é preciso lembrar o número que a gente acabou de falar para poder falar o próximo*)

Mas, os autômatos possuem um número finito de estados.

Daí que, mais cedo ou mais tarde, ele precisa voltar para um estado onde ele já passou antes



É nessa hora que o autômato perde a conta.

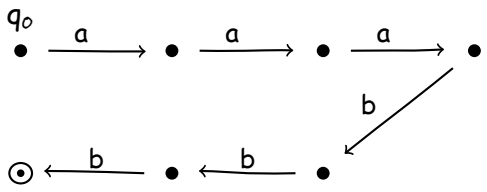
2 Enganando o autômato

E na hora em que o autômato perde a conta, não é difícil enganar ele.

Por exemplo, imagine que o nosso autômato aceita a linguagem

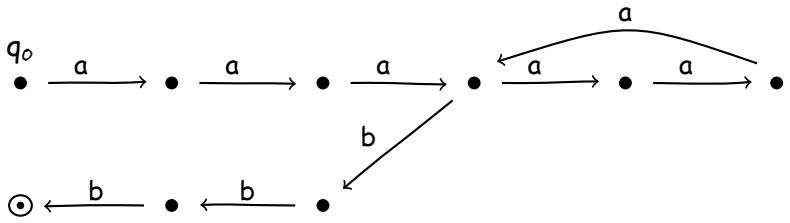
$a^n b^n$: palavras formadas por um bloco de a's
seguido por um bloco de b's do mesmo tamanho

Então, ao receber a palavra aaabbb, o autômato vai fazer um caminho assim



— (*e aceitar a palavra de entrada*)

Mas, esse autômato perde a conta depois de 5 a's, e volta para o estado q_3



Então, o autômato acredita que ele reconhece a linguagem $a^n b^n$.

Mas ele responde SIM para a palavra aaaaaabbb.

3 Uma linguagem não regular

Claro, o nosso autômato tem um defeito.

Mas o ponto é que todos os autômatos tem esse defeito.

Quer dizer, não existe nenhum autômato finito que reconhece a linguagem $a^n b^n$.

Porque?

Bom, intuitivamente a gente já sabe porque — (*i.e., os autômatos não conseguem contar*)

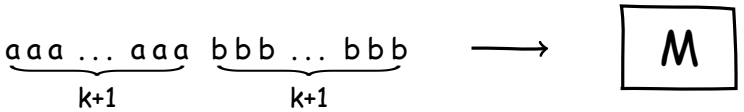
Mas agora, a gente vai fazer um raciocínio mais preciso.

Imagine que M é um autômato qualquer — (*i.e., um autômato que a gente não sabe qual é ...*)

E imagine que o autômato M reconhece a linguagem $a^n b^n$ — (*para a gente chegar em uma contradição*)

Como M é um autômato finito, ele possui uma quantidade finita k de estados.

Agora, imagine que a gente dá a seguinte palavra para o autômato M

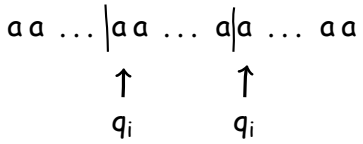


Bom, nesse caso acontecem duas coisas

i. o autômato vai responder SIM — (*porque a palavra pertence à linguagem* $\mathtt{a}^n\,\mathtt{b}^n$)

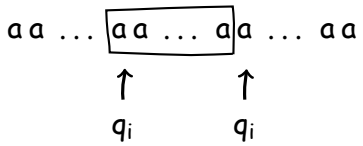
ii. o autômato vai passar duas vezes pelo mesmo estado, durante a leitura do bloco de a's — (*porque?*)

A seguir, a gente olha para a coisa em mais detalhe

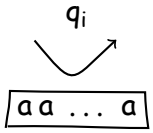


Quer dizer, aqui a gente colocou em foco dois momentos em que o autômato estava no mesmo estado — (*durante a leitura do bloco de* a*'s*)

Daí, a gente modifica o esquema assim

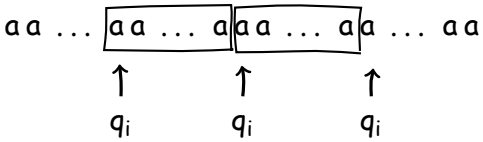


para ver que existe um bloco de a's de um certo tamanho que leva o autômato de $\mathtt{q}_i$ de volta para $\mathtt{q}_i$ — (*um loop ...*)



E agora, a gente já pode enganar o autômato M.

Quer dizer, basta duplicar esse bloco de a's — (*mantendo o restante da palavra original*)



O ponto é que no fim do segundo bloco de a's (em destaque) o autômato "já esqueceu" que ele viu o primeiro bloco — (*porque ele está no estado* $\mathtt{q}_i$ *outra vez ...*)

Daí, o restante da computação acontece de maneira igual — (*porque?*)

E o autômato M responde SIM no final.

Só que, a palavra não pertence à linguagem $\mathtt{a}^n\,\mathtt{b}^n$.

Logo, nós temos uma contradição.

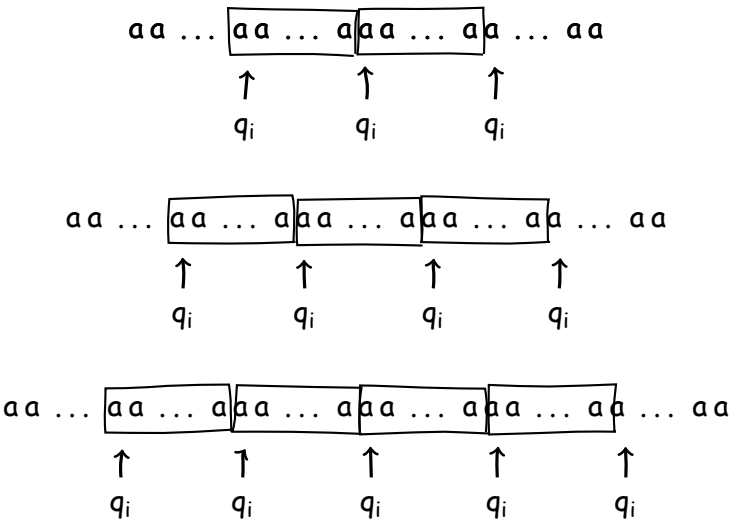
Logo, não existe autômato M que reconhece a linguagem $\mathtt{a}^n\,\mathtt{b}^n$.

Logo, a linguagem $\mathtt{a}^n\,\mathtt{b}^n$ não é regular.

4 O argumento do bombeamento

O raciocínio que nós acabamos de ver é chamado de *argumento do bombeamento*.

Porque nós podemos "bombear" quantos blocos de a's nós quisermos, para enganar o autômato



E nós podemos também remover o bloco de a's para enganar o autômato — (*porque?*)

Isso é uma ideia bem simples, que nos permite mostrar que um monte de linguagens não são regulares.

Por exemplo, considere a linguagem

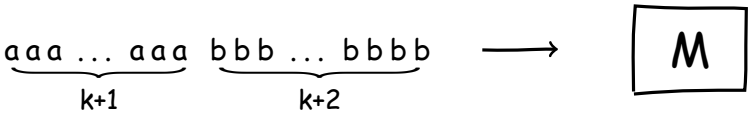
$$\text{SUC:}\quad a^n b^{n+1}$$

— (*a linguagem do sucessor ...*)

Imagine que o autômato M reconhece essa linguagem.

E imagine que M tem k estados.

Então, a gente pode aplicar o mesmo truque outra vez

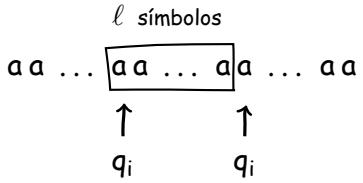


Aqui, a gente sabe duas coisas

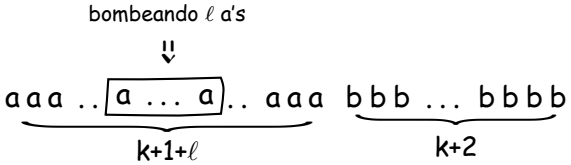
i. o autômato vai responder SIM — (*porque?*)

ii. o autômato vai passar duas vezes pelo mesmo estado, durante a leitura do bloco de a's — (*um loop*)

Colocando o loop em foco, nós vemos o seguinte



Isso significa que o autômato também aceita a palavra



Só que, a palavra $\mathtt{a}^{k+1+\ell}\,\mathtt{b}^{k+2}$ não está na linguagem SUC.

Logo, nós temos uma contradição.

Logo, a linguagem SUC não é regular.

A seguir, nós vamos ver mais exemplos.

a. A linguagem DOB

Considere a linguagem

DOB: palavras onde o número de a's é o dobro do número de b's

Por exemplo,

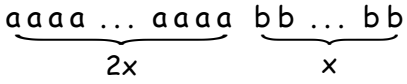
a a b, a b a, b a a, a a b b a a, a a a b a a a b b, etc

Dessa vez, as palavras da linguagem não possuem um padrão bem definido.

Mas, isso não importa.

Porque nós só precisamos de um contra-exemplo — (*porque o autômato precisa dar a resposta correta para todas as palavras de entrada, para reconhecer a linguagem*)

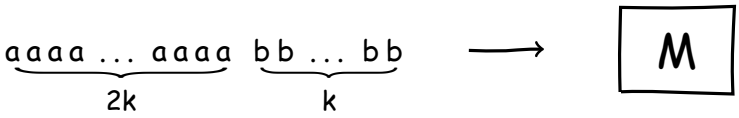
Então, nós vamos concentrar a atenção em palavras da forma



Agora, imagine que o autômato M reconhece a linguagem DOB.

E imagine que M tem k estados.

Daí, a gente aplica o nosso truque

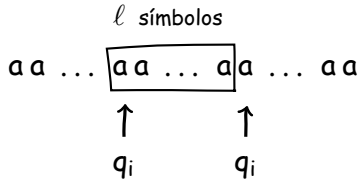


Aqui, nós sabemos que

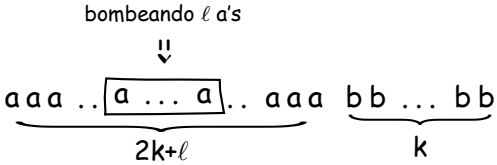
i. o autômato vai responder SIM — (*porque?*)

ii. o autômato vai passar duas vezes pelo mesmo estado, durante a leitura do bloco de a's — (*porque* $2k$ *é maior do que* k)

Colocando o loop em foco, a gente vê o seguinte



Isso significa que o autômato também aceita a palavra



Só que, a palavra $\mathtt{a}^{2k+\ell}\,\mathtt{b}^{k}$ não está na linguagem DOB.

Logo, nós temos uma contradição.

Logo, a linguagem DOB não é regular.

◇

b. A linguagem PAL

Todo mundo sabe que um *palíndromo* é uma palavra (ou frase) que é a mesma coisa quando lida nas duas direções

ana, osso, reviver, a sacada da casa, o galo ama o lago, etcte

No mundo dos a's e b's, os palíndromos são assim

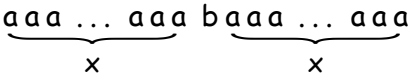
a, bab, aabaa, baabbaab, etcte

Essa é a linguagem PAL.

E a seguir, nós vamos mostrar que ela não é regular.

Mais uma vez, a linguagem não tem um padrão bem definido.

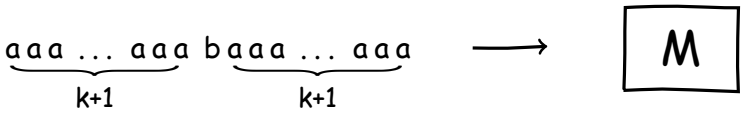
E a esperteza é concentrar a atenção nas palavras da forma



Agora, imagine que o autômato M reconhece a linguagem PAL.

E imagine que M tem k estados.

Daí, a gente aplica o nosso truque

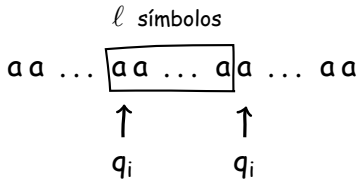


Aqui, nós sabemos duas coisas

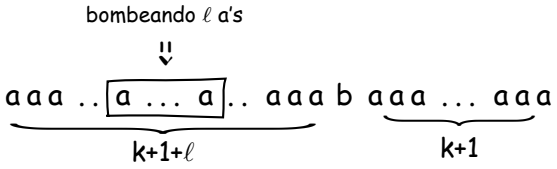
i. o autômato vai responder SIM — (*porque?*)

ii. o autômato vai passar duas vezes pelo mesmo estado, durante a leitura do bloco de a's — (*porque?*)

Colocando o loop em foco, a gente vê o seguinte



Isso significa que o autômato também aceita a palavra



Só que, a palavra $\mathtt{a}^{k+1+\ell}\,\mathtt{b}\,\mathtt{a}^{k+1}$ não está na linguagem PAL.

Logo, nós temos uma contradição.

Logo, a linguagem PAL não é regular.

◇

c. A linguagem DUP

Imagine que a gente pega uma palavra qualquer e concatena outra cópia dela no final

abaa => abaaabaa

As palavras construídas dessa maneira formam a linguagem DUP.

DUP não é uma linguagem regular.

Porque?

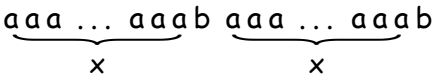
Ora, porque para reconhecer as palavras de DUP é preciso lembrar a 1a metade durante a leitura da 2a metade.

Só que os autômatos não conseguem lembrar nada (muito grande) — (*eles não conseguem nem contar ...*)

Para provar isso, a gente usa o argumento do bombeamento outra vez.

Aqui mais uma vez, não existe um padrão bem definido para as palavras da linguagem.

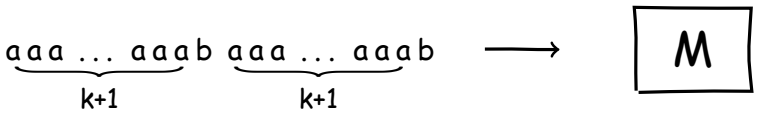
Então, a gente concentra a atenção nas palavras da forma



Agora, imagine que o autômato M reconhece a linguagem DOB.

E imagine que M tem k estados.

Daí, a gente aplica o nosso truque

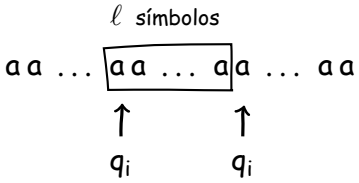


Aqui, nós sabemos que

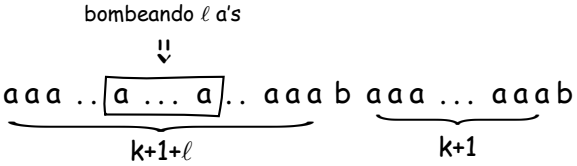
i. o autômato vai responder SIM — (*porque?*)

ii. o autômato vai passar duas vezes pelo mesmo estado, durante a leitura do bloco de a's — (*porque?*)

Colocando o loop em foco, a gente vê o seguinte



Isso significa que o autômato também aceita as palavras



Só que, a palavra $\mathtt{a}^{k+1+\ell}\,\mathtt{b}\,\mathtt{a}^{k+1}\,\mathtt{b}$ não está na linguagem DUP.

Logo, nós temos uma contradição.

Logo, a linguagem DUP não é regular.

◇

d. A linguagem dos primos

Uma máquina que não sabe contar não pode saber nada de matemática.

Então agora, a gente considera a linguagem

Pr: blocos de a's cujo tamanho é um número primo

Por exemplo,

aa, aaa, aaaaa, aaaaaaa, etc

A seguir, nós vamos argumentar que Pr não é uma linguagem regular.

Imagine que o autômato M reconhece a linguagem Pr.

E imagine que M tem k estados.

Daí, a gente considera um primo p qualquer, maior do que o número k.

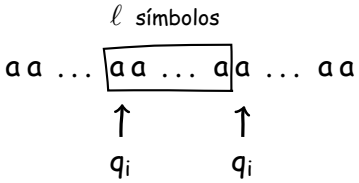
E aplica o nosso truque

Aqui, nós sabemos que

i. o autômato vai responder SIM — (*porque?*)

ii. o autômato vai passar duas vezes pelo mesmo estado, durante a leitura do bloco de a's — (*porque p é maior do que k*)

Colocando o loop em foco, a gente vê o seguinte



Só que dessa vez, o bombeamento pode não funcionar.

Porque $p + \ell$ também pode ser um número primo.

E agora?

Bom, agora é a hora para uma esperteza.

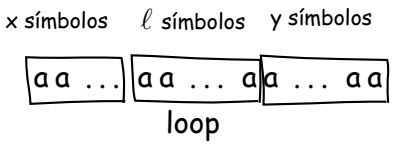
Quer dizer, a esperteza é bombear muitas vezes.

Porque mais cedo ou mais tarde a gente acaba obtendo um número que não é primo.

Quem disse?

Bom, ninguém disse nada — (*isso é só uma intuição ...*)

Então, para entender melhor as coisas a gente quebra a palavra em 3 blocos



E escreve o número p assim

$$p \quad = \quad (x + y) + \ell$$

Daí, a gente começa a bombear

$$a^{p+\ell} \quad = \quad (x + y) + \ell + \ell$$

$$a^{p+2\cdot\ell} \quad = \quad (x + y) + \ell + \ell + \ell$$

$$a^{p+3\cdot\ell} \quad = \quad (x + y) + \ell + \ell + \ell + \ell$$

E continua bombeando até obter o seguinte

$$a^{p+3\cdot\ell} \quad = \quad (\,x\,+\,y\,)\,+\,\underbrace{\ell\,+\,\ell\,+\,\ldots\,+\,\ell}_{\mathsf{x}\,+\,\mathsf{y}}$$

Agora, a gente pode reescrever o número assim

$$(\,x\,+\,y\,)\,+\,\ell\,\cdot\,(\,x\,+\,y\,)$$

e depois assim

$$(\,x\,+\,y\,)\,\cdot\,(\,1\,+\,\ell\,)$$

Sim! isso é um número composto.

Logo, a palavra $\mathsf{a}^{(\mathtt{x}+\mathtt{y})\cdot(\mathtt{1}+\ell)}$ não está na linguagem Pr.

Mas, o autômato responde SIM quando recebe essa palavra como entrada — (*porque?*)

Isso é uma contradição.

Logo, a linguagem PR não é regular.

◇

e. A linguagem dos quadrados

Considere a linguagem

QUAD: blocos de a's cujo tamanho é um quadrado perfeito

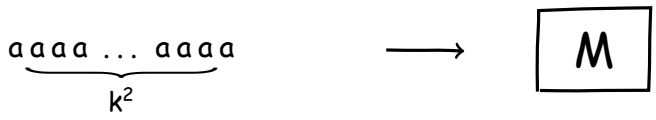
A seguir, nós vamos mostrar que QUAD não é regular.

Imagine que o autômato M reconhece a linguagem QUAD.

E imagine que M tem k estados.

Daí, a gente considera o quadrado perfeito k^2.

E aplica o nosso truque

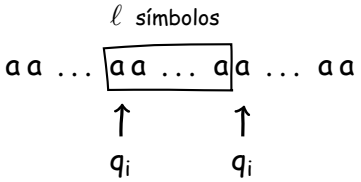


Aqui, a gente já sabe que

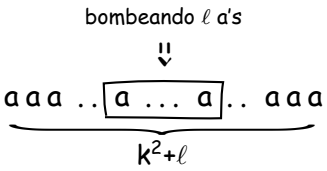
i. o autômato vai responder SIM — (*porque?*)

ii. o autômato vai passar duas vezes pelo mesmo estado, durante a leitura do bloco de a's — (*porque?*)

Colocando o loop em foco, a gente vê o seguinte



Isso significa que o autômato também vai aceitar a palavra



E agora nós observamos que

→ $k^2 + \ell$ *não é um quadrado perfeito*

Porque não?

Bom, a primeira observação é que $\ell \leq k$.

Porque a gente sempre pode considerar um loop onde todos os estados são diferentes — (*e o autômato tem apenas k estados*)

Daí, a gente olha para o próximo quadrado perfeito

$$\big(k+1\big)^2 \quad = \quad k^2 \quad + \quad \underbrace{(2k+1)}_{> \ \ell}$$

Logo, nós temos

$$k^2 \quad < \quad k^2 + \ell \quad < \quad (k+1)^2$$

Logo, $k^2 + ell$ não é um quadrado perfeito.

Mas, o autômato M aceita a palavra $\mathtt{a}^{\mathtt{k}^2+\ell}$.

Isso é uma contradição.

Logo, a linguagem QUAD não é regular.

$\diamond$