Esercizi di Fondamenti di Informatica

Matteo Belenchia Flavio Corradini

4 marzo 2025

Note

- Alcuni esercizi possono avere più di una soluzione;
- La notazione utilizzata può non sempre corrispondere con quella del docente; si consiglia in ogni caso di usare la notazione presentata a lezione;
- Gli esercizi relativi a esami passati sono identificati dalle date.

Indice

Capitolo 1

Macchine di Turing

Esercizio 1.1 (*30/01/2017, 20/07/2017, 16/12/2019*)

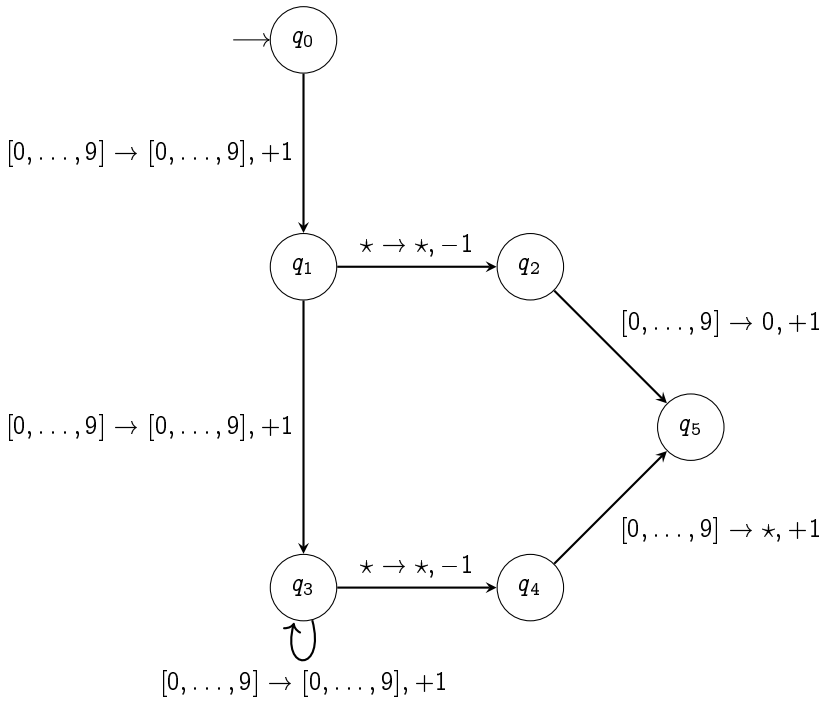
Si scriva un programma per macchina di Turing che, ricevuto sul nastro un numero decimale, lasci sul nastro solo l'intero della divisione del numero di ingresso per 10.

Input	Output atteso
231	23
19	1
7	0

Soluzione 1.1

Scriviamo la matrice di transizione e il grafo di transizione adottando una semplificazione visiva. Per indicare che una regola si applica indistintamente ad ognuna delle cifre decimali nella tabella scriviamo una colonna contrassegnata da $[0, \ldots, 9]$. Inoltre per indicare che una certa transizione lascia inalterate ognuna di queste cifre scriviamo $[0, \ldots, 9]$ nella cella che si trova nella colonna di $[0, \ldots, 9]$. L'idea dietro questa MdT è di verificare innanzitutto di quante cifre si compone il numero in input: se è una sola cifra, allora basta cancellarla e scrivere 0, mentre se è più di una cifra è sufficiente cancellare la cifra meno significativa.

	$[0, \ldots, 9]$	$\star$
q_0	$(q_1, [0, \ldots, 9], +1)$	
q_1	$(q_3, [0, \ldots, 9], +1)$	$(q_2, \star, -1)$
q_2	$(q_5, 0, +1)$	
q_3	$(q_3, [0, \ldots, 9], +1)$	$(q_4, \star, -1)$
q_4	$(q_5, \star, +1)$	
q_5		



Esercizio 1.2 (*06/02/2024*)

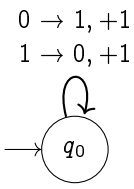
Programmare una MdT che calcoli la funzione logica NOT su una sequenza binaria.

Input	Output atteso
1	0
11011001	00100110
000	111

Soluzione 1.2

Questa MdT deve solamente sostituire ogni 0 con 1 e ogni 1 con 0.

	0	1	$\star$
q_0	$(q_0, 1, +1)$	$(q_0, 0, +1)$	



Esercizio 1.3 (*24/02/2017*)

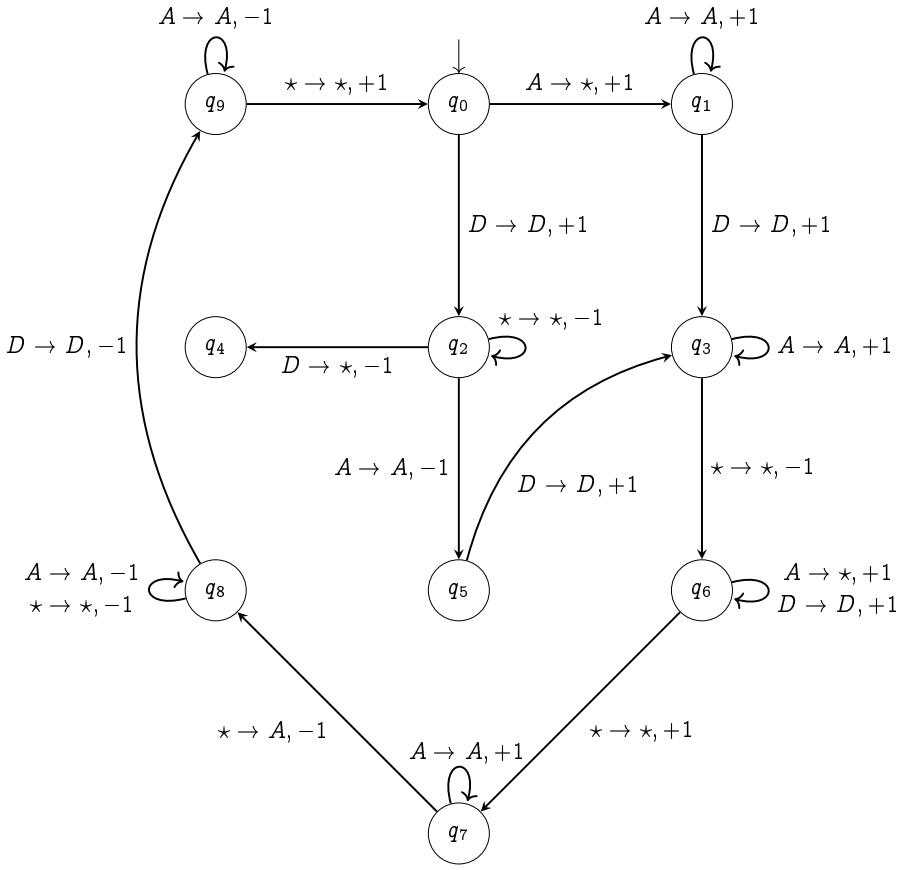
Programmare una macchina di Turing che, dato un nastro iniziale contenente due sequenze di A separate da una D, termina la sua esecuzione lasciando sul nastro la sequenza che contiene il maggior numero di A.

Input	Output atteso
AADA	AA
AADAAA	AAA
AADAA	AA
DA	A

Soluzione 1.3

La MdT deve gestire 3 casi particolari: se c'è solo la D ma nessuna A o se ci sono A solo a sinistra o solo a destra della D deve solamente cancellare la D. Nel caso ci siano A sia a sinistra che a destra si procede cancellando le A a due a due all'inizio e alla fine della stringa di input, ma per ogni coppia di A cancellate si deve anche aggiungere una A ad una sequenza che posizioniamo a destra della stringa di input. Questa sequenza sarà sarà poi l'output della MdT.

	A	D	$\star$
q_0	$(q_1, \star, +1)$	$(q_2, D, +1)$	
q_1	$(q_1, A, +1)$	$(q_3, D, +1)$	
q_2	$(q_5, A, -1)$	$(q_4, \star, -1)$	$(q_2, \star, -1)$
q_3	$(q_3, A, +1)$		$(q_6, \star, -1)$
q_4			
q_5		$(q_3, D, +1)$	
q_6	$(q_6, \star, +1)$	$(q_6, D, +1)$	$(q_7, \star, +1)$
q_7	$(q_7, A, +1)$		$(q_8, A, -1)$
q_8	$(q_8, A, -1)$	$(q_9, D, -1)$	$(q_8, \star, -1)$
q_9	$(q_9, A, -1)$		$(q_0, \star, +1)$



Esercizio 1.4 (*22/06/2017*)

Durante le sue guerre in Gallia, Giulio Cesare usava un semplice cifrario per comunicare con i suoi generali. In questo cifrario, ogni lettera era traslata di un numero fisso di posizioni. Nel cifrario originale, questo numero era 13: si aveva dunque:

in chiaro	ABCDEFGHIJKLMNOPQRSTUVWXYZ
cifrato	NOPQRSTUVWXYZABCDEFGHIJKLM

Si scriva un programma per macchina di Turing che implementi un cifrario di Cesare generalizzato. Il programma prende in ingresso un numero decimale (da 1 a 5) seguito da una stringa su A-Z, e deve lasciare sul nastro la stessa stringa, cifrata traslando ogni lettera del numero indicato di posizioni.

Input	Output atteso
3ABC	DEF
5BACO	GFHT
1BACO	CBDP

Soluzione 1.4

Scriviamo la matrice di transizione e il grafo di transizione adottando una semplificazione visiva. Per indicare che una regola si applica indistintamente ad ognuna delle lettere dell'alfabeto latino nella tabella scriviamo una colonna contrassegnata da $[A, B, \ldots, Z]$. Inoltre scriveremo una regola di transizione della forma

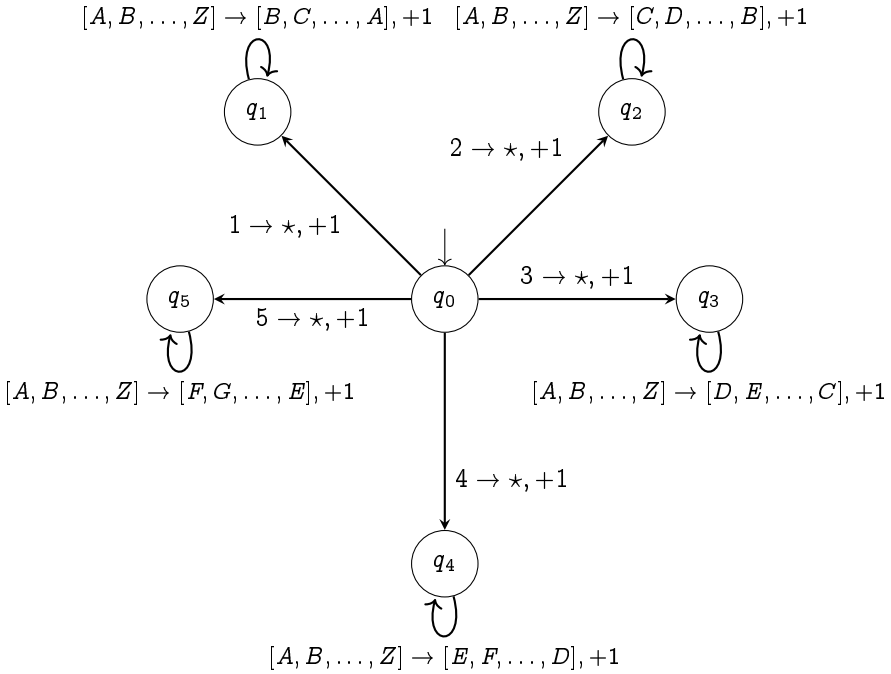
$$(q, [A, B, ..., Z]) \to (q', [B, C, ..., A], x)$$

per denotare un insieme di regole di transizione:

$$\begin{aligned}(q, A) &\to (q', B, x)\\ (q, B) &\to (q', C, x)\\ &\vdots\\ (q, Z) &\to (q', A, x)\end{aligned}$$

L'idea dietro questa MdT è di analizzare innanzitutto il numero decimale che specifica il cifrario, e successivamente cifrare il resto della stringa di input come richiesto.

	$[A, B, \ldots, Z]$	1	2	3	4	5	$\star$
q_0		$(q_1, \star, +1)$	$(q_2, \star, +1)$	$(q_3, \star, +1)$	$(q_4, \star, +1)$	$(q_5, \star, +1)$	
q_1	$(q_1, [B, C, \ldots, A], +1)$						
q_2	$(q_2, [C, D, \ldots, B], +1)$						
q_3	$(q_3, [D, E, \ldots, C], +1)$						
q_4	$(q_4, [E, F, \ldots, D], +1)$						
q_5	$(q_5, [F, G, \ldots, E], +1)$						



Esercizio 1.5

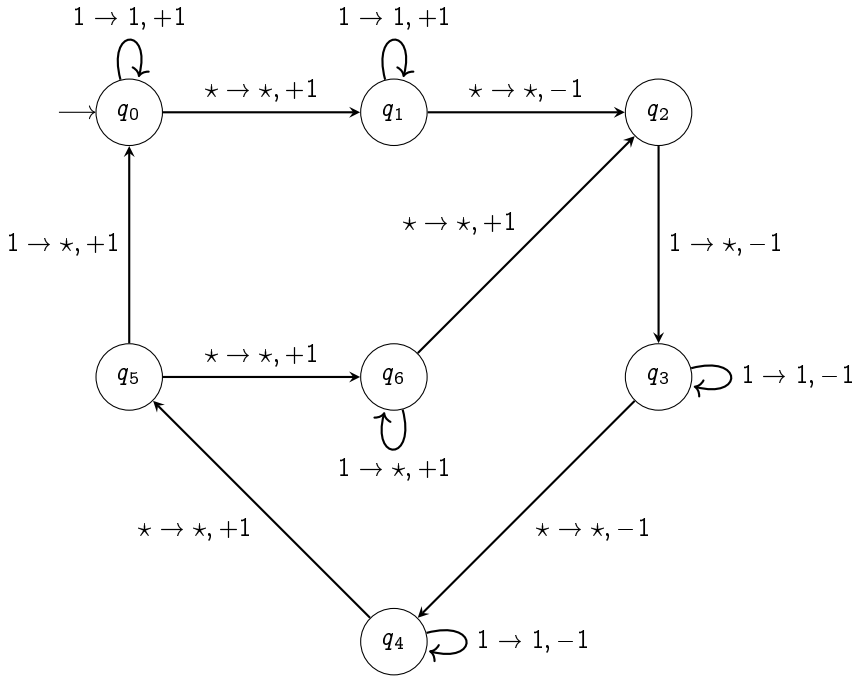
Programmare una macchina di Turing che calcoli $h(x, y) = x \dot{-} y$ (la sottrazione troncata) utilizzando il sistema numerico unario (**NB:** utilizzare la notazione per cui l'output corrisponde al numero di simboli 1 sul nastro, mentre l'input è il numero di simboli 1 meno 1, e.g: zero in input corrisponde a "1", ma in output è il nastro vuoto).

Input	Output atteso
111*11	1
111*111111	*
1111*1	111

Soluzione 1.5

L'idea per svolgere questo esercizio è di ridurre progressivamente l'input, cancellando a due a due i simboli "1" alle estremità dell'input della MdT. Se i simboli "1" a destra dello spazio bianco centrale terminano, il nastro presenta il risultato della sottrazione. Se invece finiscono prima quelli a sinistra, è sufficiente ripulire totalmente il nastro.

	1	$\star$
q_0	$(q_0, 1, +1)$	$(q_1, \star, +1)$
q_1	$(q_1, 1, +1)$	$(q_2, \star, -1)$
q_2	$(q_3, \star, -1)$	
q_3	$(q_3, 1, -1)$	$(q_4, \star, -1)$
q_4	$(q_4, 1, -1)$	$(q_5, \star, +1)$
q_5	$(q_0, \star, +1)$	$(q_6, \star, +1)$
q_6	$(q_6, \star, +1)$	$(q_2, \star, +1)$



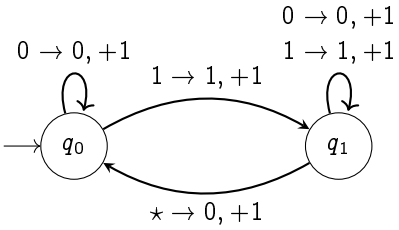
Esercizio 1.6 (*15/07/2021*)

Costruire una macchina di Turing che preso un numero naturale ne restituisce il doppio. Per la rappresentazione dei naturali si scelga la notazione che si ritiene più idonea.

Soluzione 1.6

Per raddoppiare un numero in notazione binaria è sufficiente aggiungere uno 0 a destra del numero, ad eccezione del caso di 0 che in tal caso rimane invariato.

	0	1	$\star$
q_0	$(q_0, 0, +1)$	$(q_1, 1, +1)$	
q_1	$(q_1, 0, +1)$	$(q_1, 1, +1)$	$(q_0, 0, +1)$



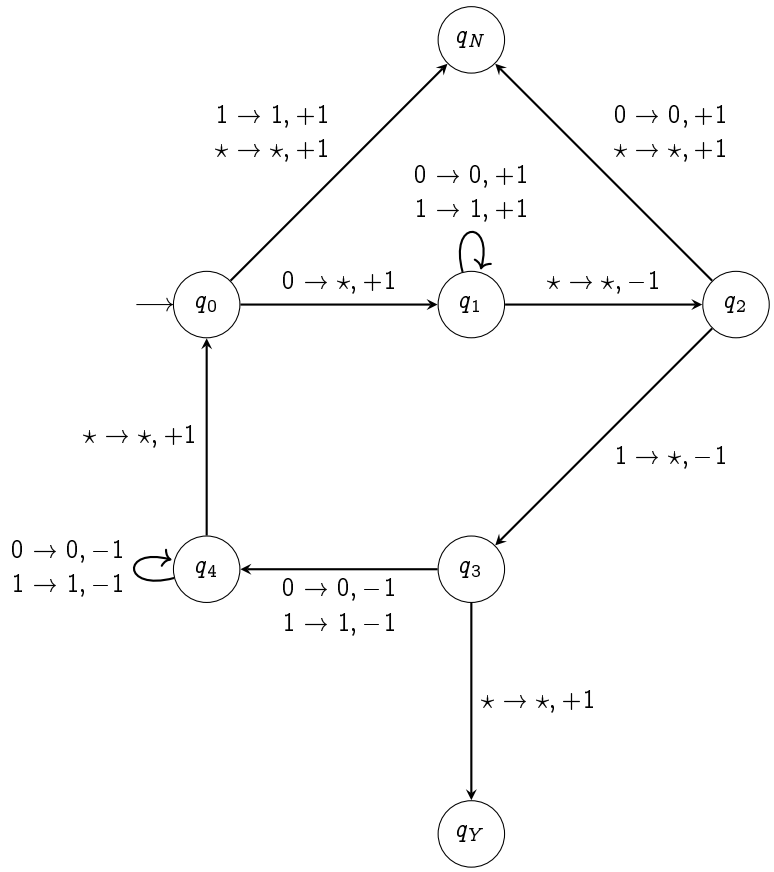
Esercizio 1.7 (*08/09/2023*)

Definire una macchina di Turing che decida il linguaggio $L = \{0^n 1^n \mid n > 0\}$ (utilizzare due stati speciali di arresto q_Y e q_N per l'accettazione e il rifiuto della stringa, senza quindi ripulire il nastro per scrivere Sì/No).

Soluzione 1.7

La macchina di Turing che proponiamo cancella a coppie lo zero in testa alla stringa e l'uno in coda alla stringa, se presenti. Se non c'è uno zero in testa alla stringa da cancellare, o dopo aver cancellato tale zero non c'è un uno in coda alla stringa da cancellare, siamo sicuri che la stringa di input non appartiene al linguaggio e andiamo nello stato di rifiuto. Se dopo aver cancellato uno zero in testa alla stringa e il corrispondente uno in fondo alla stringa ci accorgiamo che la stringa è terminata (ovvero incontriamo un simbolo $\star$ immediatamente a sinistra dell'uno che abbiamo cancellato) siamo certi che la stringa appartiene al linguaggio e andiamo nello stato di accettazione. In caso contrario ripetiamo questo procedimento ricominciando dalla cancellazione dello zero in testa alla stringa.

	0	1	$\star$
q_0	$(q_1, \star, +1)$	$(q_N, 1, +1)$	$(q_N, \star, +1)$
q_1	$(q_1, 0, +1)$	$(q_1, 1, +1)$	$(q_2, \star, -1)$
q_2	$(q_N, 0, +1)$	$(q_3, \star, -1)$	$(q_N, \star, +1)$
q_3	$(q_4, 0, -1)$	$(q_4, 1, -1)$	$(q_Y, \star, +1)$
q_4	$(q_4, 0, -1)$	$(q_4, 1, -1)$	$(q_0, \star, +1)$
q_Y			
q_N			



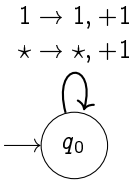
Esercizio 1.8 (*20/11/2024*)

Si fissi un alfabeto A. Costruire una macchina di Turing che diverga sempre a prescindere dal proprio input.

Soluzione 1.8

Come alfabeto scegliamo $A = \{1\}$ per semplicità. Ci sono molte possibili soluzioni, la più semplice di tutte consiste nel lasciare ogni elemento del nastro invariato e scorrere sempre verso destra.

	1	$\star$
q_0	$(q_0, 1, +1)$	$(q_0, \star, +1)$



Capitolo 2

Computabilità

Esercizio 2.1 *(20/06/2019)*

Si dimostri l'esistenza o meno di una funzione totale e calcolabile $f : \mathbb{N}^2 \to \mathbb{N}$ tale che per ogni i e j, l'i-esima Macchina di Turing, sull'input j, si ferma se e soltanto se si ferma entro $f(i, j)$ passi.

Soluzione 2.1

Una funzione f totale e calcolabile con tali caratteristiche non esiste. Per dimostrare ciò, supponiamo per assurdo che una tale funzione esista. Questa funzione per una MdT i e un naturale j restituisce il massimo numero di passi con cui M_i converge su j, se M_i converge su j, o un numero qualsiasi altrimenti (perché è totale). Allora si può stabilire il seguente algoritmo per decidere il problema dell'arresto $K' = \{(x, y) \in \mathbb{N} \mid M_x \downarrow y\}$:

```
begin
input(x);
input(y);
decodifica x per ottenere la MdT M_x;
esegui M_x su input y per f(x,y) passi;
if M_x è in una configurazione finale then
    output(SÌ)
else
    output(NO)
end
```

Per la tesi di Church-Turing tale algoritmo specifica una funzione calcolabile totale che decide il problema dell'arresto, che è assurdo, e dunque una funzione f con le caratteristiche descritte dall'esercizio non può esistere.

Esercizio 2.2 *(23/07/2019, 21/02/2022)*

Studiare l'insieme

$$I = \{i \in \mathbb{N} \mid \exists x, y \text{ tali che } \varphi_i(x) \neq \varphi_i(y)\}$$

Soluzione 2.2

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(x) \neq \varphi_i(y)$ per qualche $x, y \in \mathbb{N}$. Allora $\varphi_i(x) = \varphi_j(x) \neq \varphi_i(y) = \varphi_j(y)$ e quindi anche $j \in I$.

Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili che non sono costanti, cioè quelle funzioni f tali che per due punti x e y il valore di f è definito e differisce (la funzione sempre indefinita dunque non appartiene ad F, perché non esistono punti sulla quale è definita). Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili che non sono costanti, per esempio la funzione $g(x) = x + 1$. Quindi $I \neq \varnothing$;

- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili costanti, per esempio la funzione costante $c_0(x) = 0$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.

Dimostriamo ora la semidecidibilità di I descrivendo un algoritmo che accetti tutti e soli gli elementi di I:

```
begin
input(i);
decodifica i per ottenere la MdT M_i;
sia c_1 = 0;
sia flag = True;
while flag do
    begin
    siano x = π_1(c_1); n = π_2(c_1);
    esegui M_i su input x per n passi;
    if M_i è in una configurazione finale con output k then
        sia c_2 = 0;
        while flag do
            begin
            siano y = π_1(c_2); n = π_2(c_2);
            esegui M_i su input y per n passi;
            if M_i è in una conf. finale con output ≠ k then
                begin
                sia flag = False;
                output(SÌ)
                end
            else
                sia c_2 = c_2 + 1
            end
    else
        sia c_1 = c_1 + 1
    end
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme I, dunque I è semidecidibile.

Esercizio 2.3 *(13/09/2019)*

Sia data una funzione calcolabile $f : \mathbb{N} \to \mathbb{N}$. Quali sono le proprietà di decidibilità e semidecidibilità dell' immagine di f, $Im(f)$? Esistono funzioni f tali che $Im(f)$ è decidibile?

Soluzione 2.3

Svolgiamo questo esercizio mostrando

1. che l'immagine di una funzione non è decidibile, poi
2. che l'immagine di una funzione è semidecidibile.

e infine mostrano esempi di funzioni per cui l'immagine è decidibile.

1. Supponiamo per assurdo che l'immagine di una funzione sia decidibile. Allora possiamo definire una funzione $g : \mathbb{N} \to \mathbb{N}$ tale che, per ogni $y \in \mathbb{N}$:

$$g(y) = \begin{cases} \uparrow & \text{se } y \in Im(\varphi_y) \\ y & \text{se } y \notin Im(\varphi_y) \end{cases}$$

Naturalmente, se assumiamo che l'immagine di una funzione sia decidibile, questa funzione è calcolabile, e dunque esiste un indice m tale che $g = \varphi_m$ e quindi $Im(g) = Im(\varphi_m)$. Ma allora si ha la contraddizione

$$m \in Im(\varphi_m) \iff m \in Im(g) \iff m \notin Im(\varphi_m)$$

Segue quindi che l'immagine di una funzione non è decidibile.

2. Per dimostrare la semidecidibilità dell'immagine di una funzione f, possiamo, equivalentemente, mostrare un algoritmo che accetti tutti e soli i suoi elementi, oppure definire una funzione totale e calcolabile la cui immagine coincide esattamente con l'immagine di f. Svolgiamo l'esercizio seguendo la prima possibilità.

```
begin
input(y);
ottieni la MdT M_f che calcola f;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano x = π1(c); n = π2(c);
    esegui M_f su input x per n passi;
    if M_f è in una configurazione finale con output y then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'immagine di f, dunque $Im(f)$ è semidecidibile.

Nel caso generale l'immagine di una funzione non è decidibile, perché data una generica funzione h e un numero naturale y, non è dato sapere se esiste un x tale che $h(x) = y$: la funzione g definita precedentemente non può in alcun modo garantire che y non appaia eventualmente come risultato delle sue computazioni per un opportuno n.

Esistono però casi specifici di funzioni la cui immagine è decidibile: per esempio le funzioni costanti o una funzione che è indefinita per ogni input. Nel primo caso l'immagine è decidibile perché è sufficiente verificare se un certo y è uguale alla valutazione della funzione su, per esempio, input 0 e dare risposta affermativa in caso di riscontro positivo e risposta negativa altrimenti. Nel secondo caso basta rispondere negativamente ad ogni y.

Esercizio 2.4 (*27/09/2019*)

Si dica se la funzione f tale che, per ogni $x \in \mathbb{N}$

$$f(x) = \begin{cases} 1 & \text{se domani piove} \\ 0 & \text{altrimenti} \end{cases}$$

è calcolabile.

Soluzione 2.4

La funzione f è calcolabile. Per convincersi di ciò, basta considerare che, a seconda del meteo di domani, si ha che $f = c_1$ o $f = c_0$, ossia f corrisponde alla funzione costante 1 o alla funzione costante 0. Sia c_1 che c_0 sono funzioni calcolabili. Quindi la MdT che computa f è un elemento di un insieme composto da due elementi: una MdT M_1 che restituisce 1 per ogni input e una MdT M_0 che restituisce 0 per ogni input, tuttavia non è noto quale delle due MdT sia quella che computi f.

Esercizio 2.5 (*07/02/2020*)

Studiare l'insieme:

$$I = \{x \in \mathbb{N} \mid \exists y \text{ tali che } \varphi_x(y) = 5\}$$

Soluzione 2.5

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(y) = 5$ per un qualche $y \in \mathbb{N}$. Allora $\varphi_i(y) = \varphi_j(y) = 5$ e quindi anche $j \in I$.
Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili tali che 5 appartiene alla loro immagine. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili tali che 5 fa parte della loro immagine, per esempio la funzione $g(x) = 5x$. Quindi $I \neq \varnothing$;
- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili tali che 5 non fa parte della loro immagine, per esempio la funzione costante $c_0(x) = 0$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.
Dimostriamo ora la semidecidibilità di I descrivendo un algoritmo che accetti tutti e soli gli elementi di I:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano y = π_1(c); n = π_2(c);
    esegui M_x su input y per n passi;
    if M_x è in una configurazione finale con output 5 then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme I, dunque I è semidecidibile.

Esercizio 2.6 *(05/03/2020)*

Studiare l'insieme:

$$I = \{(x, y) \in \mathbb{N}^2 \mid \varphi_x(y) \uparrow\}$$

Soluzione 2.6

Dimostriamo che l'insieme I non è semidecidibile. Per prima cosa, notiamo che il suo complemento, $\bar{I} = \{(x, y) \mid \varphi_x(y) \downarrow\}$ è un insieme indecidibile, corrispondente al problema dell'arresto. Possiamo dimostrare la semidecidibilità di $\bar{I}$ tramite il seguente algoritmo:

```
begin
input(x);
input(y);
decodifica x per ottenere la MdT M_x;
esegui M_x su input y;
if M_x ↓ y then
    output(SÌ)
end
```

Per la tesi di Church-Turing il precedente algoritmo descrive una funzione calcolabile che accetta tutti e soli gli elementi di $\bar{I}$, confermando la sua semidecidibilità.

Per il teorema di Post, sappiamo che un insieme A è decidibile se e solo se sia A che $\bar{A}$ sono semidecidibili. Ne consegue che siccome $\bar{I}$ è semidecidibile, se anche il suo complemento I fosse semidecidibile allora $\bar{I}$ sarebbe decidibile, che è assurdo perché sappiamo che il problema dell'arresto non è decidibile. Quindi concludiamo che I non è semidecidibile.

Esercizio 2.7 *(18/06/2020)*

Studiare l'insieme:

$$I = \{(x, y) \in \mathbb{N}^2 \mid \varphi_x(y) = n_{NomeCognome}\}$$

dove il naturale $n_{NomeCognome}$ corrisponde alla codifica (nella biezione tra naturali e stringhe su un dato alfabeto) del nome e cognome del candidato all'esame.

Soluzione 2.7

In questo caso non possiamo applicare il teorema di Rice perché abbiamo un insieme di coppie (x, y). Quello che possiamo fare è definire un insieme $I' = \{x \mid \varphi_x(x) = n_{NomeCognome}\}$ e poi:

- dimostrare la sua indecidibilità, tramite una riduzione da K a I'.
- ridurre I' a I per dedurre l'indecidibilità di I.

Iniziamo verificando che I' non rispetta le funzioni. Ricordiamo che un insieme di indici I rispetta le funzioni se con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Prendiamo la funzione

$$\varphi_i(x) = \begin{cases} n_{NomeCognome} & \text{se } x = i \\ \uparrow & \text{altrimenti} \end{cases}$$

Chiaramente si ha che $i \in I'$, perché nel punto i la funzione restituisce il nome e cognome codificato. Ora prendiamo j tale che $\varphi_j = \varphi_i$. Si ha che $\varphi_i(i) = \varphi_j(i) = n_{NomeCognome}$ ma $\varphi_j(j) = \varphi_i(j) \uparrow$, quindi $j \notin I'$ e possiamo concludere che I' non rispetta le funzioni.

- Riduciamo l'insieme indecidibile $K = \{x \mid \varphi_x(x) \downarrow\}$ all'insieme I' in modo che $x \in K \iff s(x) \in I'$ per una qualche funzione totale calcolabile s. A questo scopo, definiamo una funzione parziale calcolabile $f : \mathbb{N}^2 \to \mathbb{N}$:

  $$f(x, y) = \begin{cases} n_{NomeCognome} & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{se } \varphi_x(x) \uparrow \end{cases}$$

  Inoltre per ogni x si può definire una funzione $g_x : \mathbb{N} \to \mathbb{N}$ calcolabile tale che:

  $$g_x(y) = \begin{cases} n_{NomeCognome} & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{se } \varphi_x(x) \uparrow \end{cases}$$

in altre parole, $g_x(y)$ calcola la funzione $f(x,y)$ dove il primo argomento x è fissato.

Dato che le (infinite) funzioni $g_x(y)$ sono calcolabili, hanno un proprio indice nell'enumerazione di tutte le funzioni calcolabili, e tale indice si può ottenere tramite una funzione totale calcolabile $s : \mathbb{N} \to \mathbb{N}$. Quindi per ogni x si ha che $g_x = \varphi_{s(x)}$

A questo punto possiamo osservare che per ogni x:

$$x \in K \iff \varphi_x(x) \downarrow \implies \forall y, \varphi_{s(x)}(y) = n_{NomeCognome} \implies s(x) \in I$$

$$x \notin K \iff \varphi_x(x) \uparrow \implies \forall y, \varphi_{s(x)}(y) \uparrow \implies s(x) \notin I$$

Quindi un algoritmo che decide I' ne produrrebbe uno che decide anche K: per ogni x, si calcola $s(x)$ e si verifica se appartiene ad I'; in caso affermativo $x \in K$ e in caso negativo $x \notin K$. Concludiamo che I' non è decidibile, altrimenti anche K lo sarebbe a sua volta.

- Riduciamo l'insieme I', che abbiamo appena dimostrato essere indecidibile, a I in modo che $x \in I' \iff s(x) \in I$ per una qualche funzione totale calcolabile s. La funzione che effettua la riduzione è $x \mapsto (x,x)$, infatti:

$$x \in I' \iff \varphi_x(x) = n_{NomeCognome} \iff (x,x) \in I$$

$$x \notin I' \iff \varphi_x(x) \neq n_{NomeCognome} \implies (x,x) \notin I$$

Quindi un algoritmo che decide I ne produrrebbe uno che decide anche I': per ogni x, si ottiene la coppia (x,x) e si verifica se appartiene ad I; in caso affermativo $x \in I'$ e in caso negativo $x \notin I'$. Concludiamo che I non è decidibile, altrimenti anche I' lo sarebbe a sua volta.

Dimostriamo ora la semidecidibilità di I descrivendo un algoritmo che accetti tutti e soli gli elementi di I:

```
begin
input(x);
input(y);
decodifica x per ottenere la MdT M_x;
esegui M_x su input y;
if M_x ↓ y con output n_NomeCognome then
    output(SÌ)
while True do skip
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme I, dunque I è semidecidibile.

Esercizio 2.8 (*01/07/2020, 20/07/2020, 07/09/2020*)

Dire se la funzione

$$f(x) = \begin{cases} 1 & \text{se } \varphi_x(0) \downarrow \\ 0 & \text{se } \varphi_x(0) \uparrow \end{cases}$$

è calcolabile.

Soluzione 2.8

La funzione f risulta essere la funzione caratteristica dell'insieme $I = \{x \mid \varphi_x(0) \downarrow\}$, o in altre parole, è una funzione che decide I. Quindi per stabilire se è calcolabile o meno possiamo invece studiare la decidibilità di I.
Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(0) \downarrow$. Allora si ha che $\varphi_i(0) \downarrow \iff \varphi_j(0) \downarrow$ (perché hanno lo stesso dominio) e quindi anche $j \in I$.
Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili che sono definite nel punto 0. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili definite nel punto 0, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \varnothing$;
- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili non definite nel punto 0, per esempio la funzione indefinita $\emptyset(x) = \uparrow$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile. Dato che non è decidibile, questo significa che la sua funzione caratteristica χ_I, che è proprio f, non è calcolabile.

Esercizio 2.9 (*28/09/2020*)

Studiare l'insieme:

$$A = \{x \in \mathbb{N} \mid \varphi_x(x) = x^2\}$$

ovvero, dire se A e complemento di A sono ricorsivi/ricorsivamente enumerabili.

Soluzione 2.9

In questo caso non si può utilizzare il teorema di Rice, perché A non rispetta le funzioni. Ricordiamo che un insieme di indici I rispetta le funzioni se con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Prendiamo la funzione

$$\varphi_i(x) = \begin{cases} x^2 & \text{se } x = i \\ \uparrow & \text{altrimenti} \end{cases}$$

Chiaramente si ha che $i \in A$, perché nel punto i la funzione restituisce il quadrato del proprio indice. Ora prendiamo j tale che $\varphi_j = \varphi_i$. Si ha che

$\varphi_i(i) = \varphi_j(i) = i^2$ ma $\varphi_j(j) = \varphi_i(j) \uparrow$, quindi $j \notin A$ e possiamo concludere che A non rispetta le funzioni.
Per dimostrarne quindi la sua indecidibilità riduciamo l'insieme indecidibile $K = \{x \mid \varphi_x(x) \downarrow\}$ all'insieme A in modo che $x \in K \iff s(x) \in A$ per una qualche funzione totale calcolabile s. A questo scopo, definiamo una funzione parziale calcolabile $f : \mathbb{N}^2 \to \mathbb{N}$:

$$f(x,y) = \begin{cases} y^2 & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{se } \varphi_x(x) \uparrow \end{cases}$$

Inoltre per ogni x si può definire una funzione $g_x : \mathbb{N} \to \mathbb{N}$ calcolabile tale che:

$$g_x(y) = \begin{cases} y^2 & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{se } \varphi_x(x) \uparrow \end{cases}$$

in altre parole, $g_x(y)$ calcola la funzione $f(x,y)$ dove il primo argomento x è fissato.
Dato che le (infinite) funzioni $g_x(y)$ sono calcolabili, hanno un proprio indice nell'enumerazione di tutte le funzioni calcolabili, e tale indice si può ottenere tramite una funzione totale calcolabile $s : \mathbb{N} \to \mathbb{N}$. Quindi per ogni x si ha che $g_x = \varphi_{s(x)}$
A questo punto possiamo osservare che per ogni x:

$$x \in K \iff \varphi_x(x) \downarrow \implies \forall y, \varphi_{s(x)}(y) = y^2 \implies \varphi_{s(x)}(s(x)) = s(x)^2 \iff s(x) \in A$$

$$x \notin K \iff \varphi_x(x) \uparrow \implies \forall y, \varphi_{s(x)}(y) \uparrow \implies \varphi_{s(x)}(s(x)) \neq s(x)^2 \iff s(x) \notin A$$

Quindi un algoritmo che decide A ne produrrebbe uno che decide anche K: per ogni x, si calcola $s(x)$ e si verifica se appartiene ad A; in caso affermativo $x \in K$ e in caso negativo $x \notin K$. Concludiamo che A non è decidibile, altrimenti anche K lo sarebbe a sua volta.
Dimostriamo ora la semidecidibilità di A descrivendo un algoritmo che accetti tutti e soli gli elementi di A:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
esegui M_x su input x;
if M_x ↓ x con output x^2 then
    output(SÌ)
while True do skip
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme A, dunque A è semidecidibile.
Il complemento di A è l'insieme $\bar{A} = \{x \in \mathbb{N} \mid \varphi_x(x) \neq x^2\}$. Per il teorema di Post, sappiamo che un insieme I è decidibile se e solo se sia I che $\bar{I}$ sono

semidecidibili. Ne consegue che siccome abbiamo dimostrato che A è semidecidibile, se anche il suo complemento $\bar{A}$ fosse semidecidibile allora A sarebbe anche decidibile, che è assurdo. Quindi concludiamo che $\bar{A}$ non è semidecidibile.

Esercizio 2.10 (*09/02/2021*)

Sia n un numero naturale fissato. Studiare l'insieme:

$$I = \{x \in \mathbb{N} \mid \varphi_x(x) \text{ converge in } n \text{ passi}\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.10

In questo caso non è possibile utilizzare il Teorema di Rice perché I non è un insieme di indici: $\forall x \in \mathbb{N}, \varphi_x(x)$ può convergere in n passi per una qualche MdT M_x che la computa, ma si può sempre costruire una nuova MdT M_y che calcola la stessa funzione di M_x ma esegue un numero arbitrario di passi in più (e.g. spostandosi sul nastro senza cambiarne i contenuti per certo un numero di passi e poi terminando).
Tuttavia, avendo fissato n a un certo numero naturale, si può dimostrare la decidibilità di I mediante il seguente algoritmo:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
esegui M_x su input x per n passi;
if M_x è in una configurazione finale then
    output(SÌ)
else
    output(NO)
end
```

Per la tesi di Church-Turing, tale algoritmo descrive una funzione calcolabile che decide l'insieme I, quindi l'insieme I è decidibile e anche semidecidibile.

Esercizio 2.11 (*26/02/2021*)

Studiare l'insieme:

$$I = \{x \in \mathbb{N} \mid \varphi_x(x) = 10\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.11

In questo caso non si può utilizzare il teorema di Rice, perché I non rispetta le funzioni. Ricordiamo che un insieme di indici I rispetta le funzioni se con $i \in I$

e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Prendiamo la funzione

$$\varphi_i(x) = \begin{cases} 10 & \text{se } x = i \\ \uparrow & \text{altrimenti} \end{cases}$$

Chiaramente si ha che $i \in I$, perché nel punto i la funzione restituisce esattamente 10. Ora prendiamo j tale che $\varphi_j = \varphi_i$. Si ha che $\varphi_i(i) = \varphi_j(i) = 10$ ma $\varphi_j(j) = \varphi_i(j) \uparrow$, quindi $j \notin I$ e possiamo concludere che I non rispetta le funzioni.

Per dimostrarne quindi la sua indecidibilità riduciamo l'insieme indecidibile $K = \{x \mid \varphi_x(x) \downarrow\}$ all'insieme I in modo che $x \in K \iff s(x) \in I$ per una qualche funzione totale calcolabile s. A questo scopo, definiamo una funzione parziale calcolabile $f : \mathbb{N}^2 \to \mathbb{N}$:

$$f(x, y) = \begin{cases} 10 & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{se } \varphi_x(x) \uparrow \end{cases}$$

Inoltre per ogni x si può definire una funzione $g_x : \mathbb{N} \to \mathbb{N}$ calcolabile tale che:

$$g_x(y) = \begin{cases} 10 & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{se } \varphi_x(x) \uparrow \end{cases}$$

in altre parole, $g_x(y)$ calcola la funzione $f(x, y)$ dove il primo argomento x è fissato.

Dato che le (infinite) funzioni $g_x(y)$ sono calcolabili, hanno un proprio indice nell'enumerazione di tutte le funzioni calcolabili, e tale indice si può ottenere tramite una funzione totale calcolabile $s : \mathbb{N} \to \mathbb{N}$. Quindi per ogni x si ha che $g_x = \varphi_{s(x)}$

A questo punto possiamo osservare che per ogni x:

$$x \in K \iff \varphi_x(x) \downarrow \implies \forall y, \varphi_{s(x)}(y) = 10 \implies \varphi_{s(x)}(s(x)) = 10 \iff s(x) \in I$$

$$x \notin K \iff \varphi_x(x) \uparrow \implies \forall y, \varphi_{s(x)}(y) \uparrow \implies \varphi_{s(x)}(s(x)) \neq 10 \iff s(x) \notin I$$

Quindi un algoritmo che decide I ne produrrebbe uno che decide anche K: per ogni x, si calcola $s(x)$ e si verifica se appartiene ad I; in caso affermativo $x \in K$ e in caso negativo $x \notin K$. Concludiamo che I non è decidibile, altrimenti anche K lo sarebbe a sua volta.

Dimostriamo ora la semidecidibilità di I descrivendo un algoritmo che accetti tutti e soli gli elementi di I:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
esegui M_x su input x;
if M_x ↓ x con output 10 then
    output(SÌ)
while True do skip
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme I, dunque I è semidecidibile.

Esercizio 2.12
Dimostrare per diagonalizzazione che l'insieme $I = \{n \in \mathbb{N} \mid \varphi_n(n) \downarrow\}$ non è decidibile.

Soluzione 2.12
Supponiamo per assurdo che I sia decidibile. Allora possiamo definire una funzione $f : \mathbb{N} \to \mathbb{N}$ tale che, per ogni $n \in \mathbb{N}$:

$$f(n) = \begin{cases} \uparrow & \text{se } \varphi_n(n) \downarrow \\ 0 & \text{se } \varphi_n(n) \uparrow \end{cases}$$

Naturalmente, se assumiamo I decidibile, questa funzione è calcolabile, e dunque esiste un indice m tale che $f = \varphi_m$ e quindi $dom(f) = dom(\varphi_m)$. Ma allora si ha la contraddizione

$$m \in dom(\varphi_m) \iff m \in dom(f) \iff m \notin dom(\varphi_m)$$

Segue quindi che I non è decidibile.

Esercizio 2.13
Dimostrare per diagonalizzazione che l'insieme $I = \{n \in \mathbb{N} \mid n \in Im(\varphi_n)\}$ non è decidibile.

Soluzione 2.13
Supponiamo per assurdo che I sia decidibile. Allora possiamo definire una funzione $f : \mathbb{N} \to \mathbb{N}$ tale che, per ogni $n \in \mathbb{N}$:

$$f(n) = \begin{cases} \uparrow & \text{se } n \in Im(\varphi_n) \\ n & \text{se } n \notin Im(\varphi_n) \end{cases}$$

Naturalmente, se assumiamo I decidibile, questa funzione è calcolabile, e dunque esiste un indice m tale che $f = \varphi_m$ e quindi $Im(f) = Im(\varphi_m)$. Ma allora si ha la contraddizione

$$m \in Im(\varphi_m) \iff m \in Im(f) \iff m \notin Im(\varphi_m)$$

Segue quindi che I non è decidibile.

Esercizio 2.14
Si consideri la funzione

$$f(n) = \begin{cases} \varphi_n(n) + 1 & \text{se } \varphi_n(n) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$$

a) Dire se f è calcolabile

b) Dire se esiste una estensione di f totale calcolabile

Soluzione 2.14

a) La funzione f è calcolabile. Per dimostrare ciò, invochiamo la tesi di Church-Turing stabilendo il seguente algoritmo per calcolare f:

```
begin
input(n);
decodifica n per ottenere la MdT M_n;
esegui M_n su input n;
if M_n ↓ n con output y then
    output(y + 1)
end
```

Si noti che, se n non fa parte del dominio di φ_n, la MdT che esegue il precedente algoritmo diverge al passo 3 come stabilito da $f(n)$ nel caso che $\varphi_n(n) \uparrow$.

b) Per rendere f' una estensione totale di f dobbiamo definirla in modo che restituisca un valore anche nel caso che $\varphi_n(n) \uparrow$, per esempio 0:

$$f'(n) = \begin{cases} \varphi_n(n) + 1 & \text{se } \varphi_n(n) \downarrow \\ 0 & \text{altrimenti} \end{cases}$$

Tuttavia sia con 0 che con qualsiasi altro valore venga usato, non esiste una estensione totale calcolabile di f. Assumiamo per assurdo che f' sia calcolabile ed esista un indice m tale che $f' = \varphi_m$. Allora si ha la contraddizione:

$$\varphi_m(m) = f'(m) = \varphi_m(m) + 1$$

Quindi f' non è calcolabile come non lo è qualsiasi altra estensione totale di f.

Esercizio 2.15

Si consideri la funzione totale f:

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{se } \varphi_x \text{ è totale} \\ 0 & \text{altrimenti} \end{cases}$$

a) Dimostrare che f non è calcolabile

b) Dedurre che l'insieme $Tot = \{x \in \mathbb{N} \mid \varphi_x \text{ è totale } \}$ non è decidibile.

Soluzione 2.15

a) Assumiamo per assurdo che f sia calcolabile ed esista un indice m tale che $f = \varphi_m$. Allora, considerando che f è totale, si ha la contraddizione:

$$\varphi_m(m) = f(m) = \varphi_m(m) + 1$$

Quindi f non è calcolabile.

b) Assumiamo per assurdo che l'insieme Tot sia decidibile. Quindi esiste una MdT M che può decidere, dato un numero naturale x, se la funzione φ_x è totale o meno.

Ma allora è possibile stabilire un algoritmo che calcoli f:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
if M stabilisce che φ_x è totale then
    begin
    sia y l'output di M_x sull'input x;
    output(y + 1)
    end
else
    output(0)
end
```

e stabilire per la Tesi di Church-Turing che f è calcolabile, che è assurdo. Quindi Tot non può essere decidibile.

Esercizio 2.16 *(22/06/2021, 28/02/2023)*

Studiare l'insieme:

$$I = \{x \in \mathbb{N} \mid \exists y \text{ tale che } \varphi_x(y) = x\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.16

In questo caso non si può utilizzare il teorema di Rice, perché I non rispetta le funzioni. Ricordiamo che un insieme di indici I rispetta le funzioni se con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Prendiamo la funzione

$$\varphi_i(x) = \begin{cases} i & \text{se } x = 0 \\ \uparrow & \text{altrimenti} \end{cases}$$

Chiaramente si ha che $i \in I$, perché $i \in Im(\varphi_i)$. Ora prendiamo j tale che $\varphi_j = \varphi_i$. Si ha che $i \in Im(\varphi_i) = Im(\varphi_j)$ ma $j \notin Im(\varphi_i) = Im(\varphi_j)$, quindi $j \notin I$ e possiamo concludere che I non rispetta le funzioni.

Per dimostrarne quindi la sua indecidibilità riduciamo l'insieme indecidibile $K = \{x \mid \varphi_x(x) \downarrow\}$ all'insieme I in modo che $x \in K \iff s(x) \in I$ per una qualche funzione totale calcolabile s. A questo scopo, definiamo una funzione parziale calcolabile $f : \mathbb{N}^2 \to \mathbb{N}$:

$$f(x, y) = \begin{cases} y & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{se } \varphi_x(x) \uparrow \end{cases}$$

Inoltre per ogni x si può definire una funzione $g_x : \mathbb{N} \to \mathbb{N}$ calcolabile tale che:

$$g_x(y) = \begin{cases} y & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{se } \varphi_x(x) \uparrow \end{cases}$$

in altre parole, $g_x(y)$ calcola la funzione $f(x, y)$ dove il primo argomento x è fissato.

Dato che le (infinite) funzioni $g_x(y)$ sono calcolabili, hanno un proprio indice nell'enumerazione di tutte le funzioni calcolabili, e tale indice si può ottenere tramite una funzione totale calcolabile $s : \mathbb{N} \to \mathbb{N}$. Quindi per ogni x si ha che $g_x = \varphi_{s(x)}$

A questo punto possiamo osservare che per ogni x:

$$x \in K \iff \varphi_x(x) \downarrow \implies \forall y, \varphi_{s(x)}(y) = y \implies \varphi_{s(x)}(s(x)) = s(x) \implies s(x) \in I$$

$$x \notin K \iff \varphi_x(x) \uparrow \implies \forall y, \varphi_{s(x)}(y) \uparrow \implies s(x) \notin Im(s(x)) \iff s(x) \notin I$$

Quindi un algoritmo che decide I ne produrrebbe uno che decide anche K: per ogni x, si calcola $s(x)$ e si verifica se appartiene ad I; in caso affermativo $x \in K$ e in caso negativo $x \notin K$. Concludiamo che I non è decidibile, altrimenti anche K lo sarebbe a sua volta.

Dimostriamo ora la semidecidibilità di I descrivendo un algoritmo che accetti tutti e soli gli elementi di I:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano y = π1(c); n = π2(c);
    esegui M_x su input y per n passi;
    if M_x è in una configurazione finale con output x then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme I, dunque I è semidecidibile.

Esercizio 2.17 *(01/07/2021)*

Studiare l'insieme:

$$I = \{x \in \mathbb{N} \mid Im(\varphi_x) = \varnothing\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.17

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(x) \uparrow$ per ogni $x \in \mathbb{N}$. Allora $Im(\varphi_i) = Im(\varphi_j) = \varnothing$ e quindi anche $j \in I$.
Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili la cui immagine è l'insieme vuoto, cioè le funzioni che sono indefinite su ogni input. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili la cui immagine è l'insieme vuoto, infatti questo è proprio il caso della funzione indefinita $\emptyset$ che diverge su ogni input. Quindi $I \neq \varnothing$;
- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili la cui immagine non è l'insieme vuoto, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.
Notiamo che il complemento $\bar{I} = \{x \mid Im(\varphi_x) \neq \varnothing\}$ è un insieme semidecidibile tramite il seguente algoritmo:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano y = π1(c); n = π2(c);
    esegui M_x su input y per n passi;
    if M_x è in una configurazione finale then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

che descrive una funzione calcolabile secondo la tesi di Church-Turing. Dunque $\bar{I}$ è semidecidibile. Per il teorema di Post, dato che $\bar{I}$ è semidecidibile, I deve necessariamente essere un insieme non semidecidibile, altrimenti si avrebbe per assurdo che I è un insieme decidibile (mentre tramite il teorema di Rice abbiamo dimostrato che non può esserlo).

Esercizio 2.18 *(06/09/2021)*

Sia $A \subseteq \mathbb{N}$ un insieme e sia $f : \mathbb{N} \to \mathbb{N}$ una funzione calcolabile. Dimostrare che se A è ricorsivamente enumerabile allora $f(A) = \{y \in \mathbb{N} \mid \exists x \in A, y = f(x)\}$ è ricorsivamente enumerabile. Vale anche il contrario? Ovvero da $f(A)$ ricorsivamente enumerabile si può dedurre che A è ricorsivamente enumerabile?

Soluzione 2.18

- Se A è semidecidibile (i.e. ricorsivamente enumerabile), allora esiste una funzione g che lo semidecide e una funzione h che lo enumera.

  L'insieme $f(A)$ corrisponde all'insieme degli elementi di $\mathbb{N}$ che sono immagine di f nei punti appartenenti all'insieme A. Per dimostrare che $f(A)$ é semidecidibile, dobbiamo specificare un algoritmo che definisca una funzione calcolabile che semidecide $f(A)$. Per fare ciò, possiamo sfruttare sia la funzione g che la funzione h. Proseguiamo seguendo quest'ultima possibilità.

```
begin
input(y);
ottieni la MdT M_k e la MdT M_f;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano z = π_1(c); n = π_2(c);
    esegui M_k su input z e sia x il suo risultato;
    esegui M_f su input x per n passi;
    if M_f è in una configurazione finale con output y then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che semidecide $f(A)$, dunque $f(A)$ è semidecidibile (i.e. ricorsivamente enumerabile).

- Prendiamo come insieme $A = \{x \mid \varphi_x(x) \uparrow\}$. L'insieme non è semidecidibile (i.e. non ricorsivamente enumerabile). Prendiamo poi come funzione $f : \mathbb{N} \to \mathbb{N}$ la funzione costante 1. Chiaramente $f(A) = \{1\}$ è un insieme decidibile, e di conseguenza è anche semidecidibile (i.e. ricorsivamente enumerabile). Tuttavia abbiamo per definizione che A non è ricorsivamente enumerabile, quindi la deduzione non si può fare.

Esercizio 2.19 *(27/09/2021)*

Studiare l'insieme $I = \{i \in \mathbb{N} \mid 3 \in dom(\varphi_i)\}$. E se invece fosse $I = \{i \in \mathbb{N} \mid$ l'insieme dei numeri pari $\subseteq dom(\varphi_i)\}$?

Soluzione 2.19

- Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(3) \downarrow$. Allora si ha che $\varphi_i(3) \downarrow \iff \varphi_j(3) \downarrow$ (perché hanno lo stesso dominio) e quindi anche $j \in I$.

  Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili che sono definite nel punto 3. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili che sono definite nel punto 3, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \varnothing$;
- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili il cui dominio non include 3, per esempio la funzione indefinita $\emptyset(x) = \uparrow$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.

Dimostriamo ora la semidecidibilità di I descrivendo un algoritmo che accetti tutti e soli gli elementi di I:

```
degin
input(i);
decodifica i per ottenere la MdT M_i;
esegui M_i su input 3;
if M_i ↓ 3 then
    output(SÌ)
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme I, dunque I è semidecidibile.

- Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(x) \downarrow$ per ogni x pari. Allora si ha che $\varphi_i(x) \downarrow \iff \varphi_j(x) \downarrow$ per ogni x pari (perché hanno lo stesso dominio) e quindi anche $j \in I$.

  Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili che sono definite (almeno) su ogni numero pari. Dunque l'insieme I:

  - non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili che sono definite su tutti i numeri pari, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \varnothing$;
  - non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili il cui dominio non include tutti i numeri pari, per esempio la funzione indefinita $\emptyset(x) = \uparrow$ (che non è definita su nessun numero, e quindi anche su nessun numero pari). Quindi $I \neq \mathbb{N}$.

  Quindi per il teorema di Rice I non è decidibile.

  Per dimostrare la non semidecidibilità di I purtroppo non possiamo studiare il suo complemento: l'insieme degli indici delle funzioni il cui dominio non include tutti i numeri pari è a sua volta non semidecidibile. Quindi

quello che possiamo fare è ridurre un insieme non semidecidibile, per esempio l'insieme T delle funzioni totali, a I in modo che $x \in T \iff s(x) \in I$ per una qualche funzione totale calcolabile s.

A questo scopo, definiamo una funzione parziale calcolabile $f : \mathbb{N}^2 \to \mathbb{N}$:

$$f(x,y) = \begin{cases} 0 & \text{se } \varphi_x(y)\downarrow \text{ e } \varphi_x(y+1)\downarrow \\ \uparrow & \text{se } \varphi_x(y)\uparrow \text{ o } \varphi_x(y+1)\uparrow \end{cases}$$

Inoltre per ogni x si può definire una funzione $g_x : \mathbb{N} \to \mathbb{N}$ calcolabile tale che:

$$g_x(y) = \begin{cases} 0 & \text{se } \varphi_x(y)\downarrow \text{ e } \varphi_x(y+1)\downarrow \\ \uparrow & \text{se } \varphi_x(y)\uparrow \text{ o } \varphi_x(y+1)\uparrow \end{cases}$$

in altre parole, $g_x(y)$ calcola la funzione $f(x,y)$ dove il primo argomento x è fissato.

Dato che le (infinite) funzioni $g_x(y)$ sono calcolabili, hanno un proprio indice nell'enumerazione di tutte le funzioni calcolabili, e tale indice si può ottenere tramite una funzione totale calcolabile $s : \mathbb{N} \to \mathbb{N}$. Quindi per ogni x si ha che $g_x = \varphi_{s(x)}$

A questo punto possiamo osservare che per ogni x:

$$\begin{gathered} x \in T \iff \forall y, \varphi_x(y)\downarrow \implies \forall y, \varphi_{s(x)}(y) = 0 \implies \\ \forall y(\exists k(y = 2k) \implies \varphi_{s(x)}(y)\downarrow) \iff s(x) \in I \end{gathered}$$

$$\begin{gathered} x \notin T \iff \exists y, \varphi_x(y)\uparrow \implies \exists y, \varphi_{s(x)}(y)\uparrow \implies \\ \exists y(\exists k(y = 2k) \implies \varphi_{s(x)}(y)\uparrow) \iff s(x) \notin I \end{gathered}$$

Quindi un algoritmo che accetta I ne produrrebbe uno che accetta anche T: per ogni x, si calcola $s(x)$ si esegue l'algoritmo che accetta I; se $s(x)$ viene accettato, accettiamo anche x. Concludiamo che I non è semidecidibile, altrimenti anche T lo sarebbe a sua volta.

Esercizio 2.20 (*15/11/2021*)

Studiare l'insieme $I = \{i \in \mathbb{N} \mid 3 \notin Im(\varphi_i)\}$.

Soluzione 2.20

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(x) \neq 3$ per ogni $x \in \mathbb{N}$. Allora $3 \notin Im(\varphi_i) = Im(\varphi_j)$ e quindi anche $j \in I$.

Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili tali che 3 non fa parte della loro immagine. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili la cui immagine non include 3, per esempio la funzione costante $c_0(x) = 0$. Quindi $I \neq \varnothing$;

- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili la cui immagine include il valore 3, per esempio la funzione costante $c_3(x) = 3$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.
Notiamo che il complemento $\bar{I} = \{i \in \mathbb{N} \mid 3 \in Im(\varphi_i)\}$ è un insieme semidecidibile tramite il seguente algoritmo:

```
begin
input(i);
decodifica i per ottenere la MdT M_i;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano x = π_1(c); n = π_2(c);
    esegui M_i su input x per n passi;
    if M_i è in una configurazione finale con output 3 then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

che descrive una funzione calcolabile secondo la tesi di Church-Turing. Dunque $\bar{I}$ è semidecidibile. Per il teorema di Post, dato che $\bar{I}$ è semidecidibile, I deve necessariamente essere un insieme non semidecidibile, altrimenti si avrebbe per assurdo che I è un insieme decidibile (mentre tramite il teorema di Rice abbiamo dimostrato che non può esserlo).

Esercizio 2.21 *(07/02/2022, 05/02/2025)*
Studiare l'insieme $I = \{i \in \mathbb{N} \mid \exists x \text{ tale che } \varphi_i(x) = x\}$.

Soluzione 2.21
Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(x) = x$ per un qualche $x \in \mathbb{N}$. Allora $\varphi_i(x) = \varphi_j(x) = x$ e quindi anche $j \in I$.

Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili che hanno almeno un punto fisso. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili con almeno un punto fisso, per esempio la funzione identità $id(x) = x$ (che ne ha infiniti). Quindi $I \neq \varnothing$;
- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili che non hanno punti fissi, per esempio la funzione $h(x) = x + 1$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.
Dimostriamo ora la semidecidibilità di I descrivendo un algoritmo che accetti tutti e soli gli elementi di I:

```
begin
input(i);
decodifica i per ottenere la MdT M_i;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano x = π_1(c); n = π_2(c);
    esegui M_i su input x per n passi;
    if M_i è in una configurazione finale con output x then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme I, dunque I è semidecidibile.

Esercizio 2.22 *(07/03/2022)*
Sia f una funzione calcolabile. Si dica se l'insieme $I = \{i \in \mathbb{N} \mid \varphi_i = f\}$ è un insieme decidibile. Si dica se l'insieme I è finito o infinito.

Soluzione 2.22

- Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che $\varphi_i = f$. Allora $\varphi_i = \varphi_j = f$ e quindi anche $j \in I$.

Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. Fissata f con arietà $n \in \mathbb{N}$, l'insieme I rappresenta la famiglia F di funzioni n-arie calcolabili che consiste solamente di f, cioè $F = \{f\}$. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché dalla calcolabilità di f segue che I debba contenere tutti i suoi indici. Quindi $I \neq \varnothing$;
- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono altre funzioni calcolabili oltre a f, i cui indici non appartengono a I. Per mostrare che esistono altre funzioni calcolabili che non siano f, per esempio, nel caso $n = 1$, possiamo mostrare che:
  - Se f non è la funzione indefinita, la funzione $g(x) = f(x) + 1$ è calcolabile e diversa da f.
  - Se f è la funzione indefinita, allora la funzione $h(x) = x$ è calcolabile e diversa da f.

  Un ragionamento analogo si può fare anche per gli altri valori di n. Quindi $I \neq \mathbb{N}$ per ogni scelta di $n \geq 1$.

Quindi per il teorema di Rice I non è decidibile.

- L'insieme I è infinito. Difatti per ogni funzione calcolabile f esistono infinite macchine di Turing che la computano. Per convincersi di ciò è sufficiente considerare che data una MdT M che computa f (ne deve esistere almeno una dato che f è una funzione calcolabile), è sempre possibile aggiungere nuove istruzioni che non cambiano il suo output ma che comunque portano a nuove MdT $M', M'', \ldots$ con indice diverso da M.

Esercizio 2.23 (*17/06/2022, 07/02/2023*)
Si dica se l'insieme $I = \{i \in \mathbb{N} \mid \varphi_i \text{ è una funzione costante}\}$ è un insieme decidibile. Si dica se l'insieme I è finito o infinito.

Soluzione 2.23

1. Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(x) = c$ per ogni $x \in \mathbb{N}$ e per un qualche numero naturale c. Allora $\varphi_i(x) = \varphi_j(x) = c$ per ogni $x \in \mathbb{N}$ e quindi anche $j \in I$.

   Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili che sono costanti. Dunque l'insieme I:

   - non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili costanti, per esempio la funzione costante $c_0(x) = 0$. Quindi $I \neq \varnothing$;
   - non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili che non sono costanti, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.

2. L'insieme I è infinito. Difatti per ogni funzione costante esistono infinite macchine di Turing che la computano. Per convincersi di ciò è sufficiente considerare che data una MdT M che computa una certa funzione costante (ne deve esistere almeno una dato che tutte le funzioni costanti sono calcolabili), è sempre possibile aggiungere nuove istruzioni che non cambiano il suo output ma che comunque portano a nuove MdT $M', M'', \ldots$ con indice diverso da M.

Esercizio 2.24 (*27/06/2022*)

Studiare l'insieme $I = \{i \in \mathbb{N} \mid \varphi_i \text{ è una funzione costante } \}$. Si dica se l'insieme I è finito o infinito.

Soluzione 2.24

1. Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(x) = c$ per ogni $x \in \mathbb{N}$ e per un qualche numero naturale c. Allora $\varphi_i(x) = \varphi_j(x) = c$ per ogni $x \in \mathbb{N}$ e quindi anche $j \in I$.

   Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili che sono costanti. Dunque l'insieme I:

   - non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili costanti, per esempio la funzione costante $c_0(x) = 0$. Quindi $I \neq \varnothing$;
   - non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili che non sono costanti, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \mathbb{N}$.

   Quindi per il teorema di Rice I non è decidibile.

   Per dimostrare la non semidecidibilità di I purtroppo non possiamo studiare il suo complemento: tra le funzioni che non sono costanti c'è anche la funzione indefinita, e non esiste un algoritmo che possa accettare il suo indice (in altre parole, anche il complemento di I è a sua volta non semidecidibile). Quindi quello che possiamo fare è ridurre un insieme non semidecidibile, per esempio l'insieme T delle funzioni totali, a I in modo che $x \in T \iff s(x) \in I$ per una qualche funzione totale calcolabile s.

   A questo scopo, definiamo una funzione parziale calcolabile $f : \mathbb{N}^2 \to \mathbb{N}$:

   $$f(x, y) = \begin{cases} 0 & \text{se } \varphi_x(y) \downarrow \\ \uparrow & \text{se } \varphi_x(y) \uparrow \end{cases}$$

   Inoltre per ogni x si può definire una funzione $g_x : \mathbb{N} \to \mathbb{N}$ calcolabile tale che:

$$g_x(y) = \begin{cases} 0 & \text{se } \varphi_x(y) \downarrow \\ \uparrow & \text{se } \varphi_x(y) \uparrow \end{cases}$$

in altre parole, $g_x(y)$ calcola la funzione $f(x, y)$ dove il primo argomento x è fissato.

Dato che le (infinite) funzioni $g_x(y)$ sono calcolabili, hanno un proprio indice nell'enumerazione di tutte le funzioni calcolabili, e tale indice si può ottenere tramite una funzione totale calcolabile $s : \mathbb{N} \to \mathbb{N}$. Quindi per ogni x si ha che $g_x = \varphi_{s(x)}$

A questo punto possiamo osservare che per ogni x:

$$x \in T \iff \forall y, \varphi_x(y) \downarrow \implies \forall y, \varphi_{s(x)}(y) = 0 \implies s(x) \in I$$

$$x \notin T \iff \exists y, \varphi_x(y) \uparrow \implies \exists y, \varphi_{s(x)}(y) \uparrow \implies s(x) \notin I$$

Quindi un algoritmo che accetta I ne produrrebbe uno che accetta anche T: per ogni x, si calcola $s(x)$ si esegue l'algoritmo che accetta I; se $s(x)$ viene accettato, accettiamo anche x. Concludiamo che I non è semidecidibile, altrimenti anche T lo sarebbe a sua volta.

2. L'insieme I è infinito. Difatti per ogni funzione costante esistono infinite macchine di Turing che la computano. Per convincersi di ciò è sufficiente considerare che data una MdT M che computa una certa funzione costante (ne deve esistere almeno una dato che tutte le funzioni costanti sono calcolabili), è sempre possibile aggiungere nuove istruzioni che non cambiano il suo output ma che comunque portano a nuove MdT $M', M'', \ldots$ con indice diverso da M.

Esercizio 2.25 (*15/07/2022*)

Studiare l'insieme $I = \{x \in \mathbb{N} \mid 5 \notin Im(\varphi_x)\}$.

Soluzione 2.25

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(y) \neq 5$ per ogni $y \in \mathbb{N}$. Allora $5 \notin Im(\varphi_i) = Im(\varphi_j)$ e quindi anche $j \in I$.

Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili tali che 5 non fa parte della loro immagine. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili la cui immagine non include 5, per esempio la funzione costante $c_0(x) = 0$. Quindi $I \neq \varnothing$;

- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili la cui immagine include il valore 5, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.
Notiamo che il complemento $\bar{I} = \{x \mid 5 \in Im(\varphi_x)\}$ è un insieme semidecidibile tramite il seguente algoritmo:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano y = π1(c); n = π2(c);
    esegui M_x su input y per n passi;
    if M_x è in una configurazione finale con output 5 then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

che descrive una funzione calcolabile secondo la tesi di Church-Turing. Dunque $\bar{I}$ è semidecidibile. Per il teorema di Post, dato che $\bar{I}$ è semidecidibile, I deve necessariamente essere un insieme non semidecidibile, altrimenti si avrebbe per assurdo che I è un insieme decidibile (mentre tramite il teorema di Rice abbiamo dimostrato che non può esserlo).

Esercizio 2.26 *(07/09/2022)*
Si dimostri che esiste una funzione $f : \mathbb{N} \to \{0, 1\}$ unaria totale e non calcolabile.
NB: non basta definire la funzione, ma occorre anche dimostrare la sua non calcolabilità.

Soluzione 2.26
Si può notare che un tipo particolare di funzione totale con immagine sottoinsieme di $\{0, 1\}$ è la funzione caratteristica di un insieme. Quindi per definire una funzione del tipo richiesto dall'esercizio è sufficiente scegliere un insieme non decidibile: in tal caso la funzione caratteristica di quell'insieme è sicuramente non calcolabile.
La più famosa funzione caratteristica di questo tipo è probabilmente la funzione caratteristica dell'insieme $K = \{x \mid \varphi_x(x) \downarrow\}$, ovvero:

$$f(x) = \begin{cases} 1 & \text{se } \varphi_x(x) \downarrow \\ 0 & \text{se } \varphi_x(x) \uparrow \end{cases}$$

A questo punto la dimostrazione della non calcolabilità di f segue dalla dimostrazione dell'indecidibilità di K.
In alternativa, possiamo prendere la funzione caratteristica di un altro insieme e quindi dimostrare che l'insieme non decidibile, per esempio:

$$g(x) = \begin{cases} 1 & \text{se } \varphi_x(0) \downarrow \\ 0 & \text{se } \varphi_x(0) \uparrow \end{cases}$$

In questo caso abbiamo la possibilità di dimostrare la non calcolabilità di g studiando l'insieme che g decide, ovvero l'insieme $I = \{x \mid \varphi_x(0) \downarrow\}$.
L'insieme I rispetta le funzioni, ovvero con $i \in I$ e j tale che $\varphi_i \cong \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(0) \downarrow$. Allora si ha che $\varphi_i(0) \downarrow \iff \varphi_j(0) \downarrow$ (perché hanno lo stesso dominio) e quindi anche $j \in I$.
Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili che sono definite nel punto 0. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili definite nel punto 0, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \varnothing$;
- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili non definite nel punto 0, per esempio la funzione indefinita $\emptyset(x) = \uparrow$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile, e di conseguenza la sua funzione caratteristica, che è proprio g, non è calcolabile.

Esercizio 2.27 *(19/09/2022)*
Si dica, giustificando la risposta formalmente, se la funzione $f : \mathbb{N} \to \mathbb{N}$ tale che per ogni $x \in \mathbb{N}$

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{se } \varphi_x(x) \downarrow \\ 0 & \text{altrimenti} \end{cases}$$

è calcolabile.

Soluzione 2.27
La funzione f non è calcolabile. Ci sono almeno due modi per dimostrarlo:

- Procedendo per diagonalizzazione, assumiamo che f sia calcolabile. Allora esiste sicuramente un certo indice m tale che $\varphi_m = f$. Si può notare che:

  $$\varphi_m(m) = f(m) = \varphi_m(m) + 1$$

  che è un assurdo. Quindi f non è calcolabile. L'ultimo passaggio è dato dal fatto che f è totale: quindi $f(m)$ selezionerà sicuramente la prima opzione nella sua definizione.

- Procedendo per riduzione, si può notare che f ha come risultato 0 solo nel caso che $\varphi_x(x) \uparrow$, mentre ha un risultato strettamente maggiore di 0 quando $\varphi_x(x) \downarrow$. Quindi se assumiamo che f è calcolabile allora avremmo un algoritmo per decidere l'insieme $K = \{x \mid \varphi_x(x) \downarrow\}$: si calcola $f(x)$ e si ritorna 1 se $f(x) > 0$, 0 altrimenti. Questo è chiaramente assurdo perché sappiamo che K non è decidibile, e di conseguenza f non può essere calcolabile.

Esercizio 2.28 *(03/10/2022)*

Si dica, giustificando la risposta formalmente, se la funzione $f : \mathbb{N} \to \mathbb{N}$ tale che per ogni $x \in \mathbb{N}$

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{se } \varphi_x(x) \downarrow \text{ in } x \text{ passi} \\ 0 & \text{altrimenti} \end{cases}$$

è calcolabile.

Soluzione 2.28

La funzione f è calcolabile, perché è sufficiente simulare φ_x con input x per x passi e quindi generare il risultato richiesto da f a seconda se la MdT x-esima è in una configurazione finale o meno. Quindi dimostriamo la calcolabilità di f mostrando il suo algoritmo.

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
esegui M_x su input x per x passi;
if M_x è in una configurazione finale con output y then
    output(y + 1)
else
    output(0)
end
```

Per la tesi di Church-Turing questo algoritmo descrive una funzione calcolabile, quindi concludiamo che f è calcolabile.

Esercizio 2.29 *(09/06/2023)*

Studiare l'insieme

$$I = \{i \in \mathbb{N} \mid 5 \notin Dom(\varphi_i)\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.29

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(5) \uparrow$. Allora dato che $Dom(\varphi_i) = Dom(\varphi_j)$ anche $\varphi_j(5) \uparrow$ e quindi anche $j \in I$.
Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili tali che 5 non fa parte del loro dominio. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili il cui dominio non include 5, per esempio la funzione indefinita $\emptyset(x) = \uparrow$. Quindi $I \neq \varnothing$;
- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili il cui dominio include il valore 5, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.
Notiamo che il complemento $\bar{I} = \{i \in \mathbb{N} \mid 5 \in Dom(\varphi_i)\}$ è un insieme semidecidibile tramite il seguente algoritmo:

```
begin
input(i);
decodifica i per ottenere la MdT M_i;
esegui M_i su input 5;
if M_i ↓ 5 then
    output(SÌ)
end
```

che descrive una funzione calcolabile secondo la tesi di Church-Turing. Dunque $\bar{I}$ è semidecidibile. Per il teorema di Post, dato che $\bar{I}$ è semidecidibile, I deve necessariamente essere un insieme non semidecidibile, altrimenti si avrebbe per assurdo che I è un insieme decidibile (mentre tramite il teorema di Rice abbiamo dimostrato che non può esserlo).

Esercizio 2.30 *(23/06/2023)*

Si dica se la funzione f tale che per ogni $x, y, z \in \mathbb{N}$

$$f(x, y, z) = \begin{cases} \downarrow & \text{se } \varphi_x(y) = z \\ \uparrow & \text{altrimenti} \end{cases}$$

è calcolabile.

Soluzione 2.30

La funzione f è calcolabile, perché è sufficiente simulare φ_x con input y e, in caso converga, verificare se il risultato che si ottiene è uguale a z. Quindi dimostriamo la calcolabilità di f mostrando il suo algoritmo.

```
begin
input(x);
input(y);
input(z);
decodifica x per ottenere la MdT M_x;
esegui M_x su input y;
if M_x ↓ y con output z then
    output(0)
while True do skip
end
```

Per la tesi di Church-Turing questo algoritmo descrive una funzione calcolabile, quindi concludiamo che f è calcolabile.

Esercizio 2.31 *(12/07/2023)*

Studiare l'insieme:

$$I = \{i \in \mathbb{N} \mid 0 \in Im(\varphi_i)\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.31

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(x) = 0$ per qualche $x \in \mathbb{N}$. Allora $0 \in Im(\varphi_i) = Im(\varphi_j)$ e quindi anche $j \in I$.

Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili tali che 0 fa parte della loro immagine. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili la cui immagine include 0, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \varnothing$;

- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili la cui immagine non include il valore 0, per esempio la funzione successore $s(x) = x + 1$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.

Dimostriamo ora la semidecidibilità di I descrivendo un algoritmo che accetti tutti e soli gli elementi di I:

```
begin
input(i);
decodifica i per ottenere la MdT M_i;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano x = π_1(c); n = π_2(c);
    esegui M_i su input x per n passi;
    if M_i è in una configurazione finale con output 0 then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme I, dunque I è semidecidibile.

Esercizio 2.32 *(19/09/2023, 28/09/2023)*

Studiare l'insieme:

$$I = \{i \in \mathbb{N} \mid \exists y \text{ tale che } \varphi_x(y) + 1 = 5\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.32

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(y) + 1 = 5$ per qualche $y \in \mathbb{N}$. Allora $\varphi_i(y) + 1 = \varphi_j(y) + 1 = 5$ e quindi anche $j \in I$.
Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili tali che 4 appartiene alla loro immagine, dato che 4 è l'unico naturale k per cui $k + 1 = 5$. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili tali che 4 fa parte della loro immagine, per esempio la funzione costante $c_4(x) = 4$. Quindi $I \neq \varnothing$;
- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili tali che 4 non fa parte della loro immagine, per esempio la funzione costante $c_0(x) = 0$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.
Dimostriamo ora la semidecidibilità di I descrivendo un algoritmo che accetti tutti e soli gli elementi di I:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano y = π1(c); n = π2(c);
    esegui M_x su input y per n passi;
    if M_x è in una configurazione finale con output 4 then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme I, dunque I è semidecidibile.

Esercizio 2.33 *(27/02/2024)*

Studiare l'insieme:

$$I = \{x \mid \exists y \text{ tale che } \varphi_x(y) \downarrow\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.33

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(y) \downarrow$ per qualche $y \in \mathbb{N}$. Allora $\varphi_i(y) \downarrow \implies \varphi_j(y) \downarrow$ e quindi anche $j \in I$.
Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili tali che sono definite in almeno un punto, ovvero le funzioni che non sono la funzione sempre indefinita $\emptyset$. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili tali che sono definite in almeno un punto, per esempio la funzione successore $f(x) = x + 1$ che è totale e quindi è definita per ogni punto. Quindi $I \neq \varnothing$;

- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili tali che non sono definite su nessun punto, per l'appunto la funzione sempre indefinita $\emptyset$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.
Dimostriamo ora la semidecidibilità di I descrivendo un algoritmo che accetti tutti e soli gli elementi di I:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano y = π1(c); n = π2(c);
    esegui M_x su input y per n passi;
    if M_x è in una configurazione finale then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme I, dunque I è semidecidibile.

Esercizio 2.34 *(24/06/2024)*

Studiare l'insieme:

$$I = \{x \in \mathbb{N} \mid \varphi_x(0) \downarrow \vee \varphi_x(1) \downarrow\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.34

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(0) \downarrow$ e/o $\varphi_i(1) \downarrow$. Allora $\varphi_i(0) \downarrow \implies \varphi_j(0) \downarrow$ e $\varphi_i(1) \downarrow \implies \varphi_j(1) \downarrow$, quindi anche $j \in I$.
Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili tali che sono definite nel punto 0 oppure nel punto 1. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili tali che sono definite in 0 o 1, per esempio la funzione identità $f(x) = x$ che è totale e quindi è definita per ogni punto. Quindi $I \neq \varnothing$;
- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili tali che non sono definite su 0 o 1, per esempio la funzione sempre indefinita $\emptyset$ che non è definita su nessun punto. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.
Dimostriamo ora la semidecidibilità di I descrivendo un algoritmo che accetti tutti e soli gli elementi di I:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
sia c = 0;
sia flag = True;
while flag do
    begin
    esegui M_x su input 0 per c passi;
    if M_x è in una configurazione finale then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        begin
        esegui M_x su input 1 per c passi;
        if M_x è in una configurazione finale then
            begin
            sia flag = False;
            output(SÌ)
            end
        else
            sia c = c + 1
        end
    end
end
```

Per la tesi di Church-Turing tale algoritmo descrive una funzione calcolabile che accetta l'insieme I, dunque I è semidecidibile.

Esercizio 2.35 *(08/07/2024)*
Studiare l'insieme

$$I = \{x \in \mathbb{N} \mid \varphi_x(0) \uparrow \ \texttt{AND} \ \varphi_x(1) \uparrow\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.35

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione tale che $\varphi_i(0) \uparrow$ e $\varphi_i(1) \uparrow$. Allora dato che $Dom(\varphi_i) = Dom(\varphi_j)$ anche $\varphi_j(0) \uparrow$ e $\varphi_j(1) \uparrow$ e quindi anche $j \in I$.

Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili tali che 0 e 1 non fanno parte del loro dominio. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili il cui dominio non include nè 0 nè 1, per esempio la funzione indefinita $\emptyset(x) = \uparrow$. Quindi $I \neq \varnothing$;

- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili il cui dominio include almeno 0 o 1, se non entrambi, per esempio la funzione identità $id(x) = x$ che è pure totale. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.

Notiamo che il complemento $\bar{I} = \{x \in \mathbb{N} \mid \varphi_x(0) \downarrow \ \texttt{OR} \ \varphi_x(1) \downarrow\}$ è un insieme semidecidibile tramite il seguente algoritmo:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
sia c = 0;
sia flag = True;
while flag do
    begin
    esegui M_x su input 0 per c passi;
    if M_x è in una configurazione finale then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        begin
        esegui M_x su input 1 per c passi;
        if M_x è in una configurazione finale then
            begin
            sia flag = False;
            output(SÌ)
            end
        else
            sia c = c + 1
        end
    end
end
```

che descrive una funzione calcolabile secondo la tesi di Church-Turing. Dunque $\bar{I}$ è semidecidibile. Per il teorema di Post, dato che $\bar{I}$ è semidecidibile, I deve necessariamente essere un insieme non semidecidibile, altrimenti si avrebbe per assurdo che I è un insieme decidibile (mentre tramite il teorema di Rice abbiamo dimostrato che non può esserlo).

Esercizio 2.36 *(24/07/2024)*

Dire se la funzione

$$f(x) = \begin{cases} 1 & \text{se } \varphi_x(x) \downarrow \text{ in } x^2 \text{ passi} \\ 0 & \text{altrimenti} \end{cases}$$

è calcolabile.

Soluzione 2.36

La funzione f è calcolabile, perché è sufficiente simulare φ_x con input x e, in caso converga entro x^2 passi, restituire il valore 1, altrimenti restituire 0. Quindi dimostriamo la calcolabilità di f mostrando il suo algoritmo.

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
esegui M_x su input x per x^2 passi;
if M_x è in una configurazione finale then
    output(1)
else
    output(0)
end
```

Per la tesi di Church-Turing questo algoritmo descrive una funzione calcolabile, quindi concludiamo che f è calcolabile.

Esercizio 2.37 *(17/09/2024)*

Studiare l'insieme:

$$I = \{i \in \mathbb{N} \mid Dom(\varphi_i) = \varnothing\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.37

Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che $Dom(\varphi_i) = \varnothing$, e dato che $\varphi_i = \varphi_j$, allora $Dom(\varphi_j) = Dom(\varphi_i)$ e quindi anche $j \in I$.

Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili il cui dominio è l'insieme vuoto, cioè le funzioni che sono indefinite su ogni input. Dunque l'insieme I:

- non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili il cui dominio è l'insieme vuoto, infatti questo è proprio il caso della funzione indefinita $\emptyset$ che diverge su ogni input. Quindi $I \neq \varnothing$;

- non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili il cui dominio non è l'insieme vuoto, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \mathbb{N}$.

Quindi per il teorema di Rice I non è decidibile.

Notiamo che il complemento $\bar{I} = \{i \in \mathbb{N} \mid Dom(\varphi_i) \neq \varnothing\}$ è un insieme semidecidibile tramite il seguente algoritmo:

```
begin
input(x);
decodifica x per ottenere la MdT M_x;
sia c = 0;
sia flag = True;
while flag do
    begin
    siano y = π_1(c); n = π_2(c);
    esegui M_x su input y per n passi;
    if M_x è in una configurazione finale then
        begin
        sia flag = False;
        output(SÌ)
        end
    else
        sia c = c + 1
    end
end
```

che descrive una funzione calcolabile secondo la tesi di Church-Turing. Dunque $\bar{I}$ è semidecidibile. Per il teorema di Post, dato che $\bar{I}$ è semidecidibile, I deve necessariamente essere un insieme non semidecidibile, altrimenti si avrebbe per assurdo che I è un insieme decidibile (mentre tramite il teorema di Rice abbiamo dimostrato che non può esserlo).

Esercizio 2.38 *(27/09/2024)*

Si dica se l'insieme

$$I = \{i \in \mathbb{N} \mid \varphi_i \text{ è una funzione iniettiva}\}$$

è un insieme decidibile. Si dica se l'insieme I è finito o infinito.

Soluzione 2.38

- Prima verifichiamo che l'insieme I rispetta le funzioni, ovvero che con $i \in I$ e j tale che $\varphi_i = \varphi_j$, si ha che $j \in I$. Dato che $i \in I$ si ha che φ_i è una funzione iniettiva, cioè $\forall x, y \in Dom(\varphi_i), \varphi_i(x) \neq \varphi_i(y)$. Dato che $\varphi_i = \varphi_j$ e $Dom(\varphi_i) = Dom(\varphi_j)$, allora $\forall x, y \in Dom(\varphi_j), \varphi_j(x) = \varphi_i(x) \neq \varphi_j(y) = \varphi_i(y)$ e quindi anche $j \in I$.

  Ora possiamo mostrare che l'insieme I non è decidibile applicando il teorema di Rice. L'insieme I rappresenta la famiglia F di funzioni calcolabili che sono iniettive. Dunque l'insieme I:

  - non corrisponde all'insieme vuoto, perché esistono funzioni calcolabili iniettive, per esempio la funzione identità $id(x) = x$. Quindi $I \neq \varnothing$;

  - non corrisponde all'insieme dei numeri naturali $\mathbb{N}$, perché esistono funzioni calcolabili che non sono iniettive, per esempio la funzione costante 0 $c_0(x) = 0$. Quindi $I \neq \mathbb{N}$.

  Quindi per il teorema di Rice I non è decidibile.

- L'insieme I è infinito. Difatti per ogni funzione calcolabile f esistono infinite macchine di Turing che la computano. Per convincersi di ciò è sufficiente considerare che data una MdT M che computa f (ne deve esistere almeno una dato che f è una funzione calcolabile), è sempre possibile aggiungere nuove istruzioni che non cambiano il suo output ma che comunque portano a nuove MdT $M', M'', \ldots$ con indice diverso da M.

Esercizio 2.39 *(27/09/2024)*

Studiare l'insieme:

$$I = \{x \in \mathbb{N} \mid \forall y \in \mathbb{N}, \varphi_y(x) \downarrow\}$$

ovvero, dire se I è ricorsivo, ricorsivamente enumerabile o non ricorsivamente enumerabile.

Soluzione 2.39

L'insieme I è l'insieme degli indici di funzione tali che, per ogni funzione calcolabile φ_y, l'indice è nel dominio della funzione. Questo insieme deve necessariamente essere vuoto, perché esiste una funzione calcolabile sempre indefinita che non ha nessun valore, e quindi nessun indice, nel proprio dominio. Dunque I è un insieme vuoto, e scrivere un algoritmo che lo decida è molto semplice:

```
begin
input(x);
output(NO)
end
```

Insiemi di indici

Un insieme A è un insieme di indici se per ogni $x, y \in \mathbb{N}$ tali che $x \neq y$ e $\varphi_x = \varphi_y$ si ha che $x \in A \iff y \in A$.
In altre parole, se $x \in A$ e $\varphi_x = \varphi_y$ per $x \neq y$, allora $y \in A$.

Capitolo 3

Dimostrazioni

Esercizio 3.1 (*20/06/2019, 17/06/2022, 09/06/2023, 17/09/2024*)
Enunciare formalmente e dimostrare la non calcolabilità del Problema dell'Arresto delle Macchine di Turing.

Esercizio 3.2 (*23/07/2019, 27/09/2021, 07/03/2022*)
Enunciare formalmente e dimostrare l'equivalenza tra Automi a Stati Finiti Deterministici e Automi a Stati Finiti non Deterministici.

Esercizio 3.3 (*13/09/2019, 28/09/2020, 09/02/2021, 03/10/2022, 23/06/2023, 08/07/2024*)
Enunciare formalmente e dimostrare il Teorema di Rice. Mostrare un esempio di insieme e l'applicazione del Teorema di Rice.

Esercizio 3.4 (*27/09/2019, 07/09/2022, 07/02/2023, 27/09/2024*)
Enunciare formalmente e dimostrare il Teorema di Post. Mostrare un esempio di applicazione del Teorema di Post.

Esercizio 3.5 (*07/02/2020*)
Enunciare formalmente la definizione di insieme ricorsivamente enumerabile. Si dimostri che ogni insieme ricorsivo è anche ricorsivamente enumerabile. In particolare, se non è vuoto (risp. non finito) può essere enumerato mediante una funzione calcolabile monotona crescente (risp. strettamente crescente).

Esercizio 3.6 (*05/03/2020, 19/09/2022, 06/02/2024, 26/02/2025*)
Enunciare formalmente l'esistenza della MdT Universale e darne la dimostrazione.

Esercizio 3.7 (*18/06/2020*)
Si enunci e si dimostri formalmente il teorema che lega gli insiemi semidecidibili, il dominio e il codominio delle funzioni calcolabili.

Esercizio 3.8 (*01/07/2020, 20/07/2020, 07/09/2020, 15/07/2021, 15/07/2022*)
Si enunci e si dimostri formalmente il pumping lemma per i linguaggi liberi dal contesto.

Esercizio 3.9 (*..., 15/11/2021, 27/06/2022, 28/02/2023, 20/11/2024*)
Si enunci e si dimostri formalmente il pumping lemma per i linguaggi regolari.

Esercizio 3.10 (*22/06/2021*)
Si elenchino caratterizzazioni degli insiemi ricorsivamente enumerabili e se ne diano le formali dimostrazioni.

Esercizio 3.11 (*01/07/2021*)
Si enunci e si dimostri formalmente il Teorema di Kleene.

Esercizio 3.12 (*06/09/2021, 21/02/2022, 12/07/2023, 24/07/2024*)
Enunciare i requisiti del concetto di algoritmo.

Esercizio 3.13 (*07/02/2022*)
Enunciare e dimostrare formalmente le relazioni tra insiemi ricorsivi, insiemi ricorsivamente enumerabili e insiemi non ricorsivamente enumerabili.

Esercizio 3.14 (*08/09/2023*)
Enunciare formalmente e dimostrare caratterizzazioni alternative degli insiemi ricorsivamente enumerabili.

Esercizio 3.15 (*19/09/2023, 28/09/2023*)
Enunciare formalmente e dimostrare la decidibilità, semidecidibilità o indecidibilità dell'insieme delle funzioni totali.

Esercizio 3.16 (*27/02/2024, 24/06/2024*)
Dare la definizione formale di Automa a Stati Finiti Deterministico e linguaggio accettato dall'automa.

Esercizio 3.17 (*05/02/2025*)
Dimostrare formalmente che dato un linguaggio L, se sia L, sia il suo complemento sono ricorsivamente enumerabili, allora L è ricorsivo.

Capitolo 4

Grammatiche e Automi

Esercizio 4.1 (*20/06/2019, 15/11/2021*)
Dato il linguaggio L su $\{0,1\}^\star$ costituito da stringhe tali che ogni 0 ha un 1 alla sua destra:

a) Dire a quale classe nella gerarchia di Chomsky il linguaggio l appartiene e trovare la grammatica corrispondente.

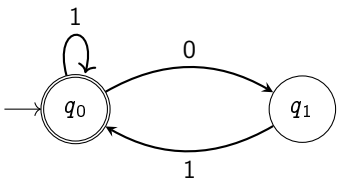
b) Trovare l'automa che lo riconosce.

Soluzione 4.1

a) L è un linguaggio regolare perché può essere generato dalla grammatica regolare $G = (N = \{S, A\}, T = \{0,1\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow 1S \mid 0A \mid 1 \mid \lambda$$
$$A \longrightarrow 1S \mid 1$$

b) Un automa a stati finiti (deterministico) che riconosce L è



Esercizio 4.2 (*23/07/2019*)
Sia L il linguaggio generato da una grammatica con le seguenti produzioni:

$$S \longrightarrow \lambda \mid aSc \mid SS$$

Dire a quale classe di linguaggi nella Gerarchia di Chomsky appartiene L. Può un sottoinsieme infinito di L essere regolare?

Soluzione 4.2

Possiamo stabilire che L sia almeno un linguaggio libero dal contesto (tipo 2) perché la grammatica che lo genera è di tipo 2. Per escludere che il linguaggio sia regolare, notiamo che il linguaggio $L' = \{a^n c^n \mid n \geq 1\}$, che non è regolare, è un sottoinsieme di L, dunque L non può essere regolare. La non regolarità di L' si può dimostrare tramite il Pumping lemma.
Assumiamo L' regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal Pumping lemma. Prendiamo dunque la stringa $z = a^p c^p$. Si ha che $z \in L'$ e $l(z) = 2p \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w. Dato che $l(uv) \leq p$, v può consistere solamente di a, e dato che $l(v) > 0$, v non può essere una stringa vuota. Quindi la stringa uv^2w ha più simboli a che b e non appartiene a L', e di conseguenza L' non è un linguaggio regolare.
Un altro sottoinsieme infinito di L è per esempio il linguaggio $L" = \{(ac)^n \mid n > 0\}$ che è un linguaggio regolare perché generato dalla grammatica regolare $G = (N = \{S, A\}, T = \{a, c\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow aA$$
$$A \longrightarrow cS \mid c$$

Esercizio 4.3 (*13/09/2019*)

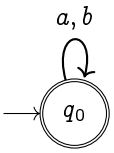
Sia L il linguaggio denotato dalla seguente espressione regolare $(a+b)^\star$. Scrivere una grammatica a struttura di frase che genera lo stesso linguaggio e l'automa che lo riconosce.

Soluzione 4.3

L è generato dalla grammatica a struttura di frase $G = (N = \{S\}, T = \{a, b\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow aS \mid bS \mid \lambda$$

Un automa a stati finiti (deterministico) che lo riconosce è



Esercizio 4.4 (*27/09/2019*)

Dire a quale classe nella Gerarchia di Chomsky appartiene il linguaggio

$$L = \{a^n a^n \mid a \in A, n \in \mathbb{N}\}$$

Soluzione 4.4

Il linguaggio genera tutte stringhe nella forma a^{2n} per $n \geq 0$ e $a \in A$, cioè

$$\lambda, aa, aaaa, \ldots$$

Quindi è un linguaggio regolare (tipo 3), perché lo si può generare tramite la grammatica regolare $G = (N = \{S\} \cup \{A_k \mid 1 \leq k \leq |A|\} \cup \{B_k \mid 1 \leq k \leq |A|\}, T = A, \mathcal{P}, S)$ con produzioni

$$\begin{aligned}
S &\longrightarrow a_1 A_1 \mid a_2 A_2 \mid \cdots \mid \lambda \\
A_1 &\longrightarrow a_1 B_1 \mid a_1 \\
B_1 &\longrightarrow a_1 A_1 \\
A_2 &\longrightarrow a_2 B_2 \mid a_2 \\
B_2 &\longrightarrow a_2 A_2 \\
&\vdots \\
A_k &\longrightarrow a_k B_k \mid a_k \\
B_k &\longrightarrow a_k A_k
\end{aligned}$$

Esercizio 4.5 *(16/12/2019)*

Si consideri la seguente grammatica $G = (N = \{S, X\}, T = \{a, b, c\}, \mathcal{P}, S)$ con produzioni

$$\begin{aligned}
S &\longrightarrow X \mid \lambda \\
X &\longrightarrow aXa \mid bXb \mid c
\end{aligned}$$

Svolgere ciascuno dei seguenti punti:

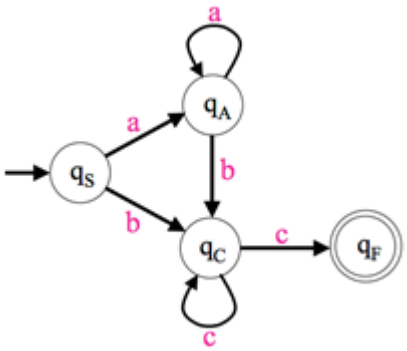
1. Di che tipo è la grammatica?
2. Mostrare una derivazione per “aaba”.
3. Mostrare una derivazione per “abbcbba”.
4. Sono ammesse stringhe vuote?
5. Qual è il linguaggio generato da G?

Soluzione 4.5

1. La grammatica è libera dal contesto (tipo 2).
2. $S \to X \to aXa \to \ldots$ non è possibile.
3. $S \to X \to aXa \to abXba \to abbXbba \to abbcbba$.
4. Sì.
5. $L = \{\alpha \cdot c \cdot \alpha^{rev} \mid \alpha \in \{a, b\}^*\} \cup \{\lambda\}$

Esercizio 4.6 *(22/06/2017, 16/12/2019)*

Dato il seguente ASFND:



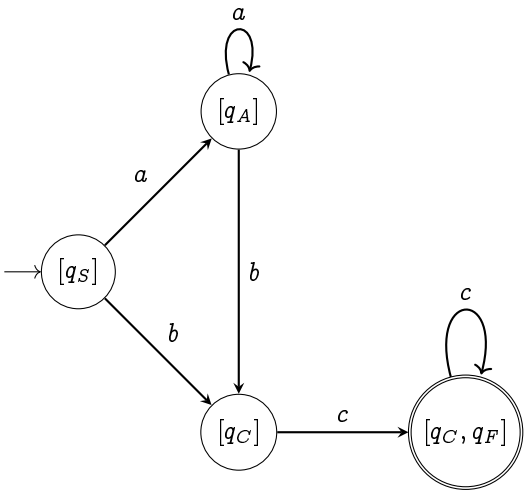
- Determinare il corrispondente ASFD.
- Definire il linguaggio riconosciuto.

Soluzione 4.6

- La matrice di transizione dell'ASFD è

	a	b	c
$[q_S]$	$\{q_A\}$	$\{q_C\}$	
$[q_A]$	$\{q_A\}$	$\{q_C\}$	
$[q_C]$			$\{q_C, q_F\}$
$[q_C, q_F]$			$\{q_C, q_F\}$

con stato finale $[q_C, q_F]$. Il grafo di transizione corrispondente è



- $L = \{a^n b c^m \mid n \geq 0, m > 0\}$

Esercizio 4.7 *(07/02/2020, 06/09/2021)*

Scrivere una grammatica a struttura di frase (06/09/2021: regolare) sull'alfabeto $\{0, 1\}$, che denota il linguaggio L formato da tutte le stringhe che contengono la stringa 00 come prefisso oppure la stringa 110 come suffisso.

- Costruire l'automa a stati finiti che riconosce L.

- Definire una espressione regolare che definisce L.

Soluzione 4.7

La grammatica a struttura di frase è $G = (N = \{S, A\}, T = \{0, 1\}, \mathcal{P}, S)$ con produzioni

$$\begin{aligned} S &\longrightarrow 00A \mid A110 \\ A &\longrightarrow 0A \mid 1A \mid \lambda \end{aligned}$$

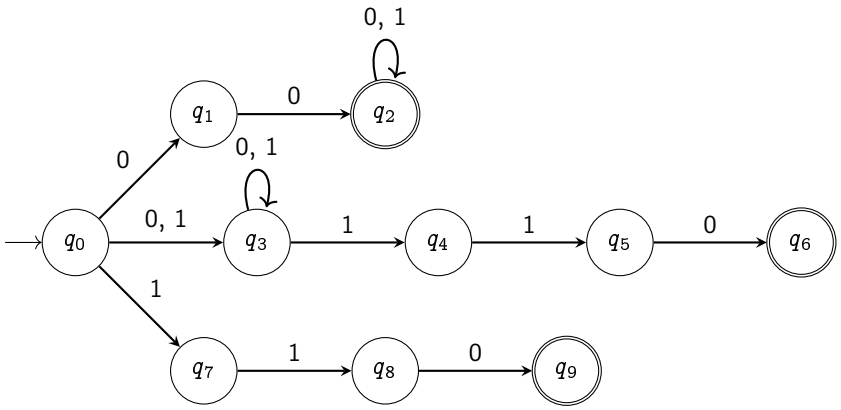
La grammatica regolare è $G = (N = \{S, A, B, C, D, E, F\}, T = \{0, 1\}, \mathcal{P}, S)$ con produzioni

$$\begin{aligned} S &\longrightarrow 0A \mid 1B \\ A &\longrightarrow 0C \mid 1B \mid 0 \\ B &\longrightarrow 1D \mid 0F \\ C &\longrightarrow 0C \mid 1C \mid 0 \mid 1 \\ D &\longrightarrow 1D \mid 0E \mid 0 \\ E &\longrightarrow 1B \mid 0F \\ F &\longrightarrow 0F \mid 1B \end{aligned}$$

oppure $G = (N = \{S, A, B, C, D, E, F, G\}, T = \{0, 1\}, \mathcal{P}, S)$ con produzioni

$$\begin{aligned} S &\longrightarrow 0A \mid 1C \mid 0C \mid 1F \\ A &\longrightarrow 0B \mid 0 \\ B &\longrightarrow 0B \mid 1B \mid 0 \mid 1 \\ C &\longrightarrow 0C \mid 1C \mid 1D \\ D &\longrightarrow 1E \\ E &\longrightarrow 0 \\ F &\longrightarrow 1G \\ G &\longrightarrow 0 \end{aligned}$$

- Un automa a stati finiti (non deterministico) che riconosce L è



- Un' espressione regolare che denota L è

$$00 \cdot (0+1)^\star + (0+1)^\star \cdot 110$$

Esercizio 4.8 (*05/03/2020*)

Quale tipologia di linguaggi sono generati da grammatiche con produzioni della forma:

$$A \longrightarrow wB$$
$$B \longrightarrow w$$

dove A e B sono simboli non terminali e w è una stringa di simboli terminali su un datoalfabeto.

Soluzione 4.8

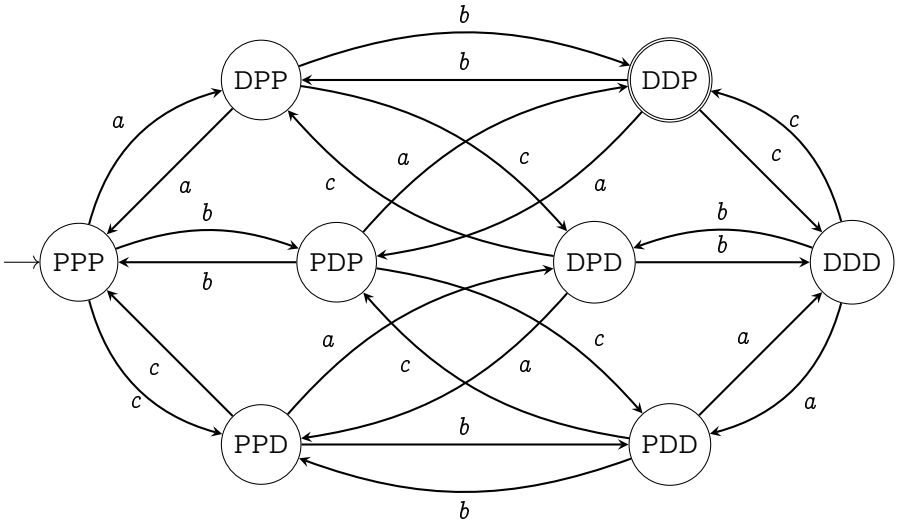
Le grammatiche con questo tipo di produzioni e con $l(w) = 1$ sono grammatiche regolari, quindi i linguaggi che generano sono linguaggi regolari (tipo 3). Anche nel caso $l(w) > 1$, sebbene la grammatica in tal caso non sia regolare, è possibile costruire una grammatica regolare che generi lo stesso linguaggio. Per fare ciò basta notare che è sufficiente sostituire ogni produzione della forma $A \longrightarrow a\alpha$ tale che $l(\alpha) > 0$ con $A \longrightarrow aX$ e $X \longrightarrow \alpha$, dove X è un nuovo simbolo non terminale, e quindi ripetere questo procedimento finché tutte le produzioni non sono della forma delle grammatiche regolari.

Esercizio 4.9 (*18/06/2020*)

Dire a quale classe appartiene il linguaggio di tutte le stringhe sull'alfabeto $\{a, b, c\}$ che contengono un numero dispari di a, un numero dispari di b e un numero pari di c.

Soluzione 4.9

Il linguaggio è regolare perché può essere riconosciuto dal seguente automa a stati finiti deterministico



L'automa codifica gli 8 possibili stati nella quale la stringa si può trovare: tutte le combinazioni da a pari, b pari, c pari (lo stato PPP) a a dispari, b dispari, c dispari (lo stato DDD). Lo stato iniziale sarà lo stato che rappresenta una stringa con tutti simboli pari, perché non avendo letto ancora nulla abbiamo 0 a, 0 b e 0 c (0 è un numero pari) mentre lo stato finale sarebbe quello che rappresenta stringhe con un numero dispari di a, dispari di b e pari di c (ciè lo stato DDP). Dopo aver definito gli stati basta tracciare i 24 archi (3 per ogni stato) per ogni possibile lettera che può essere letta.

Esercizio 4.10 (*01/07/2020, 20/07/2020*)

Si consideri l'insieme di tutte le stringhe di parentesi bilanciate. Esempio: le stringhe diparentesi (( )), ( )( ), (( ) (( ))) sono stringhe di parentesi bilanciate. Dire a quale classe di linguaggi appartiene l'insieme di stinghe di parentesi bilanciate.

Soluzione 4.10

Mostriamo che L non è un linguaggio regolare applicando il Pumping lemma.
Assumiamo L regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Prendiamo dunque la stringa $z = (^p)^p$. Si ha che $z \in L$ e $l(z) = 2p \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w.
Dato che $l(uv) \leq p$, v può consistere solamente di parentesi aperte, e dato che $l(v) > 0$ v non può essere una stringa vuota. Quindi la stringa uv^2w ha più parentesi aperte che chiuse e non appartiene a L, e di conseguenza L non è un linguaggio regolare.
Mostriamo ora che L è un linguaggio libero dal contesto definendo una grammatica libera dal contesto che lo generi, per esempio $G = (N = \{S\}, T = \{(,)\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow (S) \mid SS \mid ()$$

Esercizio 4.11 (*07/09/2020*)

Si consideri l'insieme di tutte le stringhe bilanciate 'begin' e 'end'. Esempio: le stringhe "begin begin end end", "begin end begin end" , "begin begin end begin begin end end end" sono stringhe bilanciate di 'begin' e 'end'. Dire a quale classe di linguaggi appartiene l'insieme di stinghe di 'begin' e 'end' bilanciate.

Soluzione 4.11

Mostriamo che L non è un linguaggio regolare applicando il Pumping lemma.
Assumiamo L regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Prendiamo dunque la stringa $z = \text{begin}^p\text{end}^p$. Si ha che $z \in L$ e $l(z) = 2p \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w.
Dato che $l(uv) \leq n$, v può consistere solamente di simboli 'begin', e dato che $l(v) > 0$ v non può essere una stringa vuota. Quindi la stringa uv^2w ha più simboli 'begin' che simboli 'end" e non appartiene a L, e di conseguenza L non è un linguaggio regolare.
Mostriamo ora che L è un linguaggio libero dal contesto definendo una grammatica libera dal contesto che lo generi, per esempio $G = (N = \{S\}, T = \{(,)\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow \text{begin}S\text{end} \mid SS \mid \text{begin end}$$

Esercizio 4.12 (*28/09/2020*)

A quale classe di linguaggi appartiene l'insieme di stringhe:

$$L = \{0^n 1^m 0^{m+n} \mid m, n \geq 0\}$$

Soluzione 4.12

Mostriamo che L non è un linguaggio regolare applicando il Pumping lemma.
Assumiamo L regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Prendiamo dunque la stringa $z = 0^p 1^p 0^{2p}$. Si ha che $z \in L$ e $l(z) = 4p \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w.
Dato che $l(uv) \leq p$, v può consistere solamente di zeri, e dato che $l(v) > 0$ v non può essere una stringa vuota. Quindi pompando nuovi zeri in v, per esempio con uv^3w, il numero di zeri nel prefisso di z aumenterebbe ma il numero di zeri nel suffisso di z rimarebbe invariato, violando quindi il vincolo che gli zeri in fondo alla stringa devono essere uguali al numero di uni e di zeri all'inizio della stringa. Quindi L non è un linguaggio regolare.
Mostriamo ora che L è un linguaggio di tipo 2 definendo una grammatica libera dal contesto che lo generi, per esempio la grammatica $G = (N = \{S, A, B\}, T = \{0, 1\}, \mathcal{P}, S)$ con produzioni

$$\begin{aligned} S &\longrightarrow A \mid B \mid 00 \mid 10 \mid \lambda \\ A &\longrightarrow 0A0 \mid 0B0 \mid 00 \\ B &\longrightarrow 1B0 \mid 10 \end{aligned}$$

Esercizio 4.13 (*09/02/2021*)

Si consideri la seguente grammatica $G = (N = \{S, X, Y\}, T = \{a, b, c\}, \mathcal{P}, S)$ e le seguenti produzioni $\mathcal{P}$:

$$S \longrightarrow X \mid \lambda$$
$$X \longrightarrow aXa \mid Y$$
$$Y \longrightarrow bYb \mid c$$

Svolgere ciascuno dei seguenti punti:

1. A quale classe di grammatica appartiene la grammatica G?
2. Mostrare, se esiste, una derivazione per "aaba".
3. Mostrare, se esiste, una derivazione per "abbcbba".
4. É ammessa la stringa vuota?
5. Quale è il linguaggio generato da G?

Soluzione 4.13

1. La grammatica è libera dal contesto (tipo 2).
2. $S \to X \to aXa \to aYa \to \dots$ non è possibile.
3. $S \to X \to aXa \to aYa \to abYba \to abbYbba \to abbcbba$.
4. Sì.
5. $L = \{a^n b^m c b^m a^n \mid n \geq 0, m \geq 0\} \cup \{\lambda\}$

Esercizio 4.14 (*26/02/2021*)

Sia L il linguaggio generato dalla seguente espressione regolare $(a + b)^\star$.

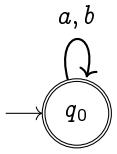
- Dire a quale classe di linguaggi appartiene L.
- Scrivere una grammatica a struttura di frase che genera L.
- Scrivere l'automa che riconosce L.

Soluzione 4.14

- L è un linguaggio regolare (tipo 3), perché è definito da una espressione regolare.
- L si può generare con la seguente grammatica a struttura di frase $G = (N = \{S\}, T = \{a, b\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow aS \mid bS \mid \lambda$$

- Un automa a stati finiti (deterministico) che riconosce L è



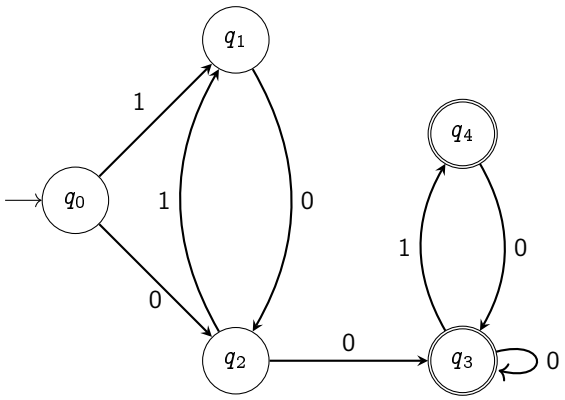
Esercizio 4.15 (*22/06/2021*)

Dire a quale classe di linguaggi sull'alfabeto $\{0, 1\}$ appartengono gli insiemi di stringhe tali che le occorrenze di 0 e di 1 soddisfano i seguenti requisiti:

- ci sono almeno due 0 consecutivi e non vi sono mai due 1 consecutivi;
- il numero di occorrenze di 0 è uguale al numero di occorrenze di 1.

Soluzione 4.15

- Il linguaggio è regolare, perché riconosciuto dal seguente automa a stati finiti deterministico:



- Per prima cosa, mostriamo che il linguaggio non è regolare applicando il Pumping lemma.

  Assumiamo L regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Prendiamo dunque la stringa $z = 0^p 1^p$. Si ha che $z \in L$ e $l(z) = 2p \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w.

  Dato che $l(uv) \leq p$, v può consistere solamente di zeri, e dato che $l(v) > 0$, v non può essere una stringa vuota. Quindi pompando nuovi zeri in v, per esempio con uv^2w, il numero di zeri aumenta rispetto al numero di uni, violando il vincolo che le loro occorrenze devono essere in egual numero. Quindi la stringa uv^2w non appartiene a L, e di conseguenza L non è un linguaggio regolare.

  Mostriamo ora che L è un linguaggio di tipo 2 definendo una grammatica libera dal contesto che lo generi, per esempio la grammatica $G = (N = \{S\}, T = \{0, 1\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow 1S0 \mid 0S1 \mid SS \mid \lambda$$

Esercizio 4.16 *(01/07/2021)*

Dire a quale classe di linguaggi sull'alfabeto $A = \{0, 1\}$ appartiene l'insieme di stringhe:

$$L = \{x \in A^\star \mid x \text{ è palindroma}\}$$

NB: Ricordiamo che una stringa è palindroma se è indifferente leggerla da destra a sinistra o da sinistra verso destra.

Soluzione 4.16

Mostriamo che L non è un linguaggio regolare applicando il Pumping lemma.
Assumiamo L regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Prendiamo dunque la stringa $z = 0^p 1 0^p$. Si ha che $z \in L$ e $l(z) = 2p + 1 \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w.
Dato che $l(uv) \leq p$, v può consistere solamente di zeri, e dato che $l(v) > 0$, v non può essere una stringa vuota. Quindi pompando nuovi zeri in v, per esempio con uv^2w, il numero di zeri a sinistra del simbolo 1 in z aumenterebbe ma il numero di zeri a destra del simbolo 1 rimarebbe invariato, rompendo la simmetria della stringa che quindi cessa di essere palindroma. Quindi L non è un linguaggio regolare.
Mostriamo ora che L è un linguaggio di tipo 2 definendo una grammatica libera dal contesto che lo generi, per esempio la grammatica $G = (N = \{S\}, T = \{0, 1\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \lambda$$

Esercizio 4.17 *(15/07/2021, 21/02/2022, 06/02/2024)*

Dire a quale classe di linguaggi sull'alfabeto $A = \{0, 1, \ldots, 9\}$ appartiene l'insieme di stringhe:

$$L = \{x \in A^\star \mid x \mod 5 = 0\}$$

NB: $\mod$ rappresenta l'operazione che restituisce il resto della divisione intera tra i numeri a cui è applicata.

Soluzione 4.17

Il linguaggio L è regolare, perché generato dalla grammatica regolare $G = (N = \{S, A\}, T = \{0, 1, \ldots, 9\}, \mathcal{P}, S)$ con produzioni

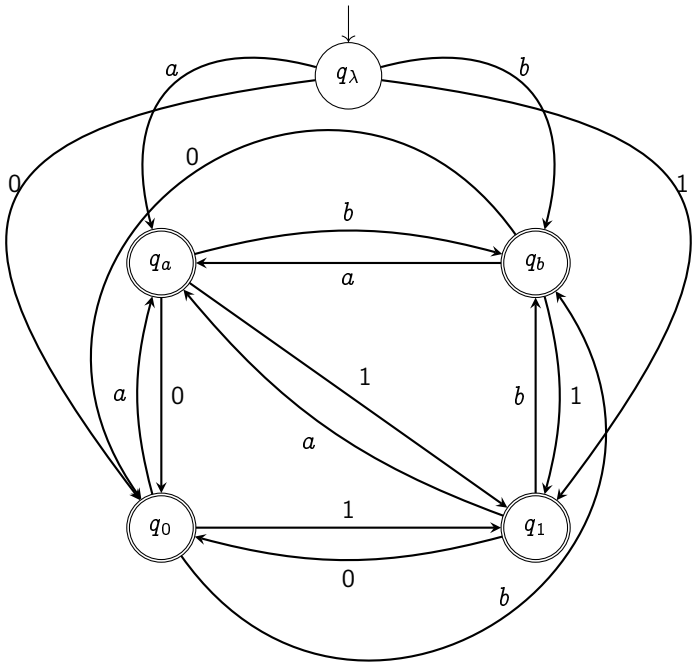
$$S \longrightarrow 0S \mid 1S \mid \ldots \mid 9S \mid 0 \mid 5$$

Esercizio 4.18

Si consideri il linguaggio L sull'alfabeto $\{a, b, 0, 1\}$ contenente tutte le parole non vuote tali che mai occorrono due simboli consecutivi uguali. A quale classe appartiene il linguaggio L?

Soluzione 4.18

Il linguaggio L è regolare, perché può essere riconosciuto da un automa a stati finiti deterministico. Per esempio, è riconosciuto dal seguente ASFD



dove ad ogni stato associamo i seguenti significati:

- q_λ: il carattere precedente è λ, che è la situazione solo nello stato iniziale dell'automa, e come da specifica del linguaggio non può essere lo stato finale.
- q_a: il carattere precedente nella stringa è a, e da questo stato si possono solo leggere simboli non a.
- q_b: il carattere precedente nella stringa è b, e da questo stato si possono solo leggere simboli non b.
- q_0: il carattere precedente nella stringa è 0, e da questo stato si possono solo leggere simboli non 0.
- q_1: il carattere precedente nella stringa è 1, e da questo stato si possono solo leggere simboli non 1.

Esercizio 4.19 (*07/02/2022*)

Si consideri il linguaggio L sull'alfabeto $\{0,1\}$ contenente tutte e sole le stringhe tali che il penultimo simbolo è 0. A quale classe appartiene il linguaggio L?

Soluzione 4.19

L è un linguaggio regolare (di tipo 3) perché generato dalla grammatica regolare $G = (N = \{S, A\}, T = \{0,1\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow 1S \mid 0A$$
$$A \longrightarrow 0A \mid 1S \mid 0 \mid 1$$

Esercizio 4.20 (*07/03/2022*)

Dire a quale classe di linguaggi sull'alfabeto $A = \{0,1\}$ appartiene l'insieme di stringhe:

$$L = \{0^n 1^m \mid n, m \in \mathbb{N} \wedge n < m\}$$

Soluzione 4.20

Iniziamo mostrando che L non è un linguaggio regolare applicando il Pumping lemma.
Assumiamo L regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Prendiamo dunque la stringa $z = 0^p 1^{p+1}$. Si ha che $z \in L$ e $l(z) = 2p + 1 \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w.
Dato che $l(uv) \leq p$, v può consistere solamente di zeri, e dato che $l(v) > 0$, v non può essere una stringa vuota. Quindi pompando nuovi zeri in v, per esempio con uv^2w, il numero di zeri diventa necessariamente pari o superiore al numero di uni; nel caso peggiore che v sia un singolo zero, con v^2 la nuova stringa ha $n = m$ che quindi non può appartenere al linguaggio L. Quindi L non è un linguaggio regolare.
Mostriamo ora che L è un linguaggio di tipo 2 definendo una grammatica libera dal contesto che lo generi, per esempio $G = (N = \{S\}, T = \{0,1\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow 0S1 \mid S1 \mid 1$$

Esercizio 4.21 (*17/06/2022*)

Dato il linguaggio L su $\{0,1\}^\star$ costituito da stringhe di 0 e di 1 alternati:

a) Dire a quale classe della gerarchia di Chomsky in linguaggio L appartiene e trovare la grammatica corrispondente

b) Trovare l'automa che lo riconosce

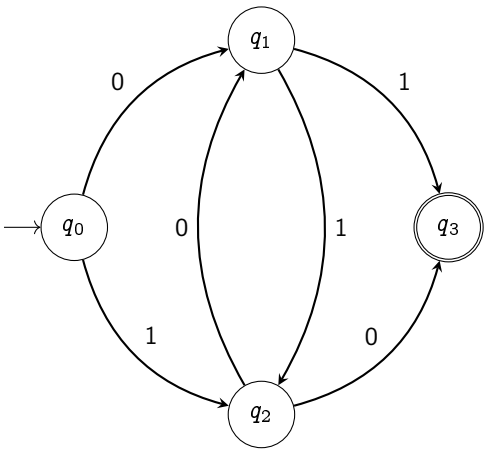
c) Trovare una espressione regolare che lo riconosce

Soluzione 4.21

a) Il linguaggio è regolare, perché può essere generato dalla grammatica regolare $G = (N = \{S, A, B\}, T = \{0, 1\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow 0A \mid 1B$$
$$A \longrightarrow 1B \mid 1$$
$$B \longrightarrow 0A \mid 0$$

b) Un automa (non deterministico) che riconosce L è



c) Un'espressione regolare che denota L è $10 \cdot (10)^\star \cdot (1 + \lambda) + 01 \cdot (01)^\star \cdot (0 + \lambda)$

Esercizio 4.22 (*27/06/2022*)

Dato il linguaggio L su $\{0, 1\}^\star$ costituito da stringhe che rappresentano un numero dispari in notazione binaria.

a) Dire a quale classe della gerarchia di Chomsky in linguaggio L appartiene e trovare la grammatica corrispondente

b) Trovare l'automa che lo riconosce

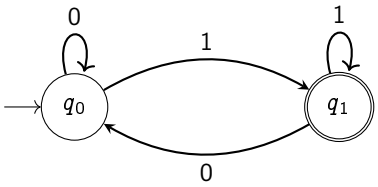
c) Trovare una espressione regolare che lo riconosce

Soluzione 4.22

a) Il linguaggio è regolare, perché può essere generato dalla grammatica regolare $G = (N = \{S\}, T = \{0, 1\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow 0S \mid 1S \mid 1$$

b) Un automa (deterministico) che riconosce L è



c) Un'espressione regolare che denota L è $(0+1)^\star \cdot 1$

Esercizio 4.23 (*15/07/2022*)

Dato il linguaggio L su $\{0,1\}^\star$ costituito dalle stringhe che contengono 01:

a) Dire a quale classe della gerarchia di Chomsky in linguaggio L appartiene e trovare la grammatica corrispondente

b) Trovare l'automa che lo riconosce

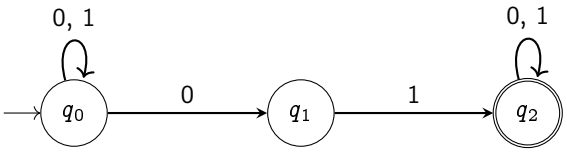
c) Trovare una espressione regolare che lo riconosce

Soluzione 4.23

a) Il linguaggio è regolare, perché può essere generato dalla grammatica regolare $G = (N = \{S, A, B\}, T = \{0,1\}, \mathcal{P}, S)$ con produzioni

$$\begin{aligned} S &\longrightarrow 0S \mid 1S \mid 0A \\ A &\longrightarrow 1B \mid 1 \\ B &\longrightarrow 0B \mid 1B \mid 0 \mid 1 \end{aligned}$$

b) Un automa (non deterministico) che riconosce L è



c) Un'espressione regolare che denota L è $(0+1)^\star \cdot 01 \cdot (0+1)^\star$

Esercizio 4.24 (*07/09/2022*)

Dire a quale classe nella Gerarchia di Chomsky appartiene il linguaggio:

$$L = \{0^n 1^n 0^m 1^m \mid n, m \in \mathbb{N}\}$$

Dire se il linguaggio L è esprimibile nel linguaggio delle espressioni regolari.

Soluzione 4.24

- Mostriamo che L non è un linguaggio regolare applicando il Pumping lemma.

  Assumiamo L regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Prendiamo dunque la stringa $z = 0^p 1^p 0^p 1^p$. Si ha che $z \in L$ e $l(z) = 4p \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w.

  Dato che $l(uv) \leq p$, v può consistere solamente di zeri, e dato che $l(v) > 0$, v non può essere una stringa vuota. Quindi pompando nuovi zeri in v, per esempio con uv^2w, il numero di zeri nella prima sequenza di zeri non è più uguale al numero di uni immediatamente seguenti. Quindi la stringa uv^2w non appartiene a L, e di conseguenza L non è un linguaggio regolare.

  Mostriamo ora che L è un linguaggio di tipo 2 definendo una grammatica libera dal contesto che lo generi, per esempio, $G = (N = \{S, A\}, T = \{0, 1\}, \mathcal{P}, S)$ con produzioni

  $$S \longrightarrow A \mid AA \mid \lambda$$
  $$A \longrightarrow 0A1 \mid 01$$

- Il linguaggio non è esprimibile come espressione regolare, perché la potenza espressiva delle espressioni regolari coincide con quello delle grammatiche regolari, mentre il linguaggio è stato dimostrato non essere regolare.

Esercizio 4.25 (*19/09/2022*)

Si fissi un $k \in \mathbb{N}$ a scelta. Dire a quale classe della Gerarchia di Chomsky appartiene il linguaggio $L = \{0^n 1^n \mid n \leq k\}$
Dire se il linguaggio L è esprimibile nel linguaggio delle espressioni regolari.

Soluzione 4.25

- Il linguaggio L è regolare perché è un linguaggio finito. Infatti, sarà un linguaggio finito per qualunque $k \in \mathbb{N}$ si scelga: se $k = 0$, allora $L = \{\lambda\}$; se $k = 1$, allora $L = \{\lambda, 01\}$, e così via.

  In ogni caso, fissato $k = 2$, mostriamo una grammatica regolare che genera L: la grammatica $G = (N = \{S, A, B, C\}, T = \{0, 1\}, \mathcal{P}, S)$ con produzioni

  $$S \longrightarrow 0A \mid \lambda$$
  $$A \longrightarrow 0B \mid 1$$
  $$B \longrightarrow 1C$$
  $$C \longrightarrow 1$$

- Dato che per ogni k fissato il linguaggio è regolare, allora è anche esprimibile tramite espressione regolare. Per esempio con $k = 2$, l'espressione regolare che denota L è $\lambda + 01 + 0011$

Esercizio 4.26 (*03/10/2022*)

Si fissi un $k \in \mathbb{N}$ a scelta. Dire a quale classe della Gerarchia di Chomsky appartiene il linguaggio

$$L = \{a^n b^m c^n \mid n \leq k, m \in \mathbb{N}\}$$

Dire se il linguaggio L è esprimibile nel linguaggio delle espressioni regolari.

Soluzione 4.26

- Mostriamo una soluzione per $k = 0$ e $k = 1$, in ogni caso la classe del linguaggio non è influenzata dalla scelta k: il linguaggio è regolare in tutti i casi.

  ($k = 0$) La grammatica è $G = \{N = \{S\}, T = \{a, b, c\}, \mathcal{P}, S\}$ con produzioni

  $$S \longrightarrow bS \mid b \mid \lambda$$

  ($k = 1$) La grammatica è $G = \{N = \{S, A, B\}, T = \{a, b, c\}, \mathcal{P}, S\}$ con produzioni

  $$S \longrightarrow aA \mid bB \mid \lambda$$
  $$A \longrightarrow bA \mid c$$
  $$B \longrightarrow bB \mid b$$

- Dato che per ogni k fissato il linguaggio è regolare, allora è anche esprimibile tramite espressioni regolari. Per esempio con $k = 0$, un'espressione regolare che denota L è $b^\star$, mentre con $k = 1$ è $b^\star + a \cdot b^\star \cdot c$

Esercizio 4.27 (*07/02/2023*)

Si consideri la grammatica $G = (N = \{S, A, B\}, T = \{a, b\}, \mathcal{P}, S)$ e le seguenti produzioni in $\mathcal{P}$

$$S \longrightarrow A \mid bB$$
$$A \longrightarrow bA \mid aB \mid a$$
$$B \longrightarrow bB \mid b \mid a$$

Svolgere ciascuno dei seguenti punti:

- Mostrare una derivazione per aba, se possibile
- Mostrare una derivazione per $abab$, se possibile
- Qual è il linguaggio generato da G?
- A quale classe appartiene il linguaggio generato da G?
- A quale classe appartiene la grammatica G?

Soluzione 4.27

- $S \to A \to aB \to abB \to aba$
- $S \to A \to aB \to abB \to \dots$ non è possibile
- Il linguaggio è $\{b^n a^i b^m a^j \mid n, m \geq 0, i, j \in \{0, 1\}\} \setminus \{\lambda\}$
- Il linguaggio generato da questa grammatica è regolare, perché si può generare tramite una grammatica regolare. Per esempio $G = (N = \{S, A, B\}, T = \{a, b\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow bS \mid aB \mid bB \mid a$$
$$B \longrightarrow bB \mid b \mid a$$

- La grammatica è libera dal contesto (tipo 2), perché ha una produzione $S \longrightarrow A$ che non è ammessa dalle grammatiche regolari.

Esercizio 4.28 (*28/02/2023*)

Sia L il linguaggio generato dalla seguente espressione regolare

$$E = a^\star(ba^\star)^\star$$

Svolgere ciascuno dei seguenti punti:

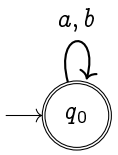
- Qual è il linguaggio denotato da E?
- A quale classe appartiene il linguaggio denotato da E?
- Scrivere una grammatica a struttura di frase che genera lo stesso linguaggio
- Scrivere l'automa che accetta lo stesso linguaggio

Soluzione 4.28

- Il linguaggio denotato da E è $\{a, b\}^\star$
- Il linguaggio denotato da E è regolare, perché le espressioni regolari possono solo descrivere linguaggi regolari.
- Una grammatica a struttura di frase che generi il linguaggio denotato da E è $G = (N = \{S\}, T = \{a, b\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow aS \mid bS \mid \lambda$$

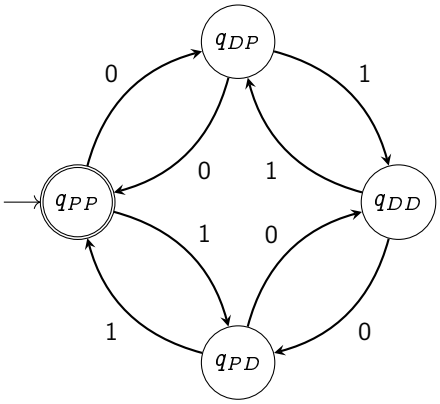
- Un automa che accetta il linguaggio denotato da E è



Esercizio 4.29 (*09/06/2023*)

Scrivere un automa a stati finiti che accetta tutte e sole le stringhe con un numero pari di 0 e un numero pari di 1. Dire a qualche classe appartiene il linguaggio accettato da tale automa.

Soluzione 4.29



dove ad ogni stato associamo i seguenti significati:

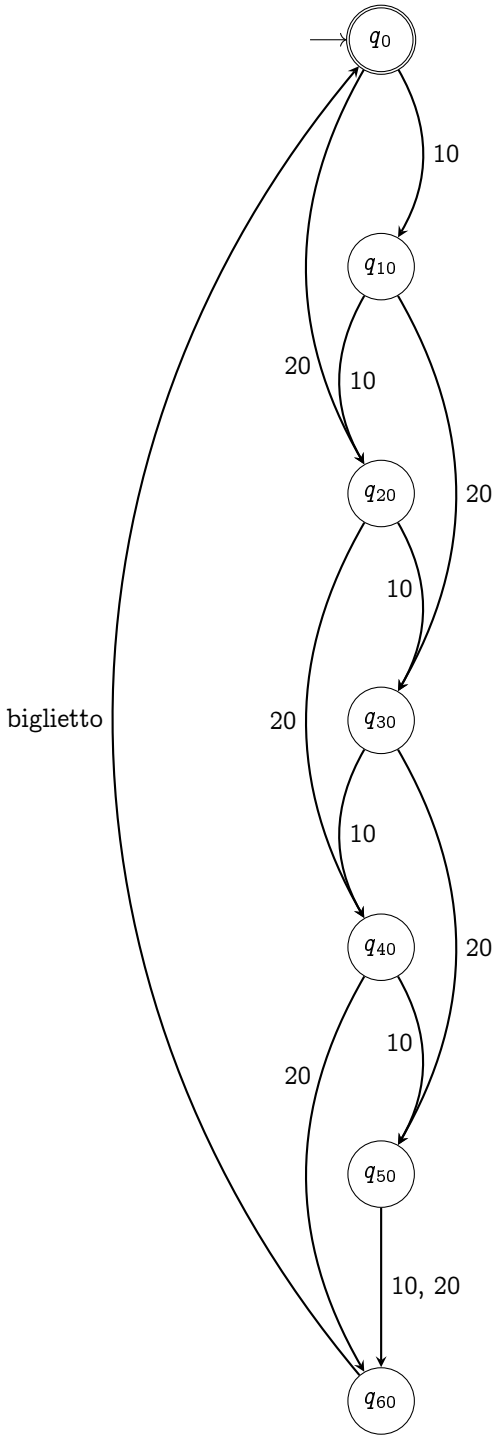
- q_{PP}: il numero di zeri e uni della stringa sono entrambi pari.
- q_{DP}: il numero di zeri nella stringa è dispari mentre il numero di uni è pari.
- q_{PD}: il numero di zeri nella stringa è pari mentre il numero di uni è dispari.
- q_{DD}: il numero di zeri e uni della stringa sono entrambi dispari.

Il linguaggio accettato da un automa a stati finiti è regolare, e quindi anche linguaggio accettato dal precedente automa è regolare.

Esercizio 4.30 (*23/06/2023*)

Scrivere un automa a stati finiti che emette un biglietto dopo che sono stati inseriti 0.60 Euro. L'automa accetta monete da 10 o da 20 centesimi di Euro e non fornisce resto. Dire a qualche classe appartiene il linguaggio accettato da tale automa.

Soluzione 4.30



dove ad ogni stato associamo i seguenti significati:

- q_0: il saldo corrente è 0 Euro.

- q_{10}: il saldo corrente è 0.10 Euro.
- q_{20}: il saldo corrente è 0.20 Euro.
- q_{30}: il saldo corrente è 0.30 Euro.
- q_{40}: il saldo corrente è 0.40 Euro.
- q_{50}: il saldo corrente è 0.50 Euro.
- q_{60}: il saldo corrente è almeno 0.60 Euro e l'automa deve emettere un biglietto senza dare resto.

Il linguaggio accettato da un automa a stati finiti è regolare, e quindi anche linguaggio accettato dal precedente automa è regolare.

Esercizio 4.31 (*12/07/2023*)

Dire a quale classe di linguaggi su $\{a, b\}$ appartiene l'insieme di stringhe

$$L = \{a^n b^m \mid n > m\}$$

Soluzione 4.31

Iniziamo mostrando che L non è un linguaggio regolare applicando il Pumping lemma.
Assumiamo L regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Prendiamo dunque la stringa $z = a^{p+1}b^p$. Si ha che $z \in L$ e $l(z) = 2p + 1 \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w.
Dato che $l(uv) \leq p$, v può consistere solamente di a, e dato che $l(v) > 0$, v non può essere una stringa vuota. In questo caso per ottenere una stringa che non appartenga a L dobbiamo in un qualche modo rimuovere almeno un simbolo a. Questo si può fare solamente se scegliamo 0 come esponente di v, ottenendo la stringa $z' = uv^0w = uw$. Così abbiamo necessariamente rimosso almeno una a e la stringa che si ottiene non può appartenere al linguaggio L, perché le a non sono più in un numero maggiore delle b. Quindi L non è un linguaggio regolare.
Mostriamo ora che L è un linguaggio di tipo 2 definendo una grammatica libera dal contesto che lo generi, per esempio $G = (N = \{S\}, T = \{a, b\}, \mathcal{P}, S)$ con produzioni

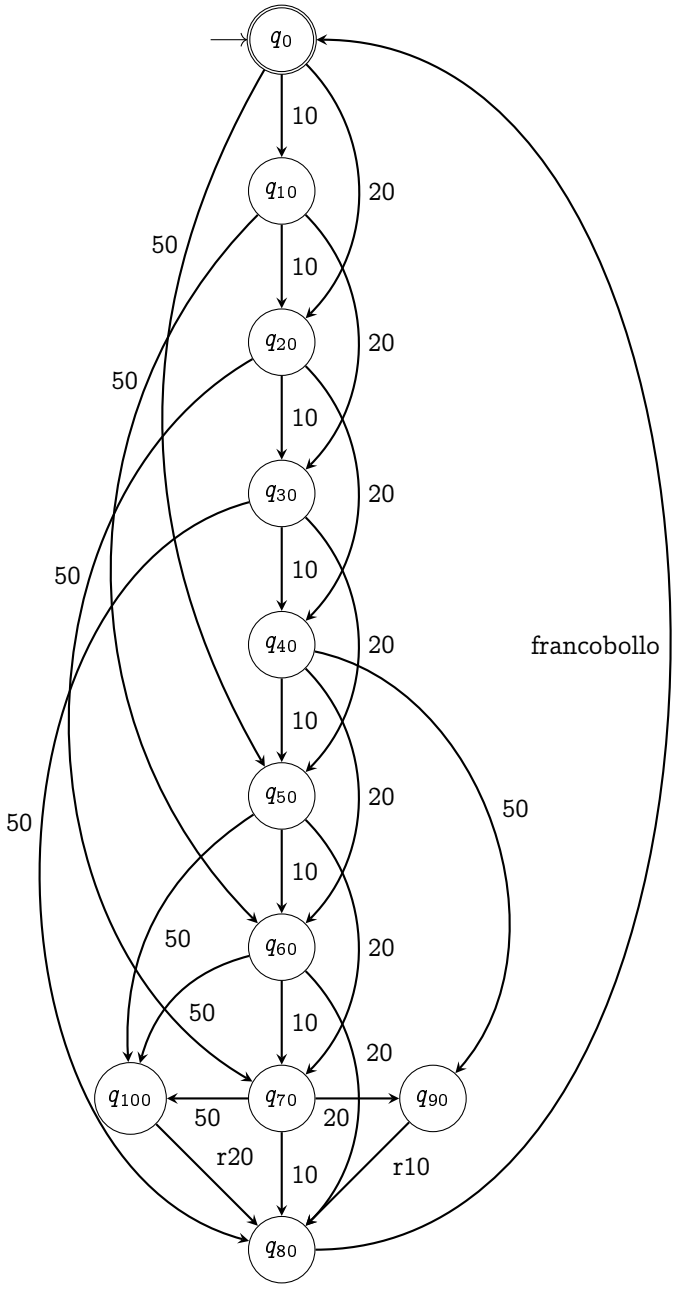
$$S \longrightarrow aSb \mid aS \mid a$$

Esercizio 4.32 (*08/09/2023, 19/09/2023, 28/09/2023*)

Progettare un automa a stati finiti che realizzi un distributore automatico di francobolli, accettando monete da 50, 20 e 10 centesimi di Euro. Il prezzo del francobollo è di Euro 0.80 e il distributore può dare un resto di massimo 20 centesimi.

Soluzione 4.32

Nell'automa indichiamo con $r10$ e $r20$ il resto del distributore. Notare che negli stati q_{90} e q_{100} si restituisce il resto all'utente per poi transizionare nello stato q_{80}, quello dove il distributore restituisce il francobollo senza resto.



Ad ogni stato associamo i seguenti significati:

- q_0: il saldo corrente è 0 Euro.
- q_{10}: il saldo corrente è 0.10 Euro.

- q_{20}: il saldo corrente è 0.20 Euro.
- q_{30}: il saldo corrente è 0.30 Euro.
- q_{40}: il saldo corrente è 0.40 Euro.
- q_{50}: il saldo corrente è 0.50 Euro.
- q_{60}: il saldo corrente è 0.60 Euro.
- q_{70}: il saldo corrente è 0.70 Euro.
- q_{80}: il saldo corrente è 0.80 Euro e il distributore deve emettere un francobollo senza dare resto.
- q_{90}: il saldo corrente è 0.90 Euro e il distributore deve dare 0.10 Euro di resto prima di emettere un francobollo.
- q_{100}: il saldo corrente è almeno 1 Euro e il distributore deve dare 0.20 Euro di resto prima di emettere un francobollo.

Esercizio 4.33 *(27/02/2024)*

Fissato un numero k qualsiasi, dire se il seguente linguaggio è regolare:

$$L_1 = \{a^n b^m \mid n \leq k, m \leq k\}$$

E questo?

$$L_2 = \{a^k b^n c^m \mid n \geq 0, k \leq n, n \leq m\}$$

Soluzione 4.33

- Scegliamo $k = 2$. Dimostriamo la regolarità di $L_1 = \{a^n b^m \mid n \leq 2, m \leq 2\}$ mostrando una possibile grammatica. Dato che il linguaggio è finito, è sufficiente fornire una grammatica che generi le stringhe $\{\lambda, a, b, aa, bb, ab, aab, abb, aabb\}$. Proponiamo la grammatica $G = (N = \{S, A_1, A_2, B\}, T = \{a, b\}, \mathcal{P}, S)$ con produzioni

$$\begin{aligned} S &\longrightarrow aA_1 \mid bB \mid a \mid b \mid \lambda \\ A_1 &\longrightarrow aA_2 \mid bB \mid a \mid b \\ A_2 &\longrightarrow bB \mid b \\ B &\longrightarrow b \end{aligned}$$

- In questo caso k non è fissato. Iniziamo col dimostrare che L_2 non è regolare usando il pumping lemma per i linguaggi regolari.

  Assumiamo L_2 regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Prendiamo dunque la stringa $z = a^p b^p c^p$. Si ha che $z \in L_2$ con $k = n = m > 0$ e $l(z) = 3p \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w.

  Dato che $l(uv) \leq p$, v può consistere solamente di a, e dato che $l(v) > 0$, v non può essere una stringa vuota. Quindi pompando nuovi simboli a in

v, per esempio con uv^2w, il numero di a diventa strettamente maggiore del numero di b, mentre abbiamo il vincolo che le a siano al più al pari delle b. Quindi la stringa uv^2w non appartiene a L_2, e di conseguenza L non è un linguaggio regolare.

Tuttavia, possiamo dimostrare adesso che L_2 non è neppure libero dal contesto, questa volta applicando il pumping lemma per i linguaggi liberi.

Assumiamo L_2 libero dal contesto e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Possiamo prendere nuovamente la stringa $z = a^p b^p c^p$, che appartiene a L_2 e come sappiamo ha lunghezza maggiore di p. Quindi $z = uvwxy$ per qualche u, v, w, x, y.

Dato che $l(vwx) \leq p$, vwx non può contenere sia a che c, e dato che $l(vx) > 0$, vx non può essere la stringa vuota. Necessariamente quindi deve essere che $u = a^p$ o $y = c^p$. Consideriamo entrambi i casi:

Caso $u = a^p$ In questo caso vwx può consistere solamente di c, solamente di b, o includere sia che b che c. In questi ultimi due casi, rimuovendo dei simboli con uv^0wx^0y, andiamo a fare in modo che il numero di b diventi inferiore rispetto a quello delle a, e quindi otteniamo una stringa che non appartiene al linguaggio. Nell'altro caso, se vwx consiste solamente di c, analogamente rimuovendo dei simboli con uv^0wx^0y otteniamo una stringa che non appartiene al linguaggio perché il numero delle c sarà inferiore al numero delle b.

Caso $y = c^p$ In questo caso vwx può consistere solamente di a, solamente di b, o includere sia che a che b. In questi ultimi due casi, pompando nuovi simboli con uv^2wx^2y, andiamo a fare in modo che il numero di b diventi superiore rispetto a quello delle c, e quindi otteniamo una stringa che non appartiene al linguaggio. Nell'altro caso, se vwx consiste solamente di a, analogamente pompando nuovi simboli con uv^2wx^2y otteniamo una stringa che non appartiene al linguaggio perché il numero delle a sarà superiore al numero delle b.

Ora possiamo mostrare che L_2 è un linguaggio dipendente dal contesto fornendo la sua grammatica. Proponiamo la grammatica monotona $G = (N = \{S, A, B, C\}, T = \{a, b, c\}, \mathcal{P}, S)$ con produzioni

$$
\begin{aligned}
S &\longrightarrow aABC \mid aBC \mid bBCC \mid bC \mid cC \mid c \mid \lambda \\
A &\longrightarrow aABC \mid aBC \\
aB &\longrightarrow abBC \mid ab \\
CB &\longrightarrow BC \\
bB &\longrightarrow bBC \mid bb \\
bC &\longrightarrow bcC \mid bc \\
cC &\longrightarrow cC \mid cc
\end{aligned}
$$

Esercizio 4.34 (*24/06/2024*)

Dire a quale classe appartiene il linguaggio L, sull'alfabeto $A = \{a, b\}$, definito come:

$$L = \{w \in A^\star \mid w \text{ ha lo stesso numero di } a \text{ e } b\}$$

Soluzione 4.34

Iniziamo mostrando che L non è un linguaggio regolare applicando il Pumping lemma.
Assumiamo L regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Prendiamo dunque la stringa $z = a^p b^p$. Si ha che $z \in L$ e $l(z) = 2p \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w.
Dato che $l(uv) \leq p$, v può consistere solamente di a, e dato che $l(v) > 0$, v non può essere una stringa vuota. Quindi pompando nuovi simboli a in v, per esempio con uv^2w, il numero di a diventa strettamente maggiore del numero di b, mentre abbiamo il vincolo che il numero delle a sia pari al numero delle b. Quindi la stringa uv^2w non appartiene a L, e di conseguenza L non è un linguaggio regolare.
Mostriamo ora che L è un linguaggio di tipo 2 definendo una grammatica libera dal contesto che lo generi, per esempio $G = (N = \{S\}, T = \{a, b\}, \mathcal{P}, S)$ con produzioni

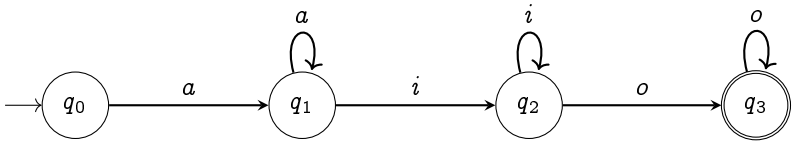
$$S \longrightarrow aSb \mid bSa \mid SS \mid \lambda$$

Esercizio 4.35 (*08/07/2024*)

Si consideri l'alfabeto delle lettere minuscole del linguaggio italiano. Dire a qualche classe di linguaggi appartiene l'insieme delle stringhe composte dalle vocali del proprio nome, nel rispetto dell'ordine di apparizione nel nome stesso.

Soluzione 4.35

Consideriamo l'esercizio per il nome "Flavio". Le vocali quindi sono $\{a, i, o\}$ nell'ordine $a < i < o$. Dimostriamo la regolarità del linguaggio mostrando un automa a stati finiti deterministico che lo riconosca.



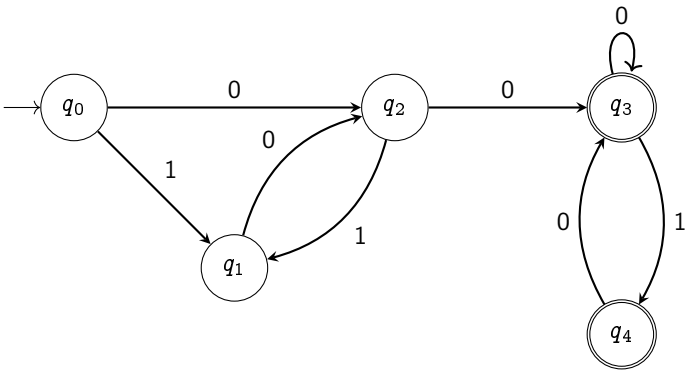
Esercizio 4.36 (*24/07/2024*)

Dire a quale classe di linguaggi sull'alfabeto $\{0, 1\}$ appartengono gli insiemi di stringhe tali che le occorrenze di 0 e di 1 soddisfano i seguenti requisiti:

- ci sono almeno due 0 consecutivi e non vi sono mai due 1 consecutivi;
- il numero di occorrenze di 0 è maggiore o uguale al numero di occorrenze di 1.

Soluzione 4.36

Il linguaggio deve rispettare entrambi i requisiti elencati. Mostriamo che il linguaggio è regolare definendo un automa a stati finiti deterministico che lo riconosca.

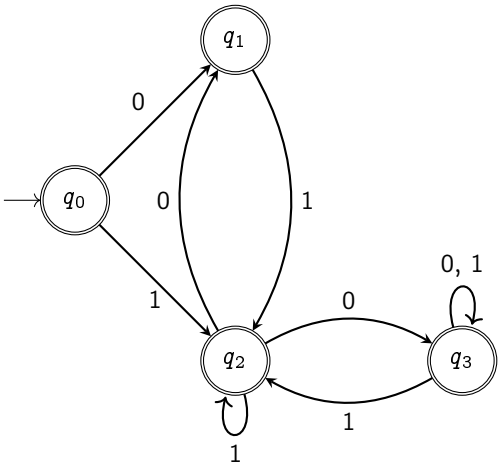


Esercizio 4.37 (*17/09/2024*)

Dire a quale classe di linguaggi sull'alfabeto $\{0, 1\}$ appartengono gli insiemi di stringhe tali che la sottostringa "00" non occorra mai né come prefisso né come suffisso delle stringhe del linguaggio.

Soluzione 4.37

Mostriamo che il linguaggio è regolare definendo un automa a stati finiti non deterministico che lo riconosca.



Esercizio 4.38 (*27/09/2024*)

Si consideri la seguente grammatica $G = (N = \{S, A\}, T = \{a, b\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow A \mid AA \mid \lambda$$
$$A \longrightarrow aAb \mid ab$$

Svolgere ciascuno dei seguenti punti:

1. Di che tipo è la grammatica?
2. Mostrare una derivazione per “abaabb”.
3. Mostrare una derivazione per “aabbab”.
4. Sono ammesse stringhe vuote?
5. Qual è il linguaggio generato da G?

Soluzione 4.38

1. La grammatica è libera dal contesto (tipo 2).
2. $S \to AA \to abA \to abaAb \to abaabb$.
3. $S \to AA \to Aab \to aAbab \to aabbab$.
4. Sì.
5. $L = \{a^n b^n a^m b^m \mid n, m \geq 0\}$

Esercizio 4.39 *(20/11/2024)*

Dire a quale classe nella Gerarchia di Chomsky appartiene il linguaggio L sull'alfabeto $\{0, 1\}$ definito da

$$L = \{(01)^n (10)^n \mid n \in \mathbb{N}\}$$

Soluzione 4.39

Iniziamo mostrando che L non è un linguaggio regolare applicando il Pumping lemma.
Assumiamo L regolare e fissiamo un $p \in \mathbb{N}$ come richiesto dal pumping lemma. Prendiamo dunque la stringa $z = (01)^p (10)^p$. Si ha che $z \in L$ e $l(z) = 4p \geq p$, quindi si può scrivere $z = uvw$ per qualche u, v, w.
Data la struttura della stringa v può consistere di qualsiasi sequenza di simboli, ma dato che $l(v) > 0$, v non può essere una stringa vuota. Tuttavia dato che $l(uv) \leq p$ e il linguaggio considerato genera solo stringhe palindrome, qualsiasi modifica a v, per esempio con uv^2w, risulterebbe in una nuova stringa non-palindroma. Quindi la stringa uv^2w non appartiene a L, e di conseguenza L non è un linguaggio regolare.
Mostriamo ora che L è un linguaggio di tipo 2 definendo una grammatica libera dal contesto che lo generi, per esempio $G = (N = \{S\}, T = \{0, 1\}, \mathcal{P}, S)$ con produzioni

$$S \longrightarrow 01S10 \mid \lambda$$

Esercizio 4.40 (*05/02/2025, 26/02/2025*)

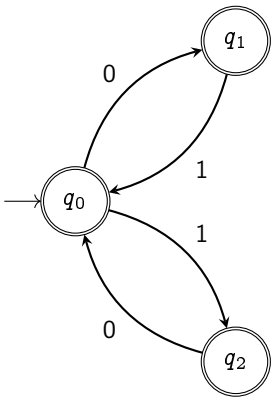
Dire a quale classe di linguaggi nella Gerarchia di Chomsky appartiene il linguaggio:

$$L = \{w \in \{0,1\}^* \mid \forall n \in \mathbb{N},\ 2 \leq 2n \leq |w|,\ \text{vale } w[2n] \neq w[2n-1]\}$$

dove $w[i]$ rappresenta il bit della stringa w alla posizione i-esima.

Soluzione 4.40

In questo linguaggio le stringhe sono tali che il secondo simbolo è diverso dal primo, il quarto dal terzo, e così via. Inoltre è inclusa la stringa vuota. Mostriamo che il linguaggio è regolare definendo un automa a stati finiti deterministico che lo riconosca.



Esercizio 4.41 (*Slides Prof. Re*)

1. Scrivere una espressione regolare che descriva l'insieme $\{0, 11, 101\}$.
2. Scrivere una espressione regolare per stringhe binarie che descriva l'insieme delle stringhe composte solo da simboli 0.
3. Scrivere una espressione regolare per tutte stringhe binarie.
4. Scrivere una espressione regolare per tutte stringhe binarie che cominciano e finiscono per 1.
5. Scrivere una espressione regolare per le stringhe binarie che contengono almeno tre 1 consecutivi.
6. Scrivere una espressione regolare per le stringhe binarie che contengono almeno tre 1.
7. Scrivere una espressione regolare per le stringhe binarie di lunghezza dispari.
8. Scrivere una espressione regolare per stringhe di testo che descriva le date in formato GG/MM/AAAA.

9. Scrivere una espressione regolare per stringhe di testo che descriva i numeri di telefono.

Soluzione 4.41

1. $0 + 11 + 101$
2. $0 \cdot 0^\star$
3. $(0 + 1)^\star \cdot (0 + 1)$
4. $1 \cdot (0 + 1)^\star \cdot 1$
5. $(0 + 1)^\star \cdot 111 \cdot (0 + 1)^\star$
6. $(0 + 1)^\star \cdot 1 \cdot (0 + 1)^\star \cdot 1 \cdot (0 + 1)^\star \cdot 1 \cdot (0 + 1)^\star$
7. $(0 + 1) \cdot (00 + 01 + 10 + 11)^\star$ oppure $(0 + 1) \cdot ((0 + 1) \cdot (0 + 1))^\star$
8. $(0+1+2+3) \cdot (0+1+2+\cdots+9) \cdot / \cdot (0+1) \cdot (0+1+2+\cdots+9) \cdot / \cdot (0+1+2+\cdots+9) \cdot (0+1+2+\cdots+9) \cdot (0+1+2+\cdots+9) \cdot (0+1+2+\cdots+9)$
9. $(0 + 1 + 2 + \cdots + 9)^\star$

Gerarchia di Chomsky

Grammatica a struttura di frase (tipo 0) Una grammatica a struttura di frase (o grammatica di tipo 0) è una quadrupla $G = (N, T, \mathcal{P}, S)$ dove:

- T e N sono alfabeti disgiunti, composti rispettivamente da quelli che sono chiamati simboli terminali e simboli non terminali di G;
- Le produzioni in P sono nella forma

$$\gamma A \delta \to \beta \text{ con } A \in N,\ \gamma, \delta, \beta \in (N \cup T)^\star$$

- S è un simbolo di N, e viene chiamato simbolo iniziale di G.

Grammatica dipendente dal contesto (tipo 1) Una grammatica dipendente dal contesto (o grammatica di tipo 1) è una quadrupla $G = (N, T, \mathcal{P}, S)$ dove:

- T e N sono alfabeti disgiunti, composti rispettivamente da quelli che sono chiamati simboli terminali e simboli non terminali di G;
- Le produzioni in P sono nella forma

$$\gamma A \delta \to \gamma \beta \delta \text{ con } A \in N,\ \gamma, \delta \in (N \cup T)^\star,\ \beta \in (N \cup T)^+$$

- S è un simbolo di N, e viene chiamato simbolo iniziale di G.

Grammatica libera dal contesto (tipo 2) Una grammatica libera dal contesto (o grammatica di tipo 2) è una quadrupla $G = (N, T, \mathcal{P}, S)$ dove:

- T e N sono alfabeti disgiunti, composti rispettivamente da quelli che sono chiamati simboli terminali e simboli non terminali di G;
- Le produzioni in P sono nella forma

$$A \to \beta \text{ con } A \in N,\ \beta \in (N \cup T)^+$$

- S è un simbolo di N, e viene chiamato simbolo iniziale di G.

Grammatica regolare (tipo 3) Una grammatica regolare (o grammatica di tipo 3) è una quadrupla $G = (N, T, \mathcal{P}, S)$ dove:

- T e N sono alfabeti disgiunti, composti rispettivamente da quelli che sono chiamati simboli terminali e simboli non terminali di G;
- Le produzioni in P sono nella forma

$$A \to aB$$

$$A \to a$$

  con $A, B \in N,\ a \in T$

- S è un simbolo di N, e viene chiamato simbolo iniziale di G.

Pumping lemma per i linguaggi regolari

Sia L un linguaggio regolare. Allora esiste un intero $p \geq 1$ che dipende solamente da L, tale che ogni stringa $z \in L$ di lunghezza almeno p può essere scritta come $z = uvw$ rispettando le seguenti condizioni:

- $l(v) \geq 1$;
- $l(uv) \leq p$;
- $uv^i w \in L,\ \forall i \geq 0$.

Pumping lemma per i linguaggi liberi dal contesto

Sia L un linguaggio libero dal contesto su un alfabeto T. Esiste una costante $p \in \mathbb{N}$ dipendente solo da L tale che, per ogni $z \in L$, se $l(z) \geq p$, allora esistono stringhe $u, v, w, x, y \in T^\star$ tali per cui $z = uvwxy$ e valgono le seguenti proprietà:

(a) $l(vx) \geq 1$;

(b) $l(vwx) \leq p$

(c) $\forall i \in \mathbb{N}, uv^i wx^i y \in L$

Capitolo 5

Il linguaggio WHILE

Esercizio 5.1 (*20/06/2019*)

Fornire la macro-istruzione nel linguaggio While per la divisione con resto di due numeri naturali.

Soluzione 5.1

Consideriamo la variabile A come dividendo e B come divisore. Quindi calcoliamo il quoziente Q e il resto della divisione R. In caso di B $= 0$, l'algoritmo non termina perché l'operazione non è definita.

```
begin
INPUT(A);
INPUT(B);
Q := 0;
while A ≥ B do
    begin
    A := A - B;
    Q := Q + 1
    end
R := A;
OUTPUT(Q);
OUTPUT(R)
end
```

Esercizio 5.2 (*23/07/2019*)

Fornire la macro-istruzione nel linguaggio While per la moltiplicazione tra due naturali. E tra due interi?

Soluzione 5.2

Consideriamo la moltiplicazione come l'operazione $Z = X \cdot Y$

```
begin
INPUT(X);
INPUT(Y);
Z := 0;
while Y > 0 do
    begin
        Z := Z + X;
        Y := Y - 1
    end
OUTPUT(Z)
end
```

Per rendere la seconda parte dell'esercizio interessante, consideriamo i numeri interi come una coppia di valori (segno, modulo). Introduciamo il segno come due variabili booleane SIGNX e SIGNY, dove il valore 1 indica un segno negativo e 0 un segno positivo.

```
begin
INPUT(SIGNX);
INPUT(X);
INPUT(SIGNY);
INPUT(Y);
Z := 0;
while Y > 0 do
    begin
        Z := Z + X;
        Y := Y - 1
    end
if SIGNX ≠ SIGNY then
    begin
    OUTPUT(1)
    end
else
    begin
    OUTPUT(0)
    end
OUTPUT(Z)
end
```

Esercizio 5.3 (*13/09/2019, 27/09/2019, 15/11/2021, 07/02/2023*)

Come è possibile estendere il linguaggio While con un nuovo comando `repeat` C `until` E dove C è un comando e E è una espressione booleana. Informalmente, il comando `repeat` C `until` E esegue C e poi verifica E, se E falsa ripete C, altrimenti esce dal ciclo.

Soluzione 5.3

Il comando si può esprimere nel linguaggio WHILE come:

```
begin
C;
while E ≠ True do
    begin
    C
    end
end
```

Esercizio 5.4 (*16/12/2019*)

Definire la sintassi del linguaggio WHILE.

Soluzione 5.4

PROGRAM	::= `begin end` | `begin` SEQCOMMANDS `end`
SEQCOMMANDS	::= COMMAND | SEQCOMMANDS`;`COMMAND
COMMAND	::= ASSIGNMENT | `while` TEST `do` COMMAND | PROGRAM
ASSIGNMENT	::= VAR `:= 0` | VAR `:= s(`VAR`)` | VAR `:= pd(`VAR`)`
TEST	::= VAR ≠ VAR
VAR	::= LETTERA | VAR LETTERA | VAR CIFRA
LETTERA	::= A | B | ... | Z
CIFRA	::= 0 | 1 | ... | 9

Esercizio 5.5 (*07/02/2020*)

Scrivere un programma While che presi due numeri in input controlla se il primo è multiplo del secondo.

Soluzione 5.5

Per rendere l'esercizio meno banale, evitiamo di usare le macro per la divisione e il resto tra numeri naturali. Presi X e Y ritorniamo 1 se X è un multiplo di Y, 0 altrimenti.

```
begin
INPUT(X);
INPUT(Y);
if Y = 0 then
    /* solo 0 è  multiplo di 0 */
    if X = 0 then
        begin OUTPUT(1) end
    else
        begin OUTPUT(0) end
else
    while X ≥ Y do
        begin X := X - Y end
    if X ≠ 0 then
        begin OUTPUT(0) end
    else
        begin OUTPUT(1) end
end
```

Esercizio 5.6 (*05/03/2020, 18/06/2020*)

È possibile estendere il linguaggio WHILE con un nuovo comando

$$\texttt{read}(x_1, \ldots, x_n)$$

che prende in input n lettere o cifre e le assegna alle variabili $x_1, \ldots, x_n$, rispettivamente?

Soluzione 5.6

Sì, possiamo estendere il linguaggio WHILE con il comando richiesto, implementando la macro come segue:

```
begin
INPUT(X1);
INPUT(X2);
⋮
INPUT(XN)
end
```

Dove il numero di variabili `Xi` dipende dall' n fissato del numero degli input. Se ci sono n input ci sono n righe di comandi INPUT che assegnano l'i-esimo valore alla variable `Xi`.

Esercizio 5.7 (*01/07/2020, 20/07/2020, 07/09/2020, 28/02/2023, 19/09/2023, 28/09/2023, 08/07/2024*)

Si introduca in WHILE il comando

`IntWhile` $e \in I_1; I_2; \ldots; I_n$ `IntDo ibegin` C_1 `iend;` $\ldots$ `; ibegin` C_n `iend;`

dove e è una espressione la cui valutazione restituisce un intero, I_i è un intervallo sugli interi, C_i è un comando ($i \in [1..n]$).

Il comando `IntWhile` ... `IntDo`, esegue il comando C_i finché la valutazione dell'espressione e restituisce un intero nell'intervallo I_i. Nel caso in cui la valutazione di e non ricade in alcuno degli intervalli $I_1, \ldots, I_n$ il comando `IntWhile` ... `IntDo` termina la sua esecuzione.

Soluzione 5.7

Utilizziamo una macro $\in$ per stabilire se un dato intero appartiene a un intervallo.

```
begin
while e ∈ I_1 ∨ ... ∨ e ∈ I_n do
    begin
    while e ∈ I_1 do begin C_1 end
    while e ∈ I_2 do begin C_2 end
    ⋮
    while e ∈ I_n do begin C_n end
    end
end
```

In alternativa ai comandi while all'interno del while principale, si possono anche usare comandi if-then-else con le stesse condizioni, col vantaggio che è possibile eseguire multipli comandi C_i quando e appartiene a più intervalli.

Esercizio 5.8 *(28/09/2020, 06/09/2021)*

Scrivere un programma in linguaggio WHILE che legga in input una sequenza di lunghezza ignota a priori di numeri interi positivi. Il programma, a partire dal primo numero introdotto, stampa ogni volta la media aritmetica di tutti i numeri introdotti. Terminare quando il numero inserito è negativo.

Soluzione 5.8

```
begin
INPUT(X);
MEDIA := 0;
CONTATORE := 0;
SOMMA := 0;
while X ≥ 0 do
    begin
    CONTATORE := CONTATORE + 1;
    SOMMA := SOMMA + X;
    MEDIA := SOMMA / CONTATORE;
    OUTPUT(MEDIA);
    INPUT(X)
    end
OUTPUT(MEDIA)
end
```

Esercizio 5.9 (*09/02/2021*)

Scrivere un programma in linguaggio WHILE che prenda in input una sequenza di caratteri nell'insieme $\{a, b, c\}$ e restituisca 1 se la stringa appartiene al linguaggio dell'Esercizio 4.13 a pagina 67 nella sezione Grammatiche e Automi, 0 altrimenti.

Soluzione 5.9

Restituiamo 1 qualora la stringa di input appartiene al linguaggio, 0 altrimenti. Per utilizzare i simboli λ, a, b e c nel linguaggio WHILE, basta considerarli come codificati in numeri naturali, per esempio $\lambda = 0$, $a = 1$, $b = 2$ e $c = 3$. Quindi laddove nel codice seguente appaiono i simboli λ, a, b o c, in realtà si stanno veramente usando i numeri 0, 1, 2 o 3.

```
begin
INPUT(CHAR);
N := 0;
M := 0;
L := 0;
if CHAR ≠ λ then
    begin
    while CHAR = a do
        begin N := N + 1; INPUT(CHAR) end
    while CHAR = b do
        begin M := M + 1; INPUT(CHAR) end
    if CHAR ≠ c then
        begin OUTPUT(0) end
    else
        begin
        while CHAR = c do
            begin L := L + 1; INPUT(CHAR) end
        if L > 1 then
            begin OUTPUT(0) end
        else
            begin
            N2 := 0;
            M2 := 0;
            while CHAR = b do
                begin M2 := M2 + 1; INPUT(CHAR) end
            while CHAR = a do
                begin N2 := N2 + 1; INPUT(CHAR) end
            if N2 ≠ N ∨ M2 ≠ M ∨ CHAR ≠ λ then
                begin OUTPUT(0) end
            else
                begin OUTPUT(1) end
            end
        end
    end
else
    begin OUTPUT(1) end
end
```

Esercizio 5.10 (*26/02/2021*)

Scrivere un programma in linguaggio WHILE che prenda in input una sequenza di caratteri nell'insieme $\{a, b, c\}$ e restituisca 1 se la stringa appartiene al linguaggio dell'Esercizio 4.14 a pagina 67 nella sezione Grammatiche e Automi, 0 altrimenti.

Soluzione 5.10

Restituiamo 1 qualora la stringa di input appartiene al linguaggio, 0 altrimenti. Per utilizzare i simboli λ, a, b e c nel linguaggio while, basta considerarli come codificati in numeri naturali, per esempio $\lambda = 0$, $a = 1$, $b = 2$ e $c = 3$. Quindi laddove nel codice seguente appaiono i simboli λ, a, b o c, in realtà si stanno veramente usando i numeri 0, 1, 2 o 3.

```
begin
INPUT(CHAR);
COUNT := 0;
while CHAR ≠ λ do
    begin
    if CHAR = c then
        begin COUNT := COUNT + 1 end
    INPUT(CHAR)
    end
if COUNT ≠ 0 then
    begin OUTPUT(0) end
else
    begin OUTPUT(1) end
end
```

Esercizio 5.11 (*22/06/2021*)

Si introduca in WHILE il comando

$$\texttt{loop } E \texttt{ do } C \texttt{ end}$$

dove E è una espressione aritmetica, la cui valutazione restituisce un intero I. La semantica intuitiva del comando `loop` è di controllare la guardia E e, se è diversa da 0, eseguire C e decrementare la guardia. Dire se il comando `loop` è esprimibile nel linguaggio WHILE.

Soluzione 5.11

Sì, il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
I := E;
while I ≠ 0 do
    begin
    C;
    I := I - 1
    end
end
```

Con questa soluzione si presuppone che se il comando C modifica la valutazione di E, il numero di iterazioni del ciclo rimane comunque invariato.

Esercizio 5.12 *(01/07/2021)*

Si consideri il comando:

$$\texttt{for } I = E_1 \texttt{ to } E_2 \texttt{ do } C \texttt{ end}$$

dove E_1, E_2 sono espressioni aritmetiche. La semantica intuitiva del `for` è di eseguire il comando C tante volte quante $E2 - E1$. Dire se il comando `for` è esprimibile nel linguaggio WHILE.

Soluzione 5.12

Sì, il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
I := E1;
E := E2;
while I < E do
    begin
    C;
    I := I - 1
    end
end
```

Notare bene che $E_2 - E_1$ potrebbe equivalere a un valore negativo. In tal caso il comando C dovrebbe essere eseguito 0 volte. Utilizzando l'operatore di confronto $<$ si evita il problema di eseguire la sottrazione, altrimenti si può usare la sottrazione troncata.

Esercizio 5.13 *(15/07/2021, 06/02/2024)*

Si consideri il comando:

$$\texttt{loopmax } E_1, E_2 \texttt{ do } C \texttt{ end}$$

la cui semantica intuitiva consiste nell'eseguire il comando C tante volte quanto vale il massimo tra E_1 ed E_2 fissato all'inizio. Dire se il comando `loopmax` è esprimibile nel linguaggio WHILE.

Soluzione 5.13

Sì, il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
if E_1 > E_2 then
    begin I := E_1 end
else
    begin I := E_2 end
while I > 0 do
    begin
    C;
    I := I - 1
    end
end
```

Esercizio 5.14 (*27/09/2021*)

Dire se il comando

$$\texttt{do } C \texttt{ while } E$$

dove C è un comando e E è una espressione booleana è esprimibile nel linguaggio WHILE. Informalmente, il comando `do` C `while` E esegue C e poi verifica E, se E falsa ripete C, altrimenti esce dal ciclo.

Soluzione 5.14

Sì, il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
C;
while E ≠ True do
    begin
    C
    end
end
```

Esercizio 5.15 (*07/02/2022, 21/02/2022*)

Dire se il comando

$$\texttt{alternate } E_1 \texttt{ by } E_2 \texttt{ do } C_1, C_2 \texttt{ endalt}$$

la cui semantica intuitiva consiste nell'eseguire C_1 un numero di volte pari a E_1/E_2 e C_2 un numero di volte pari a $E_1 \mod E_2$ è esprimibile nel linguaggio WHILE.

Soluzione 5.15

Sì, il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
Q := E_1/E_2;
R := E_1 mod E_2;
while Q ≠ 0 ∨ R ≠ 0 do
    begin
    if Q ≥ 0 then
        begin
        C_1;
        Q := Q - 1
        end
    if R ≥ 0 then
        begin
        C_2;
        R := R - 1
        end
    end
end
```

In questo caso, C_1 ed C_2 si alternano nella loro esecuzione, finché possibile.

Esercizio 5.16 (*07/03/2022*)

Dire se il comando

$$\texttt{alternate } E \texttt{ times } C_1, C_2 \texttt{ endalt}$$

con le seguenti due semantiche intuitive:

1. viene eseguito il comando $C_1; C_2$ tante volte quanto vale inizialmente E;
2. viene eseguito il comando C_1 tante volte quanto vale E_1/E_2 e successivamente il comando C_2 tante volte quanto vale $E_1 \mod E_2$

è esprimibile nel linguaggio WHILE.

Soluzione 5.16

Sì, il comando è esprimibile nel linguaggio WHILE.

1. La prima semantica si può implementare come:

```
begin
I := E;
while I ≠ 0 do
    begin
    C_1;
    C_2;
    I := I - 1
    end
end
```

2. La seconda semantica si può implementare come:

```
begin
Q := E1/E2;
R := E1 mod E2;
while Q > 0 do
    begin
    C1;
    Q := Q - 1
    end
while R > 0 do
    begin
    C2;
    R := R - 1
    end
end
```

Esercizio 5.17 (*17/06/2022, 20/11/2024*)

Dire se il comando:

$$\texttt{do } C \texttt{ until } X \texttt{ is } Y \texttt{ end}$$

dove C è un comando, X e Y sono variabili sui numeri naturali è esprimibile nel linguaggio WHILE. La semantica intuitiva è di eseguire C, poi se X ed Y hanno valori diversi ripetere C, altrimenti uscire dal ciclo.

Soluzione 5.17

Sì, il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
C;
while X ≠ Y do
    begin C end
end
```

Esercizio 5.18 (*27/06/2022, 27/02/2024*)

Dire se il comando:

$$\texttt{case } E \texttt{ do } C_1, C_2, C_3 \texttt{ end}$$

dove E è una espressone aritmetica sui numeri naturali e C_1, C_2 e C_3 sono comandi è esprimibile nel linguaggio WHILE. La semantica intuitiva del comando case è quella di valutare E e se il valore di E è 1 si esegua C_1, se il valore di E è 2 si esegua C_2, se il valore di E è 3 si esegua C_3. Per tutti gi altri valori si esca dal case.

Soluzione 5.18

Per definire questa macro ci ispiriamo al fatto che nei linguaggi di programmazione più diffusi espressioni di questo tipo, ovvero gli *switch*, sono facilmente esprimibili come una serie di if-then-else annidati. Il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
if E = 1 then
    begin C_1 end
else
    begin
    if E = 2 then
        begin C_2 end
    else
        begin
        if E = 3 then
            begin C_3 end
        end
    end
end
```

Esercizio 5.19 (*15/07/2022*)

Dire se il comando:

`mul` X `less` Y `end`

dove X e Y sono variabili sui numeri naturali è esprimibile nel linguaggio WHILE. La semantica intuitiva del comando è di dare all'esterno i multipli del valore della variabile X che sono inferiori al valore della variabile Y.

Soluzione 5.19

Sì, il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
M := 0;
while M < Y do
    begin
    OUTPUT(M);
    M := M + X
    end
end
```

Nel programma iniziamo restituendo 0 in output perché 0 è un multiplo di ogni altro numero. Usiamo la macro `OUTPUT` per restituire i valori richiesti ad ogni passo del ciclo.

Esercizio 5.20 (*07/09/2022*)

Dire se è possibile estendere il linguaggio WHILE con un nuovo comando

$$\texttt{WHILESUMO}\ C$$

dove C è un comando. Informalmente, il comando `WHILESUMO` C esegue ciclicamente il comando C. Ad ogni ciclo legge un numero intero dall'esterno e lo somma a quelli precedentemente letti. Il ciclo termina quando la somma è 0.

Soluzione 5.20

Sì, possiamo estendere il linguaggio WHILE con il comando richiesto, implementando la macro come segue:

```
begin
SOMMA := 0;
C;
INPUT(X);
SOMMA := SOMMA + X;
while SOMMA ≠ 0 do
    begin
    C;
    INPUT(X);
    SOMMA := SOMMA + X
    end
end
```

Esercizio 5.21 (*19/09/2022*)

Dire se è possibile estendere il linguaggio WHILE con un nuovo comando

$$\texttt{WParDis}\ E, C_1, C_2$$

dove E è una espressione aritmetica e C_1, C_2 sono comandi. Informalmente, il comando $WParDis\ E, C_1, C_2$ — ad ogni ciclo — valuta l'espressione aritmetica E ed esegue C_1 se la valutazione è pari, C_2 se la valutazione è dispari.

Soluzione 5.21

Sì, possiamo estendere il linguaggio WHILE con il comando richiesto, implementando la macro come segue:

```
begin
while True do
    begin
    P := E mod 2;
    if P = 0 then
        begin C1 end
    else
        begin C2 end
    end
end
```

Notare bene che dalla specifica della semantica data dall'esercizio, il comando deve continuare la sua esecuzione senza mai arrestarsi, alternando l'esecuzione di C_1 e C_2 a seconda della valutazione di E.

Esercizio 5.22 (*03/10/2022*)

Dire se è possibile estendere il linguaggio WHILE con un nuovo comando

$$\texttt{ParDis } E, C_1, C_2$$

dove E è una espressione aritmetica e C_1, C_2 sono comandi. Informalmente, il comando `ParDis` E, C_1, C_2 valuta l'espressione aritmetica E ed esegue ciclicamente C_1 infinite volte se la valutazione è pari, esegue ciclicamente C_2 infinite volte se la valutazione è dispari.

Soluzione 5.22

Sì, possiamo estendere il linguaggio WHILE con il comando richiesto, implementando la macro come segue:

```
begin
if E  mod  2 = 0 then
    begin
    while True do
        begin C1 end
    end
else
    begin
    while True do
        begin C2 end
    end
end
```

Notare bene che dalla specifica della semantica data dall'esercizio, il comando deve continuare la sua esecuzione senza mai arrestarsi, eseguendo infinite volte C_1 se la valutazione di E è un numero pari, oppure infinite volte C_2 in caso contrario.

Esercizio 5.23 (*09/06/2023, 27/09/2024*)

Si dica se e come, il comando:

$$\texttt{for } I := 1 \texttt{ to } N \texttt{ do } C$$

che esegue il comando C per $N \in \mathbb{N}$ volte, può essere codificato nel linguaggio WHILE.

Soluzione 5.23

Sì, possiamo estendere il linguaggio WHILE con il comando richiesto, implementando la macro come segue:

```
begin
I := 1;
while I ≤ N do
    begin
    C;
    I := I + 1
    end
end
```

Notare che la variabile I deve essere accessibile dal comando C (tipicamente, questo accesso avviene in sola lettura). Inoltre N è una costante e non una variabile, e quindi va trattata come tale.

Esercizio 5.24 (*09/06/2023*)

Scrivere (se possibile) un programma nel linguaggio WHILE che dati in input due naturali che rappresentano rispettivamento il tasso di propagazione di un virus (quante nuove persone ogni giorno si ammalano per ogni persona già ammalata) e la quantità di persone di una popolazione, dica quanti giorni sono necessari perché sia ammalata almeno la metà della popolazione considerando che allinizio ci sia una sola persona ammalata.

Soluzione 5.24

Sì, possiamo scrivere questo programma nel linguaggio WHILE:

```
begin
INPUT(T);
INPUT(P);
M := 1;
G := 1;
P := P / 2;
while M < P do
    begin
    M := M + M · T;
    G := G + 1
    end;
OUTPUT(G)
end
```

Esercizio 5.25 (*12/07/2023*)

Dire se è possibile estendere il linguaggio WHILE con un comando

$$\texttt{fix } X := 0 \texttt{ in } C$$

dove X è una variabile e C un comando. Informalmente, il comando `fix` inizializza una variabile a 0 e quindi esegue C, se il nuovo valore di X (ottenuto eseguendo C) è uguale al valore precedente di X il ciclo termina, altrimenti si ripete questa operazione fintanto che il valore X non smette di variare.

Soluzione 5.25

Sì, possiamo estendere il linguaggio WHILE con il comando richiesto, implementando la macro come segue:

```
begin
X := 0;
Y := X;
C;
while X ≠ Y do
    begin
    Y := X;
    C
    end
end
```

Esercizio 5.26 (*08/09/2023*)

Si dica se e come, il comando

$$\texttt{ite } I := 1 \texttt{ to } N \texttt{ do } C$$

che esegue il comando C per $N \in \mathbb{N}$ volte, può essere codificato nel linguaggio WHILE.

Soluzione 5.26

Sì, può codificare il comando richiesto nel linguaggio WHILE, implementando la macro come segue:

```
begin
I := 1;
while I ≤ N do
    begin
    C;
    I := I + 1
    end
end
```

Esercizio 5.27 *(24/06/2024)*

Dire se l'istruzione

$$\texttt{for } (STATEMENT_1,\ I,\ STATEMENT_2)\ C \texttt{ end}$$

dove

- I è un contatore sull'insieme dei numeri natural;
- $STATEMENT_1$ è una espressione artimetica che determina il valore iniziale per il contatore I;
- $STATEMENT_1$ è una espressione artimetica che determina il valore finale per il contatore I;
- C è un comando WHILE.

è esprimibile nel linguaggio WHILE.

Soluzione 5.27

Sì, può codificare l'istruzione richiesta nel linguaggio WHILE, implementando la macro come segue:

```
begin
I := STATEMENT_1;
E := STATEMENT_2;
while I < E do
    begin
    C;
    I := I + 1
    end
end
```

Esercizio 5.28 (*24/07/2024*)

Si introduca in WHILE il comando:

$$\texttt{max } E_1, E_2 \texttt{ loop } C \texttt{ end}$$

la cui semantica intuitiva consiste nell'eseguire il comando C tante volte quanto vale il massimo tra E_1 ed E_2 fissato all'inizio.

Soluzione 5.28

Sì, il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
if E1 > E2 then
    begin I := E1 end
else
    begin I := E2 end
while I > 0 do
    begin
    C;
    I := I - 1
    end
end
```

Esercizio 5.29 (*17/09/2024*)

Dire se è possibile estendere il linguaggio WHILE con un comando:

$$X = E \,?\, E_1 \,:\, E_2$$

dove X è una variabile, E è una espressione Booleana ed E_1, E_2 sono espressioni aritmetiche. Informalmente, questo operatore ternario assegna alla variabile X il risultato di E_1 se la condizione in E è verificata, altrimenti gli assegna il valore di E_2.

Soluzione 5.29

Sì, il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
if E then
    begin
    X := E_1
    end
else
    begin
    X := E_2
    end
end
```

Esercizio 5.30 (*05/02/2025*)

Si dica se (e come), il comando:

$$\texttt{do } C - \texttt{ while } E$$

dove C è un comando ed E è una espressione aritmetica, può essere codificato nel linguaggio WHILE. Il comportamento intuitivo è il seguente: si esegue C, poi si valuta l'espressione aritmetica E. Se la valutazione dell'espressione aritmetica è diversa da 0, il corpo del ciclo C viene eseguito di nuovo e successivamente la condizione E è valutata. Se la valutazione di E è uguale a 0 si esce.

Soluzione 5.30

Sì, il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
C;
while E ≠ 0 do
    begin
    C
    end
end
```

Esercizio 5.31 (*26/02/2025*)

Si dica se (e come), il comando:

$$\texttt{while } E \texttt{ count } C \texttt{ in } X$$

dove E è una espressione Booleana, C è un comando e X è una variabile sui numeri naturali può essere codificato nel linguaggio WHILE. Il comportamento intuitivo è il seguente: finché l'espressione Booleana E è vera si esegue C e poi si verifica nuovamente E, altrimenti si esce dal ciclo. Al termine del ciclo, la variabile X deve contenere il numero di iterazioni svolte.

Soluzione 5.31
Sì, il comando è esprimibile nel linguaggio WHILE implementando la macro come segue:

```
begin
X := 0;
while E = True do
    begin
    C;
    X := X + 1
    end
end
```

Macro disponibili

Possiamo considerare tutte gli operatori definiti nel linguaggio Pascal, ed operare alla stessa maniera sia sui naturali che sui numeri interi.

- Assegnamento di costante: $X := n$, con n valore numerico
- Assegnamento di variabile: $X := Y$
- Somma: $X := X + Y$, $X := Z + Y$
- Differenza: $X := X - Y$, $X := Z - Y$
- Differenza troncata: $X := X \dot{-} Y$, $X := Z \dot{-} Y$
- Prodotto: $X := X \cdot Y$, $X := Z \cdot Y$
- Divisione intera: $X := X/Y$, $X := Z/Y$, scritta anche come $X := X$ div Y oppure $X := X \div Y$.
- Resto della divisione intera (operatore di modulo): $X := X \mod Y$, $X := Z \mod Y$, scritto anche come $X := X\%Y$.
- Operatori di confronto: $X \geq Y$, $X \leq Y$, $X = Y$, $X > Y$, $X < Y$ con X, Y numerici
- Costanti booleane: `True`, `False`
- Operatori booleani: $X_\mathcal{E} \wedge Y_\mathcal{E}$, $X_\mathcal{E} \vee Y_\mathcal{E}$, $\neg X_\mathcal{E}$ con $X_\mathcal{E}, Y_\mathcal{E}$ espressioni booleane arbitrarie
- Assegnamento di espressione booleana: $X := T_\mathcal{E}$
- Appartenenza a un insieme o intervallo: $X \in I$
- While con condizione booleana: while $T_\mathcal{E}$ do $COMMAND$
- If-then con condizione booleana: if $T_\mathcal{E}$ then $COMMAND$.

- If-then-else con condizione booleana:

$$\text{if } T_{\mathcal{E}} \text{ then } COMMAND \text{ else } COMMAND$$

- Input di un valore su una variabile: $INPUT(VAR)$
- Output del valore di una variabile: $OUTPUT(VAR)$