Soluzioni Tutorato 5

Giulio Umbrella

Ex 1

Dimostrare che il seguente linguaggio A = $\{0^n | n = 2^k, k \in \mathbb{N}\}$ non e' regolare.

Per dimostare l'affermazione procediamo per assurdo usando il Pumping Lemma; cioe' dimostriamo che esiste una parola che una volta pompata non appartiene al linguaggio.

Il procedimento e' simile a quanto visto in precedenza

1. Assumiamo che il linguaggio sia regolare
2. Scegliamo una parola che rispetti la lunghezza minima
3. Suddividiamo la parola in xyz
4. Troviamo i tale che $xyiz$ non appartiene al linguaggio

Per questo esercizio, la suddivisione della parola e' immediata, dato un qualsiasi k ottengo $w = 0^k$ e quindi ottengo automaticamente che

- $y = 0^b, b > 0$
- $x = 0^a, a + b \leq k$
- z e' formata solo zeri

Applicando il pumping lemma con i = 0 otteniamo $xy^0z = xz = 0^{2^k - b}$

Il problema e' nella scelta del valore da dare ad i. Per alcuni tipi di esercizi, la scelta di i e' molto semplice, ad esempio per il linguaggi delle parole palindrome abbiamo che se $w = 0^k0^k$, basta impostare i = 0 per ridurre la lunghezza della prima meta' e ottere una parola che non appartiene al linguaggi.

In questo esercizio invece dobbiamo prestare piu' attenzione. Esistono infatti dei valori di i che possono produrre una parola che appartiene al linguaggio. Ad esempio, $2^6 - 32 = 32 = 2^5$ e quindi la parola appartiene al linguaggio.

Quindi non basta ridurre il numero di simboli, dobbiamo dimostare che la parola che si ottiene non appartiene al linguaggio per un generico K. Nella dimostrazione assumiamo che il lunguaggio sia regolare e che esista la costante del pumping lemma k, senza fare ulteriori assunzioni. La dimostrazione e' quindi paramtrizata sul valore di k, e quindi dobbiamo trattarla come un parametro generico.

Una possibile dimostrazione e' la seguente:

- $w = 0^{2^k}$
- $x = \epsilon$
- $y = 0^p, 0 < p < k$
- $z = 0^{2^k - p}$

Possiamo considerare le potenze di due come $2^2, 2^3, \ldots, 2^k, 2^{k+1}, \ldots$, se riusciamo a pompare una parole per produrne una di lunghezza compresa fra 2^ke 2^{k+1}, abbiamo produtto una parola che non appartiene al linguaggio.

Se prendiamo i = 2, $xy^2z = (0^p)^2 0^{2^k - p} = (0^{2p})0^{2^k - p} = 0^{2^k}0^p$

Ora dobbiamo dimostare che la parola non puo' appartenere al linguaggio. Se prendiamo la parola appartenente al linguaggio successiva 02^{k+1} la possiamo scomporre come $0^{2^{k+1}} = 0^{2^k 2} = 0^{2^k + 2^k} = 0^{2^k}0^{2^k}$

Dato che p e' sicuramente minore di 2^k, la parola non appartiene al linguaggio

Ex 2

Dati due linguaggi A e B definiamo lo shuffle dei due linguaggi come $\{w | w = a_1 b_1, \ldots, a_k b_k, \text{ con }, a_1, \ldots, a_k \in A, b_1, \ldots, b_k \in B \text{ con } a_i, b_i \in \Sigma\}$.

Dimostrare che se A e B sono regolari, lo shuffle di A e B e' un linguaggio regolare.

Dimostrazione informale

Per dimostrare che un linguaggio e' regolare, basta produrre un automa a stati finiti. A differenza di altri esercizi in cui abbiamo una precisa definizione del linguaggio (ad esempio stringhe binarie di lunghezza pari), in questo contesto sappiamo solo che il linguaggi e' derivato da altri linguaggi.

Questo esercizio va percio' affrontato in modo generale; sappiamo infatti solo che A e B sono regolari, ma non abbiamo indicazioni di nessun tipo sulla loro struttura. Dobbiamo costruire una dimostrazione di carattere generale che funziona per ogni linguaggio regolare A e B.

Sappiamo che A e B sono due linguaggi regolari quindi sappiamo che esistono due automi a stati finiti che li rappresentano. Attenzione, sappiamo che i due automi esistono, ma non abbiamo informazioni su cosa contengono.

Dati i due automi D_A e D_B definiamo

- $D_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$
- $D_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$

Utilizzando questi automi, costruiamo un automa D_S che riconosce le parole composte dallo shuffle di due parole di A e B. Possiamo pensare che D_S ha a disposizione D_A e D_B, quindi l'automa puo' richiamarne le funzioni di transizione. Sappiamo che nell'input i caratteri sono alternati fra la parola in A e quella in B.

L'ordine delle operazioni e' il seguente:

1. L'automa D_S parte dallo stato iniziale di A e B
2. L'automa D_S legge il primo input e richiama le funzioni di transizioni di D_A e aggiorna lo stato, infatti sappiamo che il primo simbolo appartiene al linguaggio A ; in questa fase D_B rimane allo stato iniziale
3. Il secondo input per costruzione appartiene al linguaggio B; l'automa D_S lascia D_A immutato e aggiorna D_B.
4. La procedura riparte aggiornando D_A

In questo modo, le due parole di A e B vengono processate in parallelo ma in modo asincrono.

Dimostrazione formale

Per la dimostrazione formale dobbiamo define i cinque elementi di un automa a stati finito. L'alfabeto Σ e' lo stesso dei due automi.

Funzione di transizione

La funzione di transione e' definita nel seguente modo

$\delta((x, y, A), a) = (\delta_A(x, a), y, B)$

Input: la funzione prende input una tupla di tre valori e l'input corrent

- x stato corrente di $D_A, x \in Q_A$
- y stato corrente di $D_B, y \in Q_B$
- A flag per indicare che l'input a deve essere processato usando D_A

Output: la funzione produce in output una tupla di tre elementi - (x,a) nuovo stato di D_A a partire da x con input a - y stato corrente di D_B - B flag per indicare che l'input deve essere processato usando l'automa D_B

Definiamo poi in modo simile per i passaggi da B ad A con $\delta((x, y, B), b) = (x, \delta_B(y, b), A)$.

Possiamo notare i seguenti punti:

- La procedura aggiorna D_A ma lascia immutato $D_B(oviceversa)$.
- Il flag A viene cambiato in B per indicare che il prossimo input deve essere processato usando D_B.
- Il meccanismo della funzione di transione di D_S e ' sempre lo stesso: la funzione prende in input uno stato e simbolo e produce in output uno stato.
- Gli stati di D_S sono identificati da tuple di 3 elementi

Insieme di stati

Per completare l'automa, dobbiamo definirne gli stati:

$Q_S = Q_A$ x Q_B x {A,B}

NB: X e' il prodotto cartesiano fra insiemi, ossia l'insieme prodotto da tutte le coppie di singoli elementi. Ad esempio {a,b} X {c,d} = {{a,c), (a,d), (b,c), (b,d)}

In questo modo stiamo creando tutte le possibili combinazioni di coppie di stati e flag. Non tutti gli stati saranno necessariamente percorsi, ma in questo modo stiamo creando tutte le possibili combinazioni di stati in cui possono essere A e B.

Stato iniziale

q = (q_A, q_B, A), ossia gli automi vengono processati a partire dai loro rispetti stati iniziali, e il primo automa ad essere processato e' A.

Stato accettante

F = F_A x F_B x {A}, l'automa accetta se entrambi gli stati sono accettanti; il flag A serve ad indicare che il prossimo input e' dal linguaggio A, e quindi l'input letto in precedenza era da B.

Ex 3

Sia L un linguaggio regolare su un alfabeto Σ con $\# \in \Sigma$ e sia dehash(w) la funzione che rimuove il simbolo hash dalla stringa. Ad esempio dehash(1#1) = 11, dehash(0#10#) = 010. Dimostrare che il linguaggio dehash(L) = $\{dehash(w) : w \in L\}$ e' regolare.

Ex 4

Sia L un linguaggio regolare su un alfabeto Σ. Dimostrare che il linguaggio suffixes(L) = $\{y|xy \in L$per qualche stringa x $\in \Sigma^*\}$ e' regolare.

Ex 5

Esercizi aggiuntivi

1. Dimostrare che il linguaggio $\{0^m1^n|$n/m e' un numero intero$\}$ non e' regolare

2. Siano L e M due linguaggi regolari su alfabeto {0,1}. Dimostare che il linguaggio L&M = $\{x\&y | x \in L, y \in M, |x| = |y|\}$, dove x&y e' l'and logic bit a bit. Per esempio, 101&001 = 001.
3. Siano L e M due linguaggi regolari su alfabeto {0,1}. Dimostare che il linguaggio $LXORM = \{xXORy | x \in L, y \in M, |x| = |y|\}$, dove $xXORy$ e' l'and logic bit a bit. Per esempio,$101XOR001 = 100$.