Azzolini Riccardo 2020-10-22

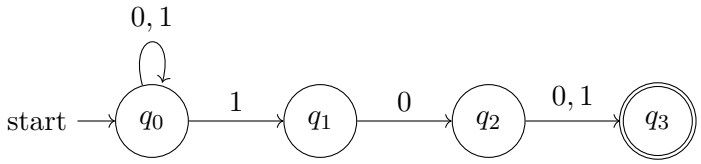
Costruzione per sottoinsiemi

1 Costruzione di un DFA dato un NFA

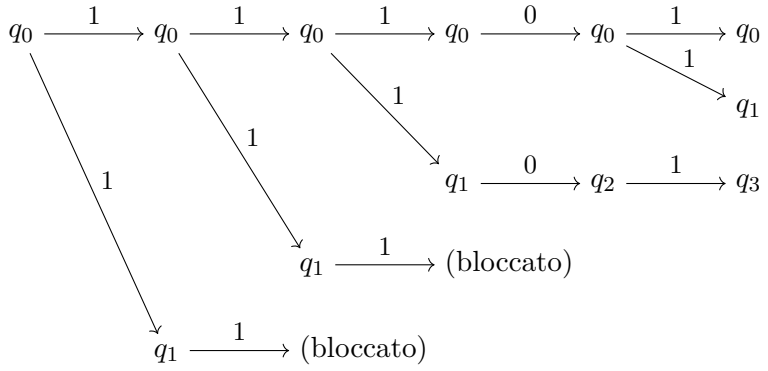
Si è già visto che, dato un DFA, è possibile costruire un NFA che riconosca lo stesso linguaggio. Adesso l'obiettivo è fare il viceversa: dato un NFA $N = \langle Q, \Sigma, \delta, q_0, F \rangle$, si vuole costruire un DFA D_N che riconosca lo stesso linguaggio riconosciuto da N.

L'idea è di costruire un DFA che simuli contemporaneamente tutte le possibili computazioni di N, usando una tecnica chiamata **costruzione per sottoinsiemi**.

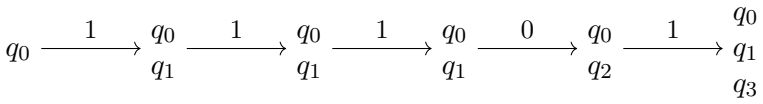
Ad esempio, considerando l'NFA rappresentato dal diagramma



il suo albero di computazione sulla stringa $w = 11101$ (già mostrato come esempio in precedenza) è:



Per "codificare" questa computazione in un DFA, si definiscono degli stati aventi una struttura interna più complessa rispetto a quelli visti finora, dando a ciascuno di essi un nome strutturato che mette insieme tutti gli stati di un livello dell'albero:



Per ogni transizione del DFA:

1. si considerano le possibili transizioni dell'NFA a partire da ciascuno degli stati q_i "impacchettati" nel nome dello stato del DFA;
2. si raccolgono tutti stati di arrivo dell'NFA nel nome dello stato di arrivo del DFA.

Questo esempio è stato fatto seguendo una specifica computazione dell'NFA. Adesso, invece, bisognerà dare una costruzione generale, che consideri tutte le possibili evoluzioni dell'automa. Tale costruzione, come già anticipato, si chiama *costruzione per sottoinsiemi* in quanto, come si vede dall'esempio, i nomi degli stati del DFA sono sottoinsiemi dell'insieme Q di stati dell'NFA. Una volta stabiliti gli stati, questa costruzione dovrà poi considerare come definire la funzione di transizione e la condizione di accettazione del DFA: la soluzione per entrambi questi aspetti sarà essenzialmente ragionare sui singoli stati dell'NFA impacchettati nei nomi degli stati del DFA.

2 Costruzione per sottoinsiemi

Dato un generico NFA $N = \langle Q, \Sigma, \delta, q_0, F\rangle$, si costruisce il DFA $D_N = \langle Q_D, \Sigma, \delta_D, \{q_0\}, F_D\rangle$, dove:

- L'alfabeto di input di D_N è lo stesso dell'NFA N (mentre tutte le altre componenti dell'automa cambiano), cosa necessaria perché i due possano riconoscere il medesimo linguaggio.
- L'insieme degli stati è $Q_D = 2^Q$, cioè l'insieme di tutti i sottoinsiemi di Q, ovvero di tutti i possibili sottoinsiemi di stati dell'NFA. Infatti, per la definizione di DFA l'unica cosa che importa degli stati è la struttura che la funzione di transizione definisce su di essi; i loro nomi possono essere fatti in qualunque modo, e qui in particolare sono insiemi, sottoinsiemi di Q (ad esempio, potrebbe esserci uno stato chiamato $\{q_0, q_1, q_3\}$).
- Il nome dello stato iniziale è l'insieme che contiene il solo stato iniziale q_0 dell'NFA: intuitivamente, osservando l'esempio precedente, si ha che le computazioni dell'NFA e del DFA partono di fatto dallo stesso stato, ma i nomi degli stati del DFA devono essere insiemi, quindi non si può mettere come stato iniziale direttamente q_0, ma piuttosto bisogna mettere $\{q_0\}$ (che va bene perché è appunto un sottoinsieme di Q).

- L'insieme degli stati finali è

$$F_D = \{S \in Q_D \mid S \cap F \neq \varnothing\}$$

cioè comprende tutti e soli gli stati del DFA i cui nomi contengono almeno uno stato finale dell'NFA.

- La funzione di transizione, per la definizione di DFA, ha la forma $\delta_D : Q_D \times \Sigma \to Q_D$, cioè prende in input
  - un insieme di stati dell'NFA, $S \in Q_D$ (ovvero $S \subseteq Q$),
  - un simbolo dell'alfabeto, $a \in \Sigma$,

  e restituisce ancora un insieme di stati dell'NFA. Formalmente:

$$\delta_D(S, a) = \bigcup_{p \in S} \delta(p, a)$$

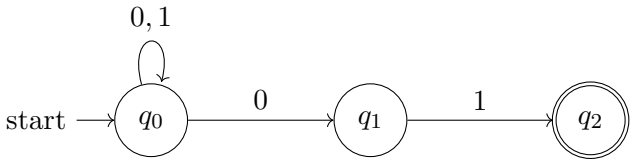
  Ad esempio, se $S = \{q_1, q_2\}$, e se la funzione di transizione dell'NFA stabilisce che $\delta(q_1, a) = \{p_1, p_2\}$ e $\delta(q_2, a) = \{p_3\}$, allora:

$$\delta_D(S, a) = \bigcup_{p \in \{q_1, q_2\}} \delta(p, a) = \delta(q_1, a) \cup \delta(q_2, a) = \{p_1, p_2\} \cup \{p_3\} = \{p_1, p_2, p_3\}$$

Come già osservato nel caso del passaggio da un DFA a un NFA, quella definita per ora è solo una tecnica di costruzione di un DFA D_N dato un NFA N; bisognerà poi dimostrare che N e D_N riconoscono effettivamente lo stesso linguaggio.

2.1 Esempio

Si consideri l'NFA N rappresentato dal diagramma



o, equivalentemente, dalla tabella

δ	0	1
$\to q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
$*q_2$	$\varnothing$	$\varnothing$

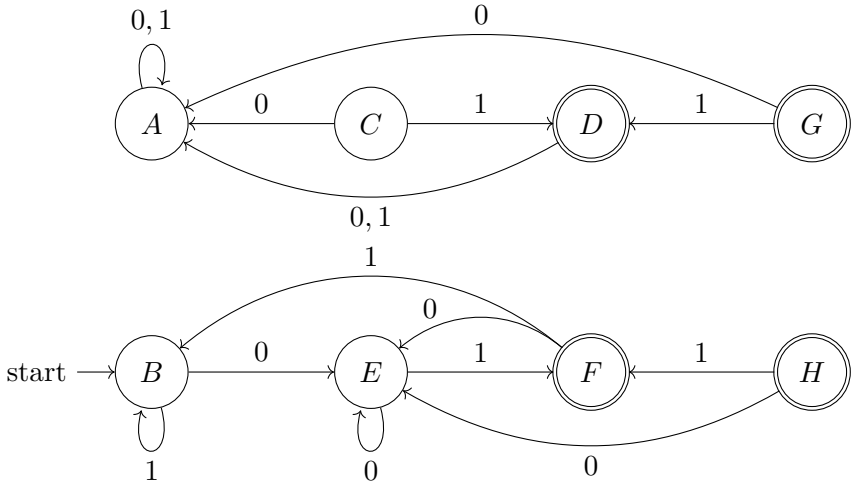
Applicando su N la costruzione per sottoinsiemi, si ricava il DFA D_N descritto nella seguente tabella:

δ_D	0	1
$\varnothing$	$\varnothing$	$\varnothing$
$\to\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\varnothing$	$\{q_2\}$
$*\{q_2\}$	$\varnothing$	$\varnothing$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\varnothing$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

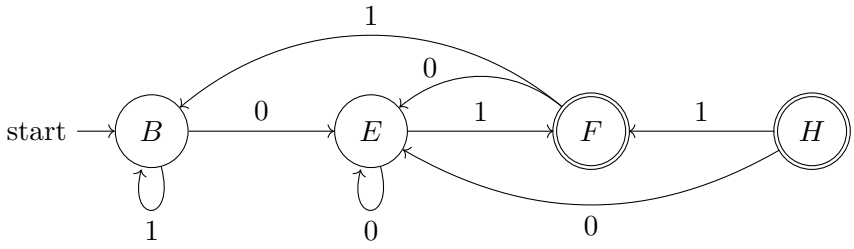
Per semplificare la lettura, e soprattutto la rappresentazione grafica, di questo automa, si possono rinominare gli stati, sostituendo ciascun sottoinsieme di Q con, ad esempio, una lettera (purché le sostituzioni vengano effettuate in modo uniforme in tutto l'automa):

δ_D		0		1	
$\varnothing$	A	$\varnothing$	A	$\varnothing$	A
$\to\{q_0\}$	B	$\{q_0, q_1\}$	E	$\{q_0\}$	B
$\{q_1\}$	C	$\varnothing$	A	$\{q_2\}$	D
$*\{q_2\}$	D	$\varnothing$	A	$\varnothing$	A
$\{q_0, q_1\}$	E	$\{q_0, q_1\}$	E	$\{q_0, q_2\}$	F
$*\{q_0, q_2\}$	F	$\{q_0, q_1\}$	E	$\{q_0\}$	B
$*\{q_1, q_2\}$	G	$\varnothing$	A	$\{q_2\}$	D
$*\{q_0, q_1, q_2\}$	H	$\{q_0, q_1\}$	E	$\{q_0, q_2\}$	F

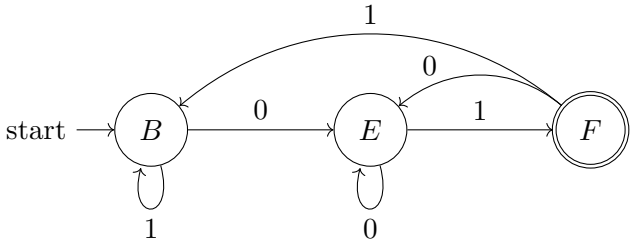
Con questi nomi, il diagramma di transizione dell'automa è:



Rispetto all'NFA di partenza, questo DFA ha un numero *esponenzialmente più grande* di stati (N aveva $|Q| = 3$ stati, e D_N ne ha $|Q_D| = 8 = 2^{|Q|}$). Si osserva però che questo diagramma è un po' "strano": ha due *componenti sconnesse*, non collegate tra di loro, cioè senza transizioni tra l'una e l'altra. Ciò è assolutamente ammesso dalla definizione di DFA, ma, siccome tutte le computazioni dell'automa partono dallo stato iniziale B, si deduce intuitivamente che qualunque stringa in input non porterà mai a uno degli stati A, C, D, G, dunque essi non possono giocare alcun ruolo nel processo di accettazione di una stringa: la costruzione per sottoinsiemi ha introdotto degli stati "inutili", che possono essere eliminati senza cambiare il linguaggio accettato dal DFA:[1]



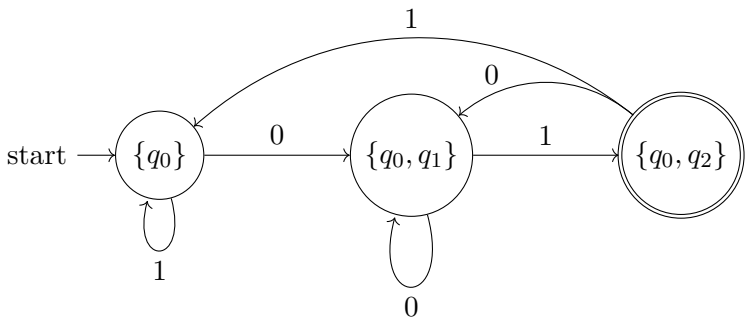
Un'altra osservazione è che anche H è irraggiungibile dallo stato iniziale, poiché ha solo transizioni uscenti, e nessuna transizione entrante (la definizione di DFA richiede che ogni stato abbia una transizione uscente per ogni simbolo, ma non pone alcun vincolo sulle transizioni entranti). Anch'esso può quindi essere eliminato senza cambiare il linguaggio riconosciuto dall'automa, ottenendo così:



δ_D	0	1
$\to B$	E	B
E	E	F
$*F$	E	B

Ritornando ai sottoinsiemi di Q come nomi degli stati, il DFA D ottenuto alla fine di tutti questi passaggi è:

[1] Questi stati possono essere eliminati appunto perché, nel problema del riconoscimento di un linguaggio, si parte sempre dallo stato iniziale. Gli automi a stati finiti possono essere applicati anche ad altri problemi, in cui invece non si identifica uno stato iniziale, e allora non è in generale possibile eliminare alcuno stato senza cambiare il significato dell'automa.



δ_D	0	1
$\rightarrow\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

Grazie a tutte le semplificazioni effettuate dopo la costruzione per sottoinsiemi, il DFA ottenuto ha esattamente lo stesso numero di stati dell'NFA di partenza. Ciò non è però possibile in generale: nel caso peggiore, il DFA equivalente a un NFA con $|Q|$ stati non può avere meno di $2^{|Q|}$ stati.

3 Valutazione differita dei sottoinsiemi

La costruzione data richiede un tempo esponenziale nel numero $|Q|$ di stati dell'NFA di partenza, in quanto produce un DFA con $2^{|Q|}$ stati, quindi deve calcolare le $2^{|Q|}$ righe della tabella di δ_D. Come già accennato nell'esempio, nel caso peggiore ciò è inevitabile, ma "spesso" invece il numero degli stati *raggiungibili* di D_N è approssimativamente uguale a $|Q|$ (il numero di stati di N). Allora, considerando esclusivamente le righe della tabella relative agli stati raggiungibili, si può costruire D_N in modo più efficiente. Questo viene fatto per mezzo di un algoritmo che parte da $\{q_0\}$ (in quanto stato iniziale, esso per definizione è sempre raggiungibile) e "trova" gli altri stati raggiungibili applicando iterativamente (induttivamente) la funzione di transizione δ dell'NFA.

L'algoritmo per determinare gli stati raggiungibili può essere descritto informalmente sotto forma di pseudocodice:

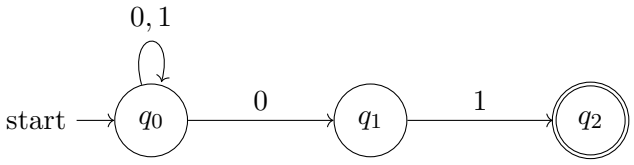
Input: $N = \langle Q, \Sigma, \delta, q_0, F \rangle$

sia $\mathcal{R} = \{\{q_0\}\}$ l'insieme degli stati raggiungibili
`while` (esiste $S \in \mathcal{R}$ che non è ancora stato trattato)
  per ogni $a \in \Sigma$, aggiungi lo stato $\delta_D(S, a) = \bigcup_{p \in S} \delta(p, a)$ a $\mathcal{R}$
`return` l'insieme degli stati raggiungibili $\mathcal{R}$

Quando l'algoritmo termina, ovvero quando non vengono più aggiunti stati nuovi all'insieme $\mathcal{R}$, quest'ultimo contiene tutti gli stati del DFA raggiungibili dallo stato iniziale, che diventa appunto l'insieme degli stati di D_N, $Q_D = \mathcal{R}$, evitando così di considerare gli stati irraggiungibili.

3.1 Esempio

Per illustrare l'applicazione dell'algoritmo, si considera lo stesso NFA N dell'esempio precedente:



δ	0	1
$\to q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
$*q_2$	$\varnothing$	$\varnothing$

Come prima cosa, si pone $\mathcal{R} = \{\{q_0\}\}$. Vengono poi eseguite le iterazioni dell'algoritmo:

1. Si tratta lo stato $\{q_0\}$:

$$\delta_D(\{q_0\}, 0) = \bigcup_{p \in \{q_0\}} \delta(p, 0) = \delta(q_0, 0) = \{q_0, q_1\}$$
$$\delta_D(\{q_0\}, 1) = \bigcup_{p \in \{q_0\}} \delta(p, 1) = \delta(q_0, 1) = \{q_0\}$$

   A $\mathcal{R}$ vengono dunque aggiunti i due stati $\{q_0\}$ (che era già presente, quindi quest'aggiunta non cambia $\mathcal{R}$) e $\{q_0, q_1\}$ (che invece è nuovo):

$$\mathcal{R} = \{\underbrace{\{q_0\}}_{\text{trattato}}, \{q_0, q_1\}\}$$

2. Si tratta lo stato $\{q_0, q_1\}$:

$$\delta_D(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\}$$
$$\delta_D(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0, q_2\}$$

   Dei due stati aggiunti questa volta a $\mathcal{R}$, $\{q_0, q_1\}$ e $\{q_0, q_2\}$, l'unico nuovo è il secondo:

$$\mathcal{R} = \{\underbrace{\{q_0\}, \{q_0, q_1\}}_{\text{trattati}}, \{q_0, q_2\}\}$$

3. Si tratta lo stato $\{q_0, q_2\}$:

$$\delta_D(\{q_0, q_2\}, 0) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\}$$
$$\delta_D(\{q_0, q_2\}, 1) = \delta(q_0, 1) \cup \delta(q_2, 1) = \{q_0\}$$

Entrambi gli stati appena trovati sono già presenti in $\mathcal{R}$, che allora non viene modificato:

$$\mathcal{R} = \{\underbrace{\{q_0\}, \{q_0, q_1\}, \{q_0, q_2\}}_{\text{trattati}}\}$$

Avendo trattato tutti gli stati in $\mathcal{R}$, l'algoritmo termina e restituisce $\{\{q_0\}, \{q_0, q_1\}, \{q_0, q_2\}\}$, che è esattamente l'insieme degli stati raggiungibili del DFA D_N ottenuto tramite la costruzione per sottoinsiemi a partire dall'NFA N (come visto prima, dalle semplificazioni effettuate "a mano").