SAPIENZA
UNIVERSITÀ DI ROMA

"Sapienza" Università di Roma
Ingegneria dell'Informazione,
Informatica e Statistica
Dipartimento di Informatica

Automi: Calcolabilità e Complessità

Appunti integrati con il libro "Introduzione alla teoria della computazione", Michael Sipser

Author
Alessio Bandiera

10 settembre 2023

Indice

Informazioni e Contatti

Prerequisiti consigliati:

- TODO: DA DECIDERE

Segnalazione errori ed eventuali migliorie:

Per segnalare eventuali errori e/o migliorie possibili, si prega di utilizzare il **sistema di Issues fornito da GitHub** all'interno della pagina della repository stessa contenente questi ed altri appunti (link fornito al di sotto), utilizzando uno dei template già forniti compilando direttamente i campi richiesti.

Gli appunti sono in continuo aggiornamento, pertanto, previa segnalazione, si prega di controllare se l'errore sia ancora presente nella versione più recente.

Licenza di distribuzione:

Contatti dell'autore e ulteriori link:

- Github: **https://github.com/ph04**
- Email: **alessio.bandiera02@gmail.com**
- LinkedIn: **Alessio Bandiera**

1 Linguaggi regolari

1.1 Automi finiti

1.1.1 Linguaggi

Definizione 1.1.1.1: Alfabeto

Si definisce **alfabeto** un qualsiasi insieme finito, non vuoto; i suoi elementi sono detti **simboli**.

Esempio 1.1.1.1 (Alfabeto)**.** $\Sigma = \{\texttt{0}, \texttt{1}, \texttt{x}, \texttt{y}, \texttt{z}\}$ è un alfabeto, composto da 5 simboli.

Definizione 1.1.1.2: Stringa

Sia Σ un alfabeto; una **stringa su** Σ è una sequenza finita di simboli di Σ; la **stringa vuota** appartiene ad ogni alfabeto, ed è denotata con ε.

- Data una stringa w di Σ, allora $|w|$ è la lunghezza di w.
- Se w ha lunghezza $n \in \mathbb{N}$, allora è possibile scrivere che $w = w_1 w_2 \cdots w_n$ con $w_i \in \Sigma$ e $i \in [1, n]$.

Esempio 1.1.1.2 (Stringa)**.** Sia $\Sigma = \{\texttt{0}, \texttt{1}, \texttt{x}, \texttt{y}, \texttt{z}\}$ un alfabeto; allora una sua possibile stringa è $w = \texttt{x1y0z}$.

Definizione 1.1.1.3: Stringa inversa

Sia Σ un alfabeto, e $w = w_1 w_2 \cdots w_n$ una sua stringa; allora si definisce l'**inversa** di w come segue: $w^{\mathcal{R}} = w_n w_{n-1} \cdots w_1$

Definizione 1.1.1.4: Concatenazione

Sia Σ un alfabeto, e $x = x_1x_2\cdots x_n, y = y_1y_2\cdots y_n$ due sue stringhe; allora xy è la stringa ottenuta attraverso la **concatenazione** di x ed y.

Per indicare una stringa concatenata con se stessa k volte, si utilizza la notazione $x^k = \underbrace{xx\cdots x}_{k}$.

Definizione 1.1.1.5: Prefisso

Sia Σ un alfabeto, ed x, y due sue stringhe; allora x è detto essere un **prefisso** di y, se $\exists z \mid xz = y$, con z stringa in Σ.

Esempio 1.1.1.3 (Prefisso)**.** Sia $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}\}$ un alfabeto; allora la stringa $x = \texttt{ab}$ è prefisso della stringa $y = \texttt{abc}$, poiché esiste una stringa $z = \texttt{c}$ tale per cui $xz = y$.

Definizione 1.1.1.6: Linguaggio

Sia Σ un alfabeto; si definisce **linguaggio** un insieme di stringhe di Σ. Un linguaggio è detto **prefisso**, se nessun suo elemento è prefisso di un altro. Il linguaggio vuoto si indica con $\varnothing$.

1.1.2 Automi finiti

Definizione 1.1.2.1: Automa finito

Un **automa finito**, o **macchina**, è una quintupla $(Q, \Sigma, \delta, q_0, F)$, dove

- Q è l'**insieme degli stati** dell'automa, un insieme *finito*
- Σ è l'**alfabeto** dell'automa, un insieme *finito*
- $\delta : Q \times \Sigma \to Q$ è la **funzione di transizione**, che definisce la relazione tra gli stati
- $q_0 \in Q$ è lo **stato iniziale**
- $F \subseteq Q$ è l'**insieme degli stati accettanti**, sui quali le stringhe possono terminare

Esempio 1.1.2.1 (Automa finito)**.** Un esempio di automa finito è il seguente:

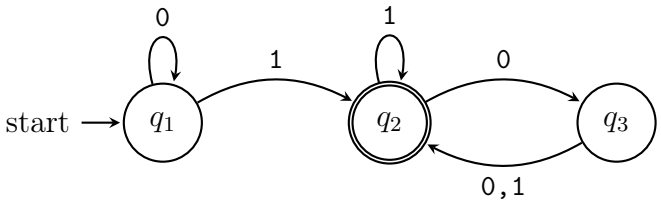


Figura 1.1: Un automa finito.

esso può essere descritto secondo la quintupla $(Q, \Sigma, \delta, q_0, F)$ come segue:

- $Q = \{q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- δ è la seguente:

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

- q_1 è lo stato iniziale
- $F = \{q_2\} \subseteq Q$

Definizione 1.1.2.2: Linguaggio di un automa

Sia M un automa; allora il **linguaggio** di M è un insieme $L(M)$ contenente tutte le stringhe accettate da M; simmetricamente, si dice che M **riconosce** $L(M)$.

Esempio 1.1.2.2 (Linguaggio di un automa)**.** Si consideri il seguente automa M_1:

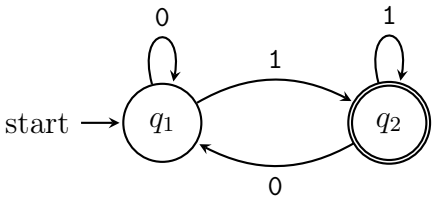


Figura 1.2: Un automa M_1.

sapendo che $\Sigma = \{0, 1\}$, che q_1 è lo stato iniziale, e che $F = \{q_2\}$, ci si convince facilmente che

$$L(M_1) = \{w \mid w = w_1 w_2 \cdots w_{n-1} 1, n \in \mathbb{N}\}$$

ovvero, M_1 accetta tutte e sole le stringhe che terminano per 1.

Definizione 1.1.2.3: Stringhe accettate

Sia $M = (Q, \Sigma, \delta, q_0, F)$ un automa, e sia $w = w_1 \cdots w_n$ una stringa tale per cui $\forall i \in [1, n] \quad w_i \in \Sigma$; allora, M **accetta** w se esiste una sequenza di stati $r_0, \ldots, r_n \in Q$ tali per cui

- $r_0 = q_0$
- $\forall i \in [1, n-1] \quad \delta(r_i, w_{i+1}) = r_{i+1}$
- $r_n \in F$

Dato un linguaggio A, si ha che M **riconosce** A se e solo se $A = \{w \mid M \text{ accetta } w\}$.

Definizione 1.1.2.4: Linguaggio regolare

Un linguaggio è detto **regolare** se e solo se esiste un automa finito che lo riconosce.

1.2 Non determinismo

1.2.1 Definizioni

Definizione 1.2.1.1: NFA

Un **NFA** (*Nondeterministic Finite Automaton*) è un automa in cui possono esistere varie scelte per lo stato successivo in ogni punto. Durante la computazione, ogni volta che viene incontrata una scelta, la macchina si *divide*, e ognuno dei vari automi risultanti computa le varie scelte indipendentemente.

Formalmente, un NFA è una quintupla $(Q, \Sigma, \delta, q_0, F)$, dove

- Q è l'**insieme degli stati**, un insieme *finito*
- Σ è l'**alfabeto** dell'automa, un insieme *finito*
- $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ è la **funzione di transizione**, che definisce la relazione tra gli stati
- $q_0 \in Q$ è lo **stato iniziale**
- $F \subseteq Q$ è l'**insieme degli stati accettanti**

dove $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$.

Se il simbolo di input successivo non compare su alcuno degli archi uscenti dallo stato occupato da una copia della macchina, quella copia cessa di proseguire; inoltre, se *una qualunque copia* della macchina è in uno stato accettante, l'NFA accetta la stringa di input. Si noti che questa divisione è descritta dall'insieme potenza $\mathcal{P}(Q)$, poiché da ogni stato si può arrivare ad un *insieme* di stati.

Gli automi deterministici vengono anche detti **DFA** (*Deterministic Finite Automaton*). Si noti che il determinismo è un caso particolare di non determinismo, dunque un DFA è sempre anche un NFA.

Esempio 1.2.1.1 (NFA)**.** Un esempio di NFA è il seguente:

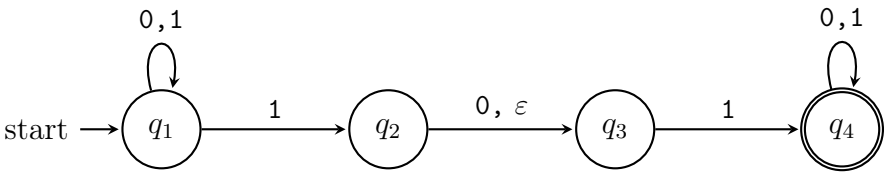


Figura 1.3: Un NFA.

esso può essere descritto secondo la quintupla $(Q, \Sigma, \delta, q_0, F)$ come segue:

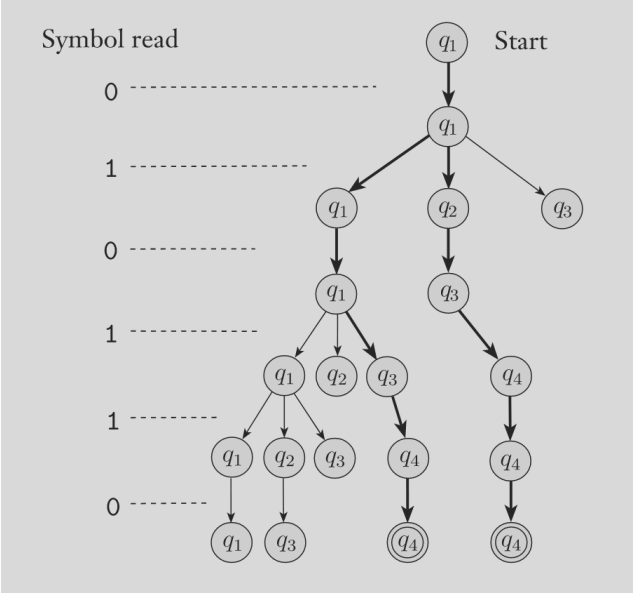
- $Q = \{q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$

- δ è la seguente:

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\varnothing$
q_2	$\{q_3\}$	$\varnothing$	$\{q_3\}$
q_3	$\varnothing$	$\{q_4\}$	$\varnothing$
q_4	$\{q_4\}$	$\{q_4\}$	$\varnothing$

- q_1 è lo stato iniziale
- $F = \{q_4\} \subseteq Q$

L'esecuzione dell'NFA mostrato è la seguente:



Si noti che nel momento in cui vengono incontrati ε-archi si giunge allo stato successivo nello stesso step dell'input incontrato, dunque gli ε-archi *non* producono uno step di esecuzione, ma vengono eseguiti non appena vengono incontrati, e vengono accorpati immediatamente.

Definizione 1.2.1.2: Stringhe accettate

Sia $N = (Q, \Sigma, \delta, q_0, F)$ un automa, e sia $w = w_1 \cdots w_n$ una stringa tale per cui $\forall i \in [1, n] \quad w_i \in \Sigma$; allora, M **accetta** w se esiste una sequenza di stati $r_0, \ldots, r_n \in Q$ tali per cui

- $r_0 = q_0$
- $\forall i \in [1, n-1] \quad r_{i+1} \in \delta(r_i, w_{i+1})$
- $r_n \in F$

Definizione 1.2.1.3: Equivalenza tra automi

Due macchine si dicono **equivalenti** se e solo se riconoscono lo stesso linguaggio.

Teorema 1.2.1.1: DFA equivalente ad NFA

Sia N un NFA; allora, esiste un DFA ad esso equivalente.

Dimostrazione. Sia $N = (Q, \Sigma_\varepsilon, \delta, q_0, F)$ l'NFA in ipotesi, tale da riconoscere un linguaggio A. Inoltre, si definisca

$$\forall k \geq 0 \quad E(R) := \bigcup_{r \in R} \delta^k(r, \varepsilon)$$

l'insieme degli stati raggiungibili da stati in R, applicando (anche ripetutamente) un numero arbitrario di ε-archi (si noti che per $k = 0$ si ha che $R \subseteq E(R)$).

Allora, sia $M = (Q', \Sigma_\varepsilon, \delta', q_0', F)$ il DFA definito come segue:

- $Q' := \mathcal{P}(Q)$, scelto tale da rappresentare ogni possibile stato di N;
- $\forall R \in Q', a \in \Sigma \quad \delta'(R, a) := \bigcup_{r \in R} E(\delta(r, a))$, scelta tale in quanto, per un certo insieme di stati $R \in Q'$ di N, a $\delta'(R, a)$ viene assegnata l'unione degli stati che sarebbero stati raggiunti in N dagli $r \in R$ con a, calcolati dunque attraverso $\delta(r, a)$, aggiungendo infine i possibili ε-archi;
- $q_0' := E(\{q_0\})$, scelto tale da far iniziare M esattamente dove aveva inizio N, comprendendo anche i possibili ε-archi iniziali;
- $F' := \{R \in Q' \mid \exists r \in R : r \in F\}$, che corrisponde all'insieme degli insiemi di stati di N contenenti almeno uno stato accettante in N.

Allora M è in grado di riconoscere A per costruzione, poiché il DFA costruito emula l'NFA di partenza, tenendo anche in considerazione gli ε-archi. Dunque, sia N che M riconoscono A, e per definizione sono di conseguenza equivalenti. □

Corollario 1.2.1.1: Linguaggi regolari con NFA

Sia A un linguaggio; allora, A è regolare se e solo se esiste un NFA che lo riconosce.

Dimostrazione.

Prima implicazione. Per definizione, A è regolare, dunque esiste un DFA che lo riconosce. Allora, poiché un DFA è sempre anche NFA, segue la tesi.

Seconda implicazione. Sia A un linguaggio tale da essere riconosciuto da un NFA; allora, per il Teorema 1.2.1.1, esiste un DFA equivalente all'NFA che riconosce A, e dunque per definizione A è regolare.

□

1.3 Operazioni regolari

1.3.1 Definizioni

Definizione 1.3.1.1: Unione

Siano A e B due linguaggi; allora, si definisce l'**unione** di A e B il seguente linguaggio:

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$

Esempio 1.3.1.1 (Unione)**.** Sia $\Sigma = \{\texttt{a}, \ldots, \texttt{z}\}$ l'alfabeto composto da 26 lettere, e siano $A = \{\texttt{uno}, \texttt{due}\}$ e $B = \{\texttt{tre}, \texttt{quattro}\}$ due linguaggi su Σ. Allora, si ha che

$$A \cup B = \{\texttt{uno}, \texttt{due}, \texttt{tre}, \texttt{quattro}\}$$

Proposizione 1.3.1.1: Chiusura dell'unione

Siano A e B due linguaggi regolari su un alfabeto Σ; allora $A \cup B$ è regolare.

Dimostrazione I. Per definizione, A e B sono linguaggi regolari, dunque esistono due automi finiti

$$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$$
$$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$$

tali da riconoscere rispettivamente A e B. Allora, sia $M = (Q, \Sigma, \delta, q_0, F)$ l'automa finito definito come segue:

- $Q := Q_1 \times Q_2 = \{(r_1, r_2) \mid r_1 \in Q_1 \wedge r_2 \in Q_2\}$, scelto tale in quanto permette di avere tutte le possibili combinazioni di stati dei due automi di partenza;
- $\forall (r_1, r_2) \in Q, a \in \Sigma \quad \delta((r_1, r_2), a) := (\delta_1(r_1, a), \delta_2(r_2, a))$, scelta tale in quanto permette di simulare entrambi gli automi di partenza contemporaneamente, mandando ogni stato di M_1 ed M_2 dove sarebbe andato nei rispettivi automi di appartenenza;
- $q_0 := (q_1, q_2)$, scelto tale in quanto deve essere lo stato in cui entrambe gli automi in ipotesi iniziavano;
- $F := (F_1 \times Q_2) \cup (Q_1 \times F_2) = \{(r_1, r_2) \mid r_1 \in F_1 \vee r_2 \in F_2\}$, scelto tale in quanto permette di simulare gli stati accettanti di entrambi gli automi, e vanno prese tutte le coppie che vedono almeno uno dei due stati come accettanti poiché altrimenti non si accetterebbero delle stringhe accettate da M_1 ed M_2 in partenza.

Allora, poiché M è in grado di simulare M_1 ed M_2 contemporaneamente, per costruzione accetterà ogni stringa di A e di B, dunque riconoscendo $A \cup B$, e di conseguenza $A \cup B$ è regolare per definizione. □

Dimostrazione II. Per definizione, A e B sono linguaggi regolari, dunque per il Teorema 1.2.1.1 esistono due NFA

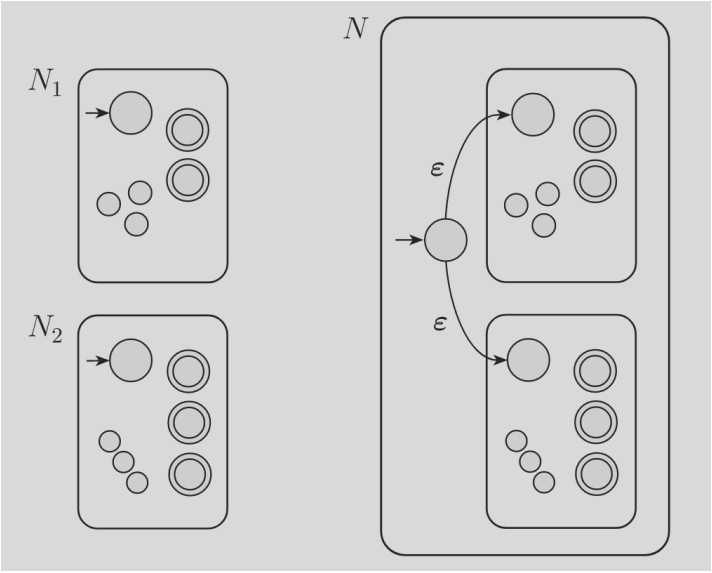
$$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$$
$$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$$

tali da riconoscere rispettivamente A e B. Allora, sia $N = (Q, \Sigma, \delta, q_0, F)$ l'NFA costruito come segue:

- $Q := \{q_0\} \cup Q_1 \cup Q_2$, dove q_0 è un nuovo stato, e Q è scelto tale da includere sia N_1 che N_2;
- $\forall q \in Q, a \in \Sigma_\varepsilon \quad \delta(q,a) := \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \wedge a = \varepsilon \\ \varnothing & q = 0 \wedge a \neq \varepsilon \end{cases}$, scelta tale da poter eseguire contemporaneamente gli NFA N_1 ed N_2, definendo per casi la funzione di transizione; infatti, si noti che si è posta $\delta(q_0, \varepsilon) := \{q_1, q_2\}$ in modo da collegare il nuovo stato q_0 a q_1 e q_2, gli stati iniziali di N_1 ed N_2 rispettivamente;
- q_0 è il nuovo stato, che rappresenta lo stato iniziale di N;
- $F := F_1 \cup F_2$, scelto tale da costruire N in modo che accetti una stringa se e solo se la accetterebbero N_1 o N_2.

Allora, l'NFA risultante N è in grado di computare contemporaneamente N_1 ed N_2, ed è dunque in grado di riconoscere A e B contemporaneamente; di conseguenza, N riconosce $A \cup B$, che risulta dunque essere regolare per definizione.

La seguente rappresentazione raffigura la dimostrazione costruttiva appena descritta:



□

Definizione 1.3.1.2: Concatenazione

Siano A e B due linguaggi; allora, si definisce la **concatenazione** di A e B il seguente linguaggio:

$$A \circ B = \{xy \mid x \in A \wedge y \in B\}$$

Esempio 1.3.1.2 (Concatenazione)**.** Sia $\Sigma = \{\texttt{a}, \ldots, \texttt{z}\}$ l'alfabeto composto da 26 lettere, e siano $A = \{\texttt{uno}, \texttt{due}\}$ e $B = \{\texttt{tre}, \texttt{quattro}\}$ due linguaggi su Σ. Allora, si ha che

$$A \circ B = \{\texttt{unotre}, \texttt{unoquattro}, \texttt{duetre}, \texttt{duequattro}\}$$

Proposizione 1.3.1.2: Chiusura della concatenazione

Siano A e B due linguaggi regolari su un alfabeto Σ; allora $A \circ B$ è regolare.

Dimostrazione. Per definizione, A e B sono linguaggi regolari, dunque per il Teorema 1.2.1.1 esistono due NFA

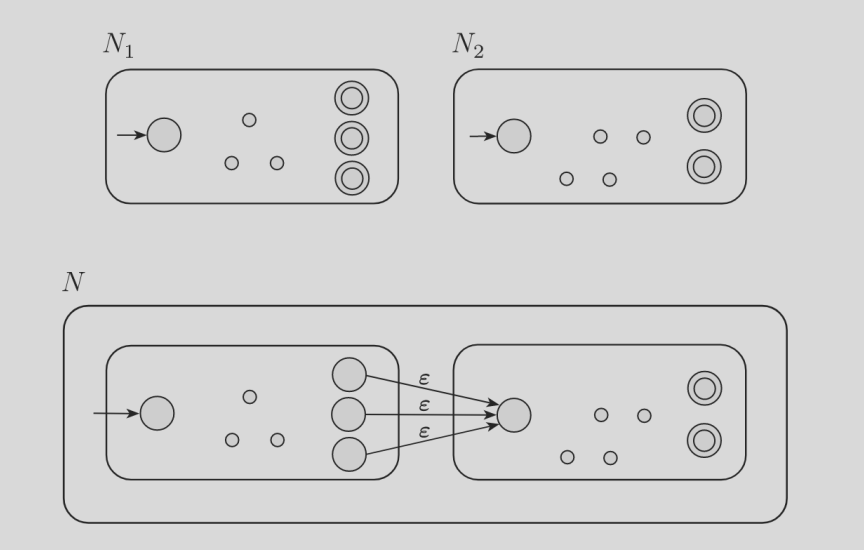
$$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$$
$$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$$

tali da riconoscere rispettivamente A e B. Allora, sia $N = (Q, \Sigma, \delta, q_0, F)$ l'NFA costruito come segue:

- $Q := Q_1 \cup Q_2$, scelto tale da includere entrambe gli automi N_1 ed N_2 di partenza;
- $q_0 := q_1$, scelto tale da far iniziare l'esecuzione dell'automa su N_1;
- $F := F_2$, scelto tale da far terminare l'esecuzione dell'automa su N_2;
- $\forall q \in Q, a \in \Sigma_\varepsilon \quad \delta(q,a) := \begin{cases} \delta_1(q,a) & q \in Q_1 - F_1 \vee (q \in F_1 \wedge a \neq \varepsilon) \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \wedge a = \varepsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$,
  scelta tale da anteporre l'esecuzione di N_1 a quella di N_2; infatti, se $q \in Q_1 - F_1$ (è uno stato non accettante di N_1), oppure $q \in F_1$ ma $a \neq \varepsilon$, l'esecuzione di N_1 non viene alterata; diversamente, se invece $q \in F_1$ e $a = \varepsilon$, all'insieme di stati $\delta_1(q, \varepsilon)$ viene aggiunto q_2, ovvero lo stato iniziale di N_2, in modo da effettuare la concatenazione tra i due NFA non deterministicamente.

Allora, l'NFA N costruito computa inizialmente N_1, e se vengono raggiunti suoi stati accettanti, l'esecuzione prosegue attraverso N_2, al fine di realizzare la concatenazione tra le stringhe. Di conseguenza, l'automa è in grado di riconoscere $A \circ B$ per costruzione, e dunque $A \circ B$ è regolare per definizione.

La seguente rappresentazione raffigura la dimostrazione costruttiva appena descritta:



□

Definizione 1.3.1.3: Star

Sia A un linguaggio; allora, si definisce l'operazione unaria **star** che definisce il seguente linguaggio:

$$A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0 \wedge \forall i \in [1,k] \quad x_i \in A\}$$

Si noti che $k = 0 \implies \varepsilon \in A^*$ per ogni linguaggio A.

Esempio 1.3.1.3 (Star)**.** Sia $\Sigma = \{\texttt{a}, \ldots, \texttt{z}\}$ l'alfabeto composto da 26 lettere, e siano $A = \{\texttt{uno}, \texttt{due}\}$ e $B = \{\texttt{tre}, \texttt{quattro}\}$ due linguaggi su Σ. Allora, si ha che

$$A^* = \{\varepsilon, \texttt{uno}, \texttt{due}, \texttt{unouno}, \texttt{unodue}, \texttt{dueuno}, \texttt{duedue}, \ldots\}$$

Proposizione 1.3.1.3: Chiusura dell'operazione star

Sia A un linguaggio regolare su un alfabeto Σ; allora A^* è regolare.

Dimostrazione. Per definizione, A è linguaggio regolare, dunque per il Teorema 1.2.1.1 esiste un NFA

$$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$$

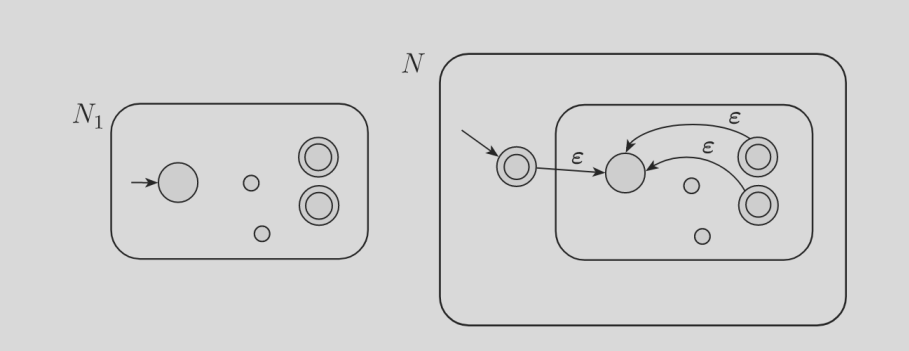
tale da riconoscere A. Allora, sia $N = (Q, \Sigma, \delta, q_0, F)$ l'NFA costruito come segue:

- $Q := \{q_0\} \cup Q_1$, dove q_0 è un nuovo stato, posto prima di N_1;
- q_0 è il nuovo stato iniziale;
- $F := \{q_0\} \cup F_1$, poiché q_0 deve essere accettante, in modo tale da accettare ε (si noti che $\varepsilon \in A^*$ per ogni linguaggio A);

- $\forall q \in Q, a \in \Sigma_\varepsilon \quad \delta(q,a) := \begin{cases} \delta_1(q,a) & q \in Q_1 - F_1 \vee (q \in F_1 \wedge a \neq \varepsilon) \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \wedge a = \varepsilon \\ \{q_1\} & q = q_0 \wedge a = \varepsilon \\ \varnothing & q = q_0 \wedge a \neq \varepsilon \end{cases}$,
  scelta tale da ricominciare l'esecuzione dell'automa ogni volta che viene raggiunto uno stato accettante in N_1; infatti, se $q \in Q_1 - F_1$ (è uno stato non accettante di N_1), oppure q è accettante e $a \neq \varepsilon$, l'esecuzione procede normalmente con $\delta_1(q,a)$; differentemente, se $q \in F_1$ ma $a = \varepsilon$, allora l'esecuzione deve ricominciare da capo per poter effettuare la concatenazione multipla delle stringhe in A che caratterizzano l'operazione star, e dunque a $\delta_1(q,a)$ viene aggiunto q_1 (lo stato inziale di N_1); infine, ponendo $\delta(q_0,\varepsilon) := \{q_1\}$ si realizza l'ε-arco iniziale che collega q_0 (il nuovo stato) a q_1, al fine di rendere N in grado di accettare ε.

Allora, poichè l'NFA N è in grado di ricominciare l'esecuzione ogni volta che questa sarebbe terminata in N_1, è in grado di accettare molteplici copie concatenate delle stringhe in A, in maniera non deterministica, e dunque per definizione N riconosce A^*, il quale risulta allora essere regolare per definizione.

La seguente rappresentazione raffigura la dimostrazione costruttiva appena descritta:



□

1.4 Espressioni regolari

1.4.1 TODO

Definizione 1.4.1.1: Espressione regolare

TODO