Automi e Linguaggi Formali
Esame del 5 Luglio 2018 – Parte II

Tempo a disposizione: 1 h 30 min

1. Dare la definizione dei linguaggi L_e e L_{ne}. Spiegare come si dimostra che L_{ne} non è ricorsivo. Si usa una riduzione.

$L_e = \{M \mid L(M) = \emptyset\}, L_{ne} = \{M \mid L(M) \neq \emptyset\}$. Si dimostra che L_{ne} non è ricorsivo, usando una riduzione da L_u a L_{ne}. Un'istanza di L_u è costituita da una coppia (M, w) e la coppia appartiene a L_u se $w \in L(M)$. Da (M, w) costruiamo un'istanza M' di L_{ne} (infatti le istanze di L_{ne} sono macchine di Turing) come segue: per un qualsiasi input x, M' lo sostituisce con w e poi simula M con input w. Se M accetta, allora M' accetta x (quindi $M' \in L_{ne}$), se M non accett,a M' fa lo stesso (quindi $L(M') = \emptyset$). E' facile capire che $(M, w) \in L_u$ sse $M' \in L_{ne}$. Visto che L_u è indecidibile (RE, ma non ricorsivo), la riduzione appena descritta mostra che anche L_{ne} è indecidibile e quindi non ricorsivo.

2. Quali proprietà dei linguaggi RE sono dette triviali?

Una proprietà P sugli RE è triviale (o banale) se $L_P = \emptyset$ oppure $L_P = \{L \mid L \in RE\}$. Insomma P è banale se non è soddisfatta da alcun linguaggio RE oppure se è soddisfatta da tutti i linguaggi RE.

3. Il linguaggio $L = \{a^k b^{2k} c^{3k} \mid k \geq 0\}$ è CF o non CF? Nel primo caso fornire una CFG che genera L (o un PDA che lo riconosce). Nel secondo caso dimostrare che L non è CF.

L non è CF e lo si dimostra usando il pumping lemma dei CFL. Se L fosse CF, allora ci sarebbe un n >0 tale che ogni stringa $w \in L$ con $|w| \geq n$, avrebbe la struttura $w = uvxyz$ con $|vxy| \leq n$ e $vy \neq \epsilon$ e inoltre tutte le stringhe, uv^ixy^iz, con $i \geq 0$ sarebbero in L. Consideriamo la stringa $w = a^n b^{2n} c^{3n} \in L$, ovviamente $|w| > n$ e quindi $w = uvxyz$ con le proprietà ricordate prima. Ora, la parte centrale di w, vxy può consistere di soli a, b o c, oppure di a e b o di b e c, ma in nessun caso di tutti e 3 i simboli a, b e c. Quindi nelle stringhe uv^ixy^iz con $i > 0$ è impossibile che il numero dei 3 simboli continui a soddisfare la condizione richiesta per essere in L. In particolare, l'unico simbolo oppure i 2 simboli che vengono "pompati" aumentando il valore di i, cresceranno, mentre il simbolo, o i 2 simboli non "pompati" resteranno inalterati. Per cui ci sono i tali che $uv^ixy^iz \notin L$ e quindi L non è CF.

4. Descrivere un PDA che accetta per pila vuota e che riconosca il seguente linguaggio $L = \{a^n b^m \mid 0 \leq n \leq m \leq 2n\}$. E' possibile costruire il PDA passando prima per una CFG che genera L. In questo caso è richiesta una dimostrazione o almeno una spiegazione convincente del fatto che la CFG generi veramente L.

Costruiamo direttamente il PDA P richiesto , senza passare per la CFG che genera L. P ha gli stati q_a e q_b e le seguenti transizioni:

$\delta(q_a, a, Z) = \{(q_a, aZ), (q_a, aaZ)\}, \delta(q_a, a, a) = \{(q_a, aa), (q_a, aaa)\}, \delta(q_a, b, a) = \{(q_b, \epsilon)\},$

$\delta(q_a, \epsilon, Z) = \{(q_a, \epsilon)\},$

$\delta(q_b, b, a) = \{(q_b, \epsilon)\}, \delta(q_b, \epsilon, Z) = \{(q_b, \epsilon)\}$

L'idea è semplice: quando si vede un a in input, esso può contare come 1 solo a oppure come 2 a. Questi 1/2 a sono inseriti sullo stack. Quando iniziano i b dell'input, per ogni b si fa il pop di una a. Se l'input è in L, c'è una sequenza di scelte di 1/2 a inseriti nello stack che fa coincidere il numero di a messi sullo stack con il numero di b della seconda parte dell'input. Per questa scelta, dopo aver considerato l'intera stringa, lo stack conterrà Z e l'ultima transizione con q_b lo svuota bloccando il calcolo di P. C'è anche la transizione $\delta(q_a, \epsilon, Z) = \{(q_a, \epsilon)\}$ che svuota lo stack e serve ad accettare la parola vuota.

La costruzione era più semplice passando per una CFG che genera L. Una tale grammatica potrebbe essere la seguente: $S \to aSb \mid aSbb \mid \epsilon$. Dalla grammatica si produce il PDA seguendo la costruzione vista nel corso. Che la grammatica data generi strighe in L è semplice da vedere, visto che ogni produzione genera 1 a e, corrispondentemente, o 1 o 2 b. E' anche facile convincersi che la grammatica produce tutto L, infatti per ogni stringa di L è semplice trovare una derivazione della grammatica che la genera. Sia $a^n b^{n+k} \in L$, con $0 <= k <= n$. Allora una derivazione che deriva questa stringa parte da S e applica k volte la produzione $S \to aSbb$, dopo di che applica $n-k$ volte la produzione $S \to aSb$ e termina con $S \to \epsilon$. La prima parte della derivazione produce $S \Rightarrow^* a^k S b^{2k}$ e la seconda parte aggiunge $n-k$ a e b: $S \Rightarrow^* a^k a^{n-k} S b^{n-k} b^{2k} \Rightarrow a^k a^{n-k} b^{n-k} b^{2k} = a^n b^{n+k}$. Ovviamente la grammatica è molto ambigua, ma questo non ha alcuna importanza per lo scopo per cui si intende usarla.

5. "Colorare" i vertici di un grafo significa assegnare etichette, tradizionalmente chiamate "colori", ai vertici del grafo in modo tale che nessuna coppia di vertici adiacenti condivida lo stesso colore. Il problema kCOLOR è il problema di trovare una colorazione di un grafo non orientato usando k colori diversi.

(a) Dimostrare che il problema 4COLOR (colorare un grafo con 4 colori) è in NP fornendo un certificato per il *Sì* che si può verificare in tempo polinomiale.

(b) Mostrare come si può risolvere il problema 3COLOR (colorare un grafo con 4 colori) usando 4COLOR come sottoprocedura.

(c) Per quali valori di k il problema kCOLOR è NP-completo?

☐ Per nessun valore: kCOLOR è un problema in P

☐ Per tutti i $k \geq 3$

☐ Per tutti i valori di k

Le risposte seguono:

a) se i vertici del grafo sono numerati da 1 a n, allora una sequenza di lunghezza n dei 4 colori disponibili, dove il colore in posizione i della sequenza è associato al vertice i, è un certificato. Per verificare che la colorazione corrispondente al certificato ha risposta SI, basta verificare che il vertice i abbia colore diverso da tutti i vertici a cui è collegato e questa operazione è lineare nella taglia del grafo , visto che basta esaminare ogni arco del grafo 2 volte: una per ciascuno dei 2 vertici collegati dall'arco.

b) Si riduce 3COLOR a 4COLOR come segue: dato un qualsiasi grafo G, istanza di 3COLOR, si aggiunge a G un vertice collegato a tutti i vertici di G. Il grafo G' così ottenuto è un'istanza di 4COLOR e infatti G è colorabile con 3 colori sse G' lo è con 4 colori. Per cui 4COLOR è almeno altrettanto intrattabile di 3COLOR.

c) I problemi kCOLOR con $k \geq 3$ sono NP-completi.