# Problema 3.17

**Enunciato**: Sia $B = \langle M_1 \rangle, \langle M_2 \rangle, \ldots$ un linguaggio riconoscibile Turing che consiste di descrizioni di macchine di Turing. Dimostrare che esiste un linguaggio decidibile $C$ composto da descrizioni di macchine di Turing tale che ogni macchina descritta in $B$ ha una macchina equivalente in $C$ e viceversa.

**Soluzione**: Costruiamo il linguaggio $C$ come segue: per ogni macchina $M$ in $B$, definiamo una sua versione normalizzata $M'$ che:

1. Mantiene esattamente lo stesso comportamento di $M$ su ogni input
2. Ha una rappresentazione canonica (ad esempio, stati numerati sequenzialmente da 1, stati inutili rimossi)

Definiamo quindi $C = \langle M' \rangle \mid M' \text{ è la forma standard di una TM da } B$

Questa costruzione garantisce che:

- Per ogni macchina in $B$ esiste una macchina equivalente in $C$ (ottenuta dalla procedura di standardizzazione)
- Per ogni macchina in $C$ esiste una macchina equivalente in $B$ (per costruzione)
- $C$ è decidibile, poiché l'appartenenza può essere determinata verificando se una macchina è nella forma standard

Quindi abbiamo costruito un linguaggio decidibile $C$ con la proprietà richiesta.

# Problema 3.18

**Enunciato**: Dimostrare che un linguaggio è decidibile se e solo se esiste un enumeratore che enumera il linguaggio nell'ordine standard delle stringhe.

**Soluzione**: Assumiamo che l'ordine standard delle stringhe sia l'ordine lessicografico.

($\Rightarrow$) Se un linguaggio $L$ è decidibile: Esiste una TM $M$ che decide $L$. Possiamo costruire un enumeratore $E$ che:

1. Genera tutte le stringhe possibili in ordine lessicografico: $\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots$
2. Per ogni stringa $s$, usa $M$ per verificare se $s \in L$
3. Se $s \in L$, emette $s$; altrimenti, passa alla stringa successiva

Poiché $M$ termina sempre, questo enumeratore emetterà tutte le stringhe in $L$ nell'ordine standard.

($\Leftarrow$) Se un enumeratore $E$ enumera $L$ in ordine standard: Possiamo costruire una TM $M$ che decide $L$ come segue:

1. Dato un input $x$, $M$ simula $E$
2. Se $E$ emette $x$, $M$ accetta
3. Se $E$ emette una stringa che viene dopo $x$ nell'ordine standard (senza aver emesso $x$), $M$ rifiuta

Poiché $E$ enumera in ordine standard, se $x \in L$, $E$ emetterà eventualmente $x$. Se $x \notin L$, $E$ emetterà eventualmente una stringa che viene dopo $x$ senza aver emesso $x$. Quindi $M$ termina sempre e decide correttamente $L$.

# Problema 3.19

**Enunciato**: Dimostrare che ogni linguaggio riconoscibile Turing infinito ha un sottoinsieme decidibile infinito.

**Soluzione**: Sia $L$ un linguaggio riconoscibile Turing infinito, e sia $M$ una TM che riconosce $L$ .

Costruiamo un sottoinsieme decidibile $D$ di $L$ come segue:

1. Inizializziamo $D = \emptyset$
2. Simuliamo $M$ su tutte le stringhe in ordine lessicografico con un limite di tempo
3. Inizialmente impostiamo il limite di tempo a 1 passo
4. Se $M$ accetta una stringa $s$ entro il limite di tempo, aggiungiamo $s$ a $D$
5. Aumentiamo il limite di tempo di 1 per il prossimo ciclo di simulazione

Questo processo garantisce che:

- Ogni stringa in $D$ è in $L$ (perché $M$ la accetta)
- $D$ è infinito (perché $L$ è infinito, e troveremo eventualmente infinite stringhe che $M$ accetta)
- $D$ è decidibile (per qualsiasi stringa $s$, possiamo determinare se $s \in D$ simulando $M$ per un numero specifico di passi)

Formalmente, definiamo $D$ come: $D = s \mid M \text{ accetta } s \text{ entro } t(s) \text{ passi}$ dove $t(s)$ è una funzione calcolabile che associa ogni stringa al suo limite di tempo.

Poiché $D \subseteq L$, $D$ è infinito, e $D$ è decidibile, abbiamo dimostrato che ogni linguaggio riconoscibile Turing infinito ha un sottoinsieme decidibile infinito.

# Problema 3.20

**Enunciato**: Dimostrare che le macchine di Turing a nastro singolo che non possono scrivere sulla porzione del nastro contenente la stringa di input riconoscono solo linguaggi regolari.

**Soluzione**: Consideriamo una TM a nastro singolo $M$ con la restrizione che non può modificare la porzione di nastro contenente l'input.

La dimostrazione si basa sul fatto che tale macchina, non potendo modificare l'input, può utilizzare solo un numero finito di configurazioni durante la lettura dell'input:

1. Definiamo una "configurazione" di $M$ durante la lettura dell'input come la coppia (stato corrente, posizione della testina nell'input).
2. Il numero di configurazioni possibili è finito poiché:
   - Gli stati di $M$ sono in numero finito
   - Le posizioni possibili nell'input sono al massimo $n$ (lunghezza dell'input)
3. Poiché $M$ non può modificare l'input, il suo comportamento è completamente determinato dalla configurazione corrente.
4. Possiamo costruire un automa a stati finiti $F$ che simula $M$:
   - Gli stati di $F$ sono le configurazioni di $M$
   - Le transizioni di $F$ corrispondono a come $M$ passa da una configurazione all'altra
   - Lo stato iniziale di $F$ è la configurazione iniziale di $M$
   - Gli stati accettanti di $F$ sono le configurazioni in cui $M$ accetta
5. Tale automa riconoscerà esattamente lo stesso linguaggio di $M$.

Poiché gli automi a stati finiti riconoscono solo linguaggi regolari, anche il linguaggio riconosciuto da $M$ deve essere regolare.

# Problema 3.21

**Enunciato**: Sia $c_1x^n + c_2x^{n-1} + \cdots + c_nx + c_{n+1}$ un polinomio con una radice in $x = x_0$. Sia $c_{max}$ il valore assoluto massimo tra i $c_i$. Dimostrare che $|x_0| < (n+1)\frac{c_{max}}{|c_1|}$.

**Soluzione**: Poiché $x_0$ è una radice del polinomio, abbiamo:
$c_1x_0^n + c_2x_0^{n-1} + \cdots + c_nx_0 + c_{n+1} = 0$

Riorganizzando per isolare il termine con $x_0^n$: $c_1x_0^n = -(c_2x_0^{n-1} + \cdots + c_nx_0 + c_{n+1})$

Applicando il valore assoluto ad entrambi i lati: $|c_1||x_0|^n = |c_2x_0^{n-1} + \cdots + c_nx_0 + c_{n+1}|$

Dalla disuguaglianza triangolare: $|c_1||x_0|^n \leq |c_2||x_0|^{n-1} + \cdots + |c_n||x_0| + |c_{n+1}|$

Utilizzando $c_{max}$ come limite superiore per i coefficienti:
$|c_1||x_0|^n \leq c_{max}(|x_0|^{n-1} + \cdots + |x_0| + 1)$

La somma a destra contiene esattamente $n+1$ termini. Considerando i due casi:

**Caso 1**: Se $|x_0| \leq 1$, l'affermazione è vera, poiché $(n+1)\frac{c_{max}}{|c_1|} > 1$ se $c_{max} \geq \frac{|c_1|}{n+1}$.

**Caso 2**: Se $|x_0| > 1$, allora: $|x_0|^{n-1} + \cdots + |x_0| + 1 < (n+1)|x_0|^{n-1}$

Quindi: $|c_1||x_0|^n < c_{max} \cdot (n+1)|x_0|^{n-1}$

Dividendo per $|x_0|^{n-1}$: $|c_1||x_0| < c_{max} \cdot (n+1)$

Dividendo per $|c_1|$: $|x_0| < (n+1)\frac{c_{max}}{|c_1|}$

Ciò dimostra la disuguaglianza stretta richiesta.