Esercizio 2.2.8 Sia A un DFA e a un simbolo di input di A tale che, per tutti gli stati q di A, valga $\delta(q, a) = q$.

a) Dimostrate per induzione su n che, per ogni $n \geq 0$, $\hat{\delta}(q, a^n) = q$, dove a^n è la stringa formata da n ripetizioni di a.

b) Dimostrate che o $\{a\}^* \subseteq L(A)$ o $\{a\}^* \cap L(A) = \emptyset$.

***! Esercizio 2.2.9** Sia $A = (Q, \Sigma, \delta, q_0, \{q_f\})$ un DFA e supponiamo che per tutti gli a in Σ si abbia $\delta(q_0, a) = \delta(q_f, a)$.

a) Dimostrate che, per ogni $w \neq \epsilon$, $\hat{\delta}(q_0, w) = \hat{\delta}(q_f, w)$.

b) Dimostrate che se x è una stringa non vuota in $L(A)$, allora per ogni $k > 0$ anche x^k (cioè x scritta un numero k di volte) è in $L(A)$.

***! Esercizio 2.2.10** Considerate il DFA con la seguente tabella di transizione:

	0	1
$\rightarrow A$	A	B
$*B$	B	A

Descrivete in termini informali il linguaggio accettato da questo DFA e dimostrate per induzione sulla lunghezza della stringa di input che la descrizione è corretta. *Suggerimento*: nel porre l'ipotesi induttiva è consigliabile formulare un enunciato su quali input portano a ciascuno stato, e non solo su quali input portano allo stato accettante.

! Esercizio 2.2.11 Ripetete l'Esercizio 2.2.10 per la seguente tabella di transizione:

	0	1
$\rightarrow *A$	B	A
$*B$	C	A
C	C	C

2.3 Automi a stati finiti non deterministici

Un automa a stati finiti non deterministico (*NFA, Nondeterministic Finite Automaton*) può trovarsi contemporaneamente in diversi stati. Questa caratteristica viene sovente espressa come capacità di "scommettere" su certe proprietà dell'input. Per esempio, quando un automa viene usato per cercare determinate sequenze di caratteri (come: parole chiave) in una lunga porzione di testo, è utile "scommettere" che ci si trova all'inizio di una di tali sequenze e usare una sequenza di stati per verificare, carattere per carattere, che compaia la stringa cercata. Vedremo un esempio di questo tipo di applicazione nel Paragrafo 2.4.

Prima di esaminare le applicazioni, è necessario definire gli automi a stati finiti non deterministici e mostrare che accettano gli stessi linguaggi accettati dai DFA. Come i DFA, gli NFA accettano proprio i linguaggi regolari. Tuttavia ci sono buone ragioni per occuparsi degli NFA. Spesso sono più succinti e più facili da definire rispetto ai DFA; inoltre, anche se è sempre possibile convertire un NFA in un DFA, quest'ultimo può avere esponenzialmente più stati di un NFA. Per fortuna casi di questo tipo sono rari.

2.3.1 Descrizione informale degli automi a stati finiti non deterministici

Come un DFA, un NFA ha un insieme finito di stati, un insieme finito di simboli di input, uno stato iniziale e un insieme di stati accettanti. Ha anche una funzione di transizione che chiameremo δ. La differenza tra DFA ed NFA sta nel tipo di δ. Per gli NFA, δ è una funzione che ha come argomenti uno stato e un simbolo di input (come quella dei DFA), ma restituisce un insieme di zero o più stati (invece di un solo stato, come nel caso dei DFA). Cominceremo da un esempio di NFA e poi passeremo a precisare le definizioni.

Esempio 2.6 La Figura 2.9 mostra un automa a stati finiti non deterministico con il compito di accettare tutte e sole le stringhe di 0 e di 1 che finiscono per 01. Lo stato q_0 è lo stato iniziale e possiamo pensare che l'automa si trovi nello stato q_0 (eventualmente insieme ad altri stati) quando non ha ancora "scommesso" che il finale 01 è cominciato. È sempre possibile che il simbolo successivo non sia il primo del suffisso 01, anche se quel simbolo è proprio 0. Dunque lo stato q_0 può operare una transizione verso se stesso sia su 0 sia su 1.

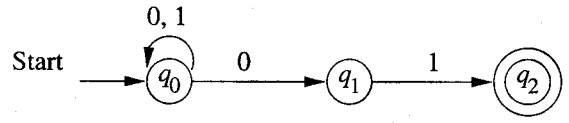

Figura 2.9 Un NFA che accetta tutte le stringhe che finiscono per 01.

Se però il simbolo successivo è 0, l'NFA scommette anche che è iniziato il suffisso 01. Un altro arco etichettato 0 conduce dunque da q_0 allo stato q_1. Si noti che ci sono due archi etichettati 0 in uscita da q_0. L'NFA ha l'opzione di andare verso q_0 oppure verso q_1, ed effettivamente fa entrambe le cose, come si vedrà quando si preciseranno le definizioni. Nello stato q_1 l'NFA verifica che il simbolo successivo sia 1, e se è così, passa allo stato q_2 e accetta.

Si osservi che non ci sono archi uscenti da q_1 etichettati 0 e non ci sono affatto archi uscenti da q_2. In tali situazioni il processo attivo dell'NFA corrispondente a quegli stati semplicemente "muore", sebbene altri processi possano continuare a esistere. Mentre un DFA ha soltanto un unico arco uscente da ogni stato per ogni simbolo di input, un NFA non ha questi vincoli; nella Figura 2.9, per esempio, abbiamo visto casi in cui il numero di archi è zero, uno e due.

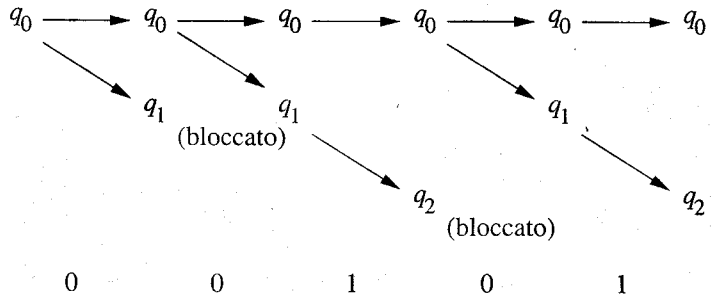

Figura 2.10 Gli stati in cui si trova un NFA mentre viene elaborata la sequenza di input 00101.

La Figura 2.10 illustra come un NFA elabora gli input. Abbiamo mostrato che cosa succede quando l'automa della Figura 2.9 riceve la sequenza di input 00101. Esso parte dal solo stato iniziale, q_0. Quando viene letto il primo 0, l'NFA può passare allo stato q_0 e allo stato q_1. In questo modo fa entrambe le cose. I due processi sono raffigurati nella seconda colonna della Figura 2.10.

A questo punto viene letto il secondo 0. Lo stato q_0 può nuovamente andare sia verso q_0 sia verso q_1. Lo stato q_1, invece, non ha transizioni su 0, perciò "muore". Quando si presenta il terzo input, un 1, bisogna considerare le transizioni sia da q_0 sia da q_1. Troviamo che q_0 va solamente verso q_0 su 1, mentre q_1 va solamente verso q_2. Dunque, dopo aver letto 001, l'NFA si trova negli stati q_0 e q_2. Poiché q_2 è uno stato accettante, l'NFA accetta 001.

Ma l'input non è finito. Il quarto simbolo, uno 0, fa morire il processo associato a q_2, mentre q_0 va sia in q_0 sia in q_1. L'ultimo input, un 1, manda q_0 in q_0 e q_1 in q_2. Dato che ci troviamo di nuovo in uno stato accettante, 00101 viene accettato. □

2.3.2 Definizione di automa a stati finiti non deterministico

Presentiamo ora le nozioni formali relative agli automi a stati finiti non deterministici. Verranno sottolineate le differenze tra i DFA e gli NFA. Un NFA si rappresenta

essenzialmente come un DFA:

$$A = (Q, \Sigma, \delta, q_0, F)$$

dove:

1. Q è un insieme finito di *stati*
2. Σ è un insieme finito di *simboli di input*
3. q_0, elemento di Q, è lo *stato iniziale*
4. F, un sottoinsieme di Q, è l'insieme degli stati *finali* (o *accettanti*)
5. δ, la *funzione di transizione*, è la funzione che ha come argomenti uno stato in Q e un simbolo di input in Σ, e restituisce un sottoinsieme di Q. Si noti che l'unica differenza tra un NFA e un DFA è nel tipo di valore restituito da δ: un insieme di stati nel caso di un NFA e un singolo stato nel caso di un DFA.

Esempio 2.7 L'NFA della Figura 2.9 può essere specificato formalmente come

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

dove la funzione di transizione δ è data dalla tabella di transizione della Figura 2.11. □

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

Figura 2.11 Tabella di transizione di un NFA che accetta tutte le stringhe che finiscono per 01.

Si osservi che la tabella di transizione può specificare la funzione di transizione tanto per un NFA quanto per un DFA. L'unica differenza è che ogni voce nella tabella di un NFA è un insieme, anche se l'insieme è un *singoletto*, ossia ha un solo membro. Inoltre, se non c'è alcuna transizione da uno stato su un dato simbolo di input, la voce adeguata è $\emptyset$, l'insieme vuoto.

2.3.3 La funzione di transizione estesa

Come per i DFA, bisogna estendere la funzione di transizione δ di un NFA a una funzione $\hat{\delta}$ che prende uno stato q e una stringa di simboli di input w, e restituisce l'insieme degli stati in cui si trova l'NFA quando parte dallo stato q ed elabora la stringa w. L'idea è suggerita nella Figura 2.10; in sostanza, se q è l'unico stato nella prima colonna, $\hat{\delta}(q, w)$ è la colonna di stati successivi alla lettura di w. Per esempio la Figura 2.10 indica che $\hat{\delta}(q_0, 001) = \{q_0, q_2\}$. In termini formali definiamo $\hat{\delta}$ per la funzione di transizione δ di un NFA come segue.

BASE $\hat{\delta}(q, \epsilon) = \{q\}$. Se nessun simbolo di input è stato letto, ci troviamo nel solo stato da cui siamo partiti.

INDUZIONE Supponiamo che w sia della forma $w = xa$, dove a è il simbolo finale di w e x è la parte restante. Supponiamo altresì che $\hat{\delta}(q, x) = \{p_1, p_2, \ldots, p_k\}$. Sia

$$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$$

Allora $\hat{\delta}(q, w) = \{r_1, r_2, \ldots, r_m\}$. In termini meno formali computiamo $\hat{\delta}(q, w)$ calcolando inizialmente $\hat{\delta}(q, x)$, e poi seguendo le transizioni etichettate a da tutti questi stati.

Esempio 2.8 Usiamo $\hat{\delta}$ per descrivere come l'NFA della Figura 2.9 elabora l'input 00101. Un compendio dei passi è:

1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$
2. $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
3. $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
4. $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
5. $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
6. $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$.

Il punto (1) è la regola di base. Il punto (2) si ottiene applicando δ all'unico stato, q_0, che si trova nell'insieme precedente: il risultato è $\{q_0, q_1\}$. Il punto (3) si ottiene prendendo l'unione, sui due stati dell'insieme precedente, di ciò che si ottiene quando si applica loro δ con input 0. In altre parole $\delta(q_0, 0) = \{q_0, q_1\}$, mentre $\delta(q_1, 0) = \emptyset$. Per il punto (4) prendiamo l'unione di $\delta(q_0, 1) = \{q_0\}$ e $\delta(q_1, 1) = \{q_2\}$. I punti (5) e (6) sono simili a (3) e (4). □

2.3.4 Il linguaggio di un NFA

Come abbiamo suggerito, un NFA accetta una stringa w se, mentre si leggono i caratteri di w, è possibile fare una sequenza di scelte dello stato successivo che porta dallo stato iniziale a uno stato accettante. Il fatto che altre scelte per i simboli di input di w conducano a uno stato non accettante, oppure non conducano ad alcuno stato (cioè che una sequenza di stati muoia), non impedisce a w di venire accettato dall'NFA nel suo insieme. Formalmente, se $A = (Q, \Sigma, \delta, q_0, F)$ è un NFA, allora

$$L(A) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

In altre parole $L(A)$ è l'insieme delle stringhe w in Σ^* tali che $\hat{\delta}(q_0, w)$ contenga almeno uno stato accettante.

Esempio 2.9 Per esemplificare, proviamo in termini formali che l'NFA della Figura 2.9 accetta il linguaggio $L = \{w \mid w \text{ termina per } 01\}$. La dimostrazione è un'induzione mutua dei seguenti tre enunciati, che caratterizzano i tre stati:

1. $\hat{\delta}(q_0, w)$ contiene q_0 per ogni w
2. $\hat{\delta}(q_0, w)$ contiene q_1 se e solo se w finisce per 0
3. $\hat{\delta}(q_0, w)$ contiene q_2 se e solo se w finisce per 01.

Per dimostrare questi enunciati, bisogna considerare in che modo A può raggiungere ciascuno stato; ossia: qual è stato l'ultimo simbolo di input, e in quale stato si trovava A prima di leggere quel simbolo?

Dato che il linguaggio di quest'automa è l'insieme delle stringhe w tali che $\hat{\delta}(q_0, w)$ contiene q_2 (perché q_2 è l'unico stato accettante), la dimostrazione dei tre enunciati, in particolare di (3), garantisce che il linguaggio dell'NFA è l'insieme delle stringhe che finiscono per 01. La dimostrazione del teorema è un'induzione su $|w|$, la lunghezza di w, a partire dalla lunghezza 0.

BASE Se $|w| = 0$, allora $w = \epsilon$. L'enunciato (1) dice che $\hat{\delta}(q_0, \epsilon)$ contiene q_0; ciò segue dalla base della definizione di $\hat{\delta}$. Circa l'enunciato (2) sappiamo che ϵ non finisce per 0 e sappiamo inoltre che $\hat{\delta}(q_0, \epsilon)$ non contiene q_1, ancora una volta per la base della definizione di $\hat{\delta}$. Dunque le ipotesi di entrambe le direzioni dell'enunciato se-e-solo-se sono false, e di conseguenza entrambe le direzioni dell'enunciato sono vere. La dimostrazione dell'enunciato (3) per $w = \epsilon$ è essenzialmente la stessa della dimostrazione dell'enunciato (2) presentata sopra.

INDUZIONE Assumiamo che $w = xa$, dove a è un simbolo, 0 oppure 1. Possiamo supporre che gli enunciati dall'(1) al (3) siano validi per x e dobbiamo dimostrarli per w. Vale a dire, assumiamo $|w| = n + 1$, dunque $|x| = n$. Assumiamo l'ipotesi induttiva per n e dimostriamola per $n + 1$.

1. Sappiamo che $\hat{\delta}(q_0, x)$ contiene q_0. Poiché esistono transizioni sia su 0 sia su 1 da q_0 verso se stesso, ne consegue che anche $\hat{\delta}(q_0, w)$ contiene q_0; dunque l'enunciato (1) è dimostrato per w.

2. (Se) Assumiamo che w finisca per 0, ossia $a = 0$. Per l'enunciato (1) applicato a x, sappiamo che $\hat{\delta}(q_0, x)$ contiene q_0. Poiché esiste una transizione da q_0 a q_1 su input 0, concludiamo che $\hat{\delta}(q_0, w)$ contiene q_1.

   (Solo-se) Supponiamo che $\hat{\delta}(q_0, w)$ contenga q_1. Dal diagramma della Figura 2.9 si ricava che c'è un solo modo di raggiungere lo stato q_1: che la sequenza di input w sia della forma $x0$. Ciò è sufficiente per dimostrare la parte solo-se dell'enunciato (2).

3. (Se) Assumiamo che w finisca per 01. Allora, se $w = xa$, sappiamo che $a = 1$ e x finisce per 0. Per l'enunciato (2) applicato a x, sappiamo che $\hat{\delta}(q_0, x)$ contiene q_1. Poiché esiste una transizione da q_1 a q_2 su input 1, concludiamo che $\hat{\delta}(q_0, w)$ contiene q_2.

   (Solo-se) Supponiamo che $\hat{\delta}(q_0, w)$ contenga q_2. Dal diagramma della Figura 2.9 si deduce che c'è un solo modo di giungere nello stato q_2: che w sia della forma $x1$, dove $\hat{\delta}(q_0, x)$ contiene q_1. Per l'enunciato (2) applicato a x, sappiamo che x finisce per 0. Dunque w finisce per 01, e abbiamo dimostrato l'enunciato (3).

□

2.3.5 Equivalenza di automi a stati finiti deterministici e non deterministici

Ci sono molti linguaggi per i quali è più facile costruire un NFA anziché un DFA, come il linguaggio (Esempio 2.6) delle stringhe che finiscono per 01; eppure, per quanto sorprendente, ogni linguaggio che può essere descritto da un NFA può essere descritto anche da un DFA. Inoltre il DFA ha in pratica circa tanti stati quanti l'NFA, sebbene abbia spesso più transizioni. Nel peggiore dei casi, tuttavia, il più piccolo DFA può avere 2^n stati, mentre il più piccolo NFA per lo stesso linguaggio ha solo n stati.

La dimostrazione che i DFA possono fare le stesse cose degli NFA si serve di un'importante costruzione, detta "costruzione per sottoinsiemi", in quanto comporta la costruzione di tutti i sottoinsiemi dell'insieme di stati dell'NFA. In generale molte dimostrazioni

sugli automi richiedono la costruzione di un automa a partire da un altro. È importante considerare la costruzione per sottoinsiemi come esempio di descrizione formale di un automa nei termini degli stati e delle transizioni di un altro, senza conoscere i particolari del secondo automa.

La costruzione per sottoinsiemi parte da un NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$. Il suo fine è la descrizione di un DFA $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ tale che $L(D) = L(N)$. Si noti che gli alfabeti di input dei due automi sono gli stessi e lo stato iniziale di D è l'insieme che contiene solo lo stato iniziale di N. Gli altri componenti di D sono costruiti come segue.

- Q_D è l'insieme dei sottoinsiemi di Q_N; vale a dire, Q_D è l'*insieme potenza* di Q_N. Osserviamo che se Q_N ha n stati, allora Q_D avrà 2^n stati. Spesso dallo stato iniziale di Q_D non è possibile raggiungere tutti questi stati. Gli stati inaccessibili possono essere eliminati. Dunque, in effetti, il numero di stati di D può essere molto inferiore a 2^n.

- F_D è l'insieme dei sottoinsiemi S di Q_N tali che $S \cap F_N \neq \emptyset$. In altre parole F_D è formato dagli insiemi di stati di N che includono almeno uno stato accettante.

- Per ogni insieme $S \subseteq Q_N$ e per ogni simbolo di input a in Σ,

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

  Cioè, per computare $\delta_D(S, a)$ consideriamo tutti gli stati p in S, rileviamo in quali insiemi di stati l'automa N va a finire partendo da p e leggendo a, e prendiamo infine l'unione di tutti questi insiemi.

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

Figura 2.12 La costruzione completa per sottoinsiemi dalla Figura 2.9.

Esempio 2.10 Sia N l'automa della Figura 2.9, che accetta tutte le stringhe che finiscono per 01. Dato che l'insieme di stati di N è $\{q_0, q_1, q_2\}$, la costruzione per sottoinsiemi produce un DFA con $2^3 = 8$ stati, corrispondenti a tutti i sottoinsiemi dei tre stati. La Figura 2.12 mostra la tabella di transizione per questi otto stati; vedremo tra breve i dettagli su come le singole voci vengano computate.

Notiamo che la tabella di transizione appartiene a un automa a stati finiti deterministico. Anche se le voci nella tabella sono insiemi, gli stati del DFA così costruito *sono* insiemi. Per chiarire questo punto, possiamo inventare nomi nuovi per questi stati, per esempio A per $\emptyset$, B per $\{q_0\}$, e così via. La tabella di transizione della Figura 2.13 definisce esattamente lo stesso automa della Figura 2.12, ma chiarisce che le voci della tabella sono singoli stati del DFA.

	0	1
A	A	A
$\rightarrow B$	E	B
C	A	D
$*D$	A	A
E	E	F
$*F$	E	B
$*G$	A	D
$*H$	E	F

Figura 2.13 Ridenominazione degli stati della Figura 2.12.

Degli otto stati della Figura 2.13, partendo dallo stato iniziale B possiamo raggiungere solo gli stati B, E ed F. Gli altri cinque stati sono inaccessibili dallo stato iniziale e potrebbero anche non esserci. Spesso si può evitare il passo di costruire voci della tabella di transizione per ogni sottoinsieme di stati (cosa che richiede un tempo esponenziale) se si compie una "valutazione differita" dei sottoinsiemi. Vediamo come.

BASE Il singoletto formato dal solo stato iniziale di N è certamente accessibile.

INDUZIONE Supponiamo di aver determinato che l'insieme S è accessibile. Allora per ogni simbolo di input a computiamo l'insieme di stati $\delta_D(S, a)$; sappiamo che anche questi insiemi di stati saranno accessibili.

Per l'esempio in esame, sappiamo che $\{q_0\}$ è uno stato del DFA D. Troviamo che $\delta_D(\{q_0\}, 0) = \{q_0, q_1\}$ e $\delta_D(\{q_0\}, 1) = \{q_0\}$. Ambedue questi fatti si deducono dal diagramma di transizione della Figura 2.9 e osservando che su 0 esistono archi uscenti da q_0 verso sia q_0 sia q_1, mentre su 1 esiste un arco diretto solo verso q_0. Abbiamo dunque una riga della tabella di transizione per il DFA: la seconda riga nella Fig. 2.12.

Uno dei due insiemi computati è "vecchio"; $\{q_0\}$ è già stato considerato. $\{q_0, q_1\}$, invece, è nuovo e le sue transizioni devono essere calcolate. Troviamo $\delta_D(\{q_0, q_1\}, 0) = \{q_0, q_1\}$ e $\delta_D(\{q_0, q_1\}, 1) = \{q_0, q_2\}$. A titolo di esempio esplicitiamo il secondo calcolo; sappiamo che

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

Ora abbiamo la quinta riga della Figura 2.12 e abbiamo scoperto un nuovo stato di D, cioè $\{q_0, q_2\}$. Un calcolo simile ci dice che

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

Questi calcoli forniscono la sesta riga della Figura 2.12, ma producono solo insiemi di stati che sono già stati esaminati.

Quindi la costruzione per sottoinsiemi è arrivata a un punto fermo; ora conosciamo tutti gli stati accessibili e le loro transizioni. L'intero DFA è rappresentato dalla Figura 2.14. Osserviamo che ha solo tre stati, cioè, casualmente, lo stesso numero di stati dell'NFA della Figura 2.9 dal quale è stato costruito. D'altra parte il DFA della Figura 2.14 ha sei transizioni, a fronte delle quattro della Figura 2.9. □

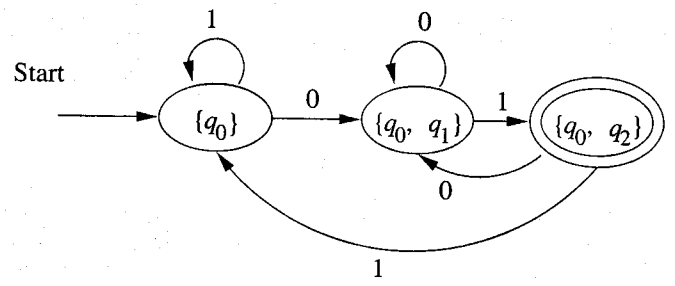

Figura 2.14 Il DFA costruito dall'NFA della Figura 2.9.

Dobbiamo dimostrare formalmente che la costruzione per sottoinsiemi è corretta, sebbene l'intuizione di ciò sia suffragata dagli esempi. Dopo aver letto la sequenza di simboli di input w, il DFA costruito si trova in un unico stato, che è l'insieme degli stati in cui si troverebbe l'NFA dopo aver letto w. Dato che gli stati accettanti del DFA sono quegli stati che includono almeno uno stato accettante dell'NFA, e anche l'NFA accetta se giunge in almeno uno dei suoi stati accettanti, possiamo concludere che il DFA e l'NFA accettano esattamente le stesse stringhe e dunque lo stesso linguaggio.

SOLO ENUNCIATO

Teorema 2.11 Se $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ è il DFA costruito dall'NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ mediante la costruzione per sottoinsiemi, allora $L(D) = L(N)$.

DIMOSTRAZIONE In primo luogo dimostriamo, per induzione su $|w|$, che

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$$

Si noti che ognuna delle funzioni $\hat{\delta}$ restituisce un insieme di stati di Q_N, ma $\hat{\delta}_D$ interpreta quest'insieme come uno degli stati di Q_D (che è l'insieme potenza di Q_N), mentre $\hat{\delta}_N$ lo interpreta come un sottoinsieme di Q_N.

BASE Sia $|w| = 0$, ossia $w = \epsilon$. Per le definizioni della base di $\hat{\delta}$ per i DFA e gli NFA, sia $\hat{\delta}_D(\{q_0\}, \epsilon)$ sia $\hat{\delta}_N(q_0, \epsilon)$ sono $\{q_0\}$.

INDUZIONE Sia w di lunghezza $n + 1$ e supponiamo vero l'enunciato per la lunghezza n. Scomponiamo w in $w = xa$, dove a è il simbolo finale di w. Per l'ipotesi induttiva, $\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x)$. Questi due insiemi di stati di N siano $\{p_1, p_2, \ldots, p_k\}$.

La parte induttiva della definizione di $\hat{\delta}$ per gli NFA ci dice

$$\hat{\delta}_N(q_0, w) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \tag{2.2}$$

La costruzione per sottoinsiemi, d'altra parte, dice che

$$\delta_D(\{p_1, p_2, \ldots, p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \tag{2.3}$$

Serviamoci ora della (2.3) e del fatto che $\hat{\delta}_D(\{q_0\}, x) = \{p_1, p_2, \ldots, p_k\}$ nella parte induttiva della definizione di $\hat{\delta}$ per i DFA:

$$\hat{\delta}_D(\{q_0\}, w) = \delta_D\big(\hat{\delta}_D(\{q_0\}, x), a\big) = \delta_D(\{p_1, p_2, \ldots, p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \tag{2.4}$$

Dunque le Equazioni (2.2) e (2.4) dimostrano che $\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$. Osservando che sia D sia N accettano w se e solo se, rispettivamente, $\hat{\delta}_D(\{q_0\}, w)$ e $\hat{\delta}_N(q_0, w)$ contengono uno stato in F_N, abbiamo portato a compimento la dimostrazione che $L(D) = L(N)$. □

Teorema 2.12 Un linguaggio L è accettato da un DFA se e solo se L è accettato da un NFA.

DIMOSTRAZIONE (Se) La parte "se" è costituita dalla costruzione per sottoinsiemi e dal Teorema 2.11.

(Solo-se) Questa parte è facile; dobbiamo soltanto convertire un DFA in un NFA identico. In modo intuitivo, se abbiamo il diagramma di transizione di un DFA, possiamo anche interpretarlo come il diagramma di transizione di un NFA, in cui c'è esattamente una scelta di transizione in qualunque caso. In termini più formali, sia $D = (Q, \Sigma, \delta_D, q_0, F)$ un DFA. Definiamo l'NFA equivalente $N = (Q, \Sigma, \delta_N, q_0, F)$, dove δ_N è definita dalla seguente regola.

- Se $\delta_D(q, a) = p$, allora $\delta_N(q, a) = \{p\}$.

È facile dimostrare, per induzione su $|w|$, che se $\hat{\delta}_D(q_0, w) = p$ allora

$$\hat{\delta}_N(q_0, w) = \{p\}$$

Lasciamo che sia il lettore a farlo. Di conseguenza la stringa w viene accettata da D se e solo se viene accettata da N; cioè $L(D) = L(N)$. □

NO

2.3.6 Un caso sfavorevole di costruzione per sottoinsiemi

Nell'Esempio 2.10 abbiamo visto che il DFA non aveva più stati di quanti ne avesse l'NFA equivalente. Come già detto, nella pratica è normale che il DFA abbia approssimativamente lo stesso numero di stati dell'NFA a partire dal quale è stato costruito. Tuttavia è possibile una crescita esponenziale del numero degli stati; tutti i 2^n stati del DFA che si possono costruire da un NFA di n stati potrebbero risultare accessibili. Il seguente esempio non raggiunge il limite, ma illustra un caso in cui il più piccolo DFA equivalente a un NFA di $n + 1$ stati ha 2^n stati.

Esempio 2.13 Consideriamo l'NFA N della Figura 2.15. $L(N)$ è l'insieme di tutte le stringhe di 0 e di 1 tali che l'n-esimo simbolo dalla fine sia 1. Intuitivamente un DFA D che accetti questo linguaggio deve ricordare gli ultimi n simboli che ha letto.

Poiché ci sono 2^n sequenze distinte di lunghezza n, se D avesse meno di 2^n stati, ci sarebbe uno stato q raggiunto dopo aver letto due sequenze distinte di n bit, poniamo $a_1 a_2 \cdots a_n$ e $b_1 b_2 \cdots b_n$.

Dato che le sequenze sono diverse, devono differire in una posizione, per esempio $a_i \neq b_i$. Supponiamo (per simmetria) che $a_i = 1$ e $b_i = 0$. Se $i = 1$, allora q deve essere sia uno stato accettante sia uno stato non accettante, visto che $a_1 a_2 \cdots a_n$ è accettato (l'n-esimo simbolo dalla fine è 1), e invece $b_1 b_2 \cdots b_n$ no. Se $i > 1$, allora consideriamo lo stato p in cui D entra dopo aver letto $i - 1$ simboli 0 da q. Allora p deve essere sia accettante sia non accettante, dato che $a_i a_{i+1} \cdots a_n 00 \cdots 0$ è accettato e $b_i b_{i+1} \cdots b_n 00 \cdots 0$ no.

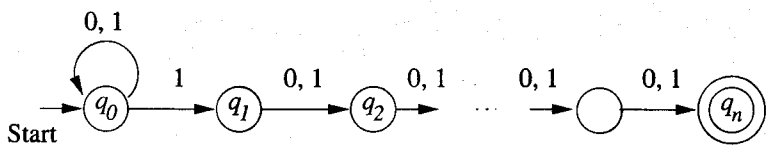

Figura 2.15 Questo NFA non ha alcun DFA equivalente con meno di 2^n stati.

Passiamo ora al funzionamento dell'NFA N della Figura 2.15. Esiste uno stato q_0 in cui l'NFA si trova sempre, a prescindere dagli input letti. Se l'input successivo è 1, N può anche "scommettere" che questo 1 sarà l'n-esimo simbolo dalla fine, quindi passa allo stato q_1 così come a q_0. Dallo stato q_1 qualunque input porta N in q_2, l'input successivo lo porta in q_3, e così via, finché, $n-1$ input dopo, si trova nello stato accettante q_n. Esprimiamo ora l'enunciato formale relativo alla condotta degli stati di N.

1. N si trova nello stato q_0 dopo aver letto una qualunque sequenza di input w.
2. N si trova nello stato q_i, per $i = 1, 2, \ldots, n$, dopo aver letto la sequenza di input w se e solo se l'i-esimo simbolo dalla fine di w è 1; in altre parole w è della forma $x1a_1a_2\cdots a_{i-1}$, dove ogni a_j è un simbolo di input.

Non dimostreremo questi enunciati formalmente; la dimostrazione è una facile induzione su $|w|$, del tipo illustrato nell'Esempio 2.9. Per completare la dimostrazione che l'automa accetta esattamente le stringhe con un 1 nell'n-esima posizione dalla fine, consideriamo l'enunciato (2) con $i = n$, secondo il quale N si trova nello stato q_n se e solo se l'n-esimo simbolo dalla fine è 1. Essendo però q_n l'unico stato accettante, tale condizione caratterizza precisamente l'insieme delle stringhe accettate da N. □

2.3.7 Esercizi

* **Esercizio 2.3.1** Convertite in un DFA il seguente NFA:

	0	1
$\rightarrow p$	$\{p, q\}$	$\{p\}$
q	$\{r\}$	$\{r\}$
r	$\{s\}$	$\emptyset$
$*s$	$\{s\}$	$\{s\}$

Esercizio 2.3.2 Convertite in un DFA il seguente NFA:

Il principio della piccionaia

Nell'Esempio 2.13 abbiamo impiegato un'importante tecnica di ragionamento, detta *principio della piccionaia*. In parole povere, se si hanno più piccioni che cellette e ogni piccione si ricovera in una celletta, allora ci deve essere almeno una celletta contenente più di un piccione. Nel nostro caso i "piccioni" sono le sequenze di n bit e le "cellette" sono gli stati. Poiché esistono meno stati che sequenze, a uno stato devono essere assegnate due sequenze.

Il principio della piccionaia può sembrare ovvio, ma dipende dal fatto che il numero delle cellette è finito. Funziona perciò per automi a stati finiti, in cui gli stati corrispondono alle cellette della piccionaia. Non si può applicare invece ad altri tipi di automi che hanno un numero infinito di stati.

Per capire perché è essenziale che il numero delle cellette sia finito, si consideri la situazione infinita in cui le cellette corrispondono agli interi $1, 2, \ldots$. Numeriamo i piccioni $0, 1, 2, \ldots$, in modo che ci sia un piccione in più rispetto alle cellette. Possiamo allora mandare il piccione i nella celletta $i+1$ per ogni $i \geq 0$. Così ogni piccione ha la sua celletta e non ci sono due piccioni obbligati a condividere lo stesso spazio.

	0	1
$\rightarrow p$	$\{q, s\}$	$\{q\}$
$*q$	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
$*s$	$\emptyset$	$\{p\}$

! **Esercizio 2.3.3** Convertite il seguente NFA in un DFA e descrivete informalmente il linguaggio che accetta.

	0	1
$\rightarrow p$	$\{p, q\}$	$\{p\}$
q	$\{r, s\}$	$\{t\}$
r	$\{p, r\}$	$\{t\}$
$*s$	$\emptyset$	$\emptyset$
$*t$	$\emptyset$	$\emptyset$

! **Esercizio 2.3.4** Definite automi a stati finiti non deterministici che accettino i seguenti linguaggi. Cercate di sfruttare il più possibile il non determinismo.

Stati trappola e DFA cui mancano alcune transizioni

Abbiamo formalmente definito un DFA in modo che per ogni stato e per ogni simbolo di input ci sia esattamente una transizione verso un altro stato. A volte, però, in situazioni in cui sappiamo che nessuna estensione della sequenza di input può essere accettata, è più opportuno fare in modo che il DFA "muoia". Per esempio osserviamo l'automa della Figura 1.2, che svolge il proprio compito riconoscendo una sola parola, `then`, e nient'altro. Quest'automa non è tecnicamente un DFA, in quanto gli mancano le transizioni sulla maggior parte dei simboli da ciascuno dei suoi stati.

Tuttavia tale automa è un NFA. Se utilizziamo la costruzione per sottoinsiemi in modo da convertirlo in un DFA, l'automa sembra quasi lo stesso, ma include uno *stato trappola*, ossia uno stato non accettante che conduce a se stesso su ogni possibile simbolo di input. Lo stato trappola corrisponde a $\emptyset$, l'insieme vuoto di stati dell'automa della Figura 1.2.

In generale possiamo aggiungere uno stato trappola a qualunque automa che abbia *non più* di una transizione per ogni stato e simbolo di input: si aggiunge una transizione verso lo stato trappola da uno stato q, su tutti i simboli di input per i quali q non ha transizioni. Il risultato sarà un DFA nel senso più stretto. A volte, perciò, faremo riferimento a un automa come un DFA se ha *al massimo* una transizione uscente da ciascuno stato su ogni simbolo, piuttosto che se ha *esattamente una* transizione.

* a) L'insieme delle stringhe sull'alfabeto $\{0, 1, \ldots, 9\}$ tali che la cifra finale sia comparsa in precedenza.

b) L'insieme delle stringhe sull'alfabeto $\{0, 1, \ldots, 9\}$ tali che la cifra finale *non* sia comparsa in precedenza.

c) L'insieme delle stringhe di 0 e di 1 tali che esistano due 0 separati da un numero di posizioni multiplo di 4. Si noti che 0 è un multiplo di 4.

Esercizio 2.3.5 Nella parte solo-se del Teorema 2.12 abbiamo omesso la dimostrazione per induzione su $|w|$ che se $\hat{\delta}_D(q_0, w) = p$ allora $\hat{\delta}_N(q_0, w) = \{p\}$. Fornite tale dimostrazione.

! **Esercizio 2.3.6** Nel riquadro su "Stati trappola e DFA cui mancano alcune transizioni" sosteniamo che, se N è un NFA che ha al massimo una scelta di stato per ogni stato e simbolo di input (ossia $\delta(q, a)$ non ha mai più di un elemento), allora il DFA D costruito da N con la costruzione per sottoinsiemi ha esattamente gli stati e le transizioni di N, più le transizioni verso un nuovo stato trappola ogni volta che a N manca una transizione per uno stato e un simbolo di input dati. Dimostrate questa affermazione.

Esercizio 2.3.7 Nell'Esempio 2.13 abbiamo affermato che l'NFA N si trova nello stato q_i, per $i = 1, 2, \ldots, n$, dopo aver letto la sequenza di input w, se e solo se l'i-esimo simbolo dalla fine di w è 1. Dimostrate quest'asserzione.

2.4 Un'applicazione: ricerche testuali

In questo paragrafo vedremo che la trattazione teorica presentata nel paragrafo precedente, in cui abbiamo considerato il problema di decidere se una sequenza di bit finisce per 01, è in effetti un modello appropriato per molti problemi reali che si presentano in applicazioni come le ricerche nel Web e il prelievo di informazioni da un testo.

2.4.1 Ricerca di stringhe in un testo

Un problema comune nell'era del Web e di altre raccolte di testi on-line è il seguente: dato un insieme di parole, trovare tutti i documenti che ne contengono una (o tutte). Un motore di ricerca è un tipico esempio di tale procedura. Un motore di ricerca usa una tecnica particolare, detta *indici invertiti*, in cui per ogni parola che compare nel Web (ci sono 100.000.000 di parole diverse) viene memorizzata una lista di tutti i luoghi in cui questa si incontra. Una macchina con una memoria centrale molto grande può tenere a disposizione le liste più frequenti, consentendo ricerche simultanee da parte di più utenti.

Le tecniche a indici invertiti non utilizzano gli automi a stati finiti, ma richiedono molto tempo per la copia delle pagine e la preparazione degli indici. Esistono svariate applicazioni affini per le quali gli indici invertiti non sono adatti, mentre le tecniche fondate sugli automi risultano appropriate. Le caratteristiche che rendono un'applicazione adeguata alle ricerche che impiegano gli automi sono due.

1. Il deposito sul quale viene condotta la ricerca cambia rapidamente. Per esempio:
   (a) gli analisti della comunicazione esplorano quotidianamente le notizie on-line del giorno in cerca di argomenti specifici; per esempio un analista finanziario potrebbe cercare le sigle di particolari azioni o nomi di aziende
   (b) un "robot della spesa" potrebbe cercare i prezzi correnti degli articoli richiesti dai suoi clienti; il robot può recuperare dal Web pagine di catalogo e cercarvi le parole che indicano il prezzo di un particolare articolo.
2. I documenti da esaminare non possono essere catalogati. Per esempio su Amazon.com un *crawler* difficilmente troverà tutte le pagine relative a ogni libro in