Eserciziario per il corso di

Verona, Settembre 2004

Fondamenti dell'Informatica: Linguaggi Formali e Calcolabilità

Dott.ssa Isabella Mastroeni
Dipartimento di Informatica
Università degli Studi di Verona
Strada Le Grazie 15
37134 Verona, Italy
mastroeni@sci.univr.it

Indice

Prefazione

Questa dispensa di esercizi svolti, parte dal presupposto che un corso particolarmente teorico e impegnativo come Fondamenti dell'Informatica venga necessariamente corredato di un supporto applicativo, dove si renda evidente l'utilità concreta di teoremi e lemmi visti a lezione, dei quali, altrimenti, si farebbe fatica a comprendere l'importanza. L'esperienza insegna infatti che molte volte se un teorema complesso è accompaganto da un esempio pratico, questo puó essere un forte incentivo alla comprensione capillare (totale) dell'argomento senza tralasciarne aspetti che potrebbero, ad un primo approccio, apparire come estremamente astratti.

Per cui, uno strumento come questo, è stato pensato e realizzato come uno strumento di approfondimento del corso e non come prima e unica sede di studio: questo leverebbe l'aspetto affascinante di ricerca e di sfida che un corso come Fondamenti dell'Informatica rappresenta.

Desidero ringraziare il Prof. Roberto Giacobazzi, il Prof. Agostino Dovier e la Dott.ssa Mila Dalla Preda per il contributo dato nella stesura della prima versione di questo eserciziario, ora ampiamente rielaborato ed arricchito. Desidero inoltre ringraziare moltissimo tutti gli studenti del corso di *Fondamenti dell'Informatica* degli anni accademici 2001/2002, 2002/2003 e 2003/2004, che con il loro inpegno e studio hanno contribuito al miglioramento di tale eserciziario al fine di renderlo più preciso, corretto e completo nelle spiegazioni. Per questo ringrazio in particolare Nicola Bombieri, Matteo Gozzi e Castellini Alberto.

CAPITOLO 1

Linguaggi Formali

Dato un insieme di simboli Σ, un linguaggio formale è un insieme di stringhe definite a partire da tale alfabeto. Un linguaggio formale può essere *accettato* da un automa a stati finiti (DFA, NFA, ε-NFA), può essere *descritto* mediante una espressione regolare e può essere *generato* mediante una grammatica.

1. Linguaggi regolari

Ricordiamo che gli automi a stati finiti sono completamente descritti dalla matrice di transizione specificando gli stati finali e notando che q_0 rappresenta sempre lo stato iniziale, da cui parte il processo di riconoscimento di un linguaggio eseguito dall'automa. La matrice di transizione sarà rappresentata come una tabella

	s_0	s_1	$s_2 \ldots$
q_0			
q_1		q_2	
$\vdots$			

Questa matrice permette di rappresentare graficamente la relazione δ di transizione. Si ha che un linguaggio è regolare se esiste un automa che lo riconosce (accetta). Per rappresentare graficamente gli automi useremo cerchietti per gli stati (quelli finali saranno cerchiati doppi) e archi etichettati con i simboli dell'alfabeto per rappresentare le transizioni tra i vari stati dovute ai simboli evidenziati nelle etichette. Nei seguenti esercizi chiederemo di dimostrare che un linguaggio è regolare, per fare questo si deve mostrare un automa che lo riconosce e si deve mostrare che effettivamente esso riconosce tutte e sole le stringhe del linguaggio considerato. In tal caso si deve mostrare per induzione sulla lunghezza delle stringhe che ogni stringa è accettata dall'automa se e solo se appartiene al linguaggio. In particolare si dimostra prima che se una stringa sta nel linguaggio allora, nell'automa, termina in uno stato finale, e poi che se una stringa non sta nel linguaggio allora, nell'automa, termina in uno stato non finale. Infine un altro classico esercizio consiste nel minimizzare un automa e per far ciò si esegue una partizione iniziale degli stati in due classi, stati finali e stati non finali. Poi si partizionano ancora le classi, considerando che se q', appartenente alla classe C, con un certo simbolo finisce in uno stato che non appartiene a C, allora q' forma una nuova classe insieme a tutti gli stati di C che con lo stesso simbolo vanno nella stessa classe, altrimenti q' rimane in C. Poi si ripete, nello stesso modo, con le transizioni dovute ad ogni altro simbolo ottenendo una nuova partizione per ciscun simbolo. A questo punto si intersecano le varie partizioni ottenute e si riparte dal risultato. Tutte queste operazioni vengono ripetute finché non si raggiunge il punto fisso del procedimento.
In questo sezione vedremo anche degli esercizi riguardanti le espressioni regolari.

Le espressioni regolari sono espressioni formali che permettono di rappresentare i linguaggi regolari, ovvero ad ogni espressione regolare corrisponde sempre un automa a stati finiti. D'ora in poi useremo la notazione $a^+ \stackrel{\text{def}}{=} aa^* = a^*a$ e si avrà che $a^+ + \varepsilon = a^*$.

ESERCIZIO 1.1. *Si provi che se $\Sigma \neq \emptyset$, allora Σ^* è numerabile.*

SOLUZIONE. Se $\Sigma \neq \emptyset$ e Σ è finito allora occorre individuare una corrispondenza biunivoca con i numeri naturali per mostrare che Σ^* è numerabile. Tale corrispondenza si ottiene, banalmente ordinando i simboli di Σ e iniziando a numerare le stringhe secondo la loro lunghezza, a partire da quella di lunghezza 0 (indicata con ε), proseguendo con quelle di lunghezza uno, che sono in numero finito, a seguire quelle di lunghezza due, anche queste in numero finito ecc. Formalmente consideriamo $\Sigma = \{a_0, \ldots, a_n\}$ e quindi possiamo eseguire le seguenti associazioni.

$$\begin{array}{rcl@{\qquad}rcl}
0 & \to & \varepsilon & n+2 & \to & a_0a_1 \\
1 & \to & a_0 & n+3 & \to & a_0a_2 \\
2 & \to & a_1 & n+4 & \to & a_0a_3 \\
3 & \to & a_2 & \ldots & & \\
\ldots & & & 2n+1 & \to & a_1a_0 \\
n & \to & a_n & 2n+2 & \to & a_1a_1 \\
n+1 & \to & a_0a_0 & \ldots & &
\end{array}$$

Se Σ è infinito per mostrare che l'insieme è numerabile si deve procedere in modo diverso. Ordiniamo le stringhe nel seguente modo:

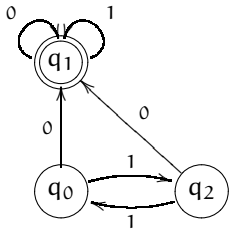
$$\begin{array}{rcl@{\qquad}rcl}
0 & \to & \varepsilon & 11 & \to & a_1a_2 \\
1 & \to & a_0 & 12 & \to & a_2a_2 \\
2 & \to & a_1 & 13 & \to & a_0a_0a_0 \\
3 & \to & a_0a_0 & 14 & \to & a_0a_0a_1 \\
4 & \to & a_0a_1 & 15 & \to & a_0a_1a_0 \\
5 & \to & a_1a_0 & 16 & \to & a_1a_0a_0 \\
6 & \to & a_1a_1 & 17 & \to & a_0a_0a_2 \\
7 & \to & a_2 & 18 & \to & a_0a_2a_0 \\
8 & \to & a_2a_0 & 19 & \to & a_2a_0a_0 \\
9 & \to & a_0a_2 & 20 & \to & a_0a_1a_1 \\
10 & \to & a_2a_1 & 21 & \to & a_1a_0a_1 \\
 & & & \ldots & &
\end{array}$$

L'idea è quella di introdurre un numero finito di elementi scrivendo tutte le possibili stringhe ottenibili con questi elementi, di lunghezza massima pari al numero di elementi introdotti. In questo modo si trattano sempre insiemi finiti. □

ESERCIZIO 1.2. *Si determini il linguaggio accettato dall'automa rappresentato mediante la seguente matrice di transizione dove $F = \{q_1\}$.*

	0	1
q_0	q_1	q_2
q_1	q_1	q_1
q_2	q_1	q_0

SOLUZIONE. Disegnamo l'automa:



Definiamo formalmente l'automa come $M = \langle\{q_0, q_1, q_2\},\{0, 1\}, \delta, q_0,\{q_1\}\rangle$ e, considerando che solo lo 0 porta nello stato finale, deduciamo che il linguaggio riconosciuto è $L(M) = \left\{ x \in \Sigma^* \;\middle|\; \text{in } x \text{ occorre almeno uno } 0 \right\}$. Dimostriamo che questo è il linguaggio effettivamente riconosciuto dall'automa. Verifichiamo cioè:

(1) $x \in L \;\Rightarrow\; x \in L(M)$: $\forall x \in \Sigma^*$ t.c. $x = v0w$ con $v, w \in \Sigma^*$ e v contenente solo 1, allora $\widehat{\delta}(q_0, x) \in F$, cioè $\widehat{\delta}(q_0, x) = q_1$
(2) $x \notin L \;\Rightarrow\; x \notin L(M)$: $\forall x \in \Sigma^*$ t.c. $x = 11\ldots1$(senza 0) allora $\widehat{\delta}(q_0, x) \notin F$, cioè $\widehat{\delta}(q_0, x) \neq q_1$

Innanzitutto, da come è costruito l'automa è banale notare che valgono le seguenti uguaglianze: $\widehat{\delta}(q_0, 1\ldots10) = q_1$ e $\forall w \in \Sigma^* \,.\, \widehat{\delta}(q_1, w) = q_1$. Sia data la stringa $x = v0w$ con $v, w \in \Sigma^*$ e v contenente solo 1 senza 0

$$\begin{aligned} \widehat{\delta}(q_0, x) &= \widehat{\delta}(q_0, v0w) \\ &= \widehat{\delta}(q_0, 1\ldots10w) \\ &= \widehat{\delta}(q_1, w) \\ &= q_1 \in F \end{aligned}$$

A questo punto dimostriamo per induzione sulla lunghezza di x la seconda condizione. Per $|x| = 0$ vale $\widehat{\delta}(q_0, \varepsilon) = q_0 \neq q_1$ (base). Supponiamo ora che se $|x| \leq n$ allora $\widehat{\delta}(q_0, x) \neq q_1$, dunque

$$\begin{aligned} \widehat{\delta}(q_0, x1) &= \delta(\widehat{\delta}(q_0, x), 1) && \text{(Per def. di } \widehat{\delta}) \\ &= \delta(q_i, 1), i \neq 1 && \text{(Per ipotesi induttiva)} \\ &\neq q_1 \end{aligned}$$

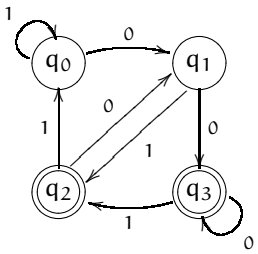
per come è definito l'automa, ovvero da uno stato non finale, con 1, si va sempre in uno stato non finale. □

ESERCIZIO 1.3. *Si verifichi che i seguenti linguaggi, con* $\Sigma = \{0, 1\}$*, sono regolari:*

(1) *l'insieme di tutte le stringhe tali che il penultimo simbolo è 0;*
(2) *l'insieme di tutte le stringhe tali che il terzultimo simbolo è 0;*

SOLUZIONE. Per dimostrare che un certo linguaggio è regolare è sufficiente costruire un automa che lo riconosca, e dimostrare che ciò avviene.

(1) Ecco l'automa che riconosce il primo linguaggio:



Formalmente $M = \langle\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_2, q_3\}\rangle$. Dimostriamo che questo automa effettivamente riconosce l'insieme di tutte le stringhe tali che il penultimo elemento è 0. Per ottenere ciò dimostriamo qualcosa di più forte, ovvero dimostriamo che per $i \in \{0, 1, 2, 3\}$ si ha

$$\widehat{\delta}(q_i, w00) = q_3 \qquad \widehat{\delta}(q_i, w10) = q_1$$
$$\widehat{\delta}(q_i, w01) = q_2 \qquad \widehat{\delta}(q_i, w11) = q_0$$

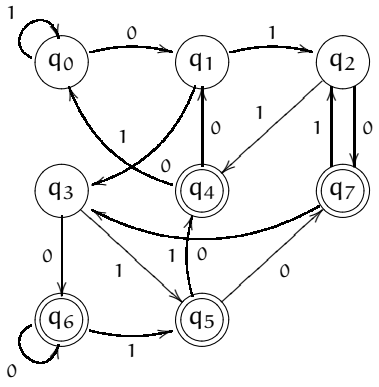
Poiché per definizione di $\widehat{\delta}$ si ha che $\widehat{\delta}(q_i, wu) = \widehat{\delta}(\widehat{\delta}(q_i, w), u) = \widehat{\delta}(q_i', u)$, allora non è più necessaria l'induzione ed è sufficiente dimostrare che per ogni i si ha

$$\widehat{\delta}(q_i, 00) = q_3 \qquad \widehat{\delta}(q_i, 10) = q_1$$
$$\widehat{\delta}(q_i, 01) = q_2 \qquad \widehat{\delta}(q_i, 11) = q_0$$

perché q_i è generico e la proprietà che stiamo cercando di verificare è sugli ultimi simboli. Questo significa che non ci interessa ciò che succede prima di questi simboli e quindi notiamo che

- se $x = 00 \Rightarrow \widehat{\delta}(q_0, 00) = \delta(\widehat{\delta}(q_0, 0), 0) = \delta(\delta(q_0, 0), 0) = \delta(q_1, 0) = q_3 \in F$;
- se $x = 01 \Rightarrow \widehat{\delta}(q_0, 01) = \delta(\widehat{\delta}(q_0, 0), 1) = \delta(\delta(q_0, 0), 1) = \delta(q_1, 1) = q_2 \in F$;
- se $x = 10 \Rightarrow \widehat{\delta}(q_0, 10) = \delta(\widehat{\delta}(q_0, 1), 0) = \delta(\delta(q_0, 1), 0) = \delta(q_0, 0) = q_1 \notin F$;
- se $x = 11 \Rightarrow \widehat{\delta}(q_0, 11) = \delta(\widehat{\delta}(q_0, 1), 1) = \delta(\delta(q_0, 1), 1) = \delta(q_0, 1) = q_0 \notin F$.

(2) Ecco l'automa che riconosce il secondo linguaggio:



da cui $M = \langle\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \{0, 1\}, \delta, q_0, \{q_4, q_5, q_6, q_7\}\rangle$. Dimostriamo ora che effettivamente tale automa riconosce il linguaggio. Per far ciò dobbiamo dimostrare che qualunque stringa in cui il terzultimo elemento è 0 viene accettata dall'automa e che nessn'altra stringa viene accettata. In particolare possiamo dimostrare qualcosa di più forte, ovvero che qualunque sia q_i con $i \in \{0, \ldots, 7\}$ e qualunque sia $w \in \Sigma^*$ si ha che

$$\begin{aligned} \widehat{\delta}(q_i, w000) &= q_6 & \widehat{\delta}(q_i, w100) &= q_3 \\ \widehat{\delta}(q_i, w001) &= q_5 & \widehat{\delta}(q_i, w101) &= q_2 \\ \widehat{\delta}(q_i, w010) &= q_7 & \widehat{\delta}(q_i, w110) &= q_1 \\ \widehat{\delta}(q_i, w011) &= q_4 & \widehat{\delta}(q_i, w111) &= q_0 \end{aligned}$$

Ora poiché per definizione $\widehat{\delta}(q_i, wu) = \widehat{\delta}(\widehat{\delta}(q_i, w), u) = \widehat{\delta}(q_i', u)$, analogamente a prima è sufficiente dimostrare che per ogni i si ha

$$\begin{aligned} \widehat{\delta}(q_i, 000) &= q_6 & \widehat{\delta}(q_i, 100) &= q_3 \\ \widehat{\delta}(q_i, 001) &= q_5 & \widehat{\delta}(q_i, 101) &= q_2 \\ \widehat{\delta}(q_i, 010) &= q_7 & \widehat{\delta}(q_i, 110) &= q_1 \\ \widehat{\delta}(q_i, 011) &= q_4 & \widehat{\delta}(q_i, 111) &= q_0 \end{aligned}$$

e questo si può verificare banalmente dalla definizione dell'automa, come abbiamo visto al punto precedente.

□

Esercizio 1.4.

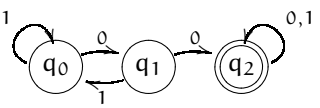
(1) *Si dimostri che il linguaggio composto da stringhe di 0 e 1 tali che:*
   - *ci sono almeno due 0 consecutivi, e*
   - *non vi sono mai due 1 consecutivi*

   è regolare.
(2) *Si determini l'automa minimo che riconosce tale linguaggio.*

Soluzione. Il linguaggio che ci interessa è composto dall'intersezione tra il linguaggio che accetta stringhe con almeno due 0 consecutivi e il linguaggio che accetta stringhe dove non occorrono due 1 consecutivi. Osserviamo che il linguaggio che non accetta stringhe con due 1 consecutivi è il complemento di quello che accetta solo le stringhe con due 1 consecutivi. Definiamo quindi:

$$\begin{aligned} L_1 &= \{w : \ w \text{ ha almeno due 0 consecutivi } \} \\ L_2 &= \{w : \ w \text{ ha almeno due 1 consecutivi } \} \end{aligned}$$

Se L_1, L_2 sono regolari, allora: $L = L_1 \cap \overline{L_2}$ è regolare per la proprietà di chiusura, dove con L indichiamo il linguaggio descritto dall'esercizio. Disegnamo l'automa M_1 relativo al linguaggio L_1:

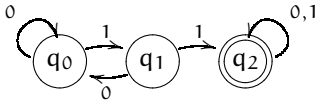


che può essere formalizzato come $M_1 = \langle\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\}\rangle$. Dimostriamo che l'automa M_1 effettivamente riconosce il linguaggio L_1. In particolare, quindi, dobbiamo dimostrare che per ogni stringa x si abbia che $x \in L_1$ se e solo se $x \in L(M_1)$. Per ottenere ciò possiamo dimostrare le seguenti implicazioni: quando

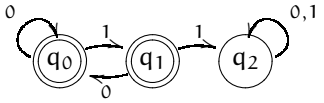
$x \in L$ allora $x \in L(M)$, quando invece $x \notin L$ allora $x \notin L(M)$. Abbiamo quattro casi possibili riguardo alla struttura delle stringhe:

- x non contiene 0 (ovvero $x \notin L$): allora banalmente dalla definizione dell'automa si ha $\widehat{\delta}(q_0, x) = q_0 \notin F$ (ovvero $x \notin L(M)$);
- x termina con uno 0 e prima contiene solo 1 (ovvero $x \notin L$): allora banalmente si verifica che $\widehat{\delta}(q_0, x) = q_1 \notin F$ (ovvero $x \notin L(M)$);
- x contiene 0 sempre seguiti da almeno un 1 (ovvero $x \notin L$): ciò significa che la stringa è del tipo $x = v01w$, dove v non contiene 0. Allora dimostriamo per induzione sulla presenza di 01 nella stringa che lo stato in cui l'automa termina non è finale. Notiamo che $\widehat{\delta}(q_0, 01) = q_0$, per definizione, supponiamo quindi che per ipotesi induttiva $\widehat{\delta}(q_0, w) = q_0$ (dove w non contiene 0 consecutivi), allora $\widehat{\delta}(q_0, v01w) = \widehat{\delta}(\widehat{\delta}(q_0, v), 01w) = \widehat{\delta}(q_0, 01w) = \widehat{\delta}(\widehat{\delta}(q_0, 01), w) = \widehat{\delta}(q_0, w) = q_i \notin F$, in quanto ricade in uno dei casi già visti (ovvero $x \notin L(M)$);
- x contiene due 0 consecutivi (ovvero $x \in L$): $x = v00w$, dove v non contiene 0 consecutivi (cade in uno dei casi precedenti ma non termina con 0 altrimenti gli 0 evidenziati non sarebbero i primi consecutivi), quindi $\widehat{\delta}(q_0, v) = q_0$, inoltre è evidente che per ogni $w \in \Sigma^*$ si ha $\widehat{\delta}(q_2, w) = q_2$, da cui $\widehat{\delta}(q_0, v00w) = \widehat{\delta}(q_0, 00w) = \widehat{\delta}(q_1, 0w) = \widehat{\delta}(q_2, w) = q_2$ (ovvero $x \in L(M)$).

Disegnamo ora l'automa M_2 relativo al linguaggio L_2.



formalizzato: $M_2 = \langle \{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\} \rangle$. La dimostrazione che l'automa M_2 effettivamente riconosce il linguaggio L_2 è analoga alla precedente. L'automa M_3, che vogliamo riconosca il linguaggio complementare a L_2 (cioè tale che $L(M_3) = \overline{L(M_2)}$), si ottiene facilmente dall'automa M_2 invertendo stati finali e non finali.

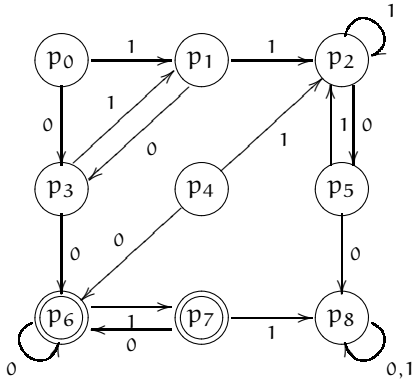


e lo si formalizza come $M_3 = \langle \{q_0', q_1', q_2'\}, \{0, 1\}, \delta, q_0', \{q_0', q_1'\} \rangle$. Dato che i linguaggi regolari sono chiusi rispetto all'intersezione proviamo a costruire l'automa richiesto dall'esercizio in modo automatico a partire da M_1 e M_3 (l'idea è: $M = M_1 \cap M_3$)
Siano $M_1 = \langle Q_1, \Sigma, \delta_1, F_1, q_0 \rangle$ e $M_3 = \langle Q_3, \Sigma, \delta_3, F_3, q_0' \rangle$ gli automi sopra definiti. Si costruisce l'automa che ha come stati il prodotto cartesiano degli stati dei due automi, quindi ogni stato del nuovo automa sarà una coppia del tipo $\langle q_i, q_j \rangle$ con $q_i \in Q_1$ e $q_j \in Q_3$. L'insieme degli stati finali sarà l'insieme costituito da coppie di stati finali, $F = \{ \langle q_i, q_j \rangle \mid q_i \in F_1,\ q_j \in F_3 \}$ (se si fosse interessati all'unione, basterebbe sostituire $\vee$ al posto di $\wedge$). È evidente che lo stato (q_i, q_j), attraverso un simbolo a, raggiunge lo stato (q_k, q_h) solo se $\delta_1(q_i, a) = q_k$ e $\delta_3(q_j, a) = q_h$.

Possiamo allora scrivere la matrice di transizione di M:

	0	1
$p_0 = (q_0, q_0')$	(q_1, q_0')	(q_0, q_1')
$p_1 = (q_0, q_1')$	(q_1, q_0')	(q_0, q_2')
$p_2 = (q_0, q_2')$	(q_1, q_2')	(q_0, q_2')
$p_3 = (q_1, q_0')$	(q_2, q_0')	(q_0, q_1')
$p_4 = (q_1, q_1')$	(q_2, q_0')	(q_0, q_2')
$p_5 = (q_1, q_2')$	(q_2, q_2')	(q_0, q_2')
$p_6 = (q_2, q_0')$	(q_2, q_0')	(q_2, q_1')
$p_7 = (q_2, q_1')$	(q_2, q_0')	(q_2, q_2')
$p_8 = (q_2, q_2')$	(q_2, q_2')	(q_2, q_2')

dove $p_0 \dots p_8$ sono i nomi che daremo agli stati del nuovo automa che risulta essere:



formalizzato $M = \langle\{p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}, \{0, 1\}, \delta, p_0, \{p_6, p_7\}\rangle$. Non resta che minimizzare l'automa, utilizzando l'algoritmo di minimizzazione.
Iniziamo dividendo gli stati dell'automa in due classi: $C_1 = \{p_6, p_7\}$ e $C_2 = \{p_0, p_1, p_2, p_3, p_4, p_5, p_8\}$, dove C_1 raccoglie gli stati finali e C_2 quelli non finali. Partizioniamo ora le classi, considerando che se partendo da q', con 0 si finisce in uno stato che non appartiene alla classe a cui appartiene q', allora q' forma una nuova classe insieme a tutti quelli della sua classe che con lo stesso simbolo vanno nella stessa classe, altrimenti rimane in quella di partenza. Poi si ripete con le transizioni dovute a 1 ottenendo una nuova partizione. A questo punto si intersecano le due partizioni ottenute e si riparte dal risultato. In Fig. 1 abbiamo rappresentato l'intero processo di minimizzazione, dove in tutte le diramazioni quelle di sinistra rappresentano le transizioni dovute allo 0, mentre quelle di destra rappresentano le transizioni dovute a 1. In tale figura osserviamo che l'ultima partizione è quella finale; si nota che gli stati p_2, p_5, p_8 possono essere considerati come un unico stato che indichiamo con $\perp$. Per semplificare l'automa notiamo che p_4 non ha archi entranti, ovvero non può essere mai raggiunto, quindi lo togliamo. L'automa minimo risulta perciò essere il seguente.

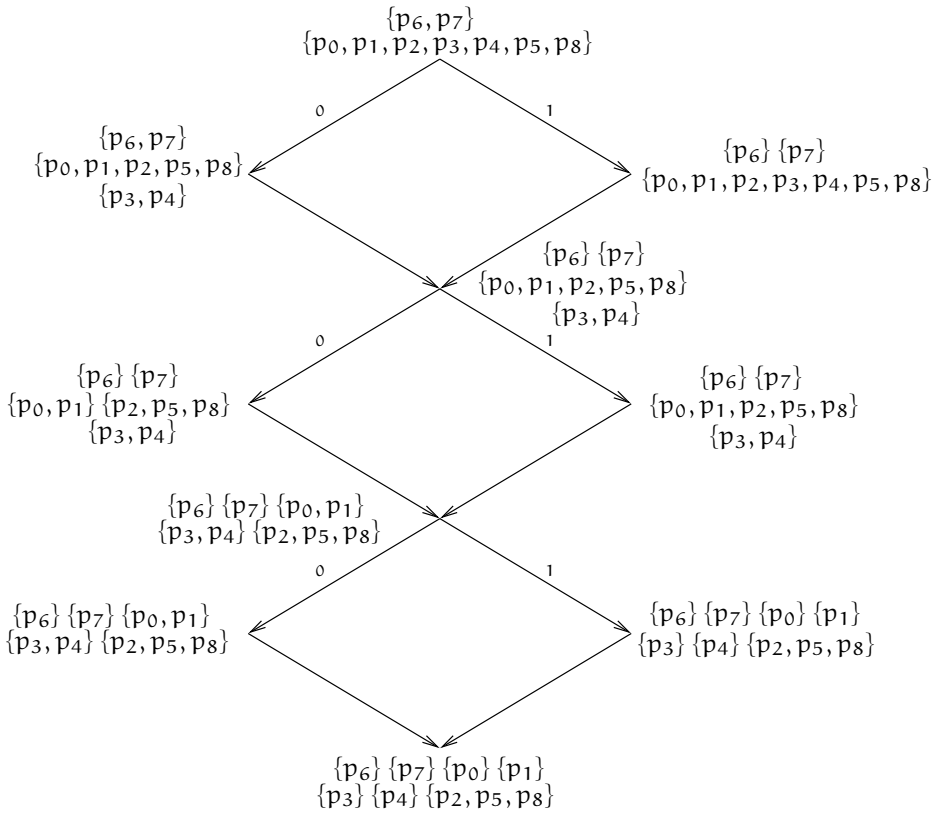
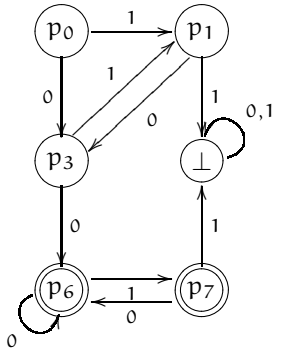


FIGURA 1. Processo di minimizzazione dell'Esercizio 1.4



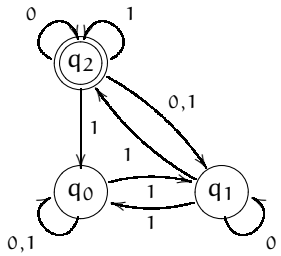
che formalizzato diventa $M = \langle\{p_0, p_1, p_3, p_6, p_7, \bot\}, \{0, 1\}, \delta, p_0, \{p_6, p_7\}\rangle$. □

ESERCIZIO 1.5. *Si determini il DFA equivalente all'NFA:*

	0	1
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_1\}$	$\{q_0, q_2\}$
q_2	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$

ove $F = \{q_2\}$. *Qual è il linguaggio accettato?*

Soluzione. Ecco il grafico dell'NFA.



Da cui $M = \langle\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\}\rangle$. Ricordiamo che, dato un NFA, $M = \langle Q, \Sigma, \delta, q_0, F\rangle$, esiste sempre un DFA $M' = \langle \wp(Q), \Sigma, \delta', \{q_0\}, F'\rangle$ equivalente, dove:

- $F' = \{P \subseteq Q : P \cap F \neq \emptyset\}$;
- $\delta'(P, a) = \bigcup_{p \in P} \delta(p, a)$, per $P \in \wp(Q)$.

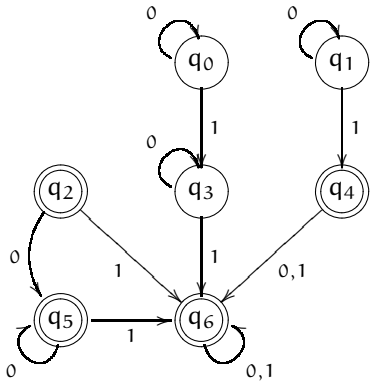
Costruiamo la matrice di transizione per il DFA:

		0	1
	$\emptyset$	$\emptyset$	$\emptyset$
q_0'	$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
q_1'	$\{q_1\}$	$\{q_1\}$	$\{q_0, q_2\}$
q_2'	$\{q_2\}$	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$
q_3'	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$
q_4'	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$
q_5'	$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$
q_6'	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$

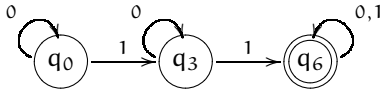
Quindi (eliminando gli apici per semplicità) si ha $F = \{q_2, q_4, q_5, q_6\}$ e la matrice diventa

	0	1
q_0	q_0	q_3
q_1	q_1	q_4
q_2	q_5	q_6
q_3	q_3	q_6
q_4	q_6	q_6
q_5	q_5	q_6
q_6	q_6	q_6

Possiamo quindi disegnare il DFA,



che formalizzato è $M = \langle\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{0, 1\}, \delta, q_0, \{q_2, q_4, q_5, q_6\}\rangle$. Notiamo che gli stati associati a q_1, q_2, q_4, q_5 possono essere eliminati, dal momento che non verranno mai raggiunti a partire dallo stato iniziale. In questo modo otteniamo il DFA cercato.

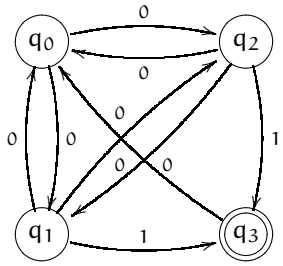


che formalizzato diventa $M = \langle\{q_0, q_3, q_6\}, \{0, 1\}, \delta, q_0, \{q_6\}\rangle$. A questo punto si può dimostrare che il linguaggio accettato è il linguaggio descritto dall'insieme di stringhe $L = \left\{ x \,\middle|\, x \text{ contiene almeno due } 1 \right\}$. Come negli esercizi precedenti dobbiamo dimostrare due implicazioni: quando $x \in L$ allora $x \in L(M)$ e quando $x \notin L$ allora $x \notin L(M)$. Possiamo notare che rispetto alla quantità di 1, una generica stringa può essere di tre tipi:

- x non contiene 1 (ovvero $x \notin L$): In tal caso abbiamo $x = 0 \dots 0$, per come è definito l'automa, questo significa che $\widehat{\delta}(q_0, x) = q_0 \notin F$ (ovvero $x \notin L(M)$);
- x contiene un solo 1 (ovvero $x \notin L$): In tal caso abbiamo $x = u1v$ con $u, v \in 0^*$, allora per come è definito l'automa abbiamo che $\widehat{\delta}(q_0, u1v) = \widehat{\delta}(q_0, 1v) = \widehat{\delta}(q_3, v) = q_3 \notin F$ (ovvero $x \notin L(M)$);
- x contiene almeno due 1 (ovvero $x \notin L$): In tal caso abbiamo $x = u1v1w$ dove $u, v \in 0^*$ e $w \in (1+0)^*$, allora per la definizione dell'automa abbiamo che $\widehat{\delta}(q_0, u1v1w) = \widehat{\delta}(q_0, 1v1w) = \widehat{\delta}(q_3, v1w) = \widehat{\delta}(q_3, 1w) = \widehat{\delta}(q_6, w) = q_6 \in F$ (ovvero $x \in L(M)$).

□

ESERCIZIO 1.6. *Si determini, usando le tecniche di trasformazione standard, il DFA minimo equivalente all'automa*



Si determini inoltre, dimostrando formalmente la propria affermazione, il linguaggio riconosciuto dal DFA minimo calcolato.

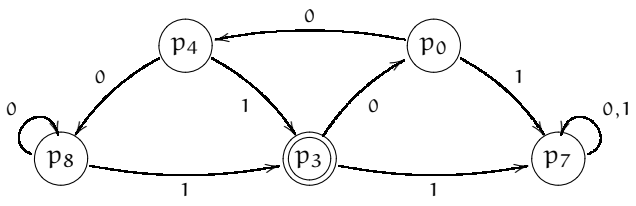
Soluzione. Per prima cosa costruiamo la matrice di transizione dell'NFA:

	0	1
q_0	$\{q_1, q_2\}$	$\{\emptyset\}$
q_1	$\{q_0, q_2\}$	$\{q_3\}$
q_2	$\{q_1, q_0\}$	$\{q_3\}$
q_3	$\{q_0\}$	$\{\emptyset\}$

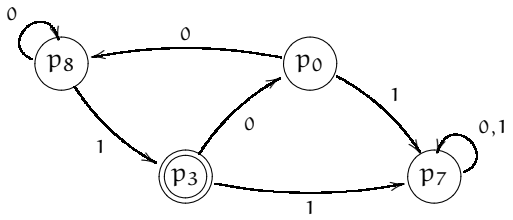
Definiamo adesso gli stati del DFA, ma non prendiamo tutti i possibili sottoinsiemi di $\{q_0, q_1, q_2, q_3\}$, prendiamo solo quelli che effettivamente vengono condiderati dall'automa. Siano innanzitutto $p_0 = \{q_0\}$, $p_1 = \{q_1\}$, $p_2 = \{q_2\}$, $p_3 = \{q_3\}$, le cui transizioni sono quelle definite nella tabella precedente. Poi prendiamo $p_4 = \{q_1, q_2\}$, $p_5 = \{q_0, q_2\}$, $p_6 = \{q_0, q_1\}$ e $p_7 = \emptyset$. Infine prendiamo l'unione di quelli raggiungibili a partire da questi, ovvero $p_8 = \{q_0, q_1, q_2\}$. Notiamo che l'unico stato contenente q_3, ovvero finale, è p_3. A questo punto possiamo costruire la nuova matrice

	0	1
p_0	p_4	p_7
p_1	p_5	p_3
p_2	p_6	p_3
p_3	p_0	p_7
p_4	p_8	p_3
p_5	p_8	p_3
p_6	p_8	p_3
p_7	p_7	p_7
p_8	p_8	p_3

Ora disegnamo l'automa solo con gli stati effettivamente raggiungibili a partire da p_0.



Ora dobbiamo minimizzare tale automa. Partiamo dalla suddivisione tra stati finali e non: $\{p_0, p_4, p_7, p_8\}$ e $\{p_3\}$. Vediamo che con 0 la suddivisione rimane inalterata, mantre con 1 è evidente che $\{p_4, p_8\}$ si divide da $\{p_0, p_7\}$. Otteniamo quindi la partizione $\{p_3\}$, $\{p_0, p_7\}$ e $\{p_4, p_8\}$. Da questa l'unica divisione avviene ancora con 0 che distingue gli stati p_0 e p_7, otteniamo quindi la suddivisione finale $\{p_0\}$, $\{p_3\}$, $\{p_7\}$, $\{p_4, p_8\}$. Per cui l'automa finale minimo è



Vediamo infine che il linguaggio riconosciuto dall'automa è

$$L = \left\{ x \in \Sigma^* \;\middle|\; |x| \geq 2, x = 0y1 \text{ con } y \in \Sigma^* \text{ e in } y \text{ ogni } 1 \text{ è seguito da almeno due } 0 \right\}$$

Dimostriamo per casi che se $x \in L$ allora $x \in L(M)$ e che se $x \notin L$ allora $x \notin L(M)$.

- Se $|x| < 2$ allora $x \notin L$ e nell'automa abbiamo che $\widehat{\delta}(p_0, x) \notin F$;
- Se $x = y0$ allora $x \notin L$ ed effettivamente per ogni p_i tale che $\widehat{\delta}(p_0, y) = p_i$ allora si ha, per definizione di M che $\delta(p_i, 0) \notin F$;
- Se $x = 1y$ allora $x \notin L$ ed effettivamente $\widehat{\delta}(p_0, 1y) = \widehat{\delta}(p_7, y) = p_7 \notin F$;
- Sia $x = 0y1$, allora vediamo per induzione su $|y|$ che se y contiene almeno un 1 seguito da 01 o $1a$ allora $\widehat{\delta}(p_8, y) = p_7$, se invece ogni 1 è seguito da almeno due 0 ma y termina con 1 allora $\widehat{\delta}(p_8, y) = p_3$, se invece termina con 10 allora $\widehat{\delta}(p_8, y) = p_0$, nell'ultimo caso invece $\widehat{\delta}(p_8, y) = p_8$.
  Sia $|y| = 3$, se $y = a00$, allora $\widehat{\delta}(p_8, a00) = p_8$, se $y = 101$, allora $\widehat{\delta}(p_8, 101) = p_7$, se $y = 11a$ allora $\widehat{\delta}(p_8, 10a) = p_7$ qualunque sia a, se $y = 001$ allora $\widehat{\delta}(p_8, 001) = p_3$ e se infine $y = 010$ allora $\widehat{\delta}(p_8, 010) = p_0$.
  Consideriamo ora valida l'ipotesi induttiva per z tale che $|z| = n$, prendiamo ora $y = za$, se in z ogni 1 è seguito da almeno due 0 e terminante con 1, allora $\widehat{\delta}(p_8, z) = p_3$, se $a = 0$ allora y termina con 10 ed effettivamente $\delta(p_3, 0) = p_0$; se invece $a = 1$ allora si hanno due 1 consecutivi ed effettivamente $\delta(p_3, 1) = p_7$. Se invece z è terminante con 10 allora $\widehat{\delta}(p_8, z) = p_0$ quindi se $a = 0$ ritroviamo che ogni 1 è seguito da almeno due 0 ed effettivamente $\delta(p_0, 0) = p_8$; se invece $a = 1$, allora abbiamo un 1 seguito da 01 e infatti $\delta(p_0, 1) = p_7$. Infine se tutti gli 1 sono seguiti da almeno due 0 allora $\widehat{\delta}(p_8, z) = p_8$, quindi se $a = 0$ abbiamo $\delta(p_8, 0) = p_8$, mentre se $a = 1$, allora abbiamo y terminante con 1 ed effettivamente $\delta(p_8, 1) = p_3$. Sia ora z con un 1 seguito da 01 o $1b$, allora $\widehat{\delta}(p_8, z) = p_7$ e, qualunque sia a non cambia la situazione, infatti $\delta(p_7, a) = p_7$. Ora se in y ogni 1 è seguito sempre da almeno due 0 allora $x \in L$ e nell'automa abbiamo che $\widehat{\delta}(p_0, 0y1) = \widehat{\delta}(p_8, y1) = \delta(p_8, 1) = p_3 \in F$. Altrimenti per ogni $p_i \in \{p_0, p_3, p_7\}$ abbiamo che $\widehat{\delta}(p_0, 0y1) = \widehat{\delta}(p_8, y1) = \delta(p_i, 1) = p_7 \notin F$.

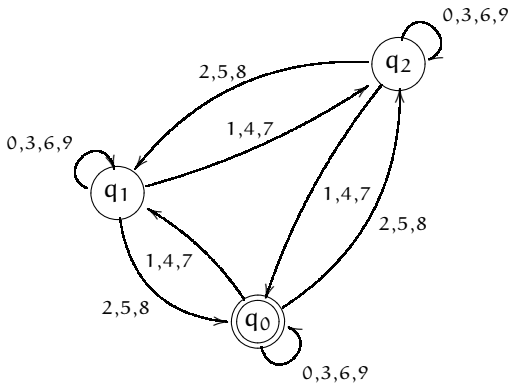
□

ESERCIZIO 1.7. *Sia* $\Sigma = \{0, \ldots, 9\}$. *Si dimostri che il linguaggio:*

$$L = \{\varepsilon\} \cup \left\{ a_n \ldots a_0 : a_i \in \Sigma, n \geq 0, a_n \neq 0, \left(\sum_{i=0}^{n} a_i 10^i \right) \bmod 3 = 0 \right\}$$

è regolare. Si dia inoltre una dimostrazione informale di correttezza dell'automa (ovvero si spieghi il principio usato per la costruzione dell'automa). Si riscriva poi l'automa con l'alfabeto $\Sigma = \{0, 1, 2\}$ *e si dia una dimostrazione formale di correttezza per lo stesso linguaggio.*

SOLUZIONE. Dobbiamo determinare un automa che accetti solo le stringhe di cifre in base 10 che rappresentano numeri divisibili per 3. Si sa che un numero in base 10 è divisibile per 3 se la somma delle cifre che lo compongono è ancora divisibile per 3; questa è l'idea che sta alla base del seguente automa.



L'idea è quella di considerare che una cifra divisibile per tre $(0, 3, 6, 9)$ non cambia lo stato di divisibilità per tre del numero. Quando invece abbiamo una cifra che aggiunge una unità rispetto alla divisibilità per tre $(1, 4, 7)$ allora per terminare in uno stato finale abbiamo bisogno di un'altra cifra che aggiunga due unità $(2, 5, 8)$ ad un multiplo di tre. Analogo il viceversa, in tal modo siamo sicuri che la somma delle sequenze di numeri che portano in uno stato finale è un multiplo di tre.
Riscriviamo ora l'utoma con l'alfabeto ristretto a $\Sigma = \{0, 1, 2\}$ (vedi figura a pag.20). Dimostriamo per induzione sulla lunghezza della stringa x che se $x \in L$, ovvero $\Sigma_i x_i \bmod 3 = 0$, allora $x \in L(M)$ (dove M è l'ultimo automa disegnato) e se $x \notin L$, ovvero $\Sigma_i x_i \bmod 3 \neq 0$, allora $x \notin L(M)$. In particolare dimostriamo che se $\Sigma_i x_i \bmod 3 = 1$ allora $\widehat{\delta}(q_0, x) = q_1$ e se $\Sigma_i x_i \bmod 3 = 2$ allora $\widehat{\delta}(q_0, x) = q_2$.

Base: Sia $|x| = 1$, allora abbiamo tre casi possibili:

Se $x = 0$ (ovvero $x \in L$), allora $\delta(q_0, 0) = q_0 \in F$
Se $x = 1$ (ovvero $x \notin L$), allora $\delta(q_0, 1) = q_1 \notin F$
Se $x = 2$ (ovvero $x \notin L$), allora $\delta(q_0, 2) = q_2 \notin F$

Induzione: Sia valido l'enunciato per ogni stringa di lunghezza n, consideriamo $x = va$ tale che $|x| = |v| + 1 = n + 1$. Questo significa che vale l'ipotesi induttiva per v. Dunque possiamo distinguere tre casi:

- $y \in L$: In tal caso, per ipotesi, abbiamo che $\Sigma_i y_i \bmod 3 = 0$, e per ipotesi induttiva abbiamo che $\widehat{\delta}(q_0, y) = q_0$, quindi possiamo

distinguere ancora tre casi:

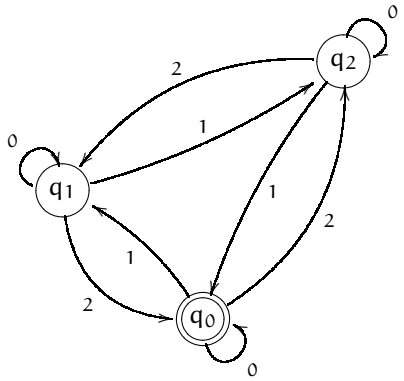
Se $a = 0$ allora $(\Sigma_i y_i + a) \bmod 3 = 0$, ovvero $x \in L$,
allora $\widehat{\delta}(q_0, y0) = \delta(q_0, 0) = q_0 \in F$
Se $a = 1$ allora $(\Sigma_i y_i + a) \bmod 3 = 1$, ovvero $x \notin L$,
allora $\widehat{\delta}(q_0, y1) = \delta(q_0, 1) = q_1 \notin F$
Se $a = 2$ allora $(\Sigma_i y_i + a) \bmod 3 = 2$, ovvero $x \notin L$,
allora $\widehat{\delta}(q_0, y2) = \delta(q_0, 2) = q_2 \notin F$

- $y \notin L$ e $\widehat{\delta}(q_0, y) = q_1$: In tal caso, per ipotesi induttiva, abbiamo che $\Sigma_i y_i \bmod 3 = 1$. Abbiamo perciò tre casi:

  Se $a = 0$ allora $(\Sigma_i y_i + a) \bmod 3 = 1$, ovvero $x \notin L$,
  allora $\widehat{\delta}(q_0, y0) = \delta(q_1, 0) = q_1 \notin F$
  Se $a = 1$ allora $(\Sigma_i y_i + a) \bmod 3 = 2$, ovvero $x \notin L$,
  allora $\widehat{\delta}(q_0, y1) = \delta(q_1, 1) = q_2 \notin F$
  Se $a = 2$ allora $(\Sigma_i y_i + a) \bmod 3 = 0$, ovvero $x \in L$,
  allora $\widehat{\delta}(q_0, y2) = \delta(q_1, 2) = q_0 \in F$

- $y \notin L$ e $\widehat{\delta}(q_0, y) = q_2$: In tal caso, per ipotesi induttiva, abbiamo che $\Sigma_i y_i \bmod 3 = 2$. Abbiamo perciò tre casi:

  Se $a = 0$ allora $(\Sigma_i y_i + a) \bmod 3 = 2$, ovvero $x \notin L$,
  allora $\widehat{\delta}(q_0, y0) = \delta(q_2, 0) = q_2 \notin F$
  Se $a = 1$ allora $(\Sigma_i y_i + a) \bmod 3 = 0$, ovvero $x \in L$,
  allora $\widehat{\delta}(q_0, y1) = \delta(q_2, 1) = q_0 \in F$
  Se $a = 2$ allora $(\Sigma_i y_i + a) \bmod 3 = 1$, ovvero $x \notin L$,
  allora $\widehat{\delta}(q_0, y2) = \delta(q_2, 2) = q_1 \notin F$



□

Esercizio 1.8. *Si verifichi che il seguente linguaggio è regolare:*

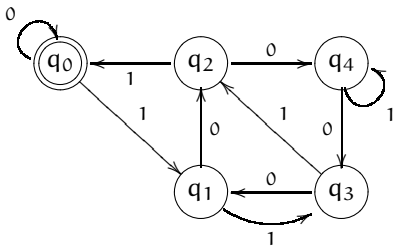
$$L = \left\{ \; x \in \{0,1\}^* \;\middle|\; x^{\#} \bmod 5 = 0 \; \right\}$$

dove $x^{\#}$ *è il numero decimale rappresentato dalla stringa binaria* x *e* `mod` *rappresenta l'operazione che restituisce il resto della divisione intera tra i numeri a cui è applicata.*

SOLUZIONE. Ricordiamo che, data la stringa binaria $x = x_0x_1 \dots x_{|x|-1}$, possiamo calcolare il decimale rappresentato nel seguente modo: $x^{\#} = \Sigma_{i=|x|-1}^{i=0} x_i * 2^{|x|-i-1}$. Inoltre, questa corrispondenza ci dice anche che se la stringa binaria y è tale che $y = x0$, allora $y^{\#} = x^{\#} * 2$, mentre se $y = x1$ allora $y^{\#} = (x^{\#} * 2) + 1$. Questi fatti sono importanti per capire come interpretare la lettura di un 1 o di uno 0 in una stringa binaria rappresentante numeri decimali. A questo punto dobbiamo determinare l'automa che riconosce le stringhe binarie rappresentanti numeri decimali divisibili per 5, ovvero tali che il loro modulo nella divisione per 5 è zero. Può essere utile capire cosa succede proprio al modulo nella divisione per 5 quando moltiplichiamo per 2 (ovvero leggiamo uno 0 nella rappresentazione binaria) o moltiplichiamo per 2 e sommiamo 1 (ovvero leggiamo un 1 nella rappresentazione binaria):

$$\begin{array}{lcll}
n \bmod 5 = 0 & \Rightarrow & (n * 2 \bmod 5 = 0 & \text{e} \quad n * 2 + 1 \bmod 5 = 1) \\
n \bmod 5 = 1 & \Rightarrow & (n * 2 \bmod 5 = 2 & \text{e} \quad n * 2 + 1 \bmod 5 = 3) \\
n \bmod 5 = 2 & \Rightarrow & (n * 2 \bmod 5 = 4 & \text{e} \quad n * 2 + 1 \bmod 5 = 0) \\
n \bmod 5 = 3 & \Rightarrow & (n * 2 \bmod 5 = 1 & \text{e} \quad n * 2 + 1 \bmod 5 = 2) \\
n \bmod 5 = 4 & \Rightarrow & (n * 2 \bmod 5 = 3 & \text{e} \quad n * 2 + 1 \bmod 5 = 4)
\end{array}$$

Queste relazioni ci suggeriscono di costruire l'automa in modo che ogni stato riconosca le stringhe con un particolare modulo nella divisione per 5. Con questa idea costruiamo il seguente automa:



A questo punto, per dimostrare la correttezza dell'automa, dimostriamo per induzione qualcosa di più forte, ovvero dimostriamo i seguenti fatti:

$$\begin{array}{lcl}
x^{\#} \bmod 5 = 0 & \Rightarrow & \widehat{\delta}(q_0, x) = q_0 \\
x^{\#} \bmod 5 = 1 & \Rightarrow & \widehat{\delta}(q_0, x) = q_1 \\
x^{\#} \bmod 5 = 2 & \Rightarrow & \widehat{\delta}(q_0, x) = q_2 \\
x^{\#} \bmod 5 = 3 & \Rightarrow & \widehat{\delta}(q_0, x) = q_3 \\
x^{\#} \bmod 5 = 4 & \Rightarrow & \widehat{\delta}(q_0, x) = q_4
\end{array}$$

Base: Consideriamo le stringhe binarie rappresentanti i più piccoli numeri naturali con diverso modulo nella divisione per 5, ovvero i numeri che vanno da 0 a 4:

$$\begin{array}{lcl}
x = 0 \;\Rightarrow\; x^{\#} \bmod 5 = 0 & \text{e} & \delta(q_0, 0) = q_0 \\
x = 1 \;\Rightarrow\; x^{\#} \bmod 5 = 1 & \text{e} & \delta(q_0, 1) = q_1 \\
x = 10 \;\Rightarrow\; x^{\#} \bmod 5 = 2 & \text{e} & \widehat{\delta}(q_0, 10) = \delta(q_1, 0) = q_2 \\
x = 11 \;\Rightarrow\; x^{\#} \bmod 5 = 3 & \text{e} & \widehat{\delta}(q_0, 11) = \delta(q_1, 1) = q_3 \\
x = 100 \;\Rightarrow\; x^{\#} \bmod 5 = 4 & \text{e} & \widehat{\delta}(q_0, 100) = \widehat{\delta}(q_1, 00) = \delta(q_2, 0) = q_4
\end{array}$$

Induzione: Consideriamo ora una generica stringa x di lunghezza n e supponiamo che per quasta valga l'ipotesi induttiva. Dimostriamo allora che

la tesi vale anche per le stringhe y di lunghezza $n+1$. Costruiamo la dimostrazione per casi, a seconda delle ipotesi su x e a seconda di come otteniamo y a partire da x, ovvero consideriamo $y = xa$ con $a \in \{0,1\}$.

$x \in L$ **(e** $x^{\#} \bmod 5 = 0$**):**

Sia $y = x0$, allora abbiamo che $y^{\#} = x^{\#} * 2$ e quindi $y^{\#} \bmod 5 = 0$, questo implica che $y \in L$. Ora notiamo che, applicando l'ipotesi induttiva $\widehat{\delta}(q_0, x) = q_0$, otteniamo la tesi $\widehat{\delta}(q_0, y) = \widehat{\delta}(q_0, x0) = \delta(q_0, 0) = q_0$. Sia $y = x1$, allora abbiamo che $y^{\#} = x^{\#} * 2 + 1$ e quindi $y^{\#} \bmod 5 = 1$, questo implica che $y \notin L$. Ora notiamo che, applicando l'ipotesi induttiva $\widehat{\delta}(q_0, x) = q_0$, otteniamo la tesi $\widehat{\delta}(q_0, y) = \widehat{\delta}(q_0, x1) = \delta(q_0, 1) = q_1$.

$x \notin L$ **e** $x^{\#} \bmod 5 = 1$**:**

Sia $y = x0$, allora abbiamo che $y^{\#} = x^{\#} * 2$ e quindi $y^{\#} \bmod 5 = 2$, questo implica che $y \notin L$. Ora notiamo che, applicando l'ipotesi induttiva $\widehat{\delta}(q_0, x) = q_1$, otteniamo la tesi $\widehat{\delta}(q_0, y) = \widehat{\delta}(q_0, x0) = \delta(q_1, 0) = q_2$. Sia $y = x1$, allora abbiamo che $y^{\#} = x^{\#} * 2 + 1$ e quindi $y^{\#} \bmod 5 = 3$, questo implica che $y \notin L$. Ora notiamo che, applicando l'ipotesi induttiva $\widehat{\delta}(q_0, x) = q_1$, otteniamo la tesi $\widehat{\delta}(q_0, y) = \widehat{\delta}(q_0, x1) = \delta(q_1, 1) = q_3$.

$x \notin L$ **e** $x^{\#} \bmod 5 = 2$**:**

Sia $y = x0$, allora abbiamo che $y^{\#} = x^{\#} * 2$ e quindi $y^{\#} \bmod 5 = 4$, questo implica che $y \notin L$. Ora notiamo che, applicando l'ipotesi induttiva $\widehat{\delta}(q_0, x) = q_2$, otteniamo la tesi $\widehat{\delta}(q_0, y) = \widehat{\delta}(q_0, x0) = \delta(q_2, 0) = q_4$. Sia $y = x1$, allora abbiamo che $y^{\#} = x^{\#} * 2 + 1$ e quindi $y^{\#} \bmod 5 = 0$, questo implica che $y \in L$. Ora notiamo che, applicando l'ipotesi induttiva $\widehat{\delta}(q_0, x) = q_2$, otteniamo la tesi $\widehat{\delta}(q_0, y) = \widehat{\delta}(q_0, x1) = \delta(q_2, 1) = q_0$.

$x \notin L$ **e** $x^{\#} \bmod 5 = 3$**:**

Sia $y = x0$, allora abbiamo che $y^{\#} = x^{\#} * 2$ e quindi $y^{\#} \bmod 5 = 1$, questo implica che $y \notin L$. Ora notiamo che, applicando l'ipotesi induttiva $\widehat{\delta}(q_0, x) = q_3$, otteniamo la tesi $\widehat{\delta}(q_0, y) = \widehat{\delta}(q_0, x0) = \delta(q_3, 0) = q_1$. Sia $y = x1$, allora abbiamo che $y^{\#} = x^{\#} * 2 + 1$ e quindi $y^{\#} \bmod 5 = 2$, questo implica che $y \notin L$. Ora notiamo che, applicando l'ipotesi induttiva $\widehat{\delta}(q_0, x) = q_3$, otteniamo la tesi $\widehat{\delta}(q_0, y) = \widehat{\delta}(q_0, x1) = \delta(q_3, 1) = q_2$.

$x \notin L$ **e** $x^{\#} \bmod 5 = 4$**:**

Sia $y = x0$, allora abbiamo che $y^{\#} = x^{\#} * 2$ e quindi $y^{\#} \bmod 5 = 3$, questo implica che $y \notin L$. Ora notiamo che, applicando l'ipotesi induttiva $\widehat{\delta}(q_0, x) = q_4$, otteniamo la tesi $\widehat{\delta}(q_0, y) = \widehat{\delta}(q_0, x0) = \delta(q_4, 0) = q_3$. Sia $y = x1$, allora abbiamo che $y^{\#} = x^{\#} * 2 + 1$ e quindi $y^{\#} \bmod 5 = 4$, questo implica che $y \notin L$. Ora notiamo che, applicando l'ipotesi induttiva $\widehat{\delta}(q_0, x) = q_4$, otteniamo la tesi $\widehat{\delta}(q_0, y) = \widehat{\delta}(q_0, x1) = \delta(q_4, 1) = q_4$.

□

ESERCIZIO 1.9. *Si provino o refutino le seguenti identità:*

(1) $r + s = s + r$
(2) $r(st) = (rs)t$
(3) $r^*(r + t)^* = (r + t)^*$
(4) $\emptyset^* = \varepsilon$

SOLUZIONE.

(1) $r + s = s + r$.
Siano R e S gli insiemi associati alle espressioni regolari r e s. Per definizione si ha che $x \in R + S$ se e solo se $x \in R \vee x \in S$, che equivale a dire $x \in R + S$ se e solo se $x \in R \cup S$. Possiamo perciò concludere che $r + s$ implica $R \cup S$ e che $s + r$ implica $S \cup R$, dove $R \cup S = S \cup R$ per la commutatività dell'unione.
(2) $r(st) = (rs)t$.
Siano R, S e T gli insiemi associati alle espressioni regolari r, s e t. Si deve mostrare che $R(ST) = (RS)T$, ovvero che $x \in R(ST)$ se e solo se $x \in (RS)T$:

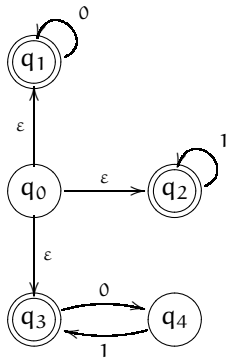
$$\begin{aligned} x \in R(ST) &\Leftrightarrow x = ry \text{ con } r \in R, y \in ST \\ &\Leftrightarrow x = rst \text{ con } r \in R, s \in S, t \in T \\ &\Leftrightarrow x = vt \text{ con } v \in RS, t \in T \\ &\Leftrightarrow x \in (RS)T \end{aligned}$$

(3) $r^*(r + t)^* = (r + t)^*$.
Siano R e T gli insiemi associati alle espressioni regolari r e t. Si deve mostrare che $R^*(R + T)^* = (R + T)^*$. Per ottenere questo verifichiamo l'inclusione nei due sensi. Dimostriamo $(R + T)^* \subseteq R^*(R + T)^*$ Sia $x \in (R + T)^*$ allora $x = x\varepsilon \in R^*(R + T)^*$. Dimostriamo invece $R^*(R + T)^* \subseteq (R + T)^*$. Se $x \in R^*(R + T)^*$ allora $x = ru$ con $r \in R^*$ e $u \in (R + T)^*$, allora $r \in R^*$ vuol dire che $r = w_1 \dots w_m$ con $w_i \in R$ e $u \in (R + T)^*$ vuol dire che $u = v_1 \dots v_n$ dove $v_i \in R$ oppure $v_i \in T$. Quindi si ha che $x = ru = (w_1 \dots w_m v_1 \dots v_n) \in (R + T)^*$. Si è quindi dimostrata l'uguaglianza.
(4) $\emptyset^* = \varepsilon$.
L'uguaglianza non è vera, dato che $\emptyset^*$ è l'insieme vuoto, mentre $\varepsilon = \{\varepsilon\}$ è l'insieme che contiene unicamente la striga vuota ε (quindi è non vuoto).

□

ESERCIZIO 1.10. *Si determini l'automa deterministico minimo per il linguaggio denotato dall'espressione regolare:* $(0^* + 1^* + (01)^*)$.

SOLUZIONE. Costruiamo l'ε-NFA che riconosce tale linguaggio:



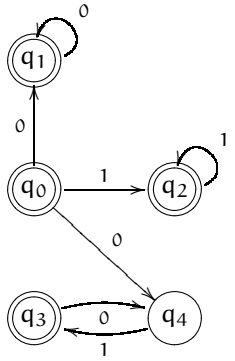
formalmente abbiamo $M = \langle\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1, \varepsilon\}, \delta, q_0, \{q_1, q_2, q_3\}\rangle$. La dimostrazione che questo automa effettivamente riconosce il linguaggio desiderato è banale e deriva dal teorema di equivalenza tra linguaggi denotati da espressioni regolari e linguaggi regolari.
Ricaviamo dall'ε-NFA, l'NFA equivalente. Si ricorda che dato un ε-NFA definito come $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ esiste sempre un NFA $M' = \langle Q, \Sigma, \delta', q_0, F'\rangle$ tale che $L(M) = L(M')$ dove

- $F' = \begin{cases} F \cup \{q_0\} & \text{Se } \varepsilon\text{-closure}(q_0) \cap F \neq \emptyset \\ F & \text{altrimenti} \end{cases}$
- $\delta'(q, a) = \hat{\delta}(q, a)$.

Definiamo la matrice di transizione di M' come

	0	1
q_0	$\{q_1, q_4\}$	$\{q_2\}$
q_1	$\{q_1\}$	$\emptyset$
q_2	$\emptyset$	$\{q_2\}$
q_3	$\{q_4\}$	$\emptyset$
q_4	$\emptyset$	$\{q_3\}$

Disegnamo M':

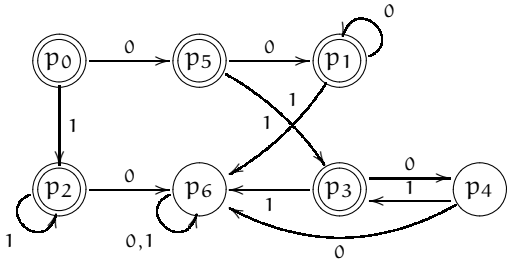


che formalmente si scrive $M' = \langle\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_0, \{q_0, q_1, q_2, q_3\}\rangle$. Dato l'NFA si ricava il DFA equivalente come visto nell'Esercizio 1.5. Definiamo la matrice di transizione per il DFA, ottenuta rinominando gli stati nel seguente

modo: $p_0 = \{q_0\}$, $p_1 = \{q_1\}$, $p_2 = \{q_2\}$, $p_3 = \{q_3\}$, $p_4 = \{q_4\}$, $p_5 = \{q_1, q_4\}$, $p_6 = \emptyset$. Gli stati mai raggiungibili da $\{q_0\}$ e quelli non considerati nella traduzione dell'automa sono omessi per semplicità denotazionale.

	0	1
p_0	p_5	p_2
p_1	p_1	p_6
p_2	p_6	p_2
p_3	p_4	p_6
p_4	p_6	p_3
p_5	p_1	p_3
p_6	p_6	p_6

Disegnamo il DFA:



definito come $M = \langle\{p_0, p_1, p_2, p_3, p_4, p_5, p_6\}, \{0, 1\}, \delta, p_0, \{p_0, p_1, p_2, p_3, p_5\}\rangle$. Sappiamo che questo automa riconosce effettivamente il linguaggio desiderato grazie ai teoremi che regolano le trasformazioni tra espressioni regolari e ε-NFA, tra ε-NFA e NFA, e tra NFA e DFA.
Applicando l'algoritmo di minimizzazione, come negli esercizi precedenti, si controlla se l'automa ottenuto è quello minimo. Per prima cosa partizioniamo gli stati dell'automa in due classi: $C_1 = \{p_0, p_1, p_2, p_3, p_5\}$ e $C_2 = \{p_4, p_6\}$ dove C_1 raccoglie gli stati finali e C_2 gli stati non finali. Partizioniamo quindi le classi, considerando gli archi etichettati con 0: ovvero otteniamo la partizione $\{p_0, p_5, p_1\}\{p_2\}\{p_3\}\{p_4, p_6\}$, ripetiamo il processo considerando gli archi etichettati con 1 e otteniamo la partizione $\{p_0, p_2, p_5\}\{p_3, p_1\}\{p_4\}\{p_6\}$. Intersechiamo ora le due partizioni e otteniamo $\{p_0, p_5\}\{p_2\}\{p_1\}\{p_3\}\{p_4\}\{p_6\}$. Iteriamo il procedimento considerando nuovamente lo 0: $\{p_0\}\{p_1\}\{p_2\}\{p_3\}\{p_4\}\{p_5\}\{p_6\}$, a questo punto ci fermiamo dato che ogni classe contiene un solo stato; questo indica che l'automa è minimo. □

2. Proprietà dei linguaggi regolari

In questa sezione vedremo come dimostrare alcune proprietà dei linguaggi regolari. In particolare utilizzeremo il Pumping Lemma per i linguaggi regolari al fine di verificare quando un linguaggio non è regolare. A questo scopo quando un linguaggio risulta essere intuitivamente non regolare, ovvero non contiene una certa regolarità nella definizione o necessita di una certa memoria per il suo riconoscimento allora per ogni costante k, presa una particolare stringa z tale che $|z| > k$, si dimostra che per ogni suddivisione di z del tipo $z = uvw$ tale che $|uv| \leq k$ e $|v| > 0$,

esiste sempre almeno un naturale $i \in \mathbb{N}$ tale che $uv^i w$ non sta nel linguaggio che stiamo considerando.

ESERCIZIO 2.1. *Si dimostri che il linguaggio*

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

non è regolare.

SOLUZIONE. Vediamo che, per riconoscere una stringa del linguaggio bisogna tenere traccia della quantità iniziale di 1 e 0 e quindi il linguaggio è intutivamente non regolare. Per dimostrare ciò utilizziamo il Pumping Lemma. Supponiamo per assurdo che il linguaggio L sia regolare, allora per il Pumping Lemma dei linguaggi regolari $\exists k \geq 0$ t.c. $\forall z \in L$ tale che $|z| \geq k$ esistono u, v, w tali che $z = uvw$, $|uv| \leq k, |v| \geq 1$ e $\forall i \geq 0\ uv^i w \in L$. Ipotizziamo dunque esista k e prendiamo $z \in L$ tale che $z = 0^k 1^k 0^{2k}$ allora $|z| > k$ infatti $|z| = 4k$. Dobbiamo suddividere la stringa

$$z = \overbrace{00\ldots0}^{k}\overbrace{11\ldots1}^{k}\overbrace{00\ldots0}^{2k}$$

Vi è un unico tipo di suddivisione di z in u, v, w tale che $|uv| \leq k$ il quale consiste in $u = 0^a$, $v = 0^b$ con $b \geq 1$, allora qualunque sia b risulta che (per $i = 0$) si ha $uw = 0^{k-b}1^k0^{2k}$ e poiché $2k \neq k + k - b$ allora $uw \notin L$, che è assurdo perché avevamo supposto che il linguaggio fosse regolare. Quindi avendo supposto il linguaggio regolare si è arrivati a un assurdo perché abbiamo trovato una stringa per cui non vale il Pumping Lemma; si puó allora concludere che il linguaggio non è regolare. □

ESERCIZIO 2.2. *Dimostrare formalmente che il linguaggio*

$$L = \left\{ 0^n 0^{2^n} \mid n \in \mathbb{N} \right\}$$

non è regolare.

SOLUZIONE. Intuitivamente il linguaggio non è regolare perché per accettare una stringa deve tenere memoria del numero di zeri che ha già letto. Per dimostrare che L non è regolare utilizziamo il Pumping Lemma per i linguaggi regolari. In particolare dimostriamo che per ogni $k \in \mathbb{N}$ esiste una $z \in L$ tale che $|z| > k$ e tale che per ogni suddivisione $z = uvw$, tale che $|uv| \leq k$ e $v > 0$, esiste un naturale $i \in \mathbb{N}$ tale per cui $uv^i w \notin L$.
Per ogni k prendiamo $z = 0^k 0^{2^k}$ e per ogni suddivisione $z = uvw$ dimostriamo che $uv^2w \notin L$. Sia $|v| = m > 0$ per le ipotesi del Pumping Lemma. È chiaro che $|uvw| = k+2^k$ per cui $|uv^2w| = k+2^k+m$. A questo punto perché la nuova stringa uv^2w stia in L la sua lunghezza deve essere $k' + 2^{k'}$ per qualche $k' \in \mathbb{N}$, ovvero $k + 2^k + m = k' + 2^{k'}$. Questo fatto implica che $k' > k$, essendo $m > 0$, e dunque esiste $m' \in \mathbb{N} \setminus \{0\}$ tale che $k' = k + m'$, percui otteniamo le seguenti equazioni:

$$\begin{aligned} &k + 2^k + m = k + m' + 2^{k+m'} \iff \\ &m = m' + 2^{k+m'} - 2^k = m' + 2^k(2^{m'} - 1) > 2^k > k \end{aligned}$$

essendo $m' > 0$. A questo punto è evidente che l'unico modo perché uv^2w stia in L è quello di prendere $|v| = m > k$, ma questo implica che $|uv| \geq |v| > k$, ovvero le uniche suddivisioni che permettono a uv^2w di restare in L non sono ammissibili dal Pumping Lemma. Questo significa che, nelle ipotesi del Pumping Lemma,

$uv^2w \notin L$, i.e. L non è regolare. □

Esercizio 2.3. *Dimostrare formalmente che il linguaggio*

$$L = \left\{ x \in \{0,1\}^* \;\middle|\; \begin{array}{l} \textit{il numero di occorrenze di } 0 \textit{ in } x \textit{ è} \\ \textit{uguale al numero di occorrenze di } 1 \end{array} \right\}$$

non è regolare.

Soluzione. Il linguaggio non è intuitivamente regolare perché per accettare una stringa deve tenere memoria distintamente del numero di 0 e del numero di 1 letti. Per dimostrare che non è regolare usiamo il Pumping Lemma. Come nell'esercizio precedente, per ogni costante k dobbiamo prendere una stringa del linguaggio la cui lunghezza è maggiore di k e poi dobbiamo ragionare sulle possibili suddivisioni della stringa. Sia allora, per ogni k, $z = 0^k1^k$. È evidente che questa è una stringa del linguaggio in quanto ha banalmente lo stesso numero di 0 e 1. A questo punto prendiamo $z = uvw$ tali che $|uv| \leq k$ e $|v| > 0$, è evidente che i queste condizioni v può solo contenere 0 e questo significa che per ogni $i > 1$ la stringa uv^iw contiene più 0 che 1 e dunque non sta nel linguaggio L. Per la verifica formale consideriamo $|v| = m$, allora

$$\forall i > 1 \,.\, uv^iw = 0^{k+m(i-1)}1^k$$

dove, essendo per ipotesi $m > 0$ e $i > 1$ si ha $k + m(i-1) > k$, ovvero $uv^iw \notin L$. Questo dimostra che L non è regolare. □

Esercizio 2.4. *Dimostrare formalmente che il linguaggio*

$$L = \left\{ 0^m1^n1^n0^m \;\middle|\; m, n \in \mathbb{N},\ m + n > 0 \right\}$$

non è regolare.

Soluzione. Il linguaggio non è intuitivamente regolare perché per accettare una stringa nella lettura dell'ultimo gruppo di 0 è necessario ricordare il numero di 0 letti all'inizio della stringa. Per dimostrare che L non è regolare utilizziamo il Pumping Lemma. Quindi per ogni $k \in \mathbb{N}$ prendiamo la stringa $z = 0^{k+1}1^k1^k0^{k+1}$ che chiaramente sta in L e tale che ha sempre almeno due 0. A questo punto prendiamo $z = uvw$ tali che $|uv| \leq k$ e $|v| > 0$, è evidente che in queste condizioni v può solo contenere 0 e questo significa che per ogni $i > 1$ la stringa uv^iw contiene più 0 nel primo gruppo che nell'ultimo gruppo di 0 e dunque non sta nel linguaggio L. Per la verifica formale consideriamo $|v| = m$, allora

$$\forall i > 1 \,.\, uv^iw = 0^{k+m(i-1)}1^k1^k0^k$$

dove, essendo per ipotesi $m > 0$ e $i > 1$ si ha $k + m(i-1) > k$, ovvero $uv^iw \notin L$. Questo dimostra che L non è regolare. □

Esercizio 2.5. *Dimostrare formalmente che il linguaggio*

$$L = \left\{ x \in \{0,1\}^* \;\middle|\; x \textit{ è palindrome} \right\}$$

non è regolare.

SOLUZIONE. Ricordiamo che una stringa è palindrome se è indifferente leggerla da destra a sinistra o da sinistra verso destra, ad esempio sono palindromi le stringhe $a^n b^m a^n$ qualunque sia il valore dei naturali m ed n. Il linguaggio non è intuitivamente regolare in quanto, un automa, per accettare una stringa deve tenere memoria di come è fatta la prima metà della stringa per riconoscere la seconda metà. Per dimostrare che il linguaggio non è regolare utilizziamo il Pumping Lemma. Dunque per ogni $k \in \mathbb{N}$ prendiamo $z = 0^k 1 0^k$ che charamente sta nel linguaggio. Consideriamo ora una generica suddivisione di z del tipo $z = uvw$ tale che $|uv| \leq k$ e $|v| > 0$. In tal caso è evidente che in queste condizioni v può solo contenere 0 e questo significa che per ogni $i > 1$ la stringa $uv^i w$ contiene più 0 nel primo gruppo che nell'ultimo gruppo di 0 e dunque non sta nel linguaggio L. Formalmente possiamo verificare che se $|v| = m$ allora

$$\forall i > 1 \,.\, uv^i w = 0^{k+m(i-1)} 1 0^k$$

dove, essendo per ipotesi $m > 0$ e $i > 1$ si ha $k + m(i-1) > k$, ovvero $uv^i w \notin L$. Questo dimostra che L non è regolare. □

ESERCIZIO 2.6. *Dimostrare formalmente che il linguaggio*

$$L = \left\{ \; 0^n 1^m \;\middle|\; m, n \in \mathbb{N},\ n < m \; \right\}$$

non è regolare.

SOLUZIONE. Il linguaggio non è intuitivamente regolare perché per accettare una stringa, quando conta il numero di 1 deve ricordarsi del numero di 0. Per dimostrare che il linguaggio non è regolare utilizziamo il Pumping Lemma. Dunque per ogni $k \in \mathbb{N}$ prendiamo $z = 0^k 1^{k+1}$ che chiaramente sta nel linguaggio. Consideriamo ora una generica suddivisione di z del tipo $z = uvw$ tale che $|uv| \leq k$ e $|v| > 0$. In tal caso è evidente che in queste condizioni v può solo contenere 0 e dunque posso sempre trovare un i tale che $uv^i w$ porta il numero di 0 ad essere maggiore del numero di 1. Formalmente possiamo verificare che se $|v| = m$ allora

$$uv^2 w = 0^{k+m} 1^{k+1}$$

dove, essendo per ipotesi $m > 0$ si ha $k + m \geq k + 1$, ovvero $uv^2 w \notin L$. Questo dimostra che L non è regolare. □

3. Linguaggi context free

Una grammatica è un insieme di regole che permettono di generare un linguaggio. Un ruolo fondamentale tra le grammatiche è costituito dalle grammatiche libere dal contesto mediante le quali vengono solitamente descritti i linguaggi di programmazione. Un linguaggio generato da un tale grammatica è detto libero dal contesto. Nei seguenti esercizi verrà richiesto di trasformare grammatiche CF in grammatiche in forma normale di Chomsky. Ricordiamo che una grammatica è in forma normale di Chomsky se ogni produzione è nella forma: $A \to a$ o $A \to AB$ oppure $S \to \varepsilon$ (solo una può essere di questo tipo). È possibile ottenere la forma normale di Chomsky nel seguente modo. Prima di tutto definiamo una transizione con una nuova variabile per ogni simbolo terminale, sostituendo le variabili ai simboli nelle transizioni esistenti. Dopo di che, dove ci sono più di due variabili associamo

al gruppo dalla seconda in poi una nuova variabile mediante una transizione e sostituiamo nelle transizioni esistenti. In altri esercizi verrà richiesto di dimostrare che un linguaggio è CF, per far ciò è sufficiente mostrare una grammatica CF che lo genera e dimostrare che ciò avviene. Ovvero bisogna dimostrare che ogni stringa appartiene al linguaggio se e solo se è generata dalla grammatica. In particolare si dimostra per induzione sulla lunghezza della stringa che ogni stringa nel linguaggio è generata dalla grammatica e si dimostra per induzione sulla lunghezza della derivazione che se esiste una derivazione nella grammatica allora la stringa generata sta nel linguaggio.
Infine facciamo vedere come è possibile trasformare una generica grammatica in una in forma normale di Greibach e come si definisce l'automa a pila corrispondente.

ESERCIZIO 3.1. *Si definisca la grammatica CF in forma normale di Chomsky che genera tutte le stringhe palindrome sull'alfabeto* $\{0,1\}$ *(ad esempio abbiamo che* $00100, 010010$ *sono palindrome mentre* $0101, 01001$ *non lo sono).*

SOLUZIONE. Consideriamo la seguente grammatica che genera stringhe palindrome sui simboli $0, 1$:

$$S \to \varepsilon|0|1|0S0|1S1$$

Dimostriamo che tale grammatica genera effettivamente tutte e solo le stringhe palindromi. In particolare dobbiamo dimostrare che se x è palindrome allora esiste una derivazione $S \Rightarrow_* x$ e che se esiste una derivazione $S \Rightarrow_* x$ allora x è palindrome. Il primo fatto viene dimostrato per induzione sulla lunghezza $|x|$ della stringa x, mentre il secondo fatto viene dimostrato per induzione sulla lunghezza k della derivazione.
1. x *palindrome allora* $S \Rightarrow_* x$*:* Per induzione su $|x|$.

   Base: $|x|=0$, ovvero $x = \varepsilon$, ed è evidente che $S \to \varepsilon$.
   Induzione: Supponiamo che per ogni y palindrome tale che $|y| < k$ esiste una derivazione $S \Rightarrow_* y$. Consideriamo x palindrome tale che $|x| = k$. Se x è palindrome allora $x = uv$ oppure $x = uav$ con $a \in \{0,1\}$ e $u, v \in \{0,1\}^*$ tali che v è uguale a u rovesciata. Se $x = uv$ allora $x = u'aav'$, con $a \in \{0,1\}$, $u = u'a$ e $v = av'$. Consideriamo ora $y = u'v'$, è facile verificare che y è ancora palindrome ed è tale che $|y| < k$, allora per ipotesi induttiva abbiamo che $S \Rightarrow_* y = u'v'$. Ma allora esiste la derivazione $S \Rightarrow_* u'Sv'$ per cui possiamo costruire la seguente derivazione:

   $$S \Rightarrow_* u'Sv' \to u'aSav' \to u'aav' = x$$

   Quindi abbiamo trovato una derivazione per x. Analogo il caso in cui $x = uav$.

2. $S \Rightarrow_k x$ *allora* x *palindrome:* Per induzione su k.

   Base: $k = 1$ allora $S \to \varepsilon$, $S \to 0$ e $S \to 1$, dove $\varepsilon, 0, 1$ sono tutte palindromi.
   Induzione: Supponiamo che se $S \Rightarrow_k x$ allora x è palindrome, vediamo cosa succede se prendiamo una derivazione lunga $k+1$. Se x è palindrome allora $x = uv$ oppure $x = uav$ con $a \in \{0,1\}$ e $u, v \in \{0,1\}^*$, tali che v è uguale a u rovesciata. In ogni caso avere una derivazione per x significa avere la derivazione $S \Rightarrow_{k-1} uSv$, allora possiamo vedere cosa può succedere ad

una generica derivazione lunga $k+1$:

$$S \Rightarrow_{k-1} uSv \rightarrow uaSav \rightarrow uaav \text{ (oppure } uabav)$$

dove $a, b \in \{0, 1\}$. Allora è evidente che in ogni caso otteniamo stringhe palindromi.

Ricordiamo che una grammatica è in forma normale di Chomsky se ogni produzione è nella forma: $A \rightarrow a$ o $A \rightarrow AB$ oppure $S \rightarrow \varepsilon$ (solo una può essere di questo tipo). Ecco come è possibile ottenere la forma normale di Chomsky. Prima di tutto definiamo una transizione con una nuova variabile per ogni simbolo terminale, sostituendo le variabili ai simboli nelle transizioni esistenti:

$$S \rightarrow \varepsilon|0|1|ASA|BSB$$
$$A \rightarrow 0$$
$$B \rightarrow 1$$

Dopo di che, dove ci sono più di due variabili associamo al gruppo dalla seconda in poi una nuova variabile mediante una transizione e sostituiamo nelle transizioni esistenti.

$$S \rightarrow \varepsilon|0|1|AD|BE$$
$$A \rightarrow 0$$
$$B \rightarrow 1$$
$$D \rightarrow SA$$
$$E \rightarrow SB$$

ottenendo in tal modo la grammatica in forma normale di Chomsky. □

Esercizio 3.2. *Dimostrare formalmente che il linguaggio*

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

è context free.

Soluzione. Costruiamo la grammatica G che genera L:

$$S \rightarrow \varepsilon|A|B$$
$$A \rightarrow 0A0|B|00$$
$$B \rightarrow 1B0|10$$

Dimostriamo ora che tale grammatica genera esattamente L. Per far ciò dobbiamo dimostrare che quando una stringa x sta nel linguaggio, $x \in L$, allora esiste una derivazione nella grammatica che genera x, $S \Rightarrow_* x$, mentre quando esiste una derivazione nella grammatica per x, $S \Rightarrow_* x$, allora si ha che $x \in L$. Il primo fatto viene sempre dimostrato per induzione sulla lunghezza di x, mentre il secondo viene dimostrato per induzione sulla lunghezza della derivazione.

1. $x \in L$ *allora* $S \Rightarrow_* x$*:* Per induzione su $|x|$.

Base: Se $|x| = 0$, allora $x = \varepsilon$ e per la definizione della grammatica abbiamo $S \rightarrow \varepsilon$;

Induzione: Supponiamo che per ogni $y \in L$ tale che $|y| < k$ allora esiste una derivazione in G per y, ovvero $S \Rightarrow_* y$. Prendiamo ora $x \in L$ tale che $|x| = k$, allora esisteranno $n, m \in \mathbb{N}$ tali che $x = 0^n 1^m 0^{m+n}$. Supponiamo $m \geq 2$ e consideriamo la stringa $y = 0^n 1^{m-1} 0^{m-1+n}$ allora $|y| < k$ e quindi esiste una derivazione $S \Rightarrow_* 0^n 1^{m-1} 0^{m-1+n} = 0^n 1^{m-2} 100^{m-2+n}$. Per come è definita la grammatica, se esiste una derivazione come quella

appena descritta, allora ne esiste ua per $S \Rightarrow_* 0^n1^{m-2}B0^{m-2+n}$ quindi possiamo costruire la derivazione

$$\begin{aligned} S \Rightarrow_* 0^n1^{m-2}B0^{m-2+n} &\to 0^n1^{m-2}1B00^{m-2+n} \\ &\to 0^n1^{m-1}100^{m-1+n} = 0^n1^m0^{m+n} = x \end{aligned}$$

Quindi abbiamo trovato una derivazione per x. In modo del tutto analogo possiamo trovare derivazioni simili quando $m < 2$. Vediamo il caso in cui $m = 1$. Supponiamo allora che $x = 0^n100^n$, possiamo costruire la seguente derivazione considerando che per ipotesi induttiva esiste la derivazione per 0^n0^n

$$S \Rightarrow_* 0^nA0^n \to 0^nB0^n \to 0^n1B00^n \to 0^n100^n = x$$

Infine quando $m = 0$ allora ci basta mostrare che la grammatica permette sempre di generare una stringa di 0 di lunghezza pari, e questo è semplicemente dimostrabile per induzione sulla lunghezza della stringa a partire dalla definizione della grammatica.

2. $S \Rightarrow_k x$ *allora* $x \in L$*:* Per induzione su k.

Base: Se $n = 1$, allora l'unica derivazione di un solo passo è $S \to \varepsilon$ e per definizione di L abbiamo che $\varepsilon \in L$;

Induzione: Supponiamo che se una derivazione è di lunghezza k allora $S \Rightarrow_k y$ implichi $y \in L$. Consideriamo allora tutte le possibili derivazioni di lunghezza $k + 1$. Se $y \in L$ allora significa che $y = 0^{2n}$, oppure $y = 1^n0^n$ oppure $y = 0^n1^m0^{n+m}$, per qualche $n, m \in \mathbb{N}$. La dimostrazione andrebbe fatta per tutti e tre i casi, noi ne considereremo uno solo, essendo gli altri casi analoghi. Sia allora $y = 0^n1^m0^{n+m}$ tale che $S \Rightarrow_k 0^n1^m0^{n+m} = 0^n1^{m-1}100^{n+m-1}$. Se una tale derivazione esiste allora, per la definizione della grammatica si ha che esiste anche la derivazione $S \Rightarrow_{k-1} 0^n1^{m-1}B0^{n+m-1}$ ma allora possiamo costruire la solo la seguente derivazione di lunghezza $k + 1$:

$$\begin{aligned} S \Rightarrow_{k-1} 0^n1^{m-1}B0^{n+m-1} &\to 0^n1^{m-1}1B00^{n+m-1} \\ &\to 0^n1^m100^{n+m} = 0^n1^{m+1}0^{n+m+1} \in L \end{aligned}$$

Analoghe dimostrazioni possono essere fatte per gli altri casi.

□

ESERCIZIO 3.3. *Sia* $|x|_a$ *definito come il numero di occorrenze del simbolo* a *nella stringa* x*. Dimostrare formalmente che il linguaggio*

$$L = \left\{ \; x \in \{0,1\}^* \;\middle|\; |x|_0 = |x|_1 \; \right\}$$

è context free.

SOLUZIONE. Per dimostrare che un linguaggio è context free dobbiamo definire una grammatica e dimostrare che il linguaggio è generato da questa grammatica. Vediamo dunque la grammatica che genera il linguaggio. Questa deve essere tale che ogniqualvolta aggiunge uno 0 (rispettivamente un 1) allora deve aggiungere anche un 1 (respettivamente uno 0).

$$S \to \varepsilon|S0S1S|S1S0S$$

Dimostriamo ora che questa grammatica effettivamente genera tutte e sole le stringhe del linguaggio L[1]. In particolare dobbiamo dimostrare che se $x \in L$ allora esiste una derivazione $S \Rightarrow_* x$ e che se esiste una derivazione $S \Rightarrow_* x$ allora $x \in L$. Il primo fatto viene dimostrato per induzione sulla lunghezza $|x|$ della stringa x, mentre il secondo fatto viene dimostrato per induzione sulla lunghezza k della derivazione.

1. $x \in L$ *allora* $S \Rightarrow_* x$*:* Per induzione su $|x|$.

   Base: $|x| = 0$, ovvero $x = \varepsilon$, allora notiamo che $S \to \varepsilon$.

   Induzione: Supponiamo che per ogni $y \in L$ tale che $|y| < k$ allora esiste una derivazione $S \Rightarrow_* y$. Prendiamo $x \in L$ tale che $|x| = k$, allora per ipotesi abbiamo che $|x|_0 = |x|_1$, in tal caso inoltre è immediato notare che in x deve esserci almeno un'occorrenza della stringa 01 oppure della stringa 10. Supponiamo ci sia 01, l'altro caso è analogo. Dunque abbiamo che $x = u01v$, consideriamo la stringa $y = uv$, allora $|y|_0 = |x|_0 - 1 = |x|_1 - 1 = |y|_1$, ovvero $y \in L$ e $|y| = h < k$, quindi per ipotesi induttiva abbiamo che $S \Rightarrow_* y$. È banale dimostrare che, per come è definita la grammatica, se $S \Rightarrow_* y$, allora possiamo sempre riordinare le produzioni della derivazione in modo che, se $y = y_0y_1y_2 \cdots y_{n-1}$, allora

   $$S \Rightarrow_* y_0Sy_1Sy_2S \dots Sy_{h-1}S \Rightarrow_h y$$

   ovvero possiamo portare tutte le produzioni del tipo $S \to \varepsilon$ alla fine della derivazione, e prima di applicare queste abbiamo esattamente la stringa finale in cui ogni simbolo terminale si trova tra due simboli non terminali S. Questo implica che se $S \Rightarrow_* y = uv$, allora esiste la derivazione

   $$S \Rightarrow Sy_0Sy_1Sy_2S \dots Sy_{h-1}S \Rightarrow_{h-1} uSv \to uv$$

   e quindi possiamo costruire la derivazione per x

   $$S \Rightarrow Sy_0Sy_1Sy_2S \dots Sy_{h-1}S \Rightarrow_{h-1} uSv \to uS0S1Sv \Rightarrow_3 u01v = x$$

2. $S \Rightarrow_k x$ *allora* $x \in L$*:* Per induzione su k.

   Base: $k = 1$, allora $S \to \varepsilon$, con $\varepsilon \in L$.

   Induzione: Supponiamo che se $S \Rightarrow_k x$ allora $x \in L$, vediamo cosa succede per le derivazioni lunghe $k + 1$. Se $S \Rightarrow_k x$ allora, per come è definita la grammatica, è evidente che per ogni coppia $u, v \in \{0, 1\}^*$ tale che $x = uv$, esista una derivazione $S \Rightarrow_{k-1} uSv \to x$. Ma allora possiamo costruire solo la seguente derivazione di lunghezza $k + 1$:

   $$S \Rightarrow_{k-1} uSv \to uSaSbSv \Rightarrow_3 uabv$$

   dove $a, b \in \{0, 1\}$ con $a \neq b$ e gli ultimi tre passi sono tutti applicazioni della produzione $S \to \varepsilon$. Allora è evidente che $|uabv|_0 = |uv|_0 + 1 = |uv|_1 + 1 = |uabv|_1$, ovvero anche la nuova stringa sta in L.

   □

[1] Si noti che la grammatica $S \to \varepsilon|0S1|1S0$ non general la stringa 0110 che invece sta nel linguaggio L, quindi non può essere la grammatica che genera L

ESERCIZIO 3.4. *Dimostrare formalmente che il linguaggio*

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

è context free.

SOLUZIONE. Per dimostrare che un linguaggio è context free dobbiamo definire una grammatica e dimostrare che il linguaggio è generato da questa grammatica. Vediamo dunque la grammatica che genera il linguaggio. Questa deve essere tale che ogniqualvolta aggiunge uno 0 allora deve aggiungerne un altro permettendo di espandere solo tra i due 0. Una volta finito di aggiungere 0 deve poter aggiungere solo 1 e sempre a coppie. Almeno due 0 o due 1 ci devono essere essendo $m+n > 0$.

$$S \to 00|0S0|A$$
$$A \to 11|1A1$$

Dimostriamo ora che questa grammatica effettivamente genera tutte e sole le stringhe del linguaggio L. In particolare dobbiamo dimostrare che se $x \in L$ allora esiste una derivazione $S \Rightarrow_* x$ e che se esiste una derivazione $S \Rightarrow_* x$ allora $x \in L$. Il primo fatto viene dimostrato per induzione sulla lunghezza $|x|$ della stringa x, mentre il secondo fatto viene dimostrato per induzione sulla lunghezza k della derivazione.
1. $x \in L$ *allora* $S \Rightarrow_* x$*:* Per induzione su $|x|$.

Base: $|x| = 2$, ovvero $x = 00$ oppure $x = 11$. Allora abbiamo le seguenti derivazioni:

$$S \to 00$$
$$S \to A \to 11$$

Induzione: Supponiamo che per ogni $y \in L$ tale che $|y| < k$ si abbia $S \Rightarrow_* y$. Consideriamo $x \in L$ tale che $|x| = k$, allora si avrà che $x = 0^m 1^n 1^n 0^m = 0^m 1^{n-1} 11 1^{n-1} 0^m$. Consideriamo $y = 0^m 1^{n-1} 1^{n-1} 0^m$, è evidente che $y \in L$ e che $|y| < k$, quindi per ipotesi induttiva esiste la derivazione $S \Rightarrow_* y$. Ovvero esiste la derivazione $S \Rightarrow_* 0^m 1^{n-2} A 1^{n-2} 0^m \to 0^m 1^{n-1} 1^{n-1} 0^m = y$. A partire da questa derivazione possiamo allora costruire la seguente derivazione:

$$S \Rightarrow_* 0^m 1^{n-2} A 1^{n-2} 0^m \to 0^m 1^{n-1} A 1^{n-1} 0^m \to 0^m 1^n 1^n 0^m = x$$

Analogo il caso in cui x sia composta solo da 0 o solo da 1.

2. $S \Rightarrow_k x$ *allora* $x \in L$*:* Per induzione su k.

Base: $k = 1$, allora $S \to 00$ e 00 appartiene a L.
Induzione: Supponiamo che se $S \Rightarrow_k x$ allora $x \in L$ e vediamo cosa succede ad una derivazione lunga $k + 1$. Se $x \in L$ allora $x = 0^m 1^n 1^n 0^m$ e quindi la derivazione è la seguente $S \Rightarrow_{k-1} 0^m 1^{n-1} A 1^{n-1} 0^m \to 0^m 1^n 1^n 0^m$. A questo punto, a partire da tale derivazione, possiamo costruire la possibile derivazione lunga $k + 1$.

$$S \Rightarrow_{k-1} 0^m 1^{n-1} A 1^{n-1} 0^m \to 0^m 1^n A 1^n 0^m \to 0^m 1^{n+1} 1^{n+1} 0^m \in L$$

Analoga la dimostrazione nel caso in cui $x = 0^{2m}$ oppure $x = 1^{2n}$.

□

ESERCIZIO 3.5. *Dimostrare formalmente che il linguaggio*

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

è context free.

SOLUZIONE. Per dimostrare che un linguaggio è context free dobbiamo definire una grammatica e dimostrare che il linguaggio è generato da questa grammatica. Vediamo dunque la grammatica che genera il linguaggio. Questa deve essere tale che ogniqualvolta aggiunge uno 0 allora deve aggiungere un 1 espandendo sempre tra 0 e 1, dopo deve aggiungere almeno un altro 1.

$$\begin{aligned} S &\to A|B \\ A &\to 0A1|B \\ B &\to 1|1B \end{aligned}$$

Dimostriamo ora che questa grammatica effettivamente genera tutte e sole le stringhe del linguaggio L. In particolare dobbiamo dimostrare che se $x \in L$ allora esiste una derivazione $S \Rightarrow_* x$ e che se esiste una derivazione $S \Rightarrow_* x$ allora $x \in L$. Il primo fatto viene dimostrato per induzione sulla lunghezza $|x|$ della stringa x, mentre il secondo fatto viene dimostrato per induzione sulla lunghezza k della derivazione.

1. $x \in L$ *allora* $S \Rightarrow_* x$*:* Per induzione su $|x|$.

   Base: $|x| = 1$, ovvero $x = 1$, ed esiste la derivazione $S \to B \to 1$.

   Induzione: Supponiamo che per ogni stringa $y \in L$ tale che $|y| < k$ si abbia che $S \Rightarrow_* y$. Consideriamo $x \in L$ tale che $|x| = k$, allora $x = 0^n 1^m$ con $n < m$. Consideriamo $y = 0^{n-1}1^{m-1}$, allora è evidente che se $n < m$ allora $n - 1 < m - 1$, quindi $y \in L$ e inoltre $|y| < k$. Per ipotesi induttiva abbiamo che $S \Rightarrow_* y$. In particolre, per come è definita la grammatica, la derivazione sarà del tipo $S \Rightarrow_* 0^{n-1}A1^{n-1} \Rightarrow_{m-n} 0^{n-1}1^{m-1} = y$. Ma allora a partire da questa derivazione possiamo definire la seguente per x.

   $$S\ \Rightarrow_*\ 0^{n-1}A1^{n-1}\ \to\ 0^n A 1^n\ \Rightarrow_{m-n} 0^n 1^m = x$$

   Analoga la dimostrazione nel caso in cui $x = 1^m$.

2. $S \Rightarrow_k x$ *allora* $x \in L$*:* Per induzione su k.

   Base: $k = 2$ allora $S \to B \to 1$ e $1 \in L$.

   Induzione: Supponiamo che se $S \Rightarrow_k x$ allora $x \in L$, vediamo cosa succede per una derivazione lunga $k + 1$. Se $x \in L$ allora $x = 0^n 1^m$, per cui la derivazione è del tipo $S \Rightarrow_{k-1} 0^n 1^{m-1} B \to 0^n 1^m$. A partire da questa possiamo allora costruire la possibile derivazione lunga $k + 1$:

   $$S\ \Rightarrow_{k-1}\ 0^n 1^{m-1} B\ \to\ 0^n 1^m B\ \to\ 0^n 1^{m+1} \in L$$

   Analoga la dimostrazione nel caso in cui $x = 1^m$.

□

ESERCIZIO 3.6. *Si trasformi la seguente grammatica in una in forma normale di Greibach e si definisca l'automa a pila corrispondente.*

$$\begin{aligned} S &\to AA|0 \\ A &\to SS|1 \end{aligned}$$

SOLUZIONE. Per prima cosa diamo un ordine alle variabili ridenominando le variabili nel seguente modo:

$$A_1 \to A_2A_2|0$$
$$A_2 \to A_1A_1|1$$

Passo 1: Poiché le parti a destra delle produzioni da A_1 iniziano con terminali o con variabili di indice maggiore, consideriamo inizialmente solo la produzione $A_2 \to A_1A_1$. In questa sostituiamo al posto della prima occorrenza di A_1 le possibili produzioni per A_1, in tal modo otteniamo la seguente grammatica equivalente:

$$\begin{array}{lcl} A_1 & \to & A_2A_2|0 \\ A_2 & \to & A_2\underbrace{A_2A_1}_{\alpha_1}|\underbrace{0A_1}_{\beta_1}|\underbrace{1}_{\beta_2} \end{array}$$

Adesso applichiamo il Lemma dell'eliminazione della ricorsione sinistra (vedi dispense) alle produzioni dalla variabile A_2 definendo la nuova variabile B e riscrivendo le produzioni nel seguente modo:

$$\begin{array}{lcl} A_1 & \to & A_2A_2|0 \\ A_2 & \to & 0A_1|1|0A_1B|1B \\ B & \to & A_2A_1|A_2A_1B \end{array}$$

Passo 2: A questo punto tutte le derivazioni da A_2, nella parte destra, iniziano con simboli terminali. Queste produzioni sono utilizzate per rimpiazzare A_2 nelle altre produzioni:

$$\begin{array}{lcl} A_1 & \to & 0A_1A_2|1A_2|0A_1BA_2|1BA_2|0 \\ A_2 & \to & 0A_1|1|0A_1B|1B \\ B & \to & A_2A_1|A_2A_1B \end{array}$$

Passo 3: Infine, quando tutte le variabili della grammatica originale sono nella forma voluta, sistemiamo anche le derivazioni dalla nuova variabile B,

$$\begin{array}{lcl} A_1 & \to & 0A_1A_2|1A_2|0A_1BA_2|1BA_2|0 \\ A_2 & \to & 0A_1|1|0A_1B|1B \\ B & \to & 0A_1A_1|1A_1|0A_1BA_1|1BA_1|0A_1A_1B|1A_1B|0A_1BA_1B|1BA_1B \end{array}$$

ottenendo la grammatica in forma normale di Greibach equivalente alla grammatica data[2].
Costruiamo ora l'automa a pila che riconosce il linguaggio generato dalla grammatica data riscritta con i nomi delle variabili originali:

$$\begin{array}{lcl} S & \to & 0|0SA|0SBA|1A|1BA \\ A & \to & 0S|0SB|1|1B \\ B & \to & 0SS|0SBS|0SSB|0SBSB|1S|1SB|1BS|1BSB \end{array}$$

[2]Notiamo che nel nostro caso, vista la semplicità della grammatica, i passi 2 e 3 potevano essere fatti contemporaneamente, in generale questo non è possibile perché le parti destre delle produzioni da B potrebbero iniziare con diversi simboli non terminali e non tutte con lo stesso come nel nostro caso.

Definiamo quindi l'automa $M = \langle\{q\},\{0,1\},\{S,A,B\},q,S,\emptyset,f\rangle$ dove la funzione di transizione f è definita dalla seguente tabella:

q	ε	0	1
S		q,ε q,SA q,SBA	q,A q,BA
A		q,S q,SB	q,ε q,B
B		q,SS q,SBS q,SSB $q,SBSB$	q,S q,SB q,BS q,BSB

□

4. Proprietà dei linguaggi context free

In questa sezione vedremo come dimostrare alcune proprietà dei linguaggi context free. In particolare utilizzeremo il Pumping Lemma per i linguaggi centext free al fine di verificare quando un linguaggio non è context free. A questo scopo quando un linguaggio risulta essere intuitivamente non context free, ovvero necessita di di memorizzare più informazioni per il riconoscimento delle sue stringhe allora per ogni costante k, presa una particolare stringa z tale che $|z| > k$, si dimostra che per ogni suddivisione di z del tipo $z = uvwxy$ tale che $|vwx| \leq k$ e $|vx| > 0$, esiste sempre almeno un naturale $i \in \mathbb{N}$ tale per cui uv^iwx^iy non sta nel linguaggio che stiamo considerando.

Esercizio 4.1. *Si verifichi formalmente se i linguaggi*

$$L_1 = \left\{ 0^{2n}10^n \;\middle|\; n \in \mathbb{N} \right\}$$
$$L_2 = \left\{ 0^{2n}10^{n^2} \;\middle|\; n \in \mathbb{N} \right\}$$

sono context free (nel caso lo siano trovare la grammatica, dimostrare la sua correttezza e evidenziare dove fallisce il Pumping Lemma).

Soluzione. Consideriamo L_1, è facile notare che il linguaggio è context free, in particolare esso è generato dalla grammatica $S \to 1 \mid 00S0$. Dimostriamo tale fatto formalmente. Supponiamo $x \in L_1$ allora dobbiamo dimostrare, per induzione su $|x|$ che esiste una derivazione $S \Rightarrow_* x$. Se $|x| = 1$, ovvero $x = 1$, allora è evidente che $S \to 1$. Supponiamo che per ogni stringa $y \in L_1$ tale che $|y| < k$ si abbia $S \Rightarrow_* y$, consideriamo $x \in L_1$ tale che $|x| = k$. Se $x \in L_1$ allora $x = 0^{2n}10^n$. Prendiamo ora $y = 0^{2n-2}10^{n-1}$, è evidente che $y \in L_1$ e che $|y| < k$, allora per ipotesi induttiva abbiamo che $S \Rightarrow_* y$. Quindi abbaimo la derivazione

$S \Rightarrow_* 0^{2n-2}S0^{n-1} \to 0^{2n-2}10^{n-1}$. Allora a partire da tale derivazione possiamo costruire quella per x:

$$S \Rightarrow_* 0^{2n-2}S0^{n-1} \to 0^{2n-2}00S00^{n-1} = 0^{2n}S0^n \to 0^{2n}10^n = x$$

D'altra parte dimostriamo ora che se $S \Rightarrow_k x$ allora $x \in L_1$ per induzione su k. Se $k = 1$ allora $S \to 1$ e $1 \in L_1$. Supponiamo ora che se $S \Rightarrow_k x$ allora $x \in L_1$, vediamo cosa succede alle derivazioni lunghe $k+1$. Se $x \in L_1$ allora $x = 0^{2n}10^n$, quindi la derivazione è del tipo $S \Rightarrow_{k-1} 0^{2n}S0^n \to 0^{2n}10^n$. A partire da questa possiamo descrivere la derivazione lunga $k+1$:

$$S \Rightarrow_{k-1} 0^{2n}S0^n \to 0^{2n}00S00^n = 0^{2n+2}S0^{n+1} \to 0^{2n+2}10^{n+1} \in L_1$$

Quindi abbiamo dimostrato che il linguaggio è context free. Vediamo ora dove sarebbe fallito il Pumping Lemma se avessimo provato ad utilizzarlo per dimostrare che il linguaggio non era context free. Per ogni $k \in \mathbb{N}$ prendiamo la stringa $z = 0^{2k}10^k$ e consideriamo ogni possibile suddivisione $z = uvwxy$. Se v o x contiene 1 allora banalmente per ogni $i > 1$ si ha che $uv^iwx^iy \notin L_1$. Se v e x sono entrambe nel primo o nel secondo gruppo di 0 allora ancora banalmente abbiamo che per ogni $i > 1$ si ha $uv^iwx^iy \notin L_1$ perché il primo gruppo non conterrebbe più esattamente il doppio di 0 rispetto al secondo. Consideriamo infine v nel primo gruppo e x nel secondo, allora $uv^iwx^iy = 0^{k'}10^{k''}$ dove $k' = 2k + |v|(i-1)$ e $k'' = k + |x|(i-1)$. Allora $uv^iwx^iy \in L_1$ se e solo se $k' = 2k''$ e questo vale per ogni suddivisione tale che $|v| = 2|x|$. Questo significa che per ogni $z \in L_1$ trovo una suddivisione che rispetta le ipotesi del Pumping Lemma e tale per cui $uv^iwx^iy \in L_1$, ovvero il teorema non dice nulla sul linguaggio L_1.

Consideriamo ora L_2, in particolare dimostriamo che tale linguaggio non è context free. Per provare ciò utilizziamo il Pumping Lemma per i linguaggi context free, ovvero per ogni costante k troviamo una stringa z del linguaggio tale che $|z| \geq k$ e, per ogni suddivisione $z = uvwxy$ tale che $|vwx| < k$ e $|vx| > 0$, dimostriamo che esiste almeno un $i \in \mathbb{N}$ tale per cui uv^iwx^iy non appartiene al linguaggio. Per ogni $k \in \mathbb{N}$ prendiamo la stringa $z = 0^{2k}10^{k^2}$ e consideriamo ogni possibile suddivisione $z = uvwxy$. Se v o x contiene 1 allora banalmente per ogni $i > 1$ si ha che $uv^iwx^iy \notin L_2$ perché aumenta il numero di 1. Se v e x sono entrambe nel primo o nel secondo gruppo di 0 allora ancora banalmente abbiamo che per ogni $i > 1$ si ha $uv^iwx^iy \notin L_2$ perché il primo gruppo non conterrebbe più un numero di 0 che sta nella stessa relazione rispetto al numero di 0 del secondo. Consideriamo infine v nel primo gruppo e x nel secondo, allora $uv^iwx^iy = 0^{2h'}10^{h^2}$ dove $2h' = 2k + |v|(i-1)$ e $h^2 = k^2 + |x|(i-1)$. Vediamo con quali ipotesi sulla suddivisione otteniamo $uv^iwx^iy \in L_2$. Quest'ultimo fatto avviene se e solo se $h' = h$, ovvero se il seguente sistema ha soluzione:

$$\begin{cases} 2h &= 2k + |v|(i-1) \\ h^2 &= k^2 + |x|(i-1) \end{cases}$$

che può essere risolto come

$$\begin{cases} 4h^2 &= 4k^2 + |v|^2(i-1)^2 + 4k|v|(i-1) \\ 4h^2 &= 4k^2 + 4|x|(i-1) \end{cases}$$

da cui l'equazione

$4k^2 + |v|^2(i-1)^2 + 4k|v|(i-1) = 4k^2 + 4|x|(i-1)$ semplificando
$4|x| = 4k|v| + |v|^2(i-1)$

Ora per ogni $i > 1$ si ha che $|v| > 0$, altrimenti $|vx| = 0$, e quindi $4|x| > 4k|v|$ ovvero $|x| > k|v|$, ma questo implica che $|vwx| \geq |vx| > k$. Sappiamo però che una tale suddivisione non è accettata nel Pumping Lemma, quindi per ogni suddivisione che rispetta le condizioni del torema si ha che $uv^i wx^i y \notin L_2$, ovvero il linguaggio non è context free. □

Esercizio 4.2. *Si dimostri formalmente che i linguaggi*

$$L_1 = \left\{ 0^k 0^{2k^2} \mid k \in \mathbb{N} \right\}$$
$$L_2 = \left\{ 0^k 0^{2^k} \mid k \in \mathbb{N} \right\}$$

non sono context free.

Soluzione. Consideriamo L_1 e dimostriamo che non è context free. Per provare ciò utilizziamo il Pumping Lemma per i linguaggi context free, ovvero per ogni costante k troviamo una stringa z del linguaggio tale che $|z| \geq k$ e, per ogni suddivisione $z = uvwxy$ tale che $|vwx| < k$ e $|vx| > 0$, dimostriamo che esiste un $i \in \mathbb{N}$ tale per cui $uv^i wx^i y$ non appartiene al linguaggio. Per ogni $k \in \mathbb{N}$ prendiamo la stringa $z = 0^k 0^{2k^2}$ e consideriamo ogni possibile suddivisione $z = uvwxy$. Dimostriamo che per ogni suddivisione abbiamo che $uv^2 wx^2 y \notin L_1$, per fare ciò consideriamo la lunghezza di questa nuova stringa $|uv^2 wx^2 y| = |z| + |vx| = k + 2k^2 + |vx|$, perché tale stringa stia il L_1 è evidente che deve esistere un $h \in \mathbb{N}$ tale che $k + 2k^2 + |vx| = h + 2h^2$. Naturalmente da tale relazione si ha $h > k$, ovvero esiste $n > 0$ tale che $h = k + n$, quindi la relazione si riscrive nella seguente equazione:

$$k + 2k^2 + |vx| = k + n + 2(k + n)^2 = k + n + 2k^2 + 2n^2 + 4kn$$

da cui risolvendo si ottiene la relazione $|vx| = 4kn + 2n^2 + n > k$ essendo $n > 0$, ma allora $|vwx| \geq |vx| > k$. Questo significa che le uniche suddivisioni che permettono a $uv^2 wx^2 y$ di restare nel linguaggio non sono ammesse dal Pumping Lemma e quindi L_1 non è context free.

Consideriamo ora L_2 e dimostriamo che non è context free. Per provare ciò utilizziamo il Pumping Lemma per i linguaggi context free, ovvero per ogni costante k troviamo una stringa z del linguaggio tale che $|z| \geq k$ e, per ogni suddivisione $z = uvwxy$ tale che $|vwx| < k$ e $|vx| > 0$, dimostriamo che esiste un $i \in \mathbb{N}$ tale per cui $uv^i wx^i y$ non appartiene al linguaggio. Per ogni $k \in \mathbb{N}$ prendiamo la stringa $z = 0^k 0^{2^k}$ e consideriamo ogni possibile suddivisione $z = uvwxy$. Dimostriamo che per ogni suddivisione abbiamo che $uv^2 wx^2 y \notin L_2$, per fare ciò consideriamo la lunghezza di questa nuova stringa $|uv^2 wx^2 y| = |z| + |vx| = k + 2^k + |vx|$, perché tale stringa stia il L_2 è evidente che deve esistere un $h \in \mathbb{N}$ tale che $k + 2k^2 + |vx| = h + 2^h$. Naturalmente da tale relazione si ha $h > k$, ovvero esiste $n > 0$ tale che $h = k + n$, quindi la relazione si riscrive nella seguente equazione:

$$k + 2^k + |vx| = k + n + 2^{(k+n)} = k + n + 2^k 2^n$$

da cui risolvendo si ottiene la relazione $|vx| = 2^k(2^n - 1) + n > 2^k > k$ essendo $n > 0$, ma allora $|vwx| \geq |vx| > k$. Questo significa che le uniche suddivisioni che permettono a $uv^2 wx^2 y$ di restare nel linguaggio non sono ammesse dal Pumping Lemma e quindi L_2 non è context free. □

ESERCIZIO 4.3. *Si dimostri formalmente che il linguaggio*

$$L = \{\ 0^n \mid n \text{ è primo}\ \}$$

non è context free.

SOLUZIONE. Dimostriamo che il linguaggio non è context free. Per provare ciò utilizziamo il Pumping Lemma per i linguaggi context free, ovvero per ogni costante k troviamo una stringa z del linguaggio tale che $|z| \geq k$ e, per ogni suddivisione $z = uvwxy$ tale che $|vwx| < k$ e $|vx| > 0$, dimostriamo che esiste un $i \in \mathbb{N}$ tale per cui uv^iwx^iy non appartiene al linguaggio. Per ogni $k \in \mathbb{N}$ prendiamo la stringa $z = 0^n$ dove n è il più piccolo numero primo tale che $n \geq k$, e consideriamo ogni possibile suddivisione $z = uvwxy$. Dimostriamo che per ogni suddivisione abbiamo che per almeno un $i > 0$ si ha $uv^iwx^iy \notin L$, per fare ciò consideriamo la lunghezza di questa nuova stringa:

$$|uv^iwx^iy| = |z| + |v|(i-1) + |x|(i-1) = n + |vx|(i-1)$$

Prendiamo ora $i = n+1$, allora otteniamo $|uv^{n+1}wx^{n+1}y| = n+|vx|n = n(|vx|+1)$. Adesso, poiché sappiamo per ipotesi che $|vx| > 0$ allora $|uv^{n+1}wx^{n+1}y| \neq n$ ed è divisibile sia per n che per $|vx|+1$, ovvero non è primo, quindi non appartiene ad L. Abbiamo in tal modo dimostrato che L non è context free. □

ESERCIZIO 4.4. *Si consideri il linguaggio*

$$L = \{\ 0^m10^n10^{mn} \mid m, n \in \mathbb{N}\ \}$$

Si dimostri che L *non è context free.*

SOLUZIONE. Dimostriamo che il linguaggio non è context free. Per provare ciò utilizziamo il Pumping Lemma per i linguaggi context free, ovvero per ogni costante k troviamo una stringa z del linguaggio tale che $|z| \geq k$ e, per ogni suddivisione $z = uvwxy$ tale che $|vwx| < k$ e $|vx| > 0$, dimostriamo che esiste un $i \in \mathbb{N}$ tale per cui uv^iwx^iy non appartiene al linguaggio. Per ogni $k \in \mathbb{N}$ prendiamo la stringa $z = 0^k10^k10^{k^2}$ e consideriamo ogni possibile suddivisione $z = uvwxy$. Dimostriamo che per ogni suddivisione abbiamo che si ha $uv^2wx^2y \notin L$. Andiamo per casi. Se v, oppure x, contiene almeno uno degli 1 allora è evidente che $uv^2wx^2y \notin L$ in quanto aumenta il numero di 1. Se entrambi v e x sono nello stesso gruppo di 0 allora è ancora evidente che $uv^2wx^2y \notin L$ in quanto si aumenta un gruppo di 0 senza aumentare gli altri due perdendo la relazione tra le quantità di 0 dei diversi gruppi. supponiamo allora che v e x siano in gruppi di zeri diversi. In particolare supponiamo che v sia nel primo gruppo e x sia nel secondo. Questo significa che $uv^2wx^2y = 0^{k+|v|}10^{k+|x|}10^{k^2}$. Perché tale stringa stia nel linguaggio deve valere che $k^2 = (k+|v|)(k+|x|) = k^2 + k|vx| + |v||x|$, ovvero $k|vx| + |v||x| = 0$, possibile se e solo se $|vx| = 0$ e dunque non ammesso dal Pumping Lemma. Infine notiamo che abbiamo esaurito tutti i casi. Infatti non può accadere che v sia nel primo gruppo di 0 e x nel terzo perché ciò significherebbe che $|w| > k$, ovvero che $|vwx| > k$, assurdo per ipotesi. Percui la dimostrazione che L non è context free è conclusa. □

ESERCIZIO 4.5. *Si consideri il linguaggio*

$$L = \{\ a_1a_2b_1b_2\ldots b_n \in \Sigma^* \mid n \geq 2,\ \exists k < n\ .\ a_1a_2 = b_kb_{k+1}\ \}$$

e si dimostri che $L' = \{\, a_1a_2b_1b_2\dots b_n \in L \mid n \text{ è potenza di } 2 \,\}$ *non è context free.*

SOLUZIONE. Dimostriamo che il linguaggio non è context free. Per provare ciò utilizziamo il Pumping Lemma per i linguaggi context free, ovvero per ogni costante k troviamo una stringa z del linguaggio tale che $|z| \geq k$ e, per ogni suddivisione $z = uvwxy$ tale che $|vwx| < k$ e $|vx| > 0$, dimostriamo che esiste un $i \in \mathbb{N}$ tale per cui uv^iwx^iy non appartiene al linguaggio. Per ogni $k \in \mathbb{N}$ prendiamo la stringa $z = a_1a_2b_1b_2\dots b_n$ con $n = 2^k$ e consideriamo ogni possibile suddivisione $z = uvwxy$. Dimostriamo che per ogni suddivisione abbiamo $uv^2wx^2y \notin L$, per fare ciò consideriamo la lunghezza di questa nuova stringa, sapendo che $|z| = 2 + 2^k$:

$$|uv^2wx^2y| = |z| + |vx| = 2 + 2^k + |vx|$$

□

Perché tale stringa stia nel linguaggio, in particolare deve anche esistere un $h \in \mathbb{N}$ tale che $2 + 2^k + |vx| = 2 + 2^h$. Essendo l'esponenziale una funzione crescente è evidente che tale relazione implica che $h > k$, essendo $|vx| > 0$, dunque esiste $m > 0$ tale che $h = k + m$. Possiamo perciò riscrivere l'equazione come $2^k + |vx| = 2^{k+m} = 2^k2^m$, ovvero $|vx| = 2^k(2^m - 1) > 2^k > k$ essendo $m > 0$. Questo significa che le uniche suddivisioni che permetterebbero a uv^2wx^2y di restare in L non sono ammesse dal Pumping Lemma. Ovvero abbiamo dimostrato che L non è context free.

ESERCIZIO 4.6. *Sia* $\Sigma = \{0, 1\}$ *e si consideri la famiglia, al variare di* $i \in \mathbb{N}$*, di insiemi:*

$$S(i) = \{w \in \Sigma^* : \textit{ il simbolo } i\textit{-esimo di } w \textit{ a partire da sinistra è uno } 0\}$$

Si collochino nella gerarchia di Chomsky i linguaggi:

$$L_1 = \bigcup_{i \text{ è un numero dispari}} S(i)$$
$$L_2 = \bigcup_{i \text{ è un quadrato perfetto}} S(i)$$

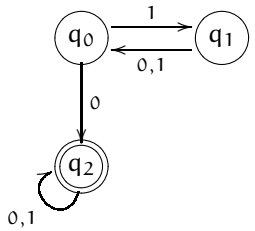
In particolare,

1. *Qualora uno di loro sia regolare, se ne fornisca l'automa minimo che lo accetta. Si dimostri che è minimo e che accetta esattamente tale linguaggio.*
2. *Qualora uno di loro non sia regolare, lo si dimostri formalmente. Si verifichi inoltre l'appartenenza (o meno) all'insieme dei linguaggi liberi dal contesto.*

SOLUZIONE. Consideriamo prima il linguaggio:

$$L_1 = \bigcup_{i \text{ è un numero dispari}} S(i)$$

Il linguaggio L_1 contiene tutte le stringhe del tipo: $0xxx\dots$, $xx0x\dots$, $xxxx0x\dots$, ecc. con $x \in \{0, 1\}$. Questo linguaggio è regolare, infatti riusciamo a costruire un automa che lo descrive:



descritto come $M = \langle\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\}\rangle$. Dimostriamo ora che $L_1 = L(M)$ dimostrando separatamente che: se $x \in L_1$ allora $x \in L(M)$, mentre se $x \notin L_1$ allora $x \notin L(M)$. Dimostriamo entrambe le cose per induzione sulla lunghezza della stringa. Notiamo comunque che è banale dimostrare (sempre per induzione sulla lunghezza della stringa) che se $z \notin L_1$ allora quando $|z|$ è pari si ha che $\widehat{\delta}(q_0, z) = q_0$, mentre quando $|z|$ è dispari allora $\widehat{\delta}(q_0, z) = q_1$.

Base:: Sia $|x| = 1$. Se $x = 0$ allora, per definizione di L_1, si ha che $x \in L_1$ e per definizione di M abbiamo $\delta(q_0, 0) = q_2 \in F$, ovvero $x \in L(M)$. Se invece $x = 1$, allora $x \notin L_1$ e inoltre per definizione di M abbiamo che $\delta(q_0, 1) = q_1 \notin F$;

Induzione:: Supponiamo che se $|y| = n$ e $y \in L_1$ allora $y \in L(M)$, mentre se $y \notin L_1$ allora $y \notin L(M)$. Consideriamo allora $x = ya$ tale che $|x| = n + 1$. Se $y \in L_1$ allora esiste $i \in \mathbb{N}$ dispari tale che $y_i = 0$, ma tale 0 si trova in posizione dispari anche dentro $x = ya$, quindi $x \in L_1$. D'altra parte notiamo che, per ipotesi induttiva si ha $\widehat{\delta}(q_0, y) = q_2 \in F$, allora per definizione di M, qualunque sia a, abbiamo $\widehat{\delta}(q_0, ya) = \delta(q_2, a) = q_2 \in F$. Se invece $y \notin L_1$ allora non esiste in y nessuna posizione dispari contenente 0. A questo punto se $a = 1$ allora siamo sicuri che $x = ya \notin L_1$. D'altra parte, in tal caso, sappiamo per ipotesi induttiva che $\widehat{\delta}(q_0, y) \notin F$ e per come è definito M, questo implica che anche $\widehat{\delta}(q_0, ya) \notin F$.
Consideriamo ora $a = 0$, in tal caso se $n + 1$ é dispari allora $x \in L_1$. Inoltre questo significa che n è pari quindi, per quanto detto abbiamo che $\widehat{\delta}(q_0, ya) = \delta(q_0, a) = q_2 \in F$. Se infine $n + 1$ è pari allora è evidente che $x = ya \notin L_1$ e d'altra parte si ha che n è dispari quindi $\widehat{\delta}(q_0, ya) - \delta(q_1, a) = q_0 \notin F$.

Banalmente si dimostra la minimalità di tale automa.

Consideriamo ora il linguaggio:

$$L_2 = \bigcup_{i \text{ è un quadrato perfetto}} S(i)$$

Notiamo che la condizione di essere un quadrato perfetto non è, intuitivamente, una condizione che rappresenta un fatto regolare, in quanto i quadrati perfetti crescono in modo esponenziale, è dunque difficile che esistano un automa o una grammatica capaci di riconoscerlo, per questo dimostriamo direttamente che il linguaggio non è CF. Dimostriamo ciò usando il Pumping Lemma per i linguaggi CF (non essendo CF, non è nemmeno regolare, dato che i linguaggi regolari sono un sottoinsieme dei linguaggi CF). Perciò per ogni costante k troviamo una stringa z del linguaggio

tale che $|z| \geq k$ e, per ogni suddivisione $z = uvwxy$ tale che $|vwx| < k$ e $|vx| > 0$, dimostriamo che esiste un $i \in \mathbb{N}$ tale per cui $uv^i wx^i y$ non appartiene al linguaggio. Allora preso $k \geq 0$ consideriamo

$$z = \overbrace{1 \dots 1}^{h-1} 0$$

dove $h = k^2$. Dimostriamo che qualunque sia la lunghezza $|vx|$ si ha che la stringa $uv^2wx^2y \notin L$. Per prima cosa supponiamo che sia v che x siano nel gruppo di 1. Allora notiamo che

$$|uv^2wx^2y| = |z| + |vx| = k^2 + |vx|$$

Affinché tale stringa stia il L deve esistere $m \in \mathbb{N}$ tale che $k^2 + |vx| = m^2$, ovvero la lunghezza della nuova stringa deve ancora essere un quadrato. Essendo il quadrato una funzione crescente ed essendo, per ipotesi del Pumping Lemma, $|vx| > 0$ si ha che $m > k$ quindi $\exists n \in \mathbb{N} \,.\, m = k + n$ con $n > 0$. Questo significa che $k^2 + |vx| = (k + n)^2 = k^2 + 2kn + n^2$, ovvero $|vx| = 2kn + n^2$. Poiché n è strettamente maggiore di 0, altrimenti $|vx| = 0$, si ha che l'equazione appena scritta implica $|vx| > k$, assurdo perché le ipotesi del Pumping Lemma impongono $|vx| \leq k$. Consideriamo ora invece il caso in cui v o x comprendano lo 0 finale. Allora notiamo che $uv^0wx^0y = 1^h$ per qualche $h \in \mathbb{N}$ e questo chiaramente significa che la nuova stringa non ha nessuno 0 in una posizione che è un quadrato perfetto perché non ha 0 in assoluto. Questo significa che il linguaggio dato non è CF. □

CAPITOLO 2

Modelli formali per il calcolo

1. Macchine di Turing

Le Macchine di Turing sono un modello per rappresentare in modo formale una macchina in grado di eseguire algoritmi costituiti da passi elementari e discreti di calcolo. Una Macchina di Turing (MdT) è completamente definita dalla matrice funzionale che definisce la funzione δ di transizione tra le configurazioni della macchina. Una configurazione è rappresentata da una coppia $\langle s_j, q_i \rangle$ costituita dal simbolo letto sul nastro s_j e dallo stato corrente della macchina q_i. Ad ogni configurazione corrisponde un'azione A_k che consiste in uno spostamento della testina verso destra (R) o sinistra (L). Il comportamento di una MdT è quindi descrivibile mediante una tabella in cui le righe rappresentano gli stati del controllo mentre le colonne rappresentano i simboli di ingresso. In una generica cella della matrice viene descritta l'azione eseguita dalla macchina nello stato corrispondente in riga e leggendo dal nastro il simbolo corrispondente in colonna.

		s_j	
q_i		$q_r\ s_k\ L$	

In questo caso, l'azione compiuta, essendo il controllo nello stato q_i e leggendo il simbolo s_j, è in questo caso:

- Scrivere il simbolo s_k al posto di s_j;
- Portarsi nello stato (interno) q_r;
- Spostare la testina sul nastro a sinistra (L).

Il tipico problema riguardante i *modelli formali per il calcolo* è quello di codificare un algoritmo per risolvere un dato problema in uno dei formalismi scelto, ad esempio le Macchine di Turing. Prima di svolgere alcuni esercizi proviamo a dare uno schema generale per la risoluzione di problemi di questo tipo, nel caso delle Macchine di Turing.

Si inizia col definire quale è la configurazione da cui la nostra macchina inizierà la computazione e quale sarà la configurazione alla fine. A questo punto descriviamo l'algoritmo con il quale intendiamo risolvere il problema. Scendendo poi in dettaglio, definiamo l'insieme di simboli necessari e il perché della loro introduzione. Definiamo infine l'insieme degli stati che vogliamo utilizzare descrivendo, per ognuno, quali operazioni esegue la macchina e in che modo evolve la computazione. Ogni stato introdotto deve essere quindi descritto in termini delle informazioni che questo stato descrive rispetto alla computazione prevista per la macchina. Alla fine costruiamo la matrice di transizione che definisce la funzione di transizione δ.

ESERCIZIO 1.1. *Costruire una Macchina di Turing che ordina in modo crescente da sinistra a destra una sequenza di* 0, 1 *sapendo che* $0 < 1$.

SOLUZIONE.

La configurazione iniziale da cui parte la computazione è del tipo

$$\begin{array}{c} \ldots\$\$\$\quad 1\quad 1\,0\ldots1\,0\$\$\$\ldots \\ \Uparrow \\ \boxed{q_0} \end{array}$$

mentre quella finale, con la stringa ordinata, è

$$\begin{array}{c} \ldots\$\$\$\quad \$\quad 0\ldots1\$\$\$\ldots \\ \Uparrow \\ \boxed{q_3} \end{array}$$

Descriviamo ora l'algoritmo con cui vogliamo risolvere il nostro problema e la funzione di ogni stato della Macchina di Turing che stiamo costruendo. Per ordinare una sequenza di 0 e 1 possiamo pensare di scorrere la stringa da sinistra a destra finché non incontriamo un 1; a questo punto vogliamo cercare uno 0 con cui scambiarlo, allora scorriamo tutta la stringa verso destra per cercare il primo 0 da destra. Una volta trovato eseguiamo lo scambio e ripetiamo l'algoritmo fino al completo ordinamento della stringa.

L'insieme dei simboli che utilizzeremo è $\Sigma = \{\$, \&, 0, 1\}$, dove & verrà utilizzato per marcare gli 1 da spostare.
Infine gli stati necessari sono

- q_0 =: la macchina va a destra finché non trova il primo 1 da spostare, quando lo trova lo marca con un & e va in q_1.
- q_1 =: scorre la stringa a destra fino in fondo e va in q_2.
- q_2 =: torna a sinistra cercando il primo 0 da sostituire con un 1, se lo trova lo sostituisce se invece trova prima un & vuol dire che la stringa è già ordinata e rimette 1 al posto di &; in ogni caso la macchina va nello stato q_3.
- q_3 =: scorre la stringa a sinistra, se trova & lo sostituisce con uno 0 completando lo scambio e va in q_0, altrimenti scorre tutta la stringa e termina.

Dunque una possibile definizione della funzione δ è

δ	$\$$	&	0	1
q_0			$q_0\ 0\ R$	$q_1\ \&\ R$
q_1	$q_2\ \$\ L$		$q_1\ 0\ R$	$q_1\ 1\ R$
q_2		$q_3\ 1\ L$	$q_3\ 1\ L$	$q_2\ 1\ L$
q_3		$q_0\ 0\ L$	$q_3\ 0\ L$	$q_3\ 1\ L$

Si osservi che qualora la stringa sia nulla, il comportamento della macchina è comunque consistente con quanto specificato dell'esercizio.

□

ESERCIZIO 1.2. *Costruire la Macchina di Turing che calcoli la somma di due numeri forniti in notazione binaria.*

SOLUZIONE. Supponiamo di inserire i due numeri di seguito separati da \$ e che all'inizio della computazione la testina sia sulla cifra meno significativa del secondo numero a partire da sinistra, cioè supponiamo che la configurazione iniziale sia del tipo

$$\begin{array}{c} \ldots\$\$\$10\$1\ \ 0\ \ \$\$\$\ldots \\ \Uparrow \\ \boxed{q_0} \end{array}$$

mentre quella finale, con la testina alla destra del risultato, è del tipo

$$\begin{array}{c} \ldots\$\$\$100\ \ \$\ \ 11\$\$\$\ldots \\ \Uparrow \\ \boxed{q_0} \end{array}$$

dove alla destra della testina troviamo una sequenza di 1 priva di significato.

Vediamo ora l'algoritmo che intendiamo implementare. Per eseguire la somma procediamo eseguendo una successione dei seguenti passi: sottraiamo un'unità al numero a destra della posizione iniziale e successivamente aggiungiamo l'unità all'altro numero fino a quando il primo diventa una stringa di 0. Per la costruzione della Macchina di Turing che eseguirà la somma useremo l'insieme di simboli $\Sigma = \{\$, 0, 1\}$, mentre gli stati necessari sono

- q_0 =: la macchina esegue la sottrazione binaria togliendo un'unità al numero a destra e va in q_1; se il numero è già zero allora la macchina rimane in q_0 e quando trova \$ termina.
- q_1 =: scorre la stringa a sinistra fino al \$ tra i due numeri e va in q_2.
- q_2 =: esegue la somma binaria di un'unità al numero a sinistra e va in q_3.
- q_3 =: scorre la stringa a destra fino al \$ tra i due numeri e va in q_4.
- q_4 =: scorre ancora la stringa a destra finché la testina non si trova dopo entrambi i numeri e va in q_0.

Perciò una possibile definizione della funzione δ è

δ	\$	0	1
q_0		$q_0\ 1\ L$	$q_1\ 0\ L$
q_1	$q_2\ \$\ L$	$q_1\ 0\ L$	$q_1\ 1\ L$
q_2	$q_3\ 1\ R$	$q_3\ 1\ R$	$q_2\ 0\ L$
q_3	$q_4\ \$\ R$	$q_3\ 0\ R$	$q_3\ 1\ R$
q_4	$q_0\ \$\ L$	$q_4\ 0\ R$	$q_4\ 1\ R$

□

ESERCIZIO 1.3. *Definire una Macchina di Turing che decida quale tra due stringhe di zeri sia la più lunga e descrivere il modo utilizzato dalla macchina per fornire la risposta.*

SOLUZIONE. Vediamo da quale tipo di configurazione iniziale partirà la macchina

$$\begin{array}{c} \ldots\$\$\$0\ldots0\ \ I\ \ 0\ldots0\$\$\$\ldots \\ \Uparrow \\ \boxed{q_0} \end{array}$$

dove I è un simbolo nuovo che utilizziamo per dividere le due stringhe di zeri. Alla fine della computazione ci possiamo trovare in due situazioni. Se le due stringhe sono uguali esse si troveranno alla sinistra della testina, altrimenti la stringa più lunga si troverà alla destra di quest'ultima.

Descriviamo ora l'algoritmo con cui intendiamo risolvere il problema. Cominciamo cercando e marcando il primo zero a sinistra del simbolo I, a questo punto ritorniamo in centro e andiamo a cercare e marcare il primo zero nella stringa di destra. Successivamente riportiamo la testina al centro e ripetiamo il procedimento. Se non trovo più zeri andando verso sinistra allora controllo a destra, se ci sono ancora zeri significa che la stringa a destra è più lunga altrimenti sono uguali. Se invece non trovo più zeri andando verso destra allora la stringa più lunga è quella di sinistra.

L'insieme di simboli che utilizzeremo è $\Sigma = \{\$, \&, *, \mathrm{I}, 0\}$, dove I va inserito per dividere le stringhe, & viene usato per marcare gli zeri contati nella stringa di sinistra mentre $*$ viene usato per marcare gli zeri contati nella stringa di destra. Infine descriviamo gli stati necessari

- q_0 =: la macchina cerca il primo zero verso sinistra, lo marca e va in q_1; se non trova alcuno zero allora va in q_2.
- q_1 =: scorre la sequenza verso destra cercando il primo zero nell'altra stringa, quando lo trova lo marca e va in q_0. Se non trova zeri la macchina va in q_4.
- q_2 =: la macchina torna verso destra dopo non aver trovato zeri a sinistra; se non trova altri zeri decide che le stringhe sono uguali e termina, se invece trova almeno uno zero va in q_3.
- q_3 =: scorre la sequenza e si ferma tra le due stringhe decidendo che quella di destra è più lunga.
- q_4 =: scorre tutta la sequenza verso sinistra e termina decidendo che la stringa di sinistra è più lunga.

A questo punto definiamo la funzione δ che implementa l'algoritmo appena descritto

δ	\$	0	I	&	$*$
q_0	q_2 \$ R	q_1 & R	q_0 I L	q_0 & L	q_0 $*$ L
q_1	q_4 \$ L	q_0 $*$ L	q_1 I R	q_1 & R	q_1 $*$ R
q_2		q_3 0 L	q_2 I R	q_2 0 R	q_2 0 R
q_3		q_3 0 L			q_3 0 L
q_4		q_4 0 L	q_4 I L	q_4 0 L	q_4 0 L

Come si può notare, abbiamo definito la funzione in modo che alla fine della computazione le stringhe siano nuovamente di zeri. □

ESERCIZIO 1.4. *Si definisca la Macchina di Turing che sposta la testina di* n *posizioni a destra rispetto alla posizione iniziale supponendo che* n *sia fornito in notazione binaria.*

SOLUZIONE. Supponiamo che la macchina parta da una configurazione iniziale del tipo

$$\ldots \$\$\$01\ldots10 \quad \mathrm{I} \quad \$\$\$\ldots$$
$$\Uparrow$$
$$\boxed{q_0}$$

dove il numero n è inserito alla sinistra della posizione iniziale.

Alla fine della computazione la testina si troverà spostata di n posizioni a destra rispetto al simbolo I che rappresenta la posizione iniziale. L'algoritmo che intendiamo implementare consiste nel togliere un'unità al numero binario e conseguentemente aggiungere un $*$ a destra. Si ripetono queste operazioni fino a quando il numero è azzerato, a questo punto facciamo scorrere alla macchina tutti gli asterischi (esattamente n) e terminiamo.

L'insieme dei simboli che utilizziamo è $\Sigma = \{\$, I, *, 0, 1\}$ mentre gli stati sono

- q_0 =: la macchina esegue la sottrazione di un'unità e va in q_1; se il numero è già zero va in q_3.
- q_1 =: aggiunge un asterisco a destra del punto iniziale e va in q_2.
- q_2 =: ritorna sulla posizione iniziale e va in q_0.
- q_3 =: scorre tutta la stringa di zeri fino a I, continua oltrepassando tutti gli asterischi e termina.

Possiamo quindi definire la funzione δ della Macchina di Turing nel seguente modo

δ	\$	0	1	I	$*$
q_0	q_3 \$ R	q_0 1 L	q_1 0 R	q_0 I L	
q_1	q_2 $*$ L	q_1 0 R	q_1 1 R	q_1 I R	q_1 $*$ R
q_2				q_0 I L	q_2 $*$ L
q_3		q_3 0 R	q_3 1 R	q_3 I R	q_3 $*$ R

□

Esercizio 1.5. *Costruire la Macchina di Turing in grado di calcolare il valore n^2 dove n è dato come input in notazione unaria (ad esempio il numero 3 è rappresentato da tre zeri).*

Soluzione. Supponiamo che la testina si trovi inizialmente sulla cifra più a destra della rappresentazione del numero n, supponiamo cioè che la configurazione iniziale sia del tipo

$$\ldots \$\,\$\,\$\,0 \ldots 0 \quad 0 \quad \$\,\$\,\$ \ldots$$
$$\Uparrow$$
$$\boxed{q_0}$$

Alla fine della computazione il risultato sarà tutto alla destra della testina espresso sempre in notazione unaria.

L'algoritmo che intendiamo utilizzare consiste nella ripetizione dei seguenti passi. Si marcano, uno alla volta, con degli $*$ gli zeri del numero dato e, ogni volta, si aggiunge uno zero alla destra della posizione iniziale fino ad aver copiato interamente l'input; a questo punto si marca in modo differente uno degli zeri e si ripetono i passi precedenti. Si continua in questo modo finché il valore n non sarà stato copiato esattamente n volte.

L'insieme dei simboli necessari è $\Sigma = \{0, 1, *, \&, \$\}$, dove $*$ serve a marcare gli zeri già copiati durante la singola copiatura del numero n, & viene utilizzato per marcare gli zeri dopo che il numero è stato copiato interamente (il numero di & indica il numero di volte in cui è stata copiata l'intera stringa), e 1 ha la stessa

funzione di $*$ per lo 0, cioè viene utilizzato per marcare il & nella singola copiatura del numero. Descriviamo infine gli stati che dobbiamo utilizzare

- q_0 =: la macchina marca lo zero o il & per aggiungere uno zero alla destra della posizione iniziale e va in q_1; se non ne trova significa che ha completato una copiatura del numero e va in q_3.
- q_1 =: aggiunge uno zero a destra del punto iniziale e va in q_2.
- q_2 =: ritorna a sinistra fino al primo asterisco e va in q_0.
- q_3 =: ripone tutti gli 1 a & per poterli ricontare nella copiatura successiva, marca un asterisco con & e va in q_4.
- q_4 =: cerca il primo $*$, lo pone a zero per poterlo contare nella copiatura successiva e va in q_5; se non trova asterischi significa che l'operazione è stata completata e termina.
- q_5 =: completa l'operazione di sostituzione degli $*$ con gli zeri e, quando trova il primo zero torna in q_0.

Possiamo ora definire la matrice di transizione

δ	0	$*$	&	1	\$
q_0	$q_1 * R$	$q_0 * L$	$q_1 1 R$	$q_0 1 L$	$q_3 \$ R$
q_1	$q_1 0 R$	$q_1 * R$		$q_1 1 R$	$q_2 0 L$
q_2	$q_2 0 L$	$q_0 * L$			
q_3		$q_4 \& R$		$q_3 \& R$	
q_4	$q_4 0 L$	$q_5 0 R$			
q_5	$q_0 0 L$	$q_5 0 R$			

□

Esercizio 1.6. *Definire la Macchina di Turing in grado di riconoscere il linguaggio* $\{a^n b^{2n} : n \in \mathbb{N}\}$ *e descrivere il modo in cui la macchina fornisce la risposta.*

Soluzione. Supponiamo che venga data in input una sequenza di simboli a e b, lo scopo della nostra macchina è quello di controllare se tale sequenza appartiene o meno al linguaggio fornito dal testo.
Consideriamo una configurazione iniziale del tipo

$$\begin{array}{c} \ldots \$\,\$\,\$ \quad a \quad a \ldots b\, b\, \$\,\$\,\$ \ldots \\ \Uparrow \\ \boxed{q_0} \end{array}$$

mentre quella finale è del tipo

$$\begin{array}{c} \ldots \$\,\$\,\$\,0 \quad * \quad a \ldots b\, b\, \$\,\$\,\$ \ldots \\ \Uparrow \\ \boxed{q_5} \end{array}$$

dove lo 0 sta ad indicare, nel caso particolare, che la stringa non appartiene al linguaggio; in generale la risposta si troverà alla sinistra della testina e sarà 1 se la stringa appartiene al linguaggio, 0 altrimenti; invece alla destra della testina possiamo trovare una stringa composta da a, b e $*$.

Il modo in cui la macchina deciderà l'appartenenza consiste nel cercare la prima a, marcarla, andare in fondo alla stringa a destra per controllare che non finisca con una a e controllare che ci siano due b. A questo punto si ripetono questi passi

finché non finisce la stringa o fino a quando una delle condizioni precedenti risulta non verificata.

L'insieme di simboli di cui necessitiamo è $\Sigma = \{a, b, *, 0, 1, \$\}$, dove $*$ è usato per marcare le a e le b già contate mentre 0 e 1 sono usati per dare la risposta. A questo punto descriviamo gli stati necessari

- q_0 =: la macchina marca la prima a andando verso destra e va in q_1; se trova una b la stringa non può appartenere al linguaggio e quindi va in q_5 mentre se non trova più a o b significa che la stringa è vuota o è stata interamente controllata e dunque va in q_6.
- q_1 =: cerca la prima b verso destra, la marca e va in q_2; se non la trova va in q_5.
- q_2 =: si sposta ancora a destra e se trova una b la marca e va in q_3; se non la trova va in q_5.
- q_3 =: scorre l'intera stringa verso destra, se trova una a va in q_5 altrimenti quando raggiunge la fine della stringa va in q_4.
- q_4 =: ritorna a sinistra fino all'inizio della stringa e va in q_0.
- q_5 =: scorre tutta la stringa verso sinistra; quando arriva alla fine scrive 0 e va in q_7.
- q_6 =: scorre tutta la stringa verso sinistra; quando arriva alla fine scrive 1 e va in q_7.
- q_7 =: termina.

Si noti che gli ultimi tre stati e i simboli 0 e 1 hanno come unica funzione quella di fornire la risposta e terminare.

A questo punto possiamo costruire la matrice di transizione della Macchina di Turing cercata

δ	a	b	\$	$*$	0	1
q_0	$q_1\ *\ R$	$q_5\ b\ L$	$q_6\ \$\ L$	$q_0\ *\ R$		
q_1	$q_1\ a\ R$	$q_2\ *\ R$	$q_5\ \$\ L$	$q_1\ *\ R$		
q_2	$q_5\ a\ L$	$q_3\ *\ R$	$q_5\ \$\ L$			
q_3	$q_5\ a\ L$	$q_3\ b\ R$	$q_4\ \$\ L$			
q_4	$q_4\ a\ L$	$q_4\ b\ L$	$q_0\ \$\ R$	$q_4\ *\ L$		
q_5	$q_5\ a\ L$	$q_5\ b\ L$	$q_7\ 0\ R$	$q_5\ *\ L$		
q_6	$q_6\ a\ L$	$q_6\ b\ L$	$q_7\ 1\ R$	$q_6\ *\ L$		
q_7						

□

2. Funzioni ricorsive

Anche per il modello formale di calcolo definito dalle funzioni ricorsive (parziali o totali che siano) bisogna codificare un algoritmo per risolvere un dato problema nel formalismo dato. Innanzitutto, dobbiamo cercare di costruire un algoritmo che risolva il nostro problema in modo ricorsivo. Questo significa che il risultato deve essere ottenuto mediante composizione (composizione di funzioni, ricorsione primitiva, minimizzazione o μ-ricorsione) ed utilizzando solamente la funzione stessa che stiamo definendo ed eventualmente altre funzioni ricorsive che siano note o a loro volta definibili in modo ricorsivo. Per semplicità le funzioni già definite nelle

dispense o in esercizi precedenti non vengono ridefinite ma si considerano come note.

ESERCIZIO 2.1. *Si definisca la funzione primitiva ricorsiva* ndiv : $\mathbb{N} \to \mathbb{N}$ *che, dato* $x \in \mathbb{N}$*, fornisce il numero dei suoi divisori.*

SOLUZIONE. Per risolvere il problema utilizziamo delle proprietà dei divisori di un numero intero. Sappiamo che un numero è divisore di x se la divisione tra x e tale numero fornisce resto 0. Poiché possediamo già una funzione ricorsiva che fornisce il resto di una divisione intera possiamo pensare di contare tutti i numeri tali che la divisione di x con essi dia 0. La funzione che fornisce il resto è *mod(x,y)* mentre un'altra funzione utile è $\overline{sg}(z)$; questa infatti, applicata a *mod(x,y)* mi restituisce 1 se il numero y è divisore di x, 0 altrimenti. Ricordiamo che:

Segno:

$$\begin{cases} sg(0) = 0 \\ sg(y+1) = S(0) \end{cases}$$

Inoltre definiamo $\overline{sg}(x) = -(1, sg(x))$.

Modulo:

Assumiamo che $\texttt{mod}\,(x, 0) = 0$.

$$\begin{cases} \texttt{mod}\,(0, x) = 0 \\ \texttt{mod}\,(y+1, x) = S(\texttt{mod}\,(y, x)) \cdot sg(|-(x, S(\texttt{mod}\,(y, x)))|) \end{cases}$$

Sommatoria:

Sia f una funzione ricorsiva primitiva binaria, allora definiamo la funzione $\Sigma_{z<y} f(x, z) = \Sigma_{z=0}^{y-1} f(x, z)$:

$$\begin{cases} \Sigma_{z<0} f(x, z) = 0 \\ \Sigma_{z<y+1} f(x, z) = +(f(x, z), \Sigma_{z<y} f(x, z)) \end{cases}$$

Vediamo allora la formalizzazione dell'algoritmo come funzione ricorsiva definita per casi.

$$\begin{cases} ndiv(0) = 0 \\ ndiv(x) = (\Sigma_{y<x+1}\ \overline{sg}(\texttt{mod}\,(x, y))) - 1 \end{cases}$$

Si noti che la sottrazione di un'unità è necessaria perché, per come è costruita la funzione, *ndiv* conterebbe come divisore anche lo zero che, come sappiamo, non è divisore di nessun naturale. □

ESERCIZIO 2.2. *Si definisca la funzione ricorsiva primitiva* primo : $\mathbb{N} \to \mathbb{N}$ *che, dato* $x \in \mathbb{N}$*, restituice* 1 *se* x *è primo,* 0 *altrimenti.*

SOLUZIONE. Per risolvere il problema dobbiamo identificare una caratteristica specifica dei numeri primi. Essa può essere la definizione stessa di numero primo, cioè un numero è primo se ha come unici divisori 1 e se stesso, questo significa che il numero di divisori è esattamente due. Infatti un numero è primo se e solo se ha esattamente due divisori. Possiamo allora utilizzare la funzione definita nell'esercizio 2.1 contando il numero di divisori di x dato in input e restituendo 1 se tale numero è proprio 2.

Dunque la definizione formale per casi è

$$\begin{cases} primo(0) = 0 \quad primo(1) = 1 \\ primo(x) = \overline{sg}(-(ndiv(x), 2)) \end{cases}$$

Cioè determiniamo il numero di divisori con *ndiv*, a questo togliamo 2; se il risultato è zero il numero è primo e restituiamo 1, altrimenti restituiamo 0. □

ESERCIZIO 2.3. *Si definisca la funzione primitiva ricorsiva* $p : \mathbb{N} \to \mathbb{N}$ *che dato* $x \in \mathbb{N}$ *restituisce l'x-esimo numero primo (si assuma* $p(0) = 0$*).*

SOLUZIONE. Per risolvere tale problema dobbiamo trovare un modo per contare i numeri primi fino ad arrivare all'x-esimo. Per fare questo potremmo partire da 0 e per ogni numero naturale controllare se è primo. In caso affermativo togliamo un'unità al numero x altrimenti lo lasciamo invariato. Terminando tale procedimento quando x diventa nullo individuiamo il valore cercato. Vediamo la definizione formale e poi ne descriviamo l'esatto funzionamento. Definiamo

$$p(x) = q(x, 0)$$

dove $q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ è tale che

$$\begin{cases} q(x, y) = q(-(x, \mathit{primo}(S(y))), S(y)) \\ q(0, y) = y \end{cases}$$

L'introduzione della nuova funzione con due parametri serve per gestire contemporaneamente sia il contatore dei numeri primi x sia il valore che alla fine sarà il numero cercato y. Si noti comunque che tale tipo di ricorsione è insolito perché l'utilizzo della funzione ricorsiva di base successore fa in modo che uno dei parametri aumenti continuamente. La terminazione è comunque garantita dal fatto che, vista l'infinitezza dell'insieme dei numeri primi, per ogni x il procedimento arriva alla conclusione perché sempre riusciremo a trovare almeno x numeri primi.

Si noti come la medesima funzione sia anche definibile per minimizzazione o μ-ricorsione nel seguente modo:

$$\begin{array}{lcl} p(0) & = & 0 \\ p(1) & = & 2 \\ p(n+1) & = & f(\Pi_{i \leq n} p(i) + 1, p(n)) \end{array}$$

dove: $f(x, y) = \mu\ z < x.\ (\mathit{sg}(z - y) = 1\ \wedge\ \mathit{primo}(z))$. Questa definizione segue dall'osservazione che, supponendo $\{p_n\}_n \in \mathbb{N}$ essere la successione dei numeri primi, esiste sempre x primo tale che: $p_n < x \leq \Pi_{i \leq n} p_i + 1$. □

ESERCIZIO 2.4. *Si definisca la funzione ricorsiva primitiva* $\mathrm{expf} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *che, dato* $x, y \in \mathbb{N}$*, restituisce l'esponente del y-esimo numero primo nella fattorizzazione in numeri primi di* x*.*

SOLUZIONE. In questo caso dobbiamo cercare di contare il numero di volte in cui il numero $p(y)$, ottenuto mediante la funzione definita nell'esercizio 2.3, è contenuto nel numero x.

Definiamo formalmente la funzione

$$\begin{cases} \mathit{expf}(0, y) = 0 \\ \mathit{expf}(x, y) = +(\overline{\mathit{sg}}(\ \mathtt{mod}\ (x, p(y))), \mathit{expf}(\mathit{div}(x, p(y)), y)) \end{cases}$$

Cioè se $p(y)$ è divisore di x aggiungiamo 1 al risultato della funzione richiamata sul quoziente tra x e $p(y)$, e y. In caso contrario sommiamo una sequenza di zeri. □

ESERCIZIO 2.5. *Definire la funzione primitiva ricorsiva* $\text{ifcubo} : \mathbb{N} \to \{0,1\}$ *che restituisce* 1 *se il valore* x *dato in input è un cubo perfetto,* 0 *altrimenti.*

SOLUZIONE. Per dimostrare che x è un cubo perfetto dobbiamo trovare un valore, sicuramente minore o uguale a x stesso, tale che moltiplicato per sè stesso tre volte dia esattamente x. Per trovare tale valore possiamo utilizzare la funzione di minimizzazione, controllando per ogni valore minore di x se il suo triplo prodotto meno x è esattamente zero. Forniamo dunque la definizione formale della funzione

$$\begin{cases} \mathit{ifcubo}(0) = \mathit{ifcubo}(1) = 1 \\ \mathit{ifcubo}(x) = sg(-(x, (\mu y < x. - (x, \cdot(y, \cdot(y, y)) = 0)))) \end{cases}$$

Quindi se esiste un y il cui triplo prodotto è x l'operatore di minimalizzazione lo restituisce e la funzione segno viene applicata a un valore diverso da zero dando 1. In caso contrario alla fine della computazione la funzione vale zero. □

ESERCIZIO 2.6. *Si definisca la funzione primitiva ricorsiva* $\text{ifscubi} : \mathbb{N} \to \{0,1\}$ *che restituisce* 1 *se il valore* x *dato in input è somma di cubi,* 0 *altrimenti.*

SOLUZIONE. La risoluzione di tale esercizio risulta semplificata se si utilizza la funzione definita nell'esercizio 2.5 Infatti per ogni valore y minore del'input x controlliamo se sia y che $(x - y)$ sono cubi mediante la funzione *ifcubo*. Definiamo dunque formalmente la funzione

$$\begin{cases} \mathit{ifscubi}(0) = 1 \\ \mathit{ifscubi}(x) = sg(-(x, (\mu y < x.\overline{sg}(\cdot(\mathit{ifcubo}(y), \mathit{ifcubo}(-(x, y)))) = 0))) \end{cases}$$

□

ESERCIZIO 2.7. *Si definisca la funzione primitiva ricorsiva* $f : \mathbb{N} \to \mathbb{N}$ *che conta il numero di occorrenze della cifra* 5 *in un numero naturale dato in base decimale.*

SOLUZIONE. La risoluzione di tale esercizio è alquanto più delicata rispetto agli esercizi fin qui risolti. In questo caso dobbiamo trovare un modo per analizzare una alla volta le cifre di un numero decimale. A questo scopo ci vengono in aiuto le funzioni `mod` e *div*, infatti eseguendo la divisione intera tra un numero decimale e 10 si ottiene lo stesso numero privato della cifra meno significativa, mentre calcolando il resto della stessa divisione si ottiene la proprio la cifra più a destra del numero (si noti che tale fatto vale per numeri in qualsiasi base purché vengano divisi per la loro base). Dunque possiamo risolvere l'esercizio costruendo una funzione che ricorsivamente scinda il numero nelle sue cifre e controlli quante di queste siano 5. Diamo quindi la definizione formale

$$\begin{cases} f(0) = f(1) = f(2) = f(3) = f(4) = f(6) = f(7) = f(8) = f(9) = 0 \\ f(5) = 1 \\ f(x) = +(f(\mathit{div}(x, 10)), f(\texttt{ mod }(x, 10))) \end{cases}$$

□

ESERCIZIO 2.8. *Si definisca una funzione primitiva ricorsiva* $f : \mathbb{N} \to \mathbb{N}$ *tale che se* $n \in \mathbb{N}$ *è l'input, essa restituisca la cifra decimale più a sinistra del numero* n.

SOLUZIONE. Si noti che l'esercizio richiede la cifra più a sinistra mentre noi sappiamo già selezionare quella più a destra. Volendo sfruttare questa capacità possiamo notare che il risultato della nostra funzione applicato a n è lo stesso di quello della funzione applicata al numero ottenuto da n togliendo la cifra più a destra, poiché tale operazione non modifica la cifra più a sinistra. Così la definizione formale della funzione è

$$\begin{cases} f(0) = 0 \\ f(1) = 1 \\ f(2) = 2 \\ f(3) = 3 \\ f(4) = 4 \\ f(5) = 5 \\ f(6) = 6 \\ f(7) = 7 \\ f(8) = 8 \\ f(9) = 9 \\ f(x) = f(\mathit{div}(x, 10)) \end{cases}$$

Abbiamo inserito anche il caso $f(0)$ per considerare la situazione in cui l'input iniziale sia 0. □

ESERCIZIO 2.9. *Si definisca la funzione primitiva ricorsiva* $f : \mathbb{N} \to \mathbb{N}$ *che dato un numero decimale in input restituisce il reverse dello stesso, ad esempio se* $x = 247$ *allora la funzione deve restituire* 742.

SOLUZIONE. In questo caso, per risolvere l'esercizio, dobbiamo cercare di sfruttare il significato della posizione delle singole cifre nel numero. Cioè un numero decimale è ottenuto come somma di ogni cifra componente moltiplicata per 10 elevato al valore indicante la posizione della cifra nel numero, meno un'unità, contando da destra; nell'esempio si ha $247 = 2 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0$. A questo punto potendo scindere il numero nelle sue cifre componenti possiamo costruire il numero reverse dell'input trovando i giusti esponenti di 10 per ogni cifra. Ad esempio per ottenere 742 la cifra 7 va moltiplicata per 10^2 dove 2 è la lunghezza del numero 247 meno un'unità, potremmo poi completare l'operazione sommando il risultato a ciò che si ottiene eseguendo ricorsivamente gli stessi passi sul numero 24. Si noti che per fare ciò che abbiamo appena descritto abbiamo bisogno di una funzione che calcoli la lunghezza del numero e di una che esegue l'effettivo reverse. Definiamo formalmente le due funzioni

$$\begin{cases} f(0) = 0 \\ f(x) = +(f(\mathit{div}(x, 10)), \cdot(\mathtt{mod}\,(x, 10), 10^{(\mathit{lung}(x)-1)})) \end{cases}$$

$$\begin{cases} \mathit{lung}(0) = 0 \\ \mathit{lung}(x) = S(\mathit{lung}(\mathit{div}(x, 10))) \end{cases}$$

□

3. Programmi for-while

Il formalismo dei linguaggi di programmazione (**for**, **while**, **repeat**, etc.), consiste in un linguaggio (sintassi) per esprimere algoritmi ed in una macchina astratta (semantica) che definisce come i vari costrutti vengono interpretati in termini di

variazione o trasformazione di stato. Nel caso dei linguaggi di programmazione, gli esercizi non riguardano la codifica di un algoritmo per risolvere un dato problema in un liguaggio di programmazione anch'esso dato. Questo tipo di esercizi è infatti materia di studio in altri corsi, quali i corsi di programmazione, algoritmi e strutture dati e linguaggi. In questo contesto concentreremo l'attenzione esclusivamente sullo studio della espressività di questi linguaggi di programmazione. L'espressività è qui intesa in termini di capacità, fornita dai costrutti del linguaggio e dalla loro semantica, di esprimere funzioni ricorsive, ovvero calcolabili. In questo senso, il tipico esercizio consiste nel definire formalmente un costrutto linguistico o un linguaggio di programmazione e studiarne l'espressività, ovvero studiare la classe di funzioni (ricorsive di base, primitive ricorsive, ricorsive totali o parziali ricorsive) che tali costrutti permettono di calcolare. Per noi, un linguaggio di programmazione sarà quindi definito in termini sintattici da una grammatica libera da contesto ed in termini semantici da un sistema di transizione che definisce formalmente la semantica dei vari costrutti linguistici. Ricordiamo a tale proposito che un linguaggio di programmazione con assegnamento (**:=**), condizionale (**if then else**) ed operazioni aritmetiche di base ma che non fornisce costrutti per l'iterazione è in grado di esprimere solo funzioni ricorsive di base. Analogamente, i linguaggi che permettono la composizione di comandi ed un costrutto per l'iterazione determinata, ovvero il cui numero di iterazioni è stabilito a priori prima dell'esecuzione dei comandi iterativi (tipo **for**), permettono di esprimere tutte e solo le funzioni primitive ricorsive (Teorema di Meyer & Ritchie); mentre i linguaggi che permettono anche iterazioni non determinate (tipo **while**) permettono di esprimere tutta la classe delle funzioni intuitivamente calcolabili della Tesi di Church.

ESERCIZIO 3.1. *Definire il linguaggio* REPEAT*:*

(1) *Definire la sintassi di* REPEAT*;*
(2) *Definire la semantica;*
(3) *Definire la* REPEAT*-calcolabilità:*
(4) *Dimostrare che una funzione* f *è* REPEAT*-calcolabile se e solo se è* WHILE*-calcolabile;*

SOLUZIONE.

(1) Per definire la sintassi del nuovo linguaggio possiamo utilizzare quella già definita per il linguaggio WHILE semplicemente sostituendo il comando **while** con il nuovo comando **repeat**. Perciò la sintassi del nostro linguaggio diventa

Espressioni:

$$E ::= 0 | x | E_1 + 1 | E_1 \neq E_2$$

Comandi:

$$C ::= x := E | C_1; C_2 | \textbf{skip} | \textbf{if } E \textbf{ then } C_1 \textbf{ else } C_2 \textbf{ fi} | \textbf{repeat } C \textbf{ until } E$$

Programmi:

$$P ::= \textbf{read } x; C; \textbf{write } y$$

(2) A questo punto dovremmo fornire la semantica di tutte le strutture sintattiche del nostro linguaggio, ma visto che la maggioranza di esse sono già state descritte nelle dispense forniamo la semantica solo del nuovo comando il cui compito consiste nel ripetere il comando C fino al momento

in cui l'espressione E diventa vera. Vediamo ora la semantica formale

$$\frac{\mathcal{E}[\![\mathsf{E}]\!]\sigma' = \mathit{ff}, \quad \langle \mathsf{C}, \sigma\rangle \to \sigma', \quad \langle \textbf{repeat } \mathsf{C} \textbf{ until } \mathsf{E}, \sigma'\rangle \to \sigma''}{\langle \textbf{repeat } \mathsf{C} \textbf{ until } \mathsf{E}, \sigma\rangle \to \sigma''}$$

Cioè si esegue il comando, si controlla la guardia e se questa risulta falsa si riesegue l'intero **repeat**. Invece se la guardia è vera dopo aver eseguito il comando si esce dal ciclo. Si noti che il comando viene sempre eseguito almeno una volta.

$$\frac{\mathcal{E}[\![\mathsf{E}]\!]\sigma' = \mathit{tt}, \quad \langle \mathsf{C}, \sigma\rangle \to \sigma'}{\langle \textbf{repeat } \mathsf{C} \textbf{ until } \mathsf{E}, \sigma\rangle \to \sigma'}$$

(3) Diciamo che una funzione $\mathsf{f} : \mathbb{N} \to \mathbb{N} \cup \{\uparrow\}$ è Repeat-calcolabile se e solo se esiste $\mathsf{P} \in$Repeat-Programmi tale che $[\![\mathsf{P}]\!] = \mathsf{f}$.

(4) Per dimostrare che una funzione è Repeat-calcolabile se e solo se è While-calcolabile è sufficiente simulare il comando **repeat** con il comando **while** e viceversa, questo perché i due linguaggi differiscono solo in questi due comandi.

$$\textbf{repeat } \mathsf{C} \textbf{ until } \mathsf{E} \;\cong\; \mathsf{C}; \textbf{ while } \neg\mathsf{E} \textbf{ do } \mathsf{C} \textbf{ endw}$$

Mentre il viceversa

$$\textbf{while } \mathsf{E} \textbf{ do } \mathsf{C} \textbf{ endw} \;\cong\; \textbf{if } \mathsf{E} = \mathit{tt} \textbf{ then repeat } \mathsf{C} \textbf{ until } \neg\mathsf{E} \textbf{ else skip fi}$$

□

Esercizio 3.2. *Ripetere i primi tre punti dell'Esercizio 3.1 per il linguaggio* Loop*, con il comando*

$$\textbf{loop } \mathsf{E} \textbf{ do } \mathsf{C} \textbf{ endl}$$

al posto del comando **repeat***, la cui la semantica intuitiva è di controllare la guardia* E *e, se è diversa da* 0*, eseguire* C *e decrementare la guardia. Dimostrare poi che una funzione è* Loop*-calcolabile se e solo se è* For*-calcolabile.*

Soluzione. Per semplicità non riportiamo l'intero linguaggio facendo riferimento a quello riportato nell'Esercizio 3.1 sia qui che negli esercizi successivi. L'unica modifica che apportiamo a tale linguaggio per renderlo il Loop-linguaggio è quella di sostituire i due comandi come riportato nel testo dell'esercizio. A questo punto in base alla semantica intuitiva del nuovo comando fornita definiamo la semantica formale

$$\frac{\mathcal{E}[\![\mathsf{E}]\!]\sigma = 0}{\langle \textbf{loop } \mathsf{E} \textbf{ do } \mathsf{C} \textbf{ endl}, \sigma\rangle \to \sigma}$$

Cioè se la guardia è zero il comando non deve fare nulla.

$$\frac{\mathcal{E}[\![\mathsf{E}]\!]\sigma = n > 0, \quad \langle \mathsf{C}, \sigma\rangle \to \sigma', \quad \langle \textbf{loop } n-1 \textbf{ do } \mathsf{C} \textbf{ endl}, \sigma'\rangle \to \sigma''}{\langle \textbf{loop } \mathsf{E} \textbf{ do } \mathsf{C} \textbf{ endl}, \sigma\rangle \to \sigma''}$$

Se invece la guardia è diversa da zero, viene eseguito C e poi, sulla memoria modificata da questo, si esegue il **loop** con la guardia decrementata. Si noti che nel **loop** invece di passare $\mathsf{E} - 1$ passiamo $n - 1$, dove n è il valore iniziale di E; il motivo di questa scelta sta nel fatto che passando il valore decrementato siamo sicuri che

prima o poi la guardia arriva a zero e il ciclo termina; se invece decrementiamo la guardia ed essa, per caso, contiene variabili modificate in C, la terminazione non sarebbe più garantita.

Per quel che riguarda la calcolabilità diciamo che una funzione $f : \mathbb{N} \to \mathbb{N} \cup \{\uparrow\}$ è LOOP-calcolabile se e solo se esiste $P \in$LOOP-programmi tale che $[\![P]\!] = f$.

Veniamo infine alla dimostrazione di equivalenza. Come in precedenza dobbiamo simulare ognuno dei due comandi con l'altro.

$$\textbf{loop } E \textbf{ do } C \textbf{ endl} \ \cong\ \textbf{for } I = 1 \textbf{ to } E \textbf{ do } C \textbf{ endf}$$
dove $I \notin \mathcal{V}ar(C)$

L'ultima posizione garantisce che la variabile I non viene modificata dall'esecuzione del corpo C. Vediamo ora il viceversa

$$\textbf{for } I = E_1 \textbf{ to } E_2 \textbf{ do } C \textbf{ endf} \ \cong$$
$$I = E_1;\ \textbf{loop } E_2 - E_1 + 1 \textbf{ do } C;\ I = I + 1 \textbf{ endl}$$

□

ESERCIZIO 3.3. *Eseguire l'Esercizio 3.2 sotituendo al comando* **loop** *il comando*

$$\textbf{loop } E_1 \textbf{ step } E_2 \textbf{ do } C \textbf{ endl}$$

la cui semantica intuitiva è quella di eseguire C *tante volte quante* E_2 *divide* E_1. *Si ottiene così il* LOOP-STEP*-linguaggio.*

SOLUZIONE. Supponendo che il nostro linguaggio abbia la sintassi già definita, definiamo la semantica in modo formale del nuovo comando

$$\frac{\mathcal{E}[\![E_1]\!]\sigma = n_1, \quad \mathcal{E}[\![E_2]\!]\sigma = n_2, \quad n_2 \leq n_1, \quad \langle C; \textbf{loop } n_1 - n_2 \textbf{ step } n_2 \textbf{ do } C \textbf{ endl}, \sigma\rangle \to \sigma'}{\langle \textbf{loop } E_1 \textbf{ step } E_2 \textbf{ do } C \textbf{ endl}, \sigma\rangle \to \sigma'}$$

Quindi se la divisione tra i due valori è possibile, viene eseguito C e poi il **loop-step** con il valore della prima guardia decrementata del valore della seconda. Anche in questo caso la scelta di usare i valori delle espressioni e non le spressioni stesse garantisce la terminazione del comando.

$$\frac{\mathcal{E}[\![E_1]\!]\sigma = n_1, \quad \mathcal{E}[\![E_2]\!]\sigma = n_2, \quad (n_2 > n_1 \ \vee \ n_2 = 0)}{\langle \textbf{loop } E_1 \textbf{ step } E_2 \textbf{ do } C \textbf{ endl}, \sigma\rangle \to \sigma}$$

Se invece la divisione non è possibile allora il comando non fa nulla. La LOOP-STEP-calcolabilità è analoga alla LOOP-calcolabilità, e per questo evitiamo di ripeterla. Infine per ciò che riguarda l'equivalenza tra il nuovo comando e il **for**, vediamo come un comando può simulare l'altro e viceversa.

$$\textbf{loop } E_1 \textbf{ step } E_2 \textbf{ do } C \textbf{ endl} \ \cong\ \textbf{for } I = 1 \textbf{ to } (E_1 \ \mathrm{div} \ E_2) \textbf{ do } C \textbf{ endf}$$
dove $I \notin \mathcal{V}ar(C)$

Dove la funzione $\div$ è la funzione ricorsiva che calcola la divisione intera. Si noti che è lecito utilizzare funzioni primitive ricorsive nei nostri programmi in quanto per un teorema sui linguaggi FOR le funzioni primitive ricorsive sono tutte e sole le funzioni FOR-calcolabili, dunque sappiamo che esiste un programma che le calcola.

$$\textbf{for } I = E_1 \textbf{ to } E_2 \textbf{ do } C \textbf{ endf} \ \cong$$
$$I = E_1;\ \textbf{loop } E_2 - E_1 + 1 \textbf{ step } 1 \textbf{ do } C \textbf{ endl}$$

□

ESERCIZIO 3.4. *Eseguire l'Esercizio 3.2 sostituendo il comando* **loop** *con il seguente comando*

$$\textbf{loopmax}\ \mathsf{E}_1, \mathsf{E}_2\ \textbf{do}\ \mathsf{C}\ \textbf{endl}$$

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. Vediamo subito la semantica formale del nuovo comando; dobbiamo controllare i valori delle due espressioni, valutare quale è il maggiore ed eseguire il comando.

$$\frac{\mathcal{E}[\![\mathsf{E}_1]\!]\sigma = n_1, \quad \mathcal{E}[\![\mathsf{E}_2]\!]\sigma = n_2, \quad n_1 \geq n_2, \quad \langle \mathsf{C}; \textbf{loopmax}\ n_1 - 1, 0\ \textbf{do}\ \mathsf{C}\ \textbf{endl}, \sigma\rangle \to \sigma'}{\langle \textbf{loopmax}\ \mathsf{E}_1, \mathsf{E}_2\ \textbf{do}\ \mathsf{C}\ \textbf{endl}, \sigma\rangle \to \sigma'}$$

Abbiamo poi

$$\frac{\mathcal{E}[\![\mathsf{E}_1]\!]\sigma = n_1, \quad \mathcal{E}[\![\mathsf{E}_2]\!]\sigma = n_2, \quad n_1 < n_2, \quad \langle \mathsf{C}; \textbf{loopmax}\ 0, n_2 - 1\ \textbf{do}\ \mathsf{C}\ \textbf{endl}, \sigma\rangle \to \sigma'}{\langle \textbf{loopmax}\ \mathsf{E}_1, \mathsf{E}_2\ \textbf{do}\ \mathsf{C}\ \textbf{endl}, \sigma\rangle \to \sigma'}$$

Vediamo infine il caso base

$$\frac{\mathcal{E}[\![\mathsf{E}_1]\!]\sigma = n_1, \quad \mathcal{E}[\![\mathsf{E}_2]\!]\sigma = n_2, \quad n_1 = n_2 = 0}{\langle \textbf{loopmax}\ \mathsf{E}_1, \mathsf{E}_2\ \textbf{do}\ \mathsf{C}\ \textbf{endl}, \sigma\rangle \to \sigma}$$

Per quel che riguarda la LOOPMAX-calcolabilità la definizione è analoga a quella dell'Esercizio 3.2, e dunque non la riportiamo. Vediamo ora la dimostrazione che una funzione è LOOPMAX-calcolabile se e solo se è FOR-calcolabile.

$$\begin{aligned}&\textbf{loopmax}\ \mathsf{E}_1, \mathsf{E}_2\ \textbf{do}\ \mathsf{C}\ \textbf{endl}\ \cong\\ &\qquad \mathsf{E} := \mathit{max}(\mathsf{E}_1, \mathsf{E}_2);\ \textbf{for}\ \mathsf{I} := 1\ \textbf{to}\ \mathsf{E}\ \textbf{do}\ \mathsf{C}\ \textbf{endf}\\ &\text{dove } \mathsf{I} \notin \mathit{var}(\mathsf{C})\end{aligned}$$

In questo caso la funzione max è la funzione primitiva ricorsiva che calcola il massimo tra due funzioni definita nelle dispense. Vediamo ora il viceversa:

$$\begin{aligned}&\textbf{for}\ \mathsf{I} := \mathsf{E}_1\ \textbf{to}\ \mathsf{E}_2\ \textbf{do}\ \mathsf{C}\ \textbf{endf}\ \cong\\ &\mathsf{I} := \mathsf{E}_1;\ \textbf{loopmax}\ \mathsf{E}_2 - \mathsf{E}_1 + 1, 0\ \textbf{do}\ \mathsf{C}\ \textbf{endl}\end{aligned}$$

□

ESERCIZIO 3.5. *Svolgere l'Esercizio 3.2 sostituendo il comando* **loop** *con il seguente*

$$\textbf{alternate}\ \mathsf{E}\ \textbf{times}\ \mathsf{C}_1, \mathsf{C}_2\ \textbf{endalt}$$

con le seguenti possibili semantiche intuitive

(1) *Viene eseguito il comando* $\mathsf{C}_1; \mathsf{C}_2$ *tante volte quanto vale inizialmente* E*;*
(2) *Viene eseguito* C_1 E div 2 *volte e successivamente* C_2 E mod 2 *volte.*

SOLUZIONE.

(1) Vediamo subito la semantica formale del primo caso

$$\frac{\mathcal{E}[\![\mathsf{E}]\!]\sigma = n, \quad n > 0, \quad \langle \mathsf{C}_1; \mathsf{C}_2; \textbf{alternate}\ n - 1\ \textbf{times}\ \mathsf{C}_1, \mathsf{C}_2\ \textbf{endalt}, \sigma\rangle \to \sigma'}{\langle \textbf{alternate}\ \mathsf{E}\ \textbf{times}\ \mathsf{C}_1, \mathsf{C}_2\ \textbf{endalt}, \sigma\rangle \to \sigma'}$$

Quindi se il valore della guardia è maggiore di zero, vengono eseguiti i due comandi e nuovamente il comando **alternate** con la guardia decrementata.

$$\frac{\mathcal{E}[\![E]\!]\sigma = n, \quad n = 0}{\langle \textbf{alternate } E \textbf{ times } C_1, C_2 \textbf{ endalt}, \sigma\rangle \to \sigma}$$

Quando la guardia diventa nulla il comando lascia la memoria inalterata. La ALTERNATE-calcolabilità è analoga alla LOOP-calcolabilità, e per questo evitiamo di ripeterla. A questo punto si vuole dimostrare che una funzione è ALTERNATE-calcolabile se e solo se è FOR-calcolabile; per questo vediamo come possiamo simulare ciascun comando con l'altro.

$$\begin{aligned}&\textbf{alternate } E \textbf{ times } C_1, C_2 \textbf{ endalt} \;\cong\\ &\qquad\qquad \textbf{for } I := 1 \textbf{ to } E \textbf{ do } C_1; C_2 \textbf{ endf}\\ &\text{dove } I \notin var(C_1) \cup var(C_2)\end{aligned}$$

Mentre il viceversa

$$\begin{aligned}&\textbf{for } I := E_1 \textbf{ to } E_2 \textbf{ do } C \textbf{ endf} \;\cong\\ &I := E_1; \textbf{ alternate } E_2 - E_1 + 1 \textbf{ times } C, I := I + 1 \textbf{ endalt}\end{aligned}$$

(2) Anche in questo caso definiamo la semantica formale

$$\frac{\mathcal{E}[\![E]\!]\sigma = n, \quad n = 0}{\langle \textbf{alternate } E \textbf{ times } C_1, C_2 \textbf{ endalt}, \sigma\rangle \to \sigma}$$

Quindi se la divisione non è possibile il comando non fa nulla.

$$\frac{\mathcal{E}[\![E]\!]\sigma = n, \quad n = 1, \quad \langle C_2, \sigma\rangle \to \sigma'}{\langle \textbf{alternate } E \textbf{ times } C_1, C_2 \textbf{ endalt}, \sigma\rangle \to \sigma'}$$

Se il valore della guardia è 1 allora il quoziente è zero e quindi si può eseguire solo C_2 una volta sola.

$$\frac{\mathcal{E}[\![E]\!]\sigma = n, \quad n \geq 2, \quad \langle C_1; \textbf{alternate } n - 2 \textbf{ times } C_1, C_2 \textbf{ endalt}, \sigma\rangle \to \sigma'}{\langle \textbf{alternate } E \textbf{ times } C_1, C_2 \textbf{ endalt}, \sigma\rangle \to \sigma'}$$

Se il valore della guardia è maggiore o uguale a 2 eseguiamo C_1 tante volte quanto il 2 è contenuto in n e C_2 quanto il resto. Per quel che riguarda la ALTERNATE-calcolabilità essa è la stessa del caso precedente. Infine dimostriamo l'equivalenza del nuovo comando con il comando **for**.

$$\begin{aligned}&\textbf{alternate } E \textbf{ times } C_1, C_2 \textbf{ endalt} \;\cong\\ &\qquad\qquad \textbf{for } I := 1 \textbf{ to } E \operatorname{div} 2 \textbf{ do } C_1 \textbf{ endf};\\ &\qquad\qquad \textbf{for } I := 1 \textbf{ to } E \bmod 2 \textbf{ do } C_2 \textbf{ endf}\\ &\text{dove } I \notin var(C_1) \cup var(C_2)\end{aligned}$$

Mentre il viceversa

$$\begin{aligned}&\textbf{for } I := E_1 \textbf{ to } E_2 \textbf{ do } C \textbf{ endf} \;\cong\\ &I := E_1;\\ &\textbf{alternate } 2 * (E_2 - E_1 + 1) \textbf{ times } C; I := I + 1, \textbf{skip endalt}\end{aligned}$$

□

ESERCIZIO 3.6. *Eseguire l'Esercizio 3.2 sostituendo il comando* **loop** *con il seguente comando*

$$\textbf{alternate } \mathsf{E}_1 \textbf{ by } \mathsf{E}_2 \textbf{ do } \mathsf{C}_1, \mathsf{C}_2 \textbf{ endalt}$$

la cui semantica intuitiva consiste nell'eseguire C_1 *un numero di volte pari a* $\mathsf{E}_1 \mathrm{div} \mathsf{E}_2$ *e* C_2 $\mathsf{E}_1 \bmod \mathsf{E}_2$ *volte.*

SOLUZIONE. Definiamo inizialmente la semantica formale del nuovo comando calcolando la divisione e il modulo ed eseguendo i due comandi il corretto numero di volte

$$\frac{\mathcal{E}[\![\mathsf{E}_1]\!]\sigma = n_1, \quad \mathcal{E}[\![\mathsf{E}_2]\!]\sigma = n_2, \quad (n_1 = 0 \ \ \vee \ \ n_2 = 0)}{\langle \textbf{alternate } \mathsf{E}_1 \textbf{ by } \mathsf{E}_2 \textbf{ do } \mathsf{C}_1, \mathsf{C}_2 \textbf{ endalt}, \sigma\rangle \to \sigma}$$

Quindi se la divisione non è possibile il comando non fa nulla.

$$\frac{\begin{array}{c}\mathcal{E}[\![\mathsf{E}_1]\!]\sigma = n_1, \quad \mathcal{E}[\![\mathsf{E}_2]\!]\sigma = n_2, \quad n_1 \leq n_2, \\ \langle \mathsf{C}_2; \textbf{ alternate } n_1 - 1 \textbf{ by } n_2 \textbf{ do } \mathsf{C}_1, \mathsf{C}_2 \textbf{ endalt}, \sigma\rangle \to \sigma'\end{array}}{\langle \textbf{alternate } \mathsf{E}_1 \textbf{ by } \mathsf{E}_2 \textbf{ do } \mathsf{C}_1, \mathsf{C}_2 \textbf{ endalt}, \sigma\rangle \to \sigma'}$$

Cioè se il divisore è più piccolo del dividendo, n_1 risulta essere proprio il resto della divisione e quindi eseguiamo C_2 esattamente n_1 volte e zero volte l'altro comando.

$$\frac{\begin{array}{c}\mathcal{E}[\![\mathsf{E}_1]\!]\sigma = n_1, \quad \mathcal{E}[\![\mathsf{E}_2]\!]\sigma = n_2, \quad n_1 > n_2, \\ \langle \mathsf{C}_1; \textbf{ alternate } n_1 - n_2 \textbf{ by } n_2 \textbf{ do } \mathsf{C}_1, \mathsf{C}_2 \textbf{ endalt}, \sigma\rangle \to \sigma'\end{array}}{\langle \textbf{alternate } \mathsf{E}_1 \textbf{ by } \mathsf{E}_2 \textbf{ do } \mathsf{C}_1, \mathsf{C}_2 \textbf{ endalt}, \sigma\rangle \to \sigma'}$$

Infine se la divisione è possibile eseguiamo il comando C_1 il numero di volte in cui E_2 è contenuto in E_1. Per la similitudine con i casi già definiti, anche in questo caso non definiamo esplicitamente la ALTERNATE-BY-calcolabilità. Dimostriamo a questo punto l'equivalenza della ALTERNATE-BY-calcolabilità con la FOR-calcolabilità.

$$\begin{array}{l}\textbf{alternate } \mathsf{E}_1 \textbf{ by } \mathsf{E}_2 \textbf{ do } \mathsf{C}_1, \mathsf{C}_2 \textbf{ endalt} \ \cong \\ \qquad \textbf{for } \mathrm{I} := 1 \textbf{ to } \mathsf{E}_1 \textbf{ div } \mathsf{E}_2 \textbf{ do } \mathsf{C}_1 \textbf{ endf}; \\ \qquad \textbf{for } \mathrm{I} := 1 \textbf{ to } \mathsf{E}_1 \bmod \mathsf{E}_2 \textbf{ do } \mathsf{C}_2 \textbf{ endf} \\ \text{dove } \mathrm{I} \notin \mathcal{var}(\mathsf{C}_1) \cup \mathcal{var}(\mathsf{C}_2)\end{array}$$

Vediamo il viceversa

$$\begin{array}{l}\textbf{for } \mathrm{I} := \mathsf{E}_1 \textbf{ to } \mathsf{E}_2 \textbf{ do } \mathsf{C} \textbf{ endf} \ \cong \\ \mathrm{I} := \mathsf{E}_1; \\ \textbf{alternate } \mathsf{E}_2 - \mathsf{E}_1 + 1 \textbf{ by } 1 \textbf{ do } \mathsf{C}; \mathrm{I} := \mathrm{I} + 1, \textbf{skip endalt}\end{array}$$

□

ESERCIZIO 3.7. *Svolgere l'Esercizio 3.2 sostituendo il comando* **loop** *con il comando*

$$\textbf{rotate } \mathsf{C}_1, \mathsf{C}_2 \textbf{ time } \mathsf{E}_1 \textbf{ by } \mathsf{E}_2$$

la cui semantica intuitiva consiste nell'eseguire almeno una volta sia C_1 *che* C_2 *e successivamente eseguire* C_1 $\mathsf{E}_1 \mathrm{div} \mathsf{E}_2$ *volte e* C_2 $\mathsf{E}_1 \bmod \mathsf{E}_2$ *volte.*

SOLUZIONE. Definiamo subito la semantica formale del nuovo comando in cui dobbiamo valutare le guardie dopo aver eseguito una volta i due comandi.

$$\frac{\langle C_1; C_2, \sigma\rangle \to \sigma', \quad \mathcal{E}[\![E_1]\!]\sigma' = n_1, \quad \mathcal{E}[\![E_2]\!]\sigma' = n_2, \quad (n_1 = 0 \ \vee \ n_2 = 0)}{\langle \textbf{rotate } C_1, C_2 \textbf{ time } E_1 \textbf{ by } E_2, \sigma\rangle \to \sigma'}$$

Quindi se la divisione non è possibile dopo aver eseguito i due comandi non si esegue altro.

$$\frac{\begin{array}{c}\langle C_1; C_2, \sigma\rangle \to \sigma', \\ \mathcal{E}[\![E_1]\!]\sigma' = n_1, \quad \mathcal{E}[\![E_2]\!]\sigma' = n_2, \quad n_1 \geq n_2, \quad n_2 \neq 0, \\ \langle \textbf{rotate } C_1, \textbf{skip time } n_1 - n_2 \textbf{ by } n_2, \sigma'\rangle \to \sigma'', \\ \langle \textbf{rotate } C_2, \textbf{skip time } n_1 \bmod n_2 \textbf{ by } 1, \sigma''\rangle \to \sigma'''\end{array}}{\langle \textbf{rotate } C_1, C_2 \textbf{ time } E_1 \textbf{ by } E_2, \sigma\rangle \to \sigma'''}$$

Cioè se la divisione è possibile, nella memoria modificata dall'esecuzione dei due comandi, eseguo C_1 il numero di volte in cui E_2 è contenuto in E_1 e C_2 esattamente il resto della divisione.

$$\frac{\begin{array}{c}\langle C_1; C_2, \sigma\rangle \to \sigma', \\ \mathcal{E}[\![E_1]\!]\sigma' = n_1, \quad \mathcal{E}[\![E_2]\!]\sigma' = n_2, \quad n_1 < n_2, \\ \langle \textbf{rotate } C_2, \textbf{skip time } n_1 \bmod n_2 \textbf{ by } 1, \sigma'\rangle \to \sigma''\end{array}}{\langle \textbf{rotate } C_1, C_2 \textbf{ time } E_1 \textbf{ by } E_2, \sigma\rangle \to \sigma''}$$

La definizione di ROTATE-calcolablità è, come in precedenza, simile alle definizioni già fatte e dunque non viene riportata.

Vediamo la dimostrazione di equivalenza con il linguaggio FOR.

$$\begin{array}{l}\textbf{rotate } C_1, C_2 \textbf{ time } E_1 \textbf{ by } E_2 \ \cong \\ \qquad\qquad C_1;\ C_2; \\ \qquad\qquad \textbf{for } I := 1 \textbf{ to } E_1 \ div\ E_2 \textbf{ do } C_1 \textbf{ endf}; \\ \qquad\qquad \textbf{for } I := 1 \textbf{ to } E_1 \bmod E_2 \textbf{ do } C_2 \textbf{ endf} \\ dove\ I \notin var(C_1) \cup var(C_2)\end{array}$$

Mentre la dimostrazione inversa

$$\begin{array}{l}\textbf{for } I := E_1 \textbf{ to } E_2 \textbf{ do } C \textbf{ endf} \ \cong \\ \textbf{if } (E_1 > E_2) \textbf{ then skip } \textbf{ else} \\ I := E_1; \\ \textbf{rotate } C; I := I + 1, \textbf{skip time } E_2 - E_1 + 1 \textbf{ by } 1 \textbf{ fi}\end{array}$$

□

CAPITOLO 3

Teoria della ricorsività

1. Insiemi e funzioni ricorsive

In questa sezione vedremo una serie di esercizi riguardanti le proprietà di decidibilità e la struttura degli insiemi di numeri naturali. Un tipico esercizio di questa parte riguarda la dimostrazione matematica di semplici proprietà degli insiemi ricorsivi e ricorsivamente enumerabili. Per ora non faremo ricorso al Teorema di Rice che, come vedremo nella prossima sezione, ci permetterà di dimostrare in modo semplice la non ricorsività di una ampia varietà di insiemi.
Ricordiamo che un insieme di naturali $I \subseteq \mathbb{N}$ è detto *ricorsivamente enumerabile* (r.e.) se esiste una funzione parziale ricorsiva ψ tale che $I = dom(\psi)$. I è *ricorsivo* se esiste una funzione ricorsiva (totale) f_I tale che:

$$f_I(x) = \begin{cases} 1 & \text{se } x \in I \\ 0 & \text{se } x \notin I \end{cases}$$

Per dimostrare le seguenti proprietà degli insiemi ricorsivi e ricorsivamente enumerabili sarà necessario ricorrere a teoremi noti, in particolare il Teorema di Post e le sue conseguenze, il Teorema smn, il Teorema di Kleene di punto fisso ed alcuni risultati di caratterizzazione degli insiemi ricorsivi per i quali si rimanda alle dispense.

Esercizio 1.1. *Dimostrare che la cardinalità di un insieme* A *è minore di quella dei numeri naturali,* $|A| < \omega$*, se e solo se* $\forall B \subseteq A$ *si ha che* B *è ricorsivo.*

Soluzione. Essendo una dimostrazione con doppia implicazione distinguiamo i due diversi casi.

- ($\Rightarrow$) Questa implicazione è dimostrata immediatamente se consideriamo che ogni sottoinsieme di un insieme finito è finito e dunque ricorsivo.
- ($\Leftarrow$) In questo caso dobbiamo invece ragionare per assurdo, supponiamo che $|A| = \omega$, ovvero che A sia un insieme infinito la cui cardinalità è uguale a quella dei numeri naturali. Possiamo allora concludere che esiste una biiezione tra gli insiemi A e $\mathbb{N}$, ma per ipotesi ogni sottoinsieme di A è ricorsivo e, vista la biiezione, concludiamo che anche ogni sottoinsieme di $\mathbb{N}$ è ricorsivo, fatto che sappiamo essere assurdo.

□

Esercizio 1.2. *Siano* f *e* g *funzioni totali ricorsive di cui* g *iniettiva. Inoltre sia l'insieme* range*(g) ricorsivo, allora vale la seguente implicazione*

$$\forall x \in \mathbb{N} \,.\, f(x) \geq g(x) \quad \Rightarrow \quad \text{range}(f) \text{ è ricorsivo}$$

Soluzione. Per dimostrare che un insieme è ricorsivo dobbiamo fare in modo che la condizione di appartenenza all'insieme sia equivalente ad una condizione decidibile.

Per individuare tale condizione consideriamo l'insieme di valori $\{\ y \mid g(y) \leq x\ \}$ dipendente da x. Essendo g una funzione totale e iniettiva con RANGE(g) ricorsivo si ha che l'insieme di elementi che portano g in un insieme finito, $[0, x]$, non può mai essere infinito, quindi per ogni x l'insieme $\{\ y \mid g(y) \leq x\ \}$ è finito. Per questo motivo possiamo determinarne il massimo, definiamo allora $\overline{n}_x = max\ \{\ y \mid g(y) \leq x\ \}$ e consideriamo l'insieme discreto $\{f(0), f(1), \ldots, f(\overline{n}_x)\} \stackrel{\text{def}}{=} A_x$. Dimostriamo che $x \in \text{RANGE}(f)$ se e solo se $x \in A_x$.

- ($\Rightarrow$) Supponiamo che $x \in \text{RANGE}(f)$, allora dobbiamo dimostrare che $x \in A_x$. Supponiamo per assurdo che $x \notin A_x$ allora $\exists z > \overline{n}_x\ .\ x = f(z) \geq g(z) > x$, infatti sappiamo che $x \in \text{RANGE}(f)$ inoltre per ipotesi per ogni punto del dominio la funzione f in quel punto è maggiore della funzione g e infine l'ultima disuguaglianza è dovuta al fatto che x non sta in A_x. È quindi evidente che la disuguaglianza è assurda imponendo $x > x$, dunque l'ipotesi fatta è assurda perciò $x \in A_x$.
- ($\Leftarrow$) Banale per definizione di A_x.

Dunque l'appartenenza di un elemento all'insieme RANGE(f) risulta essere equivalente all'appartenenza dell'elemento ad un insieme finito e quindi è decidibile. Da cui la ricorsività dell'insieme. □

ESERCIZIO 1.3. *Dimostrare che l'insieme* $F = \{\ i \mid \varphi_i = \varphi_{f(i)}\ \}$, *con* f *funzione ricorsiva, non può mai essere finito.*

SOLUZIONE. Supponiamo per assurdo che esista f ricorsiva tale che l'insieme sia finito, ovvero tale che $|\{\ i \mid \varphi_i = \varphi_{f(i)}\ \}| < \omega$, allora esiste una funzione g ricorsiva tale che per ogni i nell'insieme F si ha che $g \neq \varphi_i$. Una tale funzione esiste perché le funzioni ricorsive sono infinite. Ora dobbiamo cercare di applicare i teoremi che conosciamo sugli insiemi ricorsivi in modo da dimostrare l'infinitezza dell'insieme F. Data la struttura di tale insieme l'idea è quella di definire, in funzione della sua finitezza, una funzione ricorsiva che conduca ad un assurdo. Dunque la chiave dell'esercizio sta ora nella scelta di tale funzione, per far sì che si verifichi un assurdo dobbiamo definire la funzione in modo che l'appartenenza di un indice j ad F implichi l'uguaglianza a g. Usiamo quindi il teorema smn per spostare il valore j nell'indice della MdT corrispondente, e quindi per ricadere nella struttura della definizione di F. Vediamo dunque come definire la funzione parametrica rispetto all'indice j e all'input x.

$$\psi(j, x) = \begin{cases} g(x) & \text{se } j \in F \\ \varphi_{f(j)}(x) & \text{se } j \notin F \end{cases}$$

Essendo l'insieme F ricorsivo, in quanto finito, la definizione è effettiva e dunque ψ è parziale ricorsiva (è parziale e non totale perché è definita in funzione di $\varphi_{f(j)}$ che è parziale). Possiamo ora applicare il teorema smn, quindi possiamo dire che esiste una funzione s ricorsiva tale che $\psi(j, x) = \varphi_{s(j)}(x)$. Infine per il teorema di ricorsione di Kleene sappiamo che esiste un indice i_0 tale per cui

$$\varphi_{i_0}(x) = \varphi_{s(i_0)}(x) = \begin{cases} g(x) & \text{se } i_0 \in F \\ \varphi_{f(i_0)}(x) & \text{se } i_0 \notin F \end{cases}$$

A questo punto siamo riusciti ad ottenere la funzione in cui l'indice di cui testiamo l'appartenenza ad F è l'indice della macchina di Turing corrispondente, e tale MdT è definita in modo che se $i_0 \notin F$ allora si ha che $\varphi_{i_0} = \varphi_{s(i_0)} = \varphi_{f(i_0)}$ e dunque per

definizione di F si ha che $i_0 \in F$ perciò si verifica un assurdo. Se invece $i_0 \in F$ allora $\varphi_{i_0} = \varphi_{s(i_0)} = g(x)$, ma anche questo caso è assurdo per definizione di g. Dunque visto che l'assunzione fatta porta solo a situazioni assurde essa è falsa e perciò F è sempre infinito. □

2. Teorema di Rice

In questa sezione vedremo alcuni esercizi riguardanti le proprietà di ricorsività e non ricorsività di insiemi di numeri naturali, o equivalentemente di insiemi di programmi in un dato linguaggio di programmazione. Il Teorema di Rice ci fornisce uno strumento molto potente per questo studio. Esso infatti permette di dimostrare la non ricorsività di una ampia gamma di insiemi di numeri (o programmi), ovvero di tutti gli insiemi di numeri o programmi che rappresentino proprietà estensionali di un dato formalismo di calcolo. Ribadiamo il fatto che, in questa sezione così come in tutto il corso, una proprietà sui programmi è rappresentata estensionalmente dall'insieme dei programmi che soddisfano la proprietà stessa. Per la Tesi di Church e per la aritmetizzazione dei modelli di calcolo, le proprietà di programmi e gli insiemi di numeri di naturali che rappresentano indici in una fissata numerazione di un dato modello di calcolo (indici di MdT, indici di funzioni parziali ricorsive, ecc.) sono nozioni perfettamente equivalenti. Pertanto, in entrambi i casi, il Teorema di Rice ci fornisce un strumento per valutare la ricorsività di tali insiemi. Chiamiamo W^+ l'insieme dei programmi **while** che possono ricevere come input altri programmi **while**.

Esercizio 2.1. *Siano* P *e* Q *due programmi appartenenti a* W^+*, dire se è decidibile la seguente affermazione:* $[\![P]\!] = [\![Q]\!]$.

Soluzione. Definiamo il seguente insieme

$$\Pi = \{\ P \mid [\![P]\!] = [\![Q]\!]\ \}$$

Allora notiamo che $\Pi \neq \emptyset$ perché il programma Q appartiene all'insieme. Inoltre $\Pi \neq \mathbb{N}$ perché non tutti i programmi calcolano le stesse cose. Infine Π è estensionale, infatti se $[\![P_1]\!] = [\![P_2]\!]$ e $P_2 \in \Pi$ allora si ha che $[\![P_2]\!] = [\![Q]\!]$ e per la proprietà transitiva dell'uguaglianza si verifica che $[\![P_1]\!] = [\![Q]\!]$ dunque $P_1 \in \Pi$. Per il teorema di Rice Π non è ricorsivo.

□

Esercizio 2.2. *Dimostrare che l'insieme* $\{\ x \mid |dom(\varphi_x)| = \omega\ \}$ *non è ricorsivo.*

Soluzione. Definiamo l'insieme

$$\Pi = \{\ x \mid |W_x| = \omega\ \}$$

dove sappiamo che $W_x = dom(\varphi_x)$. Notiamo che $\Pi \neq \emptyset$ perché la funzione $\lambda x.x$ appartiene all'insieme avendo per dominio tutto $\mathbb{N}$. Inoltre $\Pi \neq \mathbb{N}$ perché la funzione $\lambda x.\uparrow$ non appartiene all'insieme avendo per dominio l'insieme $\emptyset$. Infine Π è estensionale, infatti se $\varphi_x = \varphi_y$ e $y \in \Pi$ si ha che $|W_y| = \omega$ e per l'uguaglianza delle due funzioni si verifica anche che $W_x = W_y$ dunque anche W_x ha cardinalità infinita e questo implica che $x \in \Pi$. Per il teorema di Rice Π non è ricorsivo. □

ESERCIZIO 2.3. *Dimostrare che l'insieme* $\{\ x \mid W_x = \emptyset\ \}$ *non è ricorsivo.*

SOLUZIONE. Definiamo l'insieme

$$\Pi = \{\ x \mid W_x = \emptyset\ \}$$

Allora notiamo che $\Pi \neq \emptyset$ perché la funzione $\lambda x.\uparrow$ appartiene all'insieme avendo per dominio l'insieme $\emptyset$. Inoltre $\Pi \neq \mathbb{N}$ perché la funzione $\lambda x.x$ non appartiene all'insieme avendo per dominio l'insieme $\mathbb{N}$. Infine Π è estensionale, infatti se $\varphi_x = \varphi_y$ e $y \in \Pi$ si ha che $W_y = \emptyset$ e per l'uguaglianza delle due funzioni si verifica anche che $W_x = W_y$ dunque anche W_x è l'insieme vuoto e questo implica che $x \in \Pi$. Per il teorema di Rice Π non è ricorsivo. □

ESERCIZIO 2.4. *Dimostrare che l'insieme* $\{\ (x,y) \mid W_x = W_y\ \}$ *non è ricorsivo.*

SOLUZIONE. Definiamo l'insieme

$$\Pi = \{\ (x,y) \mid W_x = W_y\ \}$$

Allora notiamo che $\Pi \neq \emptyset$ perché la coppia (x,x) appartiene all'insieme per la proprietà riflessiva dell'uguaglianza di insiemi. Inoltre $\Pi \neq \mathbb{N}$ perché se $\varphi_x = \lambda x.x$ e $\varphi_y = \lambda x.\uparrow$, sicuramente la coppia (x,y) non appartiene all'insieme avendo, le funzioni, domini diversi. Infine Π è estensionale, infatti sia $\varphi_x = \varphi_y$ e quindi $W_x = W_y$, allora, fissato un indice i, se $(x,i) \in \Pi$ si ha che $W_x = W_i$ e per la proprietà transitiva dell'uguaglianza si verifica che $W_i = W_y$, questo implica che $(y,i) \in \Pi$. Per il teorema di Rice Π non è ricorsivo. □

ESERCIZIO 2.5. *Sia* $\sim \subseteq \mathsf{PR} \times \mathsf{PR}$ *una relazione definita sull'insieme delle funzioni parziali ricorsive (PR) tale che*

$$\forall f\,.\,(f,f) \in \sim$$

Definiamo gli insiemi

$$\mathsf{R}_\sim = \{\ (f,g) \mid f \sim g\ \}$$
$$\Omega\mathsf{R}_\sim = \{\ (i,j) \mid \varphi_i \sim \varphi_j\ \}$$

Dimostrare che $\Omega\mathsf{R}_\sim$ *è ricorsivo se e solo se* $\Omega\mathsf{R}_\sim = \emptyset$ *oppure* $\Omega\mathsf{R}_\sim = \mathbb{N} \times \mathbb{N}$.

SOLUZIONE. Vediamo distintamente le due implicazioni.

- ($\Leftarrow$) Ovvio.
- ($\Rightarrow$) Supponiamo che $\Omega\mathsf{R}_\sim \neq \emptyset$ e che $\Omega\mathsf{R}_\sim \neq \mathbb{N} \times \mathbb{N}$. Notiamo che se $\varphi_x = \varphi_y$ allora sono la stessa funzione e dunque per la definizione della relazione $\varphi_x \sim \varphi_y$. Fissiamo poi un indice i e supponiamo che $(x,i) \in \Omega\mathsf{R}_\sim$, allora $\varphi_x \sim \varphi_i$, però essendo φ_x e φ_y la stessa funzione possiamo dire che $\varphi_y \sim \varphi_i$ e quindi $(y,i) \in \Omega\mathsf{R}_\sim$. Perciò possiamo concludere che il nostro insieme è estensionale e quindi $\Omega\mathsf{R}_\sim$ non è ricorsivo per il teorema di Rice. Ma questo è assurdo essendo $\Omega\mathsf{R}_\sim$ ricorsivo per ipotesi.

□

ESERCIZIO 2.6. *Dimostrare che l'insieme*

$$\{\ x \mid \varphi_x \textit{ definitivamente } \geq 3\ \}$$

non è ricorsivo. Si ricorda che definitivamente maggiore di un numero significa che esiste un valore di input dopo il quale la funzione è sempre maggiore del numero dato.

SOLUZIONE. Definiamo l'insieme

$$\Pi = \{\ x \mid\ \varphi_x \text{ definitivamente } \geq 3\ \}$$

Notiamo che $\Pi \neq \emptyset$ in quanto la funzione $\lambda x.x$ appartiene all'insieme essendo maggiore o uguale a 3 per ogni $x \geq 3$. Inoltre $\Pi \neq \mathbb{N}$ perché la funzione $\lambda x.0$ sicuramente non appartiene all'insieme essendo costantemente uguale a 0 e quindi minore di 3. Infine Π è estensionale, infatti se $\varphi_x = \varphi_y$ si ha anche che $\text{RANGE}(\varphi_x) = \text{RANGE}(\varphi_y)$, quindi se $x \in \Pi$ φ_x è definitivamente maggiore o uguale a 3 e quindi anche φ_y è definitivamente maggiore o uguale a 3, questo implica che $y \in \Pi$. Per il teorema di Rice Π non è ricorsivo. □

ESERCIZIO 2.7. *Dimostrare che l'insieme* $\{\ i \mid\ \exists x\,.\,\varphi_i(x) = x\ \}$ *non è ricorsivo.*

SOLUZIONE. Definiamo l'insieme

$$\Pi = \{\ i \mid\ \exists x\,.\,\varphi_i(x) = x\ \}$$

Notiamo che $\Pi \neq \emptyset$ perché se $\varphi_y = \lambda x.x$ allora sicuramente $y \in \Pi$ in quanto in ogni punto la funzione vale quanto l'input a cui è applicata. Inoltre $\Pi \neq \mathbb{N}$ perché se $\varphi_y = \lambda x.\uparrow$ allora sicuramente $y \notin \Pi$ perché tale funzione diverge sempre e quindi per nessun input essa può valere quanto l'input stesso. Infine Π è estensionale, infatti se $\varphi_i = \varphi_j$ e $i \in \Pi$ allora $\exists x\,.\,\varphi_i(x) = x$ ma essendo φ_i e φ_j uguali si ha che $\varphi_j(x) = \varphi_i(x) = x$ quindi $j \in \Pi$. Per il teorema di Rice Π non è ricorsivo. □

3. Ricorsività di insiemi

In questa sezione applichiamo alcuni risultati utilizzati nelle sezioni precedenti, in particolare i teoremi di Post, di Rice, di ricorsività di Kleene, per studiare la ricorsività di alcuni insiemi non banali di naturali. Il metodo per risolvere questi esercizi è quello solito: ricondurre la definizione dell'insieme da studiare ad un caso noto, per il quale esiste un risultato che ne caratterizzi la ricorsività. Questo procedimento di dimostrazione sarà formalizzato e studiato più in dettaglio, per insiemi complessi, nella prossima sezione sulla *riducibilità funzionale*.

ESERCIZIO 3.1. *Dato* $A \subseteq \mathbb{N}$*, studiare la ricorsività, al variare di* A*, dell'insieme* $\{\ i \mid\ \forall x\,.\,\varphi_i(x)\downarrow \Leftrightarrow\ x \in A\ \}$.

SOLUZIONE. Ridefiniamo equivalentemente l'insieme dato come

$$\Pi = \{\ i \mid\ W_i = A\ \}$$

Tale insieme è estensionale perché se $i \in \Pi$ e $\varphi_i = \varphi_j$ allora $W_i = W_j$, dunque se $W_i = A$ anche $W_j = A$ per transitività dell'uguaglianza, e quindi $j \in \Pi$. Per il teorema di Rice risulta quindi che Π è ricorsivo solo se $\Pi = \mathbb{N}$ oppure $\Pi = \emptyset$; ma $\Pi \neq \mathbb{N}$ perché non tutte le funzioni possono avere lo stesso dominio. Allora Π è ricorsivo solo quando è vuoto, cioè quando non esiste alcun indice i per cui $W_i = A$. Poiché per ogni i si ha che W_i è ricorsivamente enumerabile allora gli insiemi A che

rendono $\Pi = \emptyset$ sono tutti quelli che non sono ricorsivamente enumerabili. □

ESERCIZIO 3.2. *Studiare la ricorsività dell'insieme* $\{\ i \mid\ \exists y\ .\ \varphi_i(y) = \varphi_y(i)\ \}$.

SOLUZIONE. La risoluzione di tale esercizio è immediata se notiamo che

$$\forall i \exists y = i\ .\ \varphi_i(y) = \varphi_y(i)$$

dunque tale insieme è esattamente l'insieme dei numeri naturali $\mathbb{N}$ e per questo è ricorsivo. □

ESERCIZIO 3.3. *Studiare la ricorsività dell'insieme* $\{\ x \mid\ \exists n \exists p\ \textit{primo}\ .\ x = p^n\ \}$.

SOLUZIONE. Abbiamo visto nel Capitolo 2 che esistono funzioni totali che generano numeri primi e che decidono se un numero x è potenza di un numero p. Dunque per tale insieme riusciamo a costruire una funzione caratteristica totale e ricorsiva perciò esso è ricorsivo. □

ESERCIZIO 3.4. *Sia l'insieme* D *il range della funzione*

$$\psi = \begin{cases} \psi(0) = n_0 \\ \psi(n+1) = \varphi_n(n) * \psi(n) \end{cases}$$

Dire se l'insieme D *è ricorsivo.*

SOLUZIONE. Vediamo di studiare la cardinalità dell'insieme D. Consideriamo l'insieme $K = \{\ x \mid\ \varphi_x(x) \downarrow\ \}$, poiché esso è diverso da $\mathbb{N}$ sicuramente esiste m_0 tale che $\varphi_{m_0}(m_0) \uparrow$ (senza perdere di generalità supponiamo che m_0 sia il più piccolo punto fuori da K). A questo punto dimostriamo per induzione che vale che $\forall m > m_0\ .\ \psi(m) \uparrow$:

- BASE: Prendiamo $m = m_0 + 1$. Allora $\psi(m_0 + 1) = \varphi_{m_0}(m_0) * \psi(m_0) = \uparrow * c = \uparrow$, dove $c = \psi(m_0)$ è un valore definito in quanto abbiamo supposto che m_0 sia il primo valore che incontriamo non appartenente a K e quindi $\varphi_{m_0-1}(m_0 - 1)$ è definito.
- PASSO INDUTTIVO: Supponiamo che, preso m_1, si abbia $\forall m \leq m_1\ .\ \psi(m) \uparrow$, dimostriamo che questo succede anche in $m_1 + 1$. Infatti è facile verificare che $\psi(m_1 + 1) = \varphi_{m_1}(m_1) * \psi(m_1) = c * \uparrow = \uparrow$ dove $c = \varphi_{m_1}(m_1)$ è un valore qualunque, potenzialmente indefinito, che non conosciamo, mentre $\psi(m_1)$ diverge per ipotesi induttiva.

Possiamo concludere che $D = \{\psi(0), \psi(1), \ldots, \psi(m_0)\}$, perciò D è finito e per questo ricorsivo. □

ESERCIZIO 3.5. *Definiamo la relazione* $\sim_w$ *come*

$$\varphi_i \sim_w \varphi_j \ \Leftrightarrow\ \forall x\,(\varphi_i(x) \downarrow\ \wedge\ \varphi_j(x) \downarrow \Rightarrow\ \ \varphi_i(x) = \varphi_j(x))$$

Tale tipo equivalenza è detta eauivalenza debole. Dire se l'insieme

$$B = \{\ k_i \mid\ k_i = \mu y.\ \varphi_i \sim_w \varphi_y\ \ \}$$

è ricorsivo.

SOLUZIONE. Notiamo subito che se in un'implicazione la premessa è falsa e dunque l'intera implicazione è vera, per questo motivo ogni funzione è debolmente equivalente alla funzione $\lambda x.\uparrow$. Sia $\varphi_z = \lambda x.\uparrow$ allora $B \subseteq \{0, 1, \ldots, z\}$ perché se $\varphi_i \sim_w \varphi_j$ e $j < z$ allora $k_i = j$ in quanto nella ricerca del più piccolo indice di funzione debolmente equivalente a φ_i troviamo prima j. Altrimenti è immediato verificare che per ogni i maggiore di z si ha $\varphi_i \sim_w \varphi_z$ e quindi nella ricerca troviamo sempre prima z. Perciò B è finito e dunque ricorsivo. □

ESERCIZIO 3.6. *Studiare la ricorsività di* $A = \text{RANGE}(f)$*, dove* f *è la funzione definita*

$$f = \begin{cases} f(0) = n_0 \\ f(n+1) = 2\, f(n) \end{cases}$$

SOLUZIONE. Consideriamo due casi; se $n_0 = 0$, allora il range della funzione si riduce banalmente al solo singoletto $\{0\}$ e quindi è ricorsivo. Supponiamo allora $n_0 \neq 0$, ovvero $n_0 > 0$. Vista la definizione di f, proviamo a dimostrare che f è una funzione crescente, infatti

$$f(n+1) = 2\, f(n) > f(n)$$

Questo vale se $\forall n \,.\, n \neq 0$, ed $f(0) = n_0 > 0$ per ipotesi, perciò A è il range di una funzione crescente e dunque è ricorsivo per un teorema sugli insiemi ricorsivi. □

ESERCIZIO 3.7. *Studiare la ricorsività di* $B = \text{RANGE}(f)$ *al variare della funzione* h *totale ricorsiva, con* f *definita come*

$$f = \begin{cases} f(0) = n_0 \\ f(n+1) = h(n) f(n) \end{cases}$$

SOLUZIONE. Distinguiamo i casi in base al valore della funzione h. Se $\forall n \in \mathbb{N} \,.\, h(n) = 0$ la nostra funzione f si riduce a valere n_0 in 0 e 0 in tutti gli altri punti; allora $B = \{0, n_0\}$ e quindi è ricorsivo. Se $\exists n_1 \,.\, h(n_1) = 0$ allora $\forall n > n_1 \,.\, f(n) = 0$. Infatti se $n = n_1 + 1$ allora $f(n_1 + 1) = h(n_1) f(n_1) = 0\, f(n_1) = 0$, se poi supponiamo che $\forall n \leq m \,.\, f(n) = 0$ allora $f(m+1) = h(m) f(m) = h(m)\, 0 = 0$. Questo fatto implica che $B \subseteq \{f(0), f(1), \ldots, f(n_1 + 1)\}$, perciò B è finito e dunque ricorsivo. Se, infine, $\forall n \in \mathbb{N} \,.\, h(n) \neq 0$ dimostriamo che f è crescente, perché questo implicherebbe che il suo range è ricorsivo. Supponiamo che $n_0 \neq 0$ (se n_0 fosse uguale a 0 allora $B = \{0\}$, e quindi sarebbe banalmente ricorsivo), dimostriamo per induzione che $\forall n \in \mathbb{N} \,.\, f(n+1) > f(n)$. Infatti $f(0) \neq 0$ per l'ipotesi appena fatta, supponiamo che $\forall n \leq m \,.\, f(n) > f(n-1)$ vediamo cosa succede per $m+1$: $f(m+1) = h(m) f(m) = c\, f(m) > f(m)$ essendo $c = h(m) \neq 0$ per ipotesi. Quindi si è dimostrato che B è ricorsivo sempre, indipendentemente dalla funzione h. □

ESERCIZIO 3.8. *Studiare la ricorsività del seguente insieme* $\Pi = \left\{ i \,\middle|\, W_i = \emptyset \right\}$.

SOLUZIONE. Vediamo subito se possiamo dire qualcosa sulla ricorsività applicando il teorema di Rice. Notiamo che $\Pi \neq \emptyset$ in quanto sicuramente l'indice della funzione $\lambda x.\uparrow$ sta nell'insieme. Inoltre $\Pi \neq \mathbb{N}$ in quano sicuramente l'indice della funzione $\lambda x.x$ non sta nell'insieme. Infine tale insieme è estensionale, poiché se due funzioni sono uguali lo sono anche i loro domini. Questo implica che l'insieme non è ricorsivo.

Dobbiamo allora vedere se Π è r.e. o meno; supponiamo che lo sia, supponiamo cioè che $\exists y_0 \,.\, W_{y_0} = \Pi$. Allora notiamo che

$$\begin{aligned} y_0 \in W_{y_0} \quad &\Leftrightarrow \quad y_0 \in \Pi \;\Leftrightarrow\; y_0 \in \left\{ \, i \,\middle|\, \forall x \,.\, \varphi_i(x) \uparrow \, \right\} \\ &\Leftrightarrow \quad \forall x \,.\, \varphi_{y_0}(x) \uparrow \Rightarrow \; \varphi_{y_0}(y_0) \uparrow \Leftrightarrow \; y_0 \notin W_{y_0} \end{aligned}$$

Da cui si ha un assurdo, perciò possiamo concludere che tale insieme non è ricosivamente enumerabile. □

ESERCIZIO 3.9. *Sia* f *una funzione ricorsiva totale e sia* $x \in \mathbb{N}$*, dimostrare che l'insieme* $f^{-1}(W_x) = \left\{ \, y \,\middle|\, f(y) \in W_x \, \right\}$ *è ricorsivamente enumerabile.*

SOLUZIONE. Per dimostrare che un insieme è r.e. è sufficiente dimostrare che esso è uguale al range di una funzione ricorsiva parziale. Per far ciò dobbiamo costruire una funzione che termini sui valori per cui la condizione di appartenenza all'insieme vale. Ricordiamo che $W_x = \left\{ \, y \,\middle|\, \varphi_x(y) \downarrow \, \right\}$ e definiamo la funzione

$$\psi(y) = \begin{cases} y & \text{se } f(y) \in W_x \\ \uparrow & \text{altrimenti} \end{cases}$$

Allora ψ è parziale ricorsiva essendo W_x r.e. per definizione; dunque, essendo dominio e codominio di ψ uguali si ha che $\text{RANGE}(\psi)$ è ricorsivamente enumerabile ma notiamo che vale l'uguaglianza $f^{-1}(W_x) = \text{RANGE}(\psi)$, dunque $f^{-1}(W_x)$ è ricorsivamente enumerabile. □

4. Riducibilità funzionale

In questa sezione tratteremo alcuni esercizi sulla riducibilità funzionale. Il concetto di riducibilità funzionale permette di ricondurre, mediante un legame formalmente definito da una funzione ricorsiva, la ricorsività di un insieme alla ricorsività di un altro insieme. La tecnica standard per studiare la ricorsività di un insieme mediante riducibilità funzionale consiste quindi nel definire una riduzione funzionale tra l'insieme da studiare e un insieme la cui ricorsività o ricorsiva enumerabilità è nota. Ricordiamo che dati $A, B \subseteq \mathbb{N}$, A è (funzionalmente) riducibile a B, denotato con $A \preceq B$, se esiste una funzione (totale) ricorsiva f tale che

$$x \in A \quad \Leftrightarrow \quad f(x) \in B$$

Si osserva che $\preceq$ definisce una equivalenza sugli insiemi di naturali: $A \equiv B$ sse $A \preceq B$ e $B \preceq A$. Ricordiamo inoltre i risultati principali riguardanti la ricorsività di insiemi funzionalmente riducibili. Siano $A, B \subseteq \mathbb{N}$:

- $A \preceq B \quad \Rightarrow \quad \overline{A} \preceq \overline{B}$
- $A \preceq B$ e $B \in \mathsf{RE} \quad \Rightarrow \quad A \in \mathsf{RE}$
- $A \preceq B$ e B ricorsivo $\Rightarrow$ A ricorsivo
- Per ogni $A \in \mathsf{RE}$: $A \preceq K = \{x \mid \varphi_x(x) \downarrow\}$

ESERCIZIO 4.1. *Siano* A *e* B *due insiemi tali che* $A \in [B]_\equiv$ *e sia* A *ricorsivo allora si dimostri che* $\forall X \in [B]_\equiv \,.\, X$ *è ricorsivo.*

SOLUZIONE. Notiamo subito che se $A \in [B]_\equiv$ allora $A \equiv B$ per definizione di classe di equivalenza, ma questo implica che $A \preceq B$ e $B \preceq A$. Allora per un teorema sulla riducibilità funzionale il fatto che A sia ricorsivo implica che anche B sia ricorsivo. Analogamente, quindi, $\forall X \in [B]_\equiv$ si ha che $X \equiv B$ e, in particolare, $X \preceq B$ allora la

ricorsività di B implica quella di X. □

ESERCIZIO 4.2. *Sia* A *un insieme ricorsivamente enumerabile, diverso da* $\emptyset$ *e* $\mathbb{N}$ *allora si dimostri che* A *è ricorsivo se e solo se* $A \preceq \overline{A}$.

SOLUZIONE. Dimostriamo distintamente le due implicazioni.

($\Rightarrow$) Supponiamo A ricorsivo e non banale, allora anche $\overline{A}$ è non banale e per il teorema di Post $\overline{A}$ è ricorsivamente enumerabile. Le seguenti affermazioni valgono per i teoremi sugli insiemi r.e. e ricorsivi. Poiché A è ricorsivo si ha che esiste una funzione f_1 ricorsiva crescente tale che $\text{RANGE}(f_1) = A$. Inoltre poiché $\overline{A}$ è r.e. allora si ha che esiste una funzione f_2 ricorsiva tale che $\text{RANGE}(f_2) = \overline{A}$. Dobbiamo ora definire una funzione che all'appartenenza ad A di un valore x associ un valore che sicuramente non appartiene ad A, ovvero un valore di $\overline{A}$. Un tale valore ci è fornito proprio dalla funzione f_2. Analogo per gli elementi di $\overline{A}$. Dunque definiamo la seguente funzione

$$g(x) = \begin{cases} f_1(x) & \text{se } x \notin A \\ f_2(x) & \text{se } x \in A \end{cases}$$

Poiché A è ricorsivo questa è una funzione totale perciò valgono le seguenti implicazioni:

$$\begin{aligned} x \in A &\Rightarrow g(x) = f_2(x) \in \overline{A} \\ x \notin A &\Rightarrow g(x) = f_1(x) \in A \Rightarrow g(x) \notin \overline{A} \end{aligned}$$

Perciò per definizione di riducibilità funzionale si ha che $A \preceq \overline{A}$.

($\Leftarrow$) Supponiamo che A sia r.e. e $A \preceq \overline{A}$, se riusciamo a ridurre $\overline{A}$ a K, allora $\overline{A}$ sarebbe ricorsivamente enumerabile e, dunque, per il teorema di Post A sarebbe ricorsivo. Esplicitando le ipotesi abbiamo che esiste una funzione f tale che $x \in A \Leftrightarrow f(x) \in \overline{A}$ e inoltre il fatto che A sia r.e. implica che $A \preceq K$, cioè che esiste una funzione g tale che $x \in A \Leftrightarrow g(x) \in K$. Allora valgono le seguenti implicazioni:

$$\begin{aligned} x \in \overline{A} &\Leftrightarrow x \notin A \Leftrightarrow f(x) \notin \overline{A} \Leftrightarrow f(x) \in A \Leftrightarrow g(f(x)) \in K \\ x \notin \overline{A} &\Leftrightarrow x \in A \Leftrightarrow f(x) \in \overline{A} \Leftrightarrow f(x) \notin A \Leftrightarrow g(f(x)) \notin K \end{aligned}$$

Perciò $\overline{A} \preceq K$ mediante la funzione $h = g \circ f$ e quindi A è ricorsivo.

□

ESERCIZIO 4.3. *Siano* A *e* B *due insiemi ricorsivi non banali, dimostrare che* $A \equiv B$.

SOLUZIONE. Vediamo di esplicitare le ipotesi date. Se A è ricorsivo allora esiste una funzione f_A ricorsiva totale crescente tale che $\text{RANGE}(f_A) = A$, inoltre se A è ricorsivo, allora per il teorema di Post si ha che $\overline{A}$ è r.e. quindi esiste una funzione totale ricorsiva $f_{\overline{A}}$ tale che $\text{RANGE}(f_{\overline{A}}) = \overline{A}$. D'altra parte se B è ricorsivo allora esiste una funzione f_B ricorsiva totale crescente tale che $\text{RANGE}(f_B) = B$, inoltre analogamente ad A, esitste una funzione totale ricorsiva $f_{\overline{B}}$ tale che $\text{RANGE}(f_{\overline{B}}) = \overline{B}$. A partire da questi fatti possiamo costruire una funzione che all'appartenenza di

una oggetto ad A faccia corrispondere l'appartenenza di una funzione di tale oggetto a B, e viceversa. Definiamo dunque le seguenti funzioni.

$$g_A(x) = \begin{cases} f_B(x) & \text{se } x \in A \\ f_{\overline{B}}(x) & \text{se } x \notin A \end{cases}$$

$$g_B(x) = \begin{cases} f_A(x) & \text{se } x \in B \\ f_{\overline{A}}(x) & \text{se } x \notin B \end{cases}$$

Perciò valgono le seguenti implicazioni:

$$\begin{array}{l} x \in A \;\Leftrightarrow\; g_A(x) = f_B(x) \in B \;\Rightarrow\; A \preceq B \\ x \in B \;\Leftrightarrow\; g_B(x) = f_A(x) \in A \;\Rightarrow\; B \preceq A \end{array}$$

Quindi abbiamo che $A \equiv B$. □

Esercizio 4.4. *Dimostrare che vale la seguente equivalenza*

$$A \stackrel{def}{=} \left\{ x \,\middle|\, x \textit{ numero primo } \right\} \equiv \left\{ x \,\middle|\, x \textit{ numero pari } \right\} \stackrel{def}{=} B$$

Soluzione. Per dimostrare l'equivalenza dobbiamo ridurre A a B e viceversa. Vediamo di fare questi due passi distintamente.

- Per dimostrare che $A \preceq B$ dobbiamo trovare una funzione f tale che $x \in A \;\Leftrightarrow\; f(x) \in B$. Questo significa che dobbiamo costruire una funzione che ad un elemento primo associa un elemento pari e che ad un elemento non primo associa un elemento dispari. Definiamo dunque tale funzione come
$$f(x) = \begin{cases} 2 & \text{se } x \text{ è primo} \\ 1 & \text{altrimenti} \end{cases}$$
Allora valgono le seguenti implicazioni:
$$\begin{array}{lll} x \in A & \Rightarrow & x \text{ primo } \Rightarrow\; f(x) = 2 \text{ (pari) } \Rightarrow\; f(x) \in B \\ x \notin A & \Rightarrow & x \text{ non primo } \Rightarrow\; f(x) = 1 \\ & \Rightarrow & f(x) \notin B \end{array}$$
Dunque A è riducibile a B mediante f.
- Per dimostrare che $B \preceq A$ dobbiamo trovare f tale che $x \in B \;\Leftrightarrow\; f(x) \in A$. In tal caso dobbiamo costruire una funzione che dato un x pari restituisce un numero primo, mentre dato un x dispari restituisce un numero non primo. Definiamo dunque la funzione
$$f(x) = \begin{cases} 3 & \text{se } x \text{ è pari} \\ 4 & \text{altrimenti} \end{cases}$$
Allora valgono le seguenti implicazioni:
$$\begin{array}{lll} x \notin B & \Rightarrow & x \text{ dispari } \Rightarrow\; f(x) = 4 \notin A \\ x \in B & \Rightarrow & x \text{ pari } \Rightarrow\; f(x) = 3 \in A \end{array}$$
Dunque B è riducibile ad A mediante f.

Questo esercizio poteva anche essere risolto banalmente notando che entrambi gli insiemi sono ricorsivi (per entrambi nella Sez. 2 del Cap. 2 esiste una funzione primitiva ricorsiva che li calcola) e quindi sono equivalenti per quanto dimostrato nell'Esercizio 4.3. □

ESERCIZIO 4.5. *Dimostrare che l'insieme* $A = \{ x \mid W_x \neq \emptyset \}$ *è ricorsivamente enumerabile.*

SOLUZIONE. Per dimostrare che l'insieme A è r.e. è sufficiente dimostrare che $A \preceq K$. Sappiamo che il teorema smn permette di spostare un input di una funzione nell'indice della MdT che la calcola. Questo ci permette di definire funzioni in cui le condizioni sul primo input diventano condizioni sull'indice della macchina. Ciò è importante perché l'insieme che dobbiamo studiare è un insieme di indici di MdT. L'idea è quella di definire una funzione $\psi(x, y)$ e di utilizzare il teorema smn in modo da ottenere $\psi(x, y) = \varphi_{g(x)}(y)$ per qualche funzione ricorsiva g (che sappiamo esistere per il teorema). A partire da questo cerchiamo di definire tale funzione ψ in modo che si abbia $x \in A \Leftrightarrow g(x) \in K$, usando quindi tale g per eseguire la riduzione. Allora se $x \in A$, ovvero se il dominio di φ_x non è vuoto, vogliamo che $g(x) \in K$, ovvero che $\psi(x, g(x)) = \varphi_{g(x)}(g(x)) \downarrow$. Per ottenere ciò è sufficiente fare in modo che $\psi(x, y) = \varphi_{g(x)}(y)$ termini su ogni possibile input y (e quindi anche su $g(x)$) ogni qualvolta esiste almeno un valore su cui φ_x termina. Notiamo che determinare tale esistenza è un problema semidecidibile grazie alla tecnica del **dovetail**, e dunque può essere utilizzata per definire una funzione parziale ricorsiva. Quando invece $x \notin A$, ovvero quando W_x è vuoto allora è sufficiente far divergere sempre la funzione, in tal modo siamo sicuri che essa non terminerà su $g(x)$ e quindi $g(x) \notin K$. Definiamo dunque la seguente funzione con due parametri uno dei quali verrà trasferito nell'indice della funzione φ grazie al teorema smn.

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

Per quanto detto si ha $\varphi_{g(x)}(y) = \psi(x, y)$ e quindi valgono le seguenti implicazioni

$$\begin{array}{rcl} x \in A & \Rightarrow & W_x \neq \emptyset \Rightarrow \exists z \,.\, \varphi_x(z) \downarrow \Rightarrow \forall y \,.\, \varphi_{g(x)}(y) \downarrow \\ & \Rightarrow & \varphi_{g(x)}(g(x)) \downarrow \Rightarrow g(x) \in K \\ x \notin A & \Rightarrow & W_x = \emptyset \Rightarrow \forall z \,.\, \varphi_x(z) \uparrow \Rightarrow \forall y \,.\, \varphi_{g(x)}(y) \uparrow \\ & \Rightarrow & \varphi_{g(x)}(g(x)) \uparrow \Rightarrow g(x) \notin K \end{array}$$

Perciò $A \preceq K$ mediante g, ovvero A è ricorsivamente enumerabile. □

ESERCIZIO 4.6. *Dimostrare che l'insieme* $A = \{ x \mid |W_x| < \omega \}$ *non è ricorsivamente enumerabile.*

SOLUZIONE. Ragioniamo per assurdo. Supponiamo che l'insieme A sia ricorsivamente enumerabile, consideriamo l'insieme $\overline{A} = \{ x \mid |W_x| = \omega \}$ e vediamo se a partire dalle ipotesi fatte anche questo insieme è ricorsivamente enumerabile. Per fare questo cerchiamo di ridurre l'insieme $\overline{A}$ ad A. Quindi, come abbiamo fatto nell'esercizio precedente, definiamo una funzione $\psi(x, y)$ che, grazie al teorema smn, è uguale a $\varphi_{g(x)}(y)$ per qualche funzione totale ricorsiva g. A questo punto vogliamo definire ψ in modo tale che $x \in \overline{A} \Leftrightarrow g(x) \in A$. Per ottenere ciò definiamo ψ in modo che quando $x \notin \overline{A}$ ($x \in A$) essa termini sempre, e quindi abbia il dominio infinito. Invece quando $x \in \overline{A}$ ($x \notin A$) facciamo divergere la funzione in modo da avere il dominio vuoto, e quindi finito.

$$\psi(x, y) = \begin{cases} y & \text{se } x \in A \\ \uparrow & \text{altrimenti} \end{cases}$$

Allora ψ è parziale ricorsiva in quanto A è per ipotesi ricorsivamente enumerabile. Per il teorema smn valgono le seguenti implicazioni:

$$\begin{array}{rcl} x \notin \overline{A} & \Rightarrow & x \in A \Rightarrow \forall y \,.\, \varphi_{g(x)}(y) = y \Rightarrow \forall y \,.\, \varphi_{g(x)} \downarrow \\ & \Rightarrow & |W_{g(x)}| = \omega \Rightarrow g(x) \in \overline{A} \Rightarrow g(x) \notin A \\ x \in \overline{A} & \Rightarrow & x \notin A \Rightarrow \forall y \,.\, \varphi_{g(x)}(y) \uparrow \Rightarrow |W_{g(x)}| = 0 < \omega \\ & \Rightarrow & g(x) \notin \overline{A} \Rightarrow g(x) \in A \end{array}$$

Abbiamo quindi dimostrato che $\overline{A} \preceq A$, ed essendo A r.e. allora lo è anche $\overline{A}$ e questo implica, per il teorema di Post, che A è ricorsivo. Possiamo però notare che $A \neq \emptyset$ poiché l'indice della funzione che diverge sempre appartiene sicuramente ad A. Inoltre $A \neq \mathbb{N}$ in quanto l'indice della funzione identità non può appartenere ad A. Infine A è estensionale poiché funzioni uguali hanno uguale dominio. Allora per il teorema di Rice A non è ricorsivo e quindi abbiamo un assurdo che deriva dall'ipotesi che A fosse ricorsivamente enumerabile. □

ESERCIZIO 4.7. *Dimostrare che se l'insieme* A *è ricorsivo e* $B = \{\, x \mid |W_x| < \omega \,\}$ *oppure* $B = \{\, x \mid W_x = \emptyset \,\}$ *allora* $A \preceq B$.

SOLUZIONE. Dobbiamo dimostrare che $A \preceq B$ e, come negli esercizi precedenti, usiamo il teorema smn. Perciò definiamo una funzione $\psi(x, y)$ che grazie a tale teorema è uguale alla funzione $\varphi_{g(x)}(y)$ per qualche funzione totale ricorsiva g. A questo punto utilizziamo tale g per effettuare la riduzione funzionale, ovvero definiamo ψ in modo che $x \in A \Leftrightarrow g(x) \in B$. Questo significa che la funzione deve essere tale che quando $x \in A$ allora $|W_{g(x)}| < \omega$ (oppure $W_{g(x)} = \emptyset$) e quindi è sufficiente far divergere la funzione ψ. Quando invece $x \notin A$ vogliamo che $|W_{g(x)}| = \omega$ (oppure $W_{g(x)} \neq \emptyset$) e quindi facciamo terminare la funzione su ogni input. Notiamo che questa condizione di non appartenenza ad A può essere utilizzata per definire una funzione parziale ricorsiva in quanto A è ricorsivo per ipotesi. Definiamo dunque la funzione

$$\psi(x, y) = \begin{cases} y & \text{se } x \notin A \\ \uparrow & \text{altrimenti} \end{cases}$$

Per il teorema smn abbiamo che $\varphi_{g(x)}(y) = \psi(x, y)$. Perciò valgono le seguenti implicazioni:

$$\begin{array}{l} x \in A \Rightarrow \forall y \,.\, \varphi_{g(x)}(y) \uparrow \Rightarrow W_{g(x)} = \emptyset \Rightarrow |W_{g(x)}| < \omega \Rightarrow g(x) \in B \\ x \notin A \Rightarrow \forall y \,.\, \varphi_{g(x)}(y) \downarrow \Rightarrow W_{g(x)} = \mathbb{N} \Rightarrow |W_{g(x)}| = \omega \Rightarrow g(x) \notin B \end{array}$$

Perciò $A \preceq B$ mediante g, qualunque sia B. □

ESERCIZIO 4.8. *Dimostrare che se l'insieme* A *è ricorsivo e* $B = \{\, x \mid |W_x| = \omega \,\}$ *o* $B = \{\, x \mid W_x = \mathbb{N} \,\}$ *allora* $A \preceq B$.

SOLUZIONE. Dobbiamo dimostrare che $A \preceq B$. Come negli esercizi precedenti usiamo il teorema smn. Perciò definiamo una funzione $\psi(x, y)$ che grazie a tale teorema è uguale alla funzione $\varphi_{g(x)}(y)$ per qualche funzione totale ricorsiva g. A questo punto utilizziamo tale g per effettuare la riduzione funzionale, ovvero definiamo ψ in modo che $x \in A \Leftrightarrow g(x) \in B$. Questo significa che quando $x \in A$ vogliamo che la funzione ψ abbia come dominio $\mathbb{N}$ (ovvero un dominio infinito), e quindi la facciamo terminare su ogni input. Quando invece $x \notin A$ facciamo divergere

la funzione così il dominio è vuoto, ovvero finito e diverso da $\mathbb{N}$. Definiamo quindi la funzione

$$\psi(x,y) = \begin{cases} y & \text{se } x \in A \\ \uparrow & \text{altrimenti} \end{cases}$$

Per il teorema smn abbiamo quindi che esiste g totale tale che $\varphi_{g(x)}(y) = \psi(x,y)$ e perciò valgono le seguenti implicazioni:

$$\begin{aligned} x \in A &\Rightarrow \forall y \,.\, \varphi_{g(x)}(y) = y \Rightarrow W_{g(x)} = \mathbb{N} \Rightarrow |W_{g(x)}| = \omega \Rightarrow g(x) \in B \\ x \notin A &\Rightarrow \forall y \,.\, \varphi_{g(x)}(y) \uparrow \Rightarrow W_{g(x)} = \emptyset \Rightarrow |W_{g(x)}| < \omega \Rightarrow g(x) \notin B \end{aligned}$$

Da cui concludiamo che $A \preceq B$ mediante g, qualunque sia B. □

Esercizio 4.9. *Dimostrare che se l'insieme* A *è ricorsivo e* $B = \{\, i \mid \varphi_i(i) \neq 0 \,\}$ *allora* $A \preceq B$.

Soluzione. Dobbiamo trovare una funzione g tale che $x \in A \Leftrightarrow g(x) \in B$, proviamo allora a sfruttare il teorema smn come negli esercizi precedenti. Definiamo una funzione $\psi(x,y) = \varphi_{g(x)}(y)$ tale che quando $x \in A$ essa sia diversa da 0, e per questo è sufficiente farla divergere. Quando invece $x \notin A$ allora poniamo la funzione uguale a 0 per ogni input, in tal modo essa è zero anche in $g(x)$. Notiamo che questa condizione di non appartenenza può essere usata per definire una funzione parzialmente ricorsiva perché l'insieme A è per ipotesi ricorsivo. Quindi avremo che

$$\psi(x,y) = \begin{cases} 0 & \text{se } x \notin A \\ \uparrow & \text{altrimenti} \end{cases}$$

Per il teorema smn sappiamo quindi che esiste g totale tale che $\varphi_{g(x)}(y) = \psi(x,y)$, per cui valgono le seguenti implicazioni:

$$\begin{aligned} x \in A &\Rightarrow \forall y \,.\, \varphi_{g(x)}(y) \uparrow \Rightarrow \varphi_{g(x)}(g(x)) \uparrow \Rightarrow g(x) \in B \\ x \notin A &\Rightarrow \forall y \,.\, \varphi_{g(x)}(y) = 0 \Rightarrow \varphi_{g(x)}(g(x)) = 0 \Rightarrow g(x) \notin B \end{aligned}$$

Perciò $A \preceq B$ mediante g. □

Esercizio 4.10. *Dimostrare che l'insieme* A *ricorsivo e riducibile a* $\mathrm{REC} = \{\, x \mid W_x \text{ insieme ricorsivo} \,\}$.

Soluzione. Dobbiamo trovare una funzione g tale che $x \in A \Leftrightarrow g(x) \in \mathrm{REC}$. Esattamente come negli esercizi precedenti proviamo a sfruttare il teorema smn. Definiamo quindi una funzione $\psi(x,y) = \varphi_{g(x)}(y)$ tale che quando $x \in A$ il dominio di φ_x sia ricorsivo, ad esempio vuoto quando la funzione diverge. Invece quando $x \notin A$ vogliamo che il dominio non sia ricorsivo, ad esempio sia equivalente ad una condizione semidecidibile come la terminazione di $\varphi_y(y)$. Anche in questo caso possiamo usare $x \notin A$ per definire la funzione parziale ricorsiva ψ essendo A ricorsivo per ipotesi.

$$\psi(x,y) = \begin{cases} \varphi_y(y) & \text{se } x \notin A \\ \uparrow & \text{altrimenti} \end{cases}$$

Per il teorema smn sappiamo che esiste g totale tale che $\varphi_{g(x)}(y) = \psi(x,y)$ e perciò valgono le seguenti implicazioni:

$$\begin{aligned} x \in A &\Rightarrow \forall y \,.\, \varphi_{g(x)}(y) \uparrow \Rightarrow W_{g(x)} = \emptyset \text{ ricorsivo} \Rightarrow g(x) \in \mathrm{REC} \\ x \notin A &\Rightarrow (\varphi_{g(x)}(y) \downarrow \Leftrightarrow \varphi_y(y) \downarrow) \Rightarrow (y \in W_{g(x)} \Leftrightarrow y \in K) \\ &\Rightarrow W_{g(x)} = K \text{ non ricorsivo} \Rightarrow g(x) \notin \mathrm{REC} \end{aligned}$$

Perciò $A \preceq \mathsf{REC}$ mediante g. □

5. Insiemi creativi e produttivi

In questa sezione andremo oltre la mera distinzione tra ricorsività, ricorsiva enumerabilità e non ricorsiva enumerabilità. Dimostreremo infatti la creatività o la produttività di insiemi di indici. Queste proprietà caratterizzano l'appartenenza dell'insieme in esame ad una data classe di ricorsività per insiemi: gli insiemi creativi sono tutti non ricorsivi ma ricorsivamente enumerabili e completi nella classe RE (ovvero ogni insieme RE è ad essi riducibile), mentre gli insiemi produttivi sono strutturalmente non ricorsivamente enumerabili. Gli insiemi creativi e produttivi rappresentano quindi gli elementi *canonici* rispettivamente delle classi degli insiemi ricorsivamente enumerabili (RE) e non ricorsivamente enumerabili ($\overline{\mathrm{RE}}$). Questa parte conclude il corso di calcolabilità elementare.

Ricordiamo qui brevemente alcuni risultati noti sugli insiemi creativi e produttivi che verranno utilizzati per risolvere gli esercizi. Siano $A, B \subseteq \mathbb{N}$:

- A produttivo $\Rightarrow$ A non r.e.
- A creativo $\Rightarrow$ A non ricorsivo
- K è creativo e $\overline{K}$ è produttivo
- A creativo sse A completo
- A produttivo e $A \preceq B$ $\Rightarrow$ B produttivo
- A creativo e $A \preceq B$ $\Rightarrow$ ($B \in \mathsf{RE}$ $\Rightarrow$ B creativo)

Esercizio 5.1. *Dimostrare che l'insieme* $\mathsf{REC} = \{\, i \mid W_i \text{ ricorsivo} \,\}$ *è un insieme produttivo.*

Soluzione. Per dimostrare che REC è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq \mathsf{REC}$, dove $\overline{K} = \{\, i \mid \varphi_i(i) \uparrow \,\}$. In questo esercizio, come nei successivi, lavoreremo come abbiamo fatto nell'Esercizio 4.5. Sappiamo infatti che il teorema smn permette di spostare un input di una funzione nell'indice della MdT che la calcola. Questo ci permette di definire funzioni in cui le condizioni sul primo input diventano condizioni sull'indice della macchina. Ciò è importante perché l'insieme che dobbiamo studiare è un insieme di indici di MdT. L'idea è quella di definire una funzione $\psi(x, y)$ e di utilizzare il teorema smn in modo da ottenere $\psi(x, y) = \varphi_{g(x)}(y)$ per qualche funzione ricorsiva g. A partire da questo cerchiamo di definire tale funzione ψ in modo che si abbia $x \in \overline{K} \Leftrightarrow g(x) \in \mathsf{REC}$, usando quindi tale g per eseguire la riduzione. Quindi se $x \in K$ ($x \notin \overline{K}$) vogliamo fare in modo che $g(x) \notin \mathsf{REC}$ ovvero che $W_{g(x)}$ non sia ricorsivo. Un modo per fare ciò consiste nel far sì che il dominio di $\psi(x, y) = \varphi_{g(x)}(y)$ sia, ad esempio, esattamente K, che sappiamo essere non ricorsivo. Questo si ottiene banalmente facendo corrispondere la terminazione della funzione su un parametro all'appartenenza dello stesso a K, ovvero ponendo $\varphi_{g(x)}(y) = \psi(x, y)$ uguale a $\varphi_y(y)$ quando $x \in K$. D'altra parte quando $x \notin K$ vogliamo che il dominio di $\varphi_{g(x)}$ sia ricorsivo, un modo banale per ottenere ciò è quello di fare in modo che in tal caso il dominio sia vuoto, ovvero la funzione diverga sempre. Perciò se definiamo

$$\psi(x, y) = \begin{cases} \varphi_y(y) & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

vediamo che essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Inoltre per quanto detto sopra si ha che $\psi(x, y) = \varphi_{g(x)}(y)$ e

quindi valgono le seguenti implicazioni:

$$\begin{array}{rcl} x \in K & \Rightarrow & (\varphi_{g(x)}(y)\downarrow \Leftrightarrow \varphi_y(y)\downarrow) \Rightarrow (\varphi_{g(x)}(y)\downarrow \Leftrightarrow y \in K) \\ & \Rightarrow & (y \in W_{g(x)} \Leftrightarrow y \in K) \Rightarrow W_{g(x)} = K \text{ non ricorsivo} \\ & \Rightarrow & g(x) \notin \mathsf{REC} \\ x \notin K & \Rightarrow & \forall y \in \mathbb{N} \,.\, \varphi_{g(x)}(y)\uparrow \Rightarrow W_{g(x)} = \emptyset \text{ ricorsivo} \\ & \Rightarrow & g(x) \in \mathsf{REC} \end{array}$$

Da cui possiamo concludere che $\overline{K} \preceq \mathsf{REC}$ mediante g, perciò quest'ultimo è un insieme produttivo. □

Esercizio 5.2. *Sia* $A = \{\, i \mid \varphi_i(i) = 0 \,\}$*, dimostrare che il suo complementare è un insieme produttivo.*

Soluzione. Per dimostrare che $\overline{A}$ è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq \overline{A}$ che equivale a dimostrare che $K \preceq A$. Analogamente a quanto visto nell'esercizio precedente definiamo una funzione $\psi(x, y)$. A questo punto utilizziamo il teorema smn e quindi sappiamo che esiste una funzione ricorsiva g tale che $\psi(x, y) = \varphi_{g(x)}(y)$. In tal caso vogliamo utilizzare tale g in modo che $x \in K \Leftrightarrow g(x) \in A$, ovvero vogliamo fare in modo che quando $x \in K$ si abbia $\varphi_{g(x)}(g(x)) = 0$, e questo lo si ottiene ponendo la funzione uguale a 0 per ogni input quando $x \in K$. D'altra parte quando $x \notin K$, per ottenere la funzione desiderata è sufficiente che $\varphi_{g(x)}(g(x)) \neq 0$, e il modo più semplice per ottenere ciò è quello di far divergere la funzione. Perciò definiamo

$$\psi(x, y) = \begin{cases} 0 & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Per quanto detto sopra abbiamo che esiste g tale che $\psi(x, y) = \varphi_{g(x)}(y)$, valgono dunque le seguenti implicazioni

$$\begin{array}{rcl} x \in K & \Rightarrow & \psi(x, y) = 0 \Rightarrow \forall y \,.\, \varphi_{g(x)}(y) = 0 \\ & \Rightarrow & \varphi_{g(x)}(g(x)) = 0 \Rightarrow g(x) \in A \\ x \notin K & \Rightarrow & \psi(x, y)\uparrow \Rightarrow \forall y \in \mathbb{N} \,.\, \varphi_{g(x)}(y)\uparrow \\ & \Rightarrow & \varphi_{g(x)}(g(x))\uparrow \Rightarrow g(x) \notin A \end{array}$$

Da cui si ha che $K \preceq A$ mediante g. Questo implica che $\overline{K} \preceq \overline{A}$ e perciò quest'ultimo insieme è produttivo. □

Esercizio 5.3. *Sia* $\overline{\mathsf{REC}} = \{\, i \mid W_i \text{ non ricorsivo} \,\}$*, dimostrare che questo è un insieme produttivo.*

Soluzione. Per dimostrare che $\overline{\mathsf{REC}}$ è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq \overline{\mathsf{REC}}$ che equivale a dimostrare che $K \preceq \mathsf{REC}$. Il procedimento è sempre lo stesso. In questo caso vogliamo costruire $\psi(x, y) = \varphi_{g(x)}(y)$ in modo che $x \in K \Leftrightarrow g(x) \in \mathsf{REC}$. Questo significa che quando $x \in K$ allora $\varphi_{g(x)}$ deve avere un dominio ricorsivo. Un modo banale per ottenere ciò è quello di far in modo che in tal caso il dominio sia tutto $\mathbb{N}$, ovvero che la funzione termini su ogni input. Dobbiamo, però, anche fare in modo che quando $x \notin K$ il dominio sia non ricorsivo, ovvero ad esempio sia K. Quindi in tal caso la funzione deve terminare se e solo se il

secondo parametro, y, sta in K. Alla luce di tutte queste considerazioni definiamo

$$\psi(x,y) = \begin{cases} y & \text{se } x \in K \ \vee \ y \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Allora per quanto detto sopra valgono dunque le seguenti implicazioni

$$\begin{array}{rcl} x \in K & \Rightarrow & \psi(x,y) = y \ \Rightarrow \ \forall y \,.\, \varphi_{g(x)}(y) = y \\ & \Rightarrow & W_{g(x)} = \mathbb{N} \text{ ricorsivo} \ \Rightarrow \ g(x) \in \mathsf{REC} \\ x \notin K & \Rightarrow & (\varphi_{g(x)}(y) \downarrow \Leftrightarrow \ y \in K) \ \Rightarrow \ (y \in W_{g(x)} \ \Leftrightarrow \ y \in K) \\ & \Rightarrow & W_{g(x)} = K \text{ non ricorsivo} \ \Rightarrow \ g(x) \notin \mathsf{REC} \end{array}$$

Dunque $K \preceq \mathsf{REC}$ mediante g. Questo implica che $\overline{K} \preceq \overline{\mathsf{REC}}$ e perciò quest'ultimo insieme è produttivo. □

Esercizio 5.4. *Sia* $A = \{\, i \mid W_i = \emptyset \,\}$*, dimostrare che tale insieme è produttivo.*

Soluzione. Per dimostrare che A è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq A$. Come sempre definiamo una funzione $\psi(x,y)$ e utilizziamo il teorema smn per dire che esiste g totale ricorsiva tale che $\psi(x,y) = \varphi_{g(x)}(y)$. Utilizziamo tale funzione per la riduzione, ovvero facciamo in modo che $x \in \overline{K} \ \Leftrightarrow \ g(x) \in A$. Per ottenere ciò dobbiamo far sì che quando $x \in K$ ($x \notin \overline{K}$) si ottenga che il dominio di $\varphi_{g(x)}$ non sia vuoto, ad esempio sia tutto $\mathbb{N}$. D'altra parte quando $x \notin K$ è sufficiente far divergere la funzione in modo da ottenere il dominio vuoto. Definiamo allora

$$\psi(x,y) = \begin{cases} y & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Per quanto detto si ha che $\psi(x,y) = \varphi_{g(x)}(y)$ e quindi valgono le seguenti implicazioni

$$\begin{array}{rcl} x \in \overline{K} & \Rightarrow & \forall y \,.\, \varphi_{g(x)}(y) \uparrow \Rightarrow \ W_{g(x)} = \emptyset \\ & \Rightarrow & g(x) \in A \\ x \notin \overline{K} & \Rightarrow & \forall y \in \mathbb{N} \,.\, \varphi_{g(x)}(y) = y \ \Rightarrow \ W_{g(x)} = \mathbb{N} \neq \emptyset \\ & \Rightarrow & g(x) \notin A \end{array}$$

Dunque $\overline{K} \preceq A$ mediante g. Questo implica che A è un insieme produttivo. □

Esercizio 5.5. *Sia* $A = \{\, i \mid W_i \neq \mathbb{N} \,\}$*, dimostrare tale insieme è produttivo.*

Soluzione. Per dimostrare che A è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq A$. Definiamo sempre una funzione $\psi(x,y)$ e utilizziamo il teorema smn per dire che esiste una funzione g tale che $\psi(x,y) = \varphi_{g(x)}(y)$. Come sopra utilizziamo tale funzione per ottenere la riduzione, ovvero definiamo ψ in modo che $x \in \overline{K} \ \Leftrightarrow$ $g(x) \in A$. Per ottenere questo operiamo esattamente come nell'esercizio precedente infatti $W_i = \emptyset$ implica $W_i \neq \mathbb{N}$, e $W_i = \mathbb{N}$ implica banalmente $W_i \neq \emptyset$. Da cui definiamo

$$\psi(x,y) = \begin{cases} y & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Allora valgono le seguenti implicazioni

$$\begin{array}{rcl} x \notin \overline{K} & \Rightarrow & \psi(x,y) = y \;\Rightarrow\; \forall y \,.\, \varphi_{g(x)}(y) = y \\ & \Rightarrow & W_{g(x)} = \mathbb{N} \;\Rightarrow\; g(x) \notin A \\ x \in \overline{K} & \Rightarrow & \psi(x,y) \uparrow \Rightarrow\; \forall y \in \mathbb{N} \,.\, \varphi_{g(x)}(y) \uparrow \\ & \Rightarrow & W_{g(x)} = \emptyset \;\Rightarrow\; g(x) \in A \end{array}$$

Dunque $\overline{K} \preceq A$ mediante g, perciò quest'ultimo insieme è produttivo. □

Esercizio 5.6. *Dimostrare che l'insieme* $A = \left\{ x \,\middle|\, |W_x| < \omega \right\}$ *è un insieme produttivo.*

Soluzione. Per dimostrare che A è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq A$. Analogamente a sopra dobbiamo definire $\psi(x,y)$ in modo da applicare il teorema smn e spostare il parametro x nell'indice della MdT come $g(x)$, con la certezza dell'esistenza di una tale funzione ricorsiva g. Perciò dobbiamo fare in modo che $x \in \overline{K} \;\Leftrightarrow\; g(x) \in A$. Ciò si ottiene con la stessa funzione dell'esercizio precedente in quanto $W_i \neq \mathbb{N}$ implica $|W_i| < \omega$ e $|W_i| = \omega$ implica $W_i = \mathbb{N}$. Perciò definiamo

$$\psi(x,y) = \begin{cases} y & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Valgono perciò le seguenti implicazioni

$$\begin{array}{rcl} x \notin \overline{K} & \Rightarrow & \psi(x,y) = y \;\Rightarrow\; \forall y \,.\, \varphi_{g(x)}(y) = y \\ & \Rightarrow & W_{g(x)} = \mathbb{N} \;\Rightarrow\; |W_{g(x)}| = \omega \;\Rightarrow\; g(x) \notin A \\ x \in \overline{K} & \Rightarrow & \psi(x,y) \uparrow \Rightarrow\; \forall y \in \mathbb{N} \,.\, \varphi_{g(x)}(y) \uparrow \\ & \Rightarrow & W_{g(x)} = \emptyset \;\Rightarrow\; |W_{g(x)}| = 0 \;\Rightarrow\; g(x) \in A \end{array}$$

Dunque $\overline{K} \preceq A$ mediante g e perciò quest'ultimo insieme è produttivo. □

Esercizio 5.7. *Sia* $B = \left\{ i \,\middle|\, \exists y \,.\, \varphi_i(y) \downarrow \,\wedge\, \varphi_y(i) \downarrow \,\wedge\, \varphi_i(y) = \varphi_y(i) \right\}$*, dimostrare che il suo complementare è un insieme produttivo.*

Soluzione. Per dimostrare che $\overline{B}$ è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq \overline{B}$ che equivale a dimostrare che $K \preceq B$. Allora definiamo $\psi(x,y)$ seguendo un ragionamento analogo a quanto fatto precedentemente. Sappiamo, per il teorema smn, che esiste g tale che $\psi(x,y) = \varphi_{g(x)}(y)$, usiamo tale funzione per effettuare la riduzione, ovvero in modo che $x \in K \;\Leftrightarrow\; g(x) \in B$. Questo significa che quando $x \in K$ vogliamo che esista un y tale che $\varphi_{g(x)}(y) \downarrow$, $\varphi_y(g(x)) \downarrow$ e infine $\varphi_{g(x)}(y) = \varphi_y(g(x))$. Se noi facciamo in modo che in questo caso la funzioni termini su ogni input allora terminerà anche su $g(x)$ e in tal caso l'uguaglianza è banalmente verificata, $\varphi_{g(x)}(g(x)) = \varphi_{g(x)}(g(x))$. D'altra parte quando $x \notin K$, è sufficiente fare in modo che la funzione non termini mai. Definiamo allora

$$\psi(x,y) = \begin{cases} y & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Per quanto detto sappiamo che $\psi(x,y) = \varphi_{g(x)}(y)$ e quindi valgono le

seguenti implicazioni, dove come abbiamo detto $\psi(x, y) = \varphi_{g(x)}(y)$.

$$\begin{array}{rcl} x \in K & \Rightarrow & \forall y \,.\, \varphi_{g(x)}(y) \downarrow \Rightarrow\ \varphi_{g(x)}(g(x)) \downarrow \\ & \Rightarrow & \varphi_{g(x)}(g(x)) = \varphi_{g(x)}(g(x)) \ \Rightarrow\ g(x) \in B \\ x \notin K & \Rightarrow & \forall y \in \mathbb{N} \,.\, \varphi_{g(x)}(y) \uparrow \\ & \Rightarrow & \nexists y \,.\, \varphi_{g(x)}(y) \downarrow \Rightarrow\ g(x) \notin B \end{array}$$

Da cui $K \preceq B$ mediante g. Questo implica che $\overline{K} \preceq \overline{B}$ e perciò quest'ultimo insieme è produttivo. □

ESERCIZIO 5.8. *Sia* $B = \left\{ \langle i, j\rangle \,\middle|\, \exists x \,.\, \varphi_i(\varphi_j(x)) = x \right\}$, *dimostrare che il suo complementare è un insieme produttivo.*

SOLUZIONE. Per dimostrare che $\overline{B}$ è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq \overline{B}$ che equivale a dimostrare che $K \preceq B$. Come in precedenza dobbiamo definire una funzione $\psi(x, y)$ in modo che per il teorema smn, essa sia uguale a $\varphi_{g(x)}(y)$, per qualche funzione ricorsiva g. A questo punto usiamo tale funzione per la riduzione, in tal caso però l'insieme è un insieme di coppie quindi in realtà useremo g per definire una diversa funzione di riduzione f, ovvero definiamo ψ in modo che $x \in K \ \Leftrightarrow\ f(x) \in B$. Infatti se definiamo $f(x) = \langle g(x), g(x)\rangle$, allora otteniamo la riduzione. Per ottenere ciò vogliamo che quando $x \in K$ si abbia che $\varphi_{g(x)}$ sia una funzione costante in modo che termini su ogni input e dia sempre lo stesso valore, in tal modo se y è il valore costante dato come output, siamo sicuri che $\varphi_{g(x)}(\varphi_{g(x)}(y)) = y$, ovvero che $f(x) \in B$. D'altra parte per negare l'appartenenza a B è sufficiente che la funzione diverga sempre quando $x \notin K$. Definiamo allora

$$\psi(x, y) = \begin{cases} y & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Valgono dunque le seguenti implicazioni, dove come abbiamo detto $\psi(x, y) = \varphi_{g(x)}(y)$.

$$\begin{array}{rcl} x \in K & \Rightarrow & \forall y \,.\, \varphi_{g(x)}(y) = y \ \Rightarrow\ \varphi_{g(x)}(\varphi_{g(x)}(y)) = y \\ & \Rightarrow & \langle g(x), g(x)\rangle \in B \\ x \notin K & \Rightarrow & \forall y \in \mathbb{N} \,.\, \varphi_{g(x)}(y) \uparrow \\ & \Rightarrow & \nexists y \,.\, \varphi_{g(x)}(\varphi_{g(x)}(y)) = y \ \Rightarrow\ \langle g(x), g(x)\rangle \notin B \end{array}$$

Dunque $K \preceq B$ mediante g. Questo implica che $\overline{K} \preceq \overline{B}$ e perciò quest'ultimo insieme è produttivo. □

ESERCIZIO 5.9. *Sia* $B = \left\{ \langle i, j\rangle \,\middle|\, \forall x \,.\, \varphi_i(\varphi_j(x)) = x \right\}$, *dimostrare che è un insieme produttivo.*

SOLUZIONE. Per dimostrare che B è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq B$, che equivale a ridurre $K \preceq \overline{B}$ dove $\overline{B} = \{\langle i, j\rangle |\ \exists x \,.\, \varphi_i(\varphi_j(x)) \neq x\}$. Analogamente a sopra definiamo una funzione ψ e utilizziamo la funzione g ricorsiva dovuta al teorema smn in modo da definire la funzione di riduzione $f(x) = \langle g(x), g(x)\rangle$. In tal caso quindi dobbiamo fare in modo che $x \in K \ \Leftrightarrow\ f(x) \in \overline{B}$. Perciò quando $x \in K$ vogliamo che esista un y tale che $\varphi_{g(x)}(\varphi_{g(x)}(y)) \neq y$. Per ottenere questo dobbiamo trovare un y in cui la funzione diverge, ma questa non è una condizione utilizzabile per definire una funzione parziale ricorsiva, quindi dobbiamo cercare di

formularla in modo semidecidibile. Un modo può essere quello di fare sì che quando $x \in K$ si abbia un dominio finito della funzione $\varphi_{g(x)}$, perché solo in tal modo siamo sicuri che esiste almeno un elemento in cui la funzione diverge. Notiamo che se decidiamo che $x \in K$ significa che $\varphi_x(x)$ termina e questo avviene sicuramente in un numero finito di passi, possiamo allora pensare di far terminare la funzione $\psi(x, y)$ solo se tali passi non sono meno di y (questo controllo richiede un'esecuzione finita e quindi è decidibile). In tal modo se n è il numero di passi necessari per dire che $x \in K$, allora il dominio di $\varphi_{g(x)}$ sarà costituito da ogni valore più piccolo di n. Questo significa che su $n + 1$ la funzione $\varphi_{g(x)}$ diverge, e quindi riusciamo a negare l'appartenenza a B. D'altra parte se $x \notin K$, il dominio sarà tutto $\mathbb{N}$ perché per nessun n si ha che $x \in K$ è deciso in meno di n passi (essendo un fatto falso), quindi se in tal caso poniamo la funzione $\psi(x, y) = y$, siamo sicuri che per ogni y la condizione di appartenenza a B per la coppia $\langle g(x), g(x)\rangle$ vale. Perciò definiamo

$$\psi(x, y) = \begin{cases} y & \text{se } x \in K \text{ non deciso in meno di } y \text{ passi} \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Per quanto detto sopra valgono dunque le seguenti implicazioni, dove $\varphi_{g(x)}(y) = \psi(x, y)$.

$$\begin{array}{rcl} x \in K & \Rightarrow & \exists n_0 \,.\, \varphi_x(x) \downarrow \text{ in } n_0 \text{ passi } \Rightarrow \ x \in K \text{ deciso in } n_0 \text{ passi} \\ & \Rightarrow & W_{g(x)} = [0, n_0] \ \Rightarrow \ \text{se } y = n_0 + 1 \ \varphi_{g(x)}(y) \uparrow \\ & \Rightarrow & \varphi_{g(x)}(\varphi_{g(x)}(y)) \neq y \ \Rightarrow \ \langle g(x), g(x)\rangle \in \overline{B} \\ x \notin K & \Rightarrow & \forall y \in \mathbb{N} \,.\, x \in K \text{ non deciso in meno di } y \text{ passi} \\ & \Rightarrow & \forall y \,.\, \varphi_{g(x)}(y) = y \ \Rightarrow \ \varphi_{g(x)}(\varphi_{g(x)}(y)) = y \\ & \Rightarrow & \langle g(x), g(x)\rangle \notin \overline{B} \end{array}$$

Dunque $K \preceq \overline{B}$ mediante g. Questo implica che $\overline{K} \preceq B$ e perciò quest'ultimo insieme è produttivo. □

Esercizio 5.10. *Sia* $A = \{\, x \mid W_x = \{\, y \mid y \text{ primo} \,\} \,\}$*, dimostrare che è un insieme produttivo.*

Soluzione. Per dimostrare che A è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq A$, che equivale a ridurre $K \preceq \overline{A}$ dove l'insieme A può essere riscritto come $A = \{\, x \mid \varphi_x(y) \downarrow \Leftrightarrow y \text{ primo} \,\}$ e dunque il suo insieme complemento è $\overline{A} = \{\, x \mid \exists y \text{ primo} \,.\, \varphi_x(y) \uparrow \ \vee \exists y \text{ non primo} \,.\, \varphi_x(y) \downarrow \,\}$. Come in precedenza definiamo una funzione ψ tale che, per il teorema smn, $\psi(x, y) = \varphi_{g(x)}(y)$. Usiamo poi tale funzione ricorsiva g per effettuare la riduzione, ovvero definiamo ψ in modo che $x \in K \ \Leftrightarrow \ g(x) \in \overline{A}$. Quindi se $x \in K$ vogliamo che esista un valore primo in cui la funzione diverga oppure esista un valore non primo in cui la funzione termini. Per ottenere ciò è quindi sufficiente che in tal caso la funzione termini per ogni input, includendo quindi anche valori non primi. D'altra parte quando $x \notin K$ vogliamo che il dominio della funzione sia l'insieme dei numeri primi, ovvero vogliamo che la funzione termini su y se e solo se y è primo. Per ottenere ciò poniamo anche la condizione su y di essere primo quando vogliamo che $\varphi_{g(x)}$ termini. Definiamo dunque

$$\psi(x, y) = \begin{cases} y & \text{se } x \in K \ \vee \ y \text{ primo} \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Allora per quanto detto valgono le seguenti implicazioni, dove $\varphi_{g(x)}(y) = \psi(x, y)$.

$$\begin{array}{rcl} x \in K & \Rightarrow & \forall y \,.\, \varphi_{g(x)}(y) = y \;\Rightarrow\; \exists y \text{ non primo} \,.\, \varphi_{g(x)} \downarrow \\ & \Rightarrow & g(x) \in \overline{A} \\ x \notin K & \Rightarrow & (\varphi_{g(x)}(y) \downarrow \Leftrightarrow\; y \text{ primo}) \;\Rightarrow\; g(x) \in A \\ & \Rightarrow & g(x) \notin \overline{A} \end{array}$$

Dunque $K \preceq \overline{A}$ mediante g. Questo implica che $\overline{K} \preceq A$ e perciò quest'ultimo insieme è produttivo. □

ESERCIZIO 5.11. *Sia* $A = \{\, x \mid \varphi_x \text{ definitivamente } \geq 3 \,\}$, *dimostrare che è un insieme produttivo.*

SOLUZIONE. Per dimostrare che A è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq A$, che equivale a ridurre $K \preceq \overline{A}$ dove l'insieme A può essere riscritto come $A = \{\, x \mid \exists n_0 . \forall n > n_0 \,.\, \varphi_x(n) \geq 3 \,\}$ e dunque il suo complemento è $\overline{A} = \{\, x \mid \forall n . \exists n_0 > n \,.\, (\varphi_x(n_0) < 3 \;\vee\; \varphi_x(n_0)$ Come sempre dobbiamo definire una funzione ψ tale che, per il teorema smn, $\psi(x, y) = \varphi_{g(x)}(y)$. Usiamo poi tale funzione ricorsiva g per effettuare la riduzione, ovvero definiamo ψ in modo che $x \in K \Leftrightarrow g(x) \in \overline{A}$. Perciò quando $x \in K$ vogliamo che la funzione non sia mai maggiore di 3. Come nell'Esercizio 5.9 questa non è una condizione che permette di definire una funzione parziale ricorsiva quindi dobbiamo riformularla in una semidecidibile. Per questo usiamo lo stesso trucco, infatti se rendiamo il dominio finito (esattamente come abbiamo fatto nell'Esercizio 5.9) troviamo sempre un input dopo il quale la funzione diverge sempre, implicando che essa non può mai essere definitivamente maggiore o uguale a 3. D'altra parte se $x \notin K$ si avrà il dominio infinito e se poniamo l'output ad esempio uguale a 3, allora siamo sicuri che per ogni input, e quindi definitivamente, la funzione è maggiore o uguale a 3. Definiamo dunque la funzione

$$\psi(x, y) = \begin{cases} 3 & \text{se } x \in K \text{ non deciso in meno di } y \text{ passi} \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Valgono dunque le seguenti implicazioni, dove $\varphi_{g(x)}(y) = \psi(x, y)$.

$$\begin{array}{rcl} x \in K & \Rightarrow & \exists n_0 \,.\, x \in K \text{ deciso in } n_0 \text{ passi} \;\Rightarrow\; \forall n > n_0 \,.\, \varphi_{g(x)}(n) \uparrow \\ & \Rightarrow & \nexists n_0 \forall n > n_0 \,.\, \varphi_{g(x)}(n) \geq 3 \;\Rightarrow\; g(x) \notin A \\ & \Rightarrow & g(x) \in \overline{A} \\ x \notin K & \Rightarrow & \forall y \in \mathbb{N} \,.\, \varphi_{g(x)} \downarrow \Rightarrow\; \forall y \,.\, \varphi_{g(x)}(y) = 3 \\ & \Rightarrow & \forall n \geq 0 \,.\, \varphi_{g(x)}(y) \geq 3 \;\Rightarrow\; g(x) \in A \\ & \Rightarrow & g(x) \notin \overline{A} \end{array}$$

Dunque $K \preceq \overline{A}$ mediante g. Questo implica che $\overline{K} \preceq A$ e perciò quest'ultimo insieme è produttivo. □

ESERCIZIO 5.12. *Sia* $A = \{\, x \mid \forall z \,.\, \varphi_x(z) = c \,\}$ *con* $c \in \mathbb{N}$, *dimostrare che è un insieme produttivo.*

SOLUZIONE. Per dimostrare che A è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq A$, che equivale a ridurre $K \preceq \overline{A}$ dove $\overline{A} = \{\, x \mid \exists z \,.\, \varphi_x(z) \neq c \,\}$. Come

in precedenza definiamo una funzione ψ tale che, per il teorema smn, $\psi(x,y) = \varphi_{g(x)}(y)$. Usiamo poi la funzione ricorsiva g per effettuare la riduzione, ovvero definiamo ψ in modo che $x \in K \Leftrightarrow g(x) \in \overline{A}$. In tal caso quindi vogliamo che per $x \in K$ si abbia la funzione diversa da c per ogni input. È evidente che questo è un problema del tutto analogo a quello dell'Esercizio 5.11, quindi la definizione di ψ e il procedimento usato sarà lo stesso visto nell'esercizio citato. Definiamo dunque la funzione

$$\psi(x,y) = \begin{cases} c & \text{se } x \in K \text{ non deciso in meno di } y \text{ passi} \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Allora per il teorema smn esiste g totale ricorsiva tale che $\psi(x,y) = \varphi_{g(x)}(y)$; valgono dunque le seguenti implicazioni

$$\begin{array}{rcl} x \in K & \Rightarrow & (\exists n_0 \,.\, \varphi_{g(x)} \downarrow \Leftrightarrow\ y \in [0,n_0]) \ \Rightarrow\ \exists y > n_0 \,.\, \varphi_{g(x)}(y) \uparrow \\ & \Rightarrow & g(x) \in \overline{A} \\ x \notin K & \Rightarrow & \forall y \in \mathbb{N} \,.\, \varphi_{g(x)}(y) = c \\ & \Rightarrow & g(x) \notin \overline{A} \end{array}$$

Da cui $K \preceq \overline{A}$ mediante g. Questo implica che $\overline{K} \preceq A$ e perciò quest'ultimo insieme è produttivo. □

Esercizio 5.13. *Sia* $A = \{\, x \mid |\text{RANGE}(\varphi_x)| < \omega \,\}$*, dimostrare che è un insieme produttivo.*

Soluzione. Per dimostrare che A è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq A$, che equivale a ridurre $K \preceq \overline{A}$ dove $\overline{A} = \{\, x \mid |\text{RANGE}(\varphi_x)| = \omega \,\}$. In tal caso quindi vogliamo definire la funzione $\psi(x,y) = \varphi_{g(x)}(y)$ (per il teorema smn) in modo che $x \in K \Leftrightarrow g(x) \in \overline{A}$. Questo significa che se $x \in K$ allora ψ deve essere tale che il suo range sia infinito, ovvero sia tutto $\mathbb{N}$. Per ottenere ciò è sufficiente farla terminare su ogni y restituendo y stesso. D'altra parte quando $x \notin K$, vogliamo che il dominio sia finito, ma questo significa che può essere anche vuoto, quindi in tal caso possiamo far divergere la funzione. Perciò definiamo

$$\psi(x,y) = \begin{cases} y & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Allora per quanto detto valgono le seguenti implicazioni, dove $\psi(x,y) = \varphi_{g(x)}(y)$.

$$\begin{array}{rcl} x \in K & \Rightarrow & \forall y \,.\, \varphi_{g(x)}(y) = y \ \Rightarrow\ \text{RANGE}(\varphi_{g(x)}) = \mathbb{N} \\ & \Rightarrow & |\text{RANGE}(\varphi_{g(x)})| = \omega \ \Rightarrow\ g(x) \in \overline{A} \\ x \notin K & \Rightarrow & \forall y \in \mathbb{N} \,.\, \varphi_{g(x)} \uparrow \Rightarrow\ \text{RANGE}(\varphi_{g(x)}) = \emptyset \\ & \Rightarrow & |\text{RANGE}(\varphi_{g(x)})| = 0 < \omega \ \Rightarrow\ g(x) \notin \overline{A} \end{array}$$

Dunque $K \preceq \overline{A}$ mediante g. Questo implica che $\overline{K} \preceq A$ e perciò quest'ultimo insieme è produttivo. □

Esercizio 5.14. *Sia* $A = \{\, x \mid \varphi_x \text{ totale} \,\}$*, dimostrare che è un insieme produttivo.*

SOLUZIONE. Per dimostrare che A è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq A$, che equivale a ridurre $K \preceq \overline{A}$ dove l'insieme A può essere riscritto come $A = \{\, x \mid W_x = \mathbb{N} \,\}$ e dunque $\overline{A} = \{\, x \mid W_x \neq \mathbb{N} \,\}$. Definiamo quindi una funzione ψ tale che per il teorema smn, si abbia $\psi(x, y) = \varphi_{g(x)}(y)$ per qualche funzione ricorsiva g. Usiamo poi tale funzione per la riduzione, ovvero definiamo ψ in modo che $x \in K \Leftrightarrow g(x) \in \overline{A}$. Quindi se $x \in K$ si deve avere che ψ non sia totale, ovvero diverga su qualche input. Analogamente a quanto visto nell'Esercizio 5.9, questa condizione non va bene per definire una funzione parziale ricorsiva, per cui usiamo nuovamente l'artificio usato nell'esercizio citato. Quindi se $x \in K$ rendiamo il dominio della funzione finito, perciò da un certo punto in poi siamo sicuri che la funzione diverge, ovvero non è totale. Se invece $x \notin K$, allora il dominio è tutto $\mathbb{N}$, come accadeva nell'Esercizio 5.9, e quindi la funzione è totale. Definiamo perciò

$$\psi(x, y) = \begin{cases} y & \text{se } x \in K \text{ non deciso in meno di } y \text{ passi} \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tale funzione è effettiva e dunque essa è parziale ricorsiva. Allora per quanto detto valgono le seguenti implicazioni, dove $\psi(x, y) = \varphi_{g(x)}(y)$.

$$\begin{array}{lll} x \in K & \Rightarrow & \exists n_0 \,.\, x \in K \text{ deciso in } n_0 \text{ passi } \Rightarrow\ (\varphi_{g(x)} \downarrow \Leftrightarrow\ y \leq n_0) \\ & \Rightarrow & W_{g(x)} = [0, n_0] \neq \mathbb{N} \ \Rightarrow\ g(x) \notin A \\ & \Rightarrow & g(x) \in \overline{A} \\ x \notin K & \Rightarrow & \forall y \in \mathbb{N} \,.\, \varphi_{g(x)} \downarrow \\ & \Rightarrow & W_{g(x)} = \mathbb{N} \ \Rightarrow\ g(x) \in A \\ & \Rightarrow & g(x) \notin \overline{A} \end{array}$$

Dunque $K \preceq \overline{A}$ mediante g. Questo implica che $\overline{K} \preceq A$ e perciò quest'ultimo insieme è produttivo. □

ESERCIZIO 5.15. *Sia* $B = \{\, \langle x, y \rangle \mid W_x = W_y \,\}$, *dimostrare che è un insieme produttivo.*

SOLUZIONE. Cerchiamo di risolvere questo esercizio in modo analogo a quanto visto nell'Esercizio 5.9. Sappiamo già che per dimostrare che B è produttivo è sufficiente dimostrare la riduzione $\overline{K} \preceq B$, che equivale a ridurre $K \preceq \overline{B}$ dove $\overline{B} = \{\, \langle x, y \rangle \mid W_x \neq W_y \,\}$. Analogamente a quanto visto nell'esercizio citato, dobbiamo definire una funzione di riduzione $f(x) = \langle g_1(x), g_2(x) \rangle$ che permetta di portare una x che sta in K in una $f(x)$ che non sta nell'insieme B. Un'idea che può venire è quella di definire due funzioni ψ le cui funzioni totali derivate mediante il teorema smn siano le funzioni g_1 e g_2 che usiamo per definire f. In tal caso quindi dobbiamo fare in modo che $x \in K \Leftrightarrow f(x) \in \overline{B}$. Perciò quando $x \in K$ vogliamo che $W_{g_1(x)} \neq W_{g_2(x)}$, ad esempio uno sia tutto $\mathbb{N}$ e l'altro un suo sottoinsieme proprio come K. Per ottenere questo dobbiamo definire una ψ_1 totale quando $x \in K$ e una ψ_2 parziale nelle stesse condizioni su x. D'altra parte se $x \notin K$, i domini devono essere uguali tra loro e quindi, ad esempio, entrambi uguali all'insieme vuoto. Perciò definiamo

$$\psi_1(x, y) = \begin{cases} y & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases} \qquad \text{e} \qquad \psi_2(x, y) = \begin{cases} y & \text{se } x \in K \ \wedge\ y \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K r.e. la definizione di tali funzioni è effettiva e dunque esse sono parziali ricorsive. Per quanto detto sopra valgono allora le seguenti implicazioni, dove

$\varphi_{g_1(x)}(y) = \psi_1(x,y)$ e $\varphi_{g_2(x)}(y) = \psi_2(x,y)$ per il teorema smn.

$$\begin{array}{rcl} x \in K & \Rightarrow & \forall y \in \mathbb{N} \,.\, \varphi_{g_1(x)}(y)\downarrow \text{ e } \varphi_{g_1(x)}(y)\downarrow \Leftrightarrow\ y \in K \\ & \Rightarrow & W_{g_1(x)} = \mathbb{N} \text{ e } W_{g_2(x)} = K \\ & \Rightarrow & W_{g_1(x)} \neq W_{g_2(x)} \ \Rightarrow\ f(x) = \langle g_1(x), g_2(x)\rangle \in \overline{B} \\ x \notin K & \Rightarrow & \forall y \in \mathbb{N} \,.\, \varphi_{g_1(x)}(y)\uparrow \text{ e } \forall y \in \mathbb{N} \,.\, \varphi_{g_2(x)}(y)\uparrow \\ & \Rightarrow & W_{g_1(x)} = \varnothing = W_{g_2(x)} \Rightarrow\ f(x) = \langle g_1(x), g_2(x)\rangle \notin \overline{B} \end{array}$$

Dunque $K \preceq \overline{B}$ mediante f. Questo implica che $\overline{K} \preceq B$ e perciò quest'ultimo insieme è produttivo. □

Esercizio 5.16. *Siano* $A = \{\, x \mid W_x = \emptyset \,\}$ *e* $B = \{\, x \mid W_x = \mathbb{N} \,\}$ *due insiemi produttivi (vedi Esercizi 5.4 e 5.14), dire quale relazione esiste tra i due.*

Soluzione. Sappiamo per la produttività che $\overline{K} \preceq A$ e $\overline{K} \preceq B$; si nota inoltre che $\overline{A} \preceq K$ essendo $\overline{A} = \{\, x \mid W_x \neq \emptyset \,\}$ un insieme ricorsivamente enumerabile (vedi Esercizio 4.5). Perciò $A \preceq \overline{K} \preceq B$, ovvero $A \preceq B$.
D'altra parte $A \not\equiv B$ perché $B \not\preceq A$; infatti se ciò accadesse, essendo $A \preceq \overline{K}$, si avrebbe $B \preceq \overline{K}$ per transitività. Ma allora $\overline{B} \preceq K$ e quindi $\overline{B}$ sarebbe r.e. mentre si è già dimostrato che $\overline{B} = \{\, x \mid W_x \neq \mathbb{N} \,\}$ è produttivo. □

Esercizio 5.17. *Dimostrare che l'insieme* $A = \{\, x \mid \exists y \,.\, x \in W_y \ \wedge\ y \in W_x \,\}$ *è un insieme creativo.*

Soluzione. Per dimostrare che un insieme è creativo dobbiamo dimostrare che esso è ricorsivamente enumerabile e completo. Per dimostrare che è ricorsivamente enumerabile dobbiamo definire una funzione di cui esso è il range, per cui definiamo una funzione che restituisce x quando, presi x e y si ha che $x \in W_y \ \wedge\ y \in W_x$, ovvero la funzione restituisce esattamente gli elementi di A. Percui definiamo la funzione

$$\psi(x,y) = \begin{cases} x & \text{se } x \in W_y \ \wedge\ y \in W_x \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo gli insiemi W_x e W_y r.e., allora la funzione appena definita è parziale ricorsiva dunque il range(ψ) è r.e. ed è esattamente A.
Per dimostrare che è completo è sufficiente dimostrare che $K \preceq A$. Come negli esercizi precedenti questo si ottiene mediante il teorema smn, definendo una funzione $\psi(x,y)$ che è uguale a $\varphi_{g(x)}(y)$ per qualche funzione ricorsiva g. Usiamo poi tale g per effettuare la riduzione, ovvero facciamo in modo che $x \in K \ \Leftrightarrow\ g(x) \in A$. Quindi vorremmo che, se $x \in K$, allora esista y tale che $y \in W_{g(x)}$ e $g(x) \in W_y$. Questo si ottiene banalmente facendo terminare la funzione su ogni input e quindi, in particolare, anche su $g(x)$, in tal modo le due relazioni di appartenenza sono banalmente soddisfatte. D'altra parte, quando $x \notin K$, per negare l'appartenenza ad A è sufficiente rendere vuoto il dominio e quindi far divergere sempre la funzione. Definiamo dunque la funzione

$$\psi(x,y) = \begin{cases} y & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo K un insieme r.e., tale funzione è parziale ricorsiva. Allora valgono le seguenti implicazioni, dove $\varphi_{g(x)}(y) = \psi(x, y)$.

$$\begin{array}{rcl} x \in K & \Rightarrow & \forall y \,.\, \varphi_{g(x)}(y) = y \;\Rightarrow\; \varphi_{g(x)}(g(x)) = g(x) \\ & \Rightarrow & g(x) \in W_{g(x)} \;\Rightarrow\; g(x) \in A \\ x \notin K & \Rightarrow & \forall y \,.\, \varphi_{g(x)}(y) \uparrow \Rightarrow\; W_{g(x)} = \emptyset \\ & \Rightarrow & \nexists y \in W_{g(x)} \;\Rightarrow\; g(x) \notin A \end{array}$$

Perciò A è completo ed r.e., quindi è creativo. □

Esercizio 5.18. *Dimostrare che l'insieme* $A = \{\, i \mid \varphi_i(i) = 0 \,\}$ *è un insieme creativo.*

Soluzione. Per dimostrare che un insieme è creativo dobbiamo dimostrare che esso è ricorsivamente enumerabile e completo. Definiamo la funzione il cui range è esattamente A, ovvero la funzione che restituisce esattamente i valori x tali che $\varphi_x(x) = 0$.

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

Essendo la condizione decidibile, allora la funzione appena definita è parziale ricorsiva dunque il $\text{RANGE}(\psi)$ è r.e. ed è esattamente A.
Per dimostrare che è completo è sufficiente dimostrare che $K \preceq A$, ma questo è esattamente ciò che abbiamo fatto nell'Esercizio 5.2. Perciò A è anche completo e quindi creativo. □

Esercizio 5.19. *Dimostrare che l'insieme* $\overline{A} = \{\, i \mid \varphi_i(i) \neq 0 \,\}$ *è un insieme produttivo usando la definizione di produttività.*

Soluzione. Per dimostrare che $\overline{A}$ è produttivo usando la definizione di produttività dobbiamo costruire una funzione g tale che

$$W_x \subseteq \overline{A} \;\Rightarrow\; g(x) \in \overline{A} \setminus W_x$$

questo significa che dobbiamo fare in modo che quando $W_x \subseteq \overline{A}$ allora $g(x) \in \overline{A}$ ma $g(x) \notin W_x$. Per farlo possiamo pensare di sfruttare ancora il teorema smn. Definiamo cioè una funzione $\psi(x, y)$ che sappiamo essere uguale a $\varphi_{g(x)}(y)$ per qualche funzione ricorsiva totale g. Infatti se definiamo la funzione $\psi(x, y) = \varphi_{g(x)}(y)$ in modo che $g(x)$ stia in A se e solo se $g(x) \in W_x$, allora è evidente che se $W_x \subseteq \overline{A}$ si ha sicuramente $g(x) \in \overline{A}$ e $g(x) \notin W_x$, altrimenti otterrei degli assurdi. Vediamo formalmente come avviene tutto ciò. Perché si abbia che $g(x)$ stia in A se e solo se $g(x) \in W_x$, bisogna fare in modo che $\varphi_{g(x)}(g(x))$ sia 0 se e solo se $\varphi_x(g(x)) \downarrow$. Per ottenere ciò è sufficiente fare sì che $\varphi_{g(x)}$ termini su tutti gli elementi del dominio di φ_x (in particolare quindi la doppia implicazione varrà per $g(x)$). Definiamo perciò la funzione

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

Essendo la condizione semidecidibile, si ha che la funzione appena definita è parziale ricorsiva dunque per il teorema smn si ha che $\psi(x, y) = \varphi_{g(x)}(y)$. Valgono così le

seguenti implicazioni

$$\begin{array}{rcl} g(x) \in A & \Leftrightarrow & \varphi_{g(x)}(g(x)) = 0 \Leftrightarrow \psi(x, g(x)) = 0 \\ & \Leftrightarrow & \varphi_x(g(x)) \downarrow \Leftrightarrow g(x) \in W_x \end{array}$$

Questo significa che $W_x \subseteq \overline{A}$ implica $g(x) \in \overline{A}$. Infatti se $g(x) \notin \overline{A}$ allora $g(x) \in A$ e quindi per quanto appena visto abbiamo che $g(x) \in W_x$, ma questo è assurdo se consideriamo $W_x \subseteq \overline{A}$, perché si avrebbe $g(x) \in \overline{A}$. Inoltre $g(x) \notin W_x$ perché se $g(x) \in W_x \subseteq \overline{A}$ allora, sempre per le implicazioni viste sopra, $g(x) \in A$ e quindi è evidente che abbiamo nuovamente un assurdo. Dunque A è produtivo perché abbiamo trovato una funzione g ricorsiva che soddisfa la definizione di insieme produttivo. □

ESERCIZIO 5.20. *Classificare il seguente insieme:*

$$A = \left\{ x \,\middle|\, \exists y \,.\, x = 2^y \;\Rightarrow\; \varphi_{\log_2 x}(x) \downarrow \right\}$$

e il suo complementare.

SOLUZIONE. Dobbiamo classificare tale insieme. Notiamo prima di tutto che A non può essere ricorsivo perché è definito in termini di una funzione parziale ricorsiva. D'altra parte notiamo che esso è ricorsivamente enumerabile, infatti preso x possiamo decidere se esiste y tale che $x = 2^y$ in quanto siamo sicuri che $x \leq 2^x$ dunque dobbiamo fare un numero finito di controlli. Una volta deciso se esiste tale y sappiamo che, in caso negativo, l'implicazione è automaticamente vera, altrimenti abbiamo $\log_2 x \in \mathbb{N}$ e quindi siamo in grado di trovare la macchina $\varphi_{\log_2 x}(x)$ e lanciarla sull'input x. Questo significa che A è ricorsivamente enumerabile perciò possiamo dimostrare che esso è creativo, ovvero che $K \preceq A$. Come in precedenza useremo il teorema smn in modo da definire la funzione $\psi(x, y) = \varphi_{g(x)}(y)$. Nell'insieme dato abbiamo, però, come indice della MdT il valore $\log_2 x$ quindi la funzione g non può bastare per eseguire la riduzione. Avremo dunque bisogno di una funzione f tale che $\log_2 f(x) = g(x)$, ovvero $f(x) = 2^{g(x)}$. Perciò dobbiamo definire $\psi(x, y)$ in modo che $x \in K \Leftrightarrow f(x) \in A$ ovvero $x \in K \Leftrightarrow \varphi_{g(x)}(f(x)) \downarrow$. Definiamo perciò la funzione

$$\psi(x, y) = \begin{cases} y & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo la condizione semidecidibile, si ha che la funzione appena definita è parziale ricorsiva dunque per il teorema smn esiste g totale tale che $\psi(x, y) = \varphi_{g(x)}(y)$. Valgono così le seguenti implicazioni

$$\begin{array}{rcl} x \in K & \Rightarrow & \forall y \,.\, \varphi_{g(x)}(y) \downarrow \Rightarrow \varphi_{g(x)}(2^{g(x)}) \downarrow \\ & \Rightarrow & f(x) = 2^{g(x)} \in A \\ x \notin K & \Rightarrow & \forall y \,.\, \varphi_{g(x)}(y) \uparrow \Rightarrow \varphi_{g(x)}(2^{g(x)}) \uparrow \\ & \Rightarrow & f(x) = 2^{g(x)} \notin A \end{array}$$

Perciò A è completo, ricorsivamente enumerabile e quindi creativo. Questo implica anche che $\overline{A}$ è produttivo. □

ESERCIZIO 5.21. *Classificare il seguente insieme:*

$$A = \left\{ 2^{3^{\cdot^{\cdot^{\cdot^{x}}}}} \;\middle|\; \varphi_x\left(2^{3^{\cdot^{\cdot^{\cdot^{x}}}}}\right) \downarrow \right\}$$

e il suo complementare.

SOLUZIONE. Per prima cosa dimostriamo che l'insieme dato è ricorsivamente enumerabile. Infatti la funzione $2^{3^{\cdot^{\cdot^{\cdot^{x}}}}}$ è totale ed inoltre è crescente ed iniettiva, ovvero invertibile. Infatti dato un generico naturale y possiamo sempre decidere se y è della forma $2^{3^{\cdot^{\cdot^{\cdot^{x}}}}}$ trovando anche tale x semplicemente verificando, per ogni $z \leq y$, se $y = 2^{3^{\cdot^{\cdot^{\cdot^{z}}}}}$. Essendo la funzione crescente se nessun $z \leq y$ ha le proprietà richieste siamo sicuri che y non è della forma cercata. Essendo la funzione iniettiva se un tale z esiste esso è unico. Queste osservazioni permettono di concludere che per ogni naturale y possiamo estrarre la x, quando esiste, tale che $y = 2^{3^{\cdot^{\cdot^{\cdot^{x}}}}}$ e possiamo lanciare la MdT di indice x sul dato input, quindi l'insieme dato è r.e. Dimostriamo allora che A è creativo. Come l'esercizio precedente, anche questo è differente rispetto agli esercizi standard. Infatti mediante il teorema smn possiamo definire una funzione $\psi(x, y) = \varphi_{g(x)}(y)$. In tal caso però l'oggetto che deve appartenere all'insieme A deve essere una funzione di g, e in particolare deve essere $f(x) = 2^{3^{\cdot^{\cdot^{\cdot^{g(x)}}}}}$. Allora dobbiamo definire $\psi(x, y)$ in modo che $x \in K \Leftrightarrow f(x) \in A$ ovvero $x \in K \Leftrightarrow \varphi_{g(x)}(f(x)) \downarrow$. Definiamo perciò la funzione

$$\psi(x, y) = \begin{cases} y & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo la condizione semidecidibile, si ha che la funzione appena definita è parziale ricorsiva dunque per il teorema smn esiste una funzione totale g tale che $\psi(x, y) = \varphi_{g(x)}(y)$. Valgono così le seguenti implicazioni

$$\begin{aligned} x \in K &\Rightarrow \forall y \,.\, \varphi_{g(x)}(y) \downarrow \Rightarrow \varphi_{g(x)}\left(2^{3^{\cdot^{\cdot^{\cdot^{x}}}}}\right) \downarrow \\ &\Rightarrow f(x) = 2^{3^{\cdot^{\cdot^{\cdot^{x}}}}} \in A \\ x \notin K &\Rightarrow \forall y \,.\, \varphi_{g(x)}(y) \uparrow \Rightarrow \varphi_{g(x)}\left(2^{3^{\cdot^{\cdot^{\cdot^{x}}}}}\right) \uparrow \\ &\Rightarrow f(x) = 2^{3^{\cdot^{\cdot^{\cdot^{x}}}}} \notin A \end{aligned}$$

Perciò A è completo e ricorsivamente enumerabile, quindi è creativo. Questo implica anche che $\overline{A}$ è produttivo. □