- Uma E.R. define uma linguagem regular. Isto é, um conjunto de strings.

- Mas uma E.R. não reconhece se determinada string de input pertence à linguagem definida pela E.R. Para tal, necessitamos de um autómato finito.

  → é possível converter E.R. num autómato finito que reconhece as strings definidas pela E.R.

  → uma vez tendo o autómato é fácil escrever código que simula o seu comportamento.

E.R. → DFA → código

Pode parecer estranho mas é mais fácil converter a E.R. para um autómato finito não determinístico, e depois converter esse autómato para um autómato determinístico.

E.R. $\longrightarrow$ NFA $\longrightarrow$ DFA $\longrightarrow$ código

---

Nondeterministic Finite Automata (NFA)

Consiste em 5 coisas:

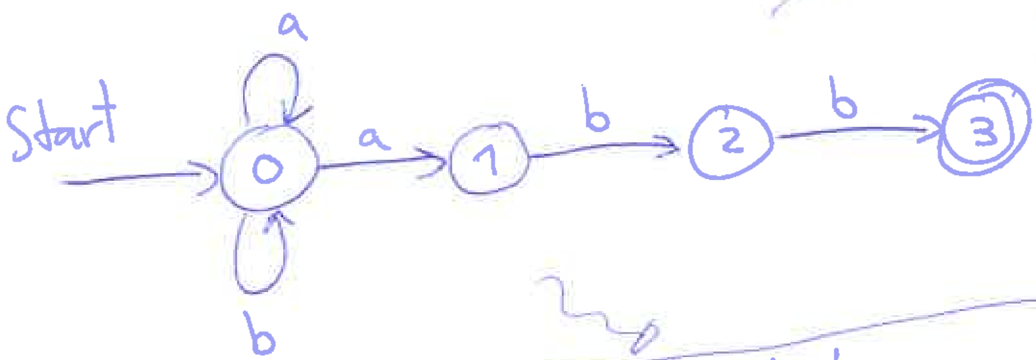
1) Conjunto finito de estados (S)
2) Alfabeto de input (Σ)
3) Função de transição (δ)
4) Estado inicial $(s_0 \in S)$
5) Conjunto de estados finais $(F \subseteq S)$

δ mapeia um (estado, símbolo) num conjunto de estados.

$\in S$ $\qquad$ $\in (\Sigma \cup \{\varepsilon\})$

string vazia

Exemplo (retirado do livro do dragão)



reconhece as strings da E.R. $(a|b)^* abb$

aceita todas as strings de a's e b's que terminam em abb.

o autómato é não determinístico porque se estivermos no estado $\underline{0}$ e o símbolo de input for $\underline{a}$, tanto podemos ficar no estado 0 como no estado 1.

Na forma de tabela:

	a	b	ε
→ 0	{0,1}	{0}	∅
1	∅	{2}	∅
2	∅	{3}	∅
* 3	∅	∅	∅

→ entradas na tabela passam a ser conjuntos de estados

Um NFA pode ter transições "espontâneas" sem consumir um símbolo de input.

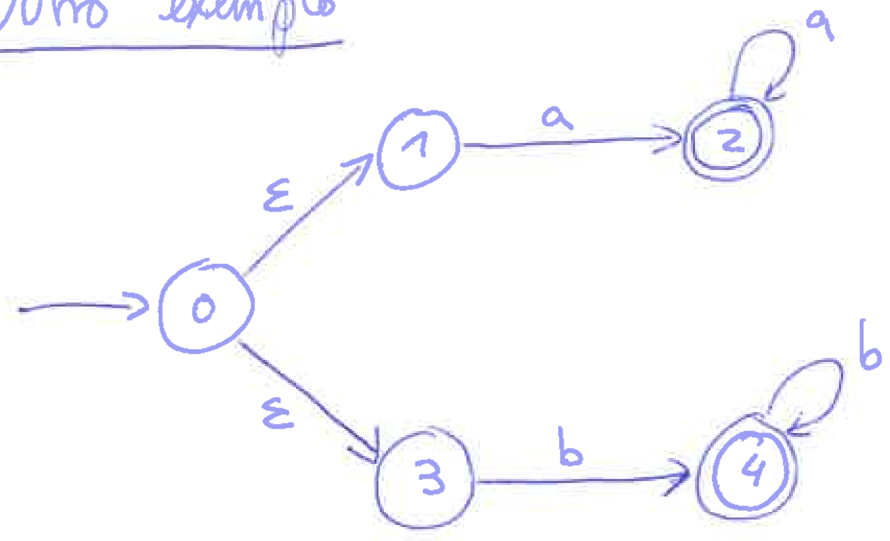
No caso deste NFA tal não acontece.

- Uma string w é aceite pelo NFA se existir um caminho desde o estado inicial até um estado final que anote a sequência de símbolos de w.

Ex: a string ab não é aceite pelo NFA dado.
a string aababb é aceite porque,

$$0 \xrightarrow{a} 0 \xrightarrow{a} 0 \xrightarrow{b} 0 \xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{b} 3$$

Outro exemplo



- ε é a string vazia.
- o autómato pode mudar do estado 0 para o estado 1 ou 3 sem consumir qualquer símbolo de input.

		a	b	ε
→	0	∅	∅	{1,3}
	1	{2}	∅	∅
*	2	{2}	∅	∅
	3	∅	{4}	∅
*	4	∅	{4}	∅

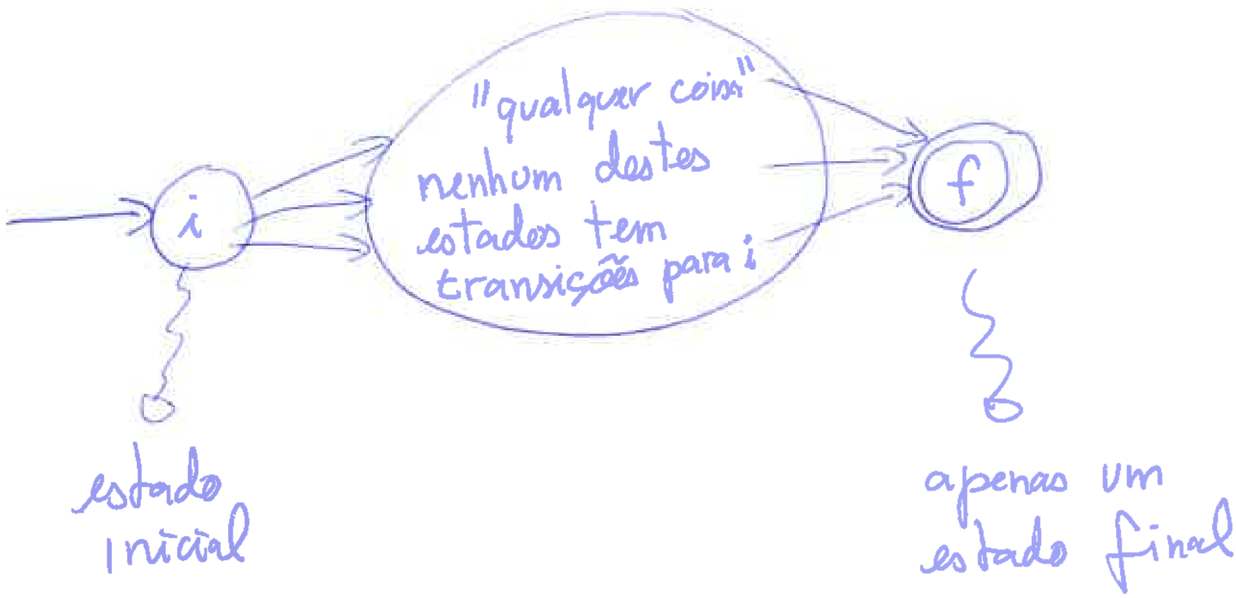
Este NFA reconhece as strings desta E.R.

$aa^* \mid bb^*$

E.R. $\longrightarrow$ NFA (construção de Thompson)

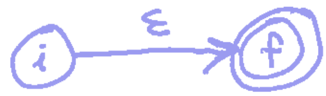
Input: E.R. r
Output: um NFA que aceita $L(r)$

O NFA é construído por indução. Vai ter esta forma



Base

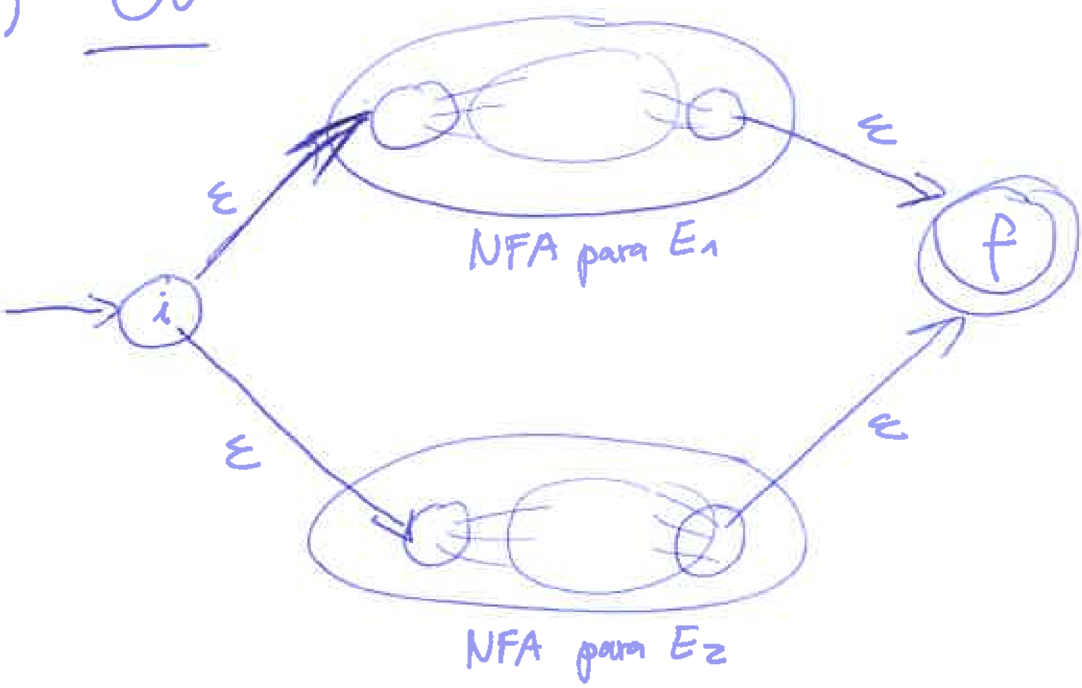
ε



símbolo a

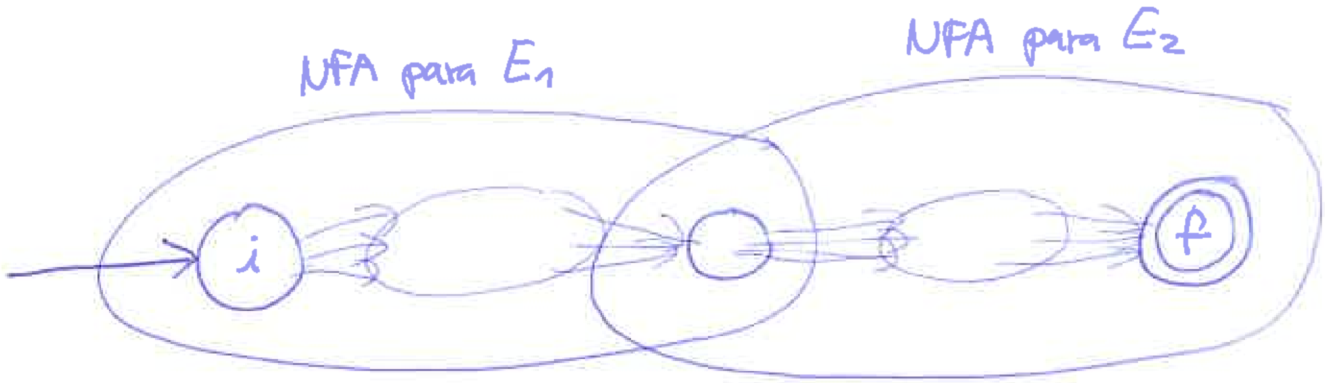
Indução

1) Ou



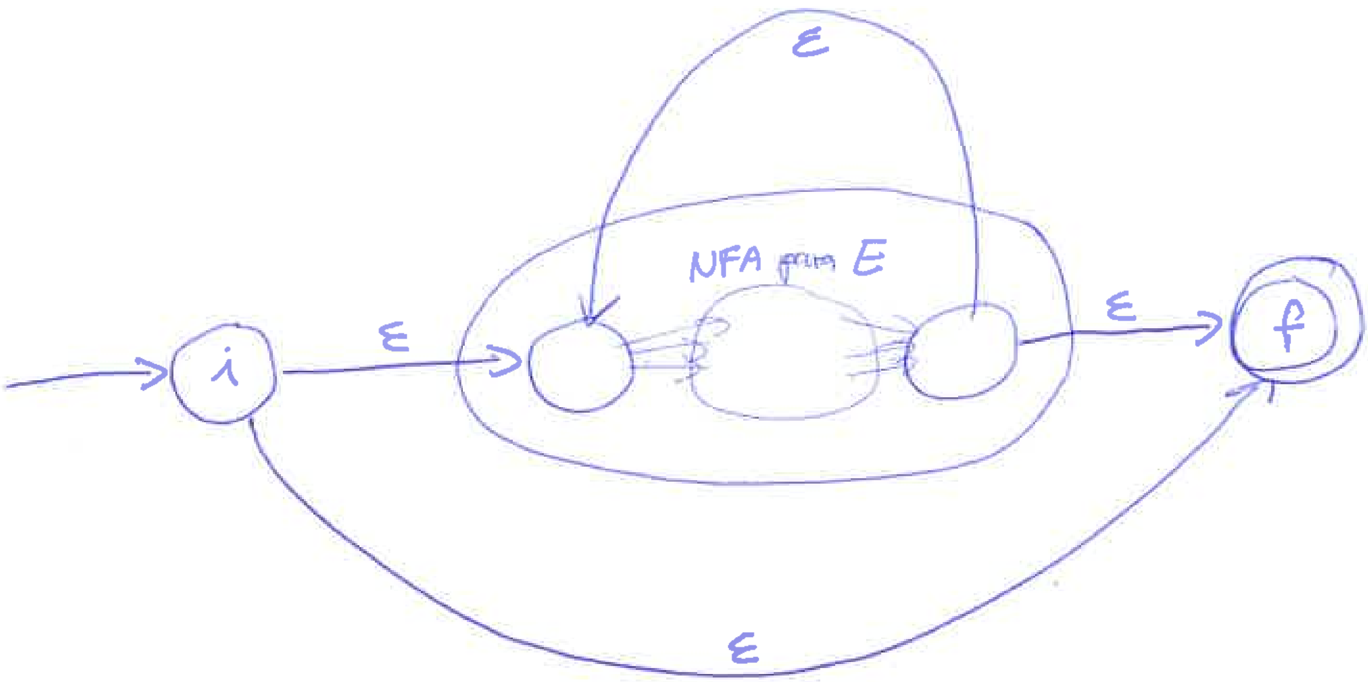
NFA para $E_1 | E_2$

2) Concatenação



NFA para E_1E_2

3) Fecho (operador *)



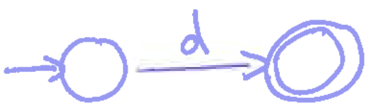
NFA para E^*

Exemplo

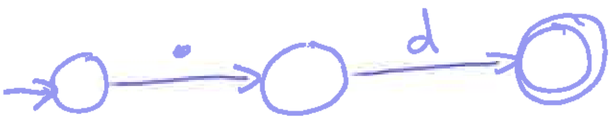
Converter a E.R. $d^*(\cdot d \mid d \cdot)\, d^*$ para um NFA.

$\cdot$

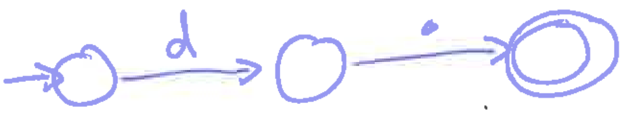
d



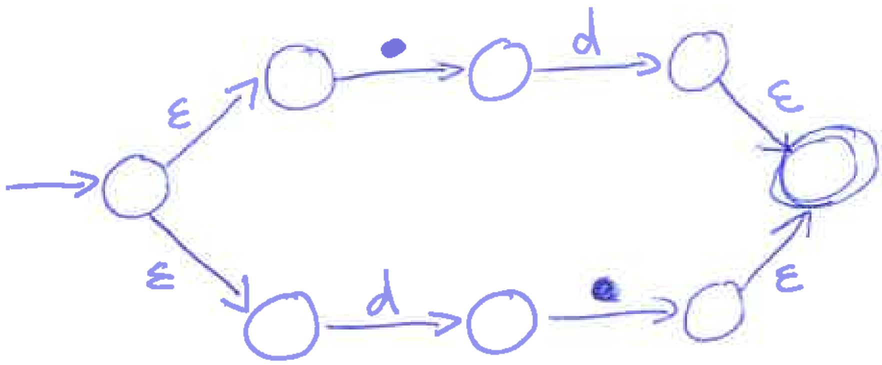
$\cdot d$



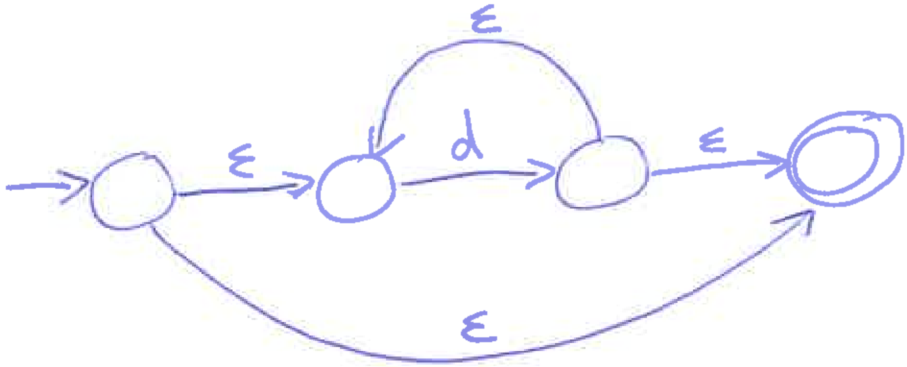
$d \cdot$

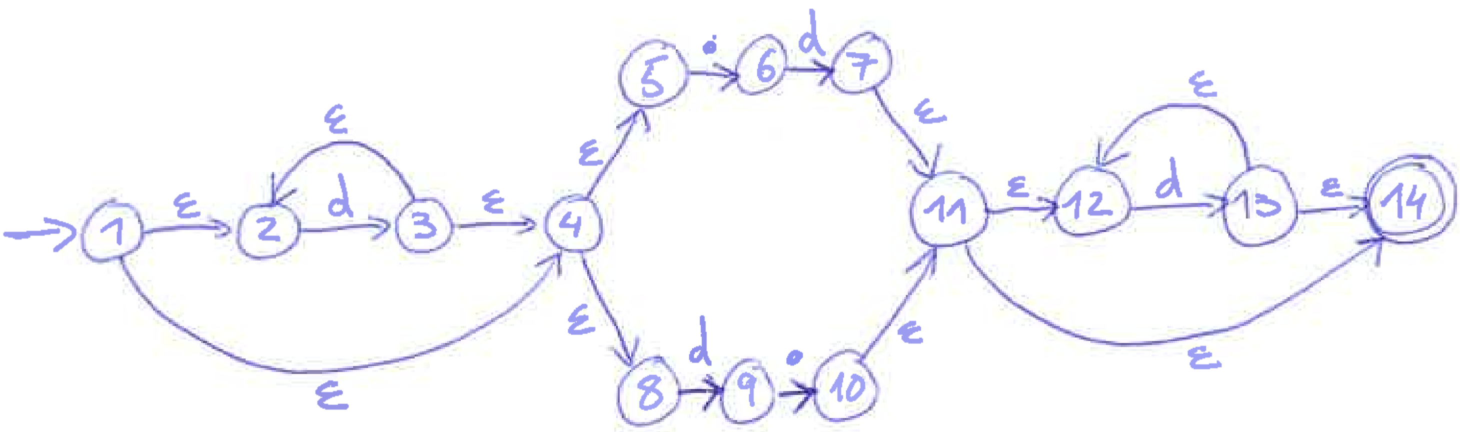


$(\cdot d \mid d \cdot)$



d^*





$$\boxed{d^*(\cdot d \mid d \cdot)\, d^*}$$