TEOREMI DI GERARCHIA

Uno dei pochi risultati che consentono di separare classi di complessità. La tecnica è quella della DIAGONALIZZAZIONE.

Il significato intuitivo è che una TM con più risorse può fare più cose.

T.H.T : Ad alto livello, assumiamo $t_1(n) << t_2(n)$. Ad es. $t_1(n) = n^2$, $t_2(n) = n^6$.

Allora, esiste $L \in DTIME(t_2(n))$ ma $L \notin DTIME(t_1(n))$.

Per dimostrare questo fatto useremo cioè fatto che

Made with Goodnotes

di TM. Fissiamo una codifica

$$[\cdot]_{TM} : \Sigma^* \to \{TM\}$$

Se una stringa $x \in \Sigma^*$ non fosse una codifica valida di TM, la possiamo mappare in una TM di default. In questo modo:

$$\forall \text{ TM } M \quad \exists\, x \text{ t.c. } [x]_{TM} = M.$$

Costruisco una TM $D(x)$ che sicuramente termina in tempo $t_2(n)$.

$\underline{D(x)}$:

- Input x t.c. $M = [x]_{TM}$
- Simulo $M(x)$ per $t_{1.5}(n)$ passi dove

$t_1 << t_{1.5} << t_2$.

- Fai l'opposto: $M(x) = ACC$ sse $D(x) = REJ$

Se invece $M(x)$ non ha terminato, non importa ($D(x) = ACC$ ad esempio).

Siccome $t_2(n) >> t_{1.5}(n)$, abbiamo che $L = L(D) \in DTIME(t_2(n))$.

Resta da dimostrare che per ogni TM Q con $\#$ passi $\leq t_1(n)$, ma tale TM sbaglia e decidere L.

Prendiamo un qualsiasi x t.c. $[x]_{TM} = Q$.

Mostriamo che per un tale x, $D(x) \neq Q(x)$.

Made with Goodnotes

Supponiamo che $Q(x)$ termini in $t_1(n)$ passi per $t_1(n) << t_{1.5}(n)$. Ovvero, $D(x)$ fa in tempo e finire la simulazione e alla fine fa l'opposto, ovvero $D(x) \neq Q(x)$.
Pertanto D sbaglia e rispondere alla domanda "$x \in L$?" $\Rightarrow L \notin DTIME(t_1(n))$.
Questa è la dimostrazione; ma ci sono alcuni dettagli da definire.
Dobbiamo stare attenti, perché $DTIME(t_1(n))$ comprende tutte le TM con # passi $\leq c \cdot t_1(n)$ per ogni costante $c \in \mathbb{N}$,
per questo motivo $t_{1.5}(n) = \omega(t_1(n))$

Made with Goodnotes

ovvero $t_1(n) = O(t_{1.5}(n))$. Inoltre queste sono notazioni asintotiche, ovvero le disuguaglianze valgono solo a partire da tutti gli $n \geq n_0$ per un certo n_0 (per n sufficientemente grande). Per questo motivo non basta avere un singolo x t.c. $Q(x) \neq D(x)$ perché x potrebbe essere troppo corto.
Dobbiamo fare in modo che questo accada per infiniti x. Usiamo padding: aggiungiamo alle codifiche di TM le stringhe $10\ldots\ldots\ldots0$ di lunghezza arbitraria.
Infine, dobbiamo essere sicuri che la simulazione

di $M(x)$ sia fattibile in $O(t_2(|x|))$ tempo; ovvero abbiamo bisogno di una TM che può simulare ogni TM M con tempo $O(t_{1.5}(n))$ in $O(t_2(n))$ tempo.
Questa sarebbe la TM universale U che abbiamo già incontrato. Quella che abbiamo visto è in grado di simulare una TM con tempo $T(n)$ in $O(T^2(n))$ tempo.
Negli anni 60' si è dimostrato che in realtà ciò è fattibile in $O(T(n) \log T(n))$. tempo.
Ultima osservazione sottigliezza: Siccome D deve

Made with Goodnotes

confare $t(n)$ peso; $t(n)$ deve essere TEMPO COSTRUTTIBILE.

DEF. $t(n)$ è TEMPO COSTRUTTIBILE se $\exists$ TM che su input 1^n ritorna un output $1^{t(n)}$ in tempo $O(t(n))$.

Abbiamo quindi dimostrato:

TEO Siano $t_1(n) \geq n$ e $t_2(n) = \omega(t_1(n) \lg t_1(n))$ e TEMPO COSTRUTTIBILE. Allora, esiste L t.c. $L \in DTIME(t_2(n)) \setminus DTIME(t_1(n))$.

COR. $P \subsetneq EXP$.

Abbiamo anche un Teorema di gerarchia di spazio. E' volentieri, ma la TM universale U simula ogni altra TM in spazio $O(s(n))$. Quindi:

TEO. Se $s_1(n) \geq \log n$ e $s_2(n) = \omega(s_1(n))$ e SPAZIO-COSTRUTTIBILE. Allora $\exists L$ t.c. $L \in SPACE(s_2(n)) \setminus SPACE(s_1(n))$.

Possiamo fare lo stesso con le classi non deterministiche? È più complicato usare la diagonalizzazione, ma è possibile.

- Buona notizia: La NTM U può simulare ogni NTM con Tempo $T(n)$

Made with Goodnotes

in tempo $O(T(n))$. Non lo dimostriamo. Intuitivamente, possiamo usare il non det. per indovinare simultaneamente la posizione delle testine sui nastri della NTM simulata.

- Cattivo motivo: Mentre quanto detto sopra sembra permettere di dimostrare il T.H.T. per $t_2(n) = \omega(t_1(n))$, non lo sappiamo fare. Il motivo è che il non-det. non è chiuso per complemento e quindi non possiamo flippare output e usare la diagonalizzazione.

È stato aperto per un po' di anni, ma poi è stato risolto. Per lo spazio osserviamo che invece non c'è problema perché per il teorema "IS" NSPACE è chiuso per complemento.
Idea di base: con overhead esponenziale posso fare l'opposto di una NTM con tempo $O(t_1(n))$ in tempo det. $2^{O(t_1(n))}$.
Sebbene questo gap sia troppo grande per simulare una T.M. non-det. interessante, la prova userà questa idea.

TEO Sia $t(n) \geq n$ e $T(n)$ t.c.

$$t(n+1) = o(T(n)).$$

Allora, $\exists$ $L \in NTIME(T(n)) \setminus NTIME(t(n))$.

Dim. Daremo le idee principali; alcune tecniche usate sono le stesse del caso deterministico. Il linguaggio L è $\subseteq 1^*$; l'input $x = 1^n$ (una a volta per semplicità scriverò n.)

Sia N_n la TM corrispondente al 1^n secondo una codifica fissata per NTM. Dividiamo l'input n in intervalli :

$[l_1, u_1], [l_2, u_2], \dots$

$l_1 \to 0$, $l_2 \to u_1 + 1$

$u_i >> l_i \; ; \; \forall i$.

Definisco una NTM $D(n)$:

- Trova i t.c. $l_i \leq n \leq u_i$
- Se $n \neq u_i$, allora simula N_i (dove i è l'indice dell'intervallo) su input $n+1$ per $T(n)$ passi. (Notare che $n+1$ sta sempre nell'intervallo i-esimo.)
- Se $n = u_i$, allora simula $N_i(l_i)$

otterremmo nostra corrente per log(T(n)) passi.
Quindi fa l'opposto come prima.
Sia $L = L(D)$. Come prima, abbiamo che
$L \in NTIME(T(n))$; qui usiamo il fatto che
la NTM universale U ha overhead costante.
Resta da mostrare che nessuna NTM con tempo
$t(n)$ può decidere correttamente L; ovvero
esiste almeno un input t.c. ogni decisore
non-det. con tempo $t(n)$ fallisce.
Sia N la NTM di tempo $t(n)$ tale che
$N(n) = D(n)\ \forall n$. Sia i t.c. $N = N_i$.

Per assurdo supponiamo $N(l_i) = D(l_i)$.
Ma cosa fa $D(l_i)$? Simula $M = N_i$ su input $l_i + 1$ per $T(l_i)$ passi. Sappiamo che $t(l_i + 1) = o(T(l_i))$ per ipotesi, c'è abbastanza tempo e quindi $D(l_i) =$
$= N(l_i + 1)$. $\Rightarrow$ $N(l_i) = N(l_i + 1)$.
Stesso ragionamento: $N(l_i + 1) = D(l_i + 1)$.
Ma che fa $D(l_i + 1)$? Simula $M = N_i$ su input $l_i + 2$ per $T(l_i + 2)$ passi.
Per ipotesi, $t(l_i + 2) = o(T(l_i + 1))$
$\Rightarrow D(l_i + 1) = N(l_i + 2) = N(l_i + 1)$

E così via... Alla fine:

$$D(M_i - 1) = N(M_i) = D(M_i) = N(\ell_i)$$

Che fa $D(M_i)$? Simula $N_i(\ell_i)$ per $\log(T(M_i))$ passi $\geq \log(M_i)$ e fa l'opposto. Contraddizione.

Basta prendere $M_i = 2^{t(\ell_i)^3}$; in questo modo $N_i(\ell_i)$ richiede $t(\ell_i)$ passi e quindi $D(M_i)$ fa l'opposto di $N_i(\ell_i)$ ■

Il valore $2^{t(\ell_i)^3}$ è arbitrario e tiene conto della simulazione della TM non determinismo stico deterministicamente.

P vs NP

Perché è difficile? Aperto dal 1956 (lettera di Gödel e Von Neumann); formalizzato nel 1970.

Il problema è abbastanza frustrabile.

È stato dimostrato che alcune tecniche sono senza speranza. Ad es. è la diagonalizzazione.

I risultati che seguono per diagonalizzazione hanno una proprietà: RELATIVIZZAZIONE.

Ad es.: Per ogni A, ∃ L decidibile

da TM con A-oracolo e tempo $T(n)$, ma non decidibile da TM con A-oracolo e tempo $o\left(\frac{T(n)}{\log(n)}\right)$.

La prova è la stessa del T.H.T punì le chiamate all'oracolo A.

Altro esempio: $NP^A \subseteq PSPACE^A$, $\forall A$.

TEO $\exists$ A t.c. $P^A = NP^A$. Inoltre, $\exists$ B t.c. $P^B \neq NP^B$.