Tutorato AFL

Linpeng Zhang

24 maggio 2019

Sommario

Per errori/dubbi/problemi: linpeng.zhang@studenti.unipd.it.

Indice

1 Lez10

1.1 Esercizi

1. Il seguente linguaggio è regolare? $L = \{a^k b^k c^i | i \leq k\}$;
2. Il seguente linguaggio è context free? $L = \{a^k b^k c^i | i \leq k\}$;
3. Il seguente linguaggio è regolare? $L = \{$ stringhe con un numero pari di a $\}$;
4. Il seguente linguaggio è context free? $L = \{$ stringhe con un numero pari di a $\}$;

1.2 Soluzioni

1. no, dimostrare con il PL per i linguaggi regolari;
2. no. Una traccia della dimostrazione è la seguente:
   Sia (per assurdo) L un linguaggio CFL. Sia n la lunghezza data dal PL. Prendiamo una stringa z tale che $|z| \geq n$. Sia $z = a^n b^n c^n = uvwxyz$, $|vwx| \leq n, vx\ neq\epsilon$.
   Come è fatto vx? Da al più due caratteri contigui tra (a, b, c). Se è fatto da soli a, allora $z = uv^0wx^0y =$ chiaramente $\#a \neq \#b$. Se è fatto da soli b, allora $z = uv^0wx^0y =$ chiaramente $\#a \neq \#b$. Se è

fatto da soli c, allora $z = uv^2wx^2y =$ chiaramente $\#c > \#a$ (oppure b). Se vx è composto da a e b, allora $z = uv^0wx^0y =$ chiaramente $\#c > \#a$ (oppure b). Se vx è composto da b e c, allora $z = uv^2wx^2y =$ chiaramente $\#c > \#a$.

3. sì. Una regexp è $R = (aa)^*$;

4. sì: un linguaggio regolare è context free;

1.3 Traccia alle risposte delle domande della simulazione

1. si dimostra per induzione sul numero di derivazioni; con una derivazione si deriva ϵ che banalmente soddisfa la proprietà; con $n+1$ derivazioni, dopo la prima si ha aS oppure $aSbS$ e con altre n derivazioni si ha aw' oppure $aw'bw''$ con w', w'' che soddisfano la proprietà. Naturalmente anche aw' la soddisfa (visto che si aggiunge una a all'inizio) e lo stesso per $aw'bw''$, visto che si aggiunge una a all'inizio e una b che è matchata dalla a aggiunta all'inizio;

2. $$\begin{array}{l} \delta(q, \epsilon, S) = \{(q, aS), (q, aSbS), (q, \epsilon)\} \\ \delta(q, a, a) = \{(q, \epsilon)\} \\ \delta(q, b, b) = \{(q, \epsilon)\} \end{array}$$

3. un DPDA che accetta per stati finiti riconosce tutti i linguaggi regolari (basta ignorare lo stack), ma non tutti i CFL: ad esempio ww^r no e un esempio è data dalle stringhe 0110 e 01100110. L'automa deterministico dopo aver letto la prima stringa (che dovrebbe accettare) deve aver consumato lo stack (prima mette 01 in pila poi leggendo 10 matcha lo stack). Ma allora se leggesse 01100110 come si dovrebbe comportare? Avrebbe già svuotato lo stack consumando solo metà input!
   Un DPDA che accetta per stack vuoto accetta solo linguaggi accettati da DPDA che hanno la proprietà del prefisso. L'intuizione è simile: se il DPDA legge x, deve averlo consumato. Quindi non può leggere x seguito da qualcos'altro, ovvero deve esserci sempre la proprietà del prefisso.
   Importanza: si è dimostrato che se un DPDA riconosce un linguaggio, allora tale linguaggio non è inerentemente ambiguo.

4. è chiaro che l'automa non svuoti mai la pila, quindi si presume che riconosca per stato finale. Quando legge una a la mette in cima. Quando legge una b, se c'è una a in cima la toglie, altrimenti si blocca (non ha transizioni). Quindi si può intuire che l'automa riconosca l'insieme delle stringhe per cui ogni prefisso ha un numero di a maggiore o uguale al numero di b. Usare la costruzione vista a lezione per la conversione.

5. (a): il consiglio presente nella consegna vi dice già come risolvere l'esercizio. Caso base: 1 derivazione, si deriva la stringa ϵ che è banalmente bilanciata. Con n+1 derivazioni, la prima può usare una delle due produzioni. Seguono n derivazioni, che portano ad avere o la concatenazione di due stringhe bilanciate o del tipo (w) con w bilanciato. Naturalmente in entrambi i casi la proprietà è rispettata e si ha la tesi.
(b): non genera ad esempio ()().

6. $[qXq] \rightarrow a[qYq][qZq]$
   $[qXq] \rightarrow a[qYp][pZq]$
   $[qXp] \rightarrow a[qYq][qZp]$
   $[qXp] \rightarrow a[qYp][pZp]$

7. non fatto;

8. non fatto;

9. se una grammatica ha h simboli, allora una parola w lunga almeno 2^h richiederà che due simboli si ripetano nell'albero di derivazione. Sia A il simbolo che si ripete, allora l'albero radicato in A deriva o xAy o w. Chiaramente si può ripetere "xAy" infinite volte. Si ha quindi il PL.