Informatica Teorica

Sezione Pradella

Appello del 13 Settembre 2006

Attenzione

I voti proposti verranno pubblicati sul sito seguente:

http://www.elet.polimi.it/upload/pradella/IT.html

Verrà data precedenza ai **laureandi**, che devono indicare questa loro qualifica nel proprio elaborato.

Gli studenti avranno tempo due giorni (48 ore, per la precisione) dalla data della pubblicazione per rifiutare il voto proposto, se sufficiente. L'eventuale rifiuto deve essere comunicato via email, facendo uso dell'indirizzo ufficiale del Poliself, non di indirizzo privato! Trascorsi i due giorni, i voti verranno registrati e trasmessi alla segreteria.

Il docente spedirà a ogni studente "rinunciatario" un esplicito messaggio di "ricevuta". In caso di mancata ricezione di tale ricevuta si consiglia di contattare il docente telefonicamente o di avvisare la segreteria didattica del DEI.

Il tempo a disposizione per lo svolgimento del tema d'esame è 1h e 30 minuti.

Esercizio 1 (punti 12)

Si considerino le seguenti regole di attraversamento di un incrocio regolato da un semaforo.

- Il semaforo è verde per 4 unità di tempo; poi passa a giallo per un'unità di tempo e rosso per 4 ulteriori unità.
- Se un veicolo attraversa l'incrocio con semaforo verde o giallo nessuna azione viene eseguite nei suoi confronti.
- Se un veicolo attraversa l'incrocio con semaforo rosso entro 1 unità di tempo da quando è diventato rosso, viene elevata contravvenzione per Euro 50.
- Se un veicolo attraversa l'incrocio con semaforo rosso dopo 1 unità di tempo da quando è diventato rosso, viene elevata contravvenzione per Euro 200.
- Per semplicità si consideri un solo veicolo alla volta; si può anche assumere che la contravvenzione sia contemporanea all'infrazione.

Si formalizzino le suddette regole in ***una sola*** delle versioni proposte nel seguito.

Versione A

Si utilizzi un'opportuna macchina astratta assumendo un tempo discreto in cui l'unità corrisponde all'unità di tempo indicata sopra.

Versione B

Si utilizzino formule del primo ordine. In tal caso si può adottare un dominio temporale sia discreto che continuo.

NB

E' vietato consegnare entrambe le soluzioni che in tal caso non sarebbero valutate.

Esercizio 2 (punti 10)

Si consideri il sottoinsieme del linguaggio C in cui siano escluse tutte le istruzioni di controllo del flusso dell'esecuzione ad eccezione dell'istruzione condizionale **if** e dell'istruzione ciclica **for**.

Si dica, giustificando brevemente la risposta, se è decidibile o no il problema di stabilire se un generico programma scritto in tale sottoinsieme del C terminerà sempre la sua esecuzione (ossia per ogni valore dei dati di ingresso) o no.

Esercizio 3 (punti 10)

Si supponga di implementare un algoritmo di merge-sort mediante una macchina RAM (non è necessario scrivere effettivamente il codice!).

Quale sarebbe la sua complessità temporale valutata a criterio di costo costante e a criterio di costo logaritmico? Giustificare brevemente la risposta. Per semplicità si può assumere che i dati da ordinare occupino *singolarmente* una quantità limitata di memoria (ad esempio 32 bit).

Parte facoltativa (ulteriori punti 5)

Si valutino anche la complessità spaziale dell'implementazione mediante RAM e la complessità sia spaziale che temporale di una possibile implementazione mediante macchina di Turing, assumendo la stessa ipotesi semplificativa della parte precedente.

Soluzioni

Esercizio 1

Versione A (traccia per la costruzione di un automa)

Assumendo un dominio temporale discreto, conviene associare ogni scatto di transizione a un'unità di tempo. Quindi si può costruire un automa a 9 stati per descrivere l'evoluzione del semaforo. Ogni transizione è etichettata da un input attr ("il veicolo attraversa durante l'unità di tempo corrente") oppure n_attr ("il veicolo non attraversa durante l'unità di tempo corrente"). La transizione etichettata attr uscente dal primo stato rosso comporta l'uscita multa50 e le successive transizioni uscenti dagli stati rossi comportano l'uscita multa200.

Versione B

Si introducano i seguenti predicati:

- v(t), (g(t), r(t), s(t)) Il semaforo è verde (rispettivamente, giallo, rosso, spento) all'istante t
- attr(t) Il veicolo attraversa l'incrocio all'istante t
- multa50(t) Viene elevata contravvenzione per 50 euro all'istante t
- multa200(t) Viene elevata contravvenzione per 200 euro all'istante t

La formula seguente formalizza le regole richieste, assumendo che il semaforo inizi a funzionare all'istante 0 (con il verde) e che la multa venga notificata immediatamente.

$$\forall t$$
$$(t < 0 \rightarrow s(t)) \wedge (0 \bmod 9 \leq t < 4 \bmod 9 \rightarrow v(t)) \wedge$$
$$(4 \bmod 9 \leq t < 5 \bmod 9 \rightarrow g(t)) \wedge (5 \bmod 9 \leq t < 9 \bmod 9 \rightarrow v(t))$$
$$\wedge$$
$$((attr(t) \wedge 0 \bmod 9 \leq t < 4 \bmod 9) \rightarrow (\neg multa50(t) \wedge \neg multa200(t))$$
$$\wedge$$
$$((attr(t) \wedge 4 \bmod 9 \leq t < 5 \bmod 9) \rightarrow (\neg multa50(t) \wedge \neg multa200(t))$$
$$\wedge$$
$$((attr(t) \wedge 5 \bmod 9 \leq t < 6 \bmod 9) \rightarrow (multa50(t) \wedge \neg multa200(t))$$
$$\wedge$$
$$((attr(t) \wedge 6 \bmod 9 \leq t < 9 \bmod 9) \rightarrow (\neg multa50(t) \wedge multa200(t))$$

Commento

Si noti che le formule di cui sopra, semplici e sistematiche, in realtà non formalizzano *esattamente* le regole enunciate; bensì formalizzano un insieme di "comportamenti" che garantiscono la soddisfazione delle regole: da queste formule, usate come *assiomi*, è possibile *derivare come teorema*, ad esempio, il fatto che *Se un veicolo attraversa l'incrocio con semaforo rosso dopo 1 unità di tempo da quando è diventato rosso, viene elevata contravvenzione per Euro 200.*

Esercizio 2

In C, diversamente da altri linguaggi, come il Pascal, il ciclo **for** ha sufficiente generalità da permettere di simulare anche il ciclo **while**. E' noto che istruzione if e ciclo while (oppure l'istruzione if e la possibilità di usare la ricorsione nella chiamata di sottoprogrammi) permettono a un normale linguaggio di programmazione di avere la stessa potenza delle Macchine di Turing. Quindi il problema posto è equivalente a stabilire se una generica MT calcola una funzione totale o no. Questo problema è notoriamente indecidibile.

Esercizio 3

La normale analisi di complessità temporale dell'algoritmo di merge-sort assume implicitamente un criterio di costo costante. E' noto che essa produce un risultato $\Theta(n.\log(n))$. Assumendo il criterio di costo logaritmico occorre tener presente che ogni accesso a un singolo elemento costa un fattore log(n) (in questa valutazione interviene solo il costo dell'indirizzamento, poiché la memorizzazione dei singoli dati richiede un numero di bit limitato a priori). Quindi la complessità temporale valutata a criterio di costo logaritmico è $\Theta(n.\log^2(n))$.

Parte facoltativa

Per valutare la complessità spaziale dell'implementazione mediante RAM occorre tener presente che sono possibili diverse realizzazioni –ricorsive e non- dell'algoritmo di merge-sort. La più efficiente dal punto di vista della complessità spaziale fa uso alternato di due array (o file) e risulta quindi $\Theta(n)$ in entrambi i criteri.

Un'analoga implementazione mediante MT richiederebbe anch'essa una quantità di memoria $\Theta(n)$, mentre avrebbe una complessità temprale $\Theta(n.\log(n))$ grazie alla natura tipicamente sequenziale sia dell'algoritmo di merge-sort che della MT (risparmiando così il fattore log(n) per l'accesso al singolo dato).