# Complementi ed Esercizi di Informatica Teorica II

Vincenzo Bonifaci

14 febbraio 2008

## 3 Problemi PSPACE-completi

Ricordiamo che un linguaggio $A$ è detto PSPACE-completo (rispetto a riduzioni tempo-polinomiali) se:

i. $A \in$ PSPACE;

ii. $A$ è PSPACE-difficile, ovvero per ogni linguaggio $B \in$ PSPACE, si ha che $B \leq_P A$.

Un problema di decisione è PSPACE-completo se lo è il corrispondente linguaggio. Un tipico esempio di problema PSPACE-completo è quello di decidere la soddisfacibilità di una formula booleana con quantificatori (QSAT).

### 3.1 Accettazione sul posto

Il problema *accettazione sul posto* (ASP) è il seguente: data una macchina di Turing $M$ e una stringa $x$, decidere se $M$ accetta $x$ senza mai lasciare le prime $|x|$ celle del nastro. Questo problema è in PSPACE. Infatti, in spazio lineare possiamo simulare il comportamento di $M$ sull'input $x$ tenendo conto del numero di passi e rifiutando l'input se $M$ rifiuta oppure se $M$ oltrepassa le prime $|x|$ celle del nastro oppure se opera per più di $|K||x||\Sigma|^{|x|}$ passi ($K$ è l'insieme degli stati di $M$, $\Sigma$ l'alfabeto di nastro), dato che nell'ultimo caso $M$ sta sicuramente ciclando.

Ora sia $L$ un linguaggio in PSPACE accettato in spazio $n^k$ da una macchina $M$. Ovviamente, $M$ accetta $x$ se e solo se accetta *sul posto* la stringa $x\sqcup^{n^k}$ (la stringa $x$ seguita da $n^k$ simboli di blank). Quindi $x \in L$ se e solo se $(M, x\sqcup^{n^k}) \in$ ASP e dato che la trasformazione può essere realizzata in tempo polinomiale, $L \leq_p$ ASP, il che dimostra che ASP è PSPACE-completo.

### 3.2 Riconoscimento di linguaggi di tipo 1

Ricordiamo che una grammatica context-sensitive $\mathcal{G}$ (grammatica di tipo 1) è costituita da un insieme di produzioni del tipo $\alpha \to \beta$ dove $\alpha \in (N \cup T)^+$ e $\beta \in (N \cup T)^*$, $|\alpha| \leq |\beta|$, avendo indicato con $N$ l'insieme dei simboli non

terminali, con $T$ quello dei simboli terminali e con $|x|$ la lunghezza della stringa $x$. È stato dimostrato che l'insieme dei linguaggi generati da grammatiche di tipo 1 (insieme dei linguaggi context-sensitive) coincide esattamente con la classe NSPACE($n$) (linguaggi riconoscibili in spazio lineare da una macchina di Turing nondeterministica). Mostriamo che il problema di decidere, data una grammatica context-sensitive $\mathcal{G}$ e una stringa $x$, se $x$ appartiene a $L(\mathcal{G})$, è PSPACE-completo. Il linguaggio corrispondente a questo problema di decisione è:

$$L_{cs} = \{(g,x) \mid g,x \in \{0,1\}^*,\ g \text{ codifica una gramm.c.s. } \mathcal{G} \text{ e } x \text{ codifica una stringa di terminali } w \text{ t.c. } w \in L(\mathcal{G})\}.$$

Per mostrare che il problema è in PSPACE, osserviamo che una macchina nondeterministica può, su input $(g,x)$, usando spazio $O(n)$, applicare nondeterministicamente una sequenza di produzioni al simbolo iniziale e poi via via alle stringhe intermedie ottenute fino ad arrivare ad una stringa di lunghezza almeno $|x|$; a quel punto, accetta se la stringa ottenuta coincide con $x$. Questo mostra che $L_{cs} \in$ NSPACE($n$) e dato che

$$\text{NSPACE}(n) \subseteq \text{NPSPACE} = \text{PSPACE}$$

dal teorema di Savitch, il problema è in PSPACE.

Per dimostrare che $L_{cs}$ è PSPACE-difficile, mostriamo prima il risultato intermedio:

$$\forall L \in \text{PSPACE}\ \exists L' \text{ di tipo 1 t.c. } L \leq_p L'.$$

Sia $L \in$ PSPACE un linguaggio accettato da una macchina di Turing in spazio polinomiale $p(n)$. Consideriamo il linguaggio $L' = \{w10^{p(|w|)} \mid w \in L\}$. Questo linguaggio può essere deciso da una macchina $M'$ che: cerca l'ultimo 1 nell'input e chiama $w$ la stringa che precede questo 1; controlla che $w$ sia seguita da $10^{p(|w|)}$ (altrimenti rifiuta); simula la macchina $M$ che decide $L$ su input $w$. Dato che $M$ decide $L$ in tempo $p(|w|)$, $M'$ decide correttamente $L'$ in spazio lineare; quindi per la proprietà ricordata sopra $L'$ è di tipo 1. La riduzione da $L$ a $L'$ è calcolabile in tempo polinomiale.

Ora vogliamo provare che $L' \leq_p L_{cs}$. Dato che $L'$ è di tipo 1, esiste una grammatica context-sensitive $\mathcal{G}$ tale che $L' = L(\mathcal{G})$. Sia $g$ la codifica di questa grammatica. Ad ogni stringa $w$, associamo la stringa $f(w) := (g, w10^{p(|w|)})$. Per costruzione $w \in L \Leftrightarrow w10^{p(|w|)} \in L' \Leftrightarrow f(w) \in L_{cs}$. Poiché la funzione $f$ può essere calcolata in tempo polinomiale abbiamo quindi che per ogni $L$ in PSPACE, $L \leq_p L_{cs}$ come volevasi dimostrare.

## 3.3 Gioco Geografia

Sia $G$ un grafo dato. Il gioco *Geografia* su $G$ è il seguente gioco a due giocatori: il primo giocatore sceglie un arco uscente dal nodo 1; questo arco porta ad un altro nodo, a partire dal quale il secondo giocatore sceglie un arco uscente, che porta ad un altro nodo (che deve essere diverso da quelli già visitati) dal quale il

primo giocatore sceglie un arco uscente, ecc. Il giocatore che per primo si trova a non poter muovere perde. Il problema *Gioco Geografia* chiede di stabilire, dato in input $G$, se il primo giocatore ha una strategia vincente a partire dal nodo 1. Intuitivamente, una strategia vincente per il giocatore $g$ è un insieme di mosse e contromosse tale che quali che siano le risposte dell'altro giocatore, il giocatore $g$ arriva a vincere se segue la strategia.

Il problema è in PSPACE. Infatti, la lunghezza di una sequenza di mosse è pari al più al numero di nodi (quindi lineare nella dimensione dell'input) e inoltre da una data situazione di gioco si possono ricavare in spazio polinomiale tutte le possibili mosse (e posizioni di gioco) successive oppure, nel caso in cui il gioco sia finito, valutare se la situazione è una vittoria per il primo giocatore o per il secondo. Quindi è possibile visitare, usando spazio di lavoro polinomiale, l'albero del gioco, cioé un albero in cui i nodi sono le posizioni e gli archi le mosse; il nodo radice è dato dalla situazione di gioco iniziale mentre le foglie dell'albero corrispondono a situazioni di gioco concluse e possono essere etichettate con `true` o `false` a seconda se si tratta di una posizione vincente o perdente per il primo giocatore; ogni altro nodo può essere considerato un OR se corrisponde ad una situazione in cui deve muovere il primo giocatore e un AND altrimenti. Il valore ottenuto in questo modo alla radice dell'albero è `true` se e solo se il primo giocatore ha una strategia vincente a partire dal nodo 1.

Mostriamo che il problema è PSPACE-completo attraverso una riduzione dal problema della soddisfacibilità di formule booleane con quantificatori: QSAT $\leq_p$ GEOGRAFIA. Sia $\mathcal{F}$ una formula booleana quantificata della forma:

$$\exists x_1 \forall x_2 \exists x_3 \ldots \forall x_k [C_1 \wedge C_2 \wedge \ldots \wedge C_m]$$

dove $C_1, \ldots, C_m$ sono tutte clausole disgiuntive. Creiamo allora un grafo con quattro vertici $a_i, x_i, \overline{x}_i, b_i$ per ogni variabile $x_i$ e $m$ vertici aggiuntivi $c_1, \ldots, c_m$ (i vertici sono quindi in tutto $4k + m$). Aggiungiamo gli archi: da $a_i$ a $x_i$, da $a_i$ a $\overline{x}_i$, da $x_i$ a $b_i$ e da $\overline{x}_i$ a $b_i$ per ogni $i$ da 1 a $k$; aggiungiamo un arco da $b_i$ a $a_{i+1}$ per ogni $i$ da 1 a $k-1$; aggiungiamo un arco da $b_k$ a $c_j$ per ogni $j$ da 1 a $m$. Infine, per ogni clausola $C_j$, aggiungiamo un arco da $c_j$ a ogni vertice corrispondente a uno dei letterali della clausola.

Un qualsiasi percorso da $a_1$ a $b_k$ determina una assegnazione di valori di verità alle variabili. Consideriamo l'assegnazione $x_i :=$ `true` nel caso in cui il percorso passi per il nodo $\overline{x}_i$ e `false` nell'altro caso. I giocatori si alternano nello scegliere questi valori di verità per le variabili (il primo sceglie le variabili quantificate esistenzialmente e il secondo quelle quantificate universalmente); arrivati alla fine, il secondo giocatore sceglie un nodo $c_j$, intuitivamente con lo scopo di mostrare che la clausola corrispondente $C_j$ non è soddisfatta dall'assegnazione. I nodi ora disponibili per il primo giocatore sono quelli corrispondenti ai letterali presenti nella clausola e il primo giocatore a questo punto vince se e solo se tra questi c'è un letterale posto a `true`. Il primo giocatore ha quindi una strategia vincente a partire dal nodo $a_1$ se e solo se la formula quantificata è soddisfacibile.

**Esercizio 3.1.** Costruire il grafo associato alla formula

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \ (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3)$$

e verificare che il primo giocatore ha una strategia vincente a partire dal nodo $a_1$.