(F1X) Teoria dell'Informazione e della Trasmissione

Complessità di Kolmogorov

Docente: *Nicolò Cesa-Bianchi* versione 1 maggio 2016

La teoria dell'informazione secondo Shannon, introdotta nel 1948, si basa sulla teoria della probabilità che era stata assiomatizzata da Kolmogorov non molti anni prima, nel 1933. Kolmogorov non era però completamente soddisfatto dalla formalizzazione di Shannon del concetto di informazione. La critica principale era che essa permette di misurare la quantità di informazione soltanto di variabili casuali e non di oggetti deterministici. Attraverso l'entropia, Shannon può calcolare l'informazione associata ad una distribuzione di probabilità su testi o immagini, ma non l'informazione contenuta in un dato testo o in una data immagine.

Assumendo che un oggetto discreto possa essere rappresentato come una sequenza di bit, Kolmogorov si chiede come misurare quanta informazione è contenuta in una data sequenza $x \in \{0,1\}^+$ di bit. In analogia con l'entropia $H(X)$, che misura la lunghezza media del miglior codice compressore per i valori di una data variabile casuale X, intorno al 1965 Kolmogorov definisce la quantità di informazione contenuta in una sequenza x come la lunghezza del più breve programma che stampa x e si arresta. Questa lunghezza costituisce la complessità di Kolmogorov di x. Nello stesso periodo, concetti analoghi erano stati formulati indipendentemente da due altri matematici, Solomonoff (americano di origini russe) e Chaitin (argentino-americano).

Per poter parlare di programmi dobbiamo definire un linguaggio e un interprete in grado di eseguire i programmi scritti nel linguaggio. Sia quindi $\mathcal{M}$ un interprete per un dato linguaggio di programmazione $\mathcal{L}$ in grado di rappresentare tutte le funzioni calcolabili. Un programma nel linguaggio $\mathcal{L}$ è rappresentato come una sequenza $p \in \{0,1\}^+$ di bit. Sia $\mathcal{M}(p)$ l'output generato eseguendo p tramite l'interprete $\mathcal{M}$. Se p non termina, allora l'output è indefinito. Se p termina, allora l'output può essere la sequenza vuota o una qualunque sequenza finita di bit. Indichiamo con $\ell(p)$ la lunghezza in bit del programma p.

Dato che vogliamo formulare una teoria che non dipende da uno specifico linguaggio di programmazione, non faremo nessuna assunzione sulla struttura di un programma p. In altre parole, vediamo un programma come una sequenza di bit senza assumere nulla di come i bit della sequenza possano essere raggruppati in blocchi lessicali (istruzioni, operandi, eccetera).

Al fine di formulare la complessità di Kolmogorov in modo tale da poterla mettere in relazione con la teoria dell'informazione di Shannon, ci concentriamo su una particolare classe di interpreti: gli interpreti liberi da prefisso. Un interprete è libero da prefisso quando per ogni programma p tale che $\mathcal{M}(p)$ è definito (ovvero $\mathcal{M}$ esegue p e poi termina) e per ogni $q \in \{0,1\}^+$, vale $\mathcal{M}(pq) = \mathcal{M}(p)$. In altre parole, quando un interprete libero da prefisso esegue un programma terminante è sempre in grado di riconoscere la fine del codice, indipendentemente dalla stringa di bit successiva. Ricordiamo che non facciamo nessuna assunzione sul linguaggio. In particolare, il linguaggio non possiede necessariamente un'istruzione di fine programma. Sia $\mathcal{M}$ un interprete libero da prefisso per un dato linguaggio $\mathcal{L}$. Sia $\mathcal{P}$ l'insieme dei programmi terminanti per $\mathcal{M}$. Ovvero, $\mathcal{P}$ contiene tutti e soli i programmi p tali che $\mathcal{M}(p)$ termina. Allora $\mathcal{P}$ è un insieme libero da prefisso. Ovvero, $p \in \mathcal{P}$

implica che non può esistere in $\mathcal{P}$ un altro programma pq per qualunque $q \in \{0,1\}^+$. Nel seguito, usiamo il termine autodelimitante per identificare un qualunque programma dell'insieme $\mathcal{P}$.

Dato un linguaggio $\mathcal{L}$ e un interprete $\mathcal{M}$, mostriamo come un qualsiasi programma p nel linguaggio $\mathcal{L}$ diventi autodelimitante con una semplice procedura che aggiunge qualche bit al programma stesso. In altre parole, trasformiamo p in p_{sd}, con $\ell(p_{\text{sd}}) > \ell(p)$, tale che p_{sd} eseguito dall'inteprete $\mathcal{M}$ lievemente modificato dia lo stesso output di p eseguito dall'interprete $\mathcal{M}$ originale. Vedremo anche che l'incremento può essere reso assai modesto, ovvero $\ell(p_{\text{sd}}) = \big(1+o(1)\big)\ell(p)$, dove $o(1) \to 0$ per $\ell(p) \to \infty$.

Dato che non lavoriamo su in linguaggio specifico, non possiamo assumere né che il linguaggio contenga un'istruzione di fine programma, né —più in generale— che un particolare blocco di bit non compaia mai all'interno di un programma, e quindi possa essere utilizzato come indicatore di fine programma. Un modo elementare per rendere un programma autodelimitante è quello di raddoppiare ogni bit e aggiungere 01 in coda. Per esempio, $p = 10010$ diventa $p_{\text{sd}} = 110000110001$. È facile verificare come un inteprete libero da prefisso possa riconoscere la fine di p_{sd}. Questa codifica però è inefficiente, dato che il programma autodelimitante ha lunghezza $\ell(p_{\text{sd}}) = 2\ell(p) + 2$.

Sia $[n]_2 \in \{0,1\}^+$ la rappresentazione binaria di un numero $n \in \mathbb{N}$. La seguente è una codifica autodelimitante più efficiente per un programma $p \in \{0,1\}^+$ con $\ell(p) > 1$,

$$p_{\text{sd}} = \big[\ell^k(p)\big]_2\, 0 \big[\ell^{k-1}(p)\big]_2\, 0 \cdots \big[\ell^2(p)\big]_2\, 0\, \big[\ell(p)\big]_2\, 0\, p\, 1$$

dove k è definito come $k = \min\big\{i \in \mathbb{N} : \ell^i(p) = 2\big\}$. Procedendo da destra a sinistra troviamo il blocco $p\,1$, cioè il programma p seguito dal bit 1. Poi troviamo una sequenza di blocchi della forma $\big[\ell^i(p)\big]_2\, 0$ dove $\ell^1(p) = \ell(p)$ e $\ell^i(p) = \ell\big(\ell^{i-1}(p)\big)$. Quindi se $\ell(p)$ è la lunghezza in bit di p, $\ell^2(p)$ è la lunghezza in bit del numero che rappresenta la lunghezza di p, e così via.

Facciamo un esempio concreto assumendo che $p \in \{0,1\}^+$ sia lungo cento bit. Allora la codifica autodelimitante di p è data da

$$p_{\text{sd}} = \underbrace{11}_{[3]_2}\, 0\, \underbrace{111}_{[7]_2}\, 0\, \underbrace{1100100}_{[100]_2}\, 0\, p\, 1$$

Infatti, il blocco che precede $p\,1$ è la codifica binaria di cento (la lunghezza in bit di p) seguita da 0. Il blocco ancora precedente è la codifica binaria di sette (la lunghezza in bit di $\ell(p)$) seguita da 0. Infine il primo blocco è la codifica binaria di tre (la lunghezza in bit di $\ell^2(p)$) seguita da 0.

Il primo blocco della codifica autodelimitante di ogni p può quindi essere $10\,0$ quando p è lungo due bit, oppure $11\,0$ come nell'esempio precedente. Possiamo quindi usare 00 per codificare il programma $p = 0$ e 01 per codificare il programma $p = 1$.

È chiaro che possiamo aggiungere qualche riga di codice ad un interprete $\mathcal{M}$ per $\mathcal{L}$ in modo da fargli eseguire i programmi codificati come sopra.

La codifica autodelimitante di p fatta con la tecnica appena spiegata ha lunghezza

$$\ell(p_{\text{sd}}) = \sum_{i=1}^{k} \big(\ell^i(p) + 1\big)$$

Il costo in bit per rendere p autodelimitante è quindi $\ell(p_{\text{sd}}) - \ell(p)$. È possibile dimostrare che questa quantità cresce appena più velocemente di $\log_2 \ell(p)$. Dato che $\log_2 n = o(n)$, possiamo assumere che per ogni $p \in \{0,1\}^+$, $\ell(p_{\text{sd}}) = \big(1 + o(1)\big)\ell(p)$.

Siamo ora pronti per definire formalmente la complessità di Kolmogorov $K_{\mathcal{L}}(x)$ di una sequenza arbitraria $x \in \{0,1\}^+$ relativemente al linguaggio $\mathcal{L}$ e all'interprete $\mathcal{M}$ per programmi terminanti e autodelimitanti in $\mathcal{L}$,

$$K_{\mathcal{L}}(x) = \min\{\ell(p) \,:\, p \in \mathcal{P},\, \mathcal{M}(p) = x\} \ .$$

Prima di addentrarci nello studio di questa funzione è utile fare la seguente osservazione.

Fatto 1 *Per ogni intero n e per ogni $x \in \{0,1\}^n$, $K_{\mathcal{L}}(x) \leq c + n + o(n)$*

Dimostrazione. Possiamo stampare x col programma p della forma `Stampa` x di lunghezza $\ell(p) = c + n$, dove c è la lunghezza della codifica dell'istruzione di stampa. Possiamo poi rendere questo programma autodelimitante aggiungendo un numero di bit pari a $o(n)$. □

Una stringa del tipo $x = 1 \cdots 1 \in \{0,1\}^n$ ha una complessità molto minore. Infatti, possiamo stampare x col programma p della forma `For` $i = 1$ `to` n `stampa` 1. Questo programma ha lunghezza $\ell(p) = c' + \log_2(n)$, dove c' è la lunghezza della codifica delle istruzioni di ciclo for e di stampa e $\log_2(n)$ è lo spazio occupato per scrivere la costante n nel programma. Come prima, possiamo rendere questo programma autodelimitante aggiungendo un numero $o(\log n)$ di bit. Quindi, $K_{\mathcal{L}}(x) \leq \big(1 + o(1)\big)\log_2(n)$

Anche se per alcune stringhe x di lunghezza n vale $K_{\mathcal{L}}(x) \leq \big(1 + o(1)\big)\log_2(n)$, il seguente risultato dimostra che per la stragrande maggioranza delle stringhe $x \in \{0,1\}^n$ vale $K_{\mathcal{L}}(x) = \Omega(\log n)$.

Fatto 2 *Per ogni intero n e per ogni $c > 0$, ci sono almeno $2^n - 2n^c + 1$ stringhe di lunghezza n tali che $K_{\mathcal{L}}(x) > c\log_2(n)$.*

Dimostrazione. Per ogni $m < n$, il numero di programmi di lunghezza al più $n - m$ è

$$\sum_{i=0}^{n-m} 2^i = 2^{n-m+1} - 1 \ .$$

Quindi restano almeno $2^n - 2^{n-m+1} + 1$ stringhe $x \in \{0,1\}^n$ tali che $K_{\mathcal{L}}(x) > n - m$. Scegliendo $m = n - c\log_2(n)$ otteniamo la tesi. □

Teorema 3 (Non computabilità di K) *Non esiste un programma che calcola $K(x)$ per qualunque $x \in \{0,1\}^+$.*

Dimostrazione. Per assurdo, assumiamo che esista un programma `Kol` che stampa la complessità di Kolmogorov di ogni stringa in ingresso. Allora si consideri il programma seguente

```
Paradox
    Per ogni stringa s in ordine lessicografico
        Se Kol(s) >= n then {
            stampa s
            exit
        }
```

Questo programma stampa la prima stringa s tale che $K(s) \geq n$. Chiaramente, se `Kol` termina per ogni s, allora il programma termina per ogni valore di n. Infatti, non possono esserci infinite stringhe s tali che $K(s) < n$ in quanto il numero di programmi con meno di n bit è finito e ogni programma stampa una sola stringa. Inoltre, detta p_n la stringa di bit che rappresenta il programma abbiamo che

$$\ell(p_n) \leq C + \big(1 + o(1)\big) \log_2(n)$$

dove C è la lunghezza della codifica autodelimitante delle istruzioni del programma (comprese le istruzioni di `Kol`, che non dipendono da n) e $\big(1 + o(1)\big) \log_2(n)$ è la lunghezza della codifica autodelimitante della costante n. Se s_n è l'output di `Paradox`, allora $K(s_n) \leq \ell(p_n) \leq C + \big(1 + o(1)\big) \log_2(n)$. Dato che n cresce più velocemente di $C + \big(1 + o(1)\big) \log_2(n)$, esiste un valore di n tale che

$$C + \big(1 + o(1)\big) \log_2(n) < n \ .$$

Questo dimostra che, per questo n, $K(s_n) < n$. Questo contraddice il fatto che s_n sia l'output di p_n. Quindi l'output di `Paradox` non può essere sempre definito, il che significa che `Kol` non termina su ogni input. □

Dimostriamo un importante risultato che ne stabilisce la sua universalità. Ovvero, la complessità di Kolmogorov di x misurata rispetto a due linguaggi di programmazione $\mathcal{L}$ e $\mathcal{L}'$ differisce per una costante c indipendente da x.

Teorema 4 (Invarianza di K) *Per ogni coppia $\mathcal{L}, \mathcal{L}'$ di linguaggi in grado di rappresentare tutte le funzioni calcolabili esiste una costante $c > 0$ tale che per ogni $x \in \{0,1\}^+$,*

$$\big|K_{\mathcal{L}}(x) - K_{\mathcal{L}'}(x)\big| \leq c \ .$$

Dimostrazione. Sia $p_{\mathcal{L}'} \in \mathcal{L}$ il programma autodelimitante in $\mathcal{L}$ che simula l'interprete $\mathcal{M}'$ per il linguaggio $\mathcal{L}'$ e lo applica all'input ottenuto leggendo i successivi bit della stringa. Quindi $\mathcal{M}$ con input $p_{\mathcal{L}'}p$ simula $\mathcal{M}'$ su input p. Ovvero, $\mathcal{M}(p_{\mathcal{L}'}p) = \mathcal{M}'(p)$. Allora, se $p_x \in \{0,1\}^+$ è il più breve programma autodelimitante in $\mathcal{L}'$ per stampare x, ovvero $K_{\mathcal{L}'}(x) = \ell(p_x)$, abbiamo che $\mathcal{M}(p_{\mathcal{L}'}p_x) = \mathcal{M}'(p_x) = x$. Quindi $p_{\mathcal{L}'}p_x$ è un programma autodelimitante per stampare x, il che implica $K_{\mathcal{L}}(x) \leq \ell(p_x) + \ell(p_{\mathcal{L}'}) = K_{\mathcal{L}'}(x) + c$ dato che $\ell(p_{\mathcal{L}'})$ non dipende da x. □

Quindi, d'ora in poi ometteremo l'indice $\mathcal{L}$ in $K_{\mathcal{L}}$ sapendo che $K(x)$ è sempre definita a meno di una costante additiva.

Introduciamo ora una variante di K che ci servirà per mettere in relazione K con l'entropia H. Dato un linguaggio $\mathcal{L}$, definiamo un interprete $\mathcal{M}$ che prende due argomenti in input. In particolare, $\mathcal{M}(p, x)$ è l'output generato dall'esecuzione del programma p con input x. Per ogni coppia di

sequenze $x, y \in \{0,1\}^+$, la complessità condizionata di Kolmogorov $K(x \mid y)$ è definita come

$$K_{\mathcal{L}}(x \mid y) = \min\{\ell(p) \,:\, p \in \mathcal{P},\, \mathcal{M}(p, y) = x\} \ .$$

Dato che vale un teorema di invarianza esattamente come per K, nel seguito scriveremo semplicemente $K(x \mid y)$.

Chiaramente, per ogni $x, y \in \{0,1\}^+$ vale che $K(x \mid y) \leq K(x)$. Infatti, un programma p che stampa x può essere utilizzato per stampare x ignorando l'input y. Inoltre, $K\big(x \mid \ell(x)\big) \leq \ell(x) + c$. Infatti, un programma per stampare x è "Stampa x la cui lunghezza è fornita in input". La lunghezza di questo programma è $\ell(x) + c$. Nel rendere il programma autodelimitante possiamo ignorare la stringa x nel codice. Infatti, la lunghezza di questa stringa è data come input, quindi l'interprete sa esattamente quanti bit occupa nel codice. Quindi il programma autodelimitante per stampare x avrà lunghezza maggiorata da $\ell(x) + c + c'$ (dove c' sono i bit aggiunti per rendere autodelimitante il blocco di c bit).

In generale, vale

$$K(x) \leq K(y) + K(x \mid y) \ . \tag{1}$$

Infatti, se p è il più breve programma autodelimitante per stampare x data y, allora possiamo ottenere un programma autodelimitante per stampare x senza conoscere y semplicemente aggiungendo a p il codice autodelimitante per generare y.

In modo del tutto analogo: se p è il più breve programma autodelimitante per stampare x data $\ell(x) = n$, allora un programma autodelimitante per stampare x senza conoscere n è ottenibile modificando p in modo da includervi una codifica autodelimitante di n. Se usiamo una codifica efficiente, come quella vista prima, questo aumenta la lunghezza di p di una quantità $\log_2(n) + o(\log n) = \mathcal{O}(\log n)$. Ciò dimostra che, per ogni $x \in \{0,1\}^n$,

$$K(x) \leq K\big(x \mid n\big) + \mathcal{O}(\log n) \ . \tag{2}$$

Dimostriamo ora la relazione fra K e H. In particolare, dimostreremo che se $X^n = (X_1, \ldots, X_n)$ è generata da una sorgente $\langle \mathcal{X}, q\rangle$ con entropia $H(X)$, allora $\frac{1}{n} K(X^n) \to H(X)$.

Teorema 5 *Sia $\langle \mathcal{X}, q\rangle$ una sorgente con $\mathcal{X} = \{0,1\}$ e sia $\langle \mathcal{X}^n, q^n\rangle$ la sua versione a blocchi, dove $q^n(x_1, \ldots, x_n) = \prod_i q(x_i)$. Allora*

$$H(X) \leq \frac{1}{n} \mathbb{E}\big[K(X^n)\big] \leq H(X) + o(1)$$

dove $H(X)$ è l'entropia di $\langle \mathcal{X}, q\rangle$.

Dimostrazione. Sia $q = \mathbb{P}(X = 1)$. Allora $H(X)$ è l'entropia binaria $H(q) = -q \log_2 q - (1 - q) \log_2(1 - q)$.

L'insieme dei più brevi programmi autodelimitanti per generare i blocchi $x \in \{0,1\}^n$ di simboli sorgente dato n forma un codice istantaneo $C_n : \{0,1\}^n \to \{0,1\}^+$, dove $C_n(x) = p_x$ con $\ell(p_x) = K(x \mid n)$. Quindi la lunghezza media delle parole di questo codice deve essere maggiore dell'entropia della sorgente,

$$\mathbb{E}\big[\ell_{C_n}(X^n)\big] = \mathbb{E}\big[K(X^n \mid n)\big] \geq H(X^n) = n\, H(q) \ .$$

Dato che $K(x) \geq K(x \mid n)$ per ogni $x \in \{0,1\}^n$, questo dimostra che $\frac{1}{n}\mathbb{E}\big[K(X^n)\big] \geq H(q)$.

Per dimostrare l'altra relazione, verifichiamo che per stampare X^n estratta dalla sorgente esiste un programma la cui lunghezza media dipende dall'entropia della sorgente. Sia $s(x)$ il numero di volte in cui il bit 1 compare in $x \in \{0,1\}^n$, ovvero

$$s(x) = \sum_{i=1}^{n} x_i \ .$$

Un programma autodelimitante per stampare x, dato n come input, è il seguente. Supponiamo che $s(x) = k$ e che x sia l'i-esima sequenza nell'ordinamento lessicografico di tutte le sequenze di n bit contenenti k ripetizioni del bit 1. Allora il programma è `stampa l'`i`-esima sequenza fra quelle che contengono` k `ripetizioni del bit` 1. La lunghezza di questo programma autodelimitante p è

$$\ell(p) = c + \ell(\text{codifica autodelimitante di } k) + \ell\Big(\text{codifica di } \binom{n}{k}\Big) \ .$$

È importante notare che non devo rendere autodelimitante la codifica di $\binom{n}{k}$ dato che n è noto e k è stato precedentemente codificato in modo autodelimitante. Usando il maggiorante

$$\log_2 \binom{n}{k} \leq n\, H\left(\frac{k}{n}\right)$$

dove $H(k/n)$ è l'entropia binaria, otteniamo

$$\ell(p) \leq c + \log_2 k + o(\log k) + n\, H\left(\frac{k}{n}\right) \ .$$

Quindi, ricordando che $k = s(x)$, utilizzando la relazione (2) fra $K(x)$ e $K(x \mid n)$, e usando $k \leq n$,

$$K(X^n) \leq K\big(X^n \mid n\big) + \mathcal{O}(\log n) \leq c + \mathcal{O}(\log n) + n\, H\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) \ .$$

A questo punto utilizziamo la disuguaglianza di Jensen, la quale afferma che se X è una variabile casuale qualunque e f è una funzione concava (come l'entropia binaria) allora vale

$$\mathbb{E}\big[f(X)\big] \leq f\big(\mathbb{E}[X]\big) \ .$$

Applicando questa disuguaglianza come segue, e notando che $\mathbb{E}[X_i] = q$ per ogni estrazione X_i di un simbolo sorgente,

$$\mathbb{E}\, H\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) \leq H\left(\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[X_i]\right) = H(q)$$

otteniamo immediatamente

$$\frac{1}{n}\mathbb{E}\big[K(X^n)\big] \leq H(q) + o(1)$$

che conclude la dimostrazione. □

Vediamo ora un altro modo di legare la complessità di Kolmogorov alla probabilità. Per ogni stringa $x \in \{0,1\}^+$, consideriamo il più breve programma p_x che la stampa. Sappiamo che l'insieme di

questi p_x costituisce un codice istantaneo $c_K : \{0,1\}^+ \to \{0,1\}^+$, dove $c_K(x) = p_x$. Possiamo vedere c_K come un codice di Shannon per la sorgente $\langle \{0,1\}^+, P_K \rangle$, dove $P_K(x) = 2^{-K(x)}$.

Estendiamo questo ragionamento per dare una diversa definizione di K. Dato un interprete libero da prefisso $\mathcal{M}$ per un linguaggio $\mathcal{L}$, definiamo la probabilità universale di una stringa $x \in \{0,1\}^+$ come

$$P_{\mathcal{M}}(x) = \sum_{p \in \{0,1\}^+ : \mathcal{M}(p) = x} 2^{-\ell(p)} \ .$$

La quantità $2^{-\ell(p)}$ è la probabilità che p sia generato da una sequenza di lanci di monete. Dato che $\mathcal{M}$ è libero da prefisso, le sequenze di lanci di monete che generano tutti i programmi che stampano x sono eventi indipendenti. Quindi $P_{\mathcal{M}}(x)$ è effettivamente la probabilità che l'interprete stampi x quando gli venga dato un programma casuale.

La seguente osservazione mostra che $P_{\mathcal{M}}$ gode di una proprietà di invarianza simile a quella di K. Cioè, se cambiamo linguaggio passando da un interprete $\mathcal{M}$ ad un inteprete $\mathcal{M}'$, la probabilità $P_{\mathcal{M}}(x)$ di una qualunque stringa x potrà variare al più di una costante moltiplicativa indipendente da x.

Teorema 6 (Invarianza di P) *Per ogni coppia $\mathcal{M}, \mathcal{M}'$ di intepreti liberi da prefisso per linguaggi in grado di rappresentare tutte le funzioni calcolabili esistono delle costanti $c, c' > 0$ tali che per ogni $x \in \{0,1\}^+$,*

$$c \ \leq \ \frac{P_{\mathcal{M}}(x)}{P_{\mathcal{M}'}(x)} \ \leq \ \frac{1}{c'} \ .$$

Dimostrazione. Ricordiamo che esiste un programma $p_{\mathcal{L}'} \in \mathcal{L}$ tale che $\mathcal{M}(p_{\mathcal{L}'} p') = \mathcal{M}'(p')$ per ogni $p' \in \mathcal{L}'$. Quindi, per ogni $p' \in \mathcal{L}'$ tale che $\mathcal{M}'(p') = x$ esiste $p \in \mathcal{L}$ tale che $\mathcal{M}(p) = x$ e $\ell(p) \leq \ell(p') + c_{\mathcal{L}'}$, dove $c_{\mathcal{L}'}$ è la lunghezza di $p_{\mathcal{L}'}$. Allora,

$$P_{\mathcal{M}}(x) = \sum_{p \,:\, \mathcal{M}(p) = x} 2^{-\ell(p)} \geq \sum_{p' \,:\, \mathcal{M}'(p') = x} 2^{-\ell(p') - c_{\mathcal{L}'}} = 2^{-c_{\mathcal{L}'}} P_{\mathcal{M}'}(x) \ .$$

Data l'arbitrarietà nella scelta di $\mathcal{L}$ e $\mathcal{L}'$ la dimostrazione è conclusa. □

Dato che fra i p che stampano x c'è anche il programma più breve di lunghezza $K(x)$, sappiamo che $P(x) \geq 2^{-K(x)}$ per ogni $x \in \{0,1\}^+$. Con un po' di sforzo, è possibile dimostrare che esiste una costante c tale che

$$2^{-K(x)} \leq P(x) \leq c\, 2^{-K(x)} \qquad \text{per ogni } x \in \{0,1\}^+.$$

Questa relazione caratterizza $K(x)$ (a meno di una costante additiva indipendente da x) come il logaritmo del reciproco della probabilità che un programma casuale stampi x.

Terminiamo costruendo una stringa infinita tale che ogni suo prefisso non può essere compresso significativamente. Dal Fatto 1 sappiamo che esistono tali stringhe, ma non è chiaro come costruirne una. Infatti, descrivere in modo finito una stringa infinita non comprimibile sembra impossibile, dato che una sua qualunque descrizione potrebbe essere usata per creare un programma che stampa la stringa.

A questo proposito, dato un interprete libero da prefisso definiamo il numero Ω di Chaitin come

$$\Omega = \sum_{p \in \{0,1\}^+ : \mathcal{M}(p) \text{ termina}} 2^{-\ell(p)} \ .$$

Questo numero è la probabilità che un programma a caso termini. Indichiamo con $\omega_1, \omega_2, \ldots$ l'espansione binaria di Ω, ovvero $\Omega = 0.\omega_1\omega_2\cdots$ dove $\omega_i \in \{0,1\}$. Chiaramente,

$$\Omega = \sum_{i=1}^{\infty} \frac{\omega_i}{2^i} \ .$$

Quindi, se indichiamo con $\Omega_n < \Omega$ il numero troncato $0.\omega_1 \cdots \omega_n$ abbiamo che

$$\Omega - \Omega_n < \sum_{i=n+1}^{\infty} \frac{1}{2^i} = 2 - \sum_{i=0}^{n} \frac{1}{2^i} = 2 - 2\big(1 - 2^{-n-1}\big) = 2^{-n} \ . \tag{3}$$

Teorema 7 *Il numero Ω non può essere compresso per più di una costante. Ovvero, esiste una costante $c > 0$ tale che, per ogni intero n, $K(\omega_1, \ldots, \omega_n) > n - c$.*

Dimostrazione. Per prima cosa, verifichiamo che la conoscenza di Ω_n ci permette di scrivere un programma p_0 per stabilire la terminazione di tutti i programmi p di lunghezza minore o uguale a n. Il codice di p_0 riceve Ω_n in input ed esegue le seguenti operazioni: ordina i programmi $p_1, p_2, \cdots \in \{0,1\}^+$ (è importante usare $\{0,1\}^+$ anche se ci interessano i programmi in $\{0,1\}^n$) usando l'ordine lessicografico e inizializza una variabile $\Omega' = 0$. Quindi esegue la prima istruzione di p_1; poi esegue la prima istruzione di p_2 e la seconda di p_1; poi la prima di p_3, la seconda di p_2 e la terza di p_1, e così via. Quando un programma p_i termina, lo elimina dalla lista dei programmi in esecuzione e somma a Ω' il suo peso $2^{-\ell(p_i)}$ (quindi, Ω' tende a Ω approssimandolo per difetto). Quando $\Omega' \geq \Omega_n$ (prima o poi succede, dato che $\Omega > \Omega_n$), abbiamo che $\Omega_n \leq \Omega' < \Omega$ e sappiamo che se un programma p con $\ell(p) \leq n$ non è nella lista dei programmi terminati allora non terminerà mai. Infatti, se p terminasse aggiungerebbe a Ω' una quantità $2^{-\ell(p)} \geq 2^{-n}$ che, usando (3), implicherebbe $\Omega' > \Omega$, contraddicendo $\Omega' < \Omega$.

Ora possiamo modificare questo codice in modo da memorizzare la stringa stampata da ogni programma terminante di lunghezza minore o uguale a n. Possiamo quindi stampare la prima stringa x_0 (in ordine lessicografico) che non appare nella lista di quelle generate da un programma terminante di lunghezza minore o uguale a n. Il programma p_0 modificato appena descritto stampa x_0 dato Ω_n e ha lunghezza costante c (non dobbiamo codificare n, basta contare i bit nell'input Ω_n). Quindi, $K\big(x_0 \mid \Omega_n\big) \leq c$. Usando (1) otteniamo

$$K(x_0) \leq K(\Omega_n) + K\big(x_0 \mid \Omega_n\big) \leq K(\Omega_n) + c \ .$$

D'altra parte, x_0 è la stringa più corta con complessità di Kolmogorov maggiore di n, cioè $K(X_0) > n$. Combinando le due disuguaglianze otteniamo

$$n < K(x_0) \leq K(\Omega_n) + c$$

che conclude la dimostrazione. □