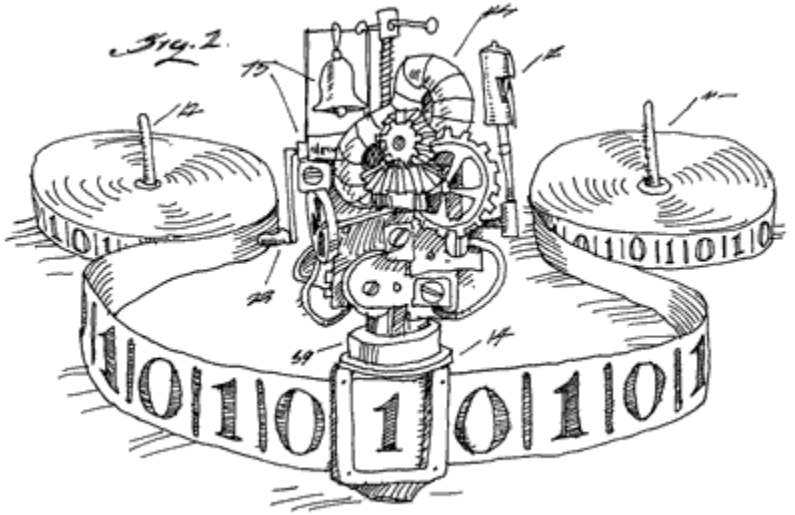
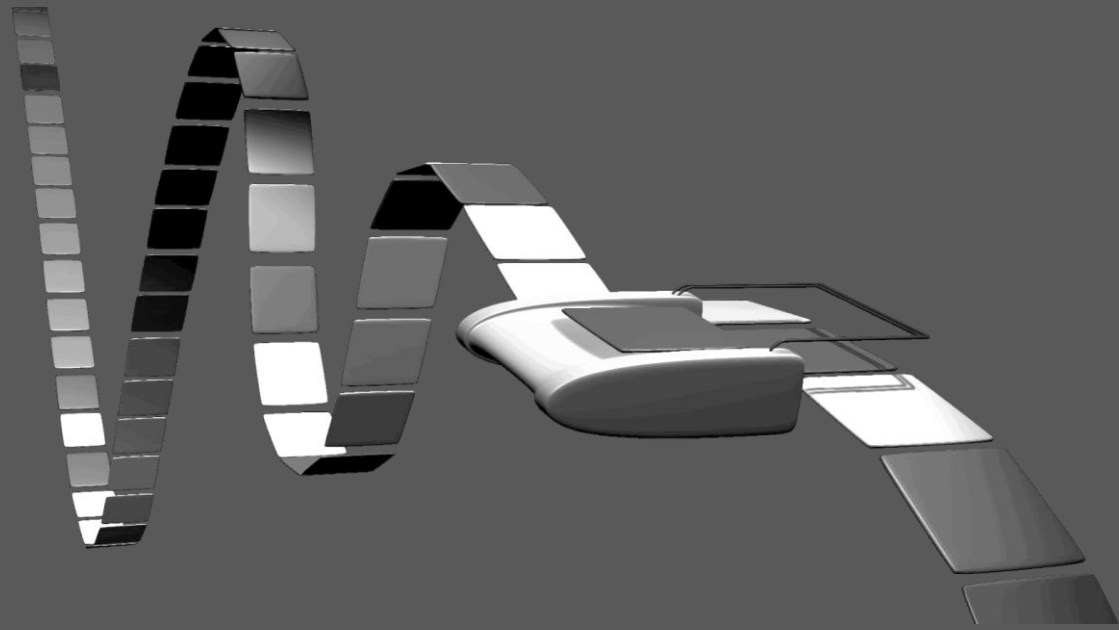
Equivalência entre AFDs e ε-AFNs (e Expressões Regulares)

Introdução à Teoria da Computação

Nicollas Mocelin Sdroievski

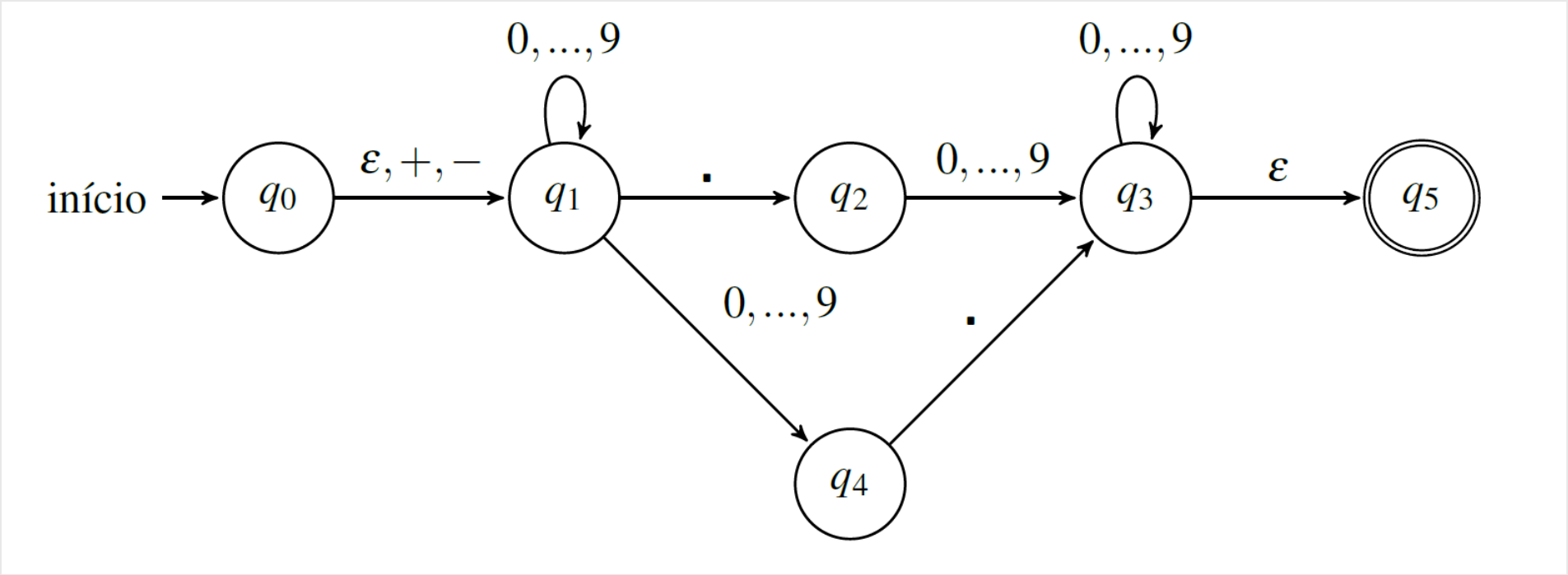


Relembrando



AFNs com transições ε

- AFNs que podem conter transições com o rótulo ε
- Permite mudanças de estado (não determinísticas) sem consumir nenhum símbolo de entrada

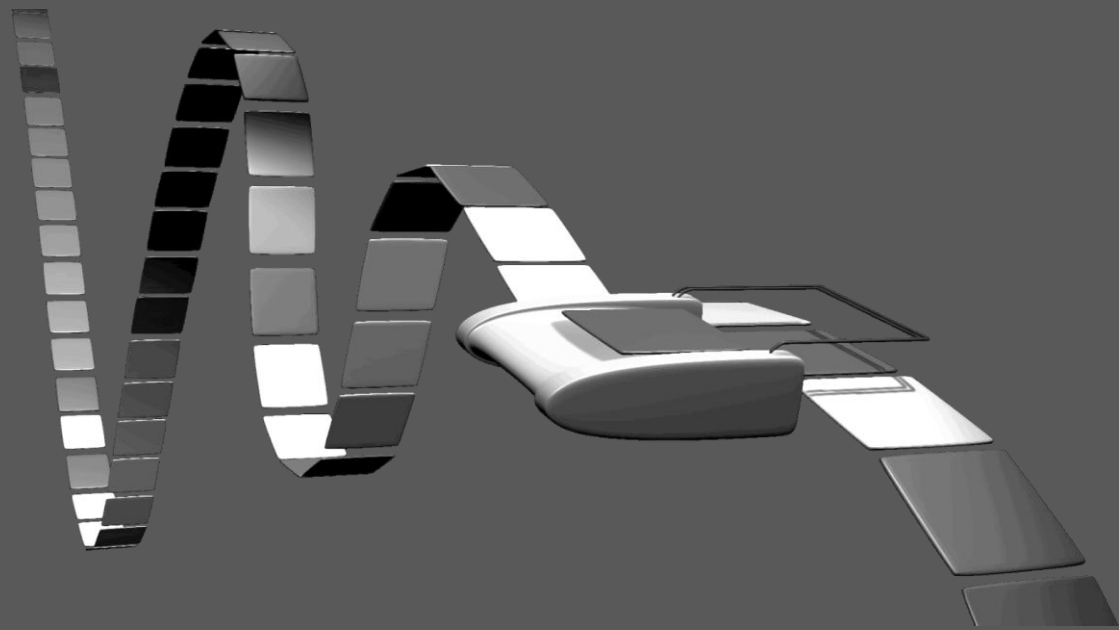


ε-fecho

- **Ideia.** O ε-Fecho de um estado q é o conjunto de todos os possíveis estados que podem ser atingidos a partir de q (incluindo o próprio q) utilizando-se uma quantidade arbitrária de transições ε

- **Definição.** Seja um ε-AFN com conjunto de estados Q e seja $q \in Q$. O conjunto ε-Fecho(q) é definido de maneira indutiva:
  - **Base:** $q \in \varepsilon$-Fecho(q)
  - **Indução:** se $p \in \varepsilon$–Fecho(q) e $r \in \delta(p, \varepsilon)$, então $r \in \varepsilon$–Fecho(q)

Equivalência



Novo algoritmo de construção de conjuntos

- **Entrada:** um ε-AFN $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$
- **Saída:** um AFD $D = (Q_D, \Sigma, \delta_D, \varepsilon\text{–Fecho}(q_0), F_D)$ tal que $L(D) = L(N)$
- **Ideia principal:** simular E com estados que são subconjuntos de $\mathcal{P}(Q_N)$, com transições para outros subconjuntos de estados
  - Agora, porém, devemos nos preocupar com as transições ε
  - O estado inicial de D é o ε-Fecho de q_0
  - Ao definir δ_D, a união é sobre o ε-Fecho de todos os estados alcançáveis com uma transição

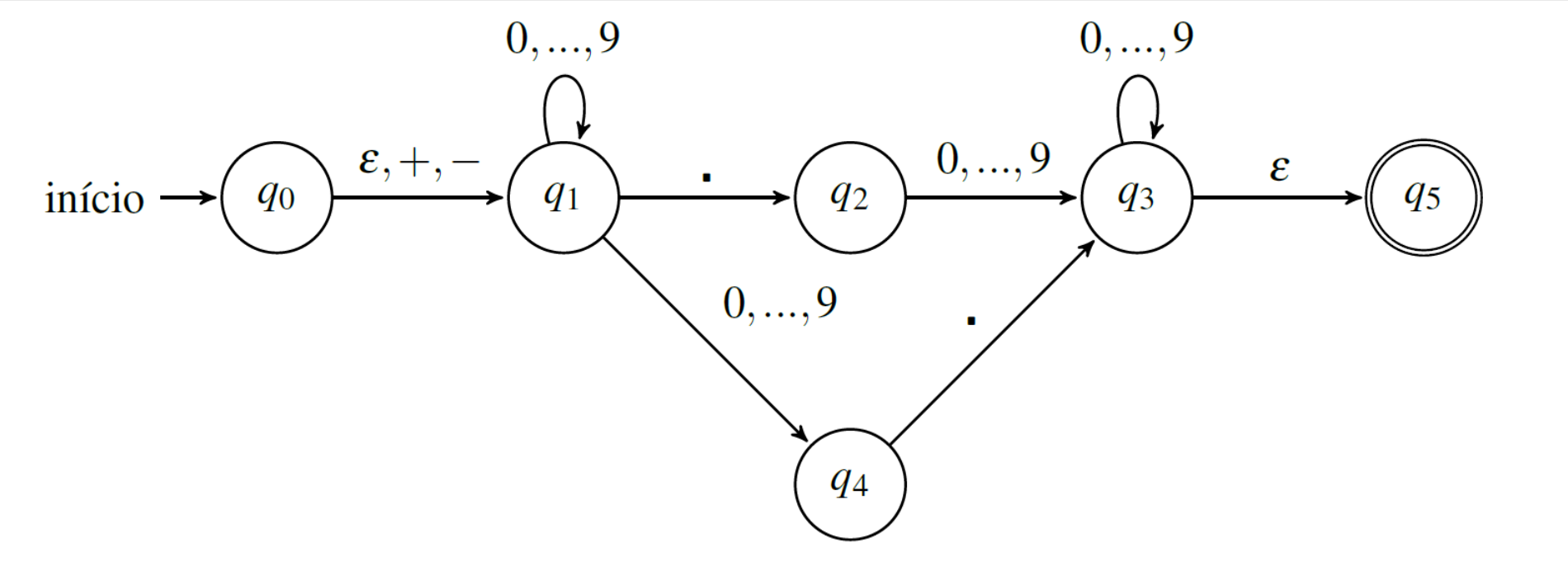
Algoritmo: definição formal

Algorithm 2 Construindo um AFD a partir de um ε-AFN

AFN_AFD $(Q_E, \Sigma, \delta_E, q_0, F_E)$

1: $Q_D = \mathscr{P}(Q_E)$
2: $q_0 = \varepsilon\text{-Fecho}(q_0)$
3: **for all** $S \subseteq Q_N$ **do**
4: **for all** $a \in \Sigma$ **do**
5: $R = \bigcup_{q_i \in S} \delta_E(q_i, a)$
6: Defina a função δ da seguinte maneira: $\delta_D(S, a) = \bigcup_{r_j \in R} \varepsilon\text{-Fecho}(r_j)$
7: $F_D = \{S \in Q_D \ : \ S \cap F_E \neq \varnothing\}$
8: Elimine os estados não alcançáveis
9: **Return** $(Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$

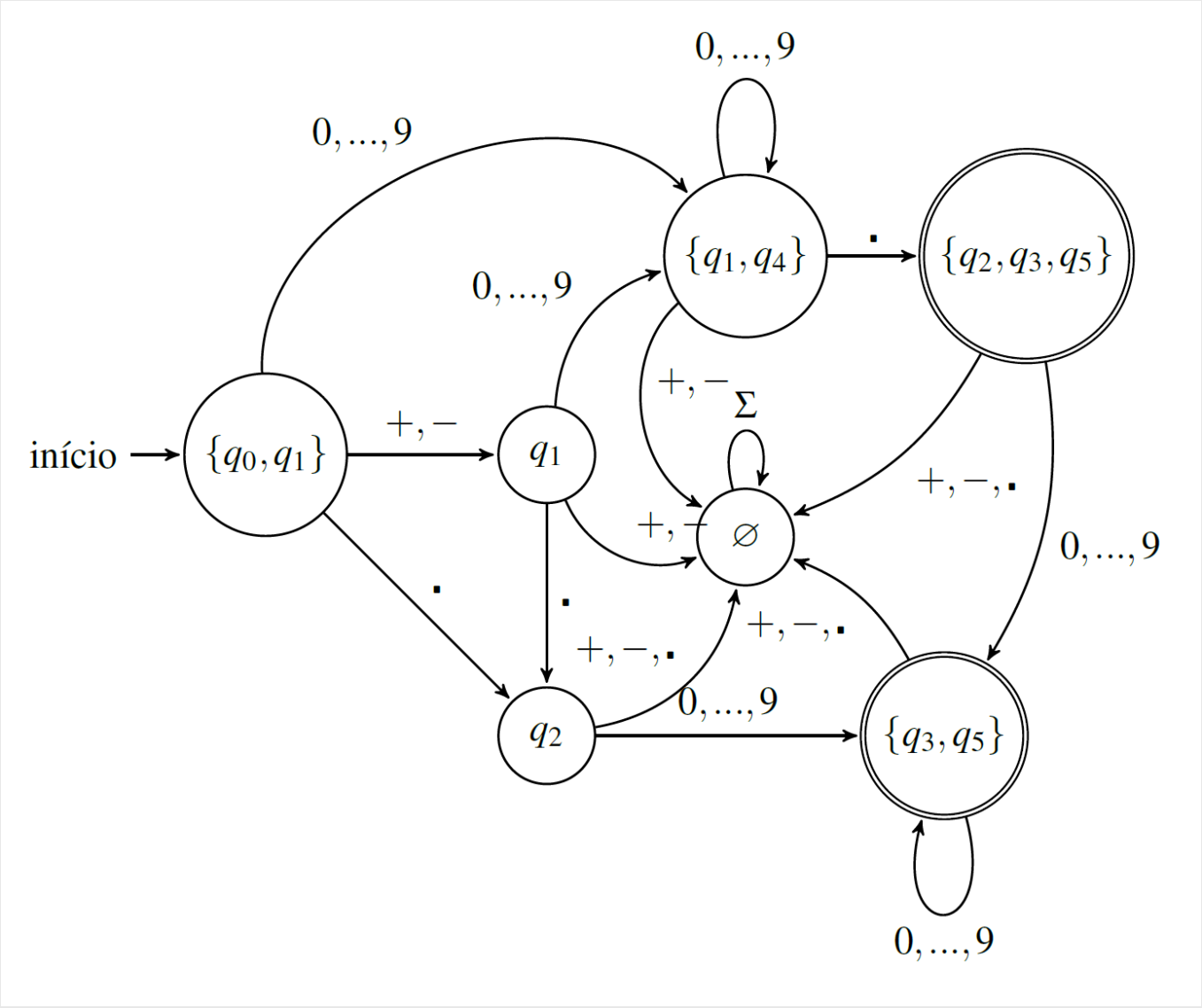
Exemplo


início
q_0
$\varepsilon, +, -$
q_1
$0, \ldots, 9$
.
q_2
$0, \ldots, 9$
q_3
$0, \ldots, 9$
ε
q_5
$0, \ldots, 9$
.
q_4

Resultado: tabela de transições

	$+, -$	$0, \ldots, 9$	.
$\rightarrow \{q_0, q_1\}$	$\{q_1\}$	$\{q_1, q_4\}$	$\{q_2\}$
$\{q_1\}$	$\varnothing$	$\{q_1, q_4\}$	$\{q_2\}$
$\{q_1, q_4\}$	$\varnothing$	$\{q_1, q_4\}$	$\{q_2, q_3, q_5\}$
$\{q_2\}$	$\varnothing$	$\{q_3, q_5\}$	$\varnothing$
$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
$*\{q_2, q_3, q_5\}$	$\varnothing$	$\{q_3, q_5\}$	$\varnothing$
$*\{q_3, q_5\}$	$\varnothing$	$\{q_3, q_5\}$	$\varnothing$

Resultado: diagrama

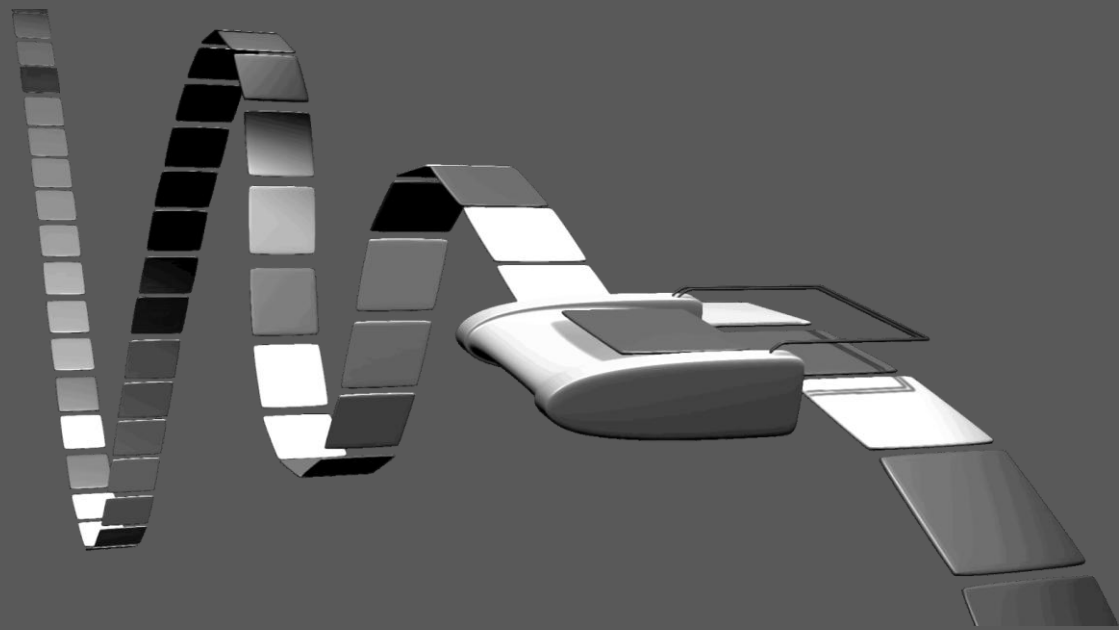


Teoremas

- **Teorema.** Se o AFD $D = (Q_D, \Sigma, \delta_D, \varepsilon\text{–Fecho}(q_0), F_D)$ é obtido pelo Algoritmo 2 a partir de um ε-AFN $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$, então $L(E) = L(D)$

- **Teorema.** Uma linguagem L é aceita por um AFD se e somente se L é aceita por um ε-AFN.

Expressões regulares



Construindo linguagens passo a passo

- Uma expressão regular é uma fórmula que indica como construir linguagens mais complicadas a partir de linguagens triviais
- **Exemplo:** $L =$ “strings que começam com 0 e tem em seguida um número arbitrário de 1’s, ou começam com 1 e tem em seguida um número arbitrário de 0’s”
- Vamos construir L passo a passo com as seguintes operações
  - Fecho
  - Concatenação
  - União

Construindo L

- Comece com as linguagens $\{0\}$ e $\{1\}$ e então

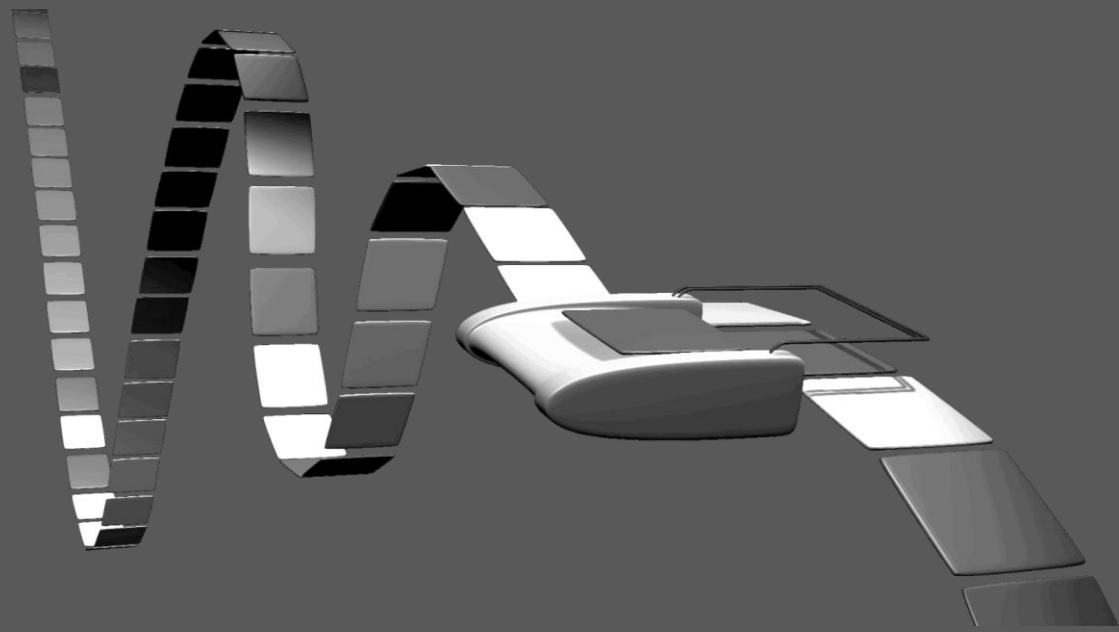
1. A partir de $\{0\}$, obtenha o **fecho** $\{0\}^* = \{\varepsilon, 0{,}00{,}000, \ldots\}$
2. A partir de $\{1\}$, obtenha o **fecho** $\{1\}^* = \{\varepsilon, 1{,}11{,}111, \ldots\}$
3. A partir de $\{0\}$ e $\{1\}^*$, obtenha a **concatenação** $\{0\} \cdot \{1\}^* = \{0{,}01{,}011, \ldots\}$
4. A partir de $\{1\}$ e $\{0\}^*$, obtenha a **concatenação** $\{1\} \cdot \{0\}^* = \{1{,}10{,}100, \ldots\}$
5. A partir de $\{0\} \cdot \{1\}^*$ e $\{1\} \cdot \{0\}^*$, obtenha a **união** $\{0{,}01{,}011, \ldots\} \cup \{1{,}10{,}100, \ldots\}$

Construindo L com uma expressão regular

- Uma expressão regular para L é $\underline{01^* + 10^*}$

1. A expressão $\underline{0^*}$ se refere ao passo 1, para obter a linguagem $\{0\}^*$
2. A expressão $\underline{1^*}$ se refere ao passo 2, para obter a linguagem $\{1\}^*$
3. A expressão $\underline{01^*}$ se refere ao passo 3, para obter a linguagem $\{0\} \cdot \{1\}^*$
4. A expressão $\underline{10^*}$ se refere ao passo 4, para obter a linguagem $\{1\} \cdot \{0\}^*$
5. A expressão $\underline{01^* + 10^*}$ se refere ao passo 5, para obter a linguagem L

Definição



Definição de expressões regulares

- **Definição.** Seja Σ um alfabeto. Uma expressão matemática consistindo de símbolos de Σ, parêntesis, '$+$' e '$*$' é uma expressão regular (ER) se tem a forma descrita recursivamente a seguir
- **Base:**
  1. $\underline{\varepsilon}$ e $\underline{\emptyset}$ são ERs que correspondem, respectivamente, às linguagens $L(\underline{\varepsilon}) = \{\varepsilon\}$ e $L(\emptyset) = \emptyset$
  2. Para todo $a \in \Sigma$, $\underline{a}$ é uma ER que corresponde a linguagem $L(\underline{a}) = \{a\}$

Definição de ERs (continuação)

Passo indutivo

1. Se E e F são ERs, então $E + F$ é uma ER que representa a união de $L(E)$ e $L(F)$. Isto é, $L(E + F) = L(E) \cup L(F)$
2. Se E e F são ERs, então EF é uma ER que representa a concatenação de $L(E)$ e $L(F)$. Isto é, $L(EF) = L(E)L(F)$
3. Se E é uma ER, então E^* é uma ER que representa o fecho de $L(E)$. Isto é, $L(E^*) = \big(L(E)\big)^*$
4. Se E é uma ER, então (E) também é uma ER, que representa a mesma linguagem de E

Terminologia e precedência

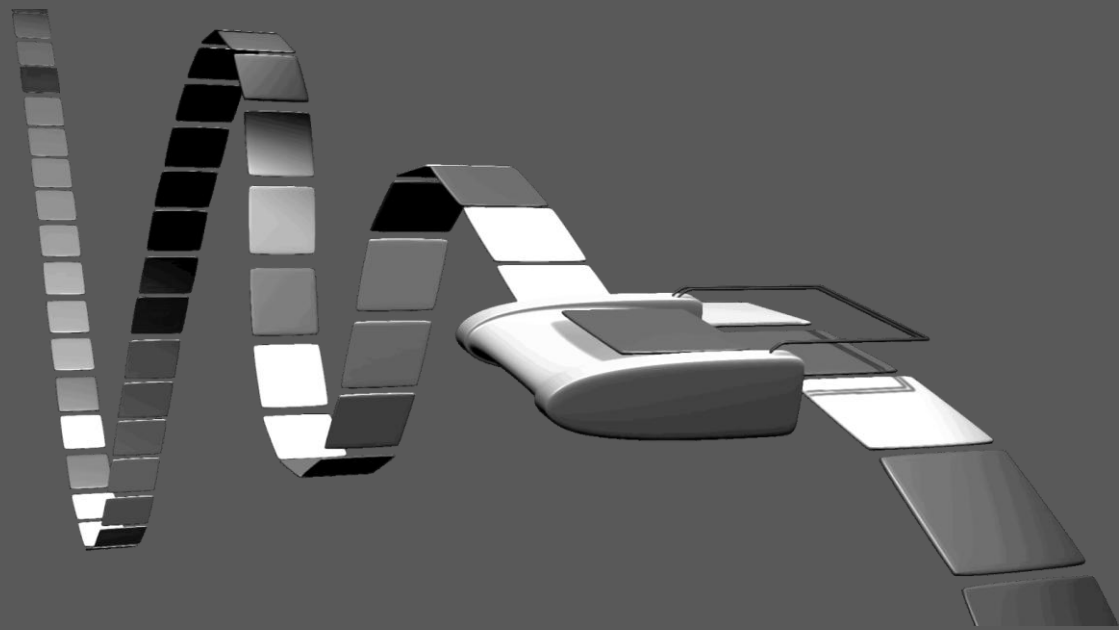
- ERs definidas pelo caso base são *elementares*, enquanto aquelas definidas pelo passo indutivo são *compostas*
- **Ordem de precedência**
    1. $*$: $\underline{01^*} = \underline{0(1)^*}$
    2. **Concatenação:** $\underline{a + bc} = \underline{a + (bc)}$
    3. $+$
- Como de costume, usamos parênteses para alterar a precedência dos operadores

Exercícios

Exercício 3.26 Seja $\Sigma = \{a, b, c\}$. Apresente a expressão regular para a linguagem das strings sobre Σ que começam e terminam com a e têm pelo menos um b. ■

Exercício 3.27 Seja $\Sigma = \{a, b\}$. Apresente a expressão regular para a linguagem das strings sobre Σ que tem tamanho ímpar. ■

Para fechar



Para fechar

- Hoje
  - Equivalência entre ε-AFNs e AFDs
    - Algoritmo similar porém com
  - Expressões regulares
    - Permitem construir linguagens mais complexas a partir de linguagens mais simples
- Próxima aula
  - Equivalência entre AFDs e expressões regulares

Referências

- [SIL25] – SILVA, M.V.G. **Autômatos, Computabilidade e Complexidade Computacional** , 2025.