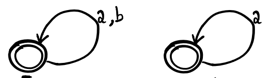
Nella scorsa lezione abbiamo visto l'NFA. L'NFA, data una certa parola, supporta un algoritmo la cui esecuzione specifica se una certa parola è riconosciuta da quell'NFA, e quindi se appartiene al linguaggio denotato dalla regular expression.

Una struttura alternativa all'NFA, ma equivalente, è il DFA, che sta per Deterministic Final-state Automaton. In che senso i DFA sono un sottoinsieme degli NFA?

- Innanzitutto, non ci sono ε-transizioni.
- La funzione di transizione può essere totale o parziale (questa differenza non è sempre messa in evidenza dai libri, come il "dragon book").

Funzione di transizione *totale*: dato un qualunque stato dell'automa che stiamo considerando, e dato uno qualunque dei simboli dell'alfabeto, la funzione di transizione è sempre definita; ossia c'è sempre una transizione uscente etichettata da quel simbolo.
Parziale: la funzione di transizione non è necessariamente definita per tutti i simboli.

$A = \{a, b\}$



TOTALE PARZIALE

Nuovamente:
Se il DFA ha funzione di transizione totale, allora, per ogni stato `s` dell'automa, e per ogni simbolo `b` dell'alfabeto, c'è esattamente una transizione uscente da `s` ed etichettata `b`.
Se ha funzione di transizione parziale, allora, per ogni stato s dell'automa, e per ogni simbolo b dell'alfabeto, c'è al più una transizione uscente da s ed etichettata b.

Definizione formale di DFA: tupla $(S, A, move_d, s_0, F)$

S: insieme degli stati

A: alfabeto: insieme dei simboli per cui avremo transizioni

$move_d$: funzione di transizione, cioè che, dato un elemento dell'alfabeto restituisce uno stato. $S \times A \longrightarrow S$

funzione "parziale"
notazione per noi superflua

s_0: stato finale, $\in S$.

F: insieme degli stati finali, $\subseteq S$.

Le ε transizioni sono in realtà ammesse per il seguente corner-case:

- $\varepsilon \in L(D)$ se $s_0 \in F$ (se s_0 è finale, allora $\varepsilon \in$ linguaggio).

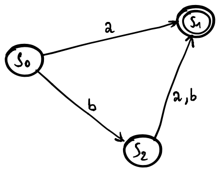
Oltretutto abbiamo:

- $w \neq \varepsilon \in L(D)$ se $\exists!$ un cammino $x_1 \dots x_k = w$ da s_0 a uno stato $s \in F$

se esiste, è unico (struttura totalmente deterministica → non più come l'NFA)

Convenzione: se $move_d$ non è definita sul punto (s, b), ossia ho uno stato s da cui non c'è una transizione uscente etichettata b, scriviamo $move_d(s, b) = \perp$ ("bottom")

SIMULAZIONE DFA CON FUNZIONE DI TRANSIZIONE PARZIALE



	a	b
s_0	s_1	s_3
s_1	$\perp$	$\perp$
s_2	s_1	s_1

ALGORITMO:

INPUT: $w\$$, DFA $D = (S, A, move_d, s_0, F)$

OUTPUT: sì/no alla domanda "$w \in L(D)$"

```
state = s0                                  % cominciamo posizionandoci sullo stato iniziale dell'automa
symbol = nextchar()                         % punta al primo elemento di w$
while (symbol = $) && (state ≠ ⊥) {         % se la funzione di transizione è totale, omettiamo questo pezzo.
        state = move_d(state, symbol)           Totale = definita in tutti i punti, il bottom è completamente inutile.
        symbol = nextchar()
}
if (state ∈ F) return YES; else return NO   % siamo arrivati allo stato finale?
```

Costi:

- Simulazione NFA: O(|w| · (n+m))
- Simulazione DFA: O(w)

Ma quanto costa passare dall'NFA al DFA equivalente (che riconosce lo stesso linguaggio)? Idea: utilizzare la ε-chiusura introdotta nella simulazione di NFA per far corrispondere sottoinsiemi di stati di NFA ad un unico stato del DFA.

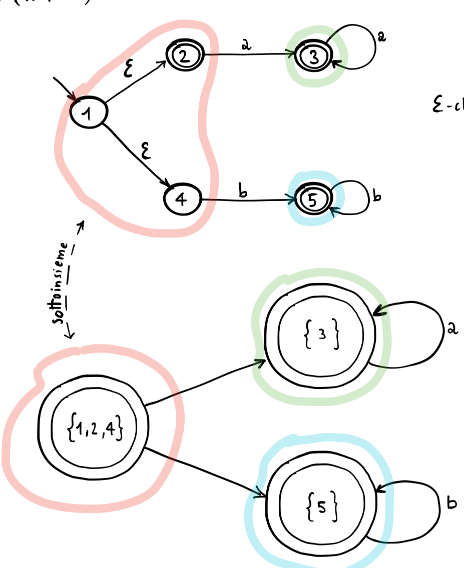
ALGORITMO DI SUBSET CONSTRUCTION (c'è sempre nel compito!)

INPUT: NFA = $(S^n, A, move_n, s_0^n, F^n)$ $n :=$ "non-deterministico"

TPUT: DFA = $(S^d, A, move_d, s_0^d, F^d)$ $d :=$ "deterministico"

```
s0^d = ε-closure({s0^n})
S^d = {s0^d}
flag s0^d come "non marcato"
while (∃ t ∈ S^d t.c. t non è marcato) {
        flag t come "marcato"
        foreach (b ∈ A) {
                t' = ε-closure(∪_{ti∈t} move_n(ti, b))
                if (t' ≠ ∅) {
                        move_d(t, b) = t'
                        if (t' ∉ S^d) {
                                aggiungi t' a S^d
                                flag t' come "non marcato"
                        }
                }
        }
}
foreach (t ∈ S^d) {
        if (t ∩ F^n) ≠ ∅ then t ∈ F^d
}
```

ESEMPIO (RAPIDO)

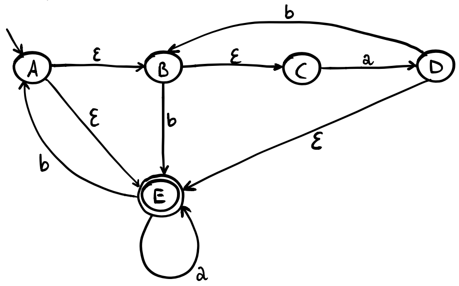


Scrivo direttamente la tabella di transizione:

ε-closure di {1}

STATI \ ELEMENTI DI A	a	b
iniziale, finale {1,2,4} ✓	{3} = t'	{5}
finale {3} ✓	{3}	∅
finale {5} ✓	∅	{5}

Esempio (esteso)



È la ε-chiusura dello stato iniziale non-deterministico che mi dà lo stato iniziale dell'automa deterministico

ε-closure di {A}

La tabella di transizione sarà così fatta:

ELEMENTI DI A / STATI	a	b
iniziale {A, B, E, C}		

Non è marcato

↓↓

① a-transizioni:

ELEMENTI DI A / STATI	a	b
{A, B, E, C} ✓	{E, D}	

~~a~~ ~~a~~ a a

E D

{E, D}

Non vado da nessuna parte

La move_d sarà definita da uno stato che contiene la ε-chiusura di E e di D. Quale sarà questo stato? Con la ε posso andare comunque solo in E e in D.

② Devo considerare le b-transizioni.

ELEMENTI DI A / STATI	a	b
{A, B, E, C} ✓	{E, D}	{E, A, B, C}
{E, D}		

~~b~~ b b ~~b~~

E A

La ε-chiusura di E e di A

Questo insieme lo ho già nella collezione → non lo aggiungo

③ Il while è finito; torno a vedere se c'è uno stato che non è marcato. Sì, è {E, D}

Per lui devo vedere quali sono i possibili move_d rispetto alla a-transizione.

ELEMENTI DI A / STATI	a	b
{A, B, E, C} ✓	{E, D}	{E, A, B, C}
{E, D} ✓	{E}	
{E}		

a ~~a~~

E

ε-chiusura di {E}, che non aggiunge nulla.

L'insieme che contiene solo la E non lo ho ancora elencato: lo aggiungo alla collezione.

④ b-Transizioni

ELEMENTI DI A / STATI	a	b
{A, B, E, C} ✓	{E, D}	{E, A, B, C}
{E, D} ✓	{E}	{A, B, C, E}
{E}		

b b

A B

ε-chiusura di {A, B}, che già ho e, non aggiungo.

⑤

ELEMENTI DI A / STATI	a	b
{A, B, E, C} ✓	{E, D}	{E, A, B, C}
{E, D} ✓	{E}	{A, B, C, E}
{E} ✓	{E}	

a

E

⑥

ELEMENTI DI A / STATI	a	b
{A, B, E, C} ✓	{E, D}	{E, A, B, C}
{E, D} ✓	{E}	{A, B, C, E}
{E} ✓	{E}	{A, B, C, E}

b

E A

Mi resta da determinare quali saranno stati finali nel DFA. Nell'NFA l'unico stato finale è `E`. Nel DFA, tutti i sottoinsiemi elencati che contengono la `E` sono finali.

DFA: