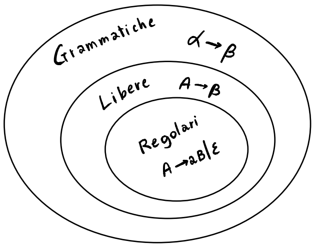
L'argomento di oggi è ricavare un NFA a partire da un'espressione regolare.
Vedremo che il linguaggio denotato dall'espressione regolare è lo stesso di quello riconosciuto dall'NFA.

RegExp
↓
NFA
↓
DFA
↓
minDFA



Dato un minDFA, possiamo ricavare l'espressione regolare, ma non lo vedremo perché è difficile.

NFA: Non-deterministic Finite-state Automata: si definiscono in maniera formale con una tupla:

$(S, A, move_n, s_0, F)$

S: insieme di stati

A: alfabeto: insieme di simboli. $\varepsilon \notin A$

s_0: uno stato $\in A$ iniziale

F: insieme di stati finali $\subseteq S$

Insieme delle parti = insieme dei sottoinsiemi

$move_n$: funzione di transizione: $S \times (A \cup \{\varepsilon\}) \rightarrow P(S)$ Dato uno stato e un elemento dell'alfabeto o ε, restituisce un insieme di stati. → Da un certo stato s ad una certa etichetta $(A \cup \{\varepsilon\})$

Convenzioni per la rappresentazione grafica delle strutture di un NFA

NFA: rappresentato da un grafo in cui:

NODI = STATI

ARCHI = $move_n$ = funzione di transizione; sono etichettati da elementi dell'insieme $A \cup \{\varepsilon\}$. Cioè, ad ogni arco, è associata un'etichetta, che è una parola dell'alfabeto o ε.

Il nodo che rappresenta S_0 si evidenzia con una piccola freccia entrante.

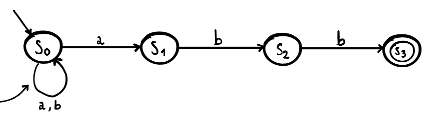
I nodi che rappresentano stati finali in F sono disegnati con un doppio cerchio.

ESEMPIO

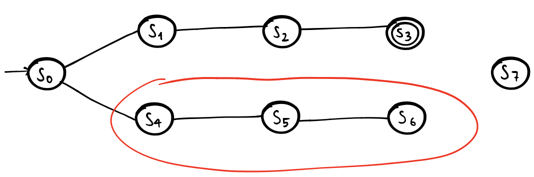
	a	b	ε
Iniziale ← S_0	$\{S_0, S_1\}$	$\{S_0\}$	$\emptyset$
S_1	$\emptyset$	$\{S_2\}$	$\emptyset$
S_2	$\emptyset$	$\{S_3\}$	$\emptyset$
S_3	$\emptyset$	$\emptyset$	$\emptyset$

In questo caso, nessuno... è ε

Più comprensibile rispetto a due frecce



$L(N) = L((a|b)^* aab)$



Non contribuisce, in quanto non raggiunge uno stato finale

Attenzione!
Non c'è scritto che tutti i cammini portano allo stato finale,
né che tutti gli stati siano raggiungibili.

L'analizzatore lessicale costruisce un'espressione regolare che denota l'insieme di tutti e quanti gli identificatori, e da questo, mediante un algoritmo (che esiste in più varianti), costruisce l'NFA che accetta il linguaggio denotato da quell'espressione regolare.

Costruzione di Thompson

Data un'espressione regolare r, costruisce un NFA N tale che L(r) = L(N).

La dimostrazione è induttiva. La costruzione è interessante, in quanto per come è organizzata, ad ogni passo si introducono al massimo due nuovi stati.

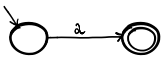
Ogni NFA intermedio ottenuto durante la costruzione ha esattamente uno stato finale, non ha archi entranti nello stato iniziale, non ha archi uscenti dallo stato finale.

Base • $r = \varepsilon$, $L(r) = \{\varepsilon\}$

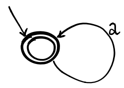
L'automa riconosce ε.

2 stati, 1 transizione

• $r = a \in A$, $L(r) = \{a\}$



E NON:

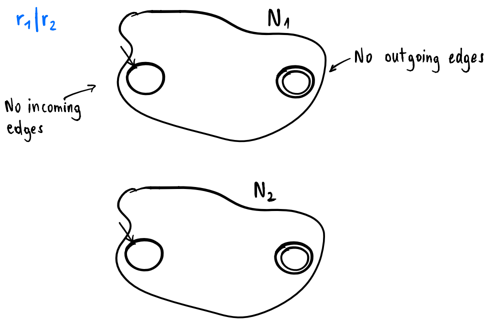


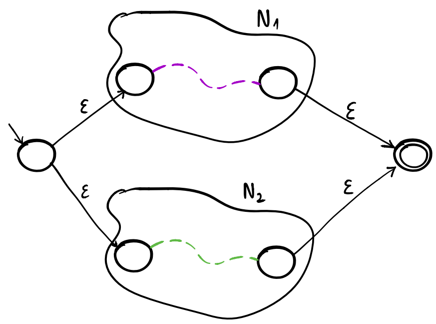
che sarebbe a^*

Passo Immagino di avere 2 automi, N_1 t.c. $L(N_1) = L(r_1)$ e N_2 t.c. $L(N_2) = L(r_2)$, dati per ipotesi induttiva

Ora vogliamo dire come costruiamo i seguenti automi:

$r_1 | r_2$





Ho introdotto 2 nuovi stati

Se $w_1 \in L(r_1)$,

$\Rightarrow \exists$ un cammino etichettato da So_1 a Sf_1

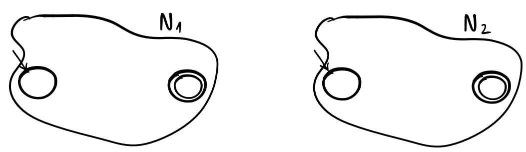
$\Rightarrow \varepsilon\, w_1\, \varepsilon$ e' lo spelling di un cammino da So a Sf

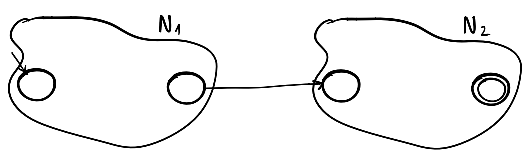
$\Rightarrow w_1 \in L(N)$

...Analogamente

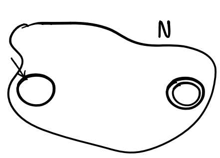
(r_1), $L((r_1))$ lo ho già per definizione

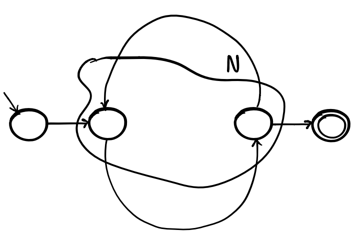
$(r_1 \cdot r_2)$, $L(r_1 \cdot r_2)$



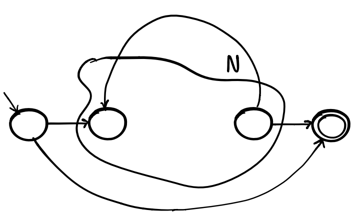


r_1^*





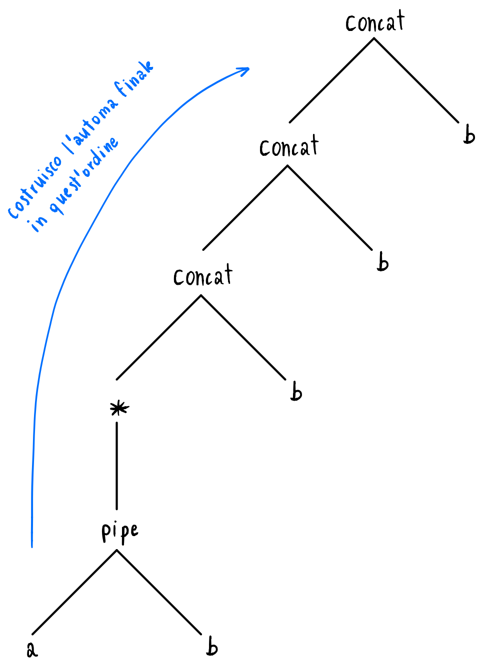
Versione di Giulia



Libro

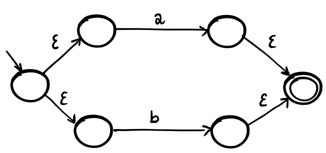
Esercizi

$(a|b)^* abb$

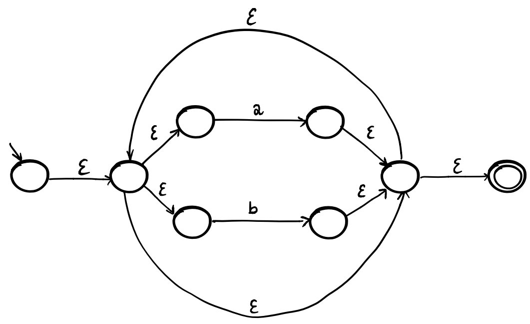


1)

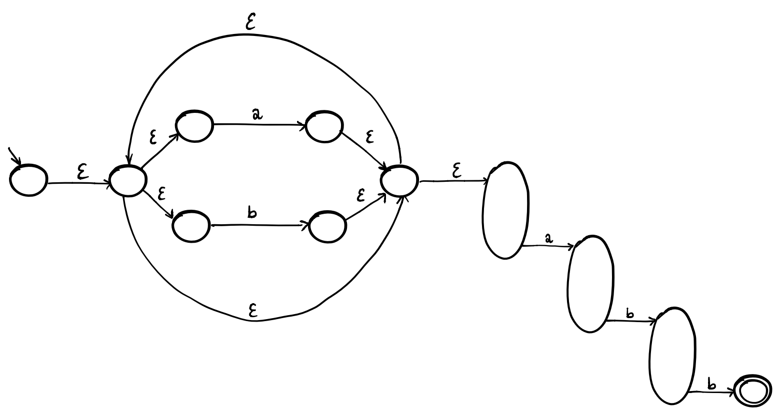
PIPE:



2) *



3) Varie concatenazioni: con a, b, b



Domande che ho fatto alla professoressa alla fine della lezione:

- Per un certo linguaggio, dobbiamo identificare l'NFA più compatto?

No, finora non ci siamo occupati di questo. Infatti, per un certo linguaggio possiamo realizzare più di un NFA.

- Ma, non dovremmo garantire che ci sia solo un modo per generare una stringa?

No, gli NFA di cui abbiamo finora parlato, ammettono potenzialmente più cammini. L'unico vincolo è che almeno uno esista.