

Calcolabilità

Pedaling lemma

Ogni f. Calcolabile ha $\# \mathbb{N}$ indici.

$\forall x$ si può costruire un insieme infinito di indici tramite l'op.
tale che: $\forall y \in A_x. P_y = P_x$.

Dimostrazione

Per ogni macchina M_x con i simboli stati, ottieni M_x , aggiungendo
 $q_{K+1}, \#_1, q_K, \#, -$ (corrisponde a uno strip) e così via

Forma normale

Esistono un predicato $T(i, x, y)$ e una f. $U(y)$ rp. c.t.t.
tali che $\forall i, x. \phi_i(x) = U(M_y.T(i, x, y))$

Cioè ogni f. ha una rappresentazione privi legate.

Dimostrazione

$T(i, x, y)$ predicato di Kleene: vero se y termina con input x.

Recupero M_i e scendisco y, controllando se termina. Definisco U

t.c. $U(y) = z$, è risultato della computazione. Il procedimento è
effettivo quindi $T \circ U$ calcolabili. Termina sempre quindi anche tutto.

Rp. perché le codifiche lo sono.

Nota: se le Molt. è deterministica c'è un solo y.

Per quelle non det., restituisce il minimo intero
che ne codifica uno (minimizzazione)



Enumerazione

Esiste f. calc. perz. t.c. $\varphi_2(i, x) = p_i(x)$

Definisce $\exists \text{N} \forall \text{x}$ universale, garantisce che esiste

Dimostrazione

Prendiamo le f. uscite nel Teo di f. normale $U(\mu_y, T(i, x, y))$.

È una f. calcolabile, definita per composizione e M-ricorsione a partire da f. r.p., quindi avrà un indice z, cioè sia $\varphi_z(i, x) = U \dots$

Applichiamo ora il Teo di f. normale, cioè $p_i(x) = U \dots$

$$\text{Perciò } \varphi_i(x) = \varphi_z(i, x)$$

Parametro

Esiste $s \in \text{Tot}$ iniettiva, t.c. $\forall i, x. \exists y. p_i(x, y) = \varphi_{s(i, x)}(y)$

x è quindi un parametro di p_i .

Dimostrazione

Prendi M e prepara lo stato iniziale. È una procedura effettiva, quindi ha indice, che è $s(i, x)$, calc tot.

Iniettività: se s non fosse iniettiva, costruisce s' t.c.

$\varphi_s(i, x) = \varphi_{s'}(i, x)$, che sappiamo essere garantito per il padding lemma. Gli indici sono generati in modo strettamente crescente, quindi s' è iniettiva.

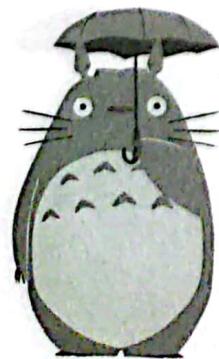
Definizione generale s_n^m

Esiste f. tot. iniettive s_n^m con m6 argomenti

t.c. $\forall i, x_1 \dots x_m, \exists y_1 \dots y_n. \varphi_{s_n^m(i, x_1 \dots x_m, y_1 \dots y_n)} =$

$$(\varphi_{s_n^m(i, x_1 \dots x_m)}^{(m)})^{-1}(y_1 \dots y_n)$$

$s(i, x) = f(x)$ viene scelta appropriata fissando le M



Punto fisso (ricorsione, Kleene II)

$\forall f \in \text{Tot}$, esiste n t.c. $p_n = f(n)$

Rosso Trasformare il mio programma per ottenere un equivalente

Dimostrazione

1- Definisco $\psi(v, z) = p_{d(v)}(z) = \begin{cases} p_{v(u)}(z) & \text{se } v \text{ converges} \\ \text{indet.} & \text{altrimenti. } d(M) \text{ è tot e} \\ & \text{iniettivo per } u \text{ parametro.} \end{cases}$

2- Quindi $f(l(l(x)))$ è calcolabile esso risolve T.C. $\psi_v(x) = l(l(x))$

che è totale quindi converge, seppiamo quindi che $p_{d(v)} = p_{\psi(v)}$

3- Calcoliamo $d(v)$ eponiamo che sia n , $d(v) = n$

Ora dimostriamo che n è punto fisso di f :

$p_n = p_{d(n)} \stackrel{1}{=} p_{\psi_n(n)} \stackrel{2}{=} p_{l(l(n))} \stackrel{3}{=} p_{f(n)} \square$

Pre-Rice

Note
esclusivo

Sia A un iirf. tale che $B \neq A \neq M$. Allora $K \subseteq_{rec} A$ oppure

$K \subseteq_{rec} \bar{A}$.

iirf: $\forall x, y$, se $x \in A$ e $p_x = p_y$ allora $y \in A$

Dimostrazione

Prendo ϕ t.c. ϕ è inolfinito ovunque, io si trova in A in \bar{A} perché $B \neq A \neq M$. Suppongo sia in \bar{A} . Dimostro $K \subseteq_{rec} A$.

Sia $i, x \in A$, allora $i \neq i_x$. Definisco allora $\psi(x, y) = \begin{cases} p_{i_x}(y) & \text{se } x \in K \\ \text{indet. } (\phi, y) & \text{altrimenti} \end{cases}$ che è calcolabile, quindi $\psi(x, y) \in p_{d(y)}(y)$.

Per tuo parmetro, $y_{S(i, x)}(y) \rightarrow b = d_x$, s.t. $i \in S(i, x)$. Allora

$x \in K \rightarrow p_{d(x)} = p_{i_x} \rightarrow f(x) \in A$ perché $i \in A$ è iirf.

Viceversa $x \notin K \rightarrow p_{d(x)} = \text{indet.} \rightarrow f(x) \in \bar{A}$.



Rice

Sia \mathcal{A} classe (insieme) di funzioni calcolabili. L'insieme $A = \{n | P_n \in \mathcal{A}\}$ è ricorsivo se e solo se $\mathcal{A} = \emptyset$ oppure $\mathcal{A} = \mathbb{N}$ (cioè le classi di tutte le funzioni calcolabili)

Dimostrazione

\mathcal{A} insieme di indici; \mathcal{A} classe di funzioni (semantica)

Se $A = \emptyset$ o $\mathcal{A} = \mathbb{N}$, ricorsivo, altrimenti applico pre-Rice e
risolvo $K \leq_{rec} A \circ K \leq_{rec} \mathcal{A}$

K

$$K = \{x | P_x(x) \downarrow\}$$

K non è ricorsivo:

Supponiamo per assurdo che la caratteristica di K sia calcolabile totale. Allora anche $f(x) = \begin{cases} P_x(x)+1 & \text{se } x \in K \\ 0 & \text{altrimenti} \end{cases}$ lo sarebbe.
Questo porta a una contraddizione perché $\forall x. f(x) \neq P_x(x)$, quindi non abbiamo indici per $f \rightarrow$ non è calcolabile.

K è RG-completo

Prendiamo A , dominio di b.calcolo Ψ , $A = \{x | \Psi(x) \downarrow\}$.

Definiamo Ψ' a due variabili: $\Psi'(x, y) = \Psi(x)$ (ignora le y quindi).

È a sua volta calcolabile, quindi ha un indice $\Psi' = p_i$

Per l'indice parametrazione abbiamo $P_i(x, y) = P_{S(i, x)}(y)$.

È quindi possibile riscrivere $A = \{x | P_{S(i, x)}(y) \downarrow\}$.

Poniamo $y = S(i, x)$, ottieniamo $A = \{x | P_{S(i, x)}(S(i, x)) \downarrow\}$

$\rightarrow \{x | S(i, x) \in K\}$, ovvero $x \in A$ se $f(x) \in K$. \square



K non è iirf

Definisco $\Psi(x,y) = \begin{cases} 1 & \text{se } x=y \\ \text{indif} & \text{altrimenti} \end{cases}$. Ho indice di inflazione $= \varphi_i(x,y)$
per parametro $= f_{\text{scars}}(y) = \varphi_{\text{tot}}(y)$.

No un punto fisso per f_i , calc tot. $\varphi_{\text{tot}} = \varphi_f(y)$.

Ora $\Psi(n,n) = 1$ ($n = n, n \in K$). Con il perduring lemma
trovo un altro indice per f_i : Sia q tale che
 $\varphi_n = \varphi_n = \varphi_q$ per definizione.

Siccome $n \neq q$, $\Psi(n,q) = \text{indif}$. $q \notin K \rightarrow K$ non è iirf

Complexität

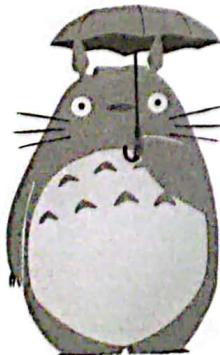
Riduzione del nr di nastri

Dato un $M \times N$ matrice che decide I in tempo $O(f)$, esiste un vettore che decide I in $O(f^2)$

Dimostrazione

1. Costruisco M' , racchiudendo i nastri nelle parentesi: è d'uso usare nuovi simboli \bar{o}_i per ricordare la posizione delle Testine su ogni nastro.
 2. Genero la configurazione iniziale di M' : $(q_0, \Sigma^k \times \delta^k (0^k))^{k+1}$.
Bastano $2k+1$ nuovi stati e un nastro per ciascuna delle $k+1$ testine.
 3. Per simulare una mossa di M , M' scorre il nastro due volte da sinistra a destra e viceversa: la prima volta determina i simboli correnti, la seconda scrive i nuovi con asc尺度 di spostamento. a destra si bisogna spostare $i+ds$.
 4. Quando M si ferma, si ferma M' rimuovendo \bar{o}_i e sostituendolo con o_i .

Siccome non si può usare più spazio che tempo, la lunghezza è al più $K = N \cdot (f(|x|) + 2) + 1$, per fare avanti e indietro 2 volte, costa NK . Un singolo passo costa quindi $O(f(|x|))$ passi. Il numero di passi totale è perciò $O(f(|x|)^2)$.



Accelerazione lineare

Se $I \in \text{TIME}(f)$, allora $\forall \epsilon \in \mathbb{R}^+, I \in \text{TIME}(\epsilon \cdot f(n) + n + 2)$

Dato un algoritmo, se ne può sempre trovare uno equivalente più veloce per una costante.

Dimostrazione - intuizione 30 €

Si può determinare quanti caratteri si possono condensare in uno.

1. Condenso il dato di ingresso in $n+2$ passi.

2. Si simula un passo da m in 6 passi. Nei primi 4 fa s_x, d_x, d_x, s_y , tornando sul carattere corrente; raccoglie i simboli che potrebbe visitare e li codifica nel proprio stato.

M' simula a blocchi in mosse di 6 , muovendosi da s_x a d_x , modificando solo 2 simboli incluso s (le altre 2 mosse)

M'allora farà $1 \times 1 + 2$ passi per condensare l'input $+ 6 \cdot \lceil \frac{6 \cdot 1 \times 1}{m} \rceil$.
Fatto in t.c. $n = \lceil \frac{6}{m} \rceil$

Gerarchia di tempo e spazio

F'appropriato: è monotone crescente e si arresta in tempo $O(f(1) + 1)$ e spazio $O(f(1))$

Se f è appropriato, allora:

$\text{TIME}(f(n)) \leq \text{TIME}(f(2n+1))^3$

$\text{SPACE}(f(n)) \leq \text{SPACE}(f(n) \times \log f(n))$

Dimostrazione

Omissa. Si basa sul fatto che $\sum_{k=1}^{\infty} k^3/n^k$ converge altra forse più

presto in $\text{TIME}(f^3)$ ma non in $\text{TIME}(f)$



Accelerazione (Blum)

$\forall f \in \text{tot t.c. } \exists$ problema I t.c., M che decide I in $\text{TIME}(f)$,
 $\exists M'$ che decide I in $\text{TIME}(f')$ tale che $f(n) > h(f'(n))$ quasi ovunque.
Non sempre esiste un alg. ottimo; avremo programmi più
veloci su una macchina vecchia che su una nuova.

Si può dimostrare che \exists una succ. di alg. sempre più efficienti:
ma non si sa come costruirle.

Laccone (Borodin)

$\exists f \in \text{tot t.c. } \text{TIME}(f) = \text{TIME}(2^f)$

È una forma speciale del teorema precedente con $h=2^n$.

Se non si usano f appropriate, all'aumentare delle risorse non
si allarga la classe dei problemi decisi con esse.

Alcune dimostrazioni utili

$\text{LogSpace} \subseteq P$

Prenolo $I \in \text{logspace}$, $\exists M$ che determina in spazio $O(\log x)$.

Si può attraversare max $O(x \log x \# Q \# \Sigma^{\log x})$ configurazioni:
 x letti in input, $\# \Sigma^{\log x}$ nastri, $\log x \# Q$ (totali) posizioni del cursore.

Pongo tutto $\leq x^5$. Applico $\log x$ a entrambi i numeri:
 $\log x + \log(\log x) + \log \# Q + \log \# \Sigma^{\log x} \leq K \log x$.

Tolgo $\log \log x$ (piccolo) e $\log \# Q$ (costante). Semplifico
 $\log x$ dai due lati: $1 + \log \# \Sigma^{\log x} \leq K$.

Quindi basta porre $K = \log \# \Sigma^{\log x}$ e ha al massimo $O(x^K)$
possibile.



$$NTIME(\ell(n)) \leq TIME(c^{\ell(n)})$$

Sia ℓ grado di non determinismo di N (la dimensione dell'albero di decisione). Ogni computazione è una sequenza di scelte, M le esegue una alla volta in ordine crescente (visita l'albero per livelli).

Se durante l'esecuzione arriva a σ_{si} , termine accettando, altrimenti pessa alla successiva. Se tutte le ~~successioni~~ arrivano a σ_{no} , M termine rifiutando.

Le successioni da visitare sono $O(\ell^{6n+1})$, dimostra con $c=6$ \square

$$P \not\subseteq EXP$$

Definiamo $EXP = \bigcup_{K \geq 1} TIME(2^{n^K})$. L'inclusione è propria perché $n^K < 2^{n^K}$ $\forall K$.

Con il teo di gerarchia abbiamo inoltre:

$$P \subseteq TIME(2^n) \not\subseteq TIME((2^{2n+1})^3) \subseteq \bigcup_{K \geq 1} TIME(2^{n^K}).$$

Tabelle di computazione

1. La macchina si arresta prima di \times^{15} passi
2. Ogni riga comincia con Δ e finisce con $\#$. Tutte le posizioni non significative sono riempite con $\#$. Il cursore non supera mai \times^{15} perché il tempo limite lo impedisce
3. Supponendo il cursore in una posizione, posso codificare lo stato nell'alfabeto. Aggiungo Σ^* a nuovi simboli
4. La config iniziale ha il cursore sul carattere subito dopo Δ di inizio testo. La testina non sarà mai sul primo Δ : codifico gli spostamenti $sx, dx, sv\Delta$ come un unico passo. Eccezione \rightarrow



5. se lo stato è o_1 , o_2 , sposto il cursore fino alla seconda colonna: lo stato di accettazione è sempre in $T(l, 2)$ per $l \leq x^k$.

Eccezione pt. 4: si ammette che lo testime veda sopra se lo stato è o_1 ma non tocca D di init. mostro

6. Se o_1 , o_2 sono in $T(p, 2)$ con $p \leq x^k$, tutte le righe q con $p \leq q \leq x^k$ sono gradi alle righe p .

7. Termino con successo se $T(x^k, 2) = o_3$.

Osservazioni:

1. $T(1, 2)$ contiene lo stato iniziale e il primo carattere del x

2. $\forall j, 2 \leq j \leq m+1, T(i, j)$ contiene il $j-1$ esimo carattere (dwh)

3. $\forall i, 1 \leq i \leq x^k, T(i, j) = \#$

4. $\forall i, T(i, s) = P$ e $T(i, x^k) = \#$

Circuit value è P-completo (cv)

Sia $I \in P$. Troviamo la logspecie de I a circuit value.

Costruzione del circuito

Tabelle di computazione di M , $\Sigma = \{x \in Q \cup \{\#\}$, simboli. Ogni simbolo in Σ viene mappato in una stringa di bit lunga $m \lceil \log_2 |\Sigma| \rceil$.

Codif. co $P = TT..tt$ e $\# = bb..bb$.

Ogni elemento è quindi una sequenza di m bit: $T(i, j) = s_{i, 1} s_{i, 2} \dots s_{i, m}$.

Abbiamo quindi una tabella con x^k righe, lunghe $m \cdot x^k$.

Siccome le f di transizione δ è fissata, il valore di $s_{i, j}$ dipende solo dalle 3 sequenze precedenti $i-1, i \pm 1, j$.

Sono quindi in grado di costruire il circuito con $3m$ ingressi e m uscite, codifica di $T(i, j)$.

La taglia del circuito è fissata (dipende da δ). Sono costanti i circuiti di prima e ultime colonne (D e $\#$)



La riduzione

- Prime e ultime colonne sono fisse, non c'è bisogno di costruire copie, anche prime righe perché input. Il resto delle caselle le riempio con copie del circuito. L'uscita è una delle uscite di \bar{C}_k . Dobbiamo ora verificare che \bar{C}_k fornisce valori esatti se $x \in I$.
- Bene, per $i \in S$. Ipotesi induktiva: $i - 1$ è ben calcolato. $T(i, j)$ dipende solo dalle tre caselle precedenti in accordo con \bar{S} .
- Immediato dedurre $T_{k, k} = \bar{C}_k$ sse $f(x) = T_k - T_1$ e viceversa per \bar{T}_k no.

Logspace?

- $\mathcal{O}(x^k)$ circuiti necessari: $2x^k$ per le colonne, x^k prime righe.
- $(x^{k-1})(x^{k-2})$ copie del circuito necessarie, di costo fisso, con indici $\leq x^k$. La rappresentazione in binario a porte a $\mathcal{O}(\log k)$

Monotone circuit value P-completo

- CV si riduce a MCV con De Morgan dall'alto in basso. Scambio T_k se è necessario ai nodi iniziali. In logspace perché visita una sola volta tutte le porte, rappr. in binario

Circuit SAT è NP-completo

- Premolo $I \in NP$, $x \in I$ sse $f(x)$ soddisfacibile. N decide I in $\mathcal{O}(T(n^k))$.
- Una computazione viene vista come una sequenza di scelte nodal.
- Fissiamo grado di nodal = 2. **Esempio grado 2?** Premolo la prima scelta e considera le altre come un nuovo stato. La macchina rallenta ma posso farlo in $\mathcal{O}(x^k)$.

- Una computazione divenuta una successione di bit

$$B = b_0, b_1, \dots, b_{x^k-1} \text{ con } b_i \in \{T, f\}$$



Non abbiamo vere tabella per N ma i costruibile fissato B .
 Il circuito è analogo a circuiti K-map e costruibile in log space.
 Essenzialmente si aggiunge nelle codificazione di un bit b_i alla fine, corrispondente alle scelte. $T(i,j)$ dipende quindi non solo dai 3n bit di codifica, ma anche da b_{i-1} , cioè della scelta precedente.
 La soddisfabilità si dimostra similmente a CV

Circuit SAT \leq SAT

Prendo un circuito $C(N, A)$ con variabili in X . Dobbiamo trovare f in logspace t.c. $f(C)$ è soddisfacibile sse C lo è.

Se $\exists V. C_V = \text{tt} \rightarrow \exists V'. V \times \in X. V'|_X = V(X) \wedge V' \models f(C)$.

Le variabili di $f(C)$ sono quelle di C + uno per ogni porto,
 il porto g , costruiamo i congiunti:

- g è la porta di uscita \rightarrow genero g : mani g intt
- $S(g) = \text{tt} \rightarrow$ genero g . se $\models f(C)$ genero $\neg g$
- $S(g) = x \rightarrow$ genero $(\neg g \vee x) \wedge (\bar{g} \vee \neg x)$: g sse x
- $S(g) = \perp \rightarrow$ genero $(\neg g \vee \perp) \wedge (\bar{g} \vee \perp)$: g sse \perp
- $S(g) = v \rightarrow$ genero $(\perp \vee g) \wedge (\perp \vee \bar{g}) \wedge (v \vee \bar{v})$: g sse $v \vee \bar{v}$
- $S(g) = \perp \wedge \dots \rightarrow$ genero $(h \vee g) \wedge (\bar{h} \vee \bar{g}) \wedge (K \vee g) \wedge (\bar{K} \vee \bar{g})$: g sse $h \wedge K$.

Sto in log space perché basta delle copie di n che richiedono $O(\log n)$ bit



HAM < SAT

Dato un grafo orientato $G = (N, A)$, trovare f.t.c. G ha un cammino hamiltoniano sse $f(G)$ è soddisfacibile.

- Introduciamo un'espressione booleana B in funzione di G . Se G ha n nodi, B ha n^2 variabili: x_{ij} ($1 \leq i, j \leq n$): il nodo j è l' i -esimo di un cammino. Siccome è in funzione, basta costruire i congiuntivi:
- 1 - j non può apparire in ulteriori posizioni diverse: $(\exists x_{ij} \vee \exists x_{ik}) \wedge \forall k \neq j, \neg x_{ik}$
 - 2 - Ogni j deve apparire nel cammino: $(x_{1j} \wedge x_{2j} \wedge \dots \wedge x_{nj})$
 - 3 - Qualche nodo deve essere in posizione i : $(x_{i1} \vee x_{i2} \vee \dots \vee x_{in})$
 - 4 - Due nodi non possono essere contemporaneamente in G : $(\exists x_{ij} \vee \exists x_{ik}) \wedge \forall j \neq k, \neg x_{ik}$
 - 5 - Se (i, j) non è un arco ($\notin A$), i.e. j non possono apparire in sequenza: $(\exists x_{ki} \vee \exists x_{k+1,j}) \wedge \forall k, 1 \leq k \leq n-1 \text{ e } (i, j) \notin A$

Ora dimostriamo $f(G)$ sia sse G ha ham.

$\forall j \exists i : V(x_{ij}) = \text{tt}$, e viceversa (3 e 2). Vale quindi una permutazione dei nodi di G . Garantisce che è un cammino e un ham non è altro che una permutazione di nodi in un grafo connesso che si dice lungo: si scrive n su un mastro, poi usa tre nastri di lavoro per i, j, k mentre scorre le clevisole 1-4. Scrivere le 5 sull'out put. Serrano quindi 4 copie di n , che richiedono 5 bit.



SAT < CRICCA

Cricca: gruppo di nodi interconnesso in un grafo

Data expr bool in ling $B = \bigwedge C_K$, costruisce il grafo $f(B) = (V, A)$ con N insieme delle occorrenze dei letterali in B e A insieme $\{(i, j) | i \in C_K \rightarrow (j \notin C_K \wedge i \neq -j)\}$. Essendo in fuc, B è soddisfatto se c'è almeno un letterale vero in ogni congiunto C_K .

Quindi B è sat se c'è una cricca di ordine pari al nr dei congiunti: basta mettere fra i nodi collegati. Per costruzione infatti un letterale non può essere connesso al suo negato, né a uno che compare nello stesso congiunto.

In logspecce perché contiene due indici sui nastri di lavoro in binario, che segnano i letterali, $O(\log n)$