Algoritmi e Principi dell'Informatica

Soluzioni al Tema d'esame
27 Gennaio 2021

1 Informatica teorica

Esercizio 1
Si considerino i seguenti linguaggi sull'alfabeto $A = \{a, b, c\}$:

$$L_1 = A^* \cdot \{b\} - (A^* \cdot (A - \{a\}) \cdot A^* \cdot \{b\}) \quad (1)$$
$$L_2 = A^* - (A^* \cdot (A - \{b\}) \cdot A^*) \quad (2)$$
$$L_3 = L_1 \cdot L_2 \quad (3)$$

dove * è la stella di Kleene, $-$ è la differenza insiemistica e $\cdot$ è la concatenazione.

Utilizzare un formalismo a potenza minima (tra tutti quelli visti a lezione) che caratterizzi il linguaggio L3.
SOLUZIONE

Si può facilmente verificare che $L_1 = a^* \cdot b$ e $L_2 = b^*$. Pertanto $L_3 = a^* \cdot b^+$. È quindi un linguaggio regolare (caratterizzato, per l'appunto, dall'espressione regolare data). Tuttavia si tratta anche di un linguaggio di tipo *star-free*, poiché, come si vede dalle espressioni originali, la stella di Kleene è applicata solo sull'intero alfabeto A. Si può quindi formalizzare il linguaggio L_3 mediante una formula MFO, come segue:

$$\begin{array}{cl} \exists x(x = 0 \wedge (a(x) \vee b(x))) & \text{all'inizio c'è una } a \text{ o una } b \\ \wedge & \\ \forall x(a(x) \rightarrow \exists y(y = x + 1 \wedge (a(y) \vee b(y)))) & \text{dopo una } a \text{ c'è una } a \text{ o una } b \\ \wedge & \\ \forall x(b(x) \rightarrow (last(x) \vee \exists y(y = x + 1 \wedge b(y)))) & \text{dopo una } b \text{ c'è una } b \text{ o è l'ultimo carattere} \\ \wedge & \\ \exists x(b(x) \wedge last(x)) & \text{l'ultimo carattere è una } b \text{ (congiunto ridondante)} \end{array}$$

Esercizio 2

1. Dire se è decidibile il problema di stabilire se, data una MT deterministica e una sequenza di suoi stati, esiste una stringa x in ingresso tale che la MT attraversa esattamente, uno per uno, la sequenza di stati desiderata durante il riconoscimento di x.
2. Dire se è semidecidibile il problema del punto 1.
3. Dire se è decibile il problema di stabilire se, data una MT deterministica e una sequenza di suoi stati, esiste una stringa x in input tale che la MT, durante il riconoscimento x, attraversa gli stati desiderati nell'ordine desiderato, ma potrebbe, tra uno stato desiderato e il successivo della sequenza, attraversarne anche altri.

Soluzione

1. Decidibile: se esiste la stringa x, essa può essere ricostruita seguendo le mosse che la MT dovrebbe fare per attraversare la sequenza di stati.
2. Semidecidibile, per quanto detto sopra.
3. Indecidibile: è sufficiente notare che, prendendo come coppia di stati da cui partire ed in cui arrivare lo stato iniziale e uno stato finale, possiamo determinare se la funzione calcolata dalla MT è definita in almeno un punto, problema notoriamente indecidibile (come si può evincere anche applicando il teorema di Rice).

2 Algoritmi e strutture dati

Esercizio 3
Si consideri il seguente problema: data una stringa x costruita su un alfabeto A, restituire il primo simbolo di x che NON compare più di una volta in x.

1. Si descriva una MT a k nastri che risolve il problema dato, e se ne diano le complessità temporale e spaziale.
2. Si descriva una macchina RAM che risolve il problema dato, e se ne diano le complessità temporale e spaziale, sia a costo costante che a costo logaritmico.

Soluzione

- Si osservi che i possibili stati della computazione attraversati dalla MT sono in numero finito. In particolare essi sono determinati dal memorizzare le seguenti informazioni:
  - per ogni carattere dell'alfabeto, se esso è stato visto nessuna, una o più di una volta. L'informazione è codificabile in modo semplice in $3^{|A|}$ distinti valori.
  - L'ordine con cui vengono visti i caratteri. L'informazione è codificabile in $(|A|)!$ distinti valori.

  Ricordando che l'alfabeto ha cardinalità finita, è sufficiente un automa a stati finiti con $3^{|A|} \times (|A|)!$ stati per eseguire il calcolo. L'automa utilizza la memoria a stati finiti per tenere traccia di tutte le informazioni necessarie mano a mano che scorre la stringa. La complessità temporale è $O(n)$, quella spaziale è costante. La MT desiderata emula il comportamento dell'automa a stati finiti.
- Come sopra. È sufficiente emulare un automa a stati finiti tramite la macchina RAM.

Esercizio 4

Si consideri un suggeritore per cercare parole che rimano con una parola data all'interno di un dizionario. In particolare, è possibile interagire con il suggeritore tramite due funzioni.

- aggiungi(p): viene fornita al suggeritore una parola p da aggiungere al dizionario. Si denoti con n lunghezza della parola.

- TROVARIME(p, r): il suggeritore stampa tutte le parole che rimano con la parola p data (di lunghezza n) per i loro ultimi r caratteri.

Si realizzino le due funzioni, AGGIUNGI, TROVARIME.

SOLUZIONE

È possibile memorizzare il dizionario come un albero con branching factor pari al numero di lettere dell'alfabeto. I nodi dello stesso hanno quindi un numero di puntatori pari alle lettere dell'alfabeto. Ogni nodo contiene anche un campo a valore booleano che indica se il nodo è corrispondente all'inizio di una parola. Ogni parola viene memorizzata, letta dalla fine all'inizio, come un percorso all'interno dell'albero con il seguente procedimento:

1. si imposta la radice come nodo corrente e l'ultima lettera della parola come lettera corrente
2. si aggiunge, se non già presente, un nodo come i-esimo figlio del nodo corrente, dove i è il numero d'ordine della lettera corrente.
3. si imposta il nodo corrente a quello appena aggiunto (o a quello che si sarebbe dovuto aggiungere ed era già presente)
4. se la lettera corrente non è la prima della parola si torna al punto 2, altrimenti si imposta il valore booleano a `vero`.

Il costo di una chiamata a AGGIUNGI è quindi lineare nella lunghezza della parola da aggiungere.

La funzione TROVARIME(p, r), è realizzata con il seguente procedimento:

1. partendo dalla radice dell'albero come nodo corrente, e dall'ultima lettera della parola come lettera corrente, per r volte
   (a) controlla se è presente l'i-esimo figlio del nodo corrente, dove i è il numero d'ordine della lettera corrente.
   (b) se non è presente, la funzione termina la sua esecuzione senza stampare nulla, altrimenti imposta il figlio come nodo corrente e la lettera precedente nella parola a quella corrente
2. effettua una visita in profondità dell'albero, a partire dal nodo raggiunto stampando ogni percorso fino ad ogni nodo con valore booleano impostato a `vero` in ordine inverso, seguito dalle ultime r lettere della parola p.

Le parole stampate sono tutte e sole quelle che hanno un suffisso da r caratteri comune alla parola data. Il costo della funzione è proporzionale ai nodi dell'albero visitati, dunque lineare nella somma dei caratteri delle parole che rimano con quella data.