Dimostriamo adesso una sorta di inverso del teorema di limitazione alla crescita.

Presentiamo prima due lemmi preliminari.

Lemma 1.

La funzione $h(x_1, \dots, x_m) = \sum_{i=1}^{m} x_i$ appartiene alla classe $\mathcal{L}_1$.

Dimostrazione.

Il risultato segue dall'osservazione che la funzione h è calcolata dal programma che segue e che ha una profondità di nidificazione eguale ad 1.

```
Y ← X_1
LOOP X_2
    Y ← Y + 1
END
LOOP X_3
    Y ← Y + 1
END
  ⋮
LOOP X_m
    Y ← Y + 1
END
```

Lemma 2

Per ogni $n \geq 1$ e $k \in N$, la funzione

$$g(x_1, \dots, x_m) = f_n^{(k)}\left(\sum_{i=1}^{m} x_i\right) \text{ appartiene ad } \mathcal{L}_n .$$

Dimostrazione

La funzione g è calcolata dal programma che segue che appartiene ad L_n.

$$Y \leftarrow \sum_{i=1}^{m} X_i$$

$$\left.\begin{array}{l} Y \leftarrow f_n(Y) \\ Y \leftarrow f_n(Y) \\ \vdots \\ Y \leftarrow f_n(Y) \end{array}\right\} k$$

Teorema

Sia P un programma ciclo che calcola $g(x_1, \ldots, x_m)$. Sia $n \geq 2$ ed esista una costante k tale che:

$$T_P(x_1, \ldots, x_m) \leq f_n^{(k)}(\max\{x_1, \ldots, x_m\})$$

allora $g \in \mathcal{L}_n$.

Dimostrazione

Il programma P sia formato da z istruzioni di L, $S_1, \ldots, S_z$.

Per dimostrare il teorema costruiremo un nuovo programma P' che calcola g ed appartiene ad L_n.

Tale nuovo programma sarà costruito associando a ciascuna istruzione S_i di P un opportuno sottoprogramma Σ_i.

A ciascun Σ_i sarà associata una variabile T_i che assume solo i valori 0 e 1 e che può essere "pensata" come un "interruttore".

Ad ogni istante una sola di tali variabili avrà il valore 1 segnalandoci quale istruzione di P è simulata in quel momento.

All'inizio porremo quindi $T_1 = 1$ e $T_i = 0$ per $2 \leq i \leq z$.

Il programma P' è quello indicato più sotto nel quale il sottoprogramma Q ha lo scopo di assegnare alla variabile C il valore $f_n^{(k)}\left(\sum_{i=1}^{m} x_i\right)$. Osserviamo che, per il lemma appena dimostrato, $Q \in L_n$.

La variabile C ha la funzione di "contatore" e controlla l'esecuzione del programma P'.

Per ipotesi il tempo di calcolo del programma P è non superiore a $f_n^{(k)}(\max\{x_1, \dots, x_m\})$.

Essendo, per proprietà delle f,

$$f_n^{(k)}(\max\{x_1, \dots, x_m\}) \leq f_n^{(k)}\left(\sum_{i=1}^{m} x_i\right)$$

possiamo essere sicuri che il corpo del programma P', che è interno al ciclo LOOP C, può essere eseguito tutte le volte che serve.

```
Q
T₁ ← 1
LOOP C
    Σ₁
    Σ₂
    ⋮
    Σ_z
END
```

Programma P'

Vediamo adesso come sono fatti i sottoprogrammi Σ_i di P' associati alle istruzioni S_i di P.

Distingueremo fondamentalmente due casi, quello in cui S_i è una istruzione di azzeramento, incremento o assegnazione e quello in cui S_i è una istruzione LOOP (associata ad una istruzione END che assumiamo essere la j-esima : S_j).

Consideriamo il primo caso.

- Se S_i è un'istruzione di azzeramento, incremento o assegnazione allora il sottoprogramma Σ_i corrispondente è il seguente:

$$
\begin{array}{l}
\text{LOOP } T_i \\
\qquad T_i \leftarrow 0 \\
\qquad T_{i+1} \leftarrow 1 \\
\qquad S_i \\
\text{END}
\end{array}
$$

Il ciclo viene eseguito una sola volta perché inizialmente $T_i = 1$.
Subito dopo T_i viene azzerato e viene "attivato" T_{i+1} ponendolo eguale a 1.
Viene infine eseguita l'istruzione S_i.
Adesso siamo pronti ad eseguire l'istruzione successiva.

- Consideriamo adesso il caso in cui S_i è l'istruzione LOOP k (sapendo che l'END ad essa associata è l'istruzione j-esima S_j).
Il sottoprogramma Σ_i corrispondente ad S_i è il seguente:

LOOP T_i
  $T_i \leftarrow 0$
  $U_i \leftarrow 1$
  $V_i \leftarrow K$
END
LOOP U_i
  $U_i \leftarrow 0$
  $T_{j+1} \leftarrow 1$
  $W_i \leftarrow 0$
END
LOOP V_i
  $V_i \leftarrow W_i$
  $W_i \leftarrow W_i + 1$
  $T_{j+1} \leftarrow 0$
  $T_{i+1} \leftarrow 1$
END

alla fine di questo ciclo V_i avrà un valore che è inferiore di 1 a quello di partenza.

I tre sottocicli hanno essenzialmente lo scopo di:
1) assegnare il valore K ad un "contatore" V_i;
2) accendere l'"interruttore" dell'istruzione che segue END
3) accendere l'"interruttore" della prima istruzione del corpo del ciclo e diminuire di 1 il valore di K (cioè di V_i)

Alla fine del sottoprogramma Σ_i (che simula LOOP k) saremo quindi rinviati

- all'"istruzione" Σ_{j+1}, cioè a quella che segue l'END corrispondente perchè è "acceso" l'interruttore T_{j+1} nel caso in cui $k = 0$
- all'"istruzione" Σ_{i+1}, cioè la prima interna al ciclo, nel caso in cui $k \neq 0$ ma in questo caso con un valore assegnato a V_i diminuito di 1 (memorizzando in tal modo il fatto che <u>un</u> ciclo è stato "innescato".

  (anche questo viene attuato "accendendo" un interruttore, T_{i+1})

Vediamo adesso in che modo questo viene ripreso nel sottoprogramma che simula l'istruzione END del programma originario.

– Il sottoprogramma Σ_j corrispondente all'istruzione END (S_j) che chiude l'istruzione LOOP (S_i) è:

$$
\begin{array}{l}
\textbf{LOOP } T_j \\
\qquad T_j \quad \leftarrow 0 \\
\qquad T_{j+1} \leftarrow 1 \\
\textbf{END} \\
\textbf{LOOP } V_i \\
\qquad T_{j+1} \leftarrow 0 \\
\qquad U_i \quad \leftarrow 1 \\
\textbf{END}
\end{array}
$$

Il primo ciclo spegne "l'interruttore" T_j e "accende" quello dell'istruzione che segue l'END in P, la $(j+1)$esima.

Il secondo ciclo a seconda del valore di V_i (che rappresenta il numero di volte che il ciclo originario deve <u>ancora</u> essere eseguito) darà l'ordine di eseguire l'istruzione $(j+1)$esima se $V_i = 0$, oppure darà l'ordine di eseguire ancora una volta il secondo ciclo di Σ_i (che simula l'istruzione LOOP di P), nel caso in cui $V_i \neq 0$.

Abbiamo pertanto ricondotto il programma originario alla successione di una serie di programmi di profondità di nidificazione 1 a loro volta inseriti in un ciclo LOOP.

P' quindi è dato dalla successione di un sotto-programma Q che calcola $C = f_n^{(k)}(\sum_{i=1}^{m} x_i)$ che appartiene ad L_n e dal sottoprogramma che esegue effettivamente la simulazione e che appartiene ad L_2.

Si ha quindi che $P' \in L_n$.

Osserviamo infine che

- l'avere assegnato a C un valore non minore del tempo di calcolo di P permette di eseguire tutte le operazioni necessarie;
  nello stesso tempo
- il meccanismo di accensione e spegnimento degli "interruttori" non permette di fare cose diverse da quelle richieste da P,

quindi effettivamente P' calcola la stessa funzione calcolata da P.