Algoritmi e Principi dell'Informatica

Appello del 3 Marzo 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 1, 2 e 3 in 1 ora e 15 minuti.

Chi deve sostenere solo il modulo di *Informatica 3* deve svolgere gli Esercizi 4 e 5 in 1 ora e 15 minuti.

NB1: i punti attribuiti ai singoli esercizi hanno senso solo con riferimento all'esame integrato e hanno valore puramente indicativo.

NB2: l'esercizio 5 verrà valutato solo se negli altri esercizi si sarà ottenuto un punteggio di almeno 11.

Esercizio 1 (punti 4)
Si consideri la seguente grammatica:

S → S a | a S | b B | b
S B → B B S S
B S → a b b a

Quale linguaggio genera? Qual è la classe di automi a potenza minima in grado di accettare lo stesso linguaggio?

Esercizio 2 (punti 4 senza restrizione; punti 8 se si applica la restrizione)

Si specifichi in logica del prim'ordine il seguente linguaggio L:

$L = \{a^n b^{2n} c^{n-1}, \text{ con } n>0\}$

Restrizione: nella scrittura delle formule si può fare ricorso <u>*esclusivamente*</u> alle seguenti funzioni, predicati e costanti la cui definizione può essere data per scontata e quindi da non specificare ulteriormente:

- $x \in L$ (predicato di appartenenza: vero se e solo se *x* appartiene a *L*)
- $s = t$ (predicato di uguaglianza: vero se e solo se *s* è uguale a *t*)
- $s \cdot t$ (funzione che indica la concatenazione delle stringhe *s* e *t*)
- *a* (costante che indica la stringa di un solo carattere "a")
- *b* (costante che indica la stringa di un solo carattere "b")
- *c* (costante che indica la stringa di un solo carattere "c")

Esercizio 3 (punti 7)

E' decidibile il problema di stabilire se, dato un generico trasduttore a stati finiti T e una stringa $y \in O^*$, esista una stringa $x \in I^*$, tale che $y = \tau(x)$, dove:

- O è l'alfabeto di output di T

- I è l'alfabeto di input di T
- τ: I* $\rightarrow$ O* è la traduzione definita da T

?

Giustificare brevemente la risposta.

Esercizio 4 (punti 5)
Qual è il numero massimo di possibili ordinamenti topologici compatibili con un DAG di n nodi?

Esercizio 5 (punti 11)
Si scriva un algoritmo che ordini una coda di interi, supposta già in memoria, assumendo che essa sia utilizzabile come una struttura astratta di cui sia nota solo l'interfaccia, a sua volta costituita **esclusivamente** dalle primitive canoniche ENQUEUE e DEQUEUE (e non altre!). Si valuti la complessità asintotica dell'algoritmo.

NB: sono preferite soluzioni che non implichino aumento della lunghezza della coda durante l'esecuzione e usino il minor numero possibile di celle ausiliarie, **in ogni caso in quantità indipendente dal numero di elmenti in coda**. Per semplicità si può assumere che la coda non contenga elementi ripetuti; questa assunzione però dimunisce il valore della soluzione di 1 punto.

Tracce delle Soluzioni

Esercizio 1

Il linguaggio generato dalla grammatica è L(G) = a*ba*. Infatti le prime due regole generano stringhe del tipo a*Sa*; quando si applica la regola `S → b B,` si ottiene una stringa del tipo a*bBa* che non potrà derivare nessun'altra stringa perché per riscrivere B occorre che esso compaia vicino a un S; quindi per poter generare una stringa terminale è necessario applicare la regola `S → b` che impedisce ulteriori derivazioni.

L(G) è chiaramente regolare anche se G non lo è; di conseguenza esso è riconoscibile da un automa a stati finiti.

Esercizio 2

Soluzione senza restrizione

Il linguaggio L può essere definito con una formula del prim'ordine nel modo seguente:

$\forall x\ (x \in L \leftrightarrow \exists n\ (x = \text{power}(a, n) \cdot \text{power}(b, 2n) \cdot \text{power}(c, n-1) \wedge n > 0))$

dove, al solito, il predicato power(x,n) è defintio dalla formula

$\forall x\ (\ (\text{power}(x,0) = \varepsilon) \wedge\ \forall n\ (n > 0) \rightarrow (\text{power}(x,n) = x \cdot \text{power}(x,n-1)$

Soluzione con restrizione

Se invece sono consentiti solo i costrutti indicati nel testo, conviene definire alcuni linguaggi di supporto:

$\forall x\ (x \in L_a \leftrightarrow x = a \vee \exists y\ (y \in L_a \wedge x = a \cdot y))$ (ossia $L_a = a^+$)

$\forall x\ (x \in L_b \leftrightarrow x = b \vee \exists y\ (y \in L_b \wedge x = b \cdot y))$ (ossia $L_b = b^+$)

$\forall x\ (x \in L_c \leftrightarrow x = c \vee \exists y\ (y \in L_c \wedge x = c \cdot y))$ (ossia $L_c = c^+$)

$\forall x\ (x \in L_1 \leftrightarrow x = a \cdot b \cdot b \vee \exists y\ (y \in L_1 \wedge x = a \cdot y \cdot b \cdot b))$ (ossia $L_1 = a^n b^{2n}$, n>0)

$\forall x\ (x \in L_2 \leftrightarrow x = a \vee \exists y\ (y \in L_2 \wedge x = a \cdot y \cdot c))$ (ossia $L_2 = a^n c^{n-1}$, n>0)

Infine definisco L:

$\forall x\ (x \in L \leftrightarrow x = a \cdot b \cdot b \vee$

$\exists y \exists z \exists w\ (x = y \cdot z \cdot w \wedge y \in L_a \wedge z \in L_b \wedge w \in L_c \wedge y \cdot z \in L_1 \wedge y \cdot w \in L_2))$

Esercizio 3

Premessa: la lunga storia della teoria degli automi dimostra che molto raramente un problema relativo agli automi a stati finiti è indecidibile; ciò grazie alle sue proprietà essenziali, riassunte nel pumping lemma.

Il problema in oggetto non costituisce un'eccezione. Infatti, anche se in generale non è detto che $|x| \leq |\tau(x)|$ a causa di eventuali mosse che non scrivono nulla in output, è possibile enumerare le stringhe di input in modo tale da garantire che l'insieme traduzione di un numero finito di esse contenga tutte le stringhe di output z tali che $|z| \leq |y|$. Precisamente si potrebbe procedere come segue:

Si enumerino tutte le stringhe $x \in I^*$ per lunghezza crescente e se ne calcolino le rispettive $\tau(x)$ finché $|\tau(x)| > |y|$. Se durante tale enumerazione si individua una sottostringa w di x tale

che x = zws, $\delta^*(q_0, z) = q$, $\delta^*(q, w) = q$ e $\eta^*(q, w) = \varepsilon$, w può essere eliminata dalla stringa x senza evidentemente cambiare il risultato della traduzione. Rimangono quindi da prendere in esame solo stringhe "a traduzione crescente" nel senso che per tali stringhe x

$$\exists k(\forall x \forall z(|z| \geq k \rightarrow |\tau(xz)| > |\tau(x)|))$$

Enumerando quindi tutte le stringhe di I* ma escludendo quelle che contengono sottostringhe del tipo w di cui sopra è possibile ottenere tutte le stringhe di lunghezza ≤ |y| che siano traduzione di qualche x. Se tra esse non figura la stringa y, ciò significa che non può essere ottenuta come risultato della traduzione τ.

Esercizio 4

Se il grafo non ha archi qualsiasi permutazione dei nodi è un ordinamento topologico compatile con esso; quindi se ne possono ottenere n!

Esercizio 5

L'impossibilità di usare memoria ausiliaria in quantità arbitraria e la necessità di accedere alla coda solo mediante le primitive canoniche obbliga a far scorrere la coda diverse volte fino ad ottenerne una trasformazione ordinata.

Un semplice algoritmo è il seguente (in JavaScript):

```
function queue_sort(Q) {
    var x, y, n, i, k;
    enqueue(Q, '#');
    n = 0;
    x = dequeue(Q);
    while (x !== '#') {
        enqueue(Q, x);
        x = dequeue(Q);
        n += 1;
    } // n contiene ora la lunghezza di Q

    for (i = 1; i <= n; i += 1) {
        x = dequeue(Q);
        for (k = 1; k < n; k += 1) {
            y = dequeue(Q);
            if (x < y) {
                enqueue(Q, x);
                x = y;
            } else {
                enqueue(Q, y);
            }
        }
```

```
        enqueue(Q, x);
    }
}
```

Questo algoritmo non aumenta la lunghezza originaria della coda, usa le variabili ausiliarie x, n, oltre ai contatori i, k e ha ovviamente complessità $O(n^2)$. Appare decisamente difficile, probabilmente impossibile, ottenere complessità asintotiche migliori di $O(n^2)$.