# Aula 2

Mario Leston

2 de março de 2021

Um sistema é uma entidade constítuida de estados e transições. Um estado intuitivamente captura as informações que são relevantes para o sistema e determina como o sistema pode evoluir no futuro. Uma transição, que pode ocorrer espontaneamente ou como resposta a um evento externo, produz uma mudança no estado fornecendo assim um novo estado. Um sistema constituído de um número finito de estados e transições é chamado de sistema finito de transições ou de autômato finito determinístico.

## 1 Autômatos Finitos Determinísticos

Um **autômato finito determinístico** (AFD) $M$ é constituído dos seguintes ingredientes:

- um conjunto finito $Q$ cujos elementos são chamados de **estados**,
- um alfabeto $\Sigma$,
- uma **função de transição** $\delta : Q \times \Sigma \to Q$,
- um estado $s \in Q$, chamado de **estado inicial**, e
- um subconjunto $F$ de $Q$, chamado de conjunto dos estados **finais** ou de **aceitação**.

É conveniente coletar estes elementos em uma tupla e escrever

$$M = (Q, \Sigma, \delta, s, F).$$

A função de transição é o objeto mais complicado na definição de um AFD. A função $\delta$ recebe um estado $p \in Q$ e um símbolo $a \in \Sigma$ e produz um "novo" estado $\delta(p, a) \in Q$. Às vezes, é conveniente escrever

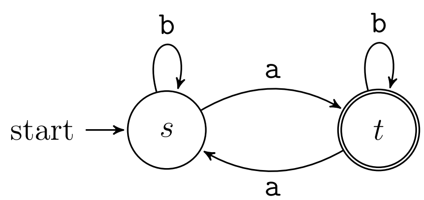
$$p \xrightarrow{a} q$$

em vez de $\delta(p, a) = q$. É também sugestivo chamar o objeto $p \xrightarrow{a} q$ de **transição** e dizer que $a$ é o **rótulo** desta transição.

**Exemplo 1.** Considere o autômato

$$(\{s, t\}, \{\mathtt{a}, \mathtt{b}\}, \{(s, \mathtt{a}, t), (s, \mathtt{b}, s), (t, \mathtt{a}, s), (t, \mathtt{b}, t)\}, \{s\}, \{t\})$$

que pode ser representado graficamente como:




## 2 Computação de um AFD

No que segue, admita que $M = (Q, \Sigma, \delta, s, F)$ é um AFD. Vamos descrever informalmente como $M$ funciona. O AFD $M$ é um reconhecedor de palavras. Assim, quando submetido a uma palavra $x \in \Sigma^*$, o AFD $M$ processa $x$ e produz uma e só uma de suas alternativas: ou $x$ é aceita por $M$, ou é rejeitada por $M$.

O AFD $M$ recebe uma palavra $x \in \Sigma^*$ como entrada. O processamento começa no estado inicial, $s$. A palavra $x$ é processada da esquerda para a direita, e a cada passo um símbolo de $x$ é consumido. Admita que o estado atual de $M$ é $p \in Q$. No início, $p = s$. Se a entrada $x$ tiver sido totalmente consumida, então a palavra $x$ é aceita se $p$ é um estado final, ou seja, se $p \in F$, caso contrário, a palavra $x$ é rejeitada. Suponha que $x$ ainda não foi totalmente consumida e seja $a \in \Sigma$ o próximo símbolo de $x$ a ser consumido. O AFD $M$ aplica a função de transição $\delta$ ao par $(p, a)$ para produzir o próximo estado $\delta(p, a)$. O processamento é então repetido com $\delta(p, a)$ fazendo o papel de $p$. Tal processamento produz um passeio pelos estados de $M$ que soletram a palavra $x$.

Para formalizar a definição acima, vamos estender a função $\delta$ criando uma nova função capaz de "processar" palavras em vez de símbolos. Assim, definimos

$$\delta^* : Q \times \Sigma^* \to Q$$

pondo-se

$$(1) \qquad \begin{aligned} &\text{(i)}\ \ \delta^*(p,\epsilon) = p \text{ para cada } p \in Q, \text{ e} \\ &\text{(ii)}\ \ \delta^*(p,ax) = \delta^*(\delta(p,a),x) \text{ para cada } p \in Q, a \in \Sigma, x \in \Sigma^*. \end{aligned}$$

**Exemplo 2.** Considere o AFD do Exemplo 1. Então

$$\begin{aligned} \delta * (\mathtt{abba}, s) &= \delta^*(\delta(s,\mathtt{a}),\mathtt{bba}) \\ &= \delta^*(t,\mathtt{bba}) \\ &= \delta^*(\delta(t,b),\mathtt{ba}) \\ &= \delta^*(t,\mathtt{ba}) \\ &= \delta^*(\delta(t,b),\mathtt{a}) \\ &= \delta^*(t,\mathtt{a}) \\ &= \delta^*(\delta(t,\mathtt{a}),\epsilon) \\ &= \delta^*(s,\epsilon) \\ &= s. \end{aligned}$$

**Implementação** Eis uma implementação simples de um AFD. Como o nome dos estados é um tanto quanto irrelevante, vamos admitir no que segue que os estados são números inteiros. Um momento de reflexão permite notar que o "conceito central" é o de uma função de transição. Uma função de transição $\delta$ pode ser implementada usando-se um `Map<Pair<Int, Char>, Int>` e a extensão $\delta^*$ de $\delta$ pode ser obtida da seguinte forma:

```
fun star(delta: Map<Pair<Int, Char>, Int>) : ((Int, String) → Int) {
    return { s, ws → ws.fold (s)
                         { p, c → delta[Pair(p, c)]
                                  ?: throw IllegalCallerException() }
    }
}
```

A função `star` recebe uma implementação `delta` de uma função de transição $\delta$ e devolve uma implementação de $\delta^*$.

## 3 Linguagem reconhecida por um AFD

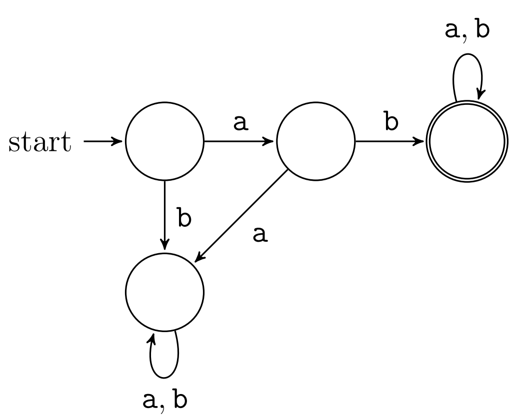
A cada AFD vamos associar uma linguagem que consiste das palavras que são aceitas pelo AFD. Seja $M = (Q, \Sigma, \delta, s, F)$ um AFD. Dizemos que uma palavra $x \in \Sigma^*$ é **aceita** (ou **reconhecida**) por $M$ se $\delta^*(s, x) \in F$, ou seja, se o estado obtido após o consumo da palavra $x$ é um estado final de $M$, caso contrário, isto é, se $\delta^*(s, x) \notin F$, então dizemos que $x$ é **rejeitada** por $M$. A **linguagem aceita** ou (**reconhecida**) por $M$, denotada $L(M)$, consiste do conjunto das palavras $x \in \Sigma^*$ que são aceitas por $M$; mais formalmente,

(2) $$L(M) = \{x \in \Sigma^* \mid \delta^*(s, x) \in F\}.$$

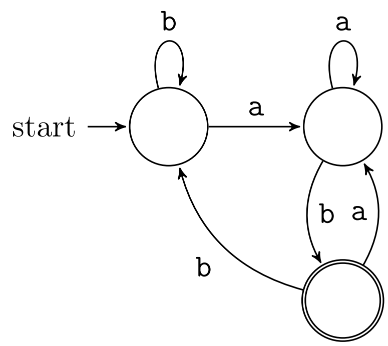
**Implementação** Agora, podemos exibir uma implementação de um AFD. A função `dfa` recebe uma implementação `delta` de uma função de transição $\delta$, um estado implementado como `s:int` e um conjunto de estados finais implementado como `fs:Set<Int>` e devolve uma função que recebe uma palavra, representada como uma `String`, e devolve `true` se, e só se, tal palavra é aceita pelo AFD.

```
fun dfa(delta: Map<Pair<Int, Char>, Int>, s: Int, fs: Set<Int>)
       : ((String) → Boolean) {
    val deltaStar = star(delta)
    return { ws → fs.contains(deltaStar(s, ws)) }
}
```

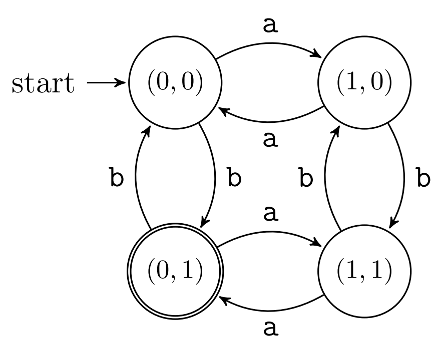
**Exemplo 3.** Eis um desenho de um AFD que reconhece o conjunto das palavras $w$ sobre $\{\mathtt{a}, \mathtt{b}\}$ tais que `ab` é um prefixo de $w$:



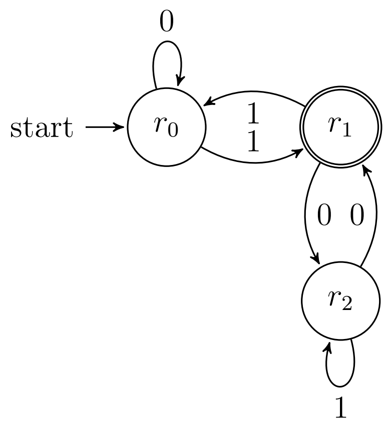
**Exemplo 4.** Eis um autômato que reconhece o conjunto das palavras $w$ sobre $\{\texttt{a}, \texttt{b}\}$ tais que $\texttt{ab}$ é um sufixo de $w$:




**Exemplo 5.** Eis um autômato que reconhece o conjunto das palavras $w$ sobre $\{\texttt{a}, \texttt{b}\}$ tais que $|w|_{\texttt{a}}$ é par e $|w|_{\texttt{b}}$ é ímpar:




**Exemplo 6.** Eis um autômato que reconhece o conjunto das palavras $w$ sobre os dígitos 0 e 1 tais que $\sum_{i=1}^{|w|} 2^{|w|-i} w_i \bmod 3 = 1$. ou seja, $w$, quando interpretado como um número na base 2, tem resto 1 se dividido por 3.


0
start
$r_0$
1
1
$r_1$
0
0
$r_2$
1