Esame 03/02/2025

1° Parte

- **(6pt)** Classificare, SENZA dimostrarlo formalmente (Fornire solo automa o grammatica e/o stringa per il Pumping Lemma **commentando brevemente**), al variare di $m, h \in \mathbb{N}$ i seguenti linguaggi sull'alfabeto $\{0, 1\}$:

$$A_{m,h} = \{\ 0^n 1^m 0^h \mid n \in 2m\mathbb{N} + m\ \} \quad B_{m,h} = \{\ 0^h 1^m 0^n \mid n \in 3m\mathbb{N} + m\ \}$$

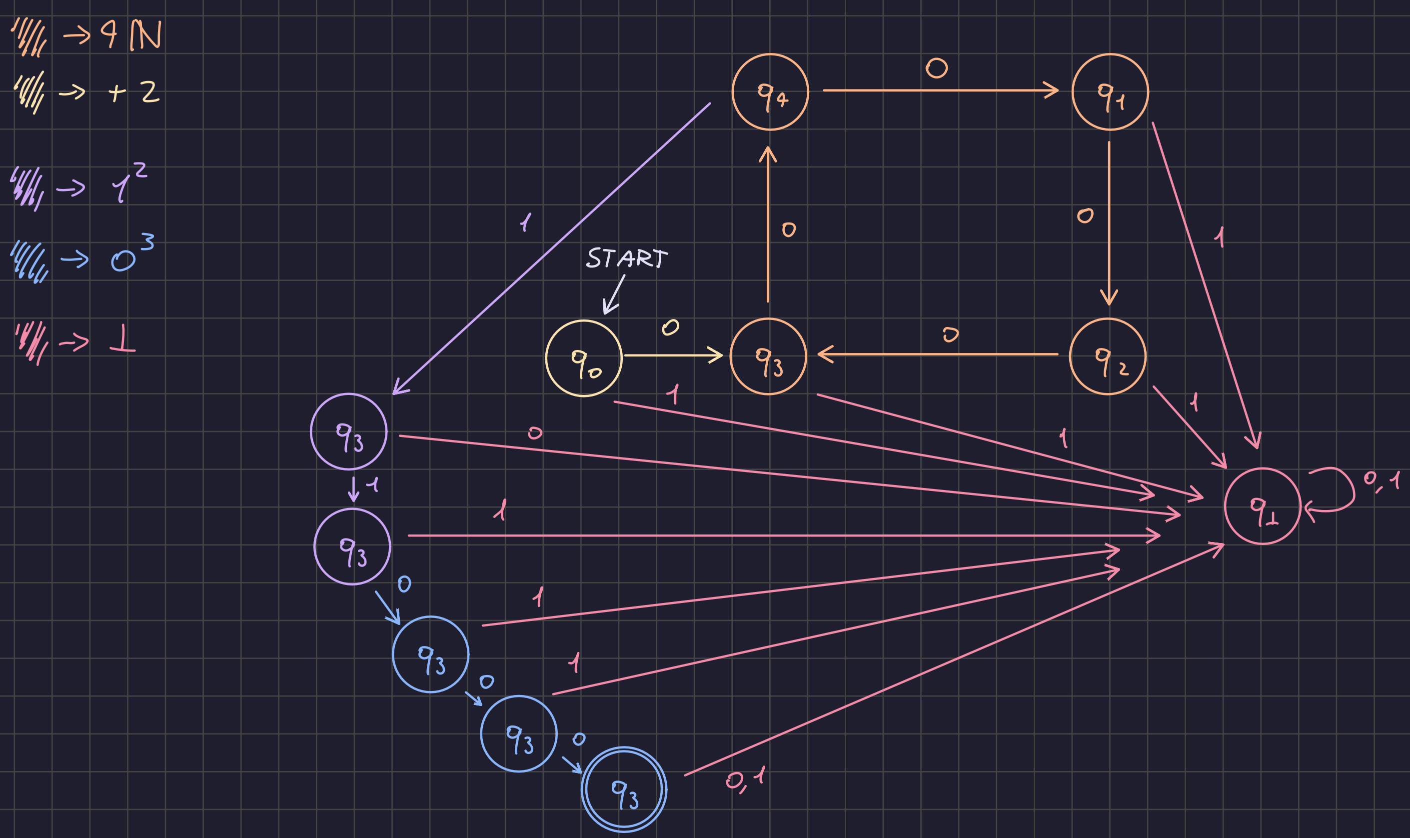
In particolare classificare $A_{2,3}$ e $B_{2,3}$.

$A_{0,0} = \{0\} = A_{0,1}$ Se $m=0 \rightarrow$ Linguaggio costante $\rightarrow A_{0,h} \in REG$

$A_{1,0} = \{0^n 1 \mid n \in 2\mathbb{N}+1\}$

$A_{2,3} = \{0^n 1^2 0^3 \mid n \in 4\mathbb{N}+2\}$

Questo linguaggio è intuitivamente regolare perchè si può riconoscere con una memoria finita siccome non ci sono dipendenze tra gruppi di simboli.
Questo linguaggio riconosce tutte le stringhe con un numero iniziale di zeri uguale ai multipli di 4 a cui e stato sommato 4 e che ha alla fine 2 uni e 3 zeri. L'automa è il seguente



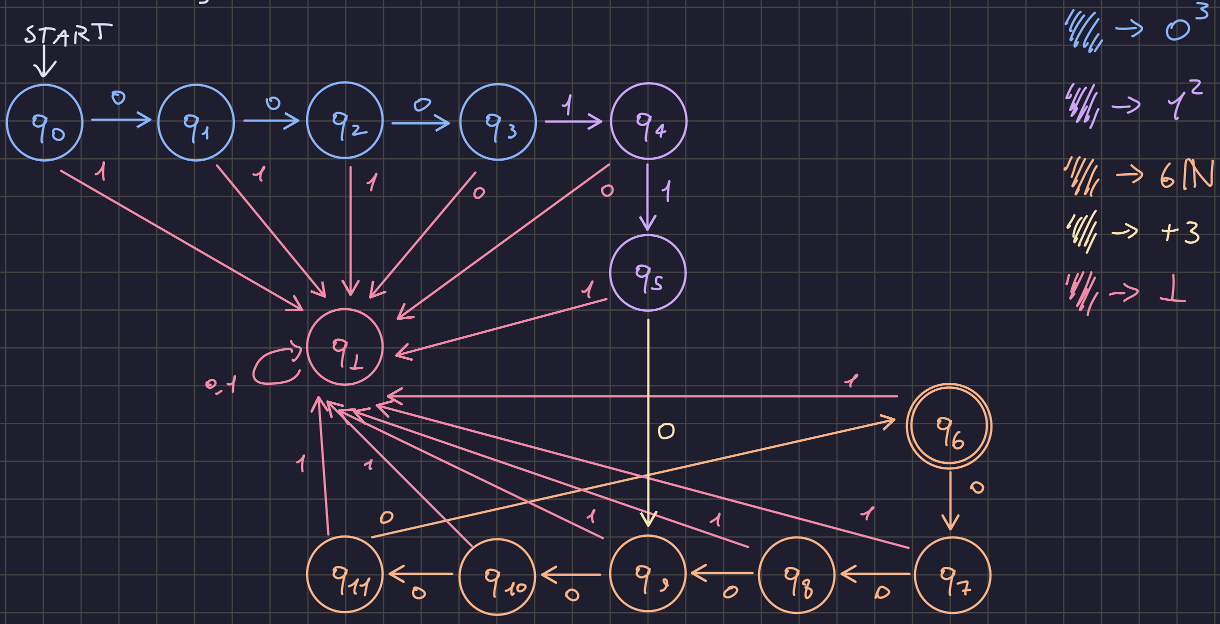
$A = \{0^n 1^m 0^h \mid n \in 2m\mathbb{N}+m,\ m,h \in \mathbb{N}\}$

Tutti i linguaggi formati dall'insieme A sono regolari perchè tutti i gruppi di simboli sono indipendenti e la funzione di n è calcolabile tramite una macchina a stati finiti.

$B_{m,h} = \{ 0^h 1^m 0^n \mid n \in 3m\mathbb{N} + m \}$

$B_{2,3} = \{ 0^3 1^2 0^n \mid n \in 6\mathbb{N} + 3 \}$

Anche questo linguaggio è intuitivamente regolare perchè si può riconoscere con una memoria finita siccome non ci sono dipendenze tra gruppi di simboli.
Questo linguaggio riconosce tutte le stringhe con 3 zeri e 2 uni all'inizio e seguiti da un numero di zeri multiplo di 6 a cui viene sommato 3.
L'automa è il seguente



$B = \{ 0^h 1^m 0^n \mid n \in 3m\mathbb{N} + m,\ m, h \in \mathbb{N} \}$

Tutti i linguaggi formati dall'insieme B sono regolari perchè tutti i gruppi di simboli sono indipendenti e la funzione di n è calcolabile tramite una macchina a stati finiti.

Classificare nella gerarchia di Chomsky i seguenti linguaggi motivando formalmente la risposta[1] :

- **(12pt)** Classificare La seguente famiglia di linguaggi al variare di $m \in \mathbb{N}$:

$$D_m = \bigcup_{h \in \mathbb{N}} \{ \, 0^n 1^m 0^{2h} \mid n \in 2m\mathbb{N} + 3h \, \}$$

In particolare dimostrare la classificazione per D_3

$D_0 = \bigcup_{h \in \mathbb{N}} \{ 0^{n+2h} \mid n = 3h \} = \{ 0^{5h} \mid h \in \mathbb{N} \} \in REG$

$D_1 = \bigcup_{h \in \mathbb{N}} \{ 0^n 1\, 0^{2h} \mid n \in 2\mathbb{N} + 3h \} \in CF$ Perchè il primo gruppo di zeri dipende dall'ultimo

$D_2 = \bigcup_{h \in \mathbb{N}} \{ 0^n 11 0^{2h} \mid n \in 4\mathbb{N} + 3h \} \in CF$ Perchè il primo gruppo di zeri dipende dall'ultimo

$D_3 = \bigcup_{h \in \mathbb{N}} \{ 0^n 1^3 0^{2h} \mid n \in 6\mathbb{N} + 3h \} = \{ 0^n 1^3 0^{2h} \mid h \in \mathbb{N},\ n \in 6\mathbb{N} + 3h \}$

Questo linguaggio è intuitivamente context free perchè il primo gruppo di zeri dipende dall'ultimo.
La grammatica che genera questo linguaggio è la seguente:

$G = S \to 0^6 S \mid 000S00 \mid 1^3$

Per dimostrare che questa grammatica genera il linguaggio deve valere: $L = L(G)$

$$x \in L \Leftrightarrow S \Rightarrow_* x$$

1) $L \subseteq L(G)$

La tesi da dimostrare è: $x \in L \rightarrow S \Rightarrow_* x$

Dimostro per induzione sulla lunghezza della stringa:

Caso base

Considero la stringa di lunghezza minima che è nel linguaggio

$$|x| = 3 \rightarrow x = 111 \in L \rightarrow S \rightarrow 111$$

Esiste una produzione per la stringa x

Passo induttivo

Suppongo che la tesi valga per stringhe di lunghezza minore di n e dimostro che vale per stringhe di lunghezza n+1:

Ipotesi induttiva $\forall x \in L . |x| \leq n . \; x \in L \rightarrow S \Rightarrow_* x$

con x della forma $\exists j \in \mathbb{N} . \exists i \in 6\mathbb{N}+3j . \; x = 0^i 1^3 0^{2j}$

Considero una stringa x di lunghezza n nel linguaggio, la espando per formare una stringa x' di lunghezza maggiore $|x'| > |x|$

$\exists x . \; |x| = n$

Ci sono due modi per espandere la stringa x e creare x':

1) Si aggiungono 6 zeri a sinistra

$$x = 0^{i-6} 1^3 0^{2j} \in L \rightarrow x' = 0^i 1^3 0^{2j} \quad \wedge \; |x'| > |x|$$

So che esiste una derivazione che genera x per ipotesi induttiva

$$x \in L \rightarrow S \overset{ii}{\Rightarrow_*} 0^{i-6} S 0^{2j} \Rightarrow 0^i 111\, 0^{2j} = x$$

Posso partire da questa derivazione per creare x'
Dimostro che esiste una derivazione che genera x'.

$$S \overset{ii}{\Rightarrow_*} 0^{i-6} S 0^{2j} \Rightarrow 0^{i-6} 0^6 S 0^{2j} \Rightarrow 0^i 1^3 0^{2j} = x' \in L \text{ perché } j \in \mathbb{N} \wedge i \in 6\mathbb{N}+3j$$

2) Si aggiungono 3 zeri a sinistra e 2 a destra

$$x = 0^{i-3} 1^3 0^{2j-2} \rightarrow x' = 0^i 1^3 0^{2j} \quad |x'| > |x|$$

So che esiste una derivazione che genera x per ipotesi induttiva

$$x \in L \rightarrow S \overset{ii}{\Rightarrow_*} 0^{i-3} S 0^{2j-2} \Rightarrow 0^{i-3} 111\, 0^{2j-2} = x$$

Dimostro che esiste una derivazione che genera x'.

$$S \overset{ii}{\Rightarrow_*} 0^{i-3} S 0^{2j-2} \Rightarrow 0^{i-3} 0^3 S 0^2 0^{2j-2} \Rightarrow 0^i 1^3 0^{2j} = x' \in L \text{ perché } j \in \mathbb{N} \wedge i \in 6\mathbb{N}+3j$$

2) $L(G) \subseteq L$

La tesi da dimostrare è: $S \Rightarrow_n x \rightarrow x \in L$

Dimostro per induzione sulla lunghezza della derivazione:

Caso base

Considero la derivazione minima che generi una stringa terminale

$$n=1 \rightarrow S \Rightarrow 111 \rightarrow 111 \in L$$

Passo induttivo

Suppongo che la tesi valga per derivazioni di lunghezza minore di n e dimostro che vale per derivazioni di lunghezza n+1:

Ipotesi induttiva $\forall k \in \mathbb{N}. k \leq n. \ S \Rightarrow_k x \rightarrow x \in L \ \wedge \ \exists j \in \mathbb{N}. \exists i \in 6\mathbb{N}+3j. \ x = 0^i 1^3 0^{2j}$

Sia $S \Rightarrow_n x$ una derivazione lunga n, allora per l'ipotesi induttiva x è nel linguaggio ed è della forma:

$\exists j \in \mathbb{N}. \exists i \in 6\mathbb{N}+3j. \ x = 0^i 1^3 0^{2j}$

$$S \Rightarrow_n x \equiv S \Rightarrow_{n-1} 0^i S 0^{2j} \Rightarrow 0^i 111 0^{2j} = 0^i 1^3 0^{2j} \in L$$

Dimostro che le stringhe generate da derivazioni di lunghezza n+1 sono nel linguaggio. Si distinguono i seguenti casi.

1) $S \Rightarrow_{n-1} 0^i S 0^{2j} \Rightarrow 0^i 0^6 S 0^{2j} \Rightarrow 0^{i+6} 111 0^{2j} = x' \in L$ perché:

$i \in 6\mathbb{N}+3j \rightarrow i+6 \in 6\mathbb{N}+3j$ perché $i+6$ é multiplo di 6 e j é rimasto invariato

2) $S \Rightarrow_{n-1} 0^i S 0^{2j} \Rightarrow 0^i 0^3 S 0^2 0^{2j} \Rightarrow 0^{i+3} S 0^{2j+2} \Rightarrow 0^{i+3} 111 0^{2j+2} = x' \in L$ perché:

La quantità di zeri a sinistra deve essere sempre 3 volte tante quante sono le coppie di zeri a destra

$$|0^i| = \frac{3}{2}|0^{2j}| \wedge |0^i| \in 6\mathbb{N} \rightarrow i = \frac{3}{2} \cdot 2j = 3j \wedge i \in 6\mathbb{N}$$
$$\rightarrow i = 3j \wedge i \in 6\mathbb{N}$$

Quindi se si aggiungono 3 zeri a sinistra e due a destra l'uguaglianza deve rimanere vera:

$$|0^{i+3}| = \frac{3}{2}|0^{2j+2}| \rightarrow i+3 = \frac{3}{2}(2j+2) = 3(j+1) \wedge i \in 6\mathbb{N}$$
$$\rightarrow i+3 = 3(j+1) \wedge i \in 6\mathbb{N}$$
$$\rightarrow i = 3(j+1) - 3 \wedge i \in 6\mathbb{N}$$
$$\rightarrow i = 3(j+1-1) \wedge i \in 6\mathbb{N}$$
$$\rightarrow i = 3j \wedge i \in 6\mathbb{N}$$

La condizione rimane valida.

Ho dimostrato che il linguaggio è context free. Ora devo dimostrare che il linguaggio non è regolare, per farlo dimostro che non vale il pumping lemma siccome è una condizione necessaria affinchè un linguaggio sia regolare. Per farlo deve valere la sua negazione:

$$\neg \text{Pumping Lemma} \Rightarrow L \notin RE$$

La negazione del pumping lemma è la seguente:

$$\forall k \in \mathbb{N}. \exists z \in L. |z| \geq k. \forall u,v,w \in \{0,1\}^*. \begin{cases} z = uvw \\ |uv| \leq k \\ |v| > 0 \end{cases} \wedge \ \exists i. \ uv^iw \notin L$$

Dimostrazione del linguaggio non RE

Prendo $k \in \mathbb{N}$ e considero una stringa nel linguaggio più lunga di k

$$z = 0^{l} 1^{3} 0^{2k} \quad \text{con } j = k$$

Le condizioni di appartenenza al linguaggio sono:

$$z \in L \Leftrightarrow |0^{l}| = \frac{3}{2}|0^{2k}| \wedge l \in 6\mathbb{N} \equiv l = 3k \wedge l \in 6\mathbb{N}$$

Cerco un indice per pompare la stringa che la faccia uscire dal linguaggio. Le suddivisioni possibili sono:

- $uv \in 0^{i}$

  $z_i = 0^{l+|v|(i-1)} 1^{3} 0^{2k}$

  - $i = 2 \rightarrow z = 0^{l+|v|} 1^{3} 0^{2k}$

    $|0^{l+|v|}| = \frac{3}{2}|0^{2k}| \Rightarrow l+|v| = 3k \Rightarrow l = 3k+|v|$ Viola le condizioni di appartenenza

    $l = 3k+|v| \neq 3k$

- $u \in 0^{i} \wedge v \in 1^{3}$ Questo caso è banale perchè per qualsiasi indice diverso da 1 il numero di uni è diverso da 3, quindi la stringa non è nel linguaggio.

- $u \in 0^{i}1^{3} \wedge v \in 0^{2k}$

  $z_i = 0^{l} 1^{3} 0^{2j+|v|(i-1)}$

  - $i = 2$

    $|0^{l}| = \frac{3}{2}|0^{2k+|v|}| \Rightarrow l = \frac{3}{2}(2k+|v|) \Rightarrow l = 3k+\frac{3}{2}|v|$ Viola le condizioni di appartenenza

    $l = 3k+\frac{3}{2}|v| \neq 3k$

Ho quindi dimostrato che il linguaggio non è regolare.

- **(14pt)** Si considerino le seguenti famiglie di linguaggi al variare di $m \in \mathbb{N}$ (NON da classificare)

$$A_m = \bigcup_{h \in \mathbb{N}} A_{m,h} \qquad B_m = \bigcup_{h \in \mathbb{N}} B_{m,h}$$

  Classificare il seguente linguaggio sull'alfabeto $\{0,1\}$:

$$C = \bigcup_{m>0} (A_m \cap B_m)$$

$$A_m = \{0^{n} 1^{m} 0^{h} \mid n \in 2m\mathbb{N}+m, h \in \mathbb{N}\}$$
$$B_m = \{0^{h} 1^{m} 0^{n} \mid n \in 3m\mathbb{N}+m, h \in \mathbb{N}\}$$

$$C = \bigcup_{m>0} \{A_m \cap B_m\}$$
$$= \{\{0^{n} 1^{m} 0^{h} \mid n \in 2m\mathbb{N}+m\} \cap \{0^{h} 1^{m} 0^{n} \mid n \in 3m\mathbb{N}+m\} \mid m>0, h \in \mathbb{N}\}$$
$$= \{0^{i} 1^{m} 0^{j} \mid m>0, i \in 2m\mathbb{N}+m, j \in 3m\mathbb{N}+m\}$$

Questo insieme ha una dipendenza fra tutti i gruppi di simboli: il primo gruppo di zeri dipende dal numero di uno e lo stesso vale per l'ultimo gruppo di zeri, quindi non è context free. Per dimostrarlo non deve valere il pumping lemma dei context free, cioè la condizione necessaria affinchè un linguaggio sia CF:

$\neg$Pumping Lemma $\rightarrow L \notin CF$

La negazione del pumping lemma dei context free è la seguente:

$$\forall k \in \mathbb{N} . \exists z \in L . |z| \geq k . \forall u, v, w, x, y . \begin{cases} z = uvwxy \\ |uvw| \leq k \\ |vx| > 0 \end{cases} \wedge \exists i \in \mathbb{N} . uv^i w x^i y \notin L$$

Condizioni di appartenenza

Le condizioni di appartenenza al linguaggio sono le seguenti:

$$z = 0^i 1^m 0^j \qquad m > 0 \wedge \exists h \in \mathbb{N} . i \in 2mh + m \wedge j \in 3mh + m$$

Pumping lemma

Prendo un qualsiasi $k \in \mathbb{N}$ e una stringa z nel linguaggio più lunga di k:

$$\begin{aligned} z &= 0^i 1^m 0^j \quad \text{con } h = k \wedge m = k+1 \\ &= 0^{2mk+m} 1^m 0^{3mk+m} \\ &= 0^{2k(k+1)+(k+1)} 1^{k+1} 0^{3k(k+1)+(k+1)} \\ &= 0^{(k+1)(2k+1)} 1^{k+1} 0^{(k+1)(3k+1)} \rightarrow |z| \geq k \end{aligned}$$

Quindi le condizioni di appartenenza sono:

$$i = (k+1)(2k+1) \wedge j = (k+1)(3k+1) \wedge m = k+1$$

Per ogni suddivisione in 5 parti di z cerco un indice i che faccia uscire dal linguaggio la stringa z con le parti v e x ripetute i volte. Le possibili suddivisioni sono le seguenti:

	$0^{(k+1)(2k+1)}$	|	1^{k+1}	|	$0^{(k+1)(3k+1)}$
1)	v x				
2)	v	x			
3)	v		x		
4)	v			x	
5)	v				x
6)			v x		
7)			v	x	
8)			v		x
9)				v	x
10)					v x

Le suddivisioni in cui una partizione si trova a cavallo tra due gruppi di simboli, cioè 2, 4, 7, 9, violano banalmente le condizioni di appartenenza perchè pompando la partizione a cavallo di due gruppi si cambia l'ordine dei simboli e quindi la stringa non è più nel linguaggio.

La suddivisione 5 viola banalmente le condizioni del linguaggio perchè tra le partizioni v e x c'è di mezzo un altro gruppo di simboli, quindi questo implica che per coprire quella distanza c'è bisogno che la partizione tra v e x (la partizione w) sia di lunghezza maggiore di k e questo viola il vincolo del pumping lemma per cui $|uvw| \leq k$

Per le altre suddivisioni cerco un indice che faccia uscire la stringa dal linguaggio:

1) $vx \in 0^{(k+1)(2k+1)}$

$z_i = 0^{(k+1)(2k+1)+|vx|(i-1)}\ 1^{k+1}\ 0^{(k+1)(3k+1)}$

- $i = 2 \rightarrow z_2 = 0^{(k+1)(2k+1)+|vx|}\ 1^{k+1}\ 0^{(k+1)(3k+1)}$

Verifico le condizioni di appartenenza

$z_2 \in L \iff \cancel{(k+1)(2k+1)}+|vx| = \cancel{(k+1)(2k+1)} \wedge (k+1)(3k+1) = (k+1)(3k+1) \checkmark \wedge k+1 = k+1 \checkmark$

$|vx| = 0$ × Viola le condizioni del pumping lemma $\rightarrow z_2 \notin L$

3) $v \in 0^{(k+1)(2k+1)}$ $x \in 1^{k+1}$

$z_i = 0^{(k+1)(2k+1)+|v|(i-1)}\ 1^{k+1+|x|(i-1)}\ 0^{(k+1)(3k+1)}$

- $i = 2 \rightarrow z_2 = 0^{(k+1)(2k+1)+|v|}\ 1^{k+1+|x|}\ 0^{(k+1)(3k+1)}$

Verifico le condizioni di appartenenza

$z_2 \in L \iff \cancel{(k+1)(2k+1)}+|v| = \cancel{(k+1)(2k+1)} \wedge (k+1)(3k+1) = (k+1)(3k+1) \checkmark \wedge \cancel{k+1}+|x| = \cancel{k+1}$

$z_2 \in L \iff |v| = 0 \wedge |x| = 0 \rightarrow |vx| = 0$ × Viola le condizioni del pumping lemma

Siccome la condizione per essere nel linguaggio non è mai vera, allora la stringa z_2 non è nel linguaggio.

$z_2 \notin L$

6) $vx \in 1^{k+1}$

$z_i = 0^{(k+1)(2k+1)}\ 1^{k+1+|vx|(i-1)}\ 0^{(k+1)(3k+1)}$

- $i = 2 \rightarrow z_2 = 0^{(k+1)(2k+1)}\ 1^{k+1+|vx|}\ 0^{(k+1)(3k+1)}$

Verifico le condizioni di appartenenza

$z_2 \in L \iff (k+1)(2k+1) = (k+1)(2k+1) \checkmark \wedge (k+1)(3k+1) = (k+1)(3k+1) \checkmark \wedge \cancel{k+1}+|vx| = \cancel{k+1}$

$z_2 \in L \iff |vx| = 0$ × Assurdo viola le condizioni del pumping lemma $\rightarrow z_2 \notin L$

8) $x \in 0^{(k+1)(3k+1)} \wedge v \in 1^{k+1}$

$z_i = 0^{(k+1)(2k+1)}\ 1^{k+1+|v|(i-1)}\ 0^{(k+1)(3k+1)+|x|(i-1)}$

- $i = 2 \rightarrow z_2 = 0^{(k+1)(2k+1)}\ 1^{k+1+|v|}\ 0^{(k+1)(3k+1)+|x|}$

Verifico le condizioni di appartenenza

$z_2 \in L \iff (k+1)(2k+1) = (k+1)(2k+1) \checkmark \wedge \cancel{(k+1)(3k+1)}+|x| = \cancel{(k+1)(3k+1)} \wedge \cancel{k+1}+|v| = \cancel{k+1}$

$z_2 \in L \iff |v| = 0 \wedge |x| = 0 \rightarrow |vx| = 0$ × Viola le condizioni del pumping lemma

$\Rightarrow z_2 \notin L$

10) $vx \in 0^{(k+1)(3k+1)}$

$$z_i = 0^{(k+1)(2k+1)} 1^{k+1} 0^{(k+1)(3k+1)+|vx|(i-1)}$$

- $i = 2 \rightarrow z_2 = 0^{(k+1)(2k+1)} 1^{k+1} 0^{(k+1)(3k+1)+|vx|}$

Verifico le condizioni di appartenenza

$$z_2 \in L \iff (k+1)(2k+1) \overset{\checkmark}{=} (k+1)(2k+1) \wedge \cancel{(k+1)(3k+1)} + |vx| = \cancel{(k+1)(3k+1)} \wedge k+1 \overset{\checkmark}{=} k+1$$

$$z_2 \in L \iff \underset{\times}{|vx| = 0}$$ Assurdo viola le condizioni del pumping lemma $\rightarrow z_2 \notin L$

Siccome per ogni suddivisione ho trovato un pompaggio che fa uscire la stringa dal linguaggio, ho dimostrato che C non è context free.

2° parte

Classificare nella teoria matematica della ricorsione i seguenti insiemi ed i loro complementi al variare di $n > 0$ dove necessario:[1] :

1. **(6pt)** Studiare gli insiemi $D_n = \mathrm{Dom}(\psi_n)$ (e i loro complementari), al variare di $n > 2$, dove:

$$\psi_n(x) = \begin{cases} \lceil\sqrt{n+x}\rceil & 4x+1 \in 2\mathbb{N}+1 \text{ e } 4x+1 \in W_x \text{ non deciso in meno di } n \text{ passi} \\ \uparrow & \text{Altrimenti} \end{cases}$$

$$\psi_n(x) = \begin{cases} \lceil\sqrt{n+x}\rceil & 4x+1 \in 2\mathbb{N}+1 \wedge \varphi_x(4x+1)\uparrow \text{ in meno di } n \text{ passi} \\ \uparrow & \text{altrimenti} \end{cases}$$

L'insieme D è l'insieme di tutti gli elementi su cui psi_n termina:

$$D_n = \{x \mid 4x+1 \in 2\mathbb{N}+1 \wedge \varphi_x(4x+1)\uparrow \text{ in meno di } n \text{ passi}\}$$

L'insieme D_n è ricorsivo perchè esiste una funzione caratteristica che riesce a dire se un elemento si trova o no nell'insieme:

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

Per dimostrare che D_n è ricorsivo fornisco lo pseudocodice che calcola la sua funzione caratteristica:

```
input(x, n)
z = 4*x+1

// Se 4x+1 non è un numero dispari allora tutta la condizione
// è falsa e quindi ritorno 0
if z % 3 ≠ 0 {
  return 0
}

costruisci phi_x // Esiste una procedura algoritmica che la calcola

// Controllo se phi termina in meno di n passi
for i = 0 to n {
  esegui il prossimo passo di phi_x(z)

  // Se termina prima di n passi ritorno 0
  if phi_x(z) ha terminato {
    return 0
  }
}

// Se non ha terminato in meno di n passi ritorno 1
return 1
```

Il complemento di D_n è il seguente

$$\overline{D_n} = \{x \mid 4x+1 \notin 2\mathbb{N}+1 \wedge \varphi_x(4x+1)\downarrow \text{ in meno di } n \text{ passi}\}$$

Siccome D_n è un insieme ricorsivo, allora è ricorsivo anche il suo complemento e la sua funzione caratteristica è:

$$F_{\overline{D_n}}(x) = \begin{cases} 1 & \text{se } x \in \overline{D_n} \\ 0 & \text{se } x \notin \overline{D_n} \end{cases} = \begin{cases} 0 & \text{se } x \in D_n \\ 1 & \text{se } x \notin D_n \end{cases}$$

Per dimostrare che è ricorsivo bisogna fornire lo pseudocodice che calcola la funzione caratteristica, però è analogo al codice di prima invertendo il return 1 con return 0

2. (12pt) Sia $R = \bigcap_{n>0} D_n$, studiare l'insieme R ed il suo complementare:

$$D_n = \{ x \mid 4x+1 \in 2\mathbb{N}+1 \land \varphi_x(4x+1)\uparrow \text{ in meno di } n \text{ passi} \}$$

$$R = \bigcap_{n>0} D_n$$

$$= \{ x \mid 4x+1 \in 2\mathbb{N}+1 \land \forall n>0 .\ \varphi_x(4x+1)\uparrow \text{ in meno di } n \text{ passi} \}$$

$$= \{ x \mid 4x+1 \in 2\mathbb{N}+1 \land \varphi_x(4x+1)\uparrow \}$$

$$\bar{R} = \{ x \mid 4x+1 \in 2\mathbb{N} \lor \varphi_x(4x+1)\downarrow \}$$

Intuitivamente R è produttivo perchè la divergenza non è decidibile. Per dimostrare che l'insieme è produttivo bisogna ridurlo funzionalmente al complemento di K:

$$\bar{K} \leq R \equiv K \leq \bar{R}$$

Definisco una funzione parziale ricorsiva su cui poi potrò applicare il teorema smn:

$$\psi(x,y) = \begin{cases} 1 & x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Questa funzione è parziale ricorsiva e fornisco l'algoritmo che la calcola

```
input(x,y)
costruisci phi_x
while true {
  esegui un passi di phi_x(x)
  if phi_x(x) ha terminato {
    return 1
  }
}
```

Siccome la funzione è parziale ricorsiva si può applicare il teorema smn:

$$\overset{smn}{\Rightarrow} \exists g \text{ totale ricorsiva} .\ \psi(x,y) = \varphi_{g(x)}(y)$$

Riduzione funzionale:

$$- x \in K \overset{def K}{\Rightarrow} \varphi_x(x)\downarrow$$

$$\overset{def \psi}{\Rightarrow} \forall y .\ \psi(x,y)\downarrow$$

$$\overset{smn}{\Rightarrow} \forall y .\ \varphi_{g(x)}(y)\downarrow$$

$$\Rightarrow \forall x .\ 4x+1 \in 2\mathbb{N} \ \varphi_{g(x)}(4x+1)\downarrow$$

$$\overset{def R}{\Rightarrow} g(x) \notin R$$

$$- x \notin K \overset{def \psi}{\Rightarrow} \forall y .\ \psi(x,y)\uparrow$$

$$\overset{smn}{\Rightarrow} \forall y .\ \varphi_{g(x)}(y)\uparrow$$

$$\Rightarrow \forall x .\ 4x+1 \in 2\mathbb{N}+1 \ \varphi_{g(x)}(4x+1)\uparrow$$

$$\overset{def R}{\Rightarrow} g(x) \in R$$

Quindi l'insieme R è produttivo

$\overline{R} = \{ x \mid 4x+1 \in 2\mathbb{N} \vee \varphi_x(4x+1)\downarrow \}$

Questo insieme è intuitivamente RE perchè si può decidere quando un elemento è nell'insieme tramite la funzione semicaratteristica:

$$\psi_{\overline{R}}(x) = \begin{cases} 1 & \text{se } x \in \overline{R} \\ \uparrow & \text{altrimenti} \end{cases}$$

Per dimostrare che questo insieme è RE fornisco l'algoritmo che calcola la sua semicaratteristica:

```
input(x)
z = 4*x+1
// Se l'input è pari so già che appartiene a R segnato
if z % 2 == 0 {
  return 1
}

// Se la macchina termina sull'input termina allora appartiene a R segnato
// Altrimenti c'è una divergenza
costruisci phi_x
while true {
  esegui il prossimo passo di phi_x(z)
  if phi_x(z) ha terminato {
    return 1
  }
}
```

Ho dimostrato che il complemento di R è RE ed essendo R Produttivo, allora il R segnato è creativo per definizione.

3. **(12pt)** Si considerino i seguenti insiemi (da non classificare):

$$X_n = \{ \lceil \sqrt{x} \rceil \mid n \in W_x \},$$

Studiare il seguente insieme (ed il suo complementare)

$$S = \bigcap_{n \in R} X_n$$

OPPURE [(**8pt**)] Studiate $S_1 = \bigcap_{n \in \mathbb{N}} X_n$

$S = \{ \lceil \sqrt{x} \rceil \mid n \in W_x \wedge n \in R \}$

$= \{ \lceil \sqrt{x} \rceil \mid \varphi_x(n)\downarrow \wedge n \in \{ i \mid 4i+1 \in 2\mathbb{N}+1 \wedge \varphi_i(4i+1)\uparrow \} \}$

$\overline{S} = \{ \lceil \sqrt{x} \rceil \mid \varphi_x(n)\uparrow \vee n \in \{ i \mid 4i+1 \in 2\mathbb{N} \vee \varphi_i(4i+1)\downarrow \} \}$

Essendo che n appartiene ad R ed R è produttivo, la condizione dell'insieme S non è decidibile, quindi anche S sarà produttivo. Per dimostrarlo bisogna ridurlo funzionalmente al complemento di K:

$\overline{K} \preceq S \equiv K \preceq \overline{S}$

$$\psi(x,y) = \begin{cases} 1 & ? \\ \uparrow & \text{altrimenti} \end{cases}$$

$$X_n = \{ \lceil \sqrt{x} \rceil \mid n \in W_x \}$$

$$S_1 = \bigcap_{n \in \mathbb{N}} X_n$$

$$= \{ \lceil \sqrt{x} \rceil \mid \forall n \in \mathbb{N} .\ n \in W_x \}$$

$$= \{ \lceil \sqrt{x} \rceil \mid \mathbb{N} \in W_x \}$$

Questo insieme è produttivo perchè non si dovrebbero controllare tutti gli infiniti input per sapere se il programma termina su tutti quanti. Per dimostrarlo riduco funzionalmente S_1 al complemento di K:

$$\overline{K} \leq S_1 \equiv K \leq \overline{S}_1$$

$$\psi(x, y) = \begin{cases} 1 & \varphi_x(x) \text{ non termina in meno di } y \text{ passi} \\ \uparrow & \text{altrimenti} \end{cases}$$

Questa funzione è parziale ricorsiva (pseudocodice di seguito) e quindi si può applicare smn:

```
input(x,y)
costruisci phi_x

for i = 0 to y {
  esegui il prossimo passo di phi_x(x)

  // Se termina prima di n passi ritorno 0
  if phi_x(x) ha terminato {
    while true
  }
}

return 1
```

- $x \in K \overset{\text{def } \psi}{\Rightarrow} \psi(x, y)\downarrow \Leftrightarrow \exists n \in \mathbb{N} \;\; y \in [0, n-1]$

$\overset{\text{smn}}{\Rightarrow} \varphi_{g(x)}(y)\downarrow \Leftrightarrow \exists n \in \mathbb{N} \;\; y \in [0, n-1]$

$\overset{\text{def } W_{g(x)}}{\Rightarrow} W_{g(x)} = [0, n-1] \neq \mathbb{N}$

$\overset{\text{def } S_1}{\Rightarrow} g(x) \notin S_1$

- $x \notin K \overset{\text{def } K}{\Rightarrow} \forall y .\ \varphi_x(x)$ non termina in y passi

$\overset{\text{def } \psi}{\Rightarrow} \forall y .\ \psi(x, y)\downarrow$

$\overset{\text{smn}}{\Rightarrow} \forall y .\ \varphi_{g(x)}(y)\downarrow$

$\overset{\text{def } W_{g(x)}}{\Rightarrow} W_{g(x)} = \mathbb{N}$

$\overset{\text{def } S_1}{\Rightarrow} g(x) \in S_1$

Quindi S_1 è produttivo

$\overline{S_1} = \{ \ulcorner \sqrt{x} \urcorner \mid \mathbb{N} \notin W_x \}$

Anche questo insieme è intuitivamente produttivo perchè bisognerebbe controllare tutti gli input.
Per dimostrarlo si riduce l'insieme al complemento di K.

$\overline{K} \preceq \overline{S_1} \equiv K \preceq S_1$

$$\psi(x, y) = \begin{cases} 1 & x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Questa funzione è parziale ricorsiva e si può applicare smn (dimostrato prima):

- $x \in K \overset{\text{def } \psi}{\Rightarrow} \forall y.\ \psi(x, y) \downarrow$

  $\overset{smn}{\Rightarrow} \forall y.\ \varphi_{g(x)}(y) \downarrow$

  $\overset{\text{def } W_{g(x)}}{\Rightarrow} W_{g(x)} = \mathbb{N}$

  $\Rightarrow g(x) \notin \overline{S_1}$

- $x \notin K \overset{\text{def } \psi}{\Rightarrow} \forall y.\ \psi(x, y) \uparrow$

  $\overset{smn}{\Rightarrow} \forall y.\ \varphi_{g(x)}(y) \uparrow$

  $\overset{\text{def } W_{g(x)}}{\Rightarrow} W_{g(x)} = \emptyset$

  $\Rightarrow g(x) \in \overline{S_1}$

Quindi il complemento di S_1 è produttivo.