Algoritmi e Principi dell'Informatica

Appello del 7 Luglio 2014

Chi deve sostenere l'esame integrato (API) deve svolgere tutti gli esercizi in 2 ore e 30 minuti.

Chi deve sostenere solo il modulo di *Informatica teorica* deve svolgere gli Esercizi 1a, 1b e 2 in 1 ora e 15 minuti.

Chi deve sostenere solo il modulo di *Informatica 3* deve svolgere gli Esercizi 3 e 4 in 1 ora e 15 minuti.

NB: i punti attribuiti ai singoli esercizi hanno senso solo con riferimento all'esame integrato e hanno valore puramente indicativo.

Esercizio 1a (punti 5)

Si consideri il linguaggio L_k, dove k è un parametro, costituito da stringhe sull'alfebto {a, b} tali che ogni volta che si trovino k a consecutive ne segua immediatamente almeno una b.

Si costruisca una rete di Petri che accetti L_3. NB: è preferibile che la rete non contenga transizioni che siano sia in ingresso che in uscita al medesimo posto.

Esercizio 1b (punti 5)

Si scriva una formula del prim'ordine che specifichi un generico linguaggio L_k per un dato valore del parametro k (la formula deve quindi dipendere da k, variabile libera) dell'esercizio 1.a usando variabili intere che indichino la posizione di un carattere nella stringa e i predicati a(i) e b(i) per indicare che il carattere in posizione i-esima è a o b, rispettivamente.

Ad esempio la formula $a(1) \wedge b(2)$ indica che la stringa deve inziare con ab.

NB: qualora fosse utile si può fare uso del predicato $\perp$ (indefinito) per indicare l'assenza di carattere in posizione i: per convenzione si può assumere che valga $\perp$ (0) e che per una stringa lunga n valga $(a(n) \vee b(n)) \wedge \perp (n+1)$.

Esercizio 2 (punti 7)

Una macchina di Turing M_i (che calcola la funzione f_i) è detta *riproducibile* se esiste un'altra macchina di Turing M_j (che calcola la funzione f_j) tale per cui $f_i=f_j$.

Si consideri l'insieme delle MT definite sull'alfabeto di due caratteri {0, 1}. Al suo interno siano:

- F l'insieme delle funzioni calcolate da macchine di Turing riproducibili.
- G l'insieme delle funzioni calcolate da macchine di Turing riproducibili e con meno di 10 stati.
- H l'insieme delle funzioni calcolate da macchine di Turing riproducibili e con più di 10 stati.

Dire se i seguenti problemi sono decidibili:

1) Stabilire se una generica macchina di Turing calcoli una funzione in F.

2) Stabilire se una generica macchina di Turing calcoli una funzione in G.

3) Stabilire se una generica macchina di Turing calcoli una funzione in H.

Esercizio 3 (punti 7)

Si consideri la traduzione $\tau(x) = a^k b^h$, dove $x \in \{a,b\}^+$, k è il numero di b in x e h il numero di a in x.

Si descriva il funzionamento di un MT a nastro singolo che implementi τ usando esclusivamente le celle di memoria contenenti x. Se ne valutino la complessità temporale e spaziale.

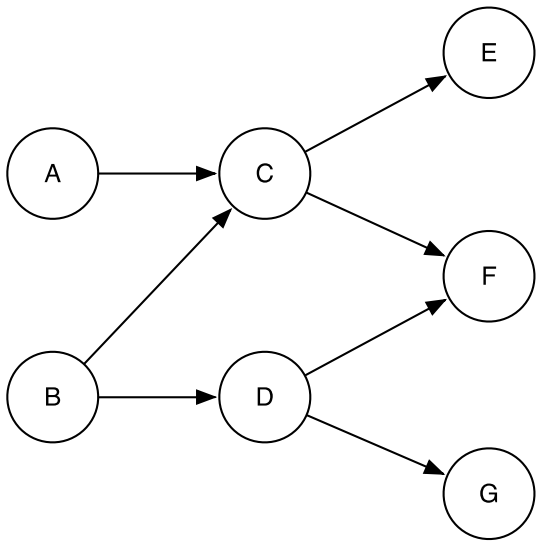
Esercizio 5 (punti 9)

Dati: un elenco di attività e di vincoli di precedenza fra esse

scrivere un algoritmo che rappresenta queste attività in un grafo di precedenze o fallisce se i vincoli di precedenza sono incoerenti (as esempio, se A precede B e B precede A, sia direttamente che indirettamente, i vincoli sono incoerenti).

Indicare la complessità dell'algoritmo

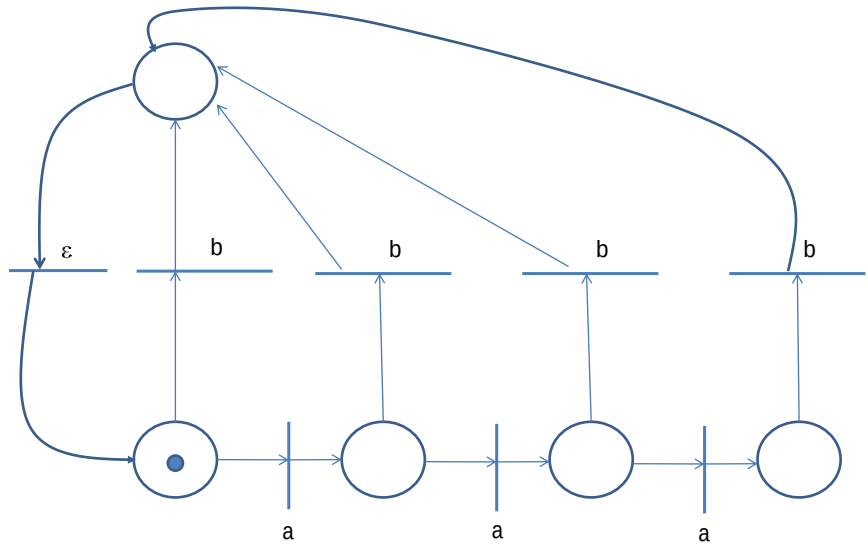
Ad esempio:
Date le 7 attività
A, B, C, D, E, F, G
con i seguenti vincoli di precedenza
A –> C, B –> C, C –> E, D –> F, B –> D, C –> F, D –> G
il grafo di precedenze risultante è



Tracce delle Soluzioni

Esercizio 1°

La rete di Petri seguente, dove è indicata la marcatura iniziale e ogni possibile marcatura seguente viene riconosciuta come finale (un solo token può marcare un solo posto), riconosce il linguaggio L_3.



Esercizio 1.b

$\forall x \, \forall i \, (((x \leq i < x+k) \rightarrow a(i)) \rightarrow b(x+k))$

Esercizio 2

Il problema 1 è decidibile ed è anche deciso. Infatti, ogni macchina di Turing è riproducibile: come è noto, data una funzione calcolabile, esistono infinite macchine di Turing che la calcolano. Quindi F coincide con l'insieme universo di tutte le funzioni calcolabili.

Il problema 2 è indecidibile per il teorema di Rice. Infatti, l'insieme delle macchine di Turing con meno di 10 stati e con alfabeto di due caratteri è finito (proprietà che abbiamo anche usato per enumerare le macchine di Turing), pertanto sarà finito (e non vuoto) anche l'insieme di funzioni calcolabili calcolate da dette macchine di Turing. Allora l'insieme G non è né l'insieme vuoto, né l'insieme di tutte le funzioni computabili, da cui l'indecidibilità per il teorema i Rice.

Il problema posto nel quesito 3 è decidibile per il teorema di Rice. Infatti, ogni funzione calcolabile è calcolabile (anche) mediante una macchina con più di 10 stati. Come scritto in precedenza, per ogni funzione computabile esistono infinite macchine di Turing che la calcolano, mentre le macchine di Turing con un numero di stati minore o uguale a 10 sono in

numero finito, quindi ogni funzione calcolabile dev'essere calcolata anche da macchine con più di 10 stati.

Pertanto H è l'insieme di tutte le funzioni computabili e il problema è allora decidibile per il teorema di Rice.

Esercizio 3

1) la macchina fa una passata scambiando le a con le b e viceversa: complessità ($\Theta(n)$)

2) la macchina fa una passata cercando coppie ba e scambiando la b con la a (scambio: costo costante; costo della passata: $\Theta(n)$)

3) se alla fine della passata non trova coppie ba, si ferma, altrimenti ripete il passo 2)

Al max si fanno n passate, quindi la complessità totale sarà $\Theta(n^2)$.

Esercizio 4

Come suggerito nel testo dell'esercizio, si costruisca un grafo che rappresenti le attività (nodi) e i relativi vincoli di precedenza (archi): i vincoli sono incoerenti se e solo se il grafo contiene cicli. Ciò può essere verificato mediante una opportuna modifica dell'algoritmo di visita Depth-first (costituito da diverse visite ricorsive) che distingua tra i nodi scoperti in una precedente visita e quelli scoperti nella visita corrente; se finisce su uno del primo tipo, la visita corrente termina, e se ne lancia un'altra; nel secondo caso la visita termina restituendo false.

Se l'algoritmo ha successo, cioè non trova cicli, il grafo ottenuto è un DAG e ad esso può essre applicato un topological sort.

FIND-LOOP(G)

/* ritorna flalse se I vincoli sono incoerenti, e quindi se il grafo contiene un ciclo*/

```
1 for each u ∈ G.V
2   u.color := 0
3 c := 0
4 for each u ∈ G.V
5   if u.color = 0
6     c := c+1
7     if FIND-LOOP-VISIT(u, c) = false
8       return false
9 return true
```

FIND-LOOP-VISIT(u, c)

```
1 u.color := c ∈
2 for each v    u.Adj
3   if v.color = c
4     return false
5   if v.color = 0
6     return FIND-LOOP-VISIT(v, c)
7 return true
```

L’algoritmo proposto è quindi O(|V| + |E|) e potrebbe essere incluso come parte integrante in un algoritmo di topological sort senza alterarne la complessità.