Esercizi Fondamenti dell'Informatica

federico.serafini@studenti.unipr.it

2021/2022

Sommario

Lo scopo principale di questo elaborato è di fornire alcune possibili soluzioni per una buona parte degli esercizi incontrati durante il corso di Fondamenti dell'Informatica. L'autore di questo testo è uno studente che ha cercato di seguire quanto più possibile *the KISS principle* con l'intenzione di rivolgersi ad un pubblico di altri studenti.

Attenzione: è possibile che ci siano alcuni errori e/o imprecisioni, per uno studio adeguato della materia è fondamentale riferirsi principalmente al materiale fornito dal docente e ad i libri di testo consigliati[1].

[1] A. Dovier, R. Giacobazzi. Fondamenti dell'informatica: Linguaggi formali, calcolabilità e complessità. Bollati Boringhieri, ISBN9788833933795, 2020.

Indice

1 Nozioni Preliminari

1.1 Definizioni, Teoremi e Lemmi

Definizione 1.1 (Cardinalità di un insieme)**.** *Sia S un insieme, la cardinalità di S si indica con $|S|$ e corrisponde al numero di elementi in S.*

Esempio. Sia $S = \{0, 1\}$, $|S| = 2$.

Definizione 1.2 (Insieme enumerabile)**.** *Sia S un insieme, S è enumerabile se è possibile stabilire una corrispondenza biunivoca tra gli elementi di S e gli elementi dell'insieme dei numeri naturali $\mathbb{N}$.*

Enumerare un insieme significa quindi stabilire un primo, secondo, terzo, etc. elemento. Notare che, essendo $\mathbb{N}$ un insieme infinito, se un insieme S è enumerabile allora S è anche un insieme infinito.

Definizione 1.3 (Cardinalità insiemi enumerabili)**.** *La cardinalità degli insiemi enumerabili si indica con $\aleph_0$ (aleph zero).*

Sia S un insieme, allora:

- S è finito se $|S| < \aleph_0$,
- S è enumerabile se $|S| = |\mathbb{N}| = \aleph_0$.

In quanto informatici, siamo interessati agli insiemi enumerabili perché i dati in informatica, essendo rappresentati da numeri, sono enumerabili.

Definizione 1.4 (Insieme delle parti)**.** *Sia S un insieme, l'*insieme delle parti *o* insieme potenza *di S si indica con $\wp(S)$ ed è l'insieme formato da tutti i sottoinsiemi di S.*

Esempio. Sia $S = \{0, 1\}$, allora $\wp(S) = \big\{ \varnothing, \{0\}, \{1\}, \{0, 1\} \big\}$.

Teorema 1.1 (Cantor)**.** *Sia S un insieme, allora*

$$|S| < |\wp(S)|,$$

$$|\wp(S)| = 2^{|S|},$$

$$|\mathbb{N}| = \aleph_0 < |\wp(\mathbb{N})| = |\mathbb{R}|.$$

1.2 Principio di Induzione Matematica

Definizione induttiva. La definizione per induzione viene utilizzata quando si vuole definire un "qualcosa" Q che è parametrico sui naturali. In generale, la definizione è composta da *caso base* e *passo induttivo*.

Caso base. Definisce il valore v_0 che assume Q nel caso base, ovvero il valore che assume Q con il parametro 0:

$$Q(0) = v_0.$$

Passo induttivo. Sia $n \in \mathbb{N}$. Il passo induttivo definisce il valore di $Q(n+1)$ in funzione di $Q(n)$ che è ipotizzato vero:

$$Q(n) = v_n \Longrightarrow Q(n+1) = f(v_n).$$

"Se è vero che $Q(n) = v_n$ allora definiamo $Q(n+1)$ come una certa funzione f applicata a $v_n = Q(n)$".

Esempio. Cos'è un numero naturale? Definiamo l'insieme $\mathbb{N}$ dei numeri naturali definendo in maniera induttiva gli elementi che vi fanno parte.

Caso base: $0 \in \mathbb{N}$.

Passo induttivo: $\forall n \in U : n \in \mathbb{N} \Longrightarrow n + 1 \in \mathbb{N}$.

Nel caso base della definizione induttiva di $\mathbb{N}$ diciamo che è sempre vero che 0 è un numero naturale, il caso base non fa riferimento a nessun'altra nozione, ovvero $0 \in \mathbb{N}$ è un *assioma*.

Il passo induttivo è tipicamente in forma di implicazione: se un elemento n appartiene ad $\mathbb{N}$, allora anche il suo successore appartiene ad $\mathbb{N}$. Se conosciamo il valore per n possiamo ricavare il valore anche di $n + 1$, in altre parole, $n + 1$ è definito in termini di n.

Un'altra possibile notazione utilizzata per la definizione di implicazioni è la seguente, in cui si utilizza una linea orizzontale per separare le *precondizioni* o *premesse* dalle *conclusioni*:

$$\text{Caso base: } \frac{}{0 \in \mathbb{N}}$$

$$\text{Passo induttivo: } \frac{n \in \mathbb{N}}{n + 1 \in \mathbb{N}}$$

Dimostrazione per induzione. Una dimostrazione per induzione si utilizza quando si vuole mostrare che una certa proprietà P vale per tutti i numeri naturali. Si procede mostrando che la proprietà è vera per il caso base. Poi, ipotizzando che la proprietà P vale per un generico $n \in \mathbb{N}$, si dimostra che è vera anche per $n + 1$.

$$\text{Caso base: } \frac{}{P(0)}$$

$$\text{Passo induttivo: } \frac{P(n)}{P(n+1)}$$

1. *Caso base*: $P(0)$, mostriamo che P vale nel caso base indicato con 0;

2. *Passo induttivo*:
$$\underbrace{P(n)}_{\textit{antecedente}} \overset{?}{\Longrightarrow} \underbrace{P(n+1)}_{\textit{conseguente}}.$$
Il passo induttivo è tipicamente in forma di implicazione e la proposizione antecedente prende il nome di *ipotesi induttiva* proprio perché viene ipotizzata vera. A questo punto si deve mostrare che vale l'implicazione; ipotizzando vero l'antecedente è necessario mostrare che è vero anche il conseguente.

Quella vista fino ad ora viene detta *induzione matematica semplice*. Esistono anche *induzione matematica completa* ed *induzione strutturale*.

Induzione Matematica completa. L'induzione matematica completa ha una ipotesi induttiva più forte: dice che una certa proprietà $P(m)$ vale per m e anche per tutti i suoi predecessori, quindi da 0 a m compreso. Non è più potente dell'induzione semplice ma a volte è più conveniente da utilizzare.

Induzione strutturale. Il principio su cui si base l'induzione strutturale è il seguente: *per mostrare che una proprietà P è vera per tutte le espressioni, è sufficiente mostrare che è vera per tutte le espressioni atomiche ed è preservata da tutti i metodi utilizzati per generare nuove espressioni a partire da quelle esistenti.*

Proseguire nella lettura per esempi di induzione semplice, completa e strutturale.

2 Alfabeti e Linguaggi

2.1 Definizioni, Teoremi e Lemmi

Definizione 2.1 (Simbolo)**.** *Un simbolo è un'entità primitiva astratta non meglio definita.*

Esempio. Tra i simboli troviamo ad esempio: lettere, caratteri numerici, ideogrammi, bandiere per segnalazioni nautiche.

Notazione. Utilizzeremo a, b, c, d (prime lettere del nostro alfabeto) per indicare simboli.

Definizione 2.2 (Alfabeto)**.** *Un alfabeto (spesso indicato con* Σ*) è un insieme* finito *di simboli.*

La finitezza dell'alfabeto è necessaria per poter distinguere i simboli gli uni con gli altri in tempo finito e con una quantità finita di memoria cosa che non sarebbe possibile se l'alfabeto preso in considerazione fosse infinito.

Esempio. $\Sigma_{bin} = \{ 0, 1 \}, \Sigma_{\text{hexa}} = \{ 1, 2, \ldots, 9, A, B, \ldots, F \}$ sono alfabeti.

Definizione 2.3 (Definizione informale di stringa)**.** *Una stringa è una sequenza finita di simboli giustapposti (posti uno dietro l'altro).*

Esempio. Se a, b, c sono simboli, $abcba$ è una stringa.

Osservazione 2.1 (Definizione formale di stringa)**.** *E' possibile definire formalmente una stringa di lunghezza* $n \in \mathbb{N}$ *per mezzo di una funzione* parziale $x : \{ 0, \ldots, n-1 \} \rightarrowtail \Sigma$ *definita come*

$$x(i) := \begin{cases} a_i, & \textit{se } 0 \leq i < n, x = a_0 \cdots a_{n-1}; \\ \uparrow, & \textit{altrimenti.} \end{cases}$$

Definizione 2.4 (Definizione informale di lunghezza di una stringa)**.** *Sia* x *una generica stringa, la sua lunghezza si indica con* $|x|$ *e corrisponde al numero di occorrenze dei simboli che la compongono. C'è una sola stringa di lunghezza* 0*, la stringa vuota* ϵ*.*

Esempio. Sia la stringa $x = aba$. $|x| = |aba| = 3$.

Osservazione 2.2 (Definizione formale di lunghezza di una stringa)**.** *E' possibile definire formalmente la lunghezza di una stringa per mezzo di una funzione totale. Sia x una stringa. $|x| : \Sigma^* \mapsto \mathbb{N}$ è definita come*

$$|x| := \begin{cases} 0, & \text{se } \forall i \in \mathbb{N} : x(i) =\uparrow; \\ 1 + \max\{\, i \in \mathbb{N} \mid x(i) =\downarrow \,\} & \text{altrimenti;} \end{cases}$$

Notare che come dominio è stato messo Σ^*. Dal momento che l'operazione "lunghezza di una stringa" prende come parametri stinghe, possiamo già intuire che Σ^* è un insieme di stringhe. Proseguire nella lettura per ulteriori approfondimenti.

Osservazione 2.3 (Definizione ricorsiva della lunghezza di una stringa)**.** *Sia $x \in \Sigma^*$, un'altra definizione formale (equivalente) per la funzione "lunghezza di una stringa" $|x| : \Sigma^* \mapsto \mathbb{N}$ è la seguente:*

$$|x| := \begin{cases} 0, & \text{se } x = \epsilon; \\ 1 + |y|, & \text{se } x = ay. \end{cases}$$

Definizione 2.5 (Prefisso, suffisso, sottostringa)**.** *Sia $x = a_1 \cdots a_n$ una stringa di lunghezza n:*

- *prefisso: $a_1 \cdots a_i$, con $1 \leq i \leq n$;*
- *suffisso: $a_i \cdots a_n$, con $1 \leq i \leq n$;*
- *sottostringa: $a_i \cdots a_j$, con $1 \leq i \leq j \leq n$;*
- *ϵ è prefisso, suffisso e sottostringa di ogni stringa, compresa di se stessa.*

Quando i prefissi o suffissi di una stringa non *sono uguali alla stringa stessa, vengono detti* propri.

Esempio. Sia abc una stringa, i suoi prefissi sono ϵ, a, ab, abc, mentre i suffissi sono ϵ, c, bc, abc,

Definizione 2.6 (Definizione informale di concatenazione)**.** *Siano x, y stringhe, la loro concatenazione si indica con $x \cdot y$ ed è la stringa ottenuta facendo seguire alla prima la seconda.*

Come accade per la moltiplicazione nell'artimetica, spesso il '$\cdot$' viene omesso e la concatenazione tra due stringhe x e y viene semplicemente indicata con xy.

Osservazione 2.4 (Definizione formale di concatenazione)**.** *Siano* $x, y \in \Sigma^*$. *E' possibile definire formalmente la concatenazione tra stringhe* xy *per mezzo di una funzione parziale* $(xy) : \{\ 0, \dots, |x + y| - 1\} \rightarrowtail \mathbb{N}$ *definita come*

$$(xy)(i) := \begin{cases} x(i), & \textit{se } 0 \leq i < |x|; \\ y(i - |x|), & \textit{se } |x| \leq i < |xy|; \\ \uparrow, & \textit{altrimenti.} \end{cases}$$

Osservazione 2.5 (Definizione induttiva di concatenazione)**.** *Siano* $x, y \in \Sigma^*$, *è possibile definire la concatenazione* xy *per induzione strutturale su* y.

$$\textit{Caso base: } \frac{y = \epsilon}{x\epsilon = x}$$

$$\textit{Passo induttivo: } \frac{y = wa}{x(wa) = (xw)a}$$

Lemma 2.1 (Proprietà associativa della concatenazione)**.** *Siano* x, y, z *tre stringhe, allora*

$$(xy)z = x(yz).$$

Dimostrazione. Siano $x, y, z \in \Sigma^*$. Mostreremo che $(xy)z = x(yz)$ per induzione sulla strutturale su z utilizzeremo quindi la definizione induttiva della concatenazione.

Caso base: se $z = \epsilon$ abbiamo che

$$\begin{aligned} (xy)z &= (xy)\epsilon && [z = \epsilon] \\ &= xy && [\text{def. concat. (caso base)}] \\ &= x(y\epsilon) && [\text{def. concat. (caso base)}]. \end{aligned}$$

Implicazione: siano $u, v, w \in \Sigma^*$,

$$(uv)w = u(vw) \overset{?}{\Longrightarrow} (uv)wa = u(vwa).$$

Passo induttivo: se $z = wa$ abbiamo che

$$
\begin{aligned}
(xy)z &= (xy)(wa) && [z = wa] \\
&= \big((xy)w\big)a && [\text{def. concat.}] \\
&= \big(x(yw)\big)a && [\text{ip. ind.}] \\
&= x\big((yw)a\big) && [\text{def. concat.}] \\
&= x\big(y(wa)\big) && [\text{def. concat.}] \\
&= x(yz) && [z = wa].
\end{aligned}
$$

□

Dalla validità del lemma segue che

$$(xy)z = x(yz) = xyz.$$

Lemma 2.2 (Identità della concatenazione)**.** *Sia x una stringa, allora*

$$\epsilon x = x = x\epsilon.$$

Dimostrazione. Sia x una generica stringa e $i \in \mathbb{N}$. Mostreremo che

$$(\epsilon x)(i) = x(i) = (x\epsilon)(i).$$

$$
\begin{aligned}
(\epsilon x)(i) &= \begin{cases} \epsilon(i), & \text{se } 0 \le i < |\epsilon|; \\ x(i - |\epsilon|), & \text{se } |\epsilon| \le i < |\epsilon| + |x|; \\ \uparrow, & \text{altrimenti.} \end{cases} && [\text{def. concat.}] \\
&= \begin{cases} \epsilon(i), & \text{se } 0 \le i < 0; \\ x(i - 0), & \text{se } 0 \le i < 0 + |x|; \\ \uparrow, & \text{altrimenti.} \end{cases} && [|\epsilon| = 0] \\
&= \begin{cases} x(i), & \text{se } 0 \le i < |x|; \\ \uparrow, & \text{altrimenti.} \end{cases} && [\nexists i \in \mathbb{N} \,.\, 0 \le i < 0] \\
&= x(i). && [\text{def. stringa}].
\end{aligned}
$$

$$
\begin{aligned}
(x\epsilon)(i) &= \begin{cases} x(i), & \text{se } 0 \le i < |x|; \\ \epsilon(i-|x|), & \text{se } |x| \le i < |x|+|\epsilon|; \\ \uparrow, & \text{altrimenti.} \end{cases} && \text{[def. concat.]} \\
&= \begin{cases} x(i), & \text{se } 0 \le i < |x|; \\ \uparrow, & \text{altrimenti.} \end{cases} && [\forall i \in \mathbb{N} : \epsilon(i) = \uparrow] \\
&= x(i) && \text{[def. stringa].}
\end{aligned}
$$

□

Definizione 2.7 (Stringhe su un alfabeto). *Sia Σ un alfabeto, le stringhe su Σ sono le stringhe di lunghezza $n \in \mathbb{N}$ (arbitraria ma finita) formate dalla giustapposizione dei simboli che compaiono in Σ, dunque una generica stringa x su Σ è definita come:*

$$x = a_1 a_2 \cdots a_n, \ \textit{ con } a_i \in \Sigma, n \ge 0.$$

Definizione 2.8 (Insieme di tutte le stringhe su un alfabeto). *Sia Σ un alfabeto, l'insieme di tutte le stringhe su Σ, denotato con Σ^* è dunque:*

$$\Sigma^* := \{\, a_1 a_2 \cdots a_n \mid a_i \in \Sigma, n \ge 0 \,\} = \bigcup_{n \in \mathbb{N}} \{\, 0, \ldots, n-1 \,\} \rightarrowtail \Sigma.$$

Siano Σ un alfabeto e $n \in \mathbb{N}$, è possibile dare una definizione di forma induttiva per Σ^n:

$$\Sigma^n := \begin{cases} \{\, \epsilon \,\}, & \textit{se } n = 0; \\ \{\, a \cdot x \mid a \in \Sigma, x \in \Sigma^{n-1} \,\}, & \textit{altrimenti.} \end{cases}$$

Osservazione 2.6. *Notare che Σ^n è l'insieme formato da tutte le stringhe di lunghezza n su Σ.*

Dimostrazione. Vogliamo mostrare che:

$$\forall n \in \mathbb{N}, \forall x \in \Sigma^n : |x| = n.$$

Procediamo quindi per induzione matematica semplice sull'esponente n di Σ e di conseguenza sulla lunghezza delle stringhe, con caso base $n = 0$ e passo induttivo $n = m+1$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (antecedente dell'implicazione).

Caso base: Se $n = 0$ dobbiamo mostrare che $|x| = 0$.

$$
\begin{aligned}
\Sigma^n &= \Sigma^0 && [n = 0] \\
&= \{ \epsilon \} && [\text{def. } \cdot^0].
\end{aligned}
$$

Sia $x \in \Sigma^n$, allora $x = \epsilon, |x| = 0 = n$ ✓.

Implicazione: sia $m \in \mathbb{N}$,

$$\forall u \in \Sigma^m : |x| = m \overset{?}{\Longrightarrow} \forall v \in \Sigma^{m+1} : |v| = m + 1.$$

Passso induttivo: se $n = m + 1$ abbiamo che

$$
\begin{aligned}
\Sigma^n &= \Sigma^{m+1} && [n = m + 1] \\
&= \{ ax \mid a \in \Sigma, x \in \Sigma^m \} && [\text{def. } \cdot^n].
\end{aligned}
$$

Sia $y \in \Sigma^{m+1}$, dunque $y = aw$ con $a \in \Sigma, w \in \Sigma^m$.

$$
\begin{aligned}
|y| &= |a| + |w| && [y = aw] \\
&= 1 + |w| && [a \in \Sigma] \\
&= 1 + m \checkmark && [w \in \Sigma^m, \text{ ip. ind.}].
\end{aligned}
$$

□

Secondo quanto appena visto, è possibile dare un'altra definizione per Σ^*:

$$\Sigma^* := \bigcup_{n \in \mathbb{N}} \Sigma^n.$$

Esempio. Siano $\Sigma = \{ 0 \}, n = 3$, allora

$$\Sigma^n = \{ 000 \}, \Sigma^* = \{ \epsilon, 0, 00, 000, 0000, \dots \}.$$

Siano $\Sigma = \{ 0, 1 \}, n = 2$, allora

$$\Sigma^n = \{ 00, 01, 10, 11 \}, \Sigma^* = \{ \epsilon, 0, 1, 00, 01, 11, 000, \dots \}.$$

Definizione 2.9 (Linguaggio formale)**.** *Sia Σ un alfabeto, un linguaggio formale (spesso indicato con L), è un* insieme *formato dalle stringhe su Σ, quindi vale che:*

$$L \subseteq \Sigma^*.$$

Notare che:

- Σ^* è esso stesso un linguaggio, il linguaggio formato da *tutte* le stringhe su Σ;
- essendo $\varnothing$ sottoinsieme di qualsiasi altro insieme, $\varnothing \subseteq \Sigma^*$ dunque $\varnothing$ è un linguaggio su qualsiasi alfabeto;
- essendo ϵ la stringa nulla su qualsiasi alfabeto Σ, $\{\, \epsilon \,\} \subseteq \Sigma^*$ dunque $\{\, \epsilon \,\}$ è un linguaggio su qualsiasi alfabeto.

Osservazione 2.7. *Sia Σ un alfabeto. Sia $L \subseteq \Sigma^*$ il linguaggio formato dall'insieme di tutte le stringhe di lunghezza 1 su Σ, allora*

$$L \cong \Sigma, \ \textit{(L e } \Sigma \textit{ sono } \text{isomorfi}\textit{)}$$

ed è possibile dare una definizione induttiva di Σ^n equivalente a quella già vista in precedenza:

$$\textit{Caso base: } \frac{}{\Sigma^0 = \{\, \epsilon \,\}}$$

$$\textit{Passo induttivo: } \frac{\Sigma^n = L}{\Sigma^{n+1} = \Sigma \cdot L}$$

da cui segue da definizione di concatenazione di due linguaggi L, M

$$L \cdot M = \{\, x \cdot y \mid x \in L, y \in M \,\}.$$

2.2 Esercizi

Esercizio 2.1

Definire i predicati $\text{prefix}(x, y), \text{suffix}(x, y), \text{substr}(x, y)$ su stringhe.

Soluzione.

$$\text{prefix}(x, y) \begin{cases} \textbf{true,} & \text{se } x = \epsilon; \\ \textbf{true,} & \text{se } \forall i, 0 \leq i < |x| : x(i) = y(i); \\ \textbf{false,} & \text{altrimenti.} \end{cases}$$

$$\text{suffix}(x, y) \begin{cases} \textbf{true,} & \text{se } x = \epsilon; \\ \textbf{true,} & \text{se } \forall i, 0 \leq i < |x| : x(i) = y(i + k), k = |y| - |x|; \\ \textbf{false,} & \text{altrimenti.} \end{cases}$$

$$\text{substr}(x,y)\begin{cases}\textbf{true,} & \text{se } x=\epsilon;\\ \textbf{true,} & \text{se } \exists k, 0 \le k < |y| \;.\; \forall i, 0 \le i < |x| : x(i) = y(i+k);\\ \textbf{false,} & \text{altrimenti.}\end{cases}$$

Esercizio 2.2

Sia x una stringa con $|x| = n$. Qual'è il numero di prefissi, suffissi e sottostringhe di x? Quanti sono i prefissi dei suoi suffissi? E i suffissi dei suoi prefissi?

Soluzione. Scegliamo una stringa di esempio $x = abc$ ed elenchiamo i prefissi:

$$\epsilon, a, ab, abc.$$

Con $|x| = 3$ otteniamo 4 prefissi, con una generica stringa s di lunghezza n otteniamo $|s| + 1$ prefissi. Ragionamento analogo si applica ai suffissi. Per quanto riguarda le sottostringhe dobbiamo ragionare un po' di più, elenchiamo le sottostringhe:

- ϵ (1 sottostringa di lunghezza 0);
- abc (1 sottostringa di lunghezza 3);
- ab, bc (2 sottostringhe di lunghezza 2);
- a, b, c (3 sottostringhe di lunghezza 1).

Notiamo che, fatta eccezione per la sottostringa ϵ, il numero di sottostringhe trovate diminuisce quando la lunghezza delle sottostringhe cercate aumenta. Il numero di sottostringhe di una stringa di lunghezza 3 é $1 + 1 + 2 + 3$. Generalizzando ad una stringa x di lunghezza n, il numero delle sue sottostringhe è la somma dei numeri naturali da 1 a n, incrementata di 1, ovvero, se chiamiamo substr(x) l'insieme formato da tutte e sole le sottostringhe di x abbiamo:

$$|\text{substr}(x)| = 1 + \sum_{i=1}^{|x|} i = 1 + \frac{|x|(|x|+1)}{2} \qquad \text{[Gauss]}.$$

Questa è la nostra ipotesi, proviamo a dimostrarne la validità con una dimostrazione formale per induzione.

Dimostrazione. Vogliamo mostrare che

$$\forall n \in \mathbb{N}, \forall x \in \Sigma^n : |\mathrm{substr}(x)| = 1 + \sum_{i=1}^{n} i$$

Dimostreremo per induzione matematica semplice sulla lunghezza della stringa n con caso base $n = 0$ e passo induttivo $n = m + 1$ con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (antecedente dell'implicazione).

Caso base: se $n = 0 = |x|$ abbiamo $x = \epsilon$. La stringa ϵ ha solo se stessa come sottostringa quindi

$$\begin{aligned}
|\mathrm{substr}(x)| &= |\mathrm{substr}(\epsilon)| && [n = 0 = |x|] \\
&= 1 \\
&= 1 + \sum_{i=1}^{|x|} i \checkmark && [|x| = n = 0].
\end{aligned}$$

Implicazione: sia $m \in \mathbb{N}$,

$$\forall u \in \Sigma^m : |\mathrm{subs}(u)| = 1 + \sum_{i=1}^{m} i \overset{?}{\Longrightarrow} \forall v \in \Sigma^{m+1} : |\mathrm{subs}(v)| = 1 + \sum_{i=1}^{m+1} i.$$

Passo induttivo: se $n = m + 1 = |y|$ abbiamo $y = wa$, con $|w| = m$ e vale che

$$
\begin{aligned}
|\text{substr}(y)| &= |\text{substr}(wa)| && [y = wa] \\
&= |\text{substr}(w)| + |\text{suffix}(wa)| - 1 && [-1 \text{ rimuove } \epsilon \text{ duplicato}] \\
&= 1 + \sum_{i=1}^{m} i + |\text{suffix}(wa)| - 1 && [|w| = m, \text{ ip. ind.}] \\
&= 1 + \frac{m(m+1)}{2} + |\text{suffix}(wa)| - 1 && [\text{Gauss}] \\
&= 1 + \frac{m(m+1)}{2} + \big((m+1)+1\big) - 1 && [|wa| = m+1] \\
&= 1 + \frac{m(m+1)}{2} + m + 1 \\
&= 1 + \frac{m(m+1) + 2m + 2}{2} && [m.c.m] \\
&= 1 + \frac{m^2 + m + 2m + 2}{2} \\
&= 1 + \frac{(m+1)(m+2)}{2} \\
&= 1 + \sum_{i=1}^{m+1} i \checkmark && [\text{Gauss}].
\end{aligned}
$$

□

Quanti sono i suffissi dei prefissi e i prefissi dei suffissi? Prendiamo una stringa di esempio $x = abc$ con lunghezza $n = 3$. Elenchiamo i suoi prefissi ϵ, a, ab, abc con il relativo numero di suffissi (che sappiamo essere $|x| + 1$):

- abc : $3 + 1$ suffissi;
- ab : $2 + 1$ suffissi;
- a : $1 + 1$ suffissi;
- ϵ : $0 + 1$ suffissi.

Sommando i “+1” di ogni riga otteniamo $|x| + 1$ perchè il numero di righe è proprio il numero di prefissi di x che è $|x| + 1$. Ciò che rimane è una somma da 0 a $|x|$. Quindi, il numero di suffissi dei prefissi di una stringa x é

$$|x| + 1 + \sum_{i=1}^{|x|} i = \sum_{i=1}^{|x|+1} i.$$

Dimostrare formalmente per esercizio.

Esercizio 2.3

Sia $\Sigma \neq \varnothing$ un alfabeto, qual'è la cardinalità di Σ^n? Dimostrare la propria affermazione.

Soluzione. Lavoriamo ad esempio con le stringhe binarie, quindi sia $\Sigma = \{ 0, 1, \}$ il nostro alfabeto di riferimento, quindi $|\Sigma| = 2$. Con n bit disponibili (quindi stringhe lunghe n) possiamo contare da 0 fino a $2^n - 1$, un totale di 2^n elementi.

Preso un generico $\Sigma \neq \varnothing$, vediamo la lunghezza n delle stringhe come i bit disponibili, quindi con $|\Sigma|$ simboli possiamo ottenere $|\Sigma|^n$ stringhe di lunghezza n. Quindi ipotizziamo che:

$$|\Sigma^n| = |\Sigma|^n,$$

proviamo a dimostrarne la validità con una dimostrazione formale per induzione.

Dimostrazione. Vogliamo mostrare che

$$\forall n \in \mathbb{N} : |\Sigma^n| = |\Sigma|^n.$$

Procederemo per induzione matematica semplice sull'esponente n, con caso base $n = 0$ e passo induttivo $n = m + 1$ con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (antecedente dell'implicazione).

Caso base: se $n = 0$ abbiamo che

$$\begin{aligned}
|\Sigma^n| &= |\Sigma^0| && [n = 0] \\
&= |\{ \epsilon \}| && [\text{def. } \cdot^0] \\
&= 1 \\
&= |\Sigma|^0 && [\forall i \in \mathbb{N}, i^0 = 1] \\
&= |\Sigma|^n \checkmark && [n = 0].
\end{aligned}$$

Implicazione: sia $m \in \mathbb{N}$,

$$|\Sigma^m| = |\Sigma|^m \overset{?}{\Longrightarrow} |\Sigma^{m+1}| = |\Sigma|^{m+1}.$$

Passo induttivo: se $n = m + 1$ abbiamo che

$$\begin{aligned}
|\Sigma^n| &= |\Sigma^{m+1}| && [n = m + 1] \\
&= |\Sigma \cdot \Sigma^m| && [\text{def. } \cdot^n] \\
&= |\Sigma||\Sigma^m| && [\text{def. concat., prop. potenze}] \\
&= |\Sigma||\Sigma|^m && [\text{ip. ind.}] \\
&= |\Sigma|^{m+1} \checkmark && [\text{prop. potenze}].
\end{aligned}$$

□

Esercizio 2.4

Trovare gli alfabeti Σ per i quali l'insieme Σ^* è esso stesso un alfabeto.

Soluzione. Per definizione, un alfabeto è un insieme finito di simboli. Σ^* è l'insieme formato da tutte le stringhe di lunghezza arbitraria ma finita su Σ. Per ogni $\Sigma \neq \varnothing$, l'insieme Σ^* essendo infinito non può essere un alfabeto, l'unica eccezione è proprio $\Sigma = \varnothing \Longrightarrow \Sigma^* = \{\, \epsilon \,\}$.

Dimostrazione. Utilizziamo le definizioni viste in questo capitolo per costruire una catena di relazioni (uguaglianze in questo caso) che parte da Σ^* ed arriva a $\{\, \epsilon \,\}$, ricordandoci che abbiamo posto $\Sigma = \varnothing$ perché ad un certo punto tornerà utile:

$$
\begin{aligned}
\Sigma^* &= \bigcup_{i\in\mathbb{N}} \Sigma^i && [\text{def. } \cdot^*]\\
&= \Sigma^0 \cup \bigcup_{i\geq 1} \Sigma^i && [0\in\mathbb{N}]\\
&= \{\,\epsilon\,\} \cup \bigcup_{i\geq 1} \Sigma^i && [\text{def. } \cdot^0]\\
&= \{\,\epsilon\,\} \cup \bigcup_{i\geq 1} \{\, ax \mid a\in\Sigma, x\in\Sigma^{i-1}\,\} && [\text{def. } \cdot^i]\\
&= \{\,\epsilon\,\} \cup \bigcup_{i\geq 1} \{\, ax \mid a\in\varnothing, x\in\Sigma^{i-1}\,\} && [\Sigma=\varnothing]\\
&= \{\,\epsilon\,\} \cup \varnothing && [\nexists a\in\varnothing]\\
&= \{\,\epsilon\,\}\checkmark && [\text{def. } \cup].
\end{aligned}
$$

□

Esercizio 2.5

Una definizione non induttiva di concatenazione è la seguente: siano $i, j \in \mathbb{N}$, siano $x = a_1 \cdots a_i$ e $y = b_1 ... b_j$ due stringhe su Σ^* allora $xy = a_1 ... a_i b_1 ... b_j$. Mostrare l'equivalenza con la definizione induttiva.

Dimostrazione. Mostriamo per induzione matematica semplice sulla lunghezza n della stringa y applicando la definizione induttiva, con caso base $n = 0$ e passo induttivo $n = m+1$ con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (antecedente dell'implicazione).

Caso base: se $n = 0$ abbiamo

$$
\begin{aligned}
xy &= x\epsilon && [|y| = n = 0]\\
&= x && [\text{def. ind. concat.}]\\
&= x b_1 \cdots b_n \checkmark && [|y| = n = 0].
\end{aligned}
$$

Implicazione: siano $u, v \in \Sigma^*$ tali che,

$$uv = ub_1 \cdots b_m \overset{?}{\Longrightarrow} u(vb) = ub_1 \cdots b_m b.$$

Passo induttivo: se $n = m + 1$ abbiamo $y = wb$ con $|w| = m$ e vale che

$$
\begin{aligned}
xy &= x(wb) && [|y| = n = m+1] \\
&= (xw)b && [\text{def. ind. concat.}] \\
&= xb_1 \cdots b_m b && [\text{ip. ind.}] \\
&= xb_1 \cdots b_{m+1}. \checkmark
\end{aligned}
$$

□

Esercizio 2.6

Verificare che se $\Sigma \neq \varnothing$, allora Σ^* è numerabile.

Soluzione. Un insieme è enumerabile quando è possibile stabilire una corrispondenza biunivoca tra gli elementi dell'insieme e l'insieme dei numeri naturali. Scegliamo un alfabeto di esempio su cui lavorare, sia $\Sigma = \{ a, b, c \}$. Σ^* è l'insieme formato da tutte le stringhe su Σ, ovvero $\Sigma^* = \{ \epsilon, a, b, c, aa, ab, ac, ba, bb, \cdots \}$. Ciò che vorremo fare è associare ad ogni elemento di Σ^* un numero naturale. Iniziamo enumerando le stringhe di lunghezza 0 (soltanto ϵ), procediamo poi con le stringhe di lunghezza 1, 2, etc. (se le stringhe lunghezza n sono più di una, occorre ordinarle secondo un qualche criterio, noi utilizzeremo l'ordine alfanumerico):

- $\epsilon \rightsquigarrow 0$;
- $a \rightsquigarrow 1, b \rightsquigarrow 2, c \rightsquigarrow 3$;
- $aa \rightsquigarrow 4, ab \rightsquigarrow 5, ac \rightsquigarrow 6, \cdots, cc \rightsquigarrow 12$;
- $\cdots$

Cerchiamo di formalizzare questo processo: $\forall i \in \mathbb{N}$, ordiniamo le stringhe $x \in \Sigma^i$ per poi associare alla stringa in posizione j-esima dell'ordinamento il seguente numero naturale:

$$\sum_{k \in \mathbb{N}, k=0}^{i-1} |\Sigma|^k + j - 1.$$

$\sum_{k=0}^{i-1}|\Sigma|^k$ corrisponde all'ultimo numero naturale associato al "passo" $i-1$ perché $|\Sigma|^k$ corrisponde al numero di elementi nell'insieme Σ^k, facendo la somma dei $|\Sigma|^k$, con $0 \leq k \leq i-1$, otteniamo il numero totale delle stringhe di lunghezza compresa tra 0 e $i-1$. Il "-1" è perché vogliamo iniziare a contare da 0.

Esempio. Consideriamo $\Sigma = \{ a, b, c \}, k \in \mathbb{N}, \forall i \in \mathbb{N}:$

- $i = 0$,
  $\Sigma^i = \Sigma^0 = \{ \epsilon \}:$
  - $j = 1, \epsilon \rightsquigarrow \sum_{k=0}^{i-1}|\Sigma|^k + j - 1 = \sum_{k=0}^{-1}|\Sigma|^k + j - 1 = j - 1 = 0.$
- $i = 1$,
  $\Sigma^i = \Sigma = \{ a, b, c \}:$
  - $j = 1, a \rightsquigarrow \sum_{k=0}^{i-1}|\Sigma|^k + j - 1 = |\Sigma|^0 + j - 1 = 1;$
  - $j = 2, b \rightsquigarrow |\Sigma|^0 + j - 1 = 2;$
  - $j = 3, c \rightsquigarrow |\Sigma|^0 + j - 1 = 3.$
- $i = 2$,
  $\Sigma^i = \Sigma^2 = \{ aa, ab, ac, ba, bb, bc, ca, cb, cc \}:$
  - $j = 1, aa \rightsquigarrow \sum_{k=0}^{i-1}|\Sigma|^k + j - 1 = |\Sigma|^0 + |\Sigma| + j - 1 = 4;$
  - $\dots$
  - $j = 9, cc \rightsquigarrow |\Sigma^0| + |\Sigma| + j - 1 = 12;$
- $i = 3$
  $\dots$

3 Automi

3.1 Rappresentazione dei Linguaggi

La rappresentazione finita di linguaggi infiniti è afrontabile da tre punti di vista:

1. *riconoscitivo-analitico*, in cui un linguaggio è visto come un insieme di stringhe accettate da strutture finite come gli automi;

2. *algebrico*, in cui il linguaggio è rappresentato da un'espressione algebrica o è la soluzione di un sistema di relazioni algebriche;

3. *generativo-sintetico*, in cui il linguaggio è visto come l'insieme delle stringhe *generate* da strutture finite dette *grammatiche*.

Gli *automi*, che ora vedremo, ricadono nell'approccio di tipo riconoscitivo-analitico.

Un automa è un oggetto composto da:

- una *t*estina $\Uparrow$ che legge, da sinistra verso destra, simboli posti su un nastro di lunghezza illimitata. A seguito della lettura di un simbolo, la testina si sposta di una posizione verso destra per leggere il simbolo successivo;

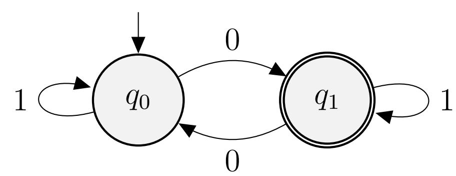
- uno *stato* interno $\boldsymbol{q}$ che a seguito della lettura di un simbolo può modificarsi o rimanere lo stesso, a seconda di come è stato definito il "comportamento" dell'automa.

$$\begin{array}{|c|c|c|c}
\hline
a_1 & a_2 & a_3 & \cdots \\
\hline
\multicolumn{1}{c}{\Uparrow} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \\
\multicolumn{1}{c}{\boxed{q}} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} &
\end{array}$$

La lettura termina quando sul nastro non rimangono più simboli da leggere, ovvero, sul nastro giace la stringa vuota ϵ. A questo punto, l'automa si può trovare in uno stato di accettazione o di rifiuto della stringa che, un simbolo alla volta, è stata letta per intero.

3.2 DFA: automi a stati finiti deterministici

Sia $\Sigma = \{\ 0, 1\ \}$ un alfabeto. La descrizione informale di un automa a stati finiti deterministico M tramite *grafo di transizione* è la seguente:



Descrizione dell'automa a stati finiti deterministico M tramite *matrice di transizione*:

	0	**1**
$\boldsymbol{q_0}$	q_1	q_0
$\boldsymbol{\dot{q}_1}$	q_0	q_1

- nella prima colonna troviamo i diversi stati q_i dell'automa: per definizione q_0 è lo stato iniziale, mentre lo stato finale (che può cambiare di automa in automa) viene identificato (in questo documento) ponendo un puntino sopra il nome dello stato ($\dot{q}_1$);
- nella prima riga troviamo i simboli in ingresso $a \in \Sigma$.

In questo caso particolare quindi, se lo stato attuale dell'automa è q e il prossimo simbolo da leggere sul nastro è a, lo stato in cui si troverà l'automa dopo la lettura del simbolo a è indicato nella cella $[q, a]$. La matrice di transizione (ed equivalentemente il grafo di transizione corrispondente), possono essere visti come la definizione di una funzione, che prende il nome di *funzione di transizione* e si indica con δ.

Esempio. Sia $\Sigma = \{\, 0, 1\, \}$. Sia $Q = \ q_0, q_1\ $ l'insieme di stati dell'automa presentato sopra. Allora la funzione $\delta : Q \times \Sigma \mapsto Q$ è definita come

$$\delta(q_0, 0) = q_1, \delta(q_0, 1) = q_0, \delta(q_1, 0) = q_0, \delta(q_1, 1) = q_1.$$

La funzione di transizione δ quindi ha:

- due argomenti in ingresso:

  - q, lo stato attuale del DFA;
  - a, il prossimo simbolo da leggere,

- uno ed un solo stato in uscita q'.

Il fatto che la funzione di transizione δ restituisce un solo ed unico stato in uscita è la motivazione dell'aggettivo *deterministico* nel nome dell'automa che descrive; infatti, per ogni coppia $\langle q, a \rangle$ in ingresso alla funzione di transizione δ, è sempre possibile determinare quale sarà il q' che si otterrà in uscita.

Notazione. Utilizzeremo la seguente notazione:

- p, q, r, s, t per indicare stati degli automi;
- P, Q, R, S, T per indicare insiemi di stati.
- $\overbrace{aa \cdots a}^{n} = a^n$ indica la stringa composta dalla giustapposizione del simbolo a con se stesso n volte;
- $a_1 a_2 \cdots a_n$ indica la stringa composta dalla giustapposizione di simboli a_i, anche diversi tra loro;
- $\overbrace{xx \cdots x}^{n} = x^n$ indica la stringa composta dalla concatenazione della stringa x con se stessa n volte;
- $x_1 x_2 \cdots x_n$ indica la stringa composta dalla concatenazione delle stringhe x_i, anche diverse tra loro.

3.2.1 Definizioni, Teoremi e Lemmi

Definizione 3.1 (DFA: automa a stati finiti deterministico)**.** *Un DFA è una quintupla* $\langle Q, \Sigma, \delta, q_0, F \rangle$*, dove:*

- $\boldsymbol{Q}$ *è un insieme finito di stati;*
- $\boldsymbol{\Sigma}$ *è l'alfabeto di input;*
- $\boldsymbol{\delta} : Q \times \Sigma \mapsto Q$ *è la funzione di transizione;*
- $\boldsymbol{q_0} \in Q$ *è lo stato iniziale;*

- $\boldsymbol{F} \subseteq Q$ *è l'insieme degli stati finali (o stati accettanti).*

Per definizione, una stringa è una giustapposizione di simboli, ciò che vedremo ora è una nuova funzione che applica più volte la δ per leggere intere stringhe dal nastro di input.

Definizione 3.2 (Funzione di transizione su stringhe)**.** *Sia M il DFA descritto dalla quintupla $\langle Q, \Sigma, \delta, q_0, F\rangle$. Sia x una stringa su Σ^*. La funzione di transizione su stringhe $\hat{\delta} : Q \times \Sigma^* \rightarrowtail Q$ è comunemente definita come:*

$$\hat{\delta}(q,x) := \begin{cases} q, & \textit{se } x = \epsilon; \\ \delta\big(\hat{\delta}(q,y),a\big), & \textit{se } x = ya. \end{cases}$$

Osservazione 3.1 (Definizione induttiva della funzione di transizione su stringhe tramite notazione semantica)**.**

$$\frac{}{\hat{\delta}(q,\epsilon) = q}$$

$$\frac{\hat{\delta}(q,xa) = q'}{\delta\big(q',a\big)} a \in \Sigma$$

Osservazione 3.2. *Nella definizione di $\hat{\delta}$ vi è un elemento di arbitrarietà. Si può infatti dimostrare che, $\hat{\delta}$ si può sostituire con la funzione $\check{\delta}$ così definita:*

$$\check{\delta}(q,x) := \begin{cases} q, & \textit{se } x = \epsilon; \\ \check{\delta}\big(\delta(q,a),y\big), & \textit{se } x = ay. \end{cases}$$

Dimostrazione. Sia M il DFA descritto dalla quintupla $\langle Q, \Sigma, \delta, q_0, F\rangle$. Dimostriamo che, per ogni $x \in \Sigma^*$ e ogni $q \in Q$, abbiamo $\hat{\delta}(q,x) = \check{\delta}(q,x)$. Sia q arbitrario. Per il caso $x = \epsilon$ abbiamo, banalmente

$$\hat{\delta}(q,\epsilon) = q = \check{\delta}(q,\epsilon).$$

Sia ora $x \neq \epsilon$, $x = a_1 \dots a_n$. Dimostreremo per induzione matematica su lunghezza $n \geq 1$ della stringa, con caso base $n = 1$ e passo induttivo $n = m + 1$ con $m \geq 1$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva:

$$\hat{\delta}(q, a_1 \cdots a_m) = \underbrace{\delta\big(\cdots\delta(}_{m} q \underbrace{, a_1)\cdots a_m\big)}_{m} = \check{\delta}(q, a_1 \cdots a_m).$$

Caso base: con $n = 1$ abbiamo

$$\hat{\delta}(q, a_1) = \delta\big(\hat{\delta}(q, \epsilon), a_1\big) = \delta(q, a_1) = \check{\delta}\big(\delta(q, a_1), \epsilon\big) = \check{\delta}(q, a_1).$$

Implicazione: sia $m \in \mathbb{N}$,

$$\hat{\delta}(q, a_1 \cdots a_m) = \underbrace{\delta\big(\cdots\delta(}_{m} q \underbrace{, a_1) \cdots a_m\big)}_{m} \overset{?}{\Longrightarrow} \hat{\delta}(q, a_1 \cdots a_{m+1}) = \underbrace{\delta\big(\cdots\delta(}_{m+1} q \underbrace{, a_1) \cdots a_{m+1}\big)}_{m+1}$$

Passo inuttivo: con $n = m + 1$ abbiamo

$$\begin{aligned}
\hat{\delta}(q, a_1 \cdots a_m a_{m+1}) &= \delta\big(\hat{\delta}(q, a_1 \cdots a_m), a_{m+1}\big) && [\text{def. } \hat{\delta}] \\
&= \delta\Big(\underbrace{\delta\big(\cdots\delta(}_{m} q \underbrace{, a_1) \cdots a_m\big)}_{m}, a_{m+1}\Big) && [\text{ip. ind.}] \\
&= \underbrace{\delta\big(\cdots\delta(}_{m+1} q \underbrace{, a_1) \cdots a_{m+1}\big)}_{m+1} && [\text{raggruppando}].
\end{aligned}$$

Analogamente:

$$\begin{aligned}
\check{\delta}(q, a_1 a_2 \cdots a_{m+1}) &= \check{\delta}\big(\delta(q, a_1), a_2 \cdots a_{m+1}\big) && [\text{def. } \check{\delta}] \\
&= \underbrace{\delta\big(\cdots\delta(}_{m} \delta(q, a_1) \underbrace{, a_2) \cdots a_{m+1}\big)}_{m} && [\text{ip. ind.}] \\
&= \underbrace{\delta\big(\cdots\delta(}_{m+1} q \underbrace{, a_1) \cdots a_{m+1}\big)}_{m+1} && [\text{raggruppando}].
\end{aligned}$$

□

Definizione 3.3 (Accettazione di una stringa)**.** *Sia M il DFA descritto dalla quintupla $\langle Q, \Sigma, \delta, q_0, F\rangle$. Sia x una stringa su Σ^*. La stringa x è* accettata *da M se e solo se:*

$$\hat{\delta}(q_0, x) \in F.$$

Una stringa x quindi è accettata da un DFA M se, partendo dallo stato iniziale q_0 e leggendo x, il DFA termina in uno degli stati accettanti.

Se durante il processo di lettura viene letto un simbolo che non sta in Σ, la lettura termina immediatamente con il DFA in uno stato di “errore” che ha come conseguenza il rifiuto della stringa.

Definizione 3.4 (Linguaggio accettato da un DFA)**.** *Sia M il DFA descritto dalla quintupla $\langle Q, \Sigma, \delta, q_0, F\rangle$, il linguaggio* accettato *da M si indica con $L(M)$ ed è l'insieme di tutte le stringhe accettate da M, ovvero:*

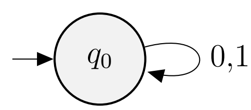
$$L(M) := \{\, x \in \Sigma^* \mid \hat{\delta}(q_0, x) \in F \,\}.$$

Teorema 3.1 (Linguaggio regolare)**.** *Un linguaggio L è regolare se esiste almeno un DFA M che accetta L, ovvero:*

$$L = L(M).$$

Esempio. DFA $M = \langle Q = \{\, q_0 \,\}, \Sigma = \{\, 0, 1 \,\}, \delta, q_0, F = \varnothing\rangle$ che accetta il linguaggio (regolare) $L = \varnothing$ ovvero "nessuna stringa è accettata":

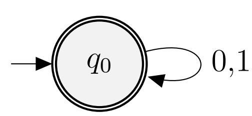
$$L(M) = \varnothing$$



	0	**1**
$\boldsymbol{q_0}$	q_0	q_0

DFA $M' = \langle Q = \{\, q_0 \,\}, \Sigma = \{\, 0, 1 \,\}, \delta, q_0, F = \{\, q_0 \,\}\rangle$ che accetta il linguaggio (regolare) $L = \Sigma^*$ ovvero "tutte le stringhe su Σ sono accettate":

$$L(M') = \Sigma^*$$



	0	**1**
$\boldsymbol{\dot{q}_0}$	q_0	q_0

Lemma 3.1 (Proprietà funzione di transizione su stringhe)**.** *Sia M il DFA descritto dalla quintupla $\langle Q, \Sigma, \delta, q_0, F \rangle$. Siano $x, y \in \Sigma^*$ due stringhe e $q \in Q$ uno stato, allora*

$$\hat{\delta}(q, xy) = \hat{\delta}\big(\hat{\delta}(q, x), y\big)$$

Dimostrazione. Vogliamo mostrare che

$$\forall x, y \in \Sigma^* : \hat{\delta}(q, xy) = \hat{\delta}\big(\hat{\delta}(q, x), y\big).$$

Dimostriamo per induzione matematica semplice sulla lunghezza n della stringa y, con caso base $n = 0$ e passo induttivo $n = m + 1$, con $m \in \mathbb{N}$; pertanto nel passo induttivo potremo sfruttare l'ipotesi induttiva (antecedente dell'implicazione).

Caso base: se $n = 0$ abbiamo

$$\begin{aligned}
\hat{\delta}(q, xy) &= \hat{\delta}(q, x\epsilon) && [|y| = 0] \\
&= \hat{\delta}(q, x) && [\epsilon \text{ è identità}] \\
&= \hat{\delta}\big(\hat{\delta}(q, x), \epsilon\big) && [\text{def. } \hat{\delta}] \\
&= \hat{\delta}\big(\hat{\delta}(q, x), y\big) \checkmark && [|y| = 0].
\end{aligned}$$

Implicazione: siano $u, v \in \Sigma^*$,

$$\hat{\delta}(q, uv) = \hat{\delta}\big(\hat{\delta}(q, u), v\big) \overset{?}{\Longrightarrow} \hat{\delta}(q, uva) = \hat{\delta}\big(\hat{\delta}(q, u), va\big).$$

Passo induttivo: Se $n = m + 1, y = wa$ abbiamo

$$\begin{aligned}
\hat{\delta}(q, xy) &= \hat{\delta}(q, xwa) && [y = wa] \\
&= \delta\big(\hat{\delta}(q, xw), a\big) && [\text{def. } \hat{\delta}] \\
&= \delta\big(\hat{\delta}\big(\hat{\delta}(q, x), w\big), a\big) && [\text{ip. ind.}] \\
&= \hat{\delta}\big(\hat{\delta}(q, x), wa\big) && [\text{def. } \hat{\delta}] \\
&= \hat{\delta}\big(\hat{\delta}(q, x), y\big) \checkmark && [y = wa].
\end{aligned}$$

$\square$

3.2.2 Esercizi

Esercizio 3.1

Siano xz, yz due stringhe e $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ un DFA, dimostrare che vale:

$$\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) \Longrightarrow \hat{\delta}(q_0, xz) = \hat{\delta}(q_0, yz).$$

Soluzione.

$$\begin{aligned} \hat{\delta}(q_0, xz) &= \hat{\delta}\big(\hat{\delta}(q_0, x), z\big) && [\text{prop. di } \hat{\delta}] \\ &= \hat{\delta}\big(\hat{\delta}(q_0, y), z\big) && [\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)] \\ &= \hat{\delta}(q_0, yz) && [\text{prop. di } \hat{\delta}]. \end{aligned}$$

Esercizio 3.2

Siano x, y stringhe e q uno stato, dimostrare che vale:

$$\hat{\delta}(q, x) = q_1, \hat{\delta}(q, xy) = q_2 \Longrightarrow \hat{\delta}(q_1, y) = q_2.$$

Soluzione.

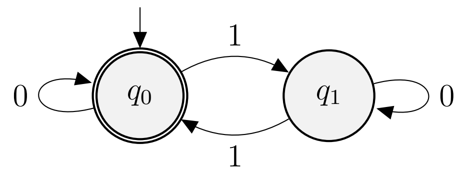
$$\begin{aligned} \hat{\delta}(q_1, y) &= \hat{\delta}\big(\hat{\delta}(q, x), y\big) && [q_1 = \hat{\delta}(q, x)] \\ &= \hat{\delta}(q, xy) && [\text{prop. di } \hat{\delta}] \\ &= q_2. \end{aligned}$$

Esercizio 3.3

Determinare il DFA che accetta il seguente linguaggio: $L = \{\, x \in \{\, 0, 1\,\}^* \mid x \text{ contiene un numero pari di `1'} \,\}$.

Soluzione. Costruiamo un DFA $M = \langle Q = \{\, q_0, q_1 \,\}, \Sigma = \{\, 0, 1 \,\}, \delta, q_0, F = \{\, q_0 \,\}\rangle$ che tiene traccia della lettura del simbolo '1'. L'idea è di passare dallo stato iniziale accettante q_0 al non accettante q_1 quando viene letto '1' e viceversa, così da tenere sempre taccia se il numero di '1' che sono stati letti è pari (M in q_0) o dispari (M in q_1).

Descrizione del DFA tramite grafo di transizione:



Descrizione del DFA tramite matrice di transizione:

	0	**1**
$\dot{\boldsymbol{q}}_{\boldsymbol{0}}$	q_0	q_1
$\boldsymbol{q_1}$	q_1	q_0

Dimostrazione. Dobbiamo dimostrare che il comportamento del DFA è stato definito correttamente e fa quello che deve. Per fare questo, dimostriamo che:

1. il DFA accetta le stringhe di interesse;
2. il DFA rifiuta tutte le stringhe non di interesse.

Notiamo che, in qualunque stato il DFA si trova, leggere '0' non comporta modifiche dello stato, ovvero

$$\forall q \in Q : \delta(q, 0) = q,$$

per semplificare il nostro lavoro possiamo considerare le sole stringhe della forma 1^n, con $n \in \mathbb{N}$, tenendo a mente che potranno apparire degli '0' ovunque senza modificare lo stato dell'automa.

Accettazione. Una generica stringa accettata dal DFA è della forma:

$$1^{2n}, n \in \mathbb{N}.$$

Vogliamo mostrare che, applicando la funzione di transizione su stringhe $\hat{\delta}$ sullo stato iniziale q_0 e una stringa della forma 1^{2n}, arriviamo in uno stato $q \in F$ che è accettante, ovvero

$$\forall n \in \mathbb{N} : \hat{\delta}(q_0, 1^{2n}) = q_0 \in F.$$

Dimostriamo la per induzione matematica semplice su n, con caso base $n = 0$ e passo induttivo $n = m + 1$, con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (antecedente dell'implicazione).

Caso base: se $n = 0$ abbiamo che

$$\begin{aligned}
\hat{\delta}(q_0, 1^{2n}) &= \hat{\delta}(q_0, 1^0) && [n = 0] \\
&= \hat{\delta}(q_0, \epsilon) && [\text{def. } \cdot^0] \\
&= q_0 \in F \checkmark && [\text{def. } \hat{\delta}].
\end{aligned}$$

Implicazione: sia $m \in \mathbb{N}$,

$$\hat{\delta}(q_0, 1^{2m}) = q_0 \overset{?}{\Longrightarrow} \hat{\delta}(q_0, 1^{2(m+1)}) = q_0.$$

Passo induttivo: Se $n = m + 1$ abbiamo che

$$\begin{aligned}
\hat{\delta}(q_0, 1^{2n}) = \hat{\delta}(q_0, 1^{2(m+1)}) &= \hat{\delta}(q_0, 111^{2m}) && [n = m + 1] \\
&= \hat{\delta}\big(\delta(q_0, 1), 11^{2m}\big) && [\text{def. } \hat{\delta}] \\
&= \hat{\delta}(q_1, 11^{2m}) && [\text{def. } \delta] \\
&= \hat{\delta}\big(\delta(q_1, 1), 1^{2m}\big) && [\text{def. } \hat{\delta}] \\
&= \hat{\delta}(q_0, 1^{2m}) && [\text{def. } \delta] \\
&= q_0 \in F \checkmark && [\text{ip. ind.}].
\end{aligned}$$

Rifiuto. Una generica stringa rifiutata dal DFA è della forma:

$$1^{2n+1}, n \in \mathbb{N}. \tag{1}$$

Vogliamo mostrare che, applicando la funzione di transizione su stringhe $\hat{\delta}$ sullo stato iniziale q_0 e una stringa della forma (1), arriviamo in uno stato $q \notin F$ di rifiuto, ovvero

$$\forall n \in \mathbb{N} : \hat{\delta}(q_0, 1^{2n+1}) = q_1 \notin F,$$

Dimostriamo per induzione matematica semplice su n, con caso base $n = 0$ e passo induttivo $n = m+1$, con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (antecedente dell'implicazione). In breve:

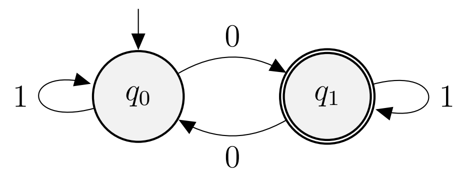
$$\hat{\delta}(q_0, 1^{2(m+1)+1}) = \hat{\delta}(q_0, 111^{2m}1) = \hat{\delta}(q_0, 1) = q_1 \notin F.$$

(Dimostrare con passaggi nel dettaglio per esercizio). □

Esercizio 3.4

Determinare il DFA che accetta il seguente linguaggio: $L = \{\, x \in \{\, 0, 1 \,\}^* \mid x \text{ contiene un numero dispari di ‘0’} \,\}$.

Soluzione. Descrizione del DFA $M = \langle Q = \{ q_0, q_1 \}, \Sigma = \{ 0, 1 \}, \delta, q_0, F = \{ q_1 \} \rangle$ tramite grafo di transizione:



Descrizione del DFA tramite matrice di transizione:

	0	**1**
$\boldsymbol{q_0}$	q_1	q_0
$\dot{\boldsymbol{q}}_{\boldsymbol{1}}$	q_0	q_1

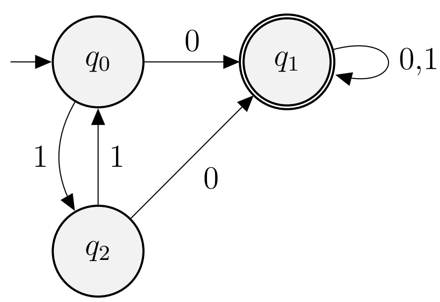
(Spiegare il ragionamento per la costruzione del DFA e dimostrare il corretto funzionamento per esercizio, notare che è tutto estremamente simile all'esercizio precedente).

Esercizio 3.5

Determinare il linguaggio accettato dal DFA descritto tramite la seguente matrice di transizione:

	0	**1**
$\boldsymbol{q_0}$	q_1	q_2
$\dot{\boldsymbol{q}}_{\boldsymbol{1}}$	q_1	q_1
$\boldsymbol{q_2}$	q_1	q_0

Soluzione. Grafo di transizione del DFA:



Osservando il DFA notiamo che: a partire dagli stati q_0, q_2, la lettura del simbolo '1' fa ciclare tra gli stati q_0, q_2 stessi, mentre la lettura del simbolo '0' porta dentro allo stato $q_1 \in F$. Lo stato q_1 non avendo archi uscenti è detto stato *assorbente*, quindi:

$$L(M) = \{\, x \in \Sigma^* \mid x \text{ contiene almeno uno '0'} \,\}.$$

Dimostrazione. Constatiamo che a partire dagli stati q_0 e q_2, la lettura del simbolo '1' fa ciclare tra gli stati q_0, q_2 stessi:

$$\forall q \in \{\, q_0, q_2 \,\} : \delta(q, 1) = q' \in \{\, q_0, q_2 \,\} \qquad \text{[def. } \delta\text{]},$$

da cui deriva che

$$\forall q \in \{\, q_0, q_2 \,\}, \forall n \in \mathbb{N} : \hat{\delta}(q, 1^n) = q' \in \{\, q_0, q_2 \,\}. \tag{2}$$

Constatiamo che q_1 è uno stato assorbente:

$$\forall a \in \Sigma : \delta(q_1, a) = q_1 \qquad \text{[def. } \delta\text{]}.$$

da cui deriva che

$$\forall x \in \Sigma^*, \forall n \in \mathbb{N} : \hat{\delta}(q_1, x) = q_1. \tag{3}$$

Rifiuto. Una generica stringa rifiutata dal DFA è della forma:

$$1^n, n \in \mathbb{N}.$$

$$\forall q \in \{\, q_0, q_2 \,\}, \forall n \in \mathbb{N} : \hat{\delta}(q, 1^n) \notin F \qquad [(2)].$$

Accettazione. Una generica stringa accettata dal DFA è della forma:

$$1^n 0 y, n \in \mathbb{N}, y \in \Sigma^*.$$

Dal momento che $q_0 \in \{\, q_0, q_2 \,\}$, abbiamo appena dimostrato che

$$\forall n \in \mathbb{N} : \hat{\delta}(q_0, 1^n 0 y) = \hat{\delta}(q', 0y), \text{ con } q' \in \{\, q_0, q_2 \,\},$$

proseguiamo

$$\begin{aligned} \hat{\delta}(q', 0y) &= \hat{\delta}\big(\delta(q', 0), y\big) && \text{[def. } \hat{\delta}\text{]} \\ &= \hat{\delta}(q_1, y) && \text{[def. } \delta\text{]} \\ &= q_1 \in F && [(3)]. \end{aligned}$$

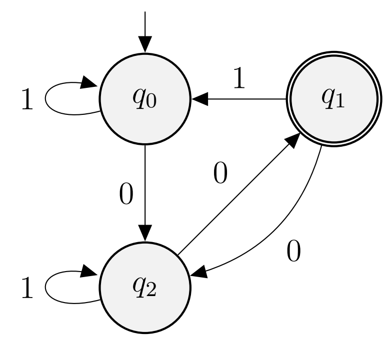
□

Esercizio 3.6

Determinare il linguaggio accettato dal DFA descritto tramite la seguente matrice di transizione:

	0	**1**
$\boldsymbol{q_0}$	q_2	q_0
$\dot{\boldsymbol{q}}_{\boldsymbol{1}}$	q_2	q_0
$\boldsymbol{q_2}$	q_1	q_2

Soluzione. Disegnamo il grafo di transizione del DFA:



Notiamo che l'unico modo per entrare in q_2 è con la lettura di uno '0' e nel momento in cui un'altro viene letto, si passa da q_2 a $q_1 \in F$. Da q_1 si esce e si torna in q_2 con la lettura di '0'. Le osservazioni fatte fino ad ora fanno pensare che le stringhe devono contenere un numero pari di '0'. Da q_1 però con la lettura di '1' si va in q_0, quindi le stringhe devono anche terminare con '0'.

$L(M) = \left\{ x \in \{0,1\}^* \mid x \text{ contiene un numero pari di '0' e termina con '0'} \right\}.$

Dimostrazione. Mostriamo che il DFA accetta tutte e sole le stringhe.

Accettazione. Una generica stringa accettata dal DFA è della forma:

$$x_1 x_2 \cdots x_n \text{ con } n \geq 1, x_i = 1^j 0 1^k 0, j, k \in \mathbb{N}.$$

In questo esercizio, con x_i intendiamo una stringa della forma appena vista. E' facile dimostrare che dagli stati q_0, q_1 la lettura di una generica x_i porta in $q_1 \in F$, ovvero

$$\forall q \in \{ q_0, q_1 \} : \hat{\delta}(q, x) = q_1 \in F. \tag{4}$$

Vogliamo mostrare che dallo stato iniziale q_0, leggendo una concatenazione di generiche x_i si arriva in $q_1 \in F$, ovvero

$$\forall n \in \mathbb{N}, n \geq 1 : \hat{\delta}\big(q_0, x_1 x_2 \cdots x_n\big) = q_1 \in F,$$

Dimostreremo per induzione matematica semplice su $n \in \mathbb{N}$ che è il numero di concatenazioni di stringhe x_i, con caso base $n = 1$, passo induttivo $n = m+1$, con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (antecedente dell'implicazione).

Caso base: se $n = 1$ abbiamo che

$$\begin{aligned} \hat{\delta}(q_0, x_1 \cdots x_n) &= \hat{\delta}(q_0, x) && [n = 1] \\ &= q_1 \in F && [(4)]. \end{aligned}$$

Implicazione: sia $m \in \mathbb{N}$,

$$\hat{\delta}(q_0, x_1 \cdots x_m) = q_1 \overset{?}{\Longrightarrow} \hat{\delta}(q_0, x_1 \cdots x_{m+1}) = q_1.$$

Passo induttivo: Se $n = m + 1$ abbiamo che

$$\begin{aligned} \hat{\delta}(q_0, x_1 \cdots x_n) &= \hat{\delta}(q_0, x_1 \cdots x_{m+1}) && [n = m + 1] \\ &= \hat{\delta}\big(\hat{\delta}(q_0, x_1 \cdots x_m), x_{m+1}\big) && [\text{prop. di } \hat{\delta}] \\ &= \hat{\delta}(q_1, x_{m+1}) && [\text{ip. ind.}] \\ &= q_1 \in F && [(4)]. \end{aligned}$$

Rifiuto. Una generica stringa rifiutata dal DFA ha tutte o alcune delle seguenti caratteristiche:

- stringhe che non terminano con '0', ovvero

$$\hat{\delta}(q_0, \epsilon) \notin F, \forall x \in \Sigma^* : \hat{\delta}(q_0, x1) \notin F.$$

Dimostrazione.

$$\hat{\delta}(q_0, \epsilon) = q_0 \notin F \qquad\qquad [\text{def. } \hat{\delta}].$$

$$\forall w \in \Sigma^* : \hat{\delta}(q_0, w1) = \hat{\delta}(q', 1) \notin F \qquad [\nexists q \in Q \,.\, \delta(q, 1) \in F].$$

□

- stringhe che contengono un numero dispari di '0', ovvero

  $$\forall n \in \mathbb{N} : \hat{\delta}(q_0, 0^{2n+1}) \notin F,$$

  considerando il fatto che potrebbero esserci degli '1' ovunque ma non sono rilevanti per questo particolare caso perché chiaramente non entrano in gioco nel conteggio degli '0' e il caso in cui la stringa termini con '1' è gia stato coperto da punto precedente.

  $$\hat{\delta}(q_0, 0^{2n+1}) = \hat{\delta}(q_0, 0^{2n}0) = \hat{\delta}(q_1, 0) = q_2 \notin F.$$

□

Esercizio 3.7

Sia Q un insieme di stati. Sia Σ un alfabeto. Quanti sono i DFA che si possono costruire fissati Q e Σ?

Soluzione. Un DFA è una quintupla $\langle Q, \Sigma, \delta, q_0, F \rangle$, se Q possiamo contare i possibili DFA nel seguente modo:

- numero di modi in cui possiamo scegliere lo stato iniziale q_0 è

  $$|Q|;$$

- numero di modi in cui possiamo scegliere l'insieme degli stati finali $F \subseteq Q$ è

  $$|\wp(Q)| = 2^{|Q|};$$

- numero di modi in cui possiamo definire la funzione di transizione δ è

$$|Q|^{|Q||\Sigma|},$$

perché ogni in ogni cella della matrice di transizione possiamo inserire $|Q|$ valori differenti, e le dimensioni della matrice sono $|Q||\Sigma|$.

Quindi, fissati Q e Σ, è possibile costruire

$$|Q| \cdot 2^{|Q|} \cdot |Q|^{|Q||\Sigma|} = 2^{|Q|} \cdot |Q|^{|Q||\Sigma|+1}$$

differenti DFA.

Esercizio 3.8

Sia L il linguaggio accettato dal DFA $M = \langle Q, \Sigma, \delta, q_0, F \rangle$, dimostrare che esiste un DFA $M' = \langle Q', \Sigma', \delta', q'_0, F' \rangle$ tale che $\Sigma = \Sigma'$ e $L(M) = L(M')$.

Soluzione. Sia $M' = \langle Q', \Sigma', \delta', q'_0, F' \rangle$ il DFA definito come segue:

- $Q' = Q \cup \{ q_\varnothing \}$ (stiamo aggiungendo uno stato "spazzatura" $q_\varnothing$ che renderemo irraggiungibile tramite la definzione che daremo a δ'.)
- $\Sigma' = \Sigma$;
- $\delta'(q, a) = \begin{cases} q_\varnothing & \text{se } q = q_\varnothing; \\ \delta(q, a) & \text{altrimenti.} \end{cases}$

  (abbiamo definito la $\delta' = \delta$ per ogni stato, fatta eccezione per q_t).
- $q'_0 = q_0$;
- $F' = F$.

Mostriamo l'equivalenza tra M' ed M, ovvero:

$$\forall x \in \Sigma^* : \hat{\delta}'(q'_0, x) = \hat{\delta}(q_0, x).$$

Procediamo per induzione matematica semplice sulla lunghezza n della stringa accettata, con caso base $n = 0$ e passo induttivo $n = m + 1$, con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (antecedente dell'implicazione).

Caso base: con $|x| = 0$ abbiamo che

$$\begin{aligned}
\hat{\delta}'(q_0', x) &= \hat{\delta}'(q_0', \epsilon) && [|x| = 0] \\
&= q_0' && [\text{def. } \hat{\delta}] \\
&= q_0 && [q_0' = q_0] \\
&= \hat{\delta}(q_0, \epsilon) && [\text{def. } \hat{\delta}] \\
&= \hat{\delta}(q_0, x) \checkmark && [x = \epsilon].
\end{aligned}$$

Implicazione: sia $u \in \Sigma^*$,

$$\hat{\delta}'(q_0, u) = \hat{\delta}(q_0, u) \overset{?}{\Longrightarrow} \hat{\delta}'(q_0, ua) = \hat{\delta}(q_0, ua).$$

Passo induttivo: con $n = m + 1$ abbiamo che $x = wa$, con $|w| = m$ e vale che

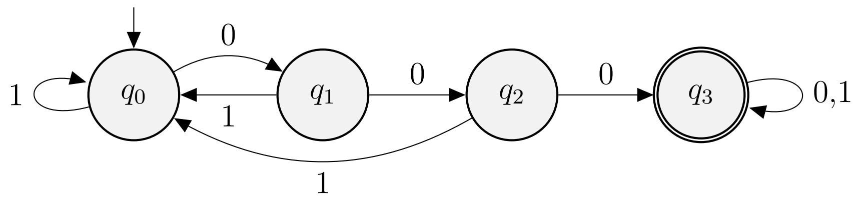
$$\begin{aligned}
\hat{\delta}'(q_0', x) &= \hat{\delta}'(q_0', wa) && [x = wa] \\
&= \delta'\big(\hat{\delta}'(q_0', w), a\big) && [\text{def. } \hat{\delta}'] \\
&= \delta'\big(\hat{\delta}(q_0, w), a\big) && [\text{ip. ind.}] \\
&= \delta\big(\hat{\delta}(q_0, w), a\big) && [\text{def. } \delta'] \\
&= \hat{\delta}(q_0, wa) && [\text{def. } \hat{\delta}] \\
&= \hat{\delta}(q_0, x) \checkmark && [x = wa].
\end{aligned}$$

Esercizio 3.9

Verificare che il seguente linguaggio su $\Sigma = \{ 0, 1 \}$ è regolare: insieme delle stringhe binarie aventi tre '0' consecutivi.

Soluzione. Per verificare che un linguaggio è regolare, dobbiamo trovare un DFA che lo accetta e dimostrarne il corretto funzionamento (le dimostrazioni del corretto funzionamento sono lasciate per esercizio).

Costruiamo un DFA in cui ogni stato rappresenta il numero di '0' consecutivi che sono stati letti. Lo stato iniziale è q_0. Per ogni stato $q_i, 0 \leq i \leq 2$, quando viene letto '1' si va nello stato q_{i+1}, mentre se viene letto '0' si torna nello stato q_0. Nel caso si leggano tre '0' consecutivi si arriva dunque in q_3 che è stato accettante ed assorbente.



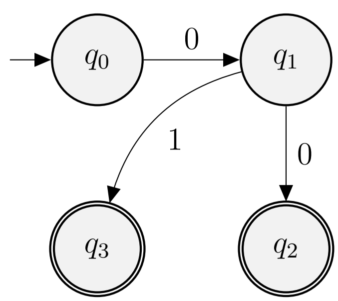
Esercizio 3.10

Verificare che il seguente linguaggio su $\Sigma = \{0, 1\}$ è regolare: insieme delle stringhe binarie aventi '0' come penultimo simbolo.

Soluzione. Sia $\Sigma = \{0, 1\}$. Sia x una generica string su Σ. Le stringhe accettate sono del tipo:

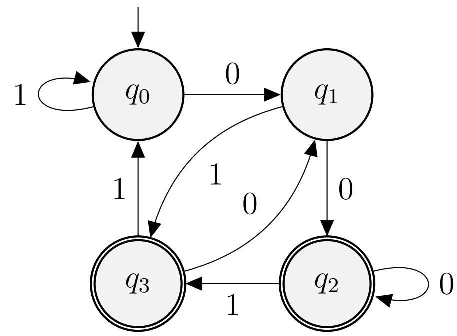
1. $x00$;
2. $x01$.

Le transizioni $q_0 \xrightarrow{0} q_1 \xrightarrow{0} \dot{q}_2$ "rilevano" che gli ultimi due simboli letti sono stati "00"; questa informazione viene poi "memorizzata" avendo $\dot{q}_2$ come stato finale. Le transazioni $q_0 \xrightarrow{0} q_1 \xrightarrow{1} \dot{q}_3$ rilevano che gli ultimi due simboli letti sono stati "01" e questa informazione viene memorizzata con $\dot{q}_3$ come stato finale.



A questo punto, $\forall q \in Q$ andiamo ad aggiungere le transizioni mancanti rispettando le proprietà degli stati $\dot{q}_2, \dot{q}_3$, ovvero:

1. partendo da q arriviamo in $\dot{q}_2 \Longleftrightarrow$ viene letto "00";
2. partendo da q arriviamo in $\dot{q}_3 \Longleftrightarrow$ viene letto "01".

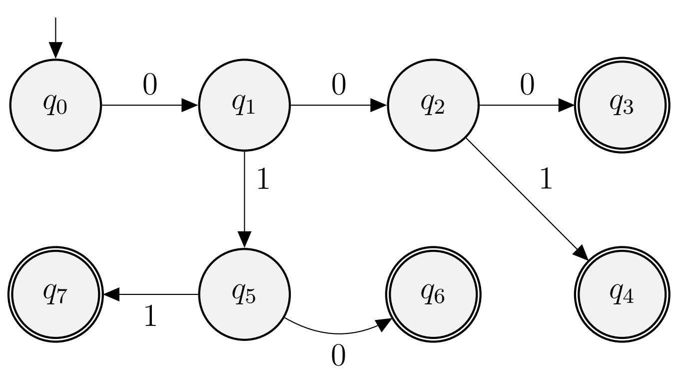


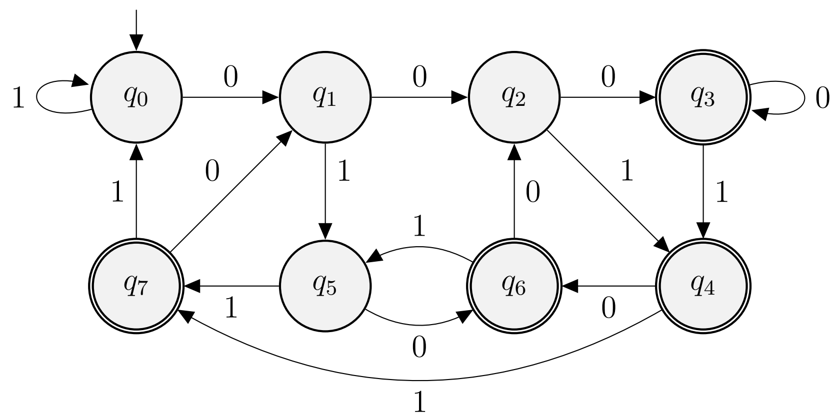
Esercizio 3.11

Verificare che il seguente linguaggio su $\Sigma = \{0, 1\}$ è regolare: insieme delle stringhe binarie aventi '0' come terzultimo simbolo.

Soluzione. Sia $\Sigma = \{0, 1\}$. Sia x una generica string su Σ. La logica è identica al DFA precedente, solo che le stringhe accettate e quindi anche gli stati aumentano (esponenzialmente):

1. $x000$, $(q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \xrightarrow{0} \dot{q}_3)$;
2. $x001$, $(q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \xrightarrow{1} \dot{q}_4)$;
3. $x010$, $(q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_5 \xrightarrow{0} \dot{q}_6)$;
4. $x011$, $(q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_5 \xrightarrow{1} \dot{q}_7)$;





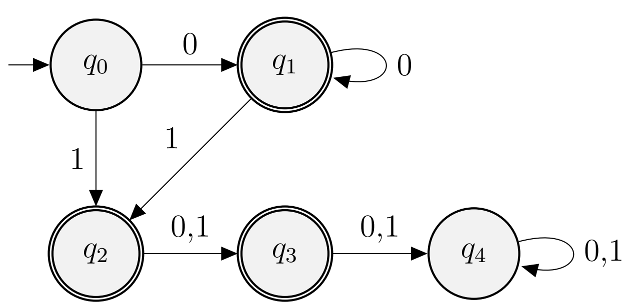
Esercizio 3.12

Verificare che il seguente linguaggio su $\Sigma = \{ 0, 1 \}$ è regolare: insieme delle stringhe binarie tali che, se interpretate come numero decimale, hanno un valore minore o uguale a 3.

Soluzione. Stringhe accettate:

- $0_2 \mapsto 0_{10} : q_0 \xrightarrow{0} \dot{q}_1$;
- $1_2 \mapsto 1_{10} : q_0 \xrightarrow{1} \dot{q}_2$;
- $10_2 \mapsto 2_{10} : q_0 \xrightarrow{1} q_2 \xrightarrow{0} \dot{q}_3$;
- $11_2 \mapsto 3_{10} : q_0 \xrightarrow{1} q_2 \xrightarrow{1} \dot{q}_3$.

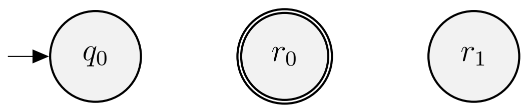
Utilizziamo q_4 come stato assorbente non finale, in modo da rifiutare tutte le stringhe con valore maggiore di 3.



Esercizio 3.13

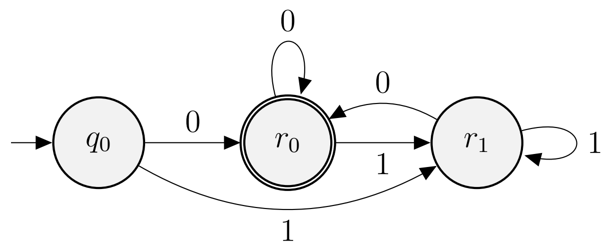
Verificare che il seguente linguaggio su $\Sigma = \{0, 1\}$ è regolare: insieme delle stringhe binarie tali che, se interpretate come numero decimale, sono divisibili per 2.

Soluzione. Avremo lo stato iniziale q_0 non accettante perché non vogliamo accettare la stringa ϵ visto che non ha interpretazione nei numeri interi senza segno. Da q_0 ci spostiamo in r_0 o r_1 a seconda se viene letto '0' o '1' rispettivamente. La r nel nome degli stati sta per "resto"; negli stati r_0, r_1 infatti, vi si arriva quando, interpretando come numero intero la stringa letta e facendone il modulo 2, si ottiene resto 0 e resto 1 rispettivamente.



Facciamo un "mapping" dei numeri da 0 a n, in accordo con quanto detto sopra, fino a che tutte le tansizioni non sono state definite (raggiunto un *punto fisso*):

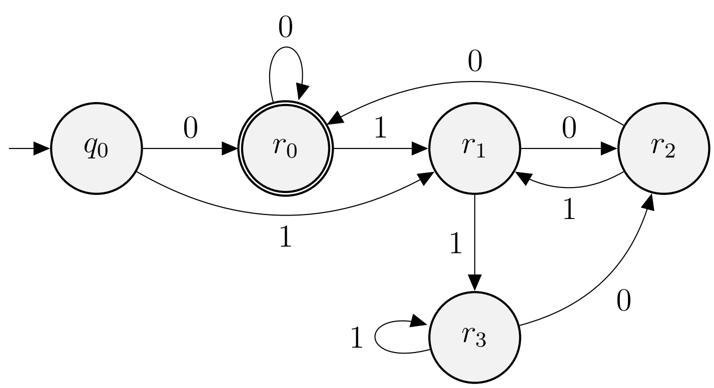
- $0_2 \mapsto 0_{10} \mod 2 = 0 : q_0 \xrightarrow{0} \dot{r}_0$;
- $1_2 \mapsto 1_{10} \mod 2 = 1 : q_0 \xrightarrow{1} r_1$;
- $10_2 \mapsto 2_{10} \mod 2 = 0 : q_0 \xrightarrow{1} r_1 \xrightarrow{0} \dot{r}_0$;
- $11_2 \mapsto 3_{10} \mod 2 = 1 : q_0 \xrightarrow{1} r_1 \xrightarrow{1} r_1$;
- $100_2 \mapsto 4_{10} \mod 2 = 0 : q_0 \xrightarrow{1} r_1 \xrightarrow{0} r_0 \xrightarrow{0} \dot{r}_0$;
- $101_2 \mapsto 5_{10} \mod 2 = 1 : q_0 \xrightarrow{1} r_1 \xrightarrow{0} r_0 \xrightarrow{1} r_1$.
- $110_2 \mapsto 6_{10} \mod 2 = 0 : q_0 \xrightarrow{1} r_1 \xrightarrow{1} r_1 \xrightarrow{0} \dot{r}_0$ (nessuna nuova transazione aggiunta, punto fisso raggiunto).



Esercizio 3.14

Verificare che il seguente linguaggio su $\Sigma = \{0, 1\}$ è regolare: insieme delle stringhe binarie tali che, se interpretate come numero decimale, sono divisibili per 4.

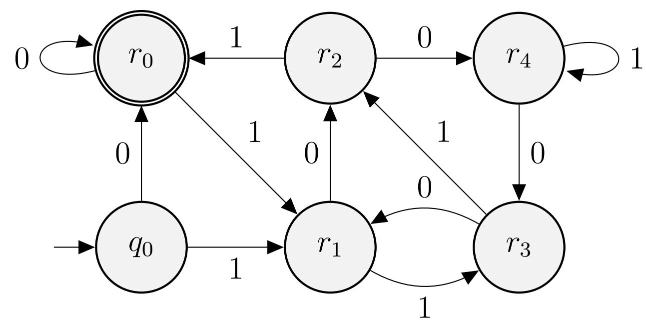
Soluzione. Ragionamento identico a quello dell'esercizio precedente.



Esercizio 3.15

Verificare che il seguente linguaggio su $\Sigma = \{0, 1\}$ è regolare: insieme delle stringhe tali che, se interpretate come numero decimale, sono divisibili per 5.

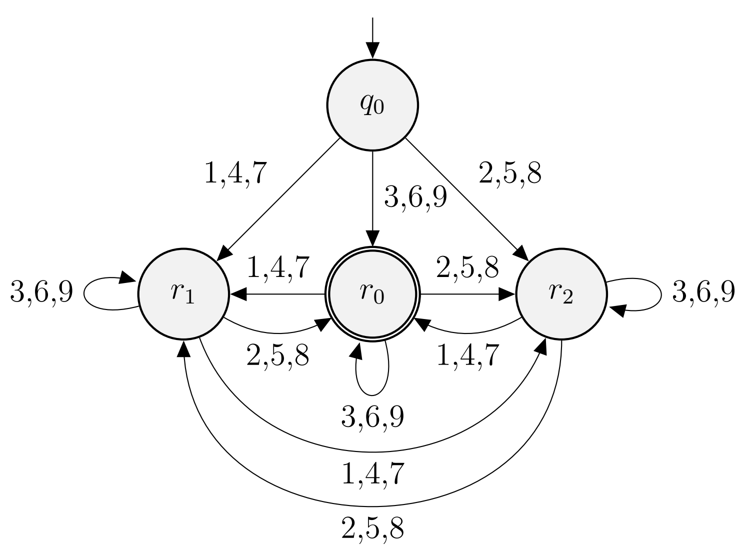
Soluzione. Ragionamento simile a quello dell'esercizio precedente.



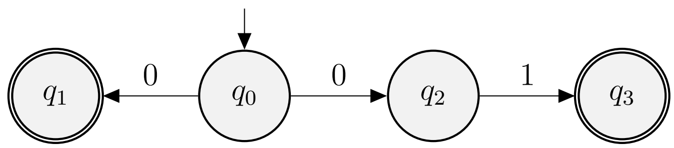
Esercizio 3.16

Veriricare che il seguente linguaggio su $\Sigma = \{0, 1\}$ è regolare: insieme delle stringhe di cifre decimali tali che, sono divisibili per 3. [*Suggerimento*: un numero decimale è divisibile per 3 se la somma delle sue cifre è un multiplo di 3.]

Soluzione. Seguendo il suggerimento, costruiamo un DFA che si muove tra gli stati $\dot{r}_0, r_1, r_2$ a seconda se la somma delle cifre che compongono la stringa divisa per 3 da resto 0, resto 1 o resto 2 rispettivamente.



3.3 NFA: automi a stati finiti non deterministici



	0	**1**
$\boldsymbol{q_0}$	$\{ q_1, q_2 \}$	$\varnothing$
$\dot{\boldsymbol{q}}_{\boldsymbol{1}}$	$\varnothing$	$\varnothing$
$\boldsymbol{q_2}$	$\varnothing$	$\{ q_3 \}$
$\dot{\boldsymbol{q}}_{\boldsymbol{3}}$	$\varnothing$	$\varnothing$

Per come sono definiti gli NFA, a partire da uno stato $q \in Q$, con la lettura un simbolo $a \in \Sigma$, è possibile transire verso nessuno stato, verso uno solo stato $q' \in Q$ o verso più stati contemporaneamente (un insieme di stati $S \subseteq Q$). Questo equivale a dire che, negli NFA, a partire da uno stato $q \in Q$, con la lettura di un simbolo $a \in \Sigma$, si transita in un insieme di stati $S \in \wp(Q)$.

3.3.1 Definizioni, Teoremi e Lemmi

Definizione 3.5 (NFA: automa a stati finiti non deterministico)**.** *Un NFA è una quintupla $\langle Q, \Sigma, \delta, q_0, F \rangle$, la cui definizione differisce dalla quintupla dei DFA solo nella funzione di transizione che è definita come:*

$$\delta : Q \times \Sigma \mapsto \wp(Q).$$

La differenza è che, negli NFA, la funzione di transizione δ non restituisce in uscita un singolo stato che sta in Q ma invece restituisce un *sottoinsieme* degli stati che stanno in Q, ovvero un elemento di $\wp(Q)$. L'aggettivo *non deterministico* nel nome è dovuto proprio a questo; l'NFA, a seguito della lettura di un simbolo può transitare verso più stati e quindi trovarsi in più stati contemporaneamente ovvero, non è più deterministico.

Come per i DFA, anche per gli NFA è possibile utilizzare la funzione di transizione δ per la definizione della funzione di transizione su stringhe $\hat{\delta}$.

Definizione 3.6 (Funzione di transizione su stringhe)**.** *Sia M l'NFA descritto dalla quintupla $\langle Q, \Sigma, \delta, q_0, F \rangle$. Sia x una stringa su Σ^*. La funzione di transizione su stringhe $\hat{\delta} : Q \times \Sigma^* \mapsto \wp(Q)$ è definita come:*

$$\hat{\delta}(q, x) := \begin{cases} \{ q \} & \textit{se } x = \epsilon; \\ \bigcup_{p \in \hat{\delta}(q,y)} \delta(p, a) & \textit{se } x = ya. \end{cases}$$

Osservazione 3.3. *Nella definizione di $\hat{\delta}$ vi è un elemento di arbitrarietà. $\hat{\delta}$ si può sostituire con la funzione $\check{\delta}$ così definita:*

$$\check{\delta}(q, x) := \begin{cases} \{ q \} & \textit{se } x = \epsilon; \\ \bigcup_{p \in \delta(q,a)} \hat{\delta}(p, y) & \textit{se } x = ay. \end{cases}$$

Definizione 3.7 (Accettazione di una stringa)**.** *Sia M un NFA descritto dalla quintupla $\langle Q, \Sigma, \delta, q_0, F \rangle$. Sia x una stringa su Σ^*. La stringa x è* accettata *da M se e solo se:*

$$\hat{\delta}(q_0, x) \cap F \neq \varnothing.$$

Una stringa x quindi è accettata da un NFA M se, partendo dallo stato iniziale q_0 e leggendo x, l'automa termina in un insieme di stati che ha intersezione non nulla con F.

Definizione 3.8 (Linguaggio accettato da un NFA)**.** *Sia M l'NFA descritto dalla quintupla $\langle Q, \Sigma, \delta, q_0, F \rangle$. Il linguaggio* accettato *da M si indica con $L(M)$ ed è l'insieme di tutte le stringhe accettate da M, ovvero:*

$$L(M) := \{ x \in \Sigma^* \mid \hat{\delta}(q_0, x) \cap F \neq \varnothing \}.$$

Teorema 3.2 (Rabin-Scott: equivalenza tra NFA e DFA)**.** *Sia $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ un NFA, allora esiste un DFA M' tale che*

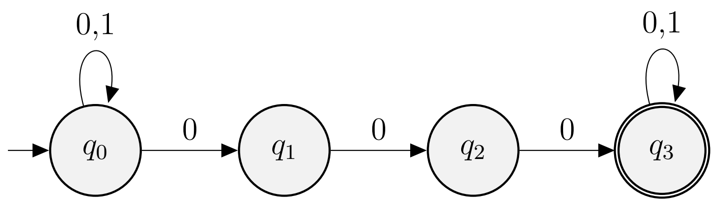
$$L(M) = L(M').$$

3.3.2 Costruzione per sottoinsiemi

Dato un NFA $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ è possibile applicare un algoritmo che permette di trovare il DFA equivalente, questo algoritmo prende il nome di *costruzione per sottoinsiemi.* Come abbiamo visto, la funzione di transizione

δ su NFA restituisce in uscita un elemento che sta in $\wp(Q)$, quindi restituisce un insieme di stati e questo comporta che l'NFA si può trovare contemporaneamente in più stati. L'idea è di costruire un DFA M' che ha uno stato per ogni possibile elemento in $\wp(Q)$, così facendo, in ogni momento è possibile "riassumere" l'insieme di stati in cui si trova l'NFA M con un singolo stato del DFA M'. Notare che, il DFA M' rispetto all'equivalente NFA M avrà un numero di stati che può risultare esponenzialmente più grande, infatti nel caso pessimo possono servire proprio $|\wp(Q)| = 2^{|Q|}$ stati.

Esempio. Sia $\Sigma = \{0, 1\}$. Grafo e matrice di transizione per un NFA che accetta il linguaggio $L = \{x \in \{0,1\}^* \mid x = w000y, w, y \in \Sigma^*\}$ (stringhe aventi almeno tre '0' consecutivi"):



	0	**1**
$\boldsymbol{q_0}$	$\{q_0, q_1\}$	$\{q_0\}$
$\boldsymbol{q_1}$	$\{q_2\}$	$\varnothing$
$\boldsymbol{q_2}$	$\{q_3\}$	$\varnothing$
$\dot{\boldsymbol{q}}_{\boldsymbol{3}}$	$\{q_3\}$	$\{q_3\}$

Costruzione per sottoinsiemi:

1. per definizione di NFA, all'interno delle celle della matrice di transizione troviamo insiemi di stati $S_1 \cdots S_n$. Tra questi identifichiamo gli stati S_i tali che $|S_i| \geq 2$ (ad esempio $\{q_0, q_1\}$) e sono raggiungibili da q_0. Se questi non sono già stati "etichettati", etichettiamo questi insiemi di stati, ovvero associamo ad ognuno di questi un nuovo e singolo stato che li rappresenta, ad esempio $\{q_0, q_1\} = q_{01}$.

   Se all'interno dell'insieme di stati che si sta etichettando è presente almeno uno stato finale, lo stato che si utilizza come etichetta diventa anche stato finale del DFA equivalente che stiamo costruendo.

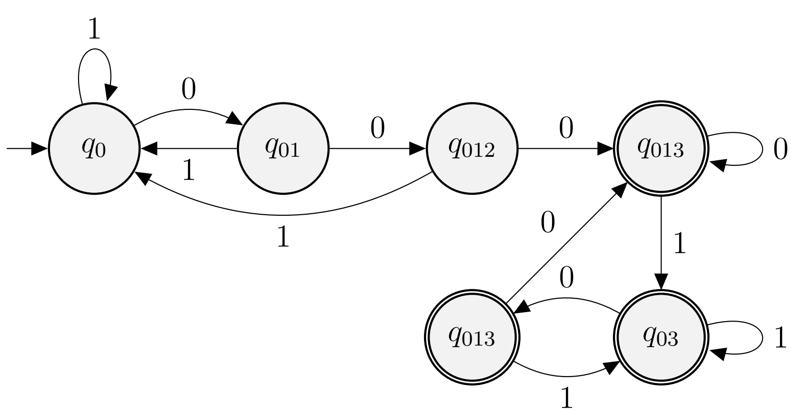
Per ogni nuovo insieme di stati trovato ed etichettato, aggiungiamo una riga alla matrice con tutte le celle vuote tranne la prima (che conterrà proprio l'insieme di stati oppure in maniera equivalente il nuovo stato-etichetta che gli abbiamo associato);

2. riempire le celle vuote, univocamente identificabili dalla coppia (S, a), inserendovi dentro $\bigcup_{s \in S} \delta(s, a)$.

3. ripetere dal punto 1 fino a che non si raggiunge un punto fisso.

	0	**1**
$\boldsymbol{q_0}$	q_{01}	q_0
$\boldsymbol{q_{01}}$	q_{012}	q_0
$\boldsymbol{q_{012}}$	q_{0123}	q_0
$\boldsymbol{\dot{q}_{0123}}$	q_{0123}	q_{03}
$\boldsymbol{\dot{q}_{03}}$	q_{013}	q_{03}
$\boldsymbol{\dot{q}_{013}}$	q_{0123}	q_{03}

Grafo di transizione del DFA equivalente con i soli stati raggiungibili a partire da q_0:



Notare che gli stati q_{013}, q_{03} potrebbero essere eliminati, il DFA generato utilizzando la costruzione per sottoinsiemi quindi non è *minimo*.

3.3.3 Esercizi

Esercizio 3.17

Sia Q un insieme di stati. Sia Σ un alfabeto. Quanti sono gli NFA che si possono costruire fissati Q e Σ?

Soluzione. Un NFA è una quintupla $M = \langle Q, \Sigma, \delta, q_0, F \rangle$, se Q possiamo contare i possibili NFA nel seguente modo:

- numero di modi in cui possiamo scegliere lo stato iniziale q_0 è
$$|Q|;$$
- numero di modi in cui possiamo scegliere l'insieme degli stati finali $F \subseteq Q$ è
$$|\wp(Q)| = 2^{|Q|};$$
- numero di modi in cui possiamo definire la funzione di transizione δ è
$$\left(2^{|Q|}\right)^{|Q||\Sigma|},$$
perché ogni in ogni cella della matrice di transizione possiamo inserire $2^{|Q|}$ valori differenti, e le dimensioni della matrice sono $|Q||\Sigma|$.

Quindi, fissati Q e Σ, è possibile costruire

$$|Q| \cdot 2^{|Q|} \cdot \left(2^{|Q|}\right)^{|Q||\Sigma|} = |Q| \cdot 2^{|Q|} \cdot 2^{|Q|^2|\Sigma|} = |Q| \cdot 2^{|Q|^2|\Sigma|+|Q|} = |Q| \cdot 2^{|Q|(|Q||\Sigma|+1)}$$

differenti NFA.

Esercizio 3.18

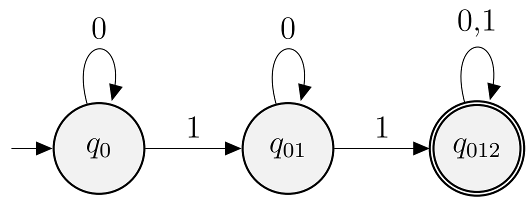
Determinare il DFA equivalente al seguente NFA e il linguaggio accettato.

	0	**1**
$\boldsymbol{q_0}$	$\{ q_0 \}$	$\{ q_0, q_1 \}$
$\boldsymbol{q_1}$	$\{ q_1 \}$	$\{ q_0, q_2 \}$
$\dot{\boldsymbol{q}}_{\boldsymbol{2}}$	$\{ q_1, q_2 \}$	$\{ q_0, q_1, q_2 \}$

Soluzione. Applichiamo la costruzione per sottoinsiemi:

	0	**1**
$\boldsymbol{q_0}$	q_0	q_{01}
$\boldsymbol{q_{01}}$	q_{01}	q_{012}
$\dot{\boldsymbol{q}}_{\mathbf{012}}$	q_{012}	q_{012}

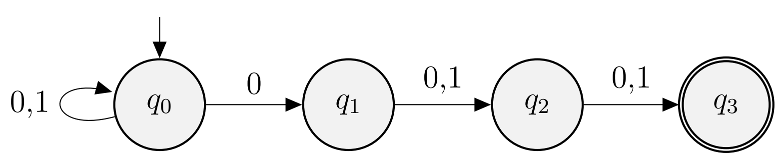
Grafo di transizione dei soli stati raggiungibili a partire da q_0:



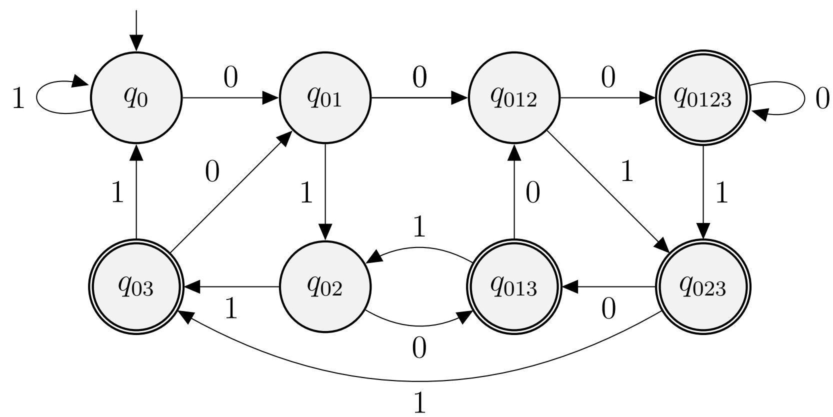
$$L(M) = \{\, x \in \Sigma^* \mid x \text{ contiene almeno due occorrenze del simbolo '1'} \,\}.$$

Esercizio 3.19

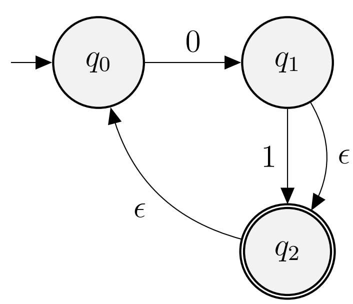
Costruire un NFA con 4 stati che accetta il seguente linguaggio: $L = \{\, x \in \{\,0,1\,\} \mid x \text{ ha '0' come terzultimo simbolo} \,\}$ e trovare il DFA equivalente.



	0	**1**
$\boldsymbol{q_0}$	q_{01}	q_0
$\boldsymbol{q_{01}}$	q_{012}	q_{02}
$\boldsymbol{q_{012}}$	q_{0123}	q_{023}
$\boldsymbol{q_{02}}$	q_{013}	q_{03}
$\boldsymbol{\dot{q}_{0123}}$	q_{0123}	q_{023}
$\boldsymbol{\dot{q}_{023}}$	q_{013}	q_{03}
$\boldsymbol{\dot{q}_{013}}$	q_{012}	q_{02}
$\boldsymbol{\dot{q}_{03}}$	q_{01}	q_0



3.4 ϵ-NFA: automi a stati finiti non deterministici con ϵ-transizioni



	$\boldsymbol{\epsilon}$	**0**	**1**
$\boldsymbol{q_0}$	$\varnothing$	$\{ q_1 \}$	$\varnothing$
$\boldsymbol{q_1}$	$\{ q_2 \}$	$\varnothing$	$\{ q_2 \}$
$\dot{\boldsymbol{q}}_{\boldsymbol{2}}$	$\{ q_0 \}$	$\varnothing$	$\varnothing$

Per come sono definiti gli ϵ-NFA, a partire da uno stato $q \in Q$, è possibile transire verso un insieme di stati $S \in \wp(Q)$ anche senza aver letto nessun simbolo, questo è possibile grazie a particolari transizioni, le *ϵ-transizioni.*

3.4.1 Definizioni, Teoremi e Lemmi

Definizione 3.9 (ϵ-NFA: automa a stati finiti non deterministico con ϵ-transizioni)**.** *Un ϵ-NFA è una quintupla $\langle Q, \Sigma, \delta, q_0, F \rangle$, la cui definizione differisce dalla quintupla degli NFA solo nella funzione di transizione che è definita come:*

$$\delta : Q \times (\Sigma \cup \{ \epsilon \}) \mapsto \wp(Q).$$

La funzione di transizione negli ϵ-NFA quindi può prendere in ingresso sia simboli dell'alfabeto che il simbolo ϵ da cui derivano le ϵ-transizioni. Prima di definire la funzione di transizione su stringhe, definiamo formalmente *ϵ-step* ed *ϵ-closure* che consistono nell'applicare delle ϵ-transizioni ad insiemi di stati.

Definizione 3.10 (Operatore)**.** *Una funzione f è detta* operatore *se dominio e codominio di f sono lo stesso insieme.*

E' lecito chiedersi cosa succede applicando più volte un operatore.

Definizione 3.11 (Applicazione iterata di un operatore)**.** *Sia S un insieme e $a \in S$. Sia $f : S \mapsto S$.*

$$f^n(a) := \begin{cases} a, & \textit{se } n = 0; \\ f\big(f^{n-1}(a)\big), & \textit{altrimenti.} \end{cases}$$

Definizione 3.12 (UCO: operatore di chiusura superiore)**.** *Sia* $(S, \subseteq)$ *un insieme ordinato dalla relazione* $\subseteq$*.* $f : S \mapsto S$ *è un UCO se e solo se valgono le seguenti condizioni:*

1. f *è* monotono*:* $\forall x, y \in S : x \subseteq y \Longrightarrow f(x) \subseteq f(y)$*;*
2. f *è* estensivo*:* $\forall x \in S : x \subseteq f(x)$*;*
3. f *è* idempotente*:* $\forall x \in S : f(x) = f\big(f(x)\big)$*.*

Definizione 3.13 (ϵ-step)**.** *Sia* M *un* ϵ*-NFA definito dalla quintupla* $\langle Q, \Sigma, \delta, q_0, F\rangle$ *e sia* $S \in \wp(Q)$*. L'operatore* ϵ*-step*: $\wp(Q) \mapsto \wp(Q)$ *è definito come:*

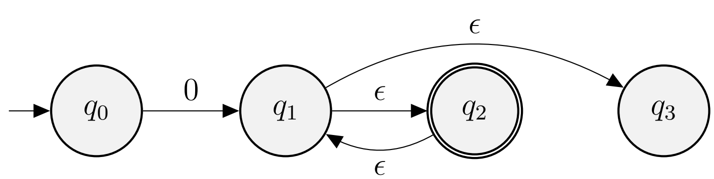
$$\epsilon\text{-}step(S) = \{\, q \in Q \mid \exists p \in S \,.\, q \in \delta(p, \epsilon) \,\}.$$

L'ϵ-step applicata ad un insieme S di stati quindi ritorna come risultato un insieme di stati che è ottenuto applicando le *epsilon*-transizioni a tutti gli elementi di S.

Definizione 3.14 (ϵ-step iterati)**.** *Sia* M *un* ϵ*-NFA definito dalla quintupla* $\langle Q, \Sigma, \delta, q_0, F\rangle$ *e sia* $S \in \wp(Q)$*,*

$$\epsilon\text{-}step^n(S) := \begin{cases} S & \textit{se } n = 0; \\ \epsilon\text{-}step(\epsilon\text{-}step^{n-1}(S)) & \textit{altrimenti.} \end{cases}$$

Esempio. Grafo di transizione di un ϵ-NFA M:



Alcuni ϵ-step sugli stati di M:

$$\epsilon\text{-step}^0\big(\{\, q_0 \,\}\big) = \{\, q_0 \,\} \qquad [\text{def. } \cdot^0].$$

$$\epsilon\text{-step}\big(\{\, q_0 \,\}\big) = \varnothing \qquad\qquad [\text{def. } \epsilon\text{-step}].$$

$$\epsilon\text{-step}\big(\{\, q_1 \,\}\big) = \{\, q_2, q_3 \,\} \qquad\qquad [\text{def. } \epsilon\text{-step}].$$

$$\begin{aligned}
\epsilon\text{-step}^2\big(\{\, q_1 \,\}\big) &= \epsilon\text{-step}\big(\epsilon\text{-step}(\epsilon\text{-step}^0(\{\, q_1 \,\}))\big) && [\text{def. } \cdot^n] \\
&= \epsilon\text{-step}\big(\epsilon\text{-step}(\{\, q_1 \,\})\big) && [\text{def. } \cdot^0] \\
&= \epsilon\text{-step}\big(\{\, q_2, q_3 \,\}\big) && [\text{def. } \epsilon\text{-step}] \\
&= \{\, q_1 \,\} && [\text{def. } \epsilon\text{-step}].
\end{aligned}$$

Definizione 3.15 (ϵ-closure)**.** *Sia M un ϵ-NFA definito dalla quintupla $\langle Q, \Sigma, \delta, q_0, F \rangle$. Sia $S \in \wp(Q)$. La ϵ-closure : $\wp(Q) \mapsto \wp(Q)$ è la chiusura riflessiva e transitiva dell'ϵ-step, ovvero:*

$$\epsilon\text{-}closure(S) := \bigcup_{i \in \mathbb{N}} \epsilon\text{-}step^i(S).$$

Per poter applicare la ϵ-closure anche ad un singolo stato q senza dover per forza utilizzare la notazione insiemistica $\{\, q \,\}$ definiamo ϵ-closure : $Q \mapsto \wp(Q)$ come:

$$\epsilon\text{-}closure(q) := \bigcup_{i \in \mathbb{N}} \epsilon\text{-}step^i(\{\, q \,\}).$$

Esempio.

$$\begin{aligned}
\epsilon\text{-closure}(q_1) &= \epsilon\text{-step}^0(\{\, q_1 \,\}) \cup \epsilon\text{-step}(\{\, q_1 \,\}) \cup \epsilon\text{-step}^2 \cdots && [\text{def. } \epsilon\text{-closure}] \\
&= \{\, q_1 \,\} \cup \{\, q_2, q_3 \,\} \cup \{\, q_1 \,\} \cup \{\, q_2, q_3 \,\} \cup \cdots && [\text{def. } \epsilon\text{-step}] \\
&= \{\, q_1, q_2, q_3 \,\} && [\text{def. } \cup].
\end{aligned}$$

Notare che anche se l'ϵ-closure è un unione su tutto $\mathbb{N}$, dal momento che gli stati dell'automa sono finiti e quindi anche il numero di transizioni sono finite, si arriva ad un certa i per la quale applicare ϵ-step^i aggiunge soltando duplicati al nostro insieme quindi il punto fisso è stato raggiunto e non serve continuare, nell'esempio il punto fisso si raggiunge per $i = 1$, ovvero per $i = 2$ viene aggiunto un duplicato.

Lemma 3.2 (Monotonia di ϵ-closure)**.** *Sia Q un insieme di stati di un ϵ-NFA. Siano $R, S \subseteq Q$. ϵ-closure è un operatore monotono quindi vale che*

$$(R \subseteq S) \Longrightarrow \big(\epsilon\text{-}closure(R) \subseteq \epsilon\text{-}closure(S)\big).$$

Dimostrazione.

$$\begin{aligned}
\epsilon\text{-closure}(R) &= \bigcup_{i\in\mathbb{N}} \epsilon\text{-step}^i(R) && [\text{def. } \epsilon\text{-closure}] \\
&\subseteq \bigcup_{i\in\mathbb{N}} \epsilon\text{-step}^i(S) && [R \subseteq S, \text{ def. } \epsilon\text{-step}] \\
&= \epsilon\text{-closure}(S).
\end{aligned}$$

□

Lemma 3.3 (Estensività di ϵ-closure)**.** *Sia S un insieme di stati di un ϵ-NFA. ϵ-closure è un operatore estensivo quindi vale che*

$$S \subseteq \epsilon\textit{-closure}(S).$$

Dimostrazione.

$$\begin{aligned}
\epsilon\text{-closure}(S) &= \bigcup_{i\in\mathbb{N}} \epsilon\text{-step}^i(S) && [\text{def. } \epsilon\text{-closure}] \\
&= \epsilon\text{-step}^0(S) \cup \bigcup_{i\geq 1} \epsilon\text{-step}^i(S) && [\text{def. } \cup] \\
&= S \cup \bigcup_{i\geq 1} \epsilon\text{-step}^i(S) && [\text{def. } \cdot^0].
\end{aligned}$$

□

Lemma 3.4 (Idempotenza di ϵ-closure)**.** *Sia $q \in Q$ uno stato.*

$$\epsilon\textit{-closure}(\epsilon\textit{-closure}(q)) = \epsilon\textit{-closure}(q).$$

Dimostrazione. Notiamo che

$$\forall S \subseteq Q, \forall i \in \mathbb{N} : \epsilon\text{-step}^i(S) \subseteq \epsilon\text{-closure}(S) \qquad [\text{def. } \epsilon\text{-closure}]. \tag{5}$$

$$
\begin{aligned}
\epsilon\text{-closure}(\epsilon\text{-closure}(q)) &= \bigcup_{i\in\mathbb{N}} \epsilon\text{-step}^i\big(\epsilon\text{-closure}(q)\big) && [\text{def. } \epsilon\text{-closure}]\\
&= \bigcup_{i\in\mathbb{N}} \epsilon\text{-step}^i\big(\epsilon\text{-closure}(q)\big) && [\text{def. } \epsilon\text{-closure}]\\
&= \epsilon\text{-step}^0\big(\epsilon\text{-closure}(q)\big) \cup \bigcup_{i\geq 1} \epsilon\text{-step}^i\big(\epsilon\text{-closure}(q)\big) && [0\in\mathbb{N}]\\
&= \epsilon\text{-closure}(q) \cup \underbrace{\bigcup_{i\geq 1} \epsilon\text{-step}^i\big(\epsilon\text{-closure}(q)\big)}_{\subseteq \epsilon\text{-closure}} && [\text{def. } \cdot^0]\\
&= \epsilon\text{-closure}(q) && [(5), \text{Lemma 4.1}].
\end{aligned}
$$

□

Lemma 3.5 (ϵ-closure è un UCO)**.** *Per la ϵ-closure valgono le seguenti condizioni:*

1. *monotonia;*
2. *idempotenza;*
3. *estensività.*

Quindi ϵ-closure è un UCO.

Lemma 3.6 (Additività di ϵ-closure)**.** *Sia Q l'insieme degli stati di un ϵ-NFA. Siano $R, S \subseteq Q$. ϵ-closure è un operatore additivo quindi vale che*

$$\epsilon\textit{-closure}(R \cup S) = \epsilon\textit{-closure}(R) \cup \epsilon\textit{-closure}(S).$$

Definizione 3.16 (Funzione di transizione su stringhe)**.** *Sia M un ϵ-NFA descritto dalla quintupla $\langle Q, \Sigma, \delta, q_0, F\rangle$. Sia x una stringa su Σ^*. La funzione di transizione su stringhe $\hat{\delta} : Q \times \Sigma^* \mapsto \wp(Q)$ è definita come:*

$$\hat{\delta}(q, x) := \begin{cases} \epsilon\textit{-closure}(q) & \textit{se } x = \epsilon;\\ \bigcup_{p\in\hat{\delta}(q,y)} \epsilon\textit{-closure}\big(\delta(p, a)\big) & \textit{se } x = ya. \end{cases}$$

Osservazione 3.4. *Nella definizione di $\hat{\delta}$ vi è un elemento di arbitrarietà. $\hat{\delta}$ si può sostituire con la funzione $\check{\delta}$ così definita:*

$$\check{\delta}(q, x) := \begin{cases} \epsilon\textit{-closure}(q) & \textit{se } x = \epsilon;\\ \bigcup_{r\in\left(\bigcup_{p\in\epsilon\textit{-closure}(q)} \delta(p,a)\right)} \hat{\delta}(r, y) & \textit{se } x = ay \end{cases}$$

Nella pratica, se un ϵ-NFA M si trova in un insieme di stati S, prima di leggere il prossimo simbolo in ingresso a, viene applicata la ϵ-closure(S) che ha come conseguenza la transizione di M dall'insieme di stati S verso un insieme $S' = \epsilon$-closure(S). Per ogni stato in S' è poi possibile procedere con la lettura del prossimo simbolo in ingresso a che farà transitare M verso un altro insieme di stati $S'' = \bigcup_{s \in S'} \delta(s, a)$, al quale verrà nuovamente applicata la ϵ-closure prima di leggere il prossimo simbolo sul nastro. Si procede in questo modo finché tutti i simboli sul nastro sono stati letti, ovvero fino a che sul nastro di ingresso rimane la stringa vuota ϵ. Con la sola stringa vuota sul nastro di ingresso, si applica un ultima volta la ϵ-closure ottenendo un ultimo insieme di stati T. Se T ha intersezione non vuota con F, allora la stringa che è stata letta un simbolo alla volta è accettata da M.

Definizione 3.17 (Accettazione di una stringa)**.** *Sia M un ϵ-NFA descritto dalla quintupla $\langle Q, \Sigma, \delta, q_0, F \rangle$. Sia x una stringa su Σ^*. La stringa x è* accettata *da M se e solo se:*

$$\hat{\delta}(q_0, x) \cap F \neq \varnothing.$$

Una stringa x quindi è accettata da un ϵ-NFA M se, partendo dallo stato iniziale q_0 e leggendo x, l'automa termina in un insieme di stati che ha intersezione non nulla con F.

Definizione 3.18 (Linguaggio accettato da un NFA)**.** *Sia M un ϵ-NFA descritto dalla quintupla $\langle Q, \Sigma, \delta, q_0, F \rangle$. Il linguaggio* accettato *da M si indica con $L(M)$ ed è l'insieme di tutte le stringhe accettate da M, ovvero:*

$$L(M) := \{\, x \in \Sigma^* \mid \hat{\delta}(q_0, x) \cap F \neq \varnothing \,\}.$$

Teorema 3.3 (Equivalenza tra ϵ-NFA e NFA)**.** *Sia $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ un ϵ-NFA, allora esiste un NFA M' tale che*

$$L(M) = L(M').$$

Corollario 3.1 (Equivalenza tra DFA, NFA e ϵ-NFA)**.** *Le classi di linguaggi accettati da DFA, NFA ed ϵ-NFA coincidono (linguaggi regolari).*

3.4.2 Eliminazione delle ϵ-transizioni

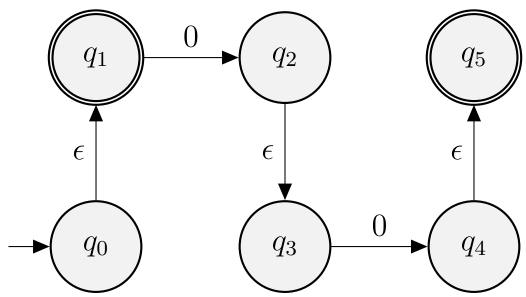
Dato un ϵ-NFA, per trovare un NFA equivalente, è necessario eliminare le ϵ-transizioni. Per fare questo, eseguiamo i seguenti passaggi $\forall q \in Q, \forall a \in \Sigma$:

1. applichiamo ϵ-closure(q), ottenendo un insieme di stati S. Se $q = q_0$ e ϵ-closure$(q_0) \cap F \neq \varnothing$ allora q_0 oltre che stato iniziale diventa anche stato accettante nell'NFA equivalente che stiamo costruendo;

2. $\forall s \in S$ applichiamo la $\delta(s, a)$ e collezioniamo gli stati ottenuti in un insieme che chiameremo $S'' = \bigcup_{s \in S} \delta(s, a)$;

3. applichiamo la ϵ-closure$(S'') = S'''$ ottenendo un insieme di stati che, se stiamo costruendo una matrice di transizione, andremo a scrivere in corrispondenza della cella $[q, a] = S'''$.

Esempio. Descrizione di un ϵ-NFA tramite grafo di transizione:



Per completare ad esempio la cella $(\dot{q}_0, 0)$ della matrice di transizione del NFA qeuivalente partiamo dallo stato q_0 del ϵ-NFA e applichiamo:

$$q_0 \overset{\epsilon\text{-closure}}{\longrightarrow} \{\, q_0, q_1 \,\} \overset{0}{\longrightarrow} \{\, q_2 \,\} \overset{\epsilon\text{-closure}}{\longrightarrow} \{\, q_2, q_3 \,\}.$$

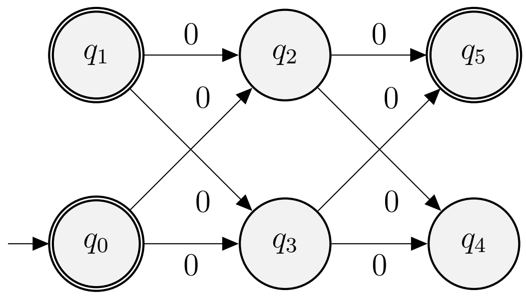
dal momento che nella ϵ-closure(q_0) è presente lo stato finale $\dot{q}_1$, lo stato q_0 nel NFA equivalente diventa stato finale. Stesso procedimento per completare la cella $(q_1, 0)$:

$$q_1 \overset{\epsilon\text{-closure}}{\longrightarrow} \{\, q_1 \,\} \overset{0}{\longrightarrow} \{\, q_2 \,\} \overset{\epsilon\text{-closure}}{\longrightarrow} \{\, q_2, q_3 \,\}.$$

Ripetendo questo processo per ogni stato dell ϵ-NFA e ogni simbolo dell'afabeto, si ottiene la matrice di transizione dell'NFA equivalente:

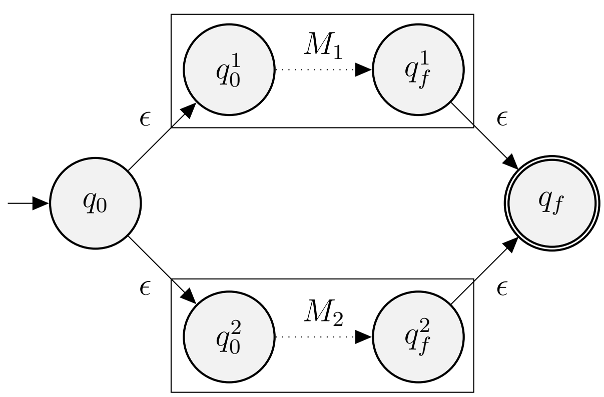
	0	**1**
$\dot{\boldsymbol{q}}_{\mathbf{0}}$	$\{ q_2, q_3 \}$	$\varnothing$
$\dot{\boldsymbol{q}}_{\mathbf{1}}$	$\{ q_2, q_3 \}$	$\varnothing$
$\boldsymbol{q}_{\mathbf{2}}$	$\{ q_4, q_5 \}$	$\varnothing$
$\boldsymbol{q}_{\mathbf{3}}$	$\{ q_4, q_5 \}$	$\varnothing$
$\boldsymbol{q}_{\mathbf{4}}$	$\varnothing$	$\varnothing$
$\dot{\boldsymbol{q}}_{\mathbf{5}}$	$\varnothing$	$\varnothing$

Grafo di transizione del NFA equivalente:



Nota su ϵ-transizioni. Le ϵ-transizioni sono utili per "cucire" insieme diversi DFA. L'unica accortezza è quella che ciasun DFA deve avere un unico stato finale; nel caso un DFA abbia più stati finali basta collegarli con ϵ-transizioni ad un nuovo stato che sarà l'unico finale. Siano ad esempio M_1, M_2 DFA e $L_1 = L(M_1), L_2 = L(M_2)$.

E' estremamente semplice costruire un ϵ-NFA che accetta il linguaggio $L_1 \cup L_2$:



Dimostrazione. Per costruzione dell'automa vale che:

$$q_0^{(1)}, q_0^{(2)} \in \epsilon\text{-closure}(q_0). \tag{6}$$

$$q_f \in \epsilon\text{-closure}\big(q_f^{(1)}\big). \tag{7}$$

$$q_f \in \epsilon\text{-closure}\big(q_f^{(2)}\big). \tag{8}$$

Mostriamo ($\Longrightarrow$): sia $x \in \Sigma^*$, se vale

$$\big(\hat{\delta}(q_0^{(1)}, x) = q_f^{(1)} \vee \hat{\delta}(q_0^{(2)}, x) = q_f^{(2)}\big)$$

allora vale anche che

$$\begin{aligned}
&\Bigg(\bigcup_{p \in \epsilon\text{-closure}(q_0)} \hat{\delta}(p, x)\Bigg) \cap \{\, q_f^{(1)}, q_f^{(2)} \,\} && [(6)]\\
&\Longleftrightarrow \hat{\delta}(q_0, \epsilon x) \cap \{\, q_f^{(1)}, q_f^{(2)} \,\} && [\text{def. } \hat{\delta}]\\
&\Longleftrightarrow \hat{\delta}(q_0, x) \cap \{\, q_f^{(1)}, q_f^{(2)} \,\} && [\epsilon \text{ è ident.}].
\end{aligned}$$

Quindi abbiamo mostrato che

$$\big(\hat{\delta}(q_0^{(1)}, x) = q_f^{(1)} \vee \hat{\delta}(q_0^{(2)}, x) = q_f^{(2)}\big) \Longrightarrow \hat{\delta}(q_0, x) \cap \{\, q_f^{(1)}, q_f^{(2)} \,\}. \tag{9}$$

allora vale anche

$$\begin{aligned}
\hat{\delta}(q_0, x) &= \hat{\delta}(q_0, x\epsilon) && [\epsilon \text{ è identità}]\\
&= \hat{\delta}\big(\hat{\delta}(q_0, x), \epsilon\big) && [\text{prop. di } \hat{\delta}]\\
&= \epsilon\text{-closure}\big(\hat{\delta}(q_0, x)\big) && [\text{def. } \hat{\delta}],
\end{aligned}$$

da cui segue che

$$\begin{aligned}
&q_f \in \epsilon\text{-closure}\big(\hat{\delta}(q_0, x)\big) && [(9), (7), (8)]\\
&= \hat{\delta}(q_0, x) && [\text{def. di } \hat{\delta}].
\end{aligned}$$

Mostriamo ($\Longleftarrow$): sia $x = wa \in \Sigma^*$, se vale

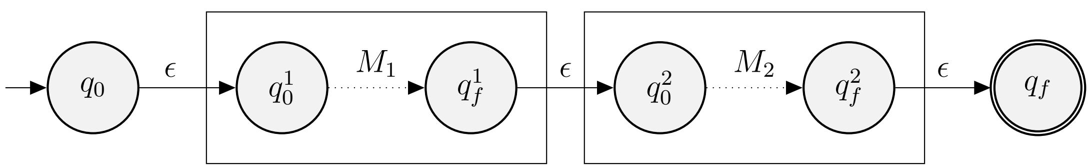
$$\hat{\delta}(q_0, x) \cap \{\, q_f \,\}$$

allora

$$
\begin{aligned}
& q_f \in \hat{\delta}(q_0, wa) && [x = wa] \\
\iff & q_f \in \bigcup_{p \in \hat{\delta}(q_0, w)} \epsilon\text{-closure}\big(\delta(p, a)\big) && [\text{def. } \hat{\delta}] \\
\iff & \exists p \in \hat{\delta}(q_0, w)\ .\ q_f \in \epsilon\text{-closure}\big(\delta(p, a)\big) \\
\iff & \exists p^{(i)} \in \delta(q_0^{(i)}, w)\ .\ q_f^{(i)} \in \epsilon\text{-closure}\big(\delta(p^{(i)}, a)\big) && [(7),\ (8),\ i = 1 \lor i = 2] \\
\iff & \big(\hat{\delta}(q_0^{(1)}, x) = q_f^{(1)} \lor \hat{\delta}(q_0^{(2)}, x) = q_f^{(2)}\big).
\end{aligned}
$$

□

E' estremamente semplice anche costruire un altro ϵ-NFA che accetta il linguaggio $L_1 \cdot L_2$:



Dimostrazione. Sia $xy \in \Sigma^*$. Vogliamo mostrare che xy è accettata da M se e solo se x è accettata da M_1 e y è accettata da M_2, ovvero

$$\big(\hat{\delta}(q_0^1, x) = q_f^1 \land \hat{\delta}(q_0^2, y) = q_f^2\big) \iff \big(\hat{\delta}(q_0, xy) \cap q_f \neq \varnothing\big).$$

(Lasciato per esercizio). □

Nota su automi a stati finiti in generale.

1. Fornire a un DFA la possibilità di produrre output, eventualmente scrivendo sul nastro, non ne aumenta la capacità computazionale perché ciò che viene scritto non potrebbe essere mai letto.

2. Fornire a un DFA la possibilità di muovere la testina sia a sinistra che a destra non ne aumenta la capacità computazionale perché i dati letti rimarrebbero sempre gli stessi.

3. La combinazione dei punti 1 e 2 permette di passare a un formalismo più potente (*macchine di Turing*).

3.4.3 Esercizi

Esercizio 3.20

Sia Q un insieme di stati. Sia Σ un alfabeto. Quanti sono gli ϵ-NFA che si possono costruire fissati Q e Σ?

Soluzione. Un ϵ-NFA è una quintupla $M = \langle Q, \Sigma, \delta, q_0, F \rangle$, se Q possiamo contare i possibili ϵ-NFA nel seguente modo:

- numero di modi in cui possiamo scegliere lo stato iniziale q_0 è
$$|Q|;$$
- numero di modi in cui possiamo scegliere l'insieme degli stati finali $F \subseteq Q$ è
$$|\wp(Q)| = 2^{|Q|};$$
- numero di modi in cui possiamo definire la funzione di transizione δ è
$$\left(2^{|Q|}\right)^{|Q|(|\Sigma|+1)},$$
perché ogni in ogni cella della matrice di transizione possiamo inserire $2^{|Q|}$ valori differenti, e le dimensioni della matrice sono $|Q|\big(|\Sigma| + 1\big)$.

Quindi, fissati Q e Σ, è possibile costruire

$$|Q| \cdot 2^{|Q|} \cdot \left(2^{|Q|}\right)^{|Q|(|\Sigma|+1)} = |Q| \cdot 2^{|Q|} \cdot 2^{|Q|^2(|\Sigma|+1)} = |Q| \cdot 2^{|Q|^2(|\Sigma|+1)+|Q|}$$

differenti ϵ-NFA.

4 Espressioni Regolari

4.1 Operazioni su Linguaggi

Definizione 4.1 (Linguaggio formale)**.** *Sia Σ un alfabeto, un linguaggio formale (in breve linguaggio, spesso indicato con L), è un* insieme *formato dalle stringhe su Σ, quindi vale che:*

$$L \subseteq \Sigma^*.$$

Notare che:

- Σ^* è esso stesso un linguaggio, il linguaggio formato da *tutte* le stringhe su Σ;
- essendo $\varnothing$ sottoinsieme di qualsiasi altro insieme, $\varnothing \subseteq \Sigma^*$ dunque $\varnothing$ è un linguaggio su qualsiasi alfabeto;
- essendo ϵ la stringa nulla su qualsiasi alfabeto Σ, $\{ \epsilon \} \subseteq \Sigma^*$ dunque $\{ \epsilon \}$ è un linguaggio su qualsiasi alfabeto.

Definizione 4.2 (Concatenazione)**.** *Siano Σ un alfabeto e $L_1, L_2 \subseteq \Sigma^*$ linguaggi, la* concatenazione *di L_1 e L_2, denotata con $L_1 \cdot L_2$, è l'insieme:*

$$L_1 \cdot L_2 := \{ x \cdot y \mid x \in L_1, y \in L_2 \}.$$

Definizione 4.3 (Concatenazione iterata)**.** *Siano Σ un alfabeto, $L \subseteq \Sigma^*$ un linguaggio e sia $n \in \mathbb{N}$, allora:*

$$L^n := \begin{cases} \{ \epsilon \}, & \textit{se } n = 0; \\ L \cdot L^{n-1}, & \textit{altrimenti.} \end{cases}$$

Lemma 4.1. *Sia I un insieme qualsiasi di indici, e sia $\{ S_i \}_{i \in I}$ una famiglia di insiemi indicizzata su I. Sia T un altro insieme. Se per ogni $i \in I$ abbiamo $S_i \subseteq T$, allora $\bigcup_{i \in I} S_i \subseteq T$.*

Dimostrazione. Dobbiamo mostrare che, preso un qualsiasi $x \in \bigcup_{i \in I} S_i$, abbiamo $x \in T$. Infatti, se $x \in \bigcup_{i \in I} S_i$, deve necessariamente esistere (almeno un) $j \in I$ tale che $x \in S_j$. Ma allora, poiché $S_j \subseteq T$, abbiamo $x \in T$. $\square$

Lemma 4.2 (Proprietà distributiva della concatenazione sull'unione)**.** *Sia I un insieme qualsiasi di indici, e sia $\{ L_i \}_{i \in I}$ una famiglia di linguaggi indicizzata su I. Sia M un altro linguaggio. Si ha che*

$$M \bigcup_{i \in I} L_i = \bigcup_{i \in I} (M L_i), \tag{10}$$

$$\left(\bigcup_{i \in I} L_i \right) M = \bigcup_{i \in I} (L_i M). \tag{11}$$

Dimostrazione. Dimostriamo la (10):

$$\begin{aligned} M \bigcup_{i \in I} L_i &= \left\{ xy \;\middle|\; x \in M, y \in \bigcup_{i \in I} L_i \right\} && \text{[def. concat.]} \\ &= \{ xy \mid x \in M, j \in I, y \in L_j \} && \text{[def. } \cup \text{]} \\ &= \bigcup_{i \in I} \{ xy \mid x \in M, y \in L_i \} && \text{[def. } \cup \text{]} \\ &= \bigcup_{i \in I} (M L_i) && \text{[def. concat.]} \end{aligned}$$

La dimostrazione della (11) è del tutto analoga. □

Lemma 4.3 (Monotonia della concatenazione)**.** *Siano L_1, L_2 e M tre linguaggi qualsiasi. Se*

$$L_1 \subseteq L_2, \tag{12}$$

allora

$$L_1 M \subseteq L_2 M, \tag{13}$$

$$M L_1 \subseteq M L_2. \tag{14}$$

Dimostrazione. Dimostriamo la (13):

$$\begin{aligned} L_1 M &= \{ xy \mid x \in L_1, y \in M \} && \text{[def. concat.]} \\ &\subseteq \{ xy \mid x \in L_2, y \in M \} && \text{[(12)]} \\ &= L_2 M && \text{[def. concat.]} \end{aligned}$$

La dimostrazione della (14) è del tutto analoga. □

Definizione 4.4 (Chiusura di Kleene)**.** *Siano Σ un alfabeto e $L \subseteq \Sigma^*$ un linguaggio, allora:*

$$L^* := \bigcup_{i \in \mathbb{N}} L^i.$$

Lemma 4.4. *(Idempotenza della chiusura di Kleene). Sia L un linguaggio, allora*

$$(L^*)^* = L^* L^* = L^*.$$

Dimostrazione. Dividiamo la dimostrazione in due parti:

1. da $(L^*)^*$ in L^*

$$\begin{aligned}
(L^*)^* &= \bigcup_{i \in \mathbb{N}} \left(\bigcup_{j \in \mathbb{N}} L^j \right)^i && [\text{def. } \cdot^*] \\
&= \bigcup_{i \in \mathbb{N}} \bigcup_{j \in \mathbb{N}} L^{ij} && [\text{def. } L^m] \\
&= \bigcup_{k \in \mathbb{N}} L^k && [\{\, ij \mid i, j \in \mathbb{N} \,\} = \mathbb{N}] \\
&= L^* && [\text{def. } \cdot^*].
\end{aligned}$$

2. da $L^* L^*$ in L^*

$$\begin{aligned}
L^* L^* &= \left(\bigcup_{i \in \mathbb{N}} L^i \right) \left(\bigcup_{j \in \mathbb{N}} L^j \right) && [\text{def. } \cdot^*] \\
&= \bigcup_{i \in \mathbb{N}} \left(L^i \bigcup_{j \in \mathbb{N}} L^j \right) && [\text{p. distrib. di } \cdot] \\
&= \bigcup_{i \in \mathbb{N}} \bigcup_{j \in \mathbb{N}} L^i L^j && [\text{p. distrib. di } \cdot] \\
&= \bigcup_{i \in \mathbb{N}} \bigcup_{j \in \mathbb{N}} L^{i+j} && [\text{def. } L^m] \\
&= \bigcup_{k \in \mathbb{N}} L^k && [\{\, i+j \mid i, j \in \mathbb{N} \,\} = \mathbb{N}] \\
&= L^* && [\text{def. } \cdot^*].
\end{aligned}$$

$\square$

Lemma 4.5 (Monotonia della chiusura di Kleene)**.** *Siano L, M due linguaggi,*

$$L \subseteq M \Longrightarrow L^* \subseteq M^*.$$

Dimostrazione. Vogliamo mostrare che, se

$$L \subseteq M, \tag{15}$$

allora $L^* \subseteq M^*$, ovvero che

$$\bigcup_{i \in \mathbb{N}} L^i \subseteq M^*. \tag{16}$$

Mostreremo che

$$\forall n \in \mathbb{N} : L^n \subseteq M^*$$

che implica la (16) in virtù del Lemma 4.1. Dimostreremo per induzione matematica semplice su n, con caso base $n = 0$ e passo induttivo $n = m + 1$ con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (parte antecedente dell'implicazione).

Caso base: se $n = 0$ abbiamo che

$$\begin{aligned}
L^n &= L^0 && [n = 0] \\
&= \{\, \epsilon \,\} && [\text{def. } \cdot^0] \\
&= M^0 && [\text{def. } \cdot^0] \\
&\subseteq \bigcup_{i \in \mathbb{N}} M^i && [0 \in \mathbb{N}] \\
&= M^* \checkmark && [\text{def. } \cdot^*].
\end{aligned}$$

Implicazione: sia $m \in \mathbb{N}$,

$$L^m \subseteq M^* \overset{?}{\Longrightarrow} L^{m+1} \subseteq M^*.$$

Passo induttivo: se $n = m + 1$ abbiamo che

$$\begin{aligned}
L^n &= L^{m+1} && [n = m+1] \\
&= LL^m && [\text{def. } \cdot^{m+1}] \\
&\subseteq LM^* && [\text{ip. ind. e monotonia di } \cdot] \\
&\subseteq MM^* && [(15) \text{ e monotonia di } \cdot] \\
&= M \bigcup_{i \in \mathbb{N}} M^i && [\text{def. } \cdot^*] \\
&= \bigcup_{i \in \mathbb{N}} \big(MM^i\big) && [\text{p. distrib. di } \cdot] \\
&= \bigcup_{i \in \mathbb{N}} M^{i+1} && [\text{def. } \cdot^{i+1}] \\
&= \bigcup_{i \geq 1} M^i && [\text{def. } \cdot^{i+1}] \\
&\subseteq \bigcup_{i \in \mathbb{N}} M^i && [\text{def. } \cup] \\
&= M^* \checkmark && [\text{def. } \cdot^*].
\end{aligned}$$

□

Lemma 4.6 (Estensività della chiusura di Kleene)**.** *Sia L un linguaggio,*

$$L \subseteq L^*.$$

Dimostrazione.

$$\begin{aligned}
L^* &= \bigcup_{n \in \mathbb{N}} L^i && [\text{def. } \cdot^*] \\
&= L^0 \cup L \cup \bigcup_{i \geq 2} L^i && [0, 1 \in \mathbb{N}] \\
&\Longrightarrow L \subseteq L^*.
\end{aligned}$$

□

Lemma 4.7 (La chiusura di Kleene è un UCO)**.** *Per la chiusura di Kleene valgono le seguenti condizioni:*

1. *monotonia;*

2. *idempotenza;*

3. *estensività.*

Quindi la chiusura di Kleene è un UCO.

4.2 Espressioni Regolari

Le *espressioni regolari*, che ora vedremo, tra i vari approcci di rappresentazione finita di linguaggi potenzialmente infiniti, ricadono in quello di tipo algebrico.

4.2.1 Definizioni, Teoremi e Lemmi

Definizione 4.5 (Definizione informale della sintassi delle espressioni regolari)**.** *Sia Σ un alfabeto:*

1. *Ogni simbolo $a \in \Sigma$ è una espressione regolare su Σ.*

2. *$\underline{\epsilon}$ è una espressione regolare su Σ.*

3. *$\underline{\varnothing}$ è una espressione regolare su Σ.*

4. *Se r e s sono espressioni regolari su Σ, $r \,\underline{\cdot}\, s$ è una espressione regolare su Σ.*

5. *Se r e s sono espressioni regolari su Σ, $r + s$ è una espressione regolare su Σ.*

6. *Se r è una espressione regolare su Σ, r^* è una espressione regolare su Σ.*

Viene spesso considerato pedante aggiungere la seguente clausola di chiusura.

7. *Niente è un'espressione regolare su Σ se non in virtù dei punti 1–6 di cui sopra.*

Definizione 4.6 (Definizione formale della sintassi di espressioni regolari, per mezzo di grammatica in forma BNF)**.** *Sia $\Sigma = \{ a_1, \ldots, a_n \}$ un alfabeto. Un'espressione regolare su Σ è una qualunque stringa generata dalla seguente grammatica:*

$$E ::= a_1 \mid \cdots \mid a_n \mid \underline{\epsilon} \mid \underline{\varnothing} \mid E_1 + E_2 \mid E_1 \,\underline{\cdot}\, E_2 \mid E^*$$

Definizione 4.7 (Definizione della semantica delle espressioni regolari, attraverso la definizione di "fa match")**.** *Sia* Σ *un alfabeto e sia* r *un'espressione regolare su* Σ*. Sia* $x \in \Sigma^*$*. Si dice che* x fa match con r, *scritto* $x \lessdot r$ *in ottemperanza alle seguenti regole:*

1. $a \lessdot \underline{a}$*, se* $a \in \Sigma$*;*

2. $\epsilon \lessdot \underline{\epsilon}$*;*

3. $x \not\lessdot \underline{\varnothing}$*, per ogni* $x \in \Sigma^*$*;*

4. $x \lessdot r_1 \underline{+} r_2$*, se* $x \lessdot r_1$ *o* $x \lessdot r_2$*;*

5. $x \lessdot r_1 \,\underline{\cdot}\, r_2$*, se esistono* $y, z \in \Sigma^*$ *tali che* $x = y \cdot z$*,* $y \lessdot r_1$*,* $z \lessdot r_2$*;*

6. $x \lessdot r^{\underline{*}}$*, se* $x \lessdot \underline{\epsilon}$ *oppure esistono* $y, z \in \Sigma^*$ *tali che* $x = y \cdot z$*,* $y \lessdot r$*,* $z \lessdot r^{\underline{*}}$*.*

Definizione 4.8 (Linguaggio denotato da un'espressione regolare)**.** *Siano* Σ *un alfabeto ed* r *una espressione regolare su* Σ*, allora il linguaggio denotato da* r *è definito come:*

$$L(r) = [\![r]\!] := \{\, x \in \Sigma^* \mid x \lessdot r \,\}.$$

Definizione 4.9 (Linguaggio regolare)**.** L *è un* linguaggio regolare *se esiste almeno una espressione regolare* r *che lo denota, ovvero*

$$L = L(r).$$

Creiamo una tabella che riassume quanto detto fino ad ora riguardo a sintassi e semantica (il significato da associato alla sintassi) delle espressioni regolari:

Simbolo metalinguaggio ER	**Sintassi**	**Semantica**
$\underline{a}$	$\underline{a}$, con $a \in \Sigma$	$[\![\underline{a}]\!] = L(\underline{a}) = \{\, a \,\}$
$\underline{\varnothing}$	$\underline{\varnothing}$	$[\![\underline{\varnothing}]\!] = L(\underline{\varnothing}) = \varnothing$
$\underline{\epsilon}$	$\underline{\epsilon}$	$[\![\underline{\epsilon}]\!] = L(\underline{\epsilon}) = \{\, \epsilon \,\}$
$\underline{\cdot}$	$r \,\underline{\cdot}\, s$	$[\![r \,\underline{\cdot}\, s]\!] = L(r) \cdot L(s)$
$\underline{+}$	$r \underline{+} s$	$[\![r \underline{+} s]\!] = L(r) \cup L(s)$
$\underline{*}$	$r^{\underline{*}}$	$[\![r^{\underline{*}}]\!] = \bigcup_{i \in \mathbb{N}} L(r)^i$
$\underline{()}$	$\underline{(}r\underline{)}$	$[\![\underline{(}r\underline{)}]\!] = L(\underline{(}r\underline{)}) = L(r)$
$\underline{+}$	$r^{\underline{+}}$	$[\![r^{\underline{+}}]\!] = L(r) \cdot L(r)^*$

Tra espressioni regolari valgono alcune relazioni che ne permettono la manipolazione algebrica. Siano r, s espressioni regolari allora

$$\begin{aligned} r = s &\iff L(r) = L(s), \\ r \preceq s &\iff L(r) \subseteq L(s). \end{aligned}$$

Lemma 4.8 (Identità del $+$). *Sia r una generica espressione regolare allora*

$$r + \varnothing = r.$$

Dimostrazione.

$$\begin{aligned} L(r + \varnothing) &= L(r) \cup L(\varnothing) && \text{[def. } +\text{]} \\ &= L(r) && \text{[def. } \cup\text{].} \end{aligned}$$

□

Lemma 4.9 (Proprietà commutativa del $+$). *Siano r, s espressioni regolari allora*

$$r + s = s + r.$$

Dimostrazione.

$$\begin{aligned} L(r + s) &= L(r) \cup L(s) && \text{[def. } +\text{]} \\ &= L(s) \cup L(r) && \text{[p. comm. di } \cup\text{]} \\ &= L(s + r) && \text{[def. } +\text{].} \end{aligned}$$

□

Lemma 4.10 (Proprietà associativa del $+$). *Siano r, s, t espressioni regolari, allora*

$$(r + s) + t = r + (s + t).$$

Dimostrazione.

$$\begin{aligned} L\big((r + s) + t\big) &= \big(L(r) \cup L(s)\big) \cup L(t) && \text{[def. } +\text{]} \\ &= L(r) \cup \big(L(s) \cup L(t)\big) && \text{[p. assoc. di } \cup\text{]} \\ &= L\big(r + (s + t)\big) && \text{[def. } +\text{].} \end{aligned}$$

□

Lemma 4.11 (Identità della concatenazione)**.** *Sia r una generica espressione regolare allora*

$$\epsilon r = r = r\epsilon$$

Dimostrazione. Dimostriamo la prima uguaglianza

$$\begin{aligned}
L(\epsilon r) &= L(\epsilon)L(r) && \text{[def. concat.]}\\
&= \{\, xy \mid x \in L(\epsilon), y \in L(r) \,\} && \text{[def. concat.]}\\
&= \{\, xy \mid x \in \{\, \epsilon \,\}, y \in L(r) \,\} && \text{[def. concat.]}\\
&= \{\, \epsilon y \mid y \in L(r) \,\} && [x = \epsilon]\\
&= \{\, y \mid y \in L(r) \,\} && [\epsilon \text{ è identità}]\\
&= L(r) && \text{[def. concat.]}.
\end{aligned}$$

La dimostrazione di della seconda uguaglianza è analoga. □

Lemma 4.12 (Elemento assorbente della concatenazione)**.** *Sia r una espressione regolare, allora*

$$r\varnothing = \varnothing.$$

Dimostrazione.

$$\begin{aligned}
L(r\varnothing) &= L(r)L(\varnothing) && \text{[def. conat.]}\\
&= \{\, xy \mid x \in L(r), y \in L(\varnothing) \,\} && \text{[def. concat.]}\\
&= \{\, xy \mid x \in L(r), y \in \varnothing \,\} && [\text{def. } \varnothing]\\
&= \varnothing && [\nexists y \in \varnothing].
\end{aligned}$$

□

Lemma 4.13 (Proprietà associativa della concatenazione)**.** *Siano r, s, t espressioni regolari, allora*

$$(rs)t = r(st).$$

Dimostrazione.

$$\begin{aligned}
L((rs)t) &= L(rs)L(t) && \text{[def. concat.]}\\
&= \big(L(r)L(s)\big)L(t) && \text{[def. concat.]}\\
&= L(r)\big(L(s)L(t)\big) && \text{[def. concat.]}\\
&= L(r)L(st) && \text{[def. concat.]}\\
&= L\big(r(st)\big) && \text{[def. concat.]}
\end{aligned}$$

$\square$

Lemma 4.14 (Proprietà distributiva della concatenazione sul $+$)**.** *Siano r, s, t espressioni regolari, allora*

$$r(s+t) = rs + rt$$

Dimostrazione.

$$\begin{aligned}
L\big(r(s+t)\big) &= L(r)\big(L(s) \cup L(t)\big) && \text{[def. +, def. concat.]} \\
&= \big(L(r)L(s)\big) \cup \big(L(r)L(t)\big) && \text{[p. distrib. concat.]} \\
&= L(rs) \cup L(rt) && \text{[def. concat.]} \\
&= L(rs + rt) && \text{[def. +].}
\end{aligned}$$

$\square$

Teorema 4.1 (Equivalenza tra espressioni regolari e DFA)**.** *Le espressioni regolari ed i DFA sono due formalismi equivalenti:*

1. *Sia r un'espressione regolare, allora esiste un DFA M tale che*

$$L(M) = L(r).$$

2. *Sia M un DFA, allora esiste un'espressione regolare r, tale che*

$$L(r) = L(M).$$

4.2.2 Costruzione di Ken Thompson

Ken Thompson è l'autore dello stumento grep e autore del sistema operativo UNIX insieme a Dennis Ritchie (creatore del C). Thompson definì anche un algoritmo per derivare un ϵ-NFA da una qualunque espressione regolare dividendola nelle sue sottoespressioni elementari, che possono essere convertite direttamente per mezzo di un insieme di regole.

4.3 Esercizi

Esercizio 4.1

Trovare le espressioni regolari che denotano i seguenti linguaggi:

1. $\{ x \in \{ 0, 1 \}^* \mid |x| = 1 \}$;
2. $\{ x \in \{ 0, 1 \}^* \mid x$ termina con '0' $\}$;
3. $\{ x \in \{ 0, 1 \}^* \mid x$ contiene un numero pari di '1' $\}$;
4. $\{ x \in \{ 0, 1 \}^* \mid x$ contiene un numero dispari di '0' $\}$;
5. $\{ x \in \{ a, b \}^* \mid |x|$ è divisibile per 3 $\}$;
6. $\{ x \in \{ a, b, c \}^* \mid x$ non contiene a o b o c $\}$;
7. $\{ x \in \{ a, b \}^* \mid$ la sottostringa ab occorre esattamente due volte ma non alla fine di $x \}$.

Soluzione.

1. $(0 + 1)$;
2. $(0 + 1)^*0$;
3. $0^*(10^*10^*)^*$;
4. $1^*(01^*01^*)^*01^*$;
5. $\big((a + b)(a + b)(a + b)\big)^*$;
6. $(a + b)^* + (a + c)^* + (b + c)^*$;
7. $b^*a^+b^+a^+b(b^+a^* + b^*a^+)$.

Esercizio 4.2

Descrivere a parole le stringhe che fanno match con l'espressione regolare $(aa^*b)^*a^*$.

Soluzione. Sono le stringhe contenenti un numero arbitrario di a e di b e ogni occorrenza di b è direttamente preceduta da almeno una occorrenza di a.

Esercizio 4.3

Sia $\Sigma = \{ a, b, c \}$, trovare un'espressione regolare r su Σ tale che $L(r)$ è formato da tutte le stringhe che contengono almeno un'occorrenza di ogni simbolo in Σ. Spiegare brevemente la risposta.

Soluzione. "Forziamo l'utente" a scrivere almeno un'occorrenza di ciascun simbolo dell'alfabeto, la nostra espressione regolare r quindi assomiglierà a qualcosa del tipo: $(abc)^+$. Stiamo però forzando anche l'ordine in cui compaiono a, b, c; risolviamo esplicitiando tutte le possibili permutazioni di abc per poi metterle in *or* tra loro: $\big((abc) + (acb) + (bac) + (bca) + (cab) + (cba)\big)^+$. A questo punto aggiungiamo la possibilità di inserire qualsiasi cosa tra un simbolo e l'altro, ovvero $(a + b + c)^*$, per alleggerire la notazione definiamo

$$\_^* = (a + b + c)^*.$$

L'espressione regolare finale è quindi
$\big((\_^*a\_^*b\_^*c\_^*)$
$+ (\_^*a\_^*c\_^*b\_^*)$
$+ (\_^*b\_^*a\_^*c\_^*)$
$+ (\_^*b\_^*c\_^*a\_^*)$
$+ (\_^*c\_^*a\_^*b\_^*)$
$+ (\_^*c\_^*b\_^*a\_^*)\big)^+$

Esercizio 4.4

Descrivere il modo in cui le espressioni regolari sono costruite.

Soluzione. Le espressioni regolari sono costruite in maniera induttiva, applicando un numero di volte arbitrario ma finito le regole descritte precedentemente.

Esercizio 4.5

Se una espressione regolare r non contiene nessuna occorrenza del simbolo $\varnothing$, è possibile che $L(r) = \varnothing$?

Soluzione. No, non è possibile creare un'espressione regolare r che denota il linguaggio $L(r) = \varnothing$ con $\varnothing \notin r$; tra le regole delle espressioni regolari definite in precedenza infatti, l'unico modo per denotare l'insieme vuoto è proprio tramite il simbolo del metalinguaggio $\varnothing$.

Esercizio 4.6

Se una espressione regolare r non contiene nessuna occorrenza del simbolo ϵ, è possibile che $L(r)$ contenga la stringa ϵ?

Soluzione. Si, preso un Σ qualsiasi ad una espressione regolare r^* su Σ abbiamo che:

$$\begin{aligned}
L(r)^* &= \bigcup_{i \in \mathbb{N}} L(r)^i && [\text{def. } \cdot^*] \\
&= L(r)^0 \cup \bigcup_{i \geq 1} L(r)^i && [0 \in \mathbb{N}] \\
&= \{\, \epsilon \,\} \cup \bigcup_{i \geq 1} L(r)^i && [\text{def. } \cdot^0] \\
&\implies \{\, \epsilon \,\} \in L(r)^*.
\end{aligned}$$

Esercizio 4.7

Siano r, s, t espressioni regolari, supponendo che $s \preceq t$ e $rt \preceq t$, dimostrare per induzione che $r^n s \preceq t$ è vera per tutte le $n \geq 0$; si deduca che $r^* s \preceq t$.

Soluzione. Vogliamo mostrare che se

$$L(s) \subseteq L(t), \tag{17}$$

$$L(rt) \subseteq L(t), \tag{18}$$

allora $L(r^n s) \subseteq L(t)$. Mostreremo che

$$\forall n \in \mathbb{N} : L(r^n s) \subseteq L(t),$$

per induzione matematica semplice su n, con caso base $n = 0$ e passo induttivo $n = m + 1$ con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (parte antecedente dell'implicazione).

Caso base: se $n = 0$ abbiamo che

$$\begin{aligned}
L(r^n s) &= L(r^0 s) && [\text{n} = 0] \\
&= L(r)^0 L(s) && [\text{def. concat.}] \\
&= L(s) && [\text{def. } L^n] \\
&\subseteq L(t) \checkmark && [(17)].
\end{aligned}$$

Implicazione: sia $m \in \mathbb{N}$,

$$L(r^m s) \subseteq L(t) \overset{?}{\implies} L(r^{m+1} s) \subseteq L(t).$$

Passo induttivo: se $n = m + 1$ abbiamo che

$$\begin{aligned}
L(r^n s) &= L(r^{m+1} s) && \text{[n = m +1]} \\
&= L(r r^m s) && [\text{def. } \cdot^{m+1}] \\
&= L(r) L(r^m s) && \text{[def. concat.]} \\
&\subseteq L(r) L(t) && \text{[ip. ind., monotonia di concat.]} \\
&= L(rt) && \text{[def. concat.]} \\
&\subseteq L(t) \checkmark && \text{[(18)].}
\end{aligned}$$

Quindi abbimo dimostrato che

$$\forall n \in \mathbb{N} : L(r^n s) \subseteq L(t),$$

che virtù del Lemma 4.1, implica

$$\bigcup_{i \in \mathbb{N}} L(r^i s) = L(r^* s) \subseteq L(t).$$

Esercizio 4.8

Quali sono le differenze, se ci sono, tra i linguaggi denotati dalle seguenti espressioni regolari:

$$\varnothing, (\varnothing^*)^*, \varnothing(\varnothing)^*$$

Soluzione.

1. $L(\varnothing) = \varnothing$.
2. $L\big((\varnothing^*)^*\big) = \{\, \epsilon \,\}$.

Dimostrazione.

$$
\begin{aligned}
L\big((\varnothing^*)^*\big) &= L(\varnothing^*) && [\text{idempotenza } \cdot^*]] \\
&= L(\varnothing)^* && [\text{def. } \cdot^*] \\
&= \bigcup_{i\in\mathbb{N}} L(\varnothing)^i && [\text{def. } \cdot^*] \\
&= L(\varnothing)^0 \cup \bigcup_{i\geq 1} L(\varnothing)^i && [0 \in \mathbb{N}] \\
&= \{\, \epsilon \,\} \cup \bigcup_{i\geq 1} L(\varnothing)^i && [\text{def. } \cdot^0] \\
&= \{\, \epsilon \,\} \cup \bigcup_{i\geq 1} \{\, xy \mid x \in L(\varnothing), y \in L(\varnothing)^{i-1} \,\} && [\text{def. } \cdot^n] \\
&= \{\, \epsilon \,\} \cup \bigcup_{i\geq 1} \{\, xy \mid x \in \varnothing, y \in L(\varnothing)^{i-1} \,\} && [\text{def. } \varnothing] \\
&= \{\, \epsilon \,\} \cup \varnothing && [\nexists x \in \varnothing] \\
&= \{\, \epsilon \,\} && [\text{def. } \cup].
\end{aligned}
$$

□

3. $L\big(\varnothing\varnothing^*\big) = \varnothing$.

   Dimostrazione.

$$
\begin{aligned}
L(\varnothing\varnothing^*) &= L(\varnothing)L(\varnothing^*) && [\text{def. concat.}] \\
&= L(\varnothing)L(\varnothing)^* && [\text{def. } \cdot^*] \\
&= \{\, xy \mid x \in L(\varnothing), y \in L(\varnothing)^* \,\} && [\text{def. concat.}] \\
&= \{\, xy \mid x \in \varnothing, y \in L(\varnothing)^i \,\} && [\text{def. } \varnothing] \\
&= \varnothing && [\nexists x \in \varnothing]
\end{aligned}
$$

□

Esercizio 4.9

Dimostrare la validità della seguente espressione regolare o uno specifico esempio per il quale la relazione non è valida:

$$r^*r + \epsilon = r^*.$$

Soluzione.

$$
\begin{aligned}
L(r^*r+\epsilon) &= L(r^*r)\cup L(\epsilon) && \text{[def. $+$]}\\
&= \big(L(r^*)L(r)\big)\cup L(\epsilon) && \text{[def. $+$]}\\
&= \left(\bigcup_{i\in\mathbb{N}}\big(L(r)^i\big)L(r)\right)\cup L(\epsilon) && \text{[def. $\cdot^*$]}\\
&= \left(\bigcup_{i\in\mathbb{N}}L(r)^iL(r)\right)\cup L(\epsilon) && \text{[p. distrib. concat.]}\\
&= \bigcup_{i\in\mathbb{N}}L(r)^{i+1}\cup L(\epsilon) && \text{[def. L^n]}\\
&= \bigcup_{i\geq 1}L(r)^{i}\cup L(\epsilon) && \text{[def. $\cdot^{i+1}$]}\\
&= \bigcup_{i\geq 1}L(r)^{i}\cup \{\,\epsilon\,\} && \text{[def. ϵ]}\\
&= \bigcup_{i\geq 1}L(r)^{i}\cup L(r)^0 && \text{[def. L^0]}\\
&= \bigcup_{i\in\mathbb{N}}L(r)^{i} && \text{[def. $\cup$]}\\
&= L(r)^* && \text{[def di $\cdot^*$]}\\
&= L(r^*) && \text{[def di $\cdot^*$].}
\end{aligned}
$$

Esercizio 4.10

Dimostrare la validità della seguente espressione regolare o uno specifico esempio per il quale la relazione non è valida:

$$(\epsilon+r)^*=r^*.$$

Soluzione.

$$
\begin{aligned}
L\big((\epsilon + r)^*\big) &= \bigcup_{i\in\mathbb{N}} L(\epsilon + r)^i && [\text{def. } \cdot^*] \\
&= \bigcup_{i\in\mathbb{N}} \big(L(\epsilon) \cup L(r)\big)^i && [\text{def. } +] \\
&= \bigcup_{i\in\mathbb{N}} \big(\{\, \epsilon \,\} \cup L(r)\big)^i && [\text{def. } \epsilon] \\
&= \bigcup_{i\in\mathbb{N}} \big(L(r)^0 \cup L(r)\big)^i && [\text{def. } \cdot^0] \\
&= \bigcup_{i\in\mathbb{N}} \big(L(r)^{0i} \cup L(r)^i\big) && [\text{p. distrib. concat.}] \\
&= \bigcup_{i\in\mathbb{N}} L(r)^i \cup L(r)^0 && [\forall i \in \mathbb{N} : L^{0i} = L^0] \\
&= \bigcup_{i\in\mathbb{N}} L(r)^i && [0 \in \mathbb{N}] \\
&= L(r)^* && [\text{def. } \cdot^*]. \\
&= L(r^*) && [\text{def. } \cdot^*].
\end{aligned}
$$

Esercizio 4.11

Dimostrare la validità della seguente espressione regolare o uno specifico esempio per il quale la relazione non è valida:

$$(r + s)^* \preceq r^* + s^*.$$

Soluzione. Siano $\Sigma = \{\, 0, 1 \,\}$ un alfabeto ed $r = 0, s = 1$ espressioni regolari. Sia $x = 01$, $x \notin L(r^* + s^*) = L(0^* + 1^*), x \in L(r + s)^* = L(0 + 1)^*$.

Esercizio 4.12

Dimostrare la validità della seguente espressione regolare o uno specifico esempio per il quale la relazione non è valida:

$$r^* + s^* \preceq (r + s)^*.$$

Soluzione. Ragioniamo sulle sottoespressioni:

- r^*:

$$\begin{aligned} L(r^*) &= \bigcup_{i\in\mathbb{N}} L(r)^i && [\text{def. } \cdot^*] \\ &\subseteq \bigcup_{i\in\mathbb{N}} \big(L(r) \cup L(s)\big)^i && [\text{def. } \cup] \\ &= \bigcup_{i\in\mathbb{N}} L(r+s)^i && [\text{def. } +] \\ &= L(r+s)^* && [\text{def. } \cdot^*] \\ &= L\big((r+s)^*\big) && [\text{def. } \cdot^*]. \end{aligned}$$

- s^*: $L(s^*) \subseteq L\big((r+s)^*\big)$ (procedimento analogo).

Dunque sappiamo che:

$$L(r^*), L(s^*) \subseteq L\big((r+s)^*\big). \tag{19}$$

Se, come dice la (19), i linguaggi denotati dalle due sottoespressioni sono sottoinsiemi di $L\big((r+s)^*\big)$, allora anche la loro unione lo è, ovvero:

$$\begin{aligned} L(r^* + s^*) &= L(r^*) \cup L(s^*) && [\text{def. } +] \\ &\subseteq L\big((r+s)^*\big) \cup L\big((r+s)^*\big) && [\text{(19) e monotonia di } \cup] \\ &\subseteq L\big((r+s)^*\big) && [\text{def. } \cup]. \end{aligned}$$

Esercizio 4.13

Dimostrare la validità della seguente espressione regolare o uno specifico esempio per il quale la relazione non è valida:

$$(r^* s^*)^* \preceq (r+s)^*.$$

Soluzione.

$$\begin{aligned}
L(r^*s^*) &= L(r^*)L(s^*) && \text{[def. concat.]}\\
&= \{\, xy \mid x \in L(r^*), y \in L(s^*) \,\} && \text{[def. concat.]}\\
&\subseteq \{\, xy \mid x \in L\big((r+s)^*\big), y \in L\big((r+s)^*\big) \,\} && \text{[(19), monotonia } \cdot^*]\\
&= L\big((r+s)^*\big)L\big((r+s)^*\big) && \text{[def. concat.]}\\
&= L(r+s)^*L(r+s)^* && [\text{def. } \cdot^*]\\
&= L(r+s)^* && [\text{idempotenza di } \cdot^*]\\
&= L\big((r+s)^*\big) && [\text{def. } \cdot^*]
\end{aligned}$$

Quindi, sapendo che vale:

$$L(r^*s^*) \subseteq L\big((r+s)^*\big), \tag{20}$$

vogliamo mostrare che $L\big((r^*s^*)^*\big) \subseteq L\big((r+s)^*\big)$, ovvero

$$\bigcup_{i\in\mathbb{N}} L(r^*s^*)^i \subseteq L\big((r+s)^*\big). \tag{21}$$

Mostreremo che

$$\forall n \in \mathbb{N} : L\big((r^*s^*)^n\big) \subseteq L\big((r+s)^*\big)$$

che implica la (21) in virtù del Lemma 4.1. Dimostreremo per induzione matematica semplice su n, con caso base $n = 0$ e passo induttivo $n = m + 1$ con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (parte antecedente dell'implicazione).

Caso base: se $n = 0$ abbiamo che

$$\begin{aligned}
L\big((r^*s^*)^n\big) &= L\big((r^*s^*)^0\big) && [n = 0]\\
&= L(r^*s^*)^0 && [\text{def. } \cdot^0]\\
&= \{\, \epsilon \,\} && [\text{def. } \cdot^0]\\
&= L(r+s)^0 && [\text{def. } \cdot^0]\\
&\subseteq \bigcup_{i\in\mathbb{N}} L(r+s)^i && [0 \in \mathbb{N}]\\
&= L(r+s)^* && [\text{def. } \cdot^*]\\
&= L\big((r+s)^*\big) \checkmark && [\text{def. } \cdot^*].
\end{aligned}$$

Implicazione: sia $m \in \mathbb{N}$,

$$L\big((r^*s^*)^m\big) \subseteq L\big((r+s)^*\big) \overset{?}{\Longrightarrow} L\big((r^*s^*)^{m+1}\big) \subseteq L\big((r+s)^*\big).$$

Passo induttivo: se $n = m+1$ abbiamo che

$$\begin{aligned}
L\big((r^*s^*)^n\big) &= L\big((r^*s^*)^{m+1}\big) && \text{[n=m+1]}\\
&= L(r^*s^*)L(r^*s^*)^m && [\text{def. } \cdot^{m+1}]\\
&\subseteq L(r^*s^*)L\big((r+s)^*\big) && [\text{ip. ind., monotonia } \cdot]\\
&\subseteq L\big((r+s)^*\big)L\big((r+s)^*\big) && [(20)\text{, monotonia } \cdot]\\
&= L(r+s)^*L(r+s)^* && [\text{def. } \cdot^*]\\
&= L(r+s)^* && [\text{idempotenza } \cdot^*]\\
&= L\big((r+s)^*\big)\checkmark && [\text{def. } \cdot^*].
\end{aligned}$$

Esercizio 4.14

Dimostrare la validità della seguente espressione regolare o uno specifico esempio per il quale la relazione non è valida:

$$(r^*s^*)^* = (r+s)^*.$$

Soluzione. Notare che nel punto precedente abbiamo dimostrato

$$(r^*s^*)^* \preceq (r+s)^*,$$

dimostreremo il "contenimento" anche in direzione opposta, ovvero

$$(r+s)^* \preceq (r^*s^*)^*,$$

da cui discende

$$(r^*s^*)^* = (r+s)^*.$$

Ragioniamo sulle sottoespressioni:

- r:

$$\begin{aligned}
L(r) &\subseteq \bigcup_{i\in\mathbb{N}} L(r)^i && [1 \in \mathbb{N}] \\
&= L(r)^* && [\text{def. } \cdot^*] \\
&= L\big((r^*)^*\big) && [\text{idempotenza } \cdot^*] \\
&= \bigcup_{h\in\mathbb{N}} \Big(\bigcup_{i\in\mathbb{N}} L(r)^i\Big)^h && [\text{def. } \cdot^*] \\
&= \bigcup_{h\in\mathbb{N}} \Big(\Big(\bigcup_{i\in\mathbb{N}} L(r)^i\Big)\{\,\epsilon\,\}\Big)^h && [\text{def. } \epsilon] \\
&\subseteq \bigcup_{h\in\mathbb{N}} \Big(\Big(\bigcup_{i\in\mathbb{N}} L(r)^i\Big)\Big(\bigcup_{j\in\mathbb{N}} L(s)^j\Big)\Big)^h && [\forall s : \epsilon \in \bigcup_{j\in\mathbb{N}} L(s)^j] \\
&= L(r^*s^*)^* && [\text{def. } \cdot^*] \\
&= L\big((r^*s^*)^*\big) && [\text{def. } \cdot^*].
\end{aligned}$$

- $s : L(s) \subseteq L\big((r^*s^*)^*\big)$ (procedimento analogo).

Dunque sappiamo che:

$$L(r), L(s) \subseteq L\big((r^*s^*)^*\big). \tag{22}$$

Se, come dice la (22), i linguaggi denotati dalle due sottoespressioni sono sottoinsiemi di $L\big((r^*s^*)^*\big)$, allora anche la loro unione lo è, ovvero:

$$\begin{aligned}
L(r+s) &= L(r) \cup L(s) && [\text{def. } +] \\
&\subseteq L\big((r^*s^*)^*\big) \cup L\big((r^*s^*)^*\big) && [(22)] \\
&= L\big((r^*s^*)^*\big) && [\text{def. } \cup].
\end{aligned}$$

Quindi

$$L(r+s) \subseteq L\big((r^*s^*)^*\big). \tag{23}$$

A questo punto vogliamo mostrare che $L\big((r+s)^*\big) \subseteq L\big((r^*s^*)^*\big)$, ovvero

$$\bigcup_{i\in\mathbb{N}} L(r+s)^i \subseteq L\big((r^*s^*)^*\big). \tag{24}$$

Mostreremo che

$$\forall n \in \mathbb{N}, L(r+s)^n \subseteq L\big((r^*s^*)^*\big),$$

che implica la (24) in virtù del Lemma 4.1. Dimostreremo per induzione matematica semplice su n, con caso base $n = 0$ e passo induttivo $n = m + 1$ con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (parte antecedente dell'implicazione).

Caso base: se $n = 0$ abbiamo che

$$\begin{aligned}
L(r+s)^n &= L(r+s)^0 && [n = 0] \\
&= \{\,\epsilon\,\} && [\text{def. } \cdot^0] \\
&\subseteq L(r^*s^*)^0 && [\text{def. } \cdot^0] \\
&\subseteq \bigcup_{i \in \mathbb{N}} L(r^*s^*)^i && [0 \in \mathbb{N}] \\
&= L(r^*s^*)^* && [\text{def. } \cdot^*] \\
&= L\big((r^*s^*)^*\big) \checkmark && [\text{def. } \cdot^*].
\end{aligned}$$

Implicazione: sia $m \in \mathbb{N}$,

$$L(r+s)^m \subseteq L\big((r^*s^*)^*\big) \overset{?}{\Longrightarrow} L(r+s)^{m+1} \subseteq L\big((r^*s^*)^*\big).$$

Passo induttivo: se $n = m + 1$ abbiamo che

$$\begin{aligned}
L(r+s)^n &= L(r+s)^{m+1} && [n = m + 1] \\
&= L(r+s)L(r+s)^m && [\text{def. } \cdot^{m+1}] \\
&\subseteq L(r+s)L\big((r^*s^*)^*\big) && [\text{ip. ind., monotonia concat.}] \\
&\subseteq L\big((r^*s^*)^*\big)L\big((r^*s^*)^*\big) && [(23), \text{ monotonia concat.}] \\
&= L(r^*s^*)^*L(r^*s^*)^* && [\text{def. } \cdot^*] \\
&= L(r^*s^*)^* && [\text{idempotenza } \cdot^*] \\
&= L\big((r^*s^*)^*\big) \checkmark && [\text{def. } \cdot^*]
\end{aligned}$$

Esercizio 4.15

Dimostrare la validità della seguente espressione regolare o uno specifico esempio per il quale la relazione non è valida:

$$(rs + r)^*r \preceq r(sr + r)^*.$$

Soluzione. Ragioniamo sulle sottoespressioni: vediamo se levando la stella di Kleene la relazione vale

$$\begin{aligned}
L((rs+r)r) &= \big(L(rs) \cup L(r)\big)L(r) && \text{[def. +, def. concat.]}\\
&= L(rs)L(r) \cup L(r)L(r) && \text{[p. distrib. } \cup\text{]}\\
&= L(rsr) \cup L(rr) && \text{[def. concat.]}\\
&= \big(L(r)L(sr)\big) \cup \big(L(r)L(r)\big) && \text{[def. concat.]}\\
&= L(r)\big(L(sr) \cup L(r)\big) && \text{[p. distrib. di } \cup\text{]}\\
&= L\big(r(sr+r)\big) && \text{[def. concat., def. +]}\\
&\subseteq \bigcup_{i\in\mathbb{N}} L\big(r(sr+r)^i\big) && [1 \in \mathbb{N}]\\
&= L\big(r(sr+r)^*\big) && \text{[def. } \cdot^*\text{].}
\end{aligned}$$

Quindi, sapendo che vale:

$$L\big((rs+r)r\big) \subseteq L\big(r(sr+r)^*\big) \tag{25}$$

vogliamo reintrodurre la stella di Kleene e mostrare che

$$L\big((rs+r)^*r\big) \subseteq L\big(r(sr+r)^*\big),$$

ovvero

$$\bigcup_{i\in\mathbb{N}} L\big((rs+r)^i r\big) \subseteq L\big(r(sr+r)^*\big). \tag{26}$$

Mostreremo che

$$\forall n \in \mathbb{N} : L\big((rs+r)^n r\big) \subseteq L\big(r(sr+r)^*\big)$$

che implica la (26) in virtù del Lemma 4.1. Dimostreremo per induzione matematica semplice su n, con caso base $n = 0$ e passo induttivo $n = m + 1$ con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva (parte antecedente dell'implicazione).

Caso base: se $n = 0$ abbiamo che

$$\begin{aligned}
L\big((rs+r)^n r\big) &= L\big((rs+r)^0 r\big) && [n = 0]\\
&= L(r) && \text{[def. } \cdot^0\text{]}\\
&= L\big(r(sr+r)^0\big) && \text{[def. } \cdot^0\text{]}\\
&\subseteq \bigcup_{i\in\mathbb{N}} L\big(r(sr+r)^i\big) && [0 \in \mathbb{N}]\\
&= L\big(r(sr+r)^*\big) \checkmark && \text{[def. } \cdot^*\text{].}
\end{aligned}$$

Implicazione: sia $m \in \mathbb{N}$,

$$L\big((rs+r)^m r\big) \subseteq L\big(r(sr+r)^*\big) \overset{?}{\Longrightarrow} L\big((rs+r)^{m+1} r\big) \subseteq L\big(r(sr+r)^*\big).$$

Passo induttivo: se $n = m+1$ abbiamo che

$$\begin{aligned}
L\big((rs+r)^n r\big) &= L\big((rs+r)^{m+1} r\big) && \text{[n=m+1]} \\
&= L\big((rs+r)(rs+r)^m r\big) && [\text{def. } \cdot^{m+1}] \\
&\subseteq L\big((rs+r)r(sr+r)^*\big) && [\text{ip. ind., monotonia } \cdot^*] \\
&\subseteq L\big(r(sr+r)^*(sr+r)^*\big) && [(25), \text{ monotonia di } \cdot^*] \\
&\subseteq L\big(r(sr+r)^*\big) \checkmark && [\text{idempotenza } \cdot^*].
\end{aligned}$$

Esercizio 4.16

Dimostrare la validità della seguente espressione regolare o uno specifico esempio per il quale la relazione non è valida:

$$(rs+r)^* r = r(sr+r)^*.$$

Soluzione.

$$\begin{aligned}
L\big((rs+r)^* r)\big) &= \big(L(rs) \cup L(r)\big)^* L(r) && [\text{def. concat., def. } +] \\
&= \left(\bigcup_{i\in\mathbb{N}} \big(L(rs) \cup L(r)\big)^i\right) L(r) && [\text{def. } \cdot^*] \\
&= \bigcup_{i\in\mathbb{N}} \Big(\big(L(rs)L(r)\big) \cup \big(L(r)L(r)\big)\Big)^i && [\text{p. distrib. concat.}] \\
&= \bigcup_{i\in\mathbb{N}} \big(L(rsr) \cup L(rr)\big)^i && [\text{p. distrib. concat.}] \\
&= L(r) \bigcup_{i\in\mathbb{N}} \big(L(sr) \cup L(r)\big)^i && [\text{p. distrib. concat.}] \\
&= L(r) \bigcup_{i\in\mathbb{N}} L(sr+r)^i && [\text{def. } +] \\
&= L(r) L\big((sr+r)^*\big) && [\text{def. } \cdot^*] \\
&= L\big(r(sr+r)^*\big) && [\text{def. concat.}]
\end{aligned}$$

Esercizio 4.17

Dimostrare la validità della seguente espressione regolare o uno specifico esempio per il quale la relazione non è valida:

$$s(rs+s)^*r = rr^*s(rr^*s)^*.$$

Soluzione. Siano $\Sigma = \{0, 1\}$ un alfabeto e $r = 0, s = 1$ espressioni regolari su Σ. La stringa $x = 10 \in L\big(s(rs+s)^*r\big), x \notin L\big(rr^*s(rr^*s)^*\big)$.

Esercizio 4.18

Dimostrare la validità della seguente espressione regolare o uno specifico esempio per il quale la relazione non è valida:

$$(r^* + s^*)^* = (rs)^*.$$

Soluzione. Siano $\Sigma = \{0, 1\}$ un alfabeto e $r = 0, s = 1$ espressioni regolari su Σ. La stringa $x = 10 \in L\big((r^*+s^*)^*\big), x \notin L\big((rs)^*\big)$.

Esercizio 4.19

Dimostrare la validità della seguente espressione regolare o uno specifico esempio per il quale la relazione non è valida:

$$\big((a(a+b))^*\varnothing^*\big)^* = (ab+aa)^*.$$

Soluzione.

$$\begin{aligned} L\big((a(a+b))^*\varnothing^*)^*\big) &= L\big(((aa+ab)^*\varnothing^*)^*\big) \\ &= L\big((aa+ab)^*\epsilon\big)^* \\ &= L\big((aa+ab)^*\big)^* \\ &= L\big((aa+ab)^*) \\ &= L\big((ab+aa)^*). \end{aligned}$$

Esercizio 4.20

I linguaggi denotati dalle espressioni regolari prendono il nome di linguaggi regolari. Sia Σ un alfabeto, tutti i linguaggi $L \subseteq \Sigma^*$ sono regolari?

Soluzione. La risposta è no, per ragioni di cardinalità: sia $\Sigma_{01} = \{ 0, 1 \}$, utilizziamo il metalinguaggio delle espressioni regolari per descrivere il linguaggio delle le espressioni regolari sull'alfabeto Σ_{01}.

1. Creiamo un nuovo alfabeto Σ_{ER01} dato dall'unione dei simboli in Σ_{01} con i simboli del metalinguaggio delle espressioni regolari, (distinguiamo i simboli che possono creare ambiguità mettendo $\hat{\cdot}$), ovvero:
$$\Sigma_{\text{ER01}} = \{ 0, 1, \hat{\epsilon}, \hat{\varnothing}, \hat{\cdot}, \hat{+}, \hat{*} \}.$$

2. Costruiamo la seguente espressione regolare che denota tutte le stringhe su Σ_{ER01}, ovvero l'insieme Σ^*_{ER01}:
$$r = (0 + 1 + \hat{\epsilon} + \hat{\varnothing} + \hat{\cdot} + \hat{+} + \hat{*})^*.$$
Notiamo anche che $L(r) = \Sigma^*_{\text{ER01}}$, quindi in $L(r)$ ci sono sicuramente tutte le espressioni regolari su Σ_{01} in più ci stanno anche altre stringhe che non sono espressioni regolari, esempio $\hat{+}\hat{*}$ ma non è un problema.

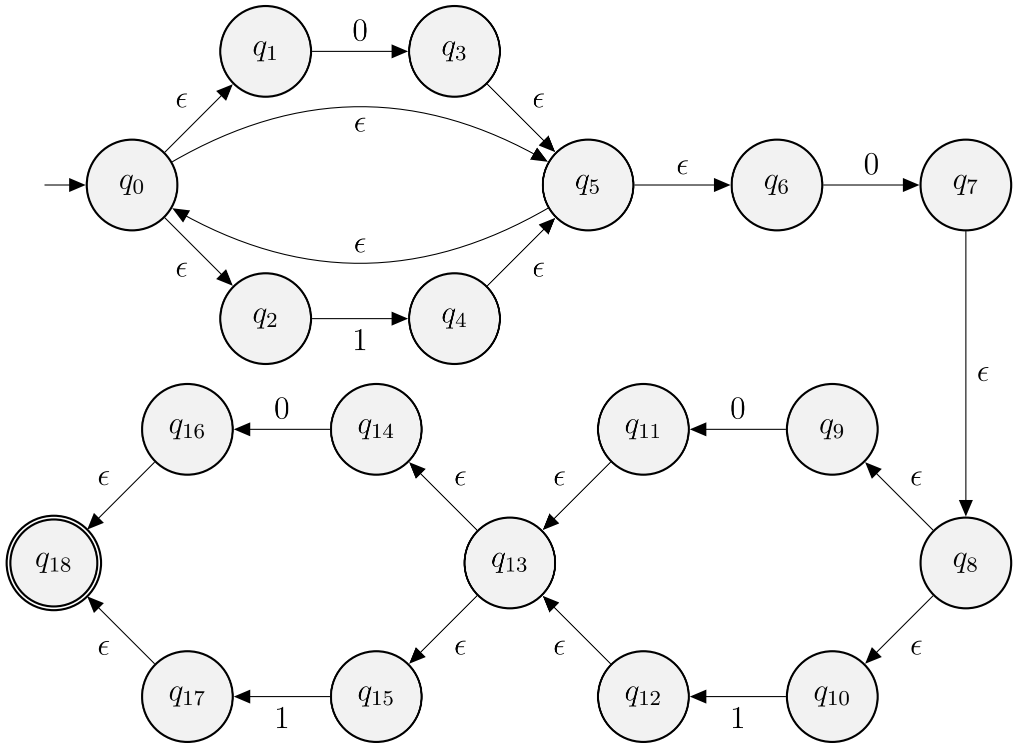
3. Trovare la cardinalità di $L(r) = \Sigma^*_{\text{ER01}}$ equivale a trovare un limite superiore al numero di espressioni regolari su Σ_{01}.
Dato un generico $\Sigma \neq \varnothing$, abbiamo visto un algoritmo per enumerare Σ^*, quindi $|\Sigma^*| = \aleph_0$. Da questo segue che $|L(r)| = |\Sigma^*_{\text{ER01}}| = \aleph_0$.
Ora sappiamo che ci sono al più $\aleph_0$ espressioni regolari su Σ_{01} e in generale su qualsiasi altro alfabeto.

4. Quanti sono i linguaggi formali $L \subseteq \Sigma^*_{01}$? Ovvero, quanti sono i sottoinsiemi di Σ^*_{01}?
$$|\wp(\Sigma^*_{01})| = 2^{|\Sigma^*_{01}|} = 2^{\aleph_0} > \aleph_0 = |\Sigma^*_{\text{ER01}}| \qquad \text{[Teorema 1.1]}.$$

Abbiamo mostrato che dato Σ, le espressioni regolari su Σ sono $\aleph_0$, quindi i linguaggi regolari su Σ sono $\aleph_0$. I linguaggi formali su Σ invece sono $\aleph_1$, questo significa che i linguaggi regolari sono soltanto una piccola parte dei linguaggi formali.

Esercizio 4.21

Determinare, tramite la costruzione di Thompson, un ϵ-NFA che riconosce il linguaggio denotato dalla seguente espressione regolare:
$$(0 + 1)^*0(0 + 1)(0 + 1).$$



Esercizio 4.22

Siano x una stringa e r una espressione regolare, esiste un algoritmo per determinare se $x \lessdot r$?

Soluzione. Si, dal momento che espressioni regolari e DFA sono due formalismi equivalenti per la descrizione finita di linguaggi potenzialmente infiniti, è possibile determinare in maniera algoritmica se $x \lessdot r$ costruendo il DFA equivalente ad r e vedendo se la stringa x è accettata.

Esercizio 4.23

Siano r, s espressioni regolari sullo stesso alfabeto Σ, esiste un algoritmo per determinare se r ed s sono equivalenti, ovvero: esiste un algoritmo per determinare se $L(r) = L(s)$?

Soluzione. Si, esiste un algoritmo per determinare se due espressioni regolari r, s sullo stesso alfabeto Σ sono equivalenti: è possibile costruire i due

DFA equivalenti M_r, M_s per poi minimizzarli e confrontare se i due DFA minimizzati sono *isomorfi*:

$$\min(M_r) \cong \min(M_s),$$

ovvero se è possibile determinare una corrispondenza biunivoca tra gli stati di $\min(M_r)$ e gli stati di $\min(M_s)$ (vedi minimizzazione nel prossimo capitolo).

5 Linguaggi Regolari

5.1 Relazioni e Classi di Equivalenza

Definizione 5.1 (Relazione di equivalenza)**.** *Sia S un insieme. Una relazione $R \subseteq S \times S$ è detta di* equivalenza *se rispetta le seguenti proprietà:*

1. riflessiva*, $\forall a \in S, a\ R\ a$;*

2. simmetrica*, $\forall a, b \in S,$ se $a\ R\ b = b\ R\ a$;*

3. transitiva*, $\forall a, b, c \in S,$ se $a\ R\ b, b\ R\ c,$ allora $a\ R\ c$.*

Definizione 5.2 (Classi di equivalenza)**.** *Sia S un insieme. Una relazione di equivalenza $R \subseteq S \times S$ induce ad una* partizione *di $S = S_1 \cup S_2 \cdots$. Le varie S_i sono dette* classi di equivalenza *e vale che:*

1. $\forall i : S_i \neq \varnothing$;

2. $\forall i, j, i \neq j : S_i \cap S_j = \varnothing$;

3. $\forall i, \forall a, b \in S_i : a\ R\ b$;

4. $\forall i, j, i \neq j, a \in S_i, b \in S_j : a \not R\ b$;

Se $a \in S_i$, $[a]_R$ indica la classe di equivalenza di appartenenza di a, quindi S_i.

Definizione 5.3. *(Relazione di equivalenza di indice finito) Sia S un insieme. Una relazione di equivalenza $R \subseteq S \times S$ è di* indice finito *se partiziona S in un numero finito di classi di equivalenza.*

Esempi. La relazione di equivalenza su $\mathbb{N}$ pià *raffinata* di tutte, quella di uguaglianza:

$$R_1 = \{ (n, m) \mid n \in \mathbb{N}, m \in \mathbb{N}, n = m \},$$
$$[\mathbb{N}]_{R_1} = \big\{ \{ 0 \}, \{ 1 \}, \{ 2 \} \cdots \big\}.$$

La relazione di equivalenza su $\mathbb{N}$ meno raffinata di tutte:

$$R_2 = \{ (n, m) \mid n \in \mathbb{N}, m \in \mathbb{N} \},$$
$$[\mathbb{N}]_{R_2} = \{ \mathbb{N} \}.$$

La relazione di equivalenza su $\mathbb{N}$ che partiziona in numeri pari e dispari:

$$\begin{aligned} R_3 &= \{\, (n, m) \mid n \mod 2 = m \mod 2 \,\}, \\ [\mathbb{N}]_{R_3} &= \big\{\, \{pari\}, \{dispari\} \,\big\}. \end{aligned}$$

La relazione di equivalenza su $\mathbb{N}$ partiziona i numeri in base al loro numero $i \geq 1$ di cifre:

$$\begin{aligned} R_4 &= \{\, (n, m) \mid 10^{i-1} \leq n, m < 10^i \,\}, \\ [\mathbb{N}]_{R_4} &= \big\{\, \{0, \dots, 9\}, \{10, \dots, 99\}, \{\, 100, \dots, 999 \,\}, \dots \big\}. \end{aligned}$$

Lemma 5.1 (Linguaggio associato ad uno stato)**.** *Sia M un DFA. Il linguaggio associato ad uno stato q, denotato con L_q, è definito come*

$$L_q := \{\, x \mid \hat{\delta}(q_0, x) = q \,\},$$

quindi

$$L(M) = \bigcup_{q \in F} L_q.$$

Lemma 5.2 (Relazione di equivalenza R_M)**.** *Sia $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ un DFA. La relazione di equivalenza indotta dall'automa M si denota con $R_M \subseteq \Sigma^* \times \Sigma^*$ ed è definita come*

$$R_M := \{\, (x, y) \mid \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) \,\}$$

Quindi $x\ R_M\ y$ (equivalente a $(x, y) \in R_M$) se e solo se, a partire da q_0, leggendo x o leggendo y il DFA arriva nello stesso stato. Notare che:

1. *le classi di equivalenza della partizione indotta da R_M sono esattamente i linguaggi associati ad ogni stato dell'automa M;*

2. *i DFA hanno un numero finito di stati quindi R_m è di indice finito.*

Dimostrazione. Procediamo dimostrando che valgono le 3 proprietà.

Riflessività: con $x = y$ abbiamo che

$$\begin{aligned} x\ R_M\ y &\iff \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) && [\text{def. di } R_M] \\ &\iff \hat{\delta}(q_0, x) = \hat{\delta}(q_0, x) && [x = y] \\ &\iff x\ R_M\ x && [\text{def. di } R_M]. \end{aligned}$$

Simmetria:

$$\begin{aligned}
x\ R_M\ y &\iff \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) && \text{[def. di } R_M\text{]}\\
&\iff \hat{\delta}(q_0, y) = \hat{\delta}(q_0, x)\\
&\iff y\ R_M\ x && \text{[def. di } R_M\text{]}
\end{aligned}$$

Transitività: vorremmo mostrare che se $x\ R_L\ y$ e $y\ R_L\ z$, allora $x\ R_L\ z$.

$$y\ R_M\ z \iff \hat{\delta}(q_0, y) = \hat{\delta}(q_0, z) \qquad \text{[def. di } R_M\text{]}, \tag{27}$$

allora

$$\begin{aligned}
x\ R_M\ y &\iff \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) && \text{[def. di } R_M\text{]}\\
&\iff \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) = \hat{\delta}(q_0, z) && [(27)]\\
&\iff x\ R_m\ z.
\end{aligned}$$

□

Definizione 5.4. *(Estensione distintiva e relazione di equivalenza R_L). Sia L un linguaggio su Σ. Siano $x, y \in L$. Una* estensione distintiva *di x e y è data da una stringa $z \in \Sigma^*$ tale che soltanto una tra xz e yz appartiene ad L. Se tale z non esiste allora*

$$x\ R_L\ y.$$

Quindi $R_L \subseteq \Sigma^* \times \Sigma^*$ è una relazione di equivalenza indotta da un L ed è definita come

$$R_L := \{\, (x, y) \mid \forall z \in \Sigma^* : xz \in L \iff yz \in L) \,\}.$$

Dimostrazione. Procediamo dimostrando che valgono le 3 proprietà.

Riflessività: con $x = y$ abbiamo che

$$\begin{aligned}
x\ R_L\ y &\iff (\forall z \in \Sigma^*, xz \in L \iff yz \in L) && \text{[def. di } R_L\text{]}\\
&\iff (\forall z \in \Sigma^*, xz \in L \iff xz \in L) && [x = y]\\
&\iff x\ R_L\ x && \text{[def. di } R_L\text{]}.
\end{aligned}$$

Simmetria:

$$\begin{aligned}
x\ R_L\ y &\iff (\forall z \in \Sigma^*, xz \in L \iff yz \in L) && \text{[def. di } R_L\text{]}\\
&\iff (\forall z \in \Sigma^*, yz \in L \iff xz \in L)\\
&\iff y\ R_L\ x && \text{def. di } R_L.
\end{aligned}$$

Transitività: vorremmo mostrare che se $w\ R_L\ x$ e $x\ R_L\ y$, allora $w\ R_L\ y$.

$$x\ R_L\ y \iff (\forall z \in \Sigma^*, xz \in L \iff yz \in L) \qquad [\text{def. di } R_L], \tag{28}$$

allora

$$\begin{aligned} w\ R_L\ x &\iff (\forall z \in \Sigma^*, wz \in L \iff xz \in L) && [\text{def. di } R_L] \\ &\iff (\forall z \in \Sigma^*, wz \in L \iff xz \in L \iff yz \in L) && [(28)] \\ &\iff w\ R_L\ y && [\text{def. di } R_L]. \end{aligned}$$

□

Notare che non abbiamo fatto ipotesi su L, quindi potrebbe essere un linguaggio regolare come no;

1. *se L non è regolare, R_L* non *è di indice finito;*
2. *se L è regolare, R_L è di indice finito e l'indice (numero di classi di equivalenza indotte da R_L) corrisponde proprio al numero di stati del DFA minimo che accetta L.*

Esempio Sia $\Sigma^* = \{ 0, 1 \}$. Sia $L = \{ x0 \mid x \in \Sigma^* \}$.

$$\begin{aligned} &(\epsilon, 0) \notin R_L : z = \epsilon, \\ &(0, 00) \in R_L : \forall z \in \Sigma^*, \\ &(0, 1) \notin R_L : z = \epsilon, \\ &(11, 101) \in R_L : \forall z \in \Sigma^*. \end{aligned}$$

$$R_L = \big\{ \{ x0 \mid x \in \Sigma^* \}, \{ y \mid y = w1 \lor y = \epsilon, w \in \Sigma^* \} \big\}.$$

Costruire per esercizio il DFA minimo che accetta L e confrontare il suo numero di stati con il numero di classi di equivalenza in R_L.

Definizione 5.5 (Relazione di equivalenza invariante a destra rispetto alla concatenazione)**.** *Una relazione $R \subseteq \Sigma^* \times \Sigma^*$ per cui vale la seguente proprietà*

$$x\ R\ y \implies \forall z \in \Sigma^* : xz\ R\ yz,$$

è una relazione di equivalenza invariante a destra *(rispetto alla concatenazione). R_L, R_M sono relazioni di equivalenza invarianti a destra.*

Intuitivamente, l'invarianza a destra rispetto alla concatenazione permette di fare il seguente ragionamento dal punto di vista di un DFA: non importa che cosa ho letto per arrivare allo stato corrente, posso "dimenticare" il passato perché l'unica cosa che conta è ciò che verrà letto da questo punto in poi. Se un DFA ha la possibilità di dimenticare ciò che ha letto, potrebbe indicare che un quantità finita di memoria finita possa essere sufficiente per accettare le stringhe di un determinato linguaggio.

Teorema 5.1. *(Myhill-Nerode: DFA minimo) Sia L un linguaggio regolare. R_L è di indice finito ed il numero di classi di equivalenza della partizione indotta da R_L è uguale al numero di stati del minimo DFA che accetta L.*

Corollario 5.1 (R_M raffina R_L)**.** *R_M è un* raffinamento *di R_L, ovvero il numero di classi di equivalenza della partizione indotta da R_L è sempre maggiore o uguale del numero di classi di equivalenza della partizione indotta da R_L.*

Teorema 5.2 (Myhill-Nerode: enunciati sui linguaggi regolari)**.** *I seguenti enunciati sono equivalenti:*

1. *$L \subseteq \Sigma^*$ è regolare;*
2. *L è l'unione di* alcune *classi di equivalenza indotte da una relazione invariante a destra di indice finito (la relazione di equivalenza in questione è R_M);*
3. *R_L è di indice finito.*

5.1.1 Minimizzazione di DFA

Teorema 5.3 (DFA minimo)**.** *Per ogni linguaggio reoglare L esiste un* unico *automa M con un minimo numero di stati tale che*

$$L(M) = L.$$

La procedura che si utilizza per la minimizzazione di DFA ha come obiettivo quello di trovare gli stati che sono equivalenti tra loro per poi "fonderli" in un unico stato, l'algoritmo quindi utilizza una relazione di equivalenza tra stati.

Definizione 5.6 (Relazione di equivalenza tra stati)**.** *Sia* $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ *un DFA. Due stati* $p, q \in Q$ *sono equivalenti tra loro,* $p\ R\ q$*, se e solo se valgono le seguenti proprietà:*

1. *l'insieme delle stringhe che portano allo stato* p *e l'insieme delle stringhe che portano allo stato* q *sono lo stesso insieme;*

2. *gli stati raggiungibili a partire da* p *con una transizione sono equivalenti agli stati raggiungibili a partire da* q *in una transizione.*

Notare la forte somiglianza tra questa definizione con la relazione di equivalenza R_L*.*

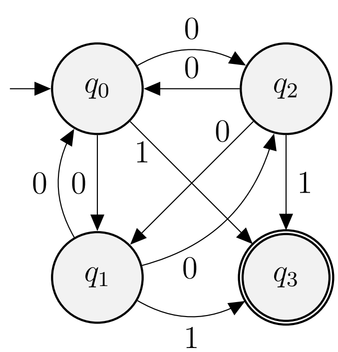
L'algoritmo di minimizzazione che vedremo andrà ad "implementare" la relazione di equivalenza tra stati (definita sopra) partendo da una partizione "grezza" con due sole classi di equivalenza: l'insieme di stati finali e l'insieme degli stati non finali, ovvero $P = F \cup Q \setminus F$ che iterando verrà raffinata fino ad ottenere la partizione desiderata.

L'idea di base per trovare i raffinamenti da applicare è che, data una partizione $P = S_1 \cup S_2 \cdots \cup S_n$, presi $p, q \in S_i$, se

$$\exists a \in \Sigma\ .\ \delta(p, a) \in S_j, \delta(q, a) = S_k \text{ con } j \neq k,$$

allora p e q non rispettano le proprietà richieste ($p \not R\ q$) e non possono stare entrambi nella classe di equivalenza S_i.

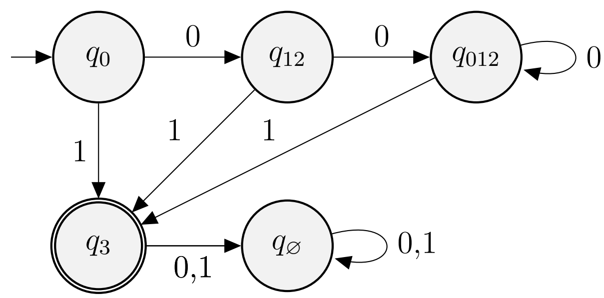
Esempio. Grafo di transizione di un NFA:



Matrice di transizione del DFA equivalente tramite costruzione per sottoinsiemi:

	0	**1**
$\boldsymbol{q_0}$	q_{12}	q_3
$\boldsymbol{q_{12}}$	q_{012}	q_3
$\boldsymbol{\dot{q}_3}$	$q_\varnothing$	$q_\varnothing$
$\boldsymbol{q_{012}}$	q_{012}	q_3
$\boldsymbol{q_\varnothing}$	$q_\varnothing$	$q_\varnothing$

Grafo di transizione del DFA con i soli stati raggiungibili partendo da q_0:



Minimizzazione:

$$\Pi_1 = \big\{ S_1 = \{\, q_0, q_{12}, q_{012}, q_\varnothing \,\}, S_2 = \{\, q_3 \,\}, \big\}$$

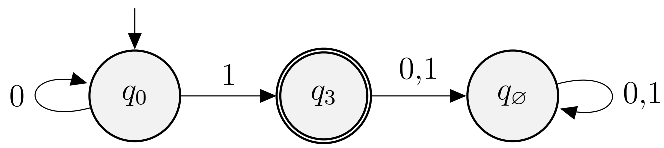
Π_1	**0**	**1**
$\boldsymbol{q_0}$	S_1	S_2
$\boldsymbol{q_{12}}$	S_1	S_2
$\boldsymbol{q_{012}}$	S_1	S_2
$\boldsymbol{q_\varnothing}$	S_1	S_1

$$\Pi_2 = \big\{ S_1 = \{\, q_0, q_{12}, q_{012} \,\}, S_2 = \{\, q_3 \,\}, S_3 = \{\, q_\varnothing \,\} \big\}$$

Π_2	**0**	**1**
$\boldsymbol{q_0}$	S_1	S_2
$\boldsymbol{q_{12}}$	S_1	S_2
$\boldsymbol{q_{012}}$	S_1	S_2

$$\Pi_3 = \big\{ \{ q_0, q_{12}, q_{012} \}, \{ q_3 \}, \{ q_t \} \big\}$$

Punto fisso raggiunto, costruiamo il DFA minimo:



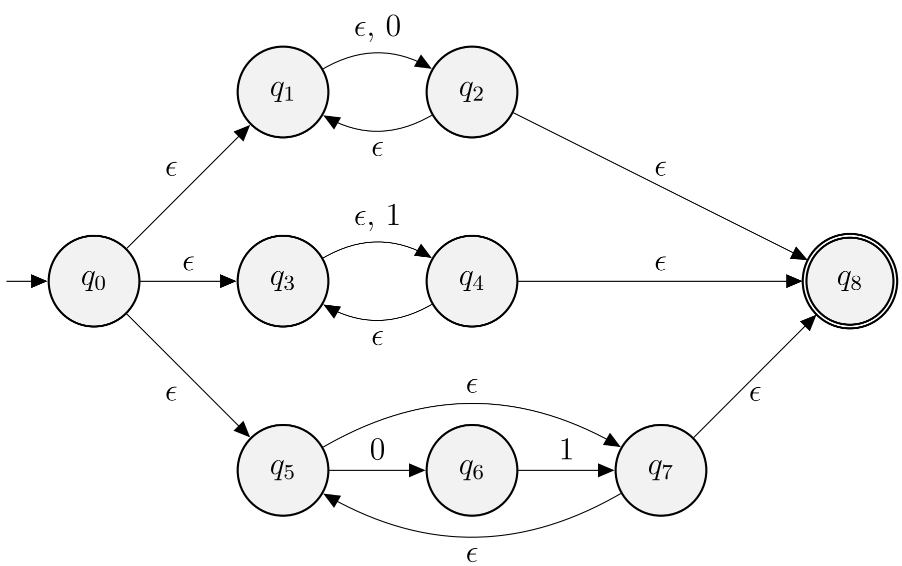
5.1.2 Esercizi

Esercizio 5.1

Trovare il DFA minimo equivalente alla seguente espressione regolare:

$$\big(0^* + 1^* + (01)^*\big).$$

Soluzione. Costruzione di Thompson:



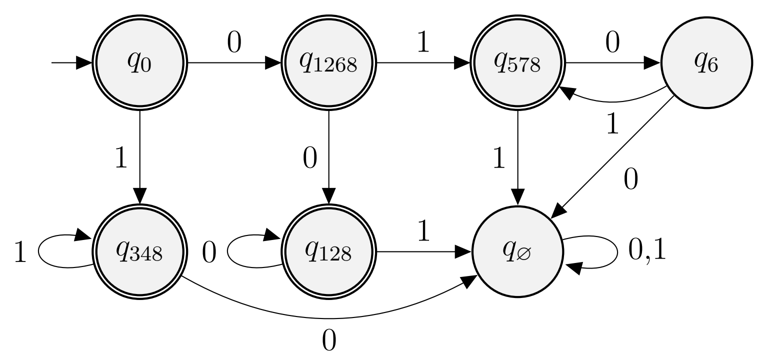
Eliminazione delle ϵ-transizioni:

	0	**1**
$\dot{q}_0$	$\{ q_1, q_2, q_6, q_8 \}$	$\{ q_3, q_4, q_8 \}$
q_1	$\{ q_1, q_2, q_8 \}$	$\varnothing$
q_2	$\{ q_1, q_2, q_8 \}$	$\varnothing$
q_3	$\varnothing$	$\{ q_3, q_4, q_8 \}$
q_4	$\varnothing$	$\{ q_3, q_4, q_8 \}$
q_5	$\{ q_6 \}$	$\varnothing$
q_6	$\varnothing$	$\{ q_5, q_7, q_8 \}$
q_7	$\{ q_6 \}$	$\varnothing$
q_8	$\varnothing$	$\varnothing$

Matrice del DFA equivalente con costruzione per sottoinsiemi:

	0	**1**
$\dot{q}_0$	q_{1268}	q_{348}
$\dot{q}_{1268}$	q_{128}	q_{578}
$\dot{q}_{348}$	$q_\varnothing$	q_{348}
$q_\varnothing$	$q_\varnothing$	$q_\varnothing$
$\dot{q}_{128}$	q_{128}	$q_\varnothing$
$\dot{q}_{578}$	q_6	$q_\varnothing$
q_6	$q_\varnothing$	q_{578}

Grafo di transizione del DFA equivalente con i soli stati raggiungibili a partire da q_0:



Minimizzazione:

$$\Pi_1 = \big\{ S_1 = \{ q_0, q_{1268}, q_{578}, q_{348}, q_{128} \}, S_2 = \{ q_6, q_\varnothing \} \big\}$$

Π_1	**0**	**1**
$\boldsymbol{q_0}$	S_1	S_1
$\boldsymbol{q_{1268}}$	S_1	S_1
$\boldsymbol{q_{578}}$	S_2	S_2
$\boldsymbol{q_{348}}$	S_2	S_1
$\boldsymbol{q_{128}}$	S_1	S_2
$\boldsymbol{q_6}$	S_2	S_1
$\boldsymbol{q_\varnothing}$	S_2	S_2

$$\Pi_2 = \big\{S_1 = \{\, q_0, q_{1268} \,\}, S_2 = \{\, q_{578} \,\}, S_3 = \{\, q_{348} \,\}, S_4 = \{\, q_{128} \,\}, S_5 = \{\, q_6, \,\}, S_6 = \{\, q_\varnothing \,\}\big\}$$

Π_2	**0**	**1**
$\boldsymbol{q_0}$	S_1	S_3
$\boldsymbol{q_{1268}}$	S_4	S_2

$$\Pi_3 = \big\{\{\, q_0 \,\}, \{\, q_{578} \,\}, \{\, q_{348} \,\}, \{\, q_{128} \,\}, \{\, q_6, \,\}, \{\, q_\varnothing \,\}, \{\, q_{1268} \,\}\big\}$$

Il DFA di partenza era già minimo.

5.2 Risultati di decidibilità

Una proposizione è decidibile quando esiste un algoritmo che ne determina la veridicità o falsità.

Definizione 5.7 (Chiusura di un insieme rispetto ad un operazione)**.** *Sia S un insieme ed $R \subseteq S$. Sia $\star : S \times S \mapsto S$ una generica operazione. R è* chiuso *rispetto a $\star$ se*

$$\forall x, y \in R : x \star y \in R,$$

ovvero eseguendo l'operazione non si esce dal sottoinsieme.

Esempio. L'insieme dei numeri interi positivi (sottoinsieme di $\mathbb{Z}$) è chiuso rispetto all'operazione di addizione perchè sommando due numeri interi positivi si ottiene sempre un numero intero positivo ma non è chiuso rispetto all'operazione di sottrazione.

Notare che, considerando l'insieme $R \subset \wp(\Sigma^*)$ come l'insieme dei linguaggi regolari, ha senso parlare di chiusura di R rispetto ad alcune operazioni insiemistiche (essendo gli elementi di R dei linguaggi e quindi insiemi).

Teorema 5.4. *I linguaggi regolari sono chiusi rispetto alle operazioni di*

1. *unione, (dalla costruzione di Thmopson);*
2. *concatenazione, (dalla costruzione di Thompson);*
3. *chiusura di Kleene, (dalla costruzione di Thompson);*
4. *complementazione, (dallo scambio di stati finali e non finali);*
5. *intersezione,* $(L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}})$.

Notare che è possibile utilizzare le proposizioni appena viste anche per cosntrapposizione, per esempio: se $L_1 \cup L_2$ non è regolare, allora almeno uno tra L_1 ed L_2 non è regolare.

Abbiamo visto che l'unione di due linguaggi regolari è regolare, quindi, unendo linguaggi due alla volta è possibile creare una unione di 3, 4, 5,... linguaggi e questa sarà ancora regolare. Consideriamo la famiglia di linguaggi per $i \in \mathbb{N} : L_i = \{ 0^i 1^i \}$, quindi $\{ L_0 = \{ \epsilon \}, L_1 = \{ 01 \}, L_2 = \{ 0011 \}, \dots \}$. Ogni L_i è regolare perchè, fissato i, è possibile costruire un automa che accetta L_i. Definiamo il linguaggio $L = \bigcup_{i \in \mathbb{N}} L_i$; L non è regolare perché è esattamente uguale a $\{ 0^n 1^n \mid n \in \mathbb{N} \}$ di cui abbiamo già dimostrato la non regolarità. Questo ci suggerisce che alcune proprietà (come la chiusura rispetto all'unione di linguaggi regolari) potrebbero non valere più se applicate infinite volte.

Considerando invece la famiglia di linguaggi $L_i = \{ x \in \{ 0, 1 \}^* \mid |x| = i \in \mathbb{N} \}$ quindi $\{ L_0 = \{ \epsilon \}, L_1 = \{ 0, 1 \}, L_2 = \{ 00, 01, 10, 11 \}, \dots \}$. Ogni L_i è regolare perchè, fissato i, è possibile costruire un automa che accetta le stringhe di lunghezza i. Definiamo il linguaggio $L = \bigcup_{i \in \mathbb{N}} L_i$; questa volta anche L è regolare (R_L è di indice finito), infatti $L = \Sigma^*$ che abbiamo visto essere regolare.

5.2.1 Esercizi

Esercizio 5.2

Sia $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ un DFA, spiegare come ottenere un DFA $\overline{M}$ con lo stesso alfabeto di input Σ tale che, accetta tutte le stringhe che non sono accettate da M.

Soluzione. $\overline{M}$ sarà del tutto uguale ad M, tranne che per l'insieme degli stati finali $\overline{F}$ definito come segue:

$$\overline{F} = Q \setminus F.$$

Abbiamo invertito gli stati accettanti con quelli non accettanti.

Esercizio 5.3

Dati due DFA M_1, M_2 con lo stesso alfabeto Σ, spiegare come costruire un DFA M che accetta le stringhe che sono accettate sia da M_1 che da M_2 ovvero $L(M) = L(M_1) \cap L(M_2)$, e trovare il numero di stati di M. [*Suggerimento:* considerare gli stati di M come coppie ordinate di stati (p, q), con $p \in M_1, q \in M_2$].

Soluzione. Il DFA M è una quintupla $M = \langle Q, \Sigma, \delta, q_0, F \rangle$, dove:

- $Q = Q_1 \times Q_2$;
- Σ;
- $\delta\big((p, q), a\big) = \big(\delta_1(p, a), \delta_2(q, a)\big)$;
- $q_0 = (q_0^{(1)}, q_0^{(2)})$;
- $F = \{\, (p, q) \mid p \in F_1, q \in F_2 \,\} = F_1 \times F_2$.

Notiamo che per costruire un DFA $M' = M_1 \cup M_2$ basta modificare l'insieme degli stati finali nel seguente modo:

$$F' = \{\, (p, q) \mid (p \in F_1, q \in Q_2) \vee (p \in Q_1, q \in F_2) \,\} = F_1 \times Q_2 \cup Q_1 \times F_2.$$

Se n_1, n_2 sono il numero di stati di M_1 ed M_2 rispettivamente, allora M avrà $n_1 \cdot n_2$ stati ovvero tutte le possibili coppie in cui il primo elemento è stato di M_1 e il secondo elemento è stato di M_2.

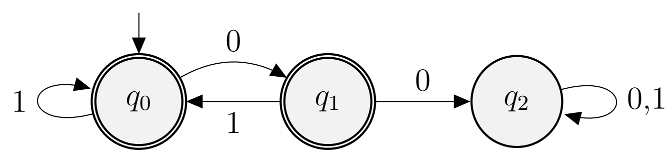
Esercizio 5.4

Dimostrare che l'insieme delle stringhe di 0 e 1 tali che:

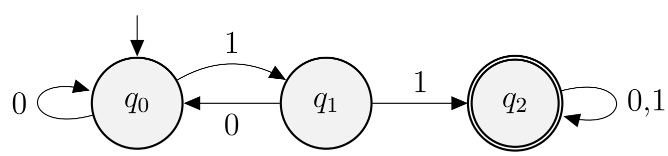
1. non hanno mai due '0' consecutivi;
2. hanno almeno due '1' consecutivi.

è un linguaggio regolare. [*Suggerimento:* dividere il problema in sottoproblemi elementari e utilizzare i risultati di questo paragrafo per combinarli.]

Soluzione. Grafo di transizione DFA di 1):



Grafo di transizione DFA di 2):



Trovati i due DFA sappiamo che L_1 ed L_2 sono regolari ma allora anche $L = L_1 \cap L_2$ è regolare visto che i linguaggi regolari sono chiusi rispetto all'intersezione. Per esercizio proviamo a costruire il DFA M tale che $L(M) = L_1 \cap L_2$.

	0	**1**
$\boldsymbol{q_0} = (\dot{q}_0, q_0)$	$(q_1, q_0) = q_3$	$(q_0, q_1) = q_1$
$\boldsymbol{q_1} = (\dot{q}_0, q_1)$	$(q_1, q_0) = q_3$	$(q_0, q_2) = q_2$
$\dot{\boldsymbol{q}}_\mathbf{2} = (\dot{q}_0, \dot{q}_2 \}$	$(q_1, q_2) = q_5$	$(q_0, q_2) = q_2$
$\boldsymbol{q_3} = (\dot{q}_1, q_0)$	$(q_2, q_0) = q_6$	$(q_0, q_1) = q_1$
$\boldsymbol{q_4} = (\dot{q}_1, q_1)$	$(q_2, q_0) = q_6$	$(q_0, q_2) = q_2$
$\dot{\boldsymbol{q}}_\mathbf{5} = (\dot{q}_1, \dot{q}_2)$	$(q_2, q_2) = q_8$	$(q_0, q_2) = q_2$
$\boldsymbol{q_6} = (q_2, q_0)$	$(q_2 q_0) = q_6$	$(q_2, q_1) = q_7$
$\boldsymbol{q_7} = (q_2, q_1)$	$(q_2, q_0) = q_6$	$(q_2, q_2) = q_8$
$\boldsymbol{q_8} = (q_2, \dot{q}_2)$	$(q_2, q_2) = q_8$	$(q_2, q_2) = q_8$

5.3 Risultati di Decidibilità

Per i linguaggi regolari le proprietà decidibili sono molteplici.

Teorema 5.5 (Appartenenza)**.** *Il problema dell'appartenenza per i linguaggi regolari è decidibile; data una descrizione di L tramite DFA o espressione regolare, esiste un algoritmo in grado di verificare se una stringa x appartiene o meno ad L.*

Teorema 5.6 (Vuoto-infinito)**.** *L'insieme delle stringhe accettata da un DFA M con n stati è*

1. *non vuoto se e solo se accetta una stringa di lunghezza minore di n;*
2. *infinito se e solo se accetta una stringa di lunghezza l, $n \leq l < 2n$.*

Corollario 5.2. *Sia M un DFA. I problemi "$L(M) = \varnothing$?" e "$L(M)$ è ininito?" sono entrambi* decidibili*, ovvero esiste un algoritmo per determinare la loro veridicità o falsità.*

Teorema 5.7 (Equivalenza)**.** *Siano M_1, M_3 due DFA. Il problema di stabilire se $L(M_1) = L(M_2)$ è decidibile definendo $L(M_3)$ come segue:*

$$L(M_3) = \left(\overline{L(M_1)} \cap L(M_2)\right) \cup \left(L(M_1) \cap \overline{L(M_2)}\right),$$

e chiedendosi se $L(M_3)$ è vuoto oppure no.

5.4 Pumping Lemma

Abbiamo visto diversi metodi per stabilire se un dato linguaggio è regolare (esibizione di un DFA, NFA, ϵ-NFA o di una espressione regolare.) Abbiamo anche già osservato che non tutti i linguaggi sono regolari, anzi solo una piccola parte sono regolari. Come si può dimostrare che un linguaggio non è regolare? Ovviamente non essere riusciti a trovare un automa o un'espressione regolare adeguati non dimostra nulla.

Lemma 5.3 (Pumping lemma)**.** *Sia L un linguaggio regolare allora $\exists n \in \mathbb{N}$ (*lunghezza di pumping*) tale che, $\forall z \in L : |z| \geq n$, esistono tre stringhe u, v, w tali che:*

1. *$z = uvw$, z viene suddivisa nelle 3 stringhe;*

2. $|v| \geq 1$, *v deve avere almeno lunghezza 1;*

3. $|uv| \leq n$, *v deve stare tra i primi p simboli.*

4. $\forall i \in \mathbb{N} : uv^i w \in L$, *v può essere "pompata" ottenendo nuove stringhe che ancora appartengono a L.*

Quindi:

- *u, la parte prima del pumping;*
- *v, la parte su cui eseguire il pumping ($v^i, i \in \mathbb{N}$);*
- *w, la parte dopo il pumping.*

Questo deriva dal fatto che i DFA permettono la rappresentazione finita linguaggi potenzialmente infiniti. Se il linguaggio accettato da un DFA con n stati è infinito, significa che accetta stringhe che hanno lunghezza $\geq n$; per fare questo, il DFA deve per forza passare più di una volta almeno da uno dei suoi stati, quindi sono presenti cicli nel DFA. Il pumping lemma si basa proprio sulla presenza di tali cicli e asserisce che, dato un linguaggio regolare L, vale la seguente formula:

$$\begin{aligned} &\exists n \in \mathbb{N} \,.\, \forall z \in L : \big(|z| \geq n \Longrightarrow \exists u, v, w \\ &.\, (z = uvw \land |v| \geq 1 \land |uv| \leq n, \land \forall i \in \mathbb{N} : uv^i w \in L)\big). \end{aligned}$$

Notare che:

$$\forall x \in \varnothing : P(x),$$

significa che se per un certo linguaggio L non esistono stringhe $x \in L$ tali che $|x| \geq n$ allora il pumping lemma vale banalmente (questo accade per i linguaggi finiti).

Il pumping lemma viene utilizzato per contrapposizione quando si vuole dimostrare che un linguaggio L *non* è regolare. Per fare ciò bisogna mostrare che vale la negazione della formula sopra, ovvero:

$$\begin{aligned} &\forall n \in \mathbb{N} : \exists z \in L \,.\, \big(|z| \geq n \land \forall u, v, w \\ &.\, \big((z = uvw \land |v| \geq 1 \land |uv| \leq n) \Longrightarrow \exists i \in \mathbb{N} \,.\, uv^i w \notin L\big)\big). \end{aligned}$$

Attenzione: se L è regolare allora per L vale il pumping lemma ma non è un se e solo se, significa che esistono linguaggi non regolari per i quali vale comunque il pumping lemma. Noi quindi useremo la contrapposizione del pumping lemma per mostrare che un linguaggio L, (di cui abbiamo il sospetto che possa non essere regolare), effettivamente non è regolare. E' utile porsi la seguente domanda per avere un'idea se un linguaggio L è regolare: *basta una quantità finita e fissata di memoria per riconoscere L?* Analogamente: è possibile scrivere un programma in C che riconosce L senza utilizzare allocazione dinamica della memoria?

Esempio. $L = \{ 0^i 1^i \mid i \in \mathbb{N} \}$ non è regolare, il problema sta nel fatto che è necessario conteggiare e ricordare il numero di '0' letti per poi contare il numero di '1' letti ma un DFA che fa questo non esiste perché il numero di '0' non è fissato. In generale, ogni volta che c'è da contare dell'informazione e questa informazione ha dimensioni non fissate arbitrariamente grandi, (ad esempio il numero di '0'), allora il linguaggio non è regolare perché avremmo bisogno di un automa con infiniti stati (infinita memoria) per accettare L.

Dimostrazione. Sia $n \in \mathbb{N}$ arbitrario la lunghezza di pumping, scegliamo una stringa $z \in L$ di lunghezza maggiore o uguale a n:

$$z = 0^n 1^n \in L.$$

Mostriamo che comunque vengano presi u, v e w tali che $z = uvw, |uv| \leq n, |v| \geq 1$, esiste almeno un $i \in \mathbb{N}$ tale per cui $uv^i w \notin L$. Essendo $|uv| \leq n$, tali modi sono riconducibili al seguente schema:

$$u = 0^a, v = 0^b, w = 0^c 1^n,$$

con

$$a + b + c = n, a + b \leq n, b \geq 1.$$

La stringa da pompare quindi è della forma:

$$\begin{aligned} 0^a 0^{ib} 0^c 1^n &= 0^a 0^{ib} 0^{n-a-b} 1^n && [c = n - a - b] \\ &= 0^{n+ib-b} 1^n. \end{aligned}$$

Pumping con $i = 2$:

$$\begin{aligned} 0^{n+ib-b} 1^n &= 0^{n+2b-b} 1^n && [i = 2] \\ &= 0^{n+b} 1^n \\ &\notin L && [b \geq 1 \Longrightarrow n + b \neq n]. \end{aligned}$$

□

5.4.1 Esercizi

Esercizio 5.5

Dimostrare che $L = \{ 0^{2^i} \mid i \in \mathbb{N} \}$ non è regolare.

Soluzione. Sia $n \in \mathbb{N}$ arbitrario la lunghezza di pumping, scegliamo una stringa $z \in L$ di lunghezza maggiore o uguale a n:

$$z = 0^{2^n} \in L.$$

Mostriamo che comunque vengano presi u, v e w tali che $z = uvw, |uv| \leq n, |v| \geq 1$, esiste almeno un $i \in \mathbb{N}$ tale per cui $uv^iw \notin L$. Essendo $|uv| \leq n$, tali modi sono riconducibili al seguente schema:

$$u = 0^a, v = 0^b, w = 0^c,$$

con

$$a + b + c = 2^n, a + b \leq n, b \geq 1.$$

La stringa da pompare quindi è della forma:

$$\begin{aligned} 0^a 0^{ib} 0^c &= 0^a 0^{ib} 0^{2^n - a - b} && [c = 2^n - a - b] \\ &= 0^{2^n + ib - b}. \end{aligned}$$

Pumping con $i = 2$:

$$\begin{aligned} 0^{2^n + ib - b}. &= 0^{2^n + 2b - b} && [i = 2] \\ &= 0^{2^n + b} \\ &\notin L. \end{aligned}$$

Mostriamo che la stringa ottenuta dopo il pumping davvero non sta in L perché la sua lunghezza $(2^n + b)$ non è una potenza di 2, ovvero cade tra due potenze di 2 consecutive:

$$\begin{aligned} 2^n &< 2^n + b && [b \geq 1] \\ &< 2^n + 2^b && [b \geq 1] \\ &\leq 2^n + 2^n && [b \leq n] \\ &= 2^{n+1}. \end{aligned}$$

Esercizio 5.6

Dimostrare che $L = \{\, 0^i \mid i \mod k = 0 \iff k = 1 \lor k = i \,\}$ non è regolare.

Soluzione. Sia $n \in \mathbb{N}$ arbitrario la lunghezza di pumping, scegliamo una stringa $z \in L$ di lunghezza p numero primo maggiore o uguale a n:

$$z = 0^p \in L.$$

Mostriamo che comunque vengano presi u, v e w tali che $z = uvw, |uv| \leq p, |v| \geq 1$, esiste almeno un $i \in \mathbb{N}$ tale per cui $uv^iw \notin L$. Essendo $|uv| \leq p$, tali modi sono riconducibili al seguente schema:

$$u = 0^a, v = 0^b, w = 0^c,$$

con

$$a + b + c = p, a + b \leq n, b \geq 1.$$

La stringa da pompare quindi è della forma:

$$\begin{aligned} 0^a0^{ib}0^c &= 0^a0^{ib}0^{p-a-b} && [c = p - a - b] \\ &= 0^{p+ib-b}. \end{aligned}$$

Pumping con $i = p + 1$:

$$\begin{aligned} 0^{p+ib-b}. &= 0^{p+(p+1)b-b} && [i = p + 1] \\ &= 0^{p+pb+b-b} \\ &= 0^{p+pb} \\ &= 0^{p(b+1)} \\ &\notin L && [b \geq 1 \Longrightarrow p(b+1) \mod p = b + 1]. \end{aligned}$$

Esercizio 5.7

Dimostrare che $L = \{\, 0^{i^2} \mid i \in \mathbb{N} \,\}$ non è regolare.

Soluzione. Sia $n \in \mathbb{N}$ arbitrario la lunghezza di pumping, scegliamo una stringa $z \in L$ di lunghezza maggiore o uguale a n:

$$z = 0^{n^2} \in L.$$

Mostriamo che comunque vengano presi u, v e w tali che $z = uvw, |uv| \leq n, |v| \geq 1$, esiste almeno un $i \in \mathbb{N}$ tale per cui $uv^iw \notin L$. Essendo $|uv| \leq n$, tali modi sono riconducibili al seguente schema:

$$u = 0^a, v = 0^b, w = 0^c.$$

con

$$a + b + c = n^2, a + b \leq n, b \geq 1.$$

La stringa da pompare quindi è della forma:

$$\begin{aligned} 0^a 0^{ib} 0^c &= 0^a 0^{ib} 0^{n^2-a-b} && [c = n^2 - a - b] \\ &= 0^{n^2+ib-b}. \end{aligned}$$

Pumping con $i = 2$:

$$\begin{aligned} 0^{n^2+ib-b} &= 0^{n^2+2b-b} && [i = 2] \\ &= 0^{n^2+b} \\ &\notin L. \end{aligned}$$

Mostriamo che la stringa ottenuta dopo il pumping non fa parte di L perché la sua lunghezza $(n^2 + b)$ non è un quadrato perfetto, ovvero cade tra due quadrati perfetti consecutivi:

$$\begin{aligned} n^2 &< n^2 + b && [b \geq 1] \\ &< n^2 + 2b && [b \geq 1] \\ &< n^2 + 2b + 1 \\ &\leq n^2 + 2n + 1 && [a + b \leq n] \\ &= (n + 1)^2. \end{aligned}$$

Esercizio 5.8

Dimostrare che $L = \{\, 0^i 1^{i+1} 0^{i+2} \mid i \in \mathbb{N} \,\}$ non è regolare.

Soluzione. Sia $n \in \mathbb{N}$ arbitrario la lunghezza di pumping, scegliamo una stringa $z \in L$ di lunghezza maggiore o uguale a n:

$$z = 0^n 1^{n+1} 0^{n+2} \in L.$$

Mostriamo che comunque vengano presi u, v e w tali che $z = uvw, |uv| \leq n, |v| \geq 1$, esiste almeno un $i \in \mathbb{N}$ tale per cui $uv^i w \notin L$. Essendo $|uv| \leq n$, tali modi sono riconducibili al seguente schema:

$$u = 0^a, v = 0^b, w = 0^c 1^{n+1} 0^{n+2}.$$

con

$$a + b + c = n, a + b \leq n, b \geq 1.$$

La stringa da pompare quindi è della forma:

$$\begin{aligned} 0^a 0^{ib} 0^c 1^{n+1} 0^{n+2} &= 0^a 0^{ib} 0^{n-a-b} 1^{n+1} 0^{n+2} && [c = n - a - b] \\ &= 0^{n+ib-b} 1^{n+1} 0^{n+2}. \end{aligned}$$

Pumping con $i = 2$:

$$\begin{aligned} 0^{n+ib-b} 1^{n+1} 0^{n+2} &= 0^{n+2b-b} 1^{n+1} 0^{n+2} && [i = 2] \\ &= 0^{n+b} 1^{n+1} 0^{n+2} \\ &\notin L && [b \geq 1]. \end{aligned}$$

Esercizio 5.9

Dimostrare che $L = \{ 0^j 1^k 0^{j+k} \mid j, k \in \mathbb{N} \}$ non è regolare.

Soluzione. Sia $n \in \mathbb{N}$ arbitrario la lunghezza di pumping, scegliamo una stringa $z \in L$ di lunghezza maggiore o uguale a n:

$$z = 0^n 1^m 0^{n+m} \in L, \text{ con } m \geq 1.$$

Mostriamo che comunque vengano presi u, v e w tali che $z = uvw, |uv| \leq n, |v| \geq 1$, esiste almeno un $i \in \mathbb{N}$ tale per cui $uv^i w \notin L$. Essendo $|uv| \leq n$, tali modi sono riconducibili al seguente schema:

$$u = 0^a, v = 0^b, w = 0^c 1^m 0^{n+m},$$

con

$$a + b + c = n, a + b \leq n, b \geq 1.$$

La stringa da pompare quindi è della forma:

$$\begin{aligned} 0^a 0^{ib} 0^c &= 0^a 0^{ib} 0^{n-a-b} 1^m 0^{n+m} && [c = n - a - b] \\ &= 0^{n+ib-b} 1^m 0^{n+m}. \end{aligned}$$

Pumping con $i = 0$:

$$\begin{aligned} 0^{n+ib-b} 1^m 0^{n+m} &= 0^{n+0b-b} 1^m 0^{n+m} && [i = 0] \\ &= 0^{n-b} 1^m 0^{n+m} \\ &\notin L && [b \geq 1 \implies n - b < n]. \end{aligned}$$

Esercizio 5.10

Dimostrare che $L = \{ 0^i 1^j \mid i, j \in \mathbb{N}, i < j \}$ non è regolare.

Soluzione. Sia $n \in \mathbb{N}$ arbitrario la lunghezza di pumping, scegliamo una stringa $z \in L$ di lunghezza maggiore o uguale a n:

$$z = 0^n 1^m \in L, \text{con } n < m.$$

Mostriamo che comunque vengano presi u, v e w tali che $z = uvw, |uv| \leq n, |v| \geq 1$, esiste almeno un $i \in \mathbb{N}$ tale per cui $uv^i w \notin L$. Essendo $|uv| \leq n$, tali modi sono riconducibili al seguente schema:

$$u = 0^a, v = 0^b, w = 0^c 1^m,$$

con

$$a + b + c = n, a + b \leq n, b \geq 1.$$

La stringa da pompare quindi è della forma:

$$\begin{aligned} 0^a 0^{ib} 0^c 1^m &= 0^a 0^{ib} 0^{n-a-b} 1^m && [c = n - a - b] \\ &= 0^{n+ib-b} 1^m. \end{aligned}$$

Pumping con $i = m + 1$:

$$\begin{aligned} 0^{n+ib-b} 1^m &= 0^{n+(m+1)b-b} 1^m && [i = m + 1] \\ &= 0^{n+mb+b-b} 1^m \\ &= 0^{n+mb} 1^m \\ &\notin L && [b \geq 1, \implies n + mb > m]. \end{aligned}$$

Esercizio 5.11

Dimostrare che $L = \{ x \mid C_0(x) = C_1(x) \}$ non è regolare.

Soluzione. Sia $n \in \mathbb{N}$ arbitrario, ci basta scegliere la stringa $z = 0^n 1^n$ ed eseguire il pumping: z sta anche nel linguaggio $L' = \{ 0^n 1^n \mid n \in \mathbb{N} \}$ di cui abbiamo già dimostrato la non regolarità proprio pompando una stringa come z.

Esercizio 5.12

Dimostrare che $L = \{ x \mid C_0(x) < C_1(x) \}$ non è regolare.

Soluzione. Siano $n, m \in \mathbb{N}, n < m$ arbitrari, ci basta scegliere la stringa $z = 0^n 1^m \in L$ ed eseguire il pumping: z sta anche nel linguaggio $L' = \{ 0^n 1^m \mid m, n \in \mathbb{N}, n < m \}$ di cui abbiamo già dimostrato la non regolarità proprio pompando una stringa come z.

Esercizio 5.13

Dimostrare che $L = \{ xx^r \mid x \in \{ 0, 1 \}^* \}$ non è regolare, con x^r definito come segue:

$$x^r := \begin{cases} x, & \text{se } x = \epsilon; \\ w^r a, & \text{se } x = aw. \end{cases}$$

Soluzione. Sia $n \in \mathbb{N}$ arbitrario la lunghezza di pumping, scegliamo una stringa $z \in L$ di lunghezza maggiore o uguale a n:

$$z = 0^n 110^n \in L.$$

Mostriamo che comunque vengano presi u, v e w tali che $z = uvw, |uv| \leq n, |v| \geq 1$, esiste almeno un $i \in \mathbb{N}$ tale per cui $uv^i w \notin L$. Essendo $|uv| \leq n$, tali modi sono riconducibili al seguente schema:

$$u = 0^a, v = 0^b, w = 0^c 110^n.$$

con

$$a + b + c = n, a + b \leq n, b \geq 1.$$

La stringa da pompare quindi è della forma:

$$\begin{aligned} 0^a 0^{ib} 0^c 110^n &= 0^a 0^{ib} 0^{n-a-b} 110^n && [c = n - a - b] \\ &= 0^{n+ib-b} 110^n. \end{aligned}$$

Pumping con $i = 0$:

$$\begin{aligned} 0^{n+ib-b} 110^n &= 0^{n-b} 110^n && [i = 0] \\ &\notin L && [b \geq 1]. \end{aligned}$$

6 Grammatiche

L'approccio *generativo-sintetico* utilizza strutture finite dette *grammatiche* in grado di *generare* insiemi di stringhe per la descrizione finita di linguaggi potenzialmente infiniti

6.1 Grammatiche Libere dal Contesto

Definizione 6.1 (Grammatica libera dal contesto). *Una* grammatica libera dal contesto*, anche detta* acontesuale *o* CF grammar*, è una quadrupla* $G = \langle V, T, P, S \rangle$*, dove:*

- $\boldsymbol{V}$ *è un insieme finito di variabili (dette anche simboli* non terminali*);*
- $\boldsymbol{T}$ *è un insieme finito di simboli* terminali *, con* $V \cap T = \varnothing$*;*
- $\boldsymbol{P}$ *è un insieme finito di* produzioni *della forma:*

  $$A \to \alpha, \ \textit{con } A \in V, \alpha \in (V \cup T)^*.$$

  Ogni produzione quindi indica una possibile "riscrittura" di una variabile in una stringa composta da variabili e/o terminali.
- $\boldsymbol{S} \in V$ *è una variabile speciale detta* simbolo iniziale*.*

Esempio 1. Grammatica che genera le stringhe binarie palindrome:

$$\langle V = \{ S \}, T = \{ \epsilon, 0, 1 \}, P, S \rangle,$$

con insieme delle produzioni P composto da:

$$\begin{aligned} &S \to \epsilon \\ &S \to 0 \\ &S \to 1 \\ &S \to 0S0 \\ &S \to 1S1. \end{aligned}$$

Quando, come in questo caso, per una variabile le riscritture sono molteplici, è possibile utilizzare una singola produzione mettendo in *or* le diverse alternative:

$$S \to \epsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1.$$

Esempio 2. Grammatica che genera il linguaggi $L = \{ 0^n 1^n \mid n \in \mathbb{N} \}$:

$$\langle V = \{ S \}, T = \{ \epsilon, 0, 1 \}, P, S \rangle,$$

con insieme delle produzioni P composto da:

$$S \to \epsilon \mid 0S1$$

Notare che: abbiamo dimostrato nel capitolo del pumping lemma che L non è regolare e abbiamo appena mostrato che L è libero dal contesto. Quindi l'insieme dei linguaggi regolari e l'insieme dei linguaggi liberi dal contesto *non* sono lo stesso insieme.

Sia $A \to \alpha \in P$. Il termine "libere dal contesto" deriva dal fatto che, la riscrittura di A in α è sempre possibile, indipendentemente da che cosa si trova a sinistra o a destra di A, ovvero, A si può sempre riscrivere in α indipendentemente dal *contesto* in cui si trova A (la parte sinistra delle produzioni infatti è sempre costituita da una sola variabile).

Meta linguaggi. Un *meta linguaggio* è un linguaggio utilizzato per la descrizione di un altro linguaggio che prende il nome di linguaggio *oggetto*. Le grammatiche (come le espressioni regolari) sono un meta linguaggio. Le variabili del meta linguaggio prendono il nome di meta variabili e spesso occorre distringuerle in qualche modo dalle variabili del linguaggio oggetto. Le variabili che compaiono nelle grammatiche acontestuali, in particolare, vengono dette meta variabili *sintattiche*, perché quello che fanno è definire la sintassi del linguaggio oggetto, senza dare informazioni esplicite sulla semantica.

Notazione.

- A, B, C, D, E, S denotano variabili;
- S denota il simbolo iniziale;
- $a, b, c, d, e, 0, 1$ denotano simboli terminali;
- X, Y, Z denotano simboli che possono essere sia terminali che variabili;
- u, v, w, x, y, z denotano stringhe di soli terminali;
- $\alpha, \beta, \gamma, \delta$ denotano generiche stringhe di simboli (sia terminali che non).

Definizione 6.2 (Derivazione immediata)**.** *Sia $G = \langle V, T, P, S\rangle$ una grammatica CF. Siano $\alpha, \beta, \gamma \in (V \cup T)^*$. Se $A \to \beta \in P$, allora $\alpha A\gamma \overset{G}{\Rightarrow} \alpha\beta\gamma$. In questo caso di dice che $\alpha A\gamma$* deriva immediatamente *$\alpha\beta\gamma$.*

Definizione 6.3 (Catena di derivazioni immediate)**.** *Sia $G = \langle V, T, P, S\rangle$ una grammatica CF. Una catena di derivazioni immediate è definita in maniera induttiva come segue:*

Caso base:

$$\forall\alpha \in (V \cup T)^* : \alpha \overset{G}{\Rightarrow}_0 \alpha.$$

Passo induttivo:

$$\forall n \in \mathbb{N}, \forall\alpha, \gamma \in (V \cup T)^* : \alpha \overset{G}{\Rightarrow}_{n+1} \gamma \iff \exists\beta \in (V \cup T)^* \,.\, \alpha \overset{G}{\Rightarrow} \beta \wedge \beta \overset{G}{\Rightarrow}_n \gamma.$$

Una definizione equivalente con notazione differente:

$$\textit{Caso base:}\quad \frac{}{\alpha \overset{G}{\Rightarrow}_0 \alpha}\,\alpha \in (V \cup T)^*.$$

$$\textit{Passo induttivo:}\quad \frac{\alpha \overset{G}{\Rightarrow} \beta, \beta \overset{G}{\Rightarrow}_n \gamma}{\alpha \overset{G}{\Rightarrow}_{n+1} \gamma}\,\alpha, \beta, \gamma \in (V \cup T)^*.$$

Secondo quanto appena visto, se esiste un $n \in \mathbb{N}$ tale per cui $\alpha \overset{G}{\Rightarrow}_n \beta$, allora $\alpha \overset{G}{\Rightarrow}_* \beta$, ovvero da α deriva β (α si riscrive in un numero di passi $n \geq 0$ in β). Notare che:

- $\overset{G}{\Rightarrow}$ è la chiusura transitiva (per def. induttiva) e la chiusura riflessiva (per def. di $\overset{G}{\Rightarrow}_0$) della relazione $\overset{G}{\Rightarrow}$;
- $\overset{G}{\Rightarrow}$ denota una relazione tra stringhe.

Ora possiamo definire la sematica di una grammatica, ovvero il linguaggio generato dalla grammatica.

Definizione 6.4 (Linguaggio generato da una grammatica)**.** *Sia G una grammatica CF, il linguaggio* generato *da G, $L(G)$ è definito come:*

$$L(G) := \{\, x \in T^* \mid S \overset{G}{\Rightarrow}_* x \,\},$$

ovvero l'insieme di stringhe composte da soli terminali ottenute con un numero arbitrario di derivazioni a partire dal simbolo iniziale S.

Definizione 6.5 (Linguaggio libero dal contesto)**.** *Sia L un linguaggio, L è libero dal contesto se esiste una grammatica CF G tale che*

$$L = L(G).$$

6.1.1 Esercizi

Esercizio 6.1

Data la seguente grammatica:

$$E \to 0 \mid 1 \mid (E \wedge E) \mid (E \vee E) \mid (\neg E),$$

dimostrare pre induzione strutturata che le espressioni (stringhe) generate hanno una forma *massimamente parentesizzata*: ovvero la precedenza di ogni operazione è esplicitata tramite una coppia di perentesi bilanciate.

Soluzione. Come prima cosa definiamo una funzione che presa una espressione booleana (stringa generata dalla grammatica) conta il numero di parentesi aperte $N_a : (V \cup T)^* \mapsto \mathbb{N}$, definita come

$$N_a(e) := \begin{cases} 0, & \text{se } e = 1 \vee e = 0; \\ 1 + N_a(e_1) + N_a(e_2), & \text{se } e = (e_1 \wedge e_2) \vee e = (e_1 \vee e_2); \\ 1 + N_a(e_1), & \text{se } e = \neg(e_1). \end{cases}$$

In maniera equivalente è possibile definire N_c che conta il numero di parentesi chiuse.

Caso base. Abbiamo due casi base.

1. $S \overset{G}{\Rightarrow} 1$:

$$\begin{aligned} N_a(1) &= 0 && [\text{def. di } N_a] \\ &= N_c(1) && [\text{def. di } N_c]; \end{aligned}$$

2. $S \overset{G}{\Rightarrow} 0$:

$$\begin{aligned} N_a(0) &= 0 && [\text{def. di } N_a] \\ &= N_c(0) && [\text{def. di } N_c]. \end{aligned}$$

Passo induttivo. Abbiamo 3 passi.

1. $S \overset{G}{\Rightarrow}_{n+1} (e_1 \wedge e_2)$:

$$\begin{aligned} N_a\big((e_1 \wedge e_2)\big) &= 1 + N_a(e_1) + N_a(e_2) && \text{[def. di } N_a\text{]} \\ &= 1 + N_c(e_1) + N_c(e_2) && \text{[ip. ind.]} \\ &= N_c\big((e_1 \wedge e_2)\big) && \text{[def. di } N_c\text{];} \end{aligned}$$

2. $S \overset{G}{\Rightarrow}_{n+1} (e_1 \vee e_2)$:

$$\begin{aligned} N_a\big((e_1 \vee e_2)\big) &= 1 + N_a(e_1) + N_a(e_2) && \text{[def. di } N_a\text{]} \\ &= 1 + N_c(e_1) + N_c(e_2) && \text{[ip. ind.]} \\ &= N_c\big((e_1 \vee e_2)\big) && \text{[def. di } N_c\text{];} \end{aligned}$$

3. $S \overset{G}{\Rightarrow}_{n+1} \neg(e_1)$:

$$\begin{aligned} N_a\big(\neg(e_1)\big) &= 1 + N_a(e_1) && \text{[def. di } N_a\text{]} \\ &= 1 + N_c(e_1) && \text{[ip. ind.]} \\ &= N_c\big(\neg(e_1)\big) && \text{[def. di } N_c\text{].} \end{aligned}$$

Esercizio 6.2

Dimostrare che il seguente linguaggio è CF:

$$L = \{\, 0^j 1^j 0^k 1^k \mid j, k \in \mathbb{N} \,\}.$$

Soluzione Troviamo una grammatica CF G tale che $L = L(G)$. Sia $G = \langle V = \{\, S, A \,\}, T = \{\, 0, 1 \,\}, P, S \rangle$, con P insieme delle produzioni contenente:

$$\begin{aligned} &S \to AA \\ &A \to \epsilon \mid 0A1. \end{aligned}$$

Ora dobbiamo dimostrare che $L(G) = L$. Inizieremo analizzando $L(A)$, ovvero il linguaggio generato dalla sola variabile A dimostrando la validità del seguente lemma:

$$L(A) = L_1 = \{\, 0^n 1^n \mid n \in \mathbb{N} \,\}. \tag{29}$$

Inizieremo mostrando per induzione matematica semplice sulla lunghezza $n \geq 1$ della derivazione che $L(A) \subseteq L_1$, ovvero che presa una qualunque stringa che sta in $L(A)$, questa sta anche in L_1:

$$\forall n \geq 1 : A \overset{G}{\Rightarrow}_n 0^{n-1}1^{n-1} \in L_1.$$

Caso base: se $n = 1$ abbiamo che

$$\begin{aligned} &A \overset{G}{\Rightarrow} \epsilon \\ &= 0^0 1^0 \in L_1. \checkmark \end{aligned}$$

Passo induttivo: se $n = m + 1$ abbiamo che

$$\begin{aligned} &A \overset{G}{\Rightarrow}_m 0^{m-1}1^{m-1} && [\text{ip.ind.}] \\ &\iff A \overset{G}{\Rightarrow}_{m-1} 0^{m-1}A1^{m-1} && [A \to \epsilon] \\ &\implies A \overset{G}{\Rightarrow}_m 0^m A1^m && [A \to 0A1] \\ &\implies A \overset{G}{\Rightarrow}_{m+1} 0^{m-1}1^{m-1} \in L_1 \checkmark && [A \to \epsilon]. \end{aligned}$$

Ora dimostriamo che vale anche $L_1 \subseteq L(A)$, ovvero presa una qualunque stringa che sta in L_1, questa sta anche in $L(A)$:

$$\forall x = 0^j 1^j \in L_1 : \exists n \in \mathbb{N} \;.\; A \overset{G}{\Rightarrow}_n x = A \overset{G}{\Rightarrow}_* x,$$

secondo quanto appena dimostrato, n esiste e vale $j + 1$.

Ora che abbiamo dimostrato il lemma (29), possiamo utilizzarlo per dimostrare che $L(G) = L$. Iniziamo mostrando che $L(G) \subseteq L$, quindi presa una qualunque stringa in $L(G)$ questa sta anche in L, ovvero:

$$x \in L(G) \iff S \overset{G}{\Rightarrow}_* x \iff \exists n \in \mathbb{N} \;.\; S \overset{G}{\Rightarrow}_n x \iff S \overset{G}{\Rightarrow} AA \overset{G}{\Rightarrow}_{n-1} x \in L.$$

$$\begin{aligned} S \overset{G}{\Rightarrow} AA \overset{G}{\Rightarrow}_{n-1} x &\iff S \overset{G}{\Rightarrow}_* 0^i 1^i A \overset{G}{\Rightarrow}_* x && [\text{lemma (29)}, i \in \mathbb{N}] \\ &\iff S \overset{G}{\Rightarrow}_* 0^i 1^i 0^j 1^j \in L \checkmark && [\text{lemma (29)}, i, j \in \mathbb{N}]. \end{aligned}$$

Ora mostriamo che $L \subseteq L(G)$: sia $x = 0^j 1^j 0^k 1^k \in L$, allora

$$\begin{aligned} S \overset{G}{\Rightarrow} AA &\overset{G}{\Rightarrow}_{j+1} 0^j 1^j A && [\text{lemma (29)}] \\ &\overset{G}{\Rightarrow}_{k+1} 0^j 1^j 0^k 1^k = x \checkmark && [\text{lemma (29)}]. \end{aligned}$$

Esercizio 6.3

Dimostrare che il seguente linguaggio è CF:

$$L = \{\, 0^j 1^k 0^k 1^j \mid j, k \in \mathbb{N} \,\}.$$

Soluzione Troviamo una grammatica CF G tale che $L = L(G)$. Sia $G = \langle V = \{\, S, A \,\}, T = \{\, 0, 1 \,\}, P, S\rangle$, con P insieme delle produzioni contenente:

$$\begin{aligned} S &\to 0S1 \mid A \\ A &\to \epsilon \mid 1A0. \end{aligned}$$

Ora dobbiamo dimostrare che $L(G) = L$, ovvero $L(G) \subseteq L, L \subseteq L(G)$. Come prima cosa mostriamo che per induzione sulla lunghezza $n \geq 1$ delle derivazioni che $L(S)$ genera stringhe della forma $0^{n-1}A1^{n-1}$. Dopodiché sfrutteremo il lemma $L(A) = L_1 = \{\, 1^n 0^n \mid n \in \mathbb{N} \,\}$ (da dimostrare per esercizio) per concludere che $L(G) \subseteq L$.

Caso base: se $n = 1$ abbiamo che

$$\begin{aligned} S &\overset{G}{\Rightarrow} A && [S \to A] \\ &= 0^0 A 1^0. \checkmark \end{aligned}$$

Passo induttivo: se $n = m + 1$ abbiamo che

$$\begin{aligned} & S \overset{G}{\Rightarrow}_m 0^{m-1}A1^{m-1} && [\text{ip. ind.}] \\ \iff & S \overset{G}{\Rightarrow}_{m-1} 0^{m-1}S1^{m-1} && [S \to A] \\ \implies & S \overset{G}{\Rightarrow}_m 0^m S 1^m && [S \to 0S1] \\ \implies & S \overset{G}{\Rightarrow}_{m+1} 0^m A 1^m \checkmark && [S \to A]. \end{aligned}$$

Se $L(S)$ genera stringhe della forma $0^{n-1}A1^{n-1}$, e $L(A) = L_1 = \{\, 1^n 0^n \mid n \in \mathbb{N} \,\}$, allora $L(G) \subseteq L$, ovvero:

$$S \overset{G}{\Rightarrow}_* 0^j A 1^j \overset{G}{\Rightarrow}_* 0^j 1^k 0^k 1^j \in L.$$

Ora mostriamo che $L \subseteq L(G)$. Sia $x = 0^j 1^k 0^k 1^j \in L$, allora

$$S \overset{G}{\Rightarrow}_{j+1} 0^j A 1^j \overset{G}{\Rightarrow}_{k+1} 0^j 1^k 0^k 1^j.$$

Esercizio 6.4

Dimostrare che il seguente linguaggio è CF:

$$L = \{\, 0^j 1^k 0^{j+k} \mid j, k \in \mathbb{N} \,\}.$$

Soluzione Troviamo una grammatica CF G tale che $L = L(G)$. Sia $G = \langle V = \{\, S, A \,\}, T = \{\, 0, 1 \,\}, P, S \rangle$, con P insieme delle produzioni contenente:

$$\begin{aligned} S &\to 0S0 \mid A \\ A &\to \epsilon \mid 1A0 \end{aligned}$$

Allora $L = L(G)$.

Dimostrazione. Lasciato per esercizio. □

Esercizio 6.5

Dimostrare che il seguente linguaggio è CF:

$$L = \{\, 0^j 1^k 0^{j-k} \mid j, k \in \mathbb{N}, j \geq k \,\}.$$

Soluzione Troviamo una grammatica CF G tale che $L = L(G)$. Sia $G = \langle V = \{\, S, A \,\}, T = \{\, 0, 1 \,\}, P, S \rangle$, con P insieme delle produzioni contenente:

$$\begin{aligned} S &\to 0S0 \mid A \\ A &\to \epsilon \mid 0A1 \end{aligned}$$

Dimostrazione. Lasciato per esercizio. □

Esercizio 6.6

Dimostrare che il seguente linguaggio è CF:

$$L = \{\, 0^j 1^{2j} 0^{3k} 1^{4k} \mid j, k \in \mathbb{N} \,\}.$$

Soluzione Troviamo una grammatica CF G tale che $L = L(G)$. Sia $G = \langle V = \{ S, A, B \}, T = \{ 0, 1 \}, P, S \rangle$, con P insieme delle produzioni contenente:

$$\begin{aligned} &S \to AB \\ &A \to \epsilon \mid 0A11 \\ &B \to \epsilon \mid 000B1111 \end{aligned}$$

Dimostrazione. Lasciato per esercizio. □

6.1.2 Alberi di derivazione

Definizione 6.6 (Albero di derivazione)**.** *Gli* alberi di derivazione *(o* parse tree*, alberi di parsing) vengono utilizzati per rappresentare graficamente le derivazioni generate da una grammatica.*

1. *ogni vertica ha un'etichetta presa tra* $V \cup T \cup \{ \epsilon \}$*;*
2. *la radice deve essere etichettata con una variabile* V *(non restringiamo la radice ad essere per forza il simbolo iniziale* S*);*
3. *i nodi interni (nodi non foglia) devono essere etichettati con variabili;*
4. *se un vertice è etichettato con* A *e* $n_1, \ldots, n_k$ *sono i figli di* A *etichettati con* $X_1, \ldots, X_k$*, allora* $A \to X_1 \cdots X_k \in P$*.*
5. *se un nodo è etichettato con* ϵ *allora tale nodo è una foglia ed è l'unico figlio di suo padre.*

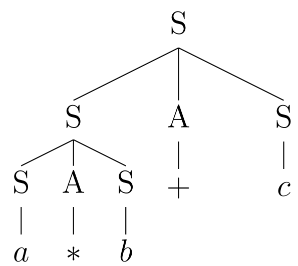
Sia T *un albero di derivazione.* T descrive *la stringa* $\alpha \in (V \cup T)^*$ *se è possibile "ricostruire"* α *leggendo le foglie dell'albero da sinistra a destra ovvero,* α giace *sulle foglie di* T*. Notare che non stiamo restringendo le foglie ad essere per forza etichettate con terminali, ammettiamo quindi la presenza di alberi di derivazione che hanno almeno una foglia etichettata con una variabile e questi sono detti alberi di derivazione* parziale*.*

Una caratteristica degli alberi di derivazione è che nascondono dettagli inutili introdotti dalle derivazioni, in particolare, preso un albero di derivazione è possibile trovare più derivazioni corrispondenti tra loro equivalenti.

Esempio. Sia $G = \langle V = \{S, A\}, T = \{a, b, c, +, *\}, P, S\rangle$, con insieme delle produzioni P composto da:

$$\begin{aligned} &S \to a \mid b \mid c \mid SAS \\ &A \to + \mid * \end{aligned}$$

Sia $x = a * b + c$. Un albero di derivazione che descrive x è il seguente:



Due derivazioni equivalenti:

$$S \overset{G}{\Rightarrow} SAS \overset{G}{\Rightarrow} SASAS \overset{G}{\Rightarrow} aASAS \overset{G}{\Rightarrow} a * SAS \cdots$$

$$S \overset{G}{\Rightarrow} SAS \overset{G}{\Rightarrow} SASAS \overset{G}{\Rightarrow} S * SAS \overset{G}{\Rightarrow} a * SAS \cdots$$

Teorema 6.1. *Sia $G = \langle V, T, S, P\rangle$ una grammatica CF. Sia $\alpha \in (V \cup T)^*$. Allora $S \overset{G}{\Rightarrow}_* \alpha$ se e solo se esiste un albero di derivazione con radice S che descrive α.*

Osservazione 6.1 (Numero nodi interni corrisponde al numero di derivazioni)**.** *Notare che, dato un albero di derivazione, il numero dei suoi nodi interni è pari al numero di derivazioni che sono state applicate per derivare la stringa $\alpha \in (V \cup T)^*$ che giace sulle foglie.*

Dimostrazione. Vogliamo dimostrare che: dato un albero di derivazione arbitrario radicato in A, con n nodi interni e che descrive una $\alpha \in (V \cup T)^*$, esiste una derivazione corrispondete $A \overset{G}{\Rightarrow}_n \alpha, n \in \mathbb{N}$. Procediamo per induzione matematica sul numero di nodi interni n dell'albero, con caso base $n = 0$ e passo induttivo $n = m + 1$, con $m \in \mathbb{N}$; pertanto, nel passo induttivo potremo sfruttare l'ipotesi induttiva.

Caso base: se $n = 0$ abbiamo solo un nodo A che è radice e foglia, quindi un albero con 1 nodo interno e la stringa che giace sull'unica foglia A è proprio α. La derivazione corrispondente è $A \overset{G}{\Rightarrow}_0 A = \alpha$.✓

Passo induttivo: se $n = m+1$ abbiamo un albero T radicato in una variabile A con $m+1$ nodi interni che descrive la una stringa $\alpha \in (V \cup T)^*$. Selezioniamo una la foglia X di T e individuiamo il nodo padre padre che chiameremo B (nodo interno di T). Ora ricaviamo un albero T' che è dato da T privato da tutti i figli del nodo B precedentemente individuato (X potrebbe non essere l'unico figlio del nodo B). Essendo B un nodo foglia all'interno dell'albero T', T' ha m nodi interni e per ipotesi induttiva vale che $A \overset{G}{\Rightarrow}_m \beta B \gamma$. Per la regola (4) di costruzione degli alberi, l'esistenza del sottoalbero radicato in B con figli $X_i \cdots X_n$ implica l'esistenza di $B \to X_1 \cdots X_n \in P$ ma allora vale anche che $A \overset{G}{\Rightarrow}_m \beta B \gamma \overset{G}{\Rightarrow} \beta X_1 \cdots X_n \gamma = \alpha$. ✓

□

Definizione 6.7 (Derivazione sinistra)**.** *Una derivazione è sinistra o leftmost se ad ogni passo di derivazione la produzione viene appilcata alla variabile che sta più a sinistra.*

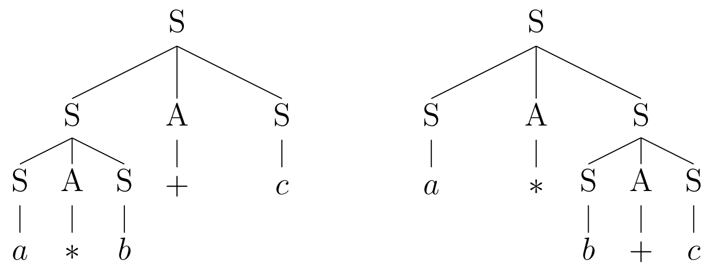
Definizione 6.8 (Derivazione destra)**.** *Una derivazione è destra o rightmost se ad ogni passo di derivazione la produzione viene appilcata alla variabile che sta più a destra.*

Definizione 6.9 (Grammatica ambigua)**.** *Una grammatica CF è ambigua quando ammette più alberi di derivazione che descrivono la stessa stringa.*

Esempio. Sia $G = \langle V = \{ S, A \}, T = \{ a, b, c, +, * \}, P, S \rangle$, con insieme delle produzioni P composto da:

$$S \to a \mid b \mid c \mid SAS$$
$$A \to + \mid *$$

Sia $x = a * b + c \in L(G)$. x è descritta da due alberi di derivazione:



Con una grammatica ambigua, a partire dalla stessa stringa è possibile costruire due alberi diversi a seconda della derivazione utilizzata e alberi differenti potrebbero associare significati differenti alla stessa stringa creando ambiguità appunto, come succede nell'esempio in cui vengono alterate le precedenze tra operazioni $(a * b) + c$, $a * (b + c)$.

Osservazione 6.2 (Indecidibilità dell'ambiguità)**.** *Determinare l'ambiguità di una grammatica è un problema* indecidibile*: non esiste e non esisterà mai un algoritmo che in tempo finito dice se una grammatica arbitraria G è ambigua oppure no.*

Notare che: è possibile dimostrare che una grammatica G non è ambigua (mostrando che ad ogni passo di derivazione si è forzati nella scelta della variabile su cui applicare la produzione), non è possibile farlo algoritmicamente per ogni G.

6.2 Forme Normali

6.2.1 Eliminazione variabili improduttive

Definizione 6.10 (Variabile produttiva)**.** *Sia $G = \langle V, T, P, S\rangle$ una grammatica CF. Una variabile $A \in V$ è* produttiva *se conduce ad una stringa di terminali, ovvero*

$$\exists A \overset{G}{\Rightarrow}_* x \in T^*.$$

Lemma 6.1. *Sia $G = \langle V, T, P, S\rangle$ una grammatica CF tale che $L(G) \neq \varnothing$. Si può calcolare una grammatica equivalente $G' = \langle V', T, P', S\rangle$ tale che per ogni $A \in V'$ esiste $x \in T^*$ tale che $A \overset{G'}{\Rightarrow}_* x$.*

Sia $G = \langle V, T, P, S\rangle$ una grammatica CF, l'obiettivo quindi è di ricavare una grammatica equivalente $G' = \langle V', T, P', S\rangle$ con un insieme V' contenente tutte e sole le variabili produttive ed un insieme P' formato da produzioni che coinvolgono solo variabili che stanno in V'. L'operatore Γ applicato a V ci aiuta a ricavare V'.

Definizione 6.11 (Operatore Γ per variabili produttive)**.** *Sia $W \subseteq V$. L'operatore $\Gamma : \wp(V) \mapsto \wp(V)$ è definito come:*

$$\Gamma(W) := \{\, A \in V \mid \exists \alpha \in (W \cup T)^* \,.\, (A \to \alpha) \in P \,\}. \tag{30}$$

L'insieme $W \subseteq V$ contiene ad ogni applicazione dell'operatore Γ le variabili la cui produttività è stata accertata nelle iterazioni precedenti; proprio per questo l'argomento di Γ è $\varnothing$, ovvero inizialmente di nessuna variabile è nota la produttività. Quindi è interessante applicare iterativamente l'operatore Γ e vedere che cosa fa.

Definizione 6.12 (Applicazione iterata di Γ)**.** *L'applicazione iterata di Γ è definita come:*

$$\Gamma^n(W) := \begin{cases} S, & \textit{se } n = 0; \\ \Gamma\big(\Gamma^{n-1}(W)\big), & \textit{altrimenti.} \end{cases}$$

Osservazione 6.3. *Γ è monotono (e anche estensivo) e l'insieme V è finito quindi applicando un numero di volte $n \le |V|$ l'operatore $\Gamma(\varnothing)$ (nel caso pessimo, ad ogni applicazione troviamo una sola variabile produttiva) è possibile trovare gli insiemi cercati V', P':*

- *$V' = \Gamma^{|V|}(\varnothing)$ (teniamo solo le variabili produttive);*
- *$P' = \{\, A \to \alpha \mid A \in V', \alpha \in (V' \cup T)^* \,\}$ (teniamo solo le produzioni che coinvolgono variabili produttive e terminali).*

Notare che la restrizione $L \neq \varnothing$ è dovuta al fatto che $L(G) = \varnothing$, S stesso risulterebbe improduttivo e verrebbe eliminato. Nel caso $L(G) = \varnothing$ allora descriveremo la grammatica come: $G = \langle \{\, S \,\}, \varnothing, \varnothing, S \rangle$.

Esempio. Sia $G = \langle V = \{\, S, A, B \,\}, T = \{\, 0, 1 \,\}, P, S \rangle$ con insieme P composto da:

$$\begin{aligned} S &\to 0A1 \mid 01B \\ A &\to 0A1 \\ B &\to 01 \mid 01B \end{aligned}$$

$$\Gamma^0(\varnothing) = \varnothing, \Gamma(\varnothing) = \{ B \}, \Gamma^2(\varnothing) = \{ S, B \}, \Gamma^3(\varnothing) = \{ S, B \}.$$

La grammatica dopo la trasformazione avrà $V' = \Gamma^3(\varnothing) = \{ S, B \}$, e P' composto da:

$$\begin{aligned} S &\to 01B \\ B &\to 01 \mid 01B \end{aligned}$$

Lemma 6.2. $A \in \Gamma^n(\varnothing)$ *se e solo se esiste un albero di derivazione di altezza minore o uguale ad* n *che ha* A *come radice e descrive una stringa di soli terminali* $x \in T$*.*

Dimostrazione. Dimostriamo ($\Longrightarrow$), ovvero che se $A \in \Gamma^n$ allora esiste l'albero di altezza minore o uguale di n. Mostriamo per induzione matematica semplice suò numero di applicazioni $n \geq 1$ dell'operatore.

Caso base: se $n = 1$, abbiamo che

$$\begin{aligned} \Gamma^n(\varnothing) &= \Gamma(\varnothing) && [n = 1] \\ &= \{ A \in V \mid \exists \alpha \in (T \cup \varnothing)^* \,.\, (A \to \alpha) \in P \} && [(30)] \\ &= \{ A \in V \mid \exists \alpha \in T^* \,.\, (A \to \alpha) \in P \}. \end{aligned}$$

Se $A \in \Gamma(\varnothing)$, significa che esiste una produzione $A \to x$, con $x = a_1 \cdots a_n \in T^*$. Quindi è possibile costruire un albero di derivazione che descrive x di altezza $1 \leq n$ perché formato da A come radice che ha come figli i nodi etichettati con $a_1, \ldots, a_n$ se e che quindi sono anche foglie. ✓

Passo induttivo: se $n = m + 1$ abbiamo che

$$\begin{aligned} \Gamma^n(\varnothing) &= \Gamma^{m+1}(\varnothing) && [n = m + 1] \\ &= \Gamma\big(\Gamma^m(\varnothing)\big) && [\text{def. di } \cdot^n] \\ &= \{ A \in V \mid \exists \alpha \in (T \cup \Gamma^m(\varnothing))^* \,.\, (A \to \alpha) \in P \} && [(30)]. \end{aligned}$$

Se $A \to \alpha$ secondo la regola (4) di costruzione degli alberi di derivazione, esiste un albero corrispondente T_A con radice in A che ha come figli $X_1, \cdots X_n = \alpha \in \big(T \cup \Gamma^m(\varnothing)\big)^*$ con $n \geq 1$. Essendo α composto da un certo numero di terminali e da un certo numero $k \leq n$ di variabili $B_1, \ldots B_k$, per queste ultime, visto la loro appartenenza in $\Gamma^m(\varnothing)$ è possibile applicare l'ipotesi induttiva e ottenere i corrispondenti alberi di derivazione $T_{B_1}, \dots, T_{B_k}$ la cui altezza (sempre per ipotesi induttiva) è minore o uguale a m. A questo punto, facendo il merge delle radici dei $T_{B_1}, \dots, T_{B_k}$ con le foglie $B_1, \dots, B_k$ dell'albero T_A si ottene un albero T' la cui altezza è minore o uguale a $m + 1$. ✓

Analogamente si dimostra anche ($\Longleftarrow$) (lasciato per esercizio). $\square$

6.2.2 Eliminazione simboli irraggiungibili

Definizione 6.13 (Simbolo raggiungibile)**.** *Sia $G = \langle V, T, P, S\rangle$ una grammatica CF. Siano $\alpha, \beta \in (V \cup T)^*$. Sia $X \in (V \cup T)$ una variabile o simbolo terminale. Il simbolo X è* raggiungibile *se esiste una derivazione di S in cui compare X, ovvero*

$$\exists S \overset{G}{\Rightarrow}_* \alpha X \beta.$$

Lemma 6.3. *Sia $G = \langle V, T, P, S\rangle$ una grammatica CF. Si può calcolare una grammatica equivalente $G' = \langle V', T', P', S\rangle$ tale che per ogni simbolo $X \in V' \cup T$ esistono $\alpha, \beta \in (V' \cup T')^*$ per cui $S \overset{G'}{\Rightarrow}_* \alpha X \beta$.*

Sia $G = \langle V, T, P, S\rangle$ una grammatica CF, l'obiettivo quindi è di ricavare una grammatica equivalente $G' = \langle V', T', P', S\rangle$ nella quale gli insiemi V', T' contengono soli simboli raggiungibili da S e l'insieme P' formato dalle sole produzioni che coinvolgono i soli simboli in V', T'. Una nuova definizione dell'operatore Γ ci aiuta a fare questo.

Definizione 6.14 (Operatore Γ per simboli raggiungibili)**.** *Sia $W \subseteq (V \cup T)$. L'operatore $\Gamma : \wp(V \cup T) \mapsto \wp(V \cup T)$ è definito come:*

$$\Gamma(W) := \{ X \in V \cup T \mid \exists A \in W \,.\, (A \to \alpha X \beta) \in P \} \cup \{ S \}. \tag{31}$$

L'insieme $W \subseteq V \cup T$ contiene ad ogni applicazione dell'operatore Γ le variabili e i simboli terminali la cui raggiungibilità a partire da S è stata accertata nelle applicazioni precedenti. S è per definizione raggiungibile da se stesso.

Osservazione 6.4. *Γ è monotono e gli insiemi V, T sono finiti quindi applicando un numero di volte $n \leq |V| + |T|$ l'operatore $\Gamma(S)$ è possibile trovare gli insiemi cercati V', T', P':*

- *$V' = \Gamma^{|V|+|T|}\big(\{ S \}\big) \cap V$ (teniamo solo le variabili raggiungibili da S);*

- $T' = \Gamma^{|V|+|T|}\big(\{ S \}\big) \cap T$ *(teniamo solo i terminali raggiungibili da S);*
- $P' = \{ A \to \alpha \mid A \in V', \alpha \in (V' \cup T')^* \}$ *(teniamo solo le produzioni che coinvolgono simboli raggiungibili).*

Esempio. Sia $G = \langle V = \{ S, A, B \}, T = \{ 0, 1, 2 \}, P, S \rangle$ con insieme P composto da:

$$\begin{aligned} S &\to \epsilon \mid 0A \\ A &\to 1 \mid A1 \\ B &\to \epsilon \mid 01 \mid 01B \end{aligned}$$

$$\Gamma^0(\varnothing) = \varnothing, \Gamma(\varnothing) = \{ S \}, \Gamma^2(\varnothing) = \{ S, A, 0 \}, \Gamma^3(\varnothing) = \{ S, A, 0, 1 \}.$$

La grammatica dopo la trasformazione avrà $V' = \{ S, A \}, T' = \{ 0, 1 \}$ e P' composto da:

$$\begin{aligned} S &\to \epsilon \mid 0A. \\ A &\to 1 \mid A1. \end{aligned}$$

Lemma 6.4. *$X \in \Gamma^n\big(\{ S \}\big)$ se e solo se esiste un albero di derivazione parziale con radice S in cui compare X come foglia di altezza minore o uguale ad n.*

Dimostrazione. Dimostriamo ($\Longrightarrow$) per induzione matematica semplice sul numero $n \in \mathbb{N}$ di applicazioni dell'operatore.

Caso base: se $n = 0$ abbiamo

$$\begin{aligned} \Gamma^n\big(\{ S \}\big) &= \Gamma^0\big(\{ S \}\big) && [n = 0] \\ &= \{ S \} && [\text{def. di } \cdot^0]. \end{aligned}$$

Se $X \in \Gamma^0\big(\{ S \}\big)$ allora $X = S$ e l'albero parziale corrispondente è formato da $S = X$ come radice e foglia e ha altezza $0 \leq n$. ✓

Passo induttivo: Se $n = m + 1$ abbiamo che

$$\begin{aligned} \Gamma^n\big(\{ S \}\big) &= \Gamma^{m+1}\big(\{ S \}\big) && [n = m + 1] \\ &= \Gamma\big(\Gamma^m\big(\{ S \}\big)\big) && [\text{def. di } \cdot^n] \\ &= \{ X \in V \cup T \mid \exists A \in \Gamma^m\big(\{ S \}\big) \,.\, (A \to \alpha X \beta) \in P \} && [(31)]. \end{aligned}$$

Se $X \in \Gamma^{m+1}(\{ S \})$, per definizione di Γ esiste la produzione $A \to \alpha X \beta$. Per la regola (4) di costruzione degli alberi esiste un albero T_A radicato in A che ha come figli $\alpha X \beta$ che sono anche foglie, quindi T_A ha altezza 1. Per definizione di Γ, $A \in \Gamma^m(\{ S \})$, allora, per ipotesi induttiva A è un nodo foglia di un albero T radicato in S e di altezza minore o uguale a m. Facendo il merge tra T e T_A si ottiene un albero T' radicato in S con una delle sue foglie etichettata con X la cui altezza è minore o uguale a $m + 1$. ✓

La dimostrazione di ($\Longleftarrow$) è analoga. □

Teorema 6.2. *Ogni linguaggio CF non vuoto è generato da una grammatica CF priva di simboli inutili.*

Dimostrazione. Immediata applicando *in ordine* i due lemmi visti. □

Notare che se applicati in ordine errato è possibile ottenere una grammatica che non è del tutto priva di simboli inutili.

Esempio. Troviamo una grammatica per cui c'è un simbolo raggiungibile solo grazie ad una variabile improduttiva. Sia $G = \langle \{ S, A \}, \{ 0, 1 \}, P, S \rangle$ con insieme P composto da:

$$\begin{aligned} &S \to 0 \mid A \\ &A \to A1 \end{aligned}$$

Applicando la rimozione dei simboli irraggiungibili la grammatica G' è equivalente a G perché tutto è raggiungibile. Applicado la rimozione delle variabili improduttive a G', otteniamo $G'' = \langle \{ S \}, \{ 0, 1 \}, P'', S \rangle$, con insieme P'' composto da:

$$S \to 0.$$

(E' rimasto il simbolo terminale non raggiungibile 1).

Simboli inutili: in generale, le variabili improduttive e simboli irraggiungibili vengono anche chiamati *simboli inutili* di una grammatica, quindi i procedimenti visti fino ad ora rientrano nelle procedure di "eliminazione dei simboli inutili" da una grammatica.

6.2.3 Eliminazione delle ϵ-produzioni

Definizione 6.15 (Variabile annullabile)**.** *Una variabile $A \in V$ è annullabile se*

$$\exists A \overset{G}{\Rightarrow}_* \epsilon.$$

Teorema 6.3. *Se $L = L(G)$ per qualche grammatica CF $G = \langle V, T, P, S \rangle$, allora $L \backslash \{ \epsilon \}$ è un linguaggio CF generato da una grammatica $G' = \langle V', T', P', S \rangle$ senza variabili impoduttive, senza simboli irraggiungibili e senza ϵ-produzioni.*

L'obiettivo è quindi quello di trovare un insieme N comosto da tutte e sole le variabili annullabili, da utilizzare per ricavare un insieme di produzioni P'' a partire da P. Per trovare N utilizziamo un nuovo operatore Γ ($N = \Gamma^{|V|}$).

Definizione 6.16 (Operatore Γ per variabili annullabili)**.** *L'operatore $\Gamma : \wp(V) \mapsto \wp(V)$ è definito come:*

$$\Gamma(W) := \{ A \in V \mid \exists A \to \alpha \in P \,.\, \alpha \setminus W = \epsilon \}. \tag{32}$$

L'insieme $W \subseteq V$ contiene ad ogni applicazione dell'operatore Γ le variabili e i simboli terminali la annullabilità è stata accertata nelle applicazioni precedenti. Nella definizione di Γ compare anche $\setminus : \wp(V \cup T) \times \wp(V) \mapsto (V \cup T)^*$ definito induttivamente come:

$$\epsilon \setminus W = \epsilon,$$

$$X\alpha \setminus W = \begin{cases} \alpha \setminus W, & \text{se } X \in W, \\ X(\alpha \setminus W), & \text{se } X \notin W. \end{cases}$$

Osservazione 6.5. *Γ è monotono e gli insiemi V è finito quindi applicando un numero di volte $n \leq |V|$ l'operatore $\Gamma(\varnothing)$ è possibile ricavare l'insieme N formato da tutte sole le variabili annullabili $N = \Gamma^{|V|}(\varnothing)$ e definire P'':*

$$P'' = \left\{ A \to \alpha_1 \dots \alpha_n \;\middle|\; \begin{array}{c} A \to X_1 \dots X_n \in P, \\ X_i \notin N, \alpha_i = X_i, \\ X_i \in N, \alpha_i = X_i \vee \alpha_i = \epsilon, \\ \alpha_1 \dots \alpha_n \neq \epsilon. \end{array} \right\}$$

Da $G = \langle V, T, P, S \rangle$, applicando iterativamente l'operatore Γ abbiamo ottenuto $G'' = \langle V, T, P'', S \rangle$ senza ϵ-produzioni. Nel caso $S \in N$ allora significa che la grammatica originale generava anche la stringa vuota, in questo caso si aggiugnerà la produzione $S \to \epsilon$. Applicando a G'' l'eliminazione dei simboli inutili si ottiene la grammatica desiderata $G' = \langle V', T', P', S \rangle$ senza ϵ-produzioni e senza simboli inutili.

Esempio. Sia $G = \langle V = \{ S, A, B \}, T = \{ 0, 1 \}, P, S \rangle$ con insieme P composto da:

$$\begin{aligned} S &\to A0A \mid A \\ A &\to 1 \mid B \\ B &\to \epsilon. \end{aligned}$$

$$\Gamma^0(\varnothing) = \varnothing, \Gamma(\varnothing) = \{ B \}, \Gamma^2(\varnothing) = \{ A, B \}, \Gamma^3(\varnothing) = \{ S, A, B \} = N.$$

La grammatica $G'' = \langle V, T, P'', S \rangle$ avrà P'' composto da:

$$\begin{aligned} S &\to \epsilon \mid 0 \mid A0 \mid 0A \mid A0A \mid A \\ A &\to 1. \end{aligned}$$

Notare che ora la variabile B è diventata un simbolo irraggiungible.

Lemma 6.5. *Sia $G = \langle V, T, P, S \rangle$ una grammatica CF e sia $G'' = \langle V, T, P'', S \rangle$ ricavata da G rimuovendo le variabili annullabili. $L(G) \setminus \{ \epsilon \} = L(G'')$.*

Dimostrazione. Iniziamo mostrando che $L(G) \subseteq L(G')$ per induzione sul numero $n \in \mathbb{N}$ di ϵ-produzioni utilizzate all'interno di un albero di derivazione radicato in un variabile di G.

Caso base: sia T un albero di derivazione radicato in una variabile di G che utilizza $n = 0$ ϵ-produzioni, e descrive una stringa $\alpha \in (T \cup V)^*$, allora è possibile costruire un albero T' identico perché l'eliminazione delle ϵ-produzioni non ha modificato nessuna delle produzioni coinvolte che quindi sono presenti anche in P'. ✓

Passo induttivo: sia T un albero di derivazione radicato in una variabile di G che utilizza $n = m + 1$ ϵ-produzioni e descrive una stringa $\alpha \in (V \cup T)^*$. Per la regola (5) di costruzione degli alberi, T ha $m + 1$ nodi terminali etichettati con ϵ, identifichiamo uno di questi nodi terminali e il suo nodo padre che chiameremo B e il nodo padre di B che chiameremo A. Per le regole (4) e (5) di costruzioni degli alberi esistono $B \to \epsilon, A \to \alpha B \beta$ rispettivamente. Chiamiamo T_A il sottoalbero di T radicato nel nodo A precedentemente identificato. La rimozione delle ϵ-produzioni sostituisce alla produzioni $B \to \epsilon, A \to \alpha B \beta$ la produzione $A \to \alpha\beta$, quindi esiste un albero T'_A di G' che ha la stessa radice di T_A e descrive la stessa stringa. Se chiamiamo $T \setminus T_A$ l'albero ricavato da

T sostituendovi il sottoalbero T_A con il nodo A, per ipotesi induttiva esiste un albero $T' \setminus T'_A$ che ha stessa radice di $T \setminus T_A$ e descrive la stessa stringa. Unendo i risultati ottenuti fino ad ora (facendo il merge di $T' \setminus T'_A$ con T'_A) si ottiene un albero che ha stessa radice di T e descrive la stessa stringa senza utilizzare ϵ-produzioni. ✓

Mostrare che vale anche viceversa per esercizio. □

6.2.4 Eliminazione Produzioni Unitarie

Definizione 6.17 (Produzione unitaria)**.** *Una produzione della forma*

$$A \to B$$

(a destra compare solo una variabile) è detta produzione unitaria*.*

Teorema 6.4. *Ogni linguaggio CF L tale che $\epsilon \notin L$ è generabile da una grammatica senza simboli inutili, ϵ-produzioni e produzioni unitarie.*

Si comincia da una grammatica G senza ϵ-produzioni, calcoliamo per ogni $A \in V$ il seguente insieme:

$$\text{seguenti}(A) := \{\, B \in V \mid A \overset{G}{\Rightarrow}_* B \,\},$$

togliamo da P le produzioni unitarie e per finire, per ogni variabile $A \in V$, consideriamo le variabili $B \in \text{seguenti}(A)$ e per ogni produzione non unitaria $B \to \beta$ con $B \neq A$, aggiungiamo la produzione $A \to \beta$.

Esempio. Sia $G = \langle V, T, P, S \rangle$ con insieme P composto da:

$$\begin{aligned} &S \to A \\ &A \to 0. \end{aligned}$$

Allora $\text{seguenti}(S) = \{\, S, A \,\}, \text{seguenti}(A) = \{\, A \,\}$. P' è composto da:

$$\begin{aligned} &S \to 0 \\ &A \to 0. \end{aligned}$$

perché $A \in \text{seguenti}(S), A \neq S$ e $A \to 0 \in P$ e non è unitaria. Notare che A è diventato simbolo irraggiungibile.

Dimostriamo che date due grammatiche CF $G = \langle V, T, P, S\rangle$ e $G' = \langle V, T, P', S\rangle$ (ottenuta da G applicando l'eliminazione delle produzioni unitarie) vale che $L(G) = L(G')$. Iniziamo mostrando che $L(G) \subseteq L(G')$ per induzione sul numero $n \in \mathbb{N}$ di produzioni unitarie all'interno di un albero di derivazione per G.

Caso base: sia T un albero radicato in una variabile di G che descrive una stringa α utilizzando $n = 0$ produzioni unitarie, allora è possibile costruire un albero identico T' di G' perché le produzioni non unitarie di P sono anche in P'. ✓

Passo induttivo: sia T un albero di derivazione G che utilizza $n = m + 1$ produzioni unitarie, con $m \in \mathbb{N}$. Identifichiamo, tra gli $m+1$ sottoalberi giustificati dall'utilizzo di $m + 1$ produzioni unitarie della forma $A \to B$, uno di profondità massima che chiameremo T_A. T_A dunque è un albero radicato in A, con un unico figlio B che ha a sua volta $i \in \mathbb{N}$ nodi figli che chiameremo X_i (esiste quindi $B \to X_1 \cdots X_n \in P$ per la regola (4) di costruzione degli alberi). Utilizziamo la notazione $T \setminus T_A$ per indicare un albero ottenuto da T sostituendovi il sottoalbero T_A con la sola radice A (quindi A è foglia di $T \setminus T_A$). Dal momento che $A \to B$ è produzione unitaria, in P' è stata sostituita dalla produzione $A \to X_i \cdots X_n$ (perché esiste la produzione $B \to X_1 \cdots X_n$) quindi è possibile costruire un albero T'_A di G' radicato in A che descrive la stessa stringa descritta da T_A. Per ipotesi induttiva è possibile anche costruire un albero $T' \setminus T'_A$ di G' che descrive la stessa stringa descritta da $T \setminus T_A$. Unendo questi ultimi due risultati, è possibile costruire un albero T' che ha la stessa radice di T e descrive la stessa stringa senza utilizzare produzioni unitarie. ✓

Mostrare che vale anche viceversa per esercizio ($L(G') \subseteq L(G)$).

Recap su semplificazione delle grammatiche: data una grammtica G, è possibile ottenere una grammatica G' priva di simboli inutili, ϵ-produzioni e produzioni unitarie eseguendo in ordine i seguenti passi:

1. eliminazione ϵ-produzioni;
2. eliminazione produzioni unitarie;
3. eliminazione variabili improduttive;

4. eliminazione simboli irraggiungibili.

Esempio: sia $G = \langle \{ A, B, C, D, S \}, \{ a, b, c \}, P, S \rangle$ una grammatica CF, con insieme P costituito da:

$$\begin{aligned} &S \to ASA \mid aB \\ &A \to B \mid S \\ &B \to b \mid \epsilon \\ &C \to CD. \end{aligned}$$

Eliminazione ϵ-produzioni:

$$\Gamma(\varnothing) = \{ B \}, \Gamma^2(\varnothing) = \{ A, B \}, \Gamma^3\{ A, B \} = \Gamma^4 = N.$$

$G_1 = \langle \{ A, B, C, D, S \}, \{ a, b, c \}, P_1, S \rangle$, con P_1 composto da:

$$\begin{aligned} &S \to ASA \mid SA \mid AS \mid S \mid aB \mid a \\ &A \to B \mid S \\ &B \to b \\ &C \to CD. \end{aligned}$$

Eliminazione produzioni unitarie:

$$\begin{aligned} &\text{seguenti}(A) = \{ A, B, S \} \\ &\text{seguenti}(B) = \{ B \} \\ &\text{seguenti}(C) = \{ C \} \\ &\text{seguenti}(S) = \{ S \}. \end{aligned}$$

$G_2 = \langle \{ A, B, C, D, S \}, \{ a, b, c \}, P_2, S \rangle$, con P_2 composto da:

$$\begin{aligned} &S \to ASA \mid SA \mid AS \mid aB \mid a \\ &A \to b \mid ASA \mid SA \mid AS \mid aB \mid a \\ &B \to b \\ &C \to CD. \end{aligned}$$

Eliminazione variabili improduttive:

$$\Gamma(\varnothing) = \{ S, A, B \} = \Gamma^2(\varnothing) = \Gamma^{|V|}(\varnothing).$$

$G_3 = \langle \{ A, B, S \}, \{ a, b, c \}, P_3, S \rangle$, con P_3 composto da:

$$\begin{aligned} &S \to ASA \mid SA \mid AS \mid aB \mid a \\ &A \to b \mid ASA \mid SA \mid AS \mid aB \mid a \\ &B \to b. \end{aligned}$$

Eliminazione simboli irraggiungibili:

$$\Gamma(\{ S \}) = \{ A, B, S, a \}, \Gamma^2(\{ S \}) = \{ A, B, S, a, b \} = \Gamma^{|V|+|T|}(\{ S \}).$$

$G_4 = \langle \{ A, B, S \}, \{ a, b \}, P_4 = P_3, S \rangle$.

6.2.5 Forma Normale di Chomsky

Il linguista Noam Chomsky è l'inventore delle grammatiche.

Teorema 6.5 (Chomsky)**.** *Ogni linguaggio CF L tale che $\epsilon \notin L$ è generato da una grammatica in cui tutte le produzioni sono della forma $A \to BC$ e $A \to a$.*

Nel caso in cui $\epsilon \in L$ allora aggiungeremo la produzione $S \to \epsilon$. A partire da una grammatica senza simboli inutili, ϵ-produzioni e produzioni unitarie il processo per portare la grammatica in forma normale di Chomsky è il seguente: visto che non ci sono ϵ-produzioni, tutte le produzioni di G sono della forma $A \to X_1 \cdots X_m \in P$, quindi è possibile applicare iterativamente i seguenti passi fino al raggiungimento di un punto fisso e ottenere G':

1. se $m = 1$ allora $A \to a$ è in forma normale di Chomsky perché non ci sono produzioni unitarie in G;

2. se $m = 2$ e $X_1, X_2 \in V$ allora siamo in forma normale di Chomsky;

3. se $m \geq 2$ e qualcuno degli $X_i \in T$, allora per ogni $X_i \in T$ introduciamo in V una nuova variabile B_i e aggiungiamo la produzione $B_i \to X_i$ e rimpiazziamo la produzione $A \to X_i \cdots X_m$ con $A \to Y_i \cdots Y_m$, dove $Y_i = X_i$ se $X_i \notin T$ e $Y_i = B_i$ altrimenti;

4. se $m > 2$ e tutti gli $X_i \in V$ allora rimpiazziamo la produzione $A \to X_i \cdots X_m$ con le produzioni $A \to BX_3 \cdots X_m$, $B \to X_1X_2$, con B nuova variabile da aggiungere a V.

Esempio: Sia $G = \langle \{ A, B, S \}, \{ a, b \}, P, S \rangle$, con P composto da:

$$\begin{aligned}
&S \to ASA \mid SA \mid AS \mid aB \mid a\\
&A \to b \mid ASA \mid SA \mid AS \mid aB \mid a\\
&B \to b.
\end{aligned}$$

Applichiamo le regole appena viste:

$$\begin{aligned}
&C \to a\\
&S \to ASA \mid SA \mid AS \mid CB \mid a\\
&A \to b \mid ASA \mid SA \mid AS \mid CB \mid a\\
&B \to b.
\end{aligned}$$

$$\begin{aligned}
&D \to AS\\
&C \to a\\
&S \to DA \mid SA \mid AS \mid CB \mid a\\
&A \to b \mid DA \mid SA \mid AS \mid CB \mid a\\
&B \to b.
\end{aligned}$$

Quindi $G' = \langle \{ A, B, C, D, S \}, \{ a, b \}, P', S \rangle$ con P' composto da:

$$\begin{aligned}
&S \to DA \mid SA \mid AS \mid CB \mid a\\
&A \to b \mid DA \mid SA \mid AS \mid CB \mid a\\
&B \to b\\
&C \to a\\
&D \to AS.
\end{aligned}$$

Lemma 6.6. *Sia G una grammatica CF senza simboli inutili, ϵ-produzioni e produzioni unitarie e sia G' la grammatica ricavata portando G in forma di Chomsky, allora $L(G) = L(G')$.*

Dimostrazione. Iniziamo mostrando che $L(G) \subseteq L(G')$. Procederemo per induzione sull'altezza degli alberi di derivazione di G, ovvero: dato un albero di derivazione radicato in una variabile A di G, che descrive una stringa x e ha altezza $n \geq 1$, esiste un albero di derivazione T' di G' di altezza $\geq n$ che descrive la stessa stringa x.

Caso base: se esiste un albero di derivazione T radicato in una variabile A di G di altezza $n = 1$ che descrive x, allora per la regola (4) di costruzione degli alberi, esiste la produzione $A \to x \in P$.

- Se $|x| = 1$ allora la produzione $A \to x$ è anche in P' (non viene toccata dal processo di trasformazione), quindi esiste anche l'albero corrispondente T' che descrive x di altezza $n = 1$; $\checkmark$
- se $x = a_1 \cdots a_k$, con $k > 1$ allora, secondo il punto 3 della trasformazione, in P' la produzione $A \to x$ viene sostituita con $A \to B_1 \cdots B_k$ e vengono aggiunte i produzioni della forma $B_i \to a_i$, con $1 \leq i \leq k$. Successivamente, se $A \to B_1 \cdots B_k$ è in forma di Chomsky ($k = 2$) allora esiste un labero di derivazione T' radicato in A che descrive x di altezza $n + 1$. Se invece $A \to B_i \cdots B_k$ non è in forma di Chomsky (perché $k > 2$), secondo il punto 4 della trasformazione, la produzione viene sostituita introducendo $j = k - 2$ produzioni della forma $V_0 \to V_1 V_2$; esiste quindi un albero di derivazione T' radicato in A che descrive x di altezza $n + 1 + j$. $\checkmark$

Passo induttivo: se esiste un albero T di derivazione di G radicato in A e di altezza $n = m + 1$ che descrive x, allora la radice A ha dei figli e per la regola (4) di costruzione degli alberi esiste la produzione $A \to \alpha \in P$. Per ogni sottoalbero di T radicato in un nodo figlio di A etichettato con variabile, esiste per ipotesi induttiva un albero di G' che descrive la stessa stringa. Portando in forma di Chomsky anche la produzione $A \to \alpha$ abbiamo tutti gli ingredienti per costruire albero di derivazione di G' che descrive x di altezza maggiore o uguale a $m + 1$. $\checkmark$

Altro verso della dimostrazione lasciato per esercizio ($L(G') \subseteq L(G)$). $\square$

6.2.6 Unfolding

Lemma 6.7. *Sia $G = \langle V, T, P, S \rangle$ una grammatica CF, sia $A \to \alpha B \gamma$ una produzione e $B \to \beta_1 \mid \cdots \mid \beta_n$ l'insieme delle B-produzioni. Sia $G' = \langle V, T, P', S \rangle$ ottenuta da G eliminando le produzioni della forma $A \to \alpha B \gamma$ e aggiungendo le produzioni $A \to \alpha\beta_1\gamma \mid \cdots \mid \alpha\beta_n\gamma$. Allora $L(G) = L(G')$.*

Dimostrazione. Iniziamo mostrando che $L(G) \subseteq L(G')$ per induzione sul numero $n \in \mathbb{N}$ di produzioni della forma $A \to \alpha B \gamma$ utilizzate all'interno di un albero radicato in una variabile A di G che descrive una stringa $\delta \in (V \cup T)^*$.

Caso base: sia T un albero radicato in una variabile di G che utilizza $n = 0$ produzioni della forma $A \to \alpha B \gamma$ e descrive una stringa δ. Dal momento che l'unfolding crea un insieme P' partendo da P modificando le sole produzioni della forma $A \to \alpha B \gamma$ e tali produzioni non vengono utilizzate all'interno di T, allora esiste un albero T' di G' identico a T che descrive δ. ✓

Passo induttivo: sia T un albero di G radicato in una variabile di G che utilizza $n = m + 1$ produzioni della forma $A \to \alpha B \gamma$ e descrive una stringa δ. Tra gli $m+1$ sottoalberi di T giustficati dagli $m+1$ usi della produzione $A \to \alpha B \gamma$, ne identifichiamo uno di profondità massima e lo chiamiamo T_A. Utilizziamo la notazione $T \setminus T_A$ per indicare un albero ottenuto da T rimpiazzandovi T_A con il solo nodo (foglia) A. Dal momento che l'unfolding ha introdotto in P' le produzioni $A \to \alpha \beta_i \gamma$ sostituendole alla produzione $A \to \alpha B \gamma$, abbiamo tutti gli ingredienti per costruire un albero T'_A di G' che ha la stessa radice di T_A e descrive la stessa stringa di T_A. Per ipotesi induttiva, con le produzioni in P' è possibile costruire anche un albero $T' \setminus T'_A$ che ha la stessa radice e descrive la stessa stringa descritta da $T \setminus T_A$. Mettendo insieme questi ultimi due risultati (facendo il merge degli alberi $T' \setminus T'_A$ e T'_A), è possibile costruire un albero T' che descrive la stessa stringa δ descritta da T.

Altro verso della dimostrazione lasciato per esercizio ($L(G') \subseteq L(G)$). $\square$

6.2.7 Eliminazione Ricorsione Sinistra

Lemma 6.8. *Sia $G = \langle V, T, P, S \rangle$ una grammatica CF e sia $A \to A\alpha_1 \mid \cdots \mid A\alpha_m$ l'insieme delle α-produzioni di G per cui A è il simbolo più a sinistra. Siano $A \to \beta_1 \mid \cdots \mid \beta_n$ le rimanenti α-produzioni. G è equivalente a $G' = \langle V \cup \{ B \}, T, P', S \rangle$ dovi si è aggiunta una nuova variabile B a V e le A-produzioni sono rimpazzate con:*

1. *$A \to \beta_i \mid \beta_i B$, con $i \in \{ 1, \ldots n \}$;*
2. *$B \to \alpha_i \mid \alpha_i B$, con $i \in \{ 1, \ldots, m \}$.*

6.2.8 Forma Normale di Greibach

Teorema 6.6. *Ogni linguaggio CF L tale che $\epsilon \notin L$ è generato da una grammatica in cui tutte le produzioni sono della forma $A \to a\alpha$, con $\alpha \in V^*$.*

Notare che una proprietà delle grammatiche in forma di Greibach è che per derivare una stringa lunga n sono necessarie proprio n derivazioni.

Portare una grammatica in forma normale di Greibach:

1. numerare le variabili da 1 a m;
2. al variare di k tra 1 ed m
   (a) cerchiamo le produzioni della forma $A_k \to A_j\alpha$ con $k > j$ e fare l'unfolding di A_j: la produzione ottenuta $A'_k \to A'_j\alpha$ ha $k > j$? Se si, ripeti unfolding;
   (b) eliminiamo la ricorsione a sinistra per le produzioni della forma $A_k \to A_k\alpha$ se presenti;
   (c) se $k < m$ incrementa k e ripeti da (a);
3. fare unfolding ovunque è possibile.

Esempio. Sia $G = \langle \{ A_1, A_2, A_3 \}, \{ a, b \}, P, S \rangle$ con P composto da:

$$\begin{aligned} A_1 &\to A_3A_3 \mid a \\ A_2 &\to A_1A_2 \mid b \\ A_3 &\to A_3A_1 \mid c, \end{aligned}$$

portare G in forma normale di Greibach.

1. Per $k = 1, \nexists A_k \to A_j\alpha \ . \ k > j$ e non c'è ricorsione a sinistra quindi incrementiamo k.
2. Per $k = 2, \exists A_k \to A_j\alpha \ . \ k > j$, ovvero $A_2 \to A_1A_2$ quindi facciamo unfolding di A_1 ottenendo le produzioni: $A_2 \to A_3A_3A_2 \mid aA_2 \mid b$. Non c'è ricorsione a sinistra da eliminare quindi incrementiamo k.

$$\begin{aligned} A_1 &\to A_3A_3 \mid a \\ A_2 &\to A_3A_3A_2 \mid aA_2 \mid b \\ A_3 &\to A_3A_1 \mid c. \end{aligned}$$

3. Per $k = 3, \nexists A_k \to A_j\alpha \;.\; k > j$ però che ricorsione a sinistra da eliminare, quindi la produzione $A_3 \to A_3A_1$ viene sostituita con: $A_3 \to c \mid cB_1$, $B_1 \to A_1 \mid A_1B_1$.

$$\begin{aligned}
&A_1 \to A_3A_3 \mid a \\
&A_2 \to A_3A_3A_2 \mid aA_2 \mid b \\
&A_3 \to c \mid cB_1 \\
&B_1 \to A_1 \mid A_1B_1.
\end{aligned}$$

4. A questo punto siamo fuori dal ciclo e tutte le produzioni di A_k con $k = m$ iniziano con un simbolo terminale. Facciamo nuovamente unfolding della variabile A_k (con $k = m$) ovunque essa compare ed iteriamo all'indietro questo procedimento ogni volta che una nuova variabile inizia con un simbolo terminale:

$$A_1 \to A_3A_3 \text{ diventa } A_1 \to cA_3 \mid cB_1A_3,$$

$$A_2 \to A_3A_3A_2 \text{ diventa } A_2 \to cA_3A_2 \mid cBA_3A_2,$$

$$B_1 \to A_1 \mid A_1B_1 \text{ diventa } B_1 \to cA_3 \mid cBA_3 \mid a \mid cA_3B_1 \mid cBA_3B_1 \mid aB_1.$$

Quindi, alla fine del procedimento, l'insieme delle produzioni è composto da:

$$\begin{aligned}
&A_1 \to cA_3 \mid cB_1A_3 \mid a \\
&A_2 \to cA_3A_2 \mid cBA_3A_2 \mid aA_2 \mid b \\
&A_3 \to c \mid cB_1 \\
&B_1 \to cA_3 \mid cB_1A_3 \mid a \mid cA_3B_1 \mid cBA_3B_1 \mid aB_1.
\end{aligned}$$

Esempio. Ricondurre alle forme normali di Chomsky e Greibach la seguente grammatica:

$$S \to \epsilon \mid 0S1.$$

1. l'unica ϵ-produzione deriva da S e la lasciamo (caso speciale);

2. non ci sono produzioni unitarie;

3. non ci sono variabili improduttive;

4. non ci sono simboli inutili;

5. forma normale di Chomsky: intoduciamo le variabili A, B, C e le produzioni $A \to 0, B \to 1, C \to AS$ (i numeri introdotti ci servono per la forma di Greibach):

$$\begin{aligned}
&S_1 \to \epsilon \mid C_4B_3\\
&A_2 \to 0\\
&B_3 \to 1\\
&C_4 \to A_2S_1.
\end{aligned}$$

6. forma normale di Greibach:

   (a) $k = 1$: facciamo unfolding di C_4 ottenendo $S_1 \to A_2S_1B_3$. Facciamo unfolding di A_2 ottenendo $S \to 0S_1B_3$.

   $$\begin{aligned}
   &S_1 \to \epsilon \mid 0S_1B_3\\
   &A_2 \to 0\\
   &B_3 \to 1\\
   &C_4 \to A_2S_1.
   \end{aligned}$$

   (b) $k = 2$: niente da fare;

   (c) $k = 3$: niente da fare;

   (d) $k = 4$: facciamo unfolding di A_2 ottenendo $C_4 \to 0S_1$.

   $$\begin{aligned}
   &S_1 \to \epsilon \mid 0S_1B_3\\
   &A_2 \to 0\\
   &B_3 \to 1\\
   &C_4 \to 0S_1.
   \end{aligned}$$

Notare che un caso particolare di grammatiche in forma normale di Greibach sono le gramatiche CF in cui le produzioni hanno tutte la forma $A \to a$ oppure $A \to aB$. Tali grammatiche sono dette *lineari destre* ed hanno l'importante caratteristica di generare esattamente i linguaggi regolari.

Altra caratteristica delle grammatiche in forma normale di Greibach è che c'è una corrispondenza 1 a 1 con un automa a pila.

6.3 Automi a Pila

Un *automa a pila* è una macchina costituita da:

1. un controllo che è un automa a stati finiti;
2. un nastro di input che come in nei DFA è di sola lettura;
3. una pila (stack) sulla quale è possibile eseguire le seguenti operazioni:
    - push(α): spinge una stringa α in testa alla pila;
    - $X =$ pop(): preleva il simbolo in testa alla pila;
    - empty(): vero se e solo se la lista è vuota.

    Non ci sono limiti alla dimensione dello stack $\Longrightarrow$ memoria illimitata.

Le azioni possibili di un atuoma a pila sono di 2 tipi differenti:

1. leggere *contemporaneamente* dal nastro e dalla pila (operazione di pop):
    - muovere la testina di una posizione più a destra sul nastro;
    - cambiare stato;
    - inserire una stringa (anche vuota) sullo stack (operazione di push)
2. leggere solo dalla pila (simile ad una ϵ-transizione):
    - cambiare stato;
    - inserire stringa (anche vuota) sullo stack.

Definizione 6.18 (Automa a pila)**.** *Un automa a pila* non deterministico *(APND) è una 7-upla:* $M = \langle Q, \Sigma, R, \delta, q_0, Z_0, F \rangle$*, dove*

- $\boldsymbol{Q}$ *è un insieme (finito) di stati;*
- $\boldsymbol{\Sigma}$ *è un alfabeto (finito) di input;*
- $\boldsymbol{R}$ *è l'alfabeto (finito) della pila;*
- $\boldsymbol{q_0}$ *è lo stato iniziale;*
- $\boldsymbol{Z_0}$ *è il simbolo iniziale sulla pila;*

- $\boldsymbol{F} \subseteq Q$ *è l'inzieme degli stati finali;*
- $\boldsymbol{\delta} : Q \times (\Sigma \cup \{\, \epsilon \,\}) \times R \to \wp_f(Q \times R^*)$ *è la funzione di transizione.*

La funzione di transizione quindi prende in input una tripla formata da:

- stato $q \in Q$;
- simbolo $a \in \Sigma$ (da origine ad un mossa di "tipo 1") oppure ϵ (da origine ad una "mossa di tipo 2");
- simbolo $a \in R$ dal top dello stack.

e restituisce una coppia formata dal nuovo stato $q \in Q$ e una stringa $x \in R^*$ di cui viene falla la push(x) sullo stack. Notare che essendo R^* un insieme infinito, è necessario specificare insieme delle parti *finite* con la notazione $\wp_f(Q \times R^*)$.

Notare che il non determinismo è essenziale per l'accettazione di linguaggi acontestuali.

Definizione 6.19 (Descrizione istantanea di un APND)**.** *La* descrizione istantanea *di un APND è una tripla (q, x, γ), dove*

- $q \in Q$ *è lo stato corrente;*
- $x \in \Sigma^*$ *è ciò che rimane da leggere sul nastro;*
- $\gamma \in R^*$ *è la sequenza dei simboli sullo stack.*

Se $(p, \gamma) \in \delta(q, a, Z)$, un *passo di computazione* dell'APND è definito da $(q, aw, Z\alpha) \mapsto_M (p, w, \gamma\alpha)$, con $a \in \Sigma \cup \{\, \epsilon \,\}$ (per catturare entrambi i tipi di mossa).

Definizione 6.20 (Linguaggio accettato da un APND)**.** *Il linguaggio accettato da un APND si può definire in due diversi modi:*

1. *accettazione per pila vuota:*

$$L_P = \{\, x \in \Sigma^* \mid \exists q \in Q \; . \; (q_0, x, Z_0) \mapsto_M^* (q, \epsilon, \epsilon) \,\};$$

2. *accettazione per stato finale:*

$$L_F = \{\, x \in \Sigma^* \mid \exists q_f \in F \; . \; \exists \gamma \in R^* \; . \; (q_0, x, Z_0) \mapsto_M^* (q_f, \epsilon, \gamma) \,\}.$$

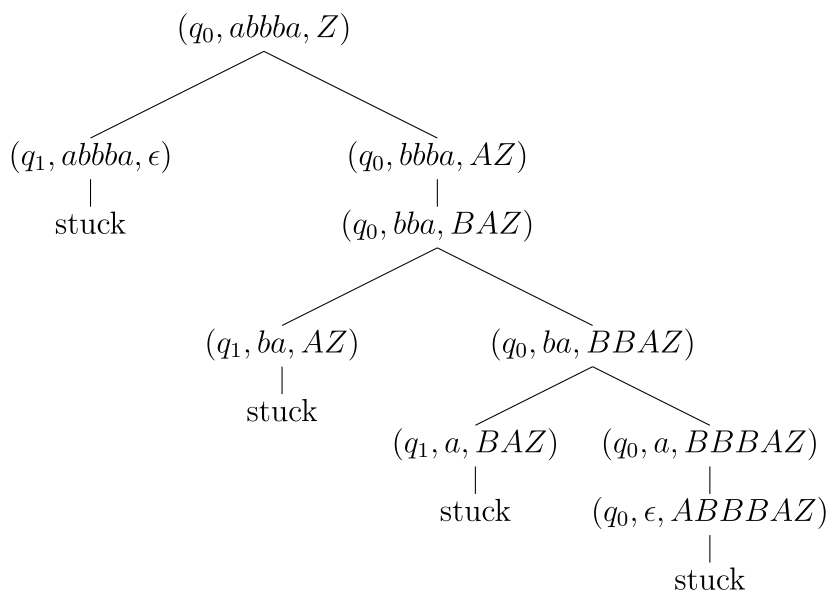
Osservazione 6.6. *Per ogni automa che accetta per stato finale ne esiste uno equivalente che accetta per pilavuota e viceversa. Intuitivamente, un automa che termina per stato finale può essere modificato per svuotare la sua pila una volta raggiunto uno stato finale e un automa che accetta per pila vuota può inserire un simbolo speciale Z_f prima del simbolo Z_0 che quando viene letto porta l'automa in uno stato finale.*

Esempio. L'APND $M = \langle \{ q_0, q_1 \}, \{ a, b \}, \{ Z, A, B \}, \delta, q_0, Z, \varnothing \rangle$, con δ definita da

q_0	$\boldsymbol{\epsilon}$	$\boldsymbol{a}$	$\boldsymbol{b}$
$\boldsymbol{Z}$	q_1, ϵ	q_0, AZ	q_0, BZ
$\boldsymbol{A}$		q_0, AA q_1, ϵ	q_0, BA
$\boldsymbol{B}$		q_0, AB	q_0, BB q_1, ϵ

q_1	$\boldsymbol{\epsilon}$	$\boldsymbol{a}$	$\boldsymbol{b}$
$\boldsymbol{Z}$	q_1, ϵ		
$\boldsymbol{A}$		q_1, ϵ	
$\boldsymbol{B}$			q_1, ϵ

riconosce il linguaggio $L = \left\{ ww^r \mid w \in \{ a, b \}^* \right\}$. Il linguaggio accettato da questo automa coincide con le stringhe palindrome su $\{ a, b \}$ di lunghezza pari, quindi ad esempio la stringa *abbba* non verrebbe accettata:



E' possibile modificare la funzione δ per fare si che tutte le stringhe palindrome vengano accettate:

q_0	$\boldsymbol{\epsilon}$	$\boldsymbol{a}$	$\boldsymbol{b}$
$\boldsymbol{Z}$	q_1, ϵ	q_0, AZ q_1, Z	q_0, BZ q_1, Z
$\boldsymbol{A}$		q_0, AA q_1, ϵ q_1, A	q_0, BA q_1, A
$\boldsymbol{B}$		q_0, AB q_1, B	q_0, BB q_1, ϵ q_1, B

q_1	$\boldsymbol{\epsilon}$	$\boldsymbol{a}$	$\boldsymbol{b}$
$\boldsymbol{Z}$	q_1, ϵ		
$\boldsymbol{A}$		q_1, ϵ	
$\boldsymbol{B}$			q_1, ϵ

Esempio. APND deterministico $M = \langle \{ q_0, q_1 \}, \{ a, b, c \}, \{ Z, A \}, \delta, q_0, Z, \{ q_1 \} \rangle$ che accetta il linguaggio $L = \{ a^n cb^m \mid n, m \in \mathbb{N}, n \neq m \}$ per stato finale:

q_0	$\boldsymbol{\epsilon}$	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$
$\boldsymbol{Z}$		q_0, AZ		q_2, B
$\boldsymbol{A}$		q_0, AA		q_1, A
$\boldsymbol{B}$				

q_1	$\boldsymbol{\epsilon}$	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$
$\boldsymbol{Z}$	q_2, B			
$\boldsymbol{A}$			q_1, ϵ	
$\boldsymbol{B}$			q_1, B	

q_2	$\boldsymbol{\epsilon}$	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$
$\boldsymbol{Z}$				
$\boldsymbol{A}$				
$\boldsymbol{B}$			q_1, B	

Teorema 6.7. *Se $L = L_p(M)$ con M APND, allora $L = L_p(M')$ con M' APND con 1 solo stato.*

Dimostrazione. Intuitivamente, è possibile codificare tutta l'informazione degli stati Q all'interno dei simboli scrivibili sullo stack, ovvero aumentando i simboli di R. $\square$

Teorema 6.8. *Se esiste un APND M con 1 stato tal che $\epsilon \notin L = L_p(M)$, allora L è CF.*

Dimostrazione. Dato un APND con uno stato, troviamo la grammatica corrispondente $G = \langle R, \Sigma, P, Z_0 \rangle)$ utilizzando la seguente regola per costruire l'insieme delle produzioni:

$$P = \{\, Z \to aZ_1 \cdots Z_k \mid (q, \epsilon, Z_1 \cdots Z_k) \in \delta(q, a, Z) \,\}. \tag{33}$$

Dimostriamo per induzione sulla lunghezza $n \in \mathbb{N}$ della stringa che

1. $$(q, x, Z) \to_M^* (q, \epsilon, \epsilon) \Longrightarrow Z \overset{G}{\Rightarrow}_* x;$$

2. $$Z \overset{G}{\Rightarrow}_* x \Longrightarrow (q, x, Z) \to_M^* (q, \epsilon, \epsilon),$$

da cui segue che

$$(q, x, Z) \to_M^* (q, \epsilon, \epsilon) \Longleftrightarrow Z \overset{G}{\Rightarrow}_* x.$$

($\Longrightarrow$) **Caso base:** se $n = 1$ allora x è composta da un solo simbolo a e vale che

$$(q, a, Z) \to_M (q, \epsilon, \epsilon) \Longrightarrow Z \to a \in P \Longrightarrow Z \overset{G}{\Rightarrow}_* a\checkmark \qquad [(33)].$$

Passo induttivo: se $n = m + 1, m \in \mathbb{N}$ allora $x = aw$ e vale che

$$(q, aw, Z) \to_M (q, w, Z_1 \cdots Z_m) \to_M^* (q, \epsilon, \epsilon)$$

allora

$$\begin{aligned} Z &\overset{G}{\Rightarrow} aZ_1 \cdots Z_k && [(33)] \\ &\overset{G}{\Rightarrow}_* w && [\text{ip. ind.}]. \end{aligned}$$

($\Longleftarrow$) **Caso base:** se $n = 1$ allora x è composta da un solo simbolo a e vale che

$$Z \overset{G}{\Rightarrow}_* a \Longrightarrow (q, a, Z) \to_M (q, \epsilon, \epsilon) \qquad [(33)].$$

Passo induttivo: se $n = m + 1, m \in \mathbb{N}$ allora $x = aw$ e vale che

$$Z \overset{G}{\Rightarrow} aZ_1 \cdots Z_k \overset{G}{\Rightarrow}_* w$$

allora

$$\begin{aligned} (q, aw, Z) &\to_M (q, w, Z_1 \cdots Z_m) && [(33)] \\ &\to_M^* (q, \epsilon, \epsilon) && [\text{ip. ind.}]. \end{aligned}$$

$\square$

Nel caso in cui $\epsilon \in L$ allora $(q, \epsilon, Z_0) \to (q, \epsilon, \epsilon) \iff Z_0 \overset{G}{\Rightarrow} \epsilon$ che sappiamo essere vero perché l'eliminazione delle ϵ-produzione ne lascia soltanto una che ha come lhs il simbolo iniziale della grammatica.

Teorema 6.9. *Se $\epsilon \notin L$ ed L è CF, allora esiste un APND tale che $L = L_p(M)$.*

Dimostrazione. Data una grammatica CF $G = \langle V, T, P, S \rangle$ in forma normale di Greibach, ricaviamo un APND a 1 stato utilizzando la seguente regola per la definizione della δ:

$$A \to a\alpha \in P \iff (q, \alpha) \in \delta(q, a, A). \tag{34}$$

Dimostriamo per induzione su numero $n \geq 1$ di passi di derivazione/computazione che:

1. $$xA\beta \overset{G}{\Rightarrow}_* xy\alpha\beta \implies (q, y, A\beta) \to^*_M (q, \epsilon, \alpha\beta), \text{ con } \alpha, \beta \in V^*,$$

2. $$(q, y, A\beta) \to^*_M (q, \epsilon, \alpha\beta) \implies xA\beta \overset{G}{\Rightarrow}_* xy\alpha\beta, \text{ con } \alpha, \beta \in V^*,$$

da cui segue che:

$$xA\beta \overset{G}{\Rightarrow}_* xy\alpha\beta \iff (q, y, A\beta) \to^*_M (q, \epsilon, \alpha\beta), \text{ con } \alpha, \beta \in V^*,$$

e

$$S \overset{G}{\Rightarrow}_* y \iff (q, y, S) \to^m_M (q, \epsilon, \epsilon).$$

($\Longrightarrow$) **Caso base:** $n = 1$,

$$xA\beta \overset{G}{\Rightarrow} xa\alpha\beta \implies A \to a\alpha \in P \iff (q, a, A\beta) \to_M (q, \epsilon, \alpha\beta) \quad [(34)].$$

Passo induttivo: se $n = m + 1$,

$$xA\beta \overset{G}{\Rightarrow} xa\gamma\beta \overset{G}{\Rightarrow}_m xaw\alpha\beta,$$

allora

$$\begin{aligned} (q, aw, A\beta) &\to_M (q, w, \gamma\beta) && [(34)] \\ &\to^m_M (q, \epsilon, \alpha\beta) && [\text{ip. ind.}]. \end{aligned}$$

$$xa\alpha\beta \overset{G}{\Rightarrow}_m xaw\gamma \iff (q, w, \alpha\beta) \to_M^m (q, \epsilon, \gamma), \text{ con } \gamma \in V^* \quad \text{[ip. ind.]}.$$

($\Longleftarrow$) **Caso base:** $n = 1$,

$$(q, a, A\beta) \to_M (q, \epsilon, \alpha\beta) \Longrightarrow A \to a\alpha \in P \iff xA\beta \overset{G}{\Rightarrow} xa\alpha\beta \quad [(34)].$$

Passo induttivo: se $n = m + 1$,

$$(q, aw, A\beta) \to_M (q, w, \gamma\beta) \to_M^m (q, \epsilon, \alpha\beta)$$

allora

$$\begin{aligned} xA\beta &\overset{G}{\Rightarrow} xa\gamma\beta && [(34)] \\ &\overset{G}{\Rightarrow}_m xaw\alpha\beta && \text{[ip. ind.]}. \end{aligned}$$

□

Nel caso in cui $\epsilon \in L$ allora $S \to \epsilon \in P$, basta definire δ in modo che $(q, \epsilon) \in \delta(q, \epsilon, S)$.

Osservazione 6.7. *C'è una corrispondenza 1 a 1 tra una grammatica in forma di Greibach ed un APND con un solo stato.*

Esempio. Sia $G = \langle \{ S, A, B \}, \{ a, b \}, P, S \rangle$ con insieme P definito come segue:

$$\begin{aligned} &S \to Aaab \mid S \\ &A \to Aab \mid A \mid B \\ &B \to a \mid aB. \end{aligned}$$

1. Eliminazione ϵ-produzioni: niente da fare.

2. Eliminazione produzioni unitarie:

$$\text{seguenti}(B) = \{ B \},$$

$$\text{seguenti}(A) = \{ A, B \},$$

$$\text{seguenti}(S) = \{\, S \,\},$$

$$\begin{aligned}
S &\to Aaab \\
A &\to Aab \mid a \mid aB \\
B &\to a \mid aB.
\end{aligned}$$

3. Eliminazione variabili improduttive: niente da fare.

4. Eliminazione simboli irraggiungibili: niente da fare.

5. Forma di Chomsky:

$$\begin{aligned}
S &\to FD \\
A &\to ED \mid a \mid CB \\
B &\to a \mid CB \\
C &\to a \\
D &\to b \\
E &\to AC \\
F &\to EC
\end{aligned}$$

6. Forma di Greibach:

   - S, A, B, C, D ok;
   - $E \to AC$ non ok, facciamo unfolding di A ottenendo:

   $$E \to EDC \mid aC \mid CBC.$$

   Facciamo unfolding di C nella produzione $E \to CBC$ ottenendo $E \to aBC$ ed eliminiamo ricorsione a sinistra nella produzione $E \to EDC$ ottenendo

   $$\begin{aligned}
   E &\to aC \mid aBC \mid aCG \mid aBCG \\
   G &\to DC \mid DCG
   \end{aligned}$$

Insieme P fino ad ora:

$$\begin{aligned}
&S \to FD \\
&A \to ED \mid a \mid CB \\
&B \to a \mid CB \\
&C \to a \\
&D \to b \\
&E \to aC \mid aBC \mid aCG \mid aBCG \\
&F \to EC \\
&G \to DC \mid DCG
\end{aligned}$$

- F non ok, facciamo unfolding ottenendo
$$F \to aCC \mid aBCC \mid aCGC \mid aBCGC;$$
- G non ok, facciamo unfolding ottenendo $G \to bC \mid bCG$;

Insieme P fino ad ora:

$$\begin{aligned}
&S \to FD \\
&A \to ED \mid a \mid CB \\
&B \to a \mid CB \\
&C \to a \\
&D \to b \\
&E \to aC \mid aBC \mid aCG \mid aBCG \\
&F \to aCC \mid aBCC \mid aCGC \mid aBCGC \\
&G \to bC \mid bCG
\end{aligned}$$

Insieme P finale (con unfolding per S, A, B):

$$\begin{aligned}
&S \to aCCD \mid aBCCD \mid aCGCD \mid aBCGCD \\
&A \to aCD \mid aBCD \mid aCGD \mid aBCGD \mid a \mid aB \\
&B \to a \mid aB \\
&C \to a \\
&D \to b \\
&E \to aC \mid aBC \mid aCG \mid aBCG \\
&F \to aCC \mid aBCC \mid aCGC \mid aBCGC \\
&G \to bC \mid bCG
\end{aligned}$$

APND con 1 stato corrispondente:

q	$\boldsymbol{\epsilon}$	$\boldsymbol{a}$	$\boldsymbol{b}$
$\boldsymbol{S}$		q, CCD $q, BCCD$ $q, CGCD$ $q, BCGCD$	
$\boldsymbol{B}$		q, B q, ϵ	
$\boldsymbol{C}$		q, ϵ	
$\boldsymbol{D}$			q, ϵ
$\boldsymbol{G}$			q, C q, CG

Passaggio facoltativo: rinomina dei simboli di R utilizzando espressioni regolari, per capire meglio come lavora l'automa.

q	$\boldsymbol{\epsilon}$	$\boldsymbol{a}$	$\boldsymbol{b}$
$\boldsymbol{S}$		q, AAB q, A^+AAB $q, A(BA)^+AB$ $q, A^+A(BA)^+AB$	
$\boldsymbol{A^+}$		q, A^+ q, ϵ	
$\boldsymbol{A}$		q, ϵ	
$\boldsymbol{B}$			q, ϵ
$\boldsymbol{(BA)^+}$			q, A $q, A(BA)^+$

6.4 Pumping lemma

Esercizio 6.7

$L = \{ 0^a 1^b 0^{ab} \mid a, b \geq 1 \}$ non è CF. Per ogni $n \geq 1$ scegliamo $z = 0^a 1^b 0^{ab}$, con $a = b = n$, quindi $z \in L, |z| \geq n$. Tutti i modi di partizionare z in $uvwxy$ tali che $|vx| \geq 1, |vwx| \leq n$ sono:

1. $uvwx = 0^h, y = 0^{n-h}1^n0^{n^2}$, con $h \leq n$;

2. $u = 0^h, vwx = 0^k 1^m, y = 1^{n-m} 0^{n^2}$, con $k \geq 1, k + m \leq n$;

3. $u = 0^n 1^h, vwx = 1^m, y = 1^{n-m-h} 0^{n^2}$, con $m \leq n$;

4. $u = 0^n 1^h, vwx = 1^k 0^m, y = 0^{n^2 - m}$, con $k + m \leq n$;

5. $u = 0^n 1^n 0^h, vwx = 0^m, y = 0^{n^2 - m - h}$, con $m \leq n$.

Mostriamo che in ciascuna delle casistiche otteniamo una stringa $z' = uv^0 wx^0 y = uwy \notin L$:

1. $z' = 0^{n_1} 1^n 0^{n^2} \notin L$, $[n_1 < n]$;

2. $z' = 0^{n_1} 1^{n_2} 0^{n^2} \notin L$ $[n_1 < n \lor n_2 < n]$;

3. $z' = 0^n 1^{n_1} 0^{n^2} \notin L$ $[n_1 < n]$;

4. $z' = 0^n 1^{n_1} 0^{n_2} \notin L$ $[nn_1 \neq n_2]$;

   Dimostrazione. Siano $1 \leq t_1 + t_2 \leq n$ il numero di '1' e di '0' che sono stati rimossi dopo aver posto $i = 0$ che ha causato la rimozione delle sottostringhe v ed x rispettivamente ($1 \leq t_1 + t_2 \leq n$ è giustificato dal fatto che $|vx| \geq 1$). Quindi, se $n1 = n - t_1, n_2 = n^2 - t_2$, vogliamo dimostrare che $nn1 \neq n_2$ perché nel caso contrario, ovvero $nn_1 = n_2$ la stringa $z' = 0^n 1^{n_1} 1^{n_2}$ ottenuta dopo il pumping sarebbe ancora in L.

   $$\begin{aligned} nn_1 &= n(n - t_1) && [n_1 = n - t_1] \\ &= n^2 - nt_1 \\ &< n^2 - t_2 && [1 \leq t_1 + t_2 \leq n] \\ &\Longrightarrow nn_1 \neq n_2 \\ &\Longrightarrow z' = 0^n 1^{n_1} 0^{n_2} \notin L. \end{aligned}$$

   □

5. $z' = 0^n 1^n 0^{n_1} \notin L$ $[n_1 < n^2]$.

Esercizio 6.8

$L = \{\, 0^p \mid p \text{ è primo} \,\}$ non è CF. Per ogni $n \geq 1$ scegliamo $z = 0^p$, con p primo $\geq n$, quindi $z \in L$ e $|z| \geq n$. Consideriamo tutti i modi di partizionare z in $uvwxy$ tali che $|vx| \geq 1, |vwx| \leq n$ utilizzando $a, b, c, d, e \in \mathbb{N}$, tali che

$$a + b + c + d + e = p, b + d \geq 1, b + c + d \leq n.$$

$$\begin{aligned}
a + ib + c + id + e &= a + c + e + i(b + d) \\
&= \underbrace{a + b + c + d + e}_{p} + i(b + d) - b - d \\
&= \underbrace{a + b + c + d + e}_{p} + (i - 1)(b + d) \\
&= \underbrace{a + b + c + d + e}_{p} + (p + 1 - 1)(b + d) \quad [i = p + 1] \\
&= p + p(b + d),
\end{aligned}$$

quindi $z' = 0^{p+p(bd)} \notin L$.

7 Teoria della Calcolabilità

7.1 Funzioni primitive ricorsive

La classe $\mathcal{P}$ delle funzioni *primitive ricorsive* è la più piccola classe di funzioni (ovvero l'intersezione di tutte le classi di funzioni), contenente:

1. funzioni ricorsive di base;
2. combinatori (composizione e ricorsione primitiva).

Le funzioni primitive ricorsive quindi sono chiuse per composizione e ricorsione primitiva (applicazione dei combinatori a una funzione primitiva ricorsiva genera una nuova funzione che è ancora primitiva ricorsiva).

7.1.1 μ-operatore limitato di minimizzazione

Intuitivamente, è il costrutto `for` del "linguaggio" delle funzioni primitive ricorsive. E' una iterazione sulla varaibile z che parte con il valere 0 e viene incrementata fino ad un limite massimo in cui $z = y - 1$, dunque si tratta di *iterazione determinata*.

Sia f una funzione primitiva ricorsiva $n+1$-aria (con $n+1$ argomenti). Sia $y \in \mathbb{N}$. Definiamo ora l'insieme degli $z < y$ tali che "azzerano" la funzione f quando z è proprio l'$n+1$-esimo argomento di f:

$$Z^f_{z<y}(x_1, \dots x_n) := \{\, z \in \mathbb{N} \mid f(x_1, \dots, x_n, z) = 0, z < y \,\}.$$

Il μ-operatore limitato di minimizzazione è definito come:

$$\mu z < y.\big(f(x_1, \dots, x_n) = 0\big) := \begin{cases} \min Z^f_{z<y}(x_1, \dots, x_n), & \text{se } Z \neq \varnothing; \\ y, & \text{altrimenti.} \end{cases}$$

Quindi se l'insieme Z non è vuoto, il μ-operatore limitato di minimizzazione restituisce il più piccolo elemento di Z, altrimenti restituisce y. Definiamo ora una funzione g di $n+1$ argomenti che "implementa" il μ-operatore limitato di minimizzazione (l'$n+1$-esimo argomento sarà proprio il limite y):

$$g(x_1, \dots, x_n, y) = \sum_{v<y} \Big(\underbrace{\prod_{u \leq v} \operatorname{sg}\big(f(x_1, \dots x_n, u)\big)}_{\text{produttoria di 0 e 1}} \Big)$$

Chiamiamo z il più piccolo $u < y$ che azzera f, abbiamo due casi:

1. z non esiste, quindi

$$\forall u < y : \text{sg}\big(f(x_1, \dots x_n, u)\big) = 1. \tag{35}$$

Ogni produttoria lavora al varaire di u che va da 0 fino a v, con v che a sua volta va da 0 fino a $y - 1$ quindi possiamo concludere che è sempre vero che $u \leq v < y$. Quindi, se vale (35), ogni produttoria eseguirà $\underbrace{1 \cdot 1 \cdot 1 \cdots 1}_{v} = 1$.

La sommatoria invece lavora con l'indice v che va da 0 fino a $y - 1$ quindi fa in totale y somme. y somme di produttorie, ciascuna delle quali da 1 come risultato, da come risultato finale proprio y ed è ciò che volevamo perché non siamo riusciti a trovare $z < y$ che azzera f, quindi restituiamo y che è il limite.

2. z esiste, abbiamo detto che è il più piccolo $u < y$ che azzera f quindi vale che:

$$\forall u < z : \text{sg}\big(f(x_1, \dots x_n, u)\big) = 1,$$
$$\forall u \geq z : \text{sg}\big(f(x_1, \dots x_n, u)\big) = 0.$$

Le produttorie varranno tutte 1 fino a che l'indice u su cui lavorano resta minore di z (ovvero fino a che l'indice v della sommatoria è minore di z). Mentre tutte le produttorie che lavorano con una u che può diventare maggiore o uguale a z daranno come riultato 0 (perché passando da $u = z$ ci saranno prodotti di 0 e 1 contenenti almeno uno 0). La sommatoria quindi, esegue $\underbrace{1 + 1 + \cdots + 1}_{z} + \underbrace{0 + 0 + \cdots + 0}_{y-z}$ che da come risultato finale proprio z che è ciò che si voleva.

Esercizio 7.1

Mostrare che la seguente funzione è primitiva ricorsiva:

$$f(x) = \begin{cases} x, & \text{se } x \text{ è dispari;} \\ x/2, & \text{se } x \text{ è pari.} \end{cases}$$

Soluzione. Scriviamo due funzioni primitive ricorsive $g(x), h(x)$ che sono le funzioni caratteristiche per l'insieme dei numeri pari e dei numeri dispari rispettivamente:

$$g(x) = \overline{\mathrm{sg}}\big(\mathrm{mod}(x, 2)\big),$$
$$h(x) = 1 - g(x).$$

Ora utilizziamole insieme per definire f:

$$f(x) = h(x) \cdot x + g(x) \cdot x/2.$$

Esercizio 7.2

Mostrare che la classe delle funzioni primitive ricorsive è chiusa rispetto alla definizione per casi. Significa che se $f_1, f_2, \ldots, f_m, g_1, g_2, \ldots, g_m$ sono funzioni primitive ricorsive di n variabili tali che per ogni $x_1, \ldots, x_n$ c'è esattamente un i tale per cui $g_i(x_1, \ldots, x_n) = 0$, allora f è definita come segue è primitiva ricorsiva:

$$f(x_1, \ldots, x_n) = \begin{cases} f_1(x_1, \ldots, x_n), & \text{se } g_1(x_1, \ldots, x_n) = 0, \\ f_2(x_1, \ldots, x_n), & \text{se } g_2(x_1, \ldots, x_n) = 0, \\ \vdots & \\ f_m(x_1, \ldots, x_n), & \text{se } g_m(x_1, \ldots, x_n) = 0. \end{cases}$$

Soluzione.

$$\begin{aligned} f(x_1, \ldots x_n) = & \overline{\mathrm{sg}}\big(g_1(x_1, \ldots, x_n)\big) \cdot f_1(x_1, \ldots, x_n) + \\ & \overline{\mathrm{sg}}\big(g_2(x_1, \ldots, x_n)\big) \cdot f_2(x_1, \ldots, x_n) + \\ & \cdots \\ & \overline{\mathrm{sg}}\big(g_m(x_1, \ldots, x_n)\big) \cdot f_m(x_1, \ldots, x_n). \end{aligned}$$

Esercizio 7.3

Dedurre dall'esercizio precedente che f definita come segue è primitiva ricorsiva:

$$f(x) = \begin{cases} x + 7 & \text{se } x \equiv 0 \mod 3; \\ x^2 + 6 & \text{se } x \equiv 1 \mod 3; \\ x^3 + 12 & \text{se } x \equiv 2 \mod 3. \end{cases}$$

Soluzione. Definiamo la funzione primitiva ricorsiva $g(x, y, n)$ che ritorna 1 se x, y sono congrui modulo n e 0 altrimenti:

$$g(x, y, n) = \overline{\mathrm{sg}}\big(|\mathrm{mod}(x, n) - \mathrm{mod}(y, n)|\big).$$

$$f(x) = g(x, 0, 3) \cdot (x + 7) + g(x, 1, 3) \cdot (x^2 + 6) + g(x, 2, 3) \cdot (x^3 + 12).$$

Esercizio 7.4

Scrivere una funzione primitiva ricorsiva che preso $x \in \mathbb{N}$ restituisce il numero dei suoi divisori.

Soluzione. I divisori di 0 sono infiniti ma noi diremo che sono 0. Nella sommatoria c'è il minore stretto perché la i parte da 0, e per definizione $\mathrm{mod}(x, 0) = 0$ quindi se arrivassimo fino a x compreso otterremmo 1 divisore in più.

$$\begin{cases} \mathrm{divn}(0) = 0; \\ \mathrm{divn}(x) = \sum_{i<x} \overline{\mathrm{sg}}\big(\mathrm{mod}(x, i)\big). \end{cases}$$

Esercizio 7.5

Scrivere una funzione primitiva ricorsiva che preso $x \in \mathbb{N}$ restituisce 1 se x è primo e 0 altrimenti.

Soluzione.

$$\begin{cases} \mathrm{isp}(0) = 0; \\ \mathrm{isp}(x) = \overline{\mathrm{sg}}\big(|\mathrm{divn}(x) - 2|\big). \end{cases}$$

Esercizio 7.6

Scrivere una funzione primitiva ricorsiva che presi $x, y \in \mathbb{N}$ restituisce 1 se x è primo e maggiore di y, 0 altrimenti.

Soluzione.

$$\mathrm{pg}(x, y) = \mathrm{isp}(x) \cdot \mathrm{sg}(x - y)$$

Esercizio 7.7

Scrivere una funzione primitiva ricorsiva che preso x restituisce 1 se è un quadrato perfetto e 0 altrimenti.

Soluzione. Utilizziamo una funzione primitiva ricorsiva di appoggio che restituisce la radice quadrata dell'argomento nel caso i cui l'argomento è un quadrato perfetto mentre restituisce il risultato arrotondato in eccesso altrimenti.

$$\mathrm{sqrt}(x) = \mu z < x \,.\, (x - z^2 = 0).$$

Ora possiamo definire una funzione primitiva ricorsiva che individua i quadrati perfetti:

$$\mathrm{isquad}(x) = \overline{\mathrm{sg}}\Big(\big(\mathrm{sqrt}(x) \cdot \mathrm{sqrt}(x)\big) - x\Big).$$

7.1.2 Insiemi Ricorsivi e Ricorsivamente Enumerabili

Definizione 7.1 (Dominio)**.** *Il dominio di una funzione φ è l'insieme degli $x \in \mathbb{N}$ tali per cui la φ è definita, ovvero:*

$$dom(\varphi) = \{\, x \in \mathbb{N} \mid \varphi(x) \downarrow \}.$$

Definizione 7.2 (Range)**.** *Il range di una funzione φ è l'insieme dei valori $y \in \mathbb{N}$ che può assumere φ, ovvero:*

$$range(\varphi) = \{\, y \in \mathbb{N} \mid \exists x \in \mathbb{N} \,.\, \varphi(x) \downarrow, \varphi(x) = y \,\}.$$

Definizione 7.3 (Insieme ricorsivo)**.** *Un insieme $A \subseteq \mathbb{N}$ è* ricorsivo *se esiste una funzione ricorsiva f_A tale che:*

$$f_A(x) = \begin{cases} 1, se\ x \in A; \\ 0, se\ x \notin A. \end{cases}$$

Tale funzione è detta funzione caratteristica*.*

Definizione 7.4 (Insieme ricorsivamente enumerabile)**.** *Un insieme $A \subseteq \mathbb{N}$ è* ricorsivamente enumerabile *se esiste una funzione parziale ricorsiva ψ tale che $A = dom(\psi)$.*

Osservazione 7.1 (Funzione semicaratteristica)**.** *La funzione ψ_A definita come:*

$$\psi_A(x) = \begin{cases} 1, & se\ x \in A; \\ \uparrow, & se\ x \notin A. \end{cases}$$

è detta semicaratteristica *e vale che $A = dom(\psi_A)$.*

Esercizio 7.8

Dimostrare che il seguente insieme è ricorsivo:

$$A = \{ x \in \mathbb{N} \mid \exists n \in \mathbb{N} \,.\, x = 2n + 1 \}.$$

Soluzione. Troviamo la funzione caratteristica f_A e mostriamo che è ricorsiva: definita come:

$$f_A(x) = \mathrm{sg}\big(\mathrm{mod}(x, 2)\big).$$

f_A è primitiva ricorsiva.

Esercizio 7.9

Dire se il seguente insieme è ricorsivo o ricorsivamente enumerabile. $A =$ range(f), con f:

$$\begin{cases} f(0) = n_0, \\ f(n+1) = 2f(n). \end{cases}$$

Soluzione.

1. Se $n_0 = 0$ allora range$(f) = \{ 0 \} = A$, quindi A essendo un insieme finito è ricorsivo.
2. Se $n_0 \neq 0$ allora $\exists n \in \mathbb{N} \,.\, f(n) = 0$. Se $f(n) = 0$ allora $\forall m \geq n$: $f(m) = 0$ quindi A è finito e dunque ricorsivo.

Esercizio 7.10

Dire se il seguente insieme è ricorsivo o ricorsivamente enumerabile. $B =$range(g), con h ricorsiva e g:

$$\begin{cases} g(0) = n_0, \\ g(n+1) = h(n)g(n). \end{cases}$$

Soluzione.

- Se $n_0 = 0$ allora range$(g) = \{ 0 \} = B$, quindi B essendo un insieme finito è ricorsivo.
- Se $n_0 \neq 0$, g è ricorsiva.

– Se $\forall n \in \mathbb{N} : h(n) \neq 0$, allora g è monotona non decrescente quindi per il teorema di Kleene A è ricorsivo.

– Se $\exists n \in \mathbb{N} \ . \ h(n) = 0$, allora $\forall m > n : g(m) = 0$, quindi $\text{range}(g) = B$ è finito e dunque B è ricorsivo.

Esercizio 7.11

Sia f una funzione ricorsiva. Sia A un insieme ricorsivamente enumerabile. Dimostrare che l'immagine inversa di f ed A, ovvero l'insieme $f^{-1}(A) = \{ x \in \mathbb{N} \mid f(x) \in A \}$ è ricorsivamente enumerabile.

Soluzione. Sia ψ_A la funzione semicaratteristica di A. Per decidere l'appartenenza di un elemento x all'insieme $f^{-1}(A)$ bisogna:

1. determinare $f(x)$ che è sempre possibile visto che f è ricorsiva;
2. determinare se $f(x) \in A$ che è semidecidibile dal momento che A è ricorsivamente enumerabile.

L'insieme $f^{-1}(A)$ dunque è ricorsivamente enumerabile e possiamo definire la sua funzione semicaratteristica $\psi_{f^{-1}(A)}$ come segue, utilizzando la funzione semicaratteristica ψ_A dell'insieme A:

$$\psi_{f^{-1}(A)}(x) = \begin{cases} 1, & \text{se } \psi_A\big(f(x)\big) \downarrow; \\ \uparrow, & \text{altrimenti.} \end{cases}$$

Esercizio 7.12

Dire se l'insieme $D = \text{range}(\psi)$ è ricorsivo con ψ:

$$\begin{cases} \psi(0) = n_0, \\ \psi(n+1) = \varphi_n(n)\psi(n) \end{cases}$$

e φ funzione parziale ricorsiva.

Soluzione. ψ è parziale dal momento che è definita in termini della funzione φ.

- Se $n_0 = 0$ allora D contiene il solo elemento 0 e quindi è ricorsivo;
- se $n_0 \neq 0$, perché D sia ricorsivo deve esistere un enumeratore f ricorsivo monotono non decrescente tale che $\text{range}(f) = D = \text{range}(\psi)$ oppure D deve essere un insieme finito. Utilizziamo il dovetailing per definire f. Cerchiamo di capire cosa c'è in D:
  - se $\forall n \in \mathbb{N} : \varphi(n) \downarrow$ abbiamo due casi:
    - $\nexists n \,.\, \varphi(n) = 0$ allora abbiamo f che è monotona non decrescente e quindi D è ricorsivo;
    - $\exists n \,.\, \varphi(n) = 0$ allora $\forall m > n : \psi(m) = 0$ quindi D è finito e dunque ricorsivo;
  - se invece $\exists n \,.\, \varphi(n) \uparrow$, sia n_0 il più piccolo n per cui $\varphi(n) \uparrow$, allora $\forall m > n_0 : \psi(m) \uparrow$ quindi il $\text{range}(\psi) = D$ è finito e D è ricorsivo.

In tutti i casi D è ricorsivo.