Linguagens Formais e Autômatos

Aula 7: Autômatos Finitos

Prof. Dr. Rodrigo Xavier de Almeida Leão

Cientista de Dados e Big Data

Autômatos Finitos Determinísticos (AFDs) são reconhecedores eficientes de linguagens regulares.

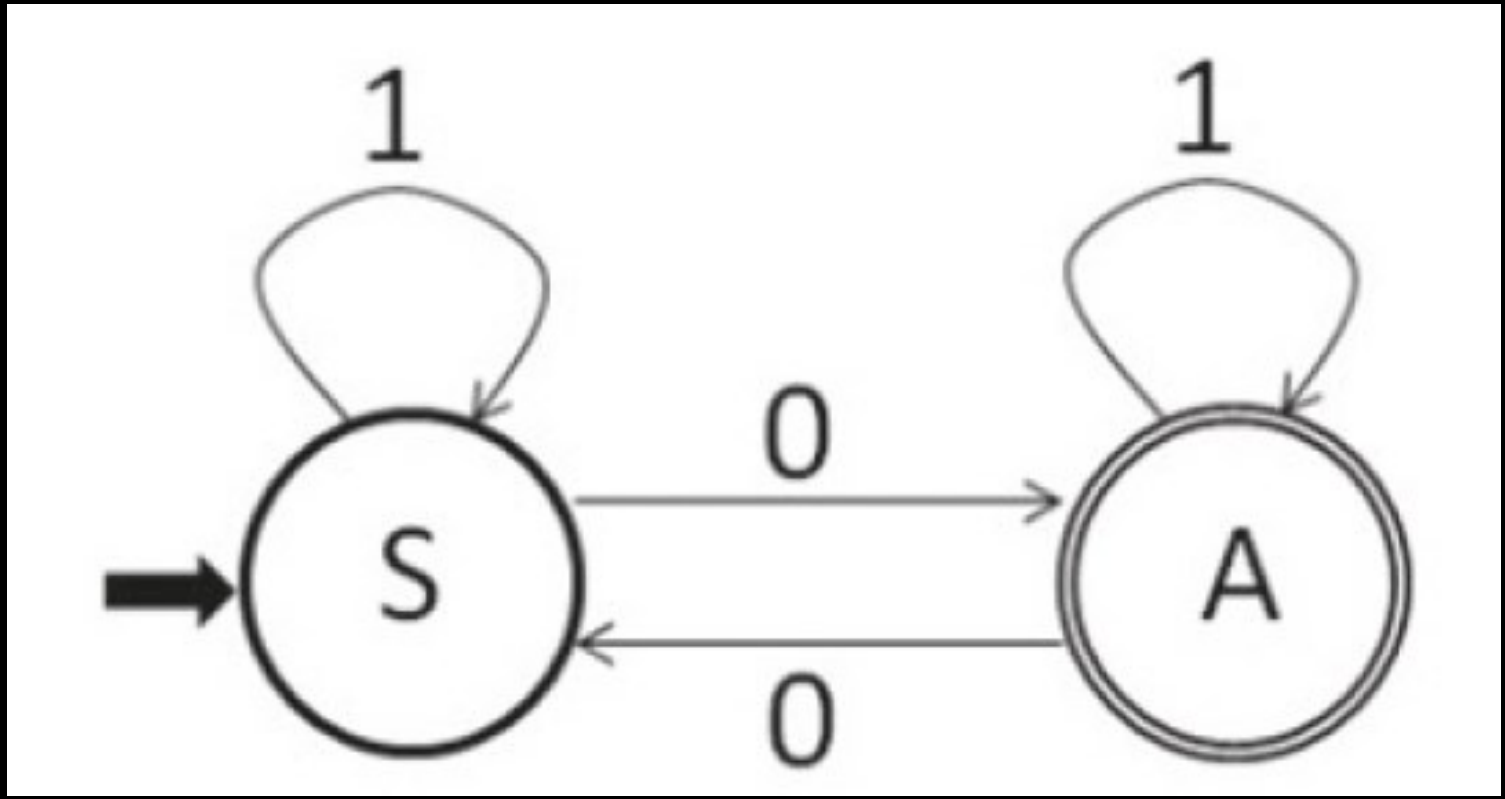
$$S \rightarrow 0A \mid 1S$$

$$A \rightarrow 0S \mid 1A \mid \epsilon$$

Esta gramática gera as cadeias sobre o alfabeto $\Sigma = \{0,1\}$, que têm quantidade ímpar de caracteres '0'. Por exemplo, esta gramática gera a cadeia 01010 da seguinte forma:

$$S \Rightarrow 0A \Rightarrow 01A \Rightarrow 010S \Rightarrow 0101S \Rightarrow 01010A \Rightarrow 01010$$

Programar a análise sintática é resolver o problema oposto. Dada a cadeia 01010, podemos determinar se ela está na linguagem gerada pela gramática? Para programar a solução podemos guardar sempre qual é o único não terminal que está no final da cadeia sendo gerada.



Chamamos esta solução de AFD – Autômato Finito Determinístico. A ideia é que cada bola representa um "estado". Começamos pelo estado marcado com a seta, no caso, o estado 'S'. Ao lermos um 0 vamos para o estado 'A'; ao lermos um '1', permanecemos no estado 'A'. Depois, ao lermos o segundo '0', voltamos ao estado 'S'. Ao terminarmos de ler a cadeia de entrada, se estivermos em um estado marcado com círculo duplo, neste caso no estado 'A', dizemos que a cadeia foi 'reconhecida'. Caso contrário, a cadeia não foi reconhecida pelo AFD (GARCIA, 2017).

Em um AFD, dado um estado e um símbolo do alfabeto de entrada, existe uma seta, ou transição para exatamente um único estado. Assim , o AFD está sempre em exatamente um estado, por isso é chamado de determinístico. Observamos que o AFD possui estados. No AFD da Figura 2.1 os estados são 'S' e 'A'. O estado 'S' é o estado inicial (marcado com uma seta), no qual o AFD inicia sua execução. O estado 'A' é um estado final e se o autômato termina em um estado final (marcado com círculo duplo), ele reconheceu com sucesso a cadeia lida. Uma forma alternativa de representar este mesmo AFD é a Tabela 2.1:

	0	1
→ S	A	S
*A	S	A

Neste caso, o estado inicial é marcado com a seta. Os estados finais são marcados com o símbolo '*' e, para sabermos para onde vai a seta do estado q com entrada a basta consultar a linha correspondente ao estado q e a coluna correspondente ao símbolo de entrada a. As

Formalmente um AFD M é uma tupla $(Q, \Sigma, \delta, q_0, F)$, onde:

- Q é um conjunto finito e não vazio de estados.
- Σ é o alfabeto de entrada.
- $\delta : Q \times \Sigma \rightarrow Q$ é a função de transição de estados.
- $q_0 \in Q$ é o estado inicial
- $F \subseteq Q$ é o conjunto de estados finais.

No AFD da Figura 2.1 temos que:

- $Q = (S, A)$;
- $\Sigma = \{0,1\}$;
- $\delta : Q \times \Sigma \rightarrow Q$ é a função na qual:

$\delta(S,0) = A$, $\delta(S,1) = S$, $\delta(A,0) = S$ e $\delta(A,1) = A$;

- S é o estado inicial;
- $\{A\}$ é o conjunto de estados finais.

Para entendermos como o AFD funciona é útil estendermos a função $\delta : Q \times \Sigma \to Q$ para esta extensão, isto é, a função $\hat{\delta}$,representa a ação do AFD ao ler uma cadeia, ou seja, ao ler mais de um caractere. Sua definição é intuitiva: se ao lermos um 0 temos que $\delta(S,0) = A$, e o AFD fica no estado A . Ao lermos um segundo 0, temos $\delta(A,0) = S$ e o AFD retorna ao estado S. Isso significa que $\hat{\delta}(S,00) = S$. Formalmente podemos definir $\hat{\delta}$ em função de δ da seguinte forma:

- $\hat{\delta}(q,\epsilon) = q$, para todo $q \in Q$.
- $\hat{\delta}(q,aw) = \hat{\delta}(\delta(q,a),w)$, para todo $q \in Q, a \in \Sigma, w \in \Sigma^*$.

Aplicando esta definição ao AFD da Figura 2.1, com entrada 00, temos o seguinte:

$$\hat{\delta}(S,00) = \hat{\delta}(\delta(S,0),0) = \hat{\delta}(A,0) = \hat{\delta}(\delta(A,0),\epsilon) = \hat{\delta}(S,\epsilon) = S$$

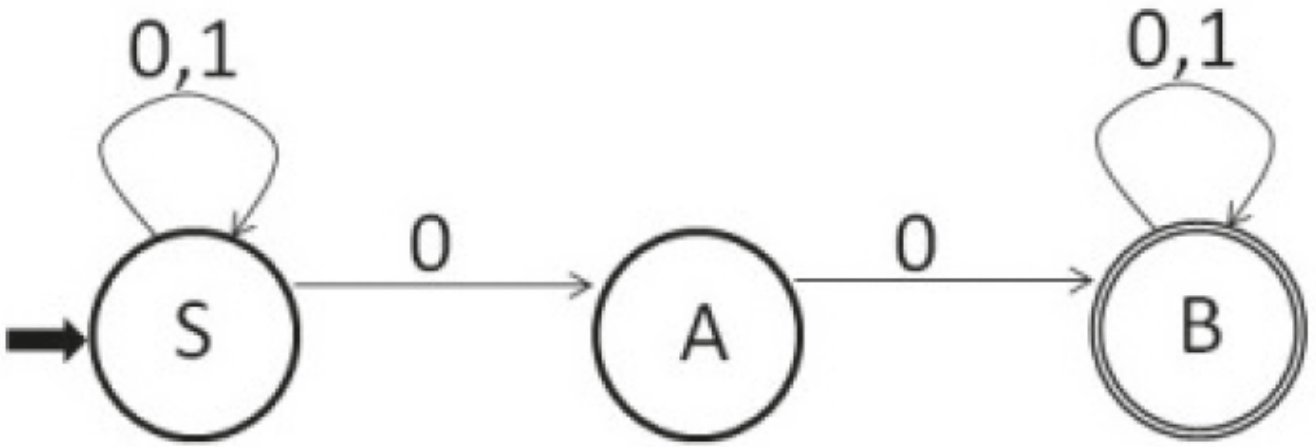
Portanto, a definição funciona como esperávamos.

Dado um AFD M = (Q,Σ,δ,q_0,F), definimos a linguagem reconhecida por M como:

- $T(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0,w) \in F\}$

No caso do autômato da Figura 2.1 temos que T(M)={0,01,10,011,101, 110,000,011,...}, o conjunto de todas as cadeias com número ímpar de caracteres 0. Gostaríamos que o AFD fosse uma boa forma de resolver o problema da análise sintática para todas as linguagens regulares. Entretanto, não é claro como obter um AFD de algumas gramáticas regulares. Por exemplo, a gramática G_2 - descrita logo a seguir -, gera as cadeias sobre o alfabeto $\Sigma = \{0,1\}$ que possuem a subcadeia '00', isto é, que têm dois caracteres 0 seguidos. G_2 possui as regras:

- $S \rightarrow 0S \mid 1S \mid 0A$
- $A \rightarrow 0B$
- $B \rightarrow 0B \mid 1B \mid \epsilon$



Fonte: elaborada pelo autor.

Podemos dizer que nesta figura a função δ não retorna um estado, mas um conjunto de estados. Por exemplo, $\delta(S,0) = \{S, A\}$, enquanto que $\delta(A,1) = \varnothing$, ambos os casos violam a condição de que para cada estado e símbolo de entrada as setas nos levem a exatamente um estado. Neste caso, dizemos que a figura representa um AFND – Autômato Finito Não Determinístico. O AFND da Figura 2.2 também pode ser representado em outra forma , como apresentado na Tabela 2.2:

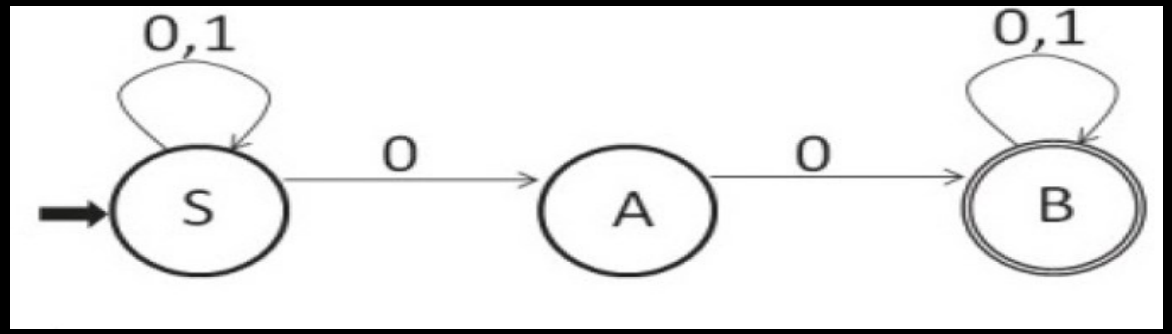
	0	1
→ S	{S, A}	{S}
A	B	∅
*B	B	B

Podemos interpretar um AFND como podendo seguir diversos caminhos ao ler uma entrada. Se ao menos 1 dos caminhos levar o AFND a um estado final, dizemos que o AFND reconhece w.

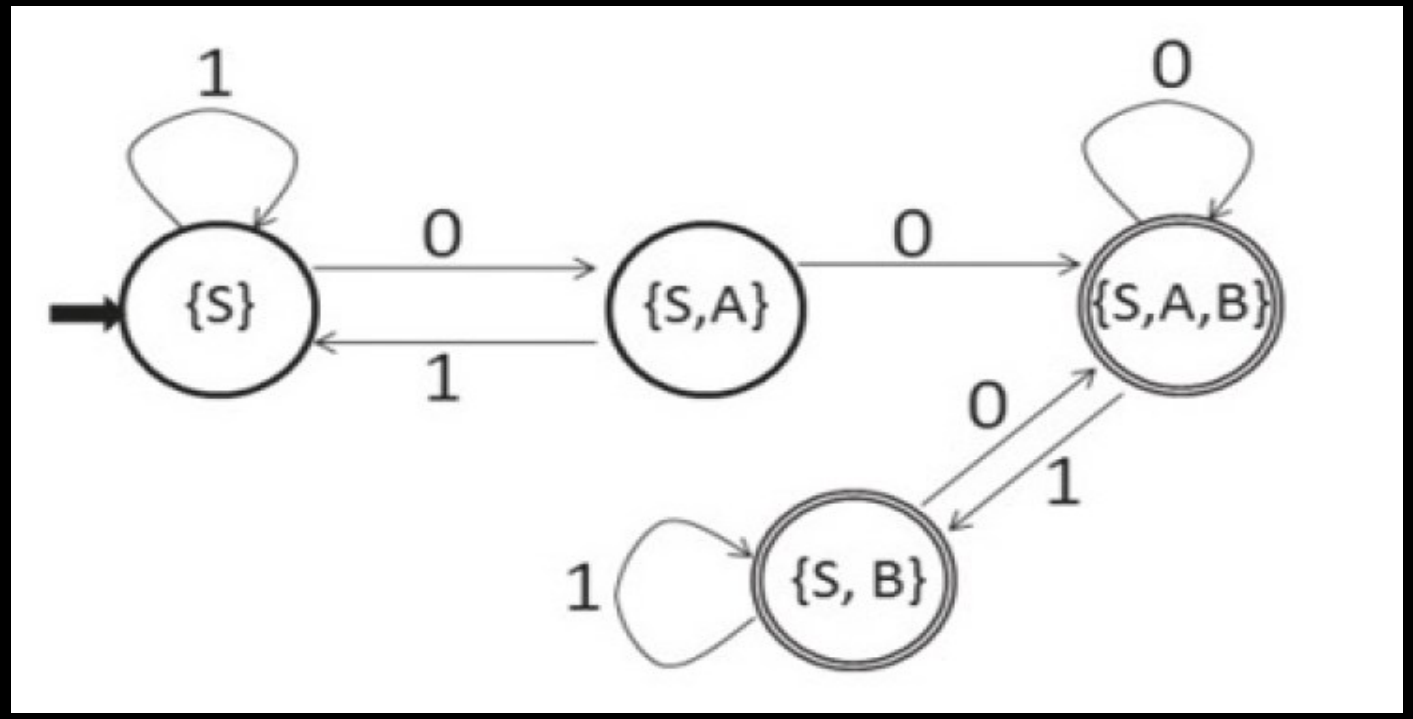
Observe que a função δ neste caso não retorna um estado, mas sim um conjunto de estados, em particular pode ser o conjunto vazio. Assim a assinatura da função deve ser $\delta : Q \times \Sigma \rightarrow \wp(Q)$.

O AFND da Figura 2.2 pode ser transformado em um AFD. Dissemos que um AFND pode seguir diversos caminhos ao ler uma entrada w. Vamos fazer um AFD que guarda os estados de todos esses possíveis caminhos. Como o AFND em questão tem apenas três estados, cada um dos possíveis caminhos seguidos ao lermos uma cadeia w estará em um desses três estados, portanto, os caminhos combinados poderão estar em nenhum estado, ou somente no estado {S}, ou em uma combinação de estados, por exemplo, {S, A, B}. Na pior das hipóteses há $2^3 = 8$ estados possíveis para esses caminhos combinados. Podemos assim desenhar o AFD da Figura 2.3:

AFDN



AFD



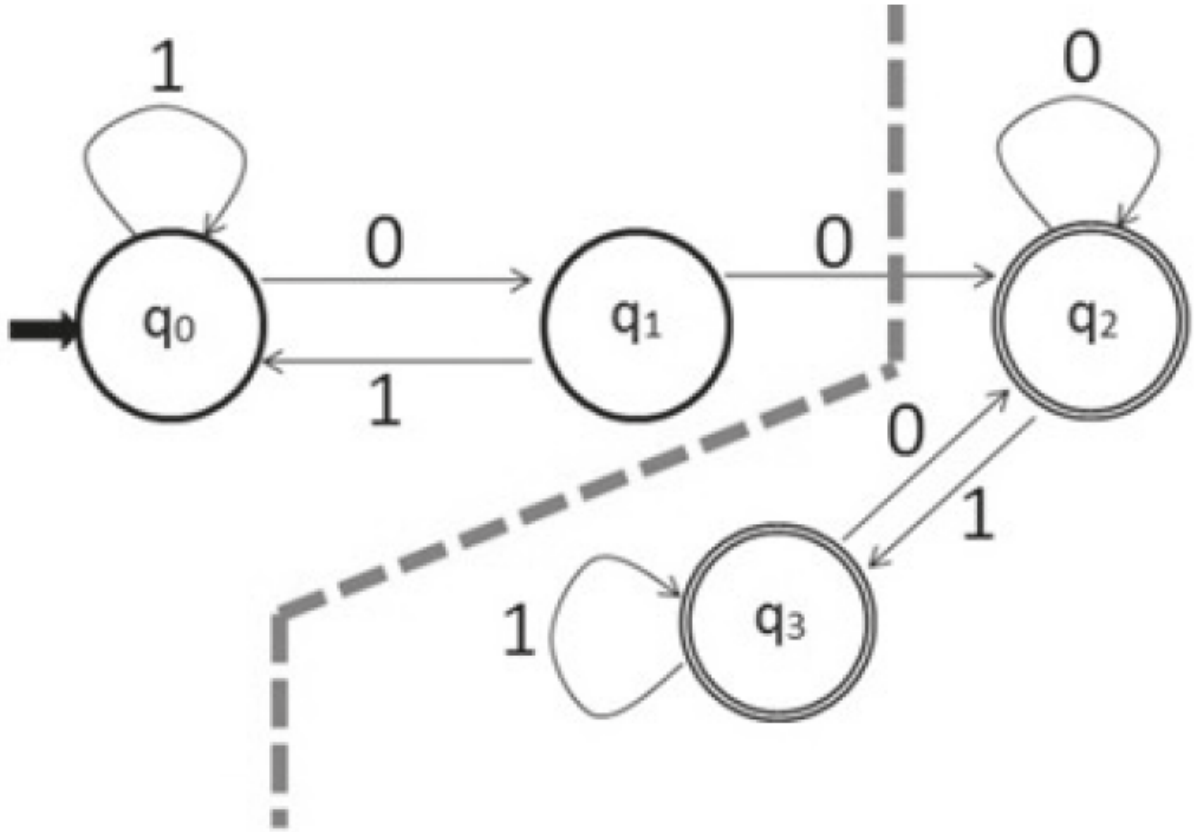
Conforme explicado, cada estado do AFD corresponde a todos os possíveis estados que o AFND estaria após ler determinada entrada. Em

Os estados finais do AFD correspondem aos conjuntos de estados do AFND que contém ao menos um estado final. No exemplo {S,A,B} e {S,B} são estados finais, porque contém o estado final B.

Dada uma gramática regular sabemos obter um AFND equivalente, a construção anterior nos mostra que dado um AFND podemos obter um AFD equivalente, portanto, para toda gramática regular G sabemos obter um AFD M equivalente, isto é, tal que $T(M) = L(G)$. Isso significa que podemos fazer a análise sintática de forma eficiente para todas as linguagens regulares. Se fizermos um programa que simule um AFD, este consome tempo proporcional ao tamanho da entrada, pois para cada símbolo da cadeia de entrada o AFD precisa fazer um movimento entre dois estados. O consumo de memória, por sua vez, é constante

A Figura 2.4 exibe o mesmo AFD que a Figura 2.3. Seus estados foram renomeados para a forma mais usual $q_0, q_1, \ldots$, e desenhamos uma linha tracejada para facilitar o entendimento da melhoria que pode ser feita no AFD.

Figura 2.4 | AFD da figura 2.3 com estados renomeados



Na Figura 2.4 há uma seta que ultrapassa a linha tracejada da esquerda para a direita, e não há seta no sentido oposto. Isto significa que uma vez ultrapassada a seta, não há "volta", o que corresponde à especificação da linguagem, composta pelas cadeias sobre $\Sigma = \{0,1\}$ que possuem a subcadeia '00'. Uma vez que a subcadeia '00' ocorra na cadeia, qualquer continuação da cadeia terá esta característica.

Em um AFD, os estados q e r são equivalentes, se para toda entrada possível o AFD se comporta da mesma forma (reconhece as mesmas entradas) estando em q ou estando em r.

Na Figura 2.4 há uma seta que ultrapassa a linha tracejada da esquerda para a direita, e não há seta no sentido oposto. Isto significa que uma vez ultrapassada a seta, não há "volta", o que corresponde à especificação da linguagem, composta pelas cadeias sobre $\Sigma = \{0,1\}$ que possuem a subcadeia '00'. Uma vez que a subcadeia '00' ocorra na cadeia, qualquer continuação da cadeia terá esta característica.

Em um AFD, os estados q e r são equivalentes, se para toda entrada possível o AFD se comporta da mesma forma (reconhece as mesmas entradas) estando em q ou estando em r.

Dado um AFD M = $(Q, \Sigma, \delta, q_0, F)$, e estados $q, r \in Q$, definimos que q e r são equivalentes, ou $q \equiv r$, quando:

- $\forall w \in \Sigma^*, \delta(q, w) \in F \Leftrightarrow \delta(r, w) \in F$

Na Figura 2.4 é apresentado o caso que $q_2 \equiv q_3$ porque depois da linha tracejada qualquer entrada sempre leva a um estado final. Em geral, é mais fácil identificar estados que não são equivalentes do que estados que são equivalentes. Se fizermos a negação da definição de equivalência temos que:

$q \not\equiv r$ se, e somente se, $\exists w \in \Sigma^*, \delta(q, w) \in F \Leftrightarrow \delta(r, w) \notin F$