

Autômatos finitos não determinísticos

Linguagens Formais e Autômatos



Prof. Daniel Saad Nogueira
Nunes

IFB – Instituto Federal de Brasília,
Campus Taguatinga



Sumário

1 Introdução



Introdução

- Em DFAs, dado um estado qualquer q e um símbolo de entrada, sabemos exatamente qual será o próximo estado.
- Em outras palavras, todo passo de computação é gerado unicamente do passo anterior.
- Chamamos isso de **determinismo**.



Introdução

- Em uma situação de **não-determinismo**, podemos derivar diferentes passos de computação a partir de um único ponto.
- Isto é, dado um estado q e um símbolo de entrada, podemos produzir como resultado, mais de um possível estado.



Introdução

- O **não-determinismo** pode ser visto como uma generalização do **determinismo**, visto que, todo modelo determinístico também é um não-determinístico.
- Estudaremos a versão não-determinística dos autômatos finitos: os **autômatos finitos não-determinísticos**, ou simplesmente, **NFAs**.



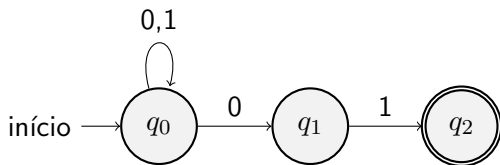
Sumário

2 NFAs



NFAs

Tome o seguinte NFA:



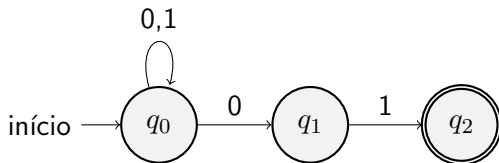
Este autômato reconhece a linguagem

$$L = \{w \mid w \in \{0, 1\}^* \text{ e } w \text{ termina com } 01\}$$



NFAs

Tome o seguinte NFA:

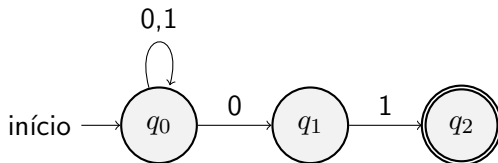


Ao se deparar com o símbolo 0 no estado q_0 , o que o autômato faz?
Para qual estado ele vai?



NFAs

Tome o seguinte NFA:

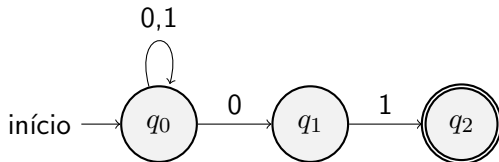


Ele permanece em q_0 e vai para q_1 , simultaneamente!



NFAs

Tome o seguinte NFA:

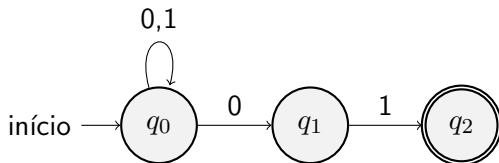


Esse autômato tem a capacidade de adivinhar quando estamos nos dois últimos símbolos da entrada.



NFAs

Tome o seguinte NFA:



Como ele opera quando a entrada é $w = 00101$?



NFAs

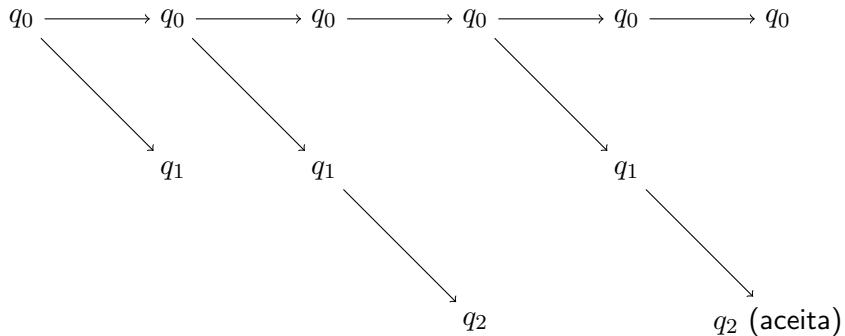


Figura: Processamento do NFA sobre a entrada $w = 00101$



NFAs: definição formal

Definição (Definição formal de um NFA)

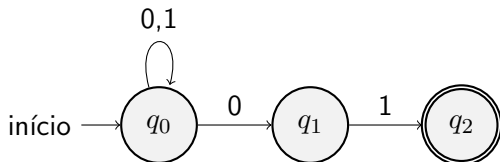
Um NFA é uma 5-tupla $(Q, \Sigma, \delta, q_0, F)$, em que:

- Q é um conjunto finito de **estados**,
- Σ é o **alfabeto de entrada**,
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ é a **função de transição**,
- $q_0 \in Q$ é o estado inicial,
- $F \subseteq Q$ é o **conjunto de estados de aceitação**.



NFAs: definição formal

Para o NFA de exemplo abaixo, temos a seguinte função de transição:



	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	\emptyset	$\{q_2\}$
q_2	\emptyset	\emptyset



NFAs: noção formal de computação

Definição (Aceitação em NFAs)

Seja $N = (Q, \Sigma, \delta, q_0, F)$ um NFA e $w = w_1w_2 \dots w_n \in \Sigma^*$. Dizemos que N aceita w se existe uma sequência de estados r_0, r_1, \dots, r_n com as seguintes condições:

- $r_0 = q_0$
- $r_{i+1} \in \delta(r_i, w_{i+1}), 0 \leq i < n$
- $r_n \in F$

Em outras palavras, basta que um ramo de computação chegue em um estado de aceitação ao final da entrada w para que N aceite w .



Sumário

3 Equivalência



Equivalência de NFAs e DFAs

- Apesar de ser uma característica interessante, NFAs não resolvem mais problemas que DFAs.
- Os dois formalismos reconhecem a mesma classe de linguagem, as **linguagens regulares**.
- Conseguimos mostrar que todo NFA possui um DFA equivalente, isto é, que reconhece a mesma linguagem.



Equivalência de NFAs e DFAs

Ideia da prova

A ideia é, a partir de um NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$, construir um DFA $D = (Q_D, \sigma, \delta_D, q_0, F_D)$.



Equivalência de NFAs e DFAs

Ideia da prova

No caso $Q_D = \mathcal{P}(Q_N)$, ou seja, se N possui $|Q_N|$ estados, D possuirá $2^{|Q_N|}$ estados. Contudo, alguns estados poderão ser excluídos em uma etapa posterior, pois não serão acessíveis do estado inicial. Cada um desses estados é rotulado com um subconjunto de Q_N .



Equivalência de NFAs e DFAs

Ideia da prova

O conjunto de estados de aceitação F_D será um subconjunto de Q_D de forma que todo $S \in F_D$, temos que se $X \in F_N$, temos que $X \in S$. Em outras palavras, os estados de aceitação em F_D serão aqueles que, tem ao menos, um estado de aceitação em F_N .



Equivalência de NFAs e DFAs

Ideia da prova

Por fim, para qualquer $S \in Q_D$ e um símbolo $a \in \Sigma$, temos que:

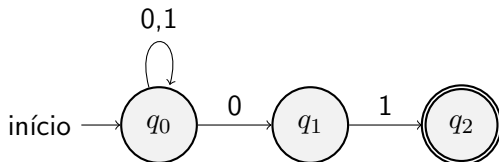
$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$$

.



Equivalência de NFAs e DFAs

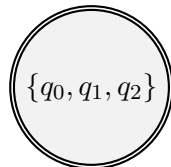
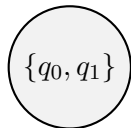
Tomando o NFA de exemplo:





Equivalência de NFAs e DFAs

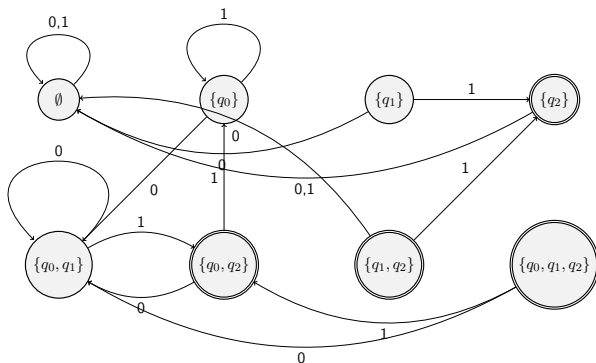
O DFA equivalente teria os seguintes estados:





Equivalência de NFAs e DFAs

Adicionando as transições, teríamos:





Sumário

4 ϵ -NFA



ϵ -NFA

- ϵ -NFAs estendem a noção de NFAs ao possibilitar uma **transição livre**, isto é, aquela que não consome um símbolo da entrada.
- Apesar dessa nova capacidade, esse formalismo não computa mais problemas que os NFAs ou DFAs. Mas pode ser utilizado convenientemente.
- Esses formalismos também tem uma proximidade muito grande com as **expressões regulares**.

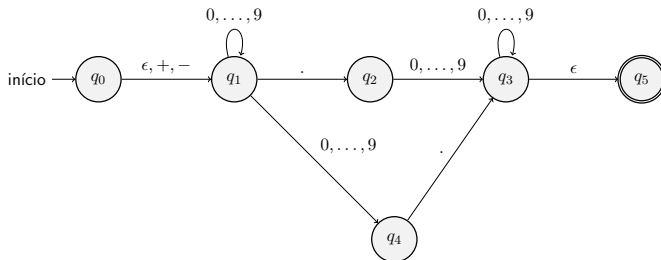


ϵ -NFAs: um exemplo

- Suponha que o objetivo seja reconhecer um número.
- Números:
 - ① Opcionalmente começam com um sinal ($-$ ou $+$).
 - ② São compostos de uma sequência de dígitos.
 - ③ Um ponto decimal.
 - ④ Outra sequência de dígitos.
- 2 e 4 são opcionais, mas pelo menos você deve ter um deles. Por exemplo $.50$ e $50.$ são ambos válidos, assim como 50.50 , ou $+50.50$ e -50.50 .



ϵ -NFAs: um exemplo





ϵ -NFAs: um exemplo

- Como modificar o ϵ -NFA anterior para aceitar números cujo uso do ponto decimal também seja opcional?



ε-NFAs: definição formal

Definição (Definição formal de um ε-NFA)

Um ε-NFA é uma 5-tupla $(Q, \Sigma, \delta, q_0, F)$, em que:

- Q é um conjunto finito de **estados**,
- Σ é o **alfabeto de entrada**,
- $\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow \mathcal{P}(Q)$ é a **função de transição**,
- $q_0 \in Q$ é o estado inicial,
- $F \subseteq Q$ é o **conjunto de estados de aceitação**.

Em relação aos NFAs, apenas a função δ muda, que passa a aceitar o símbolo ϵ .



ϵ -NFAs: conversão de ϵ -NFA para NFA

- Usar transições do tipo ϵ não adiciona poder computacional em relação aos DFAs ou NFAs.
- De fato, qualquer ϵ -NFA pode ser convertido em um NFA equivalente.
- Para mostrar isso, precisamos de algumas noções.



ϵ -NFAs: conversão de ϵ -NFA para NFA

Definição (ECLOSE)

O fecho- ϵ , chamado de ECLOSE, pode ser definido recursivamente da seguinte forma:

- $q \in \text{ECLOSE}(q)$.
- Se $p \in \text{ECLOSE}(q)$ e $r \in \delta(p, \epsilon)$, então, $r \in \text{ECLOSE}(q)$.

Em outras palavras, $\text{ECLOSE}(q)$ contém todos os estados atingíveis por q usando apenas transições ϵ



ε-NFAs: conversão de ε-NFA para NFA

Algoritmo para computar NFA equivalente ao ε-NFA

Seja $E = (Q, \Sigma_E, \delta_E, q_0, F_E)$ um ε-NFA. Construiremos um NFA $N = (Q, \Sigma_N, \delta_N, q_0, F_N)$ da seguinte forma.

- 1 Se $q \in F_E$ e $q \in \text{ECLOSE}(p)$, então $q \in F_N$.

$$F_N = \{s \in Q \mid \text{ECLOSE}(s) \cap F_E \neq \emptyset\}$$

- 2 Se $s' \in \text{ECLOSE}(s)$ e $t \in \delta_E(s', a)$, então $t \in \delta_N(s, a)$.

$$\delta_N(s, a) = \bigcup_{s' \in \text{ECLOSE}(s)} \delta(s', a)$$