13/12/2024

Teo: TQBF è PSpace hard

Dim: devo mostrare che ogni problema in PSpace si riduce in Tempo $poly(n)$ a TQBF, la dim mera è idee già viste.

garantisce anche spazio polinomiale

Osserviamo che esiste una TM M che decide A (linguaggio generico in PSPACE) in spazio $O(n^a)$ per qualche a, vogliamo costruire $R: \{0,1\}^* \to \{0,1\}^*$ che restituisce una TQBF ϕ_M tale che $x \in A \iff M(x) = accept \iff R(x) = \phi_M$ è vera.

Precisazione: Ci sono formulazioni equivalenti di una TQBF,

- Assumere che $\exists, \forall$ sono alternati. Se non lo fossero si possono aggiungere $\leq n$ DUMMY VARIABLES.
- Permettere nelle formule "$\to$" e "$\leftrightarrow$"; si possono trasformare in CNF

- permettere che i quantificatori non siano tutti all'inizio.

$$\exists x_1 \forall x_2 [x_1 \to x_2] \wedge \dots$$

Sono modi equivalenti di definire un TM BF.
Cosa possiamo dire di $M(x)$? Sappiamo che ha $2^{O(n^2)}$ configurazioni (SPAZIO LIMITA TEMPO)
Uno di queste sarà C_{start}, la configurazione iniziale $q_0 \sqcup; 1$.
Possiamo assumere senza perdita di generalità che esiste UNICA configurazione accettante C_{acc}. Il fatto che $M(x)$ accetta corrisponde al fatto che $\exists$ cammino $C_{start} \rightarrow C_{acc}$ nel grafo $G_{M,x}$ definito come nel Teorema di Savitch.

Oss: Il numero di mesi è esponenziale, siccome M è deterministica, una configurazione porta solo in altra. Una riduzione $R(x)$ darà ϕ_M tale che ϕ_M è soddisfacibile $\iff \exists$ cammino $C_{start} \rightarrow C_{acc}$. Basta che ϕ_M sia calcolabile in tempo polinomiale, questo fatto sarà immediato.

Idea 1) potremmo dire che ϕ_M è del

Tipo: $\phi_M = \exists c_1 \exists c_2 \dots \exists c_\ell$ t.c.

$c_1 = c_{start}$, $c_\ell = c_{acc}$ ed i nodi intermedi sono quelli del cammino $c_{start} \to c_{acc}$

Le c_i sono le config. codificabili con $O(n^a)$ bit, essendo quante $2^{O(n^a)}$.

Poi in dettaglio procediamo come in cook levin, definiamo una sottoformula

$\phi_M = \phi_{yields}(c_{start}, c_2) \wedge \phi_{yields}(c_2, c_3) \dots$

$\wedge \phi_{yields}(c_{\ell-1}, c_{acc})$

ϕ_{yields} è la stessa usata nella dimo. di Cook-Levin. $\phi_{yields}(c_i, c_j) = True$ se c_j segue a c_i in accordo con le regole della δ.

Problema: $\ell \leq$ numer conf $= 2^{O(n^a)}$ è quindi in spazio esponenziale.

Idea 2) Usare l'algo. ricorsivo del Teo. di Savitch. Ovvero, costruiamo $\phi_k(c_0, c_1)$ che è vera $\iff$ c'è un cammino da c_0 a c_1 di lunghezza 2^k.

Fatto questo, $R(\varphi)$ ritorna l'output di $\phi_{O(n^2)}(C_{start}, C_{accept})$

$\Leftrightarrow$ Vera $\Leftrightarrow \exists$ cammino $C_{start} \to C_{acc}$ di lunghezza $2^{O(n^2)}$

- Caso base ($K=0$): $\phi_0(C_0, C_1) =$

$\phi_{yields}(C_0, C_1) \Rightarrow C_1$ può risultare da C_0

oppure $C_0 = C_1$

- Caso induttivo: $\phi_K(C_0, C_1) =$

$\exists\, C_{mid}$ per cui

$$\phi_{K-1}(C_0, C_{mid}) \wedge \phi_{K-1}(C_{mid}, C_1)$$

Questo ancora non va bene: dato che la nostra ϕ_K ha dimensione

$$|\phi_K| = O(n^2) + 2|\phi_{K-1}|$$

$$= O(n^K \cdot n^2)$$ raddoppio ad ogni passo

Idea Finale) Meno (cioè 2) ma scritta in maniera compatta, definiamo

$\phi_K(C_0, C_1) = \exists\, C_{mid}$ t.c. $\forall D, D'$ la seguente è vera

$$\left(\begin{array}{c} (D, D') = (C_0, C_{mid}) \\ \vee \\ (D, D') = (C_{mid}, C_1) \end{array} \right) \Longrightarrow \phi_{K-1}(D, D')$$

Ispezione equivalente che richiama ϕ_{K-1} solo una volta, quindi per le dimensioni:

$$|\phi_K| = O(n^a) + |\phi_{K-1}|$$

$$\Rightarrow |\phi_K| = O(K \cdot n^a)$$

$$\Rightarrow |\phi_{O(n^a)}| = O(n^{2a}) \Rightarrow \text{polinomiale.} \quad \blacksquare$$

Concludiamo con la completezza con un classico risultato. Il Teorema di

Sappiamo che NSPACE = CONPSPACE = PSPACE, la versione "ristretta" è NLOG = CONLOG.

Teo (Immerman-Szelepcsényi): NLOG = CONLOG

Dim: basta dimostrare che PATH ∈ CONLOG. Questo è vero se $\overline{\text{PATH}} \in$ NLOG, sappiamo che $\forall A \in$ NLOG, $A \leq_m^L$ PATH, Se PATH ∈ CONLOG allora NLOG ⊆ CONLOG, d'altra parte

se $B \leq^L_m PATH \Rightarrow \overline{B} \leq^L_m PATH$ e se $\overline{PATH} \in NLOG$

$\exists$ NTM che decide $\overline{B}$ in spazio $O(\log(n))$, quindi $coNLOG \subseteq NLOG$. Questo sembra sorprendente: esiste un certificato polinomiale del fatto che non esiste un cammino da $s \rightsquigarrow t$ che può essere verificato in spazio $O(\log(n))$.

Verificatore per NLOG:

nastro del certificato può essere $O(n^k)$ ma può essere letto solo 1 volta

Questa def. è equivalente a quella che usa il non determinismo.

Vediamo il certificato

Input: grafo (G, s, t)

dato G, siano R_ℓ i vertici raggiungibili da s in al più ℓ passi, e definiamo $r_\ell = |R_\ell|$

Ad esempio, $R_0 = \{s\}$, raggiunge al più se stesso $\Rightarrow r_0 = 1$

Il certificato è diviso in più parti: certificato per r_1; per r_2 ... per r_n, per $s \rightsquigarrow t$ (non c'è il cammino).

Idea) Una volta che V (verificatore) ha verificato il cert. per r_ℓ, deve leggere solo di ℓ, r_ℓ sul nastro di lavoro per verificare il cert. di $r_{\ell+1}$.

Ma come sono fatti i certificati?

Supponiamo che V sia convinto di r_n, e lo abbia verificato. r_n = numero vertici raggiungibili da s in n passi $\Rightarrow$ dim. massimo di passi per un cammino = n, il certificato per (non esiste cammino $s \to t$) sarà una cosa del tipo

$s \rightsquigarrow v_2$; $s \rightsquigarrow v_3$; ...;

↳ sono in numero r_n, ma verifica che t non sia fra questi.

Il verificatore controlla che ogni cammino è in G (usando $O(\log(n))$ spazio) poi controlla che ci sono r_n cammini; contando in $O(\log(n))$ spazio, e controlla che t non è nodo finale dei cammini. Non devono essere cammini ripetuti, quindi

Si assume che nel certificato i nodi (endpoint) siano in ordine lessicografico così è possibile controllare che non ci siano ripetizioni salvando 1 solo vertice.

Per il resto facciamo il certificato per r_{l+1} quando già è verificato r_l.

Il certificato sarà fatto così:

$v_1 \in R_{l+1}$ perché

$v_2 \notin R_{l+1}$ perché

:

$v_n \in R_{l+1}$ perché ...

Serve comunque solo n piccoli certificati che certificano se un vertice è o non è raggiungibile.

Un $O(\log(n))$ spazio possono contare $r_{l+1} = |R_{l+1}|$

Resta da specificare come fare a certificare una cosa del tipo

$$v_i \in R_{l+1} \dots$$

Il certificato è un cammino $s \to v_i$ in al più $l+1$ passi. Per fare ciò V controlla che il cammino sia valido e di lunghezza corretta, usando $O(\log(n))$ spazio.

$$v_i \notin R_{l+1}$$

per verificare ciò, sappiamo già (per ipotesi) che $v_i \notin R_l$, il certificato ricorda a V tutti i vertici in R_l, può essere verificato usando solo r_n usando $O(\log(n))$ spazio. ■