$\mathcal{P}$-completezza di CIRCUIT VALUE

Mostriamo che $\forall I \in \mathcal{P} \;.\; I \leq_{\text{LogSpace}}$ CIRCUIT VALUE.

Data una macchina M a un nastro che decide I in tempo polinomiale (se è a k nastri possiamo applicare il teorema di riduzione dei nastri) e un input x, possiamo costruire la *tabella di computazione* T:

- se M si arresta in al più $n^k - 2$ passi ($n = |x|$), T è una matrice $n^k \times n^k$ di simboli, in cui la riga i contiene il nastro dopo i passi di calcolo;
- i simboli sono presi dall'alfabeto $\Sigma' = \Sigma \cup (\Sigma \times Q)$ e c'è un unico simbolo della forma (σ, q) in ogni riga, che indica che la testina si trova in quella posizione e lo stato è q;
- la prima colonna contiene i respingenti, e supponiamo che M non la visiti mai (quindi parte dalla seconda cella);
- l'ultima colonna non è raggiungibile per limitazioni di tempo;
- quando la macchina va in uno stato di arresto, la testina viene spostata in seconda posizione, anche passando sopra eventuali respingenti e influendo sul valore di k;
- se nella seconda colonna c'è $(\sigma, \text{Sì}/\text{No})$ alla riga i, tutte le rige successive succesisve sono uguali alla i-esima;
- quindi M accetta l'input se e solo se $T(n^k, 2) = (\sigma, \text{Sì})$.

Se M non rispetta tutte le condizioni necessarie per costruire una tabella con questi vincoli, si può trovare una macchina M' che calcola la stessa funzione e richiede tempo polinomiale.

Ad ogni riga cambiano al più due celle (quella su cui scrive e quella su cui si ferma la testina, che potrebbero essere la stessa). Il valore di $T(i, j)$ dipende solamente da $T(i-1, j-1), T(i-1, j), T(i-1, j+1)$ in base alla funzione di transizione δ.

Costruiamo un circuito $\overline{C}$ che calcola δ.

Per farlo, diamo una rappresentazione binaria per ogni stato $\rho \in \Sigma'$: $\rho \mapsto (s_1, \ldots, s_m) \in \{\text{t}, \text{f}\}^m$, con $m = \lceil \log |\Sigma| \rceil$. Ogni riga di T ha mn^k bit, indicati con $s_{i,j,l}$ (riga, colonna, bit).

Per quanto detto sopra,

$$\begin{aligned} s_{i,j,l} = F_l(&s_{i-1,j-1,1}, \ldots, s_{i-1,j-1,m}, \\ &s_{i-1,j,1}, \ldots, s_{i-1,j,m}, \\ &s_{i-1,j+1,1}, \ldots, s_{i-1,j+1,m}), \end{aligned}$$

e $\overline{C}$ può implementare $F = (F_1, \ldots, F_m)$; sarà un circuito con $3m$ ingressi e m uscite. $\overline{C}$ ha costo costante (dipende da δ ma non da x).

A questo punto si costruisce il circuito C, che ha una copia di $\overline{C}$, opportunamente connessa, per ogni posizione della tabella (tolta la prima riga e la prima e ultima colonna). C ha $3mn^k$ ingressi, e possiamo aggiungere una porta di uscita che è vera se $T(n^k, 2) = (\sigma, \text{Sì})$. C simula l'esecuzione di M, è vero se e solo se $x \in I$.

Per costruire C serve un numero costante di contatori rappresentabili in binario e che non superano n^k, quindi richiede spazio logaritmico.