Teoria dos Autômatos e Linguagens Formais

aula 01: Os autômatos finitos

1 Introdução

Os autômatos finitos são a máquina computadora mais simples que existe.

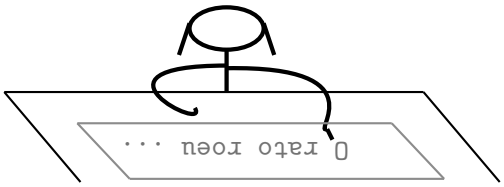
E a máquina computadora mais simples que existe é uma máquina que só realiza leitura.

Agora, o mais legal é que os autômatos fazem leitura do mesmo jeito que a gente faz leitura.

Mas, como é que a gente faz leitura?

Bom, a gente começa colocando o dedo no começo da frase.

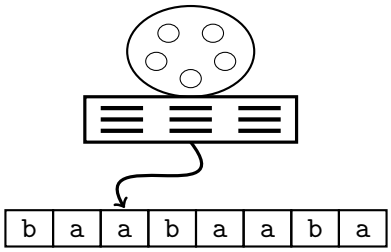
Daí, a gente vai deslizando o dedo e examinando as letrinhas uma por uma



Quando a gente já tem prática, ao invés do dedo a gente usa o olho — (*ou alguma coisa que fica dentro do olho ...*)

Quer dizer, o autômato faz as coisas do mesmo jeito.

Só que ao invés do dedo, o autômato usa uma antena — (*ou uma cabeça de leitura*)



Por enquanto, as palavras que o autômato vai ler serão formadas apenas por `a`'s e `b`'s.

Legal!

Mas, não dá pra fazer nada só fazendo leitura.

Então, o primeiro exemplo que nós vamos ver é um autômato que não faz nada.

Quer dizer, a ideia é que o autômato é controlado por uma *caixinha de regras*.

Daí que, para construir um autômato que não faz nada, basta utilizar regras que não fazem nada — (*isso faz sentido, não é?*)

Uma regra que não faz nada é uma regra que deixa as coisas do jeito que estão.

Então, as regras que a gente vai utilizar são as seguintes

- `Se o símbolo que está sendo lido é um 'a',`
  `então deixe as coisas do jeito que estão`
- `Se o símbolo que está sendo lido é um 'b',`
  `então deixe as coisas do jeito que estão`

Utilizando o formalismo matemático, a gente denota o jeito como as coisas estão por `q`.

E daí, a gente pode escrever as regras assim

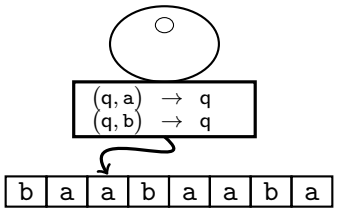
$$(\mathtt{q}, \mathtt{a}) \longrightarrow \mathtt{q} \qquad \text{e} \qquad (\mathtt{q}, \mathtt{b}) \longrightarrow \mathtt{q}$$

onde o significado mais preciso dessas regras é

→ `Quando as coisas estão do jeito que estão e você lê 'a' (ou 'b'),`
  `deixe as coisas do jeito que estão`

Pronto.

Agora nós já temos o nosso primeiro exemplo completo



2 Os autômatos finitos

Certo.

Mas, uma máquina que não faz nada também não serve pra nada.

Quer dizer, se ao final da leitura a gente pergunta: "*E daí, o que você entendeu?*"

A máquina iria responder: "*Eu não entendi nada*"
— (*ou ela não diria nada, o que é basicamente a mesma coisa ...*)

De fato, a gente pode treinar um algum animal dos que usam em laboratórios ou até mesmo o nosso animal de estimação para fazer a mesma coisa.

Quer dizer, a gente ensina ele a colocar o dedo no começo da frase.

E depois ensina a ele deslizar o dedo até o final, olhando para a ponta do dedo.

Daí, vai parecer que esse animal treinado está lendo.

Mas se você perguntar no final o que é que ele entendeu, ele só vai dar uma grande risada ...

Só que a gente pode ensinar nosso animal de estimação a fazer alguma coisa de útil.

E a gente pode ensinar o autômato também ...

Para isso, basta fazer com que as coisas mudem a medida que a leitura acontece.

Quer dizer, o caso mais simples possível é aquele onde as coisas podem estar de um jeito ou de outro.

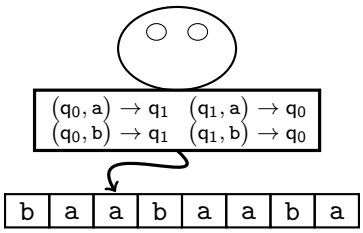
Usando o formalismo da matemática, a gente representa isso como q_0 e q_1.

E daí, a gente define as seguintes regras

$$\begin{array}{lll} (q_0, a) & \longrightarrow & q_1 \\ (q_0, b) & \longrightarrow & q_1 \end{array} \qquad\qquad \begin{array}{lll} (q_1, a) & \longrightarrow & q_0 \\ (q_1, b) & \longrightarrow & q_0 \end{array}$$

Quer dizer,

→ `Ao ler 'a' (ou 'b'),`
`se as coisas estão de um jeito, coloque as coisas do outro jeito`
`e vice-versa`



Quer dizer, a gente pode pensar que q_0 e q_1 são como duas luzinhas que ficam acendendo e apagando durante a computação.

Quando a leitura acaba, a gente pergunta outra vez: "*E daí, o que você entendeu?*"

E dessa vez a máquina pode dar duas respostas: q_0 ou q_1.

Mas, o que isso significa?

Bom, isso depende.

Quer dizer, depende de como a coisa começa.

Se a computação começa em q_0 e a palavra de entrada é

a	a	b	a

então no final a resposta é q_0.

Mas se a palavra de entrada for

b	a	a

ou

b	a	a	b	a

então no final a resposta é q_1.

Quer dizer

- Se a palavra de entrada tem tamanho par, então a resposta é q_0
- E se a palavra de entrada tem tamanho ímpar, então a resposta é q_1

— (*quando a computação começa em* $\mathtt{q}_0$)

E esse é o significado da resposta do autômato!

A gente também pode treinar o nosso pet para fazer isso.

E um dia, alguém ensinou isso pra você também ...

3 Mais autômatos finitos

Agora a gente já entendeu como os autômatos funcionam.

E daí, é fácil construir um autômato que realiza a seguinte tarefa

→ *Verificar se a quantidade de* $\mathtt{a}$*'s na palavra de entrada é par ou ímpar*

Bom, a gente já sabe que a resposta vai ser dada na forma de $\mathtt{q}_0$ e $\mathtt{q}_1$.

Mas dessa vez, a leitura de um $\mathtt{b}$ não muda o jeito como as coisas estão.

Daí que, as regras que a gente vai usar são as seguintes

$$\begin{array}{lll} (\mathtt{q}_0, \mathtt{a}) & \longrightarrow & \mathtt{q}_1 \\ (\mathtt{q}_0, \mathtt{b}) & \longrightarrow & \mathtt{q}_0 \end{array} \qquad\qquad \begin{array}{lll} (\mathtt{q}_1, \mathtt{a}) & \longrightarrow & \mathtt{q}_0 \\ (\mathtt{q}_1, \mathtt{b}) & \longrightarrow & \mathtt{q}_1 \end{array}$$

Se a computação começa em $\mathtt{q}_0$, então

- a resposta $\mathtt{q}_0$ significa que o número de $\mathtt{a}$'s é par
- a resposta $\mathtt{q}_1$ significa que o número de $\mathtt{a}$'s é ímpar

E aqui nós já temos mais um autômato.

3.1 Diagrama de estados e transições

Nós dissemos há pouco que $\mathtt{q}_0$ e $\mathtt{q}_1$ são como luzinhas que ficam acendendo e apagando durante a computação.

Mas a terminologia correta é que $\mathtt{q}_0, \mathtt{q}_1$ são os *estados da computação* — (*porque eles indicam como as coisas estão*)

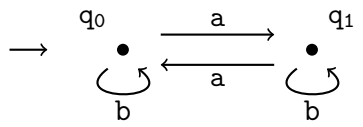
Daí a gente nota que as regras indicam como os estado da computação muda a medida que os símbolos vão sendo lidos

$$(\mathtt{q}_i, \mathtt{a}) \longrightarrow \mathtt{q}_j \qquad \text{e} \qquad (\mathtt{q}_i, \mathtt{b}) \longrightarrow \mathtt{q}_k$$

E por isso, elas são chamadas de *regras de transição* de estados — (*ou regras de mudança de estado*)

A computação do autômato é determinada completamente pelos estados e as regras de transição.

E a maneira mais conveniente de apresentar essa informação é por meio de um *diagrama de estados e transições*



onde a setinha ao lado de $\mathtt{q}_0$ indica que ele é o *estado inicial* — (*i.e., o estado onde a computação começa*)

exemplos

a) Paridade de a's e b's

Imagine que nós queremos construir um autômato que realiza a seguinte tarefa

→ *Verificar se a quantidade de* a*'s é par ou ímpar*
e se a quantidade de b*'s é par ou ímpar*

Bom, dessa vez as coisas podem estar de 4 maneiras diferentes

- quantidade par de a's e quantidade par de b's
- quantidade par de a's e quantidade ímpar de b's
- quantidade ímpar de a's e quantidade par de b's
- quantidade ímpar de a's e quantidade ímpar de b's

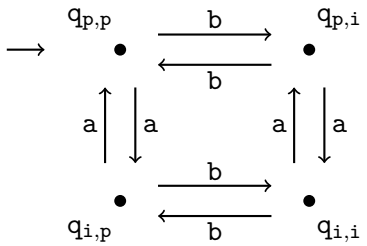
Daí que, o autômato deve ter 4 estados: $\mathtt{q}_0$, $\mathtt{q}_1$, $\mathtt{q}_2$, $\mathtt{q}_3$.

Mas, é mais conveniente escrever as coisas assim: $\mathtt{q}_{00}$, $\mathtt{q}_{01}$, $\mathtt{q}_{10}$, $\mathtt{q}_{11}$.

Ou então assim: $\mathtt{q}_{pp}$, $\mathtt{q}_{pi}$, $\mathtt{q}_{ip}$, $\mathtt{q}_{ii}$.

Quer dizer, o primeiro índice indica a paridade dos a's, e o segundo indica a paridade dos b's.

Daí, com essa interpretação, não é difícil chegar ao seguinte diagrama de estados e transições



◇

b) Blocos de tamanho par

Agora imagine que nós queremos um autômato que realiza a seguinte tarefa

→ *Verificar se todos os blocos de* a*'s na palavra de entrada possuem tamanho par*

Quer dizer, um *bloco* é uma sequência de símbolos consecutivos iguais.

Daí que, nós sempre podemos decompor a palavra de entrada em blocos

$$\texttt{aaabaabbbaab} \quad \Rightarrow \quad \texttt{aaa} \mid \texttt{b} \mid \texttt{aa} \mid \texttt{bbb} \mid \texttt{aa} \mid \texttt{b}$$

E a tarefa consiste em verificar se todos os blocos de a's da palavra possuem tamanho par.

No exemplo acima, a resposta é NÃO.

Certo.

Mas como a gente constrói um autômato que verifica isso ?

Bom, a ideia é que o autômato vai "contando" os a's do bloco — (*usando os estados* q_0 *e* q_1)

O bloco chega ao fim quando a gente lê um b.

E daí a gente vai saber se ele tem tamanho par ou ímpar.

Quando o bloco tem tamanho par, a gente apenas espera o próximo bloco aparecer — (*fazendo a leitura dos* b*'s*)

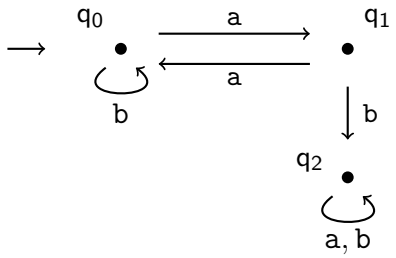
Mas, o que acontece quando o bloco tem tamanho ímpar?

Bom, nesse caso a gente já sabe que a resposta é NÃO.

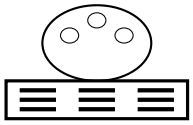
Mas essa resposta só é dada no final da leitura.

Daí que, a ideia é mudar para um estado q_2 daonde a gente não sai nunca mais — (*porque a gente já sabe que a resposta é* NÃO)

A coisa fica assim



Esse é um autômato com 3 luzinhas — (*ou 3 estados*)



Daí que, no final da leitura ele pode dar 3 respostas: q_0, q_1 ou q_2.

Mas, o que isso significa?

Bom, a gente já sabe que o q_2 significa NÃO.

E não é difícil ver que o q_0 significa SIM.

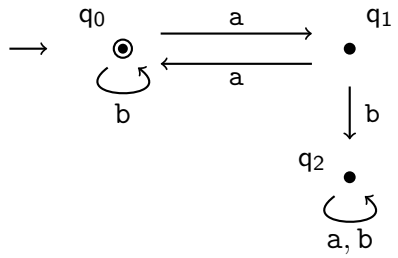
Mas e o q_1*, o que significa?*

Bom, se a computação acaba em q_1, isso significa que

- no meio da leitura, não foi encontrado nenhum bloco ímpar de a's
- mas a computação está terminando após a leitura de uma quantidade ímpar de a's

Daí que, q_1 também deve dar resposta NÃO.

A gente indica a resposta de cada estado no diagrama da seguinte maneira



quer dizer, os estados com resposta SIM são destacados com um círculo. ◇

c) Todo b é precedido por ao menos 3 a's

Agora imagine que nós queremos construir um autômato para a seguinte tarefa

→ *Verificar se todo símbolo* b *da palavra de entrada é precedido imediatamente por ao menos 3* a*'s*

Por exemplo, todas as palavras abaixo tem essa propriedade

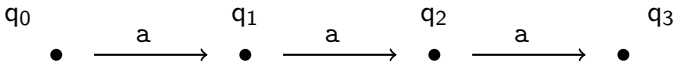
aaaa, aaabaa, aaaabaaab, etc.

Mas essas outras não tem

baaa, aabaaa, aaabbaaab, etc.

Bom, dessa vez nós vamos precisar contar o tamanho dos blocos de a's — (*mas só até 3 ...*)

Isso pode ser feito com os estados q_0, q_1, q_2, q_3, da seguinte maneira



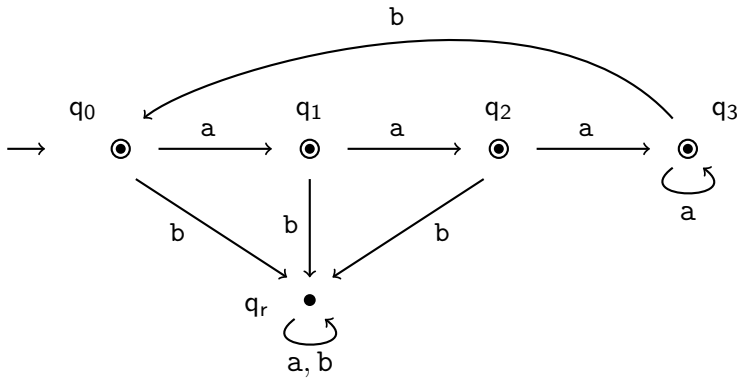
— (*veja que isso é a mesma coisa que contar com os dedos ...*)

Daí, o b só pode aparecer no estado q_3.

Quando o b aparece, nós reiniciamos a contagem — (*voltando para* q_0)

E se ele não aparece, nós ficamos em q_3 esperando ele aparecer — (*lendo os* a*'s que aparecem*)

E se o b aparecer antes da hora, nós vamos para o estado que sempre diz NÃO.



Os estados daonde a gente nunca sai mais são chamados de *estados de rejeição*.

Porque a ideia é que o autômato *rejeita* as palavras que não tem a propriedade.

E *aceita* as palavras que tem a propriedade.

Daí que, os estados que respondem SIM são chamados de *estados de aceitação* — (*nos livros, esses estados são chamados de estados finais*) ◇

d) Reconhecimento de padrão

Agora imagine que nós queremos um autômato para a seguinte tarefa

→ *Verificar se a palavra de entrada contém o padrão* `aaba` *ou não*

Por exemplo, as palavras abaixo contém o padrão

`aaba, bbbaaba, aaabbaabaab, etc`

Mas essas outras não contém

`baba, bbbbaba, aaabbabbaab, etc`

Bom, aqui a gente poderia perguntar: "*Como é que você realiza essa tarefa?*"

E você poderia responder:"*Ora, eu apenas vejo se o padrão está lá ou não*".

Certo.

Mas então, a gente poderia perguntar: "*Como é que o nosso animal de estimação realizaria essa tarefa?*"

Quer dizer, o animal do laboratório não sabe o que é um padrão ...

Então, a gente poderia pensar como é que a gente ensina esse animalzinho a realizar essa tarefa — (*mesmo sem ele saber o que é um padrão ...*)

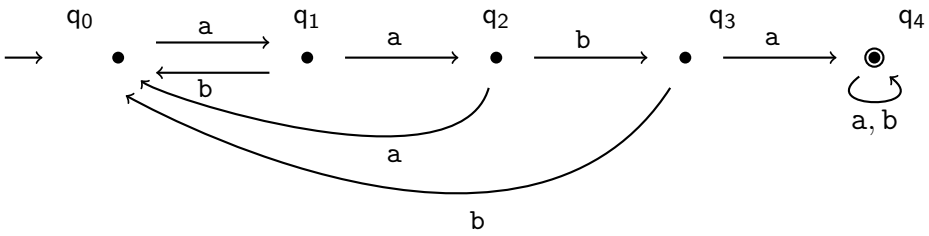
E daí, a gente poderia chegar nas seguintes instruções

```
- Espere até que apareça o primeiro 'a'
- Daí, veja se o próximo símbolo também é um 'a'
    se não for, comece tudo outra vez
```

```
- Daí, veja se o próximo símbolo é um 'b'
    se não for, comece tudo outra vez
- Daí, veja se o próximo símbolo é um 'a'
    se for, aguarde o final da leitura e responda SIM
    e se não for, comece tudo outra vez
```

Isso é como um pequeno programinha, não é?

Mas também pode ser colocado na forma de um diagrama de estados e transições



Só que esse autômato não está correto ...

Quer dizer, ele responde NÃO para a palavra de entrada: aaaba.

Mas a resposta correta é SIM.

Hmm ...

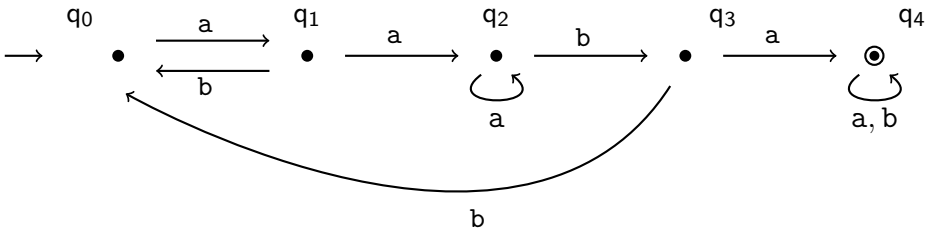
O que está acontecendo?

Bom, o que está acontecendo é que a leitura do terceiro a não devia levar a gente de volta ao início.

Porque nesse ponto, nós já temos dois a's consecutivos.

E o ba que vem em seguida completa o padrão.

Mas agora que a gente viu isso, não é difícil corrigir o autômato



◇