

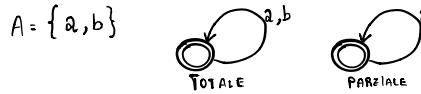
Nella scorsa lezione abbiamo visto l'NFA. L'NFA, data una certa parola, supporta un algoritmo la cui esecuzione specifica se una certa parola è riconosciuta da quell'NFA, e quindi se appartiene al linguaggio denotato dalla regular expression.

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



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 $(S, A, \text{move}_d, s_0, F)$

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. $S \times A \rightarrow S$

s_0 : stato finale, $\in S$.

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

funzione "parziale"
notazione per noi superflua
↓

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

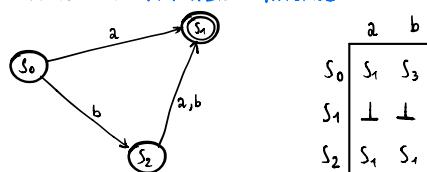
- $\epsilon \in L(D)$ se $s_0 \in F$ (se s_0 è finale, allora $\epsilon \in$ linguaggio).

Oltre tutto abbiamo:

- $w \neq \epsilon \in L(D)$ 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 \rightarrow 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 $\text{move}_d(s, b) = \perp$ ("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 $D = (S, A, \text{move}_d, s_0, F)$

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

1. state = s_0
 2. symbol = nextchar()
 3. while (symbol = \$) && (state ≠ \perp) {
 4. state = move_d (state, symbol)
 5. symbol = nextchar()
 6. }
 7. if (state ∈ F) return YES; else return NO
- % cominciamo posizionandoci sullo stato iniziale dell'automa
% punta al primo elemento di $w \$$
% se la funzione di transizione è totale, omettiamo questo pezzo.
Totale = definita in tutti i punti, il bottom è completamente inutile.
- % siamo arrivati allo stato finale ?

Costi:

- Simulazione NFA: $O(|w| \cdot (n+m))$
- 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!)

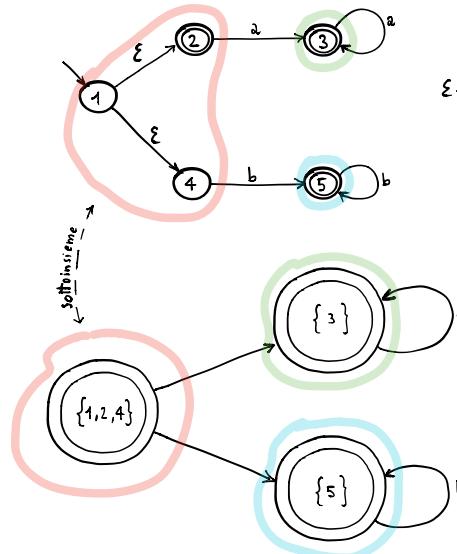
INPUT: NFA = $(S^n, A, move_n, s_0^n, F^n)$ $n :=$ "non-deterministico"
 INPUT: DFA = $(S^d, A, move_d, s_0^d, F^d)$ $d :=$ "deterministico"

```

1    $s_0^d = \epsilon\text{-closure}(\{s_0^n\})$ 
2    $S^d = \{s_0^d\}$ 
3   flag  $s_0^d$  come "non marcato"
4   while ( $\exists t \in S^d$  t.c.  $t$  non è marcato) {
5       flag  $t$  come "marcato"
6       foreach ( $b \in A$ ) {
7            $t' = \epsilon\text{-closure}(\bigcup_{t_i \in t} move_n(t_i, b))$ 
8           if ( $t' \neq \emptyset$ ) {
9                $move_d(t, b) = t'$ 
10              if ( $t' \notin S^d$ ) {
11                  aggiungi  $t'$  a  $S^d$ 
12                  flag  $t'$  come "non marcato"
13              }
14          }
15      }
16  }
17  foreach ( $t \in S^d$ ) {
18      if ( $t \cap F^n \neq \emptyset$  then  $t \in F^d$ 
19  }

```

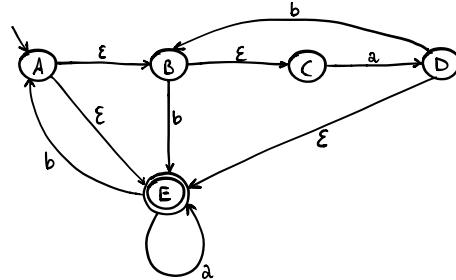
ESEMPIO (RAPIDO)



Scrivo direttamente la tabella di transizione:

ϵ -closure di $\{1\}$	ELEMENTI DI A	
	a	b
iniziale $\{1,2,4\}$	$\{3\} = t'$	$\{5\}$
finale $\{3\}$	$\{3\}$	\emptyset
finale $\{5\}$	\emptyset	$\{5\}$

ESEMPIO (ESTESO)



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

La tabella di transizione sarà così fatta:

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

↓
Non è marcato

① a-transizioni:

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

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

Non vado da nessuna parte

② 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}		

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

L'insieme che contiene solo la E non lo ho ancora elencato:

lo aggiungo alla collezione.

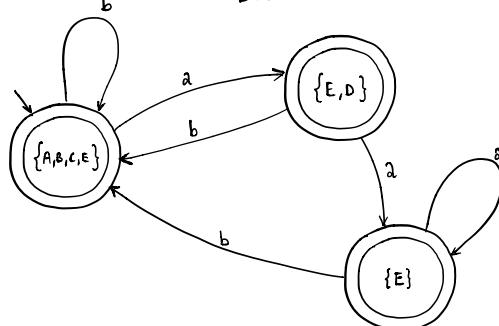
⑤

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

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:



② Devo considerare le b-transizioni.

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

La ϵ -chiusura di E e di A
Questo insieme lo ho già nella collezione → non lo aggiungo

④ b-Transizioni

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

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