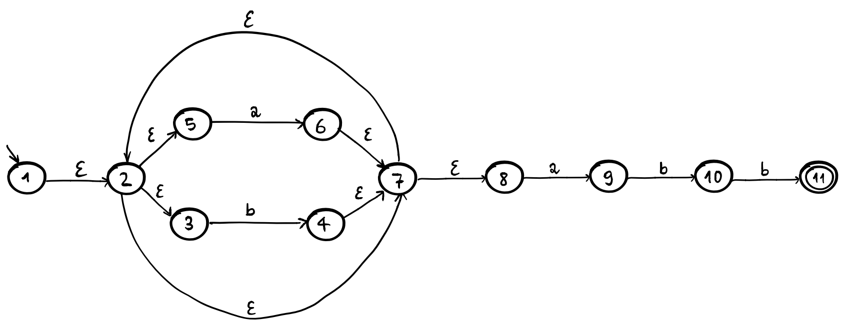
Abbiamo visto, studiando l'analisi lessicale:

1. Definizione induttiva delle espressioni regolari. $1 \rightarrow 3$: costruzioni di Thompson
2. Definizione di NFA come riconoscitore di un linguaggio denotato da un'espressione regolare.
3. Definizione di DFA, e trasformazione di NFA in DFA mediante subset construction.

Oggi un po' di pratica sulle costruzioni.

Espressione regolare $r = (a|b)^* abb$

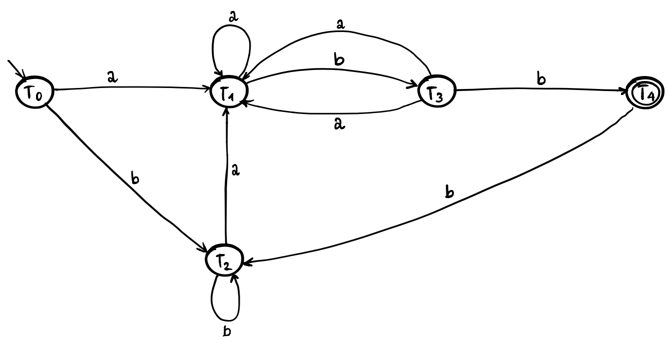
Il suo NFA:



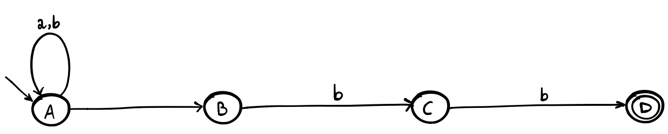
Proviamo a fare la subset construction dell'NFA per ottenere il corrispondente DFA.

	a	b
$T_0 = \{0,1,2,4,6,7\}$	T_1	T_2
$T_1 = \{3,8,6,1,2,4,7\}$	T_1	T_3
$T_2 = \{5,6,1,2,4,7\}$	T_1	T_2
$T_3 = \{5,9,6,1,2,4,7\}$	T_1	T_4
$T_4 = \{5,10,6,1,2,4,7\}$	T_1	T_2

Disegnamo il DFA:



Vediamo un NFA alternativo per lo stesso linguaggio:



Domanda: "È normale che l'NFA sia più piccolo in memoria rispetto al DFA?". Memoria = quantità di nodi e di archi.

Proprietà del DFA minimo

Sia D un DFA con funzione di transizione totale (un DFA che ha funzione di transizione totale è, ad esempio, quello dell'esercizio prima proposto).

Abbiamo min(D) = subset(reverse(subset(reverse(D)))).

Per ottenere il reverse DFA è sufficiente, detto grezzamente, invertire ogni arco; inoltre, lo stato iniziale diventa quello finale, e quello finale diventa quello iniziale.

Un linguaggio L si dice *regolare* se: (4 condizioni **equivalenti**).
esiste un'espressione regolare `r` t.c. L = L(`r`)
⇔ esiste un NFA N | L = L(N)

⇔ esiste un DFA D t.c. L = L(D) ⇔

esiste una grammatica G regolare t.c. L = L(G).

Ricordiamo che una grammatica è regolare se è libera (le produzioni sono del tipo `A -> aB`, o `A -> ε`).

Sostanzialmente, abbiamo 4 armi per dimostrare che un certo linguaggio è regolare.

Esercizio
{`w` | `w` è una stringa sull'alfabeto {`a`, `b`}, e `b` occorre un numero dispari di volte }
Soluzione: vedi lezione successiva.