Linguaggi di Programmazione

A.A. 2003/2004

Corso di Laurea Triennale in "Informatica e Comunicazione Digitale" – Sede di Taranto

Prova di esonero: 23/4/2004 ore 8.30 – 11.00 (fac simile)

1. Spiegare la classificazione delle grammatiche proposta da Noam Chomsky (*4 punti*)

Una grammatica G si dice

- di tipo 0 nel caso generale in cui le produzioni non sono soggette a vincoli;
- di tipo 1, o contestuale, se per ogni produzione $u \to v$ si ha $|u| \leq |v|$;
- di tipo 2, o acontestuale, se per ogni produzione ha la forma $A \to v$, con A simbolo ausiliario;
- di tipo 3, o regolare, se ogni produzione presenta la forma $A \to aB$ oppure $A \to a$, con A e B simboli ausiliari e a simbolo terminale.

La definizione di una grammatica di tipo 1 è equivalente alla richiesta che ogni produzione abbia la forma:

$w_1 A w_2 \to w_1 v w_2$

ove w_1, w_2, $v \in (N \cup V)^*$, $v \neq \varepsilon$, e $A \in N$, che si può leggere:

"*il simbolo* A *può essere sostituito con la parola* v *se si trova nel contesto* w_1- w_2", giustificando così il "contestuale" riferito alle grammatiche di tipo 1.

Per le grammatiche di tipo 2 le sostituzioni di simboli effettuate non dipendono mai dal contesto.

È evidente che ogni grammatica regolare è acontestuale, ogni grammatica acontestuale è contestuale, ogni grammatica contestuale è di tipo 0, ma non viceversa. (inserire disegno con diagrammi di Venn)

---

2. Scrivere nella notazione EBNF un frammento della grammatica che genera i costrutti **le espressioni booleane** corrette del Pascal e dimostrare che la frase **not a or b** è una espressione booleana (*6 punti*)

⟨espressione booleana⟩ ::= ⟨espressione booleana semplice⟩ |
⟨espressione booleana semplice⟩ ⟨operatore logico binario⟩ ⟨espressione booleana semplice⟩
⟨operatore logico binario⟩ ::= **or** | **and**
⟨espressione booleana semplice⟩ ::= ⟨termine booleano⟩ | **not** ⟨termine booleano⟩
⟨termine booleano⟩ ::= ⟨identificatore⟩ | ⟨relazione aritmetica⟩ | ⟨costante booleana⟩ |
(⟨espressione booleana⟩ **)**
⟨identificatore⟩ ::= ⟨lettera⟩ | ⟨lettera⟩ ⟨alfanumerico⟩
⟨alfanumerico⟩ ::= ⟨simbolo⟩ | ⟨simbolo⟩ ⟨alfanumerico⟩
⟨simbolo⟩ ::= ⟨lettera⟩ | ⟨cifra⟩
⟨lettera⟩ ::= a | b | … | z
⟨cifra⟩ ::= 0 | 1 | … | 9

Per dimostrare che è una espressione booleana, costruire l'albero sintattico o mostrare la generazione a partire dall'assioma ⟨espressione booleana⟩

---

3. Spiegare i concetti della semantica denotazionale assiomatica (*4 punti*).

In questo caso la semantica di un programma P di un linguaggio di programmazione *L*, che sull'input i produce l'output o, è descritta mediante una relazione $R_P(i,o)$ che lega i dati di ingresso ai dati di uscita. In generale si ricorre a due predicati per descrivere $R_P(i,o)$: se il primo (detto precondizione) è vero sui dati di ingresso e se il programma termina su quei dati, allora il secondo (detto postcondizione) è vero sui dati di uscita. Per definire la semantica di un programma occorre conoscere la semantica assiomatica per ciascuna istruzione del linguaggio in cui è scritto il programma. Essa definirà come le precondizioni e le postcondizioni siano correlate per ciascuna istruzione del linguaggio, dove le precondizioni descrivono la situazione della memoria prima dell'esecuzione dell'istruzione, le postcondizioni descrivono la nuova situazione dopo l'esecuzione di tale istruzione.

Questo viene ottenuto fornendo assiomi (onde il nome di metodo assiomatico) o regole di inferenza per ciascuna istruzione.

Alcuni esempi di assiomi sono:

{P(i)} *readln(n)* {P(n)} ← assioma!!

{P} *writeln(n)* {P} ← assioma!!

{P(exp)} *n := exp* {P(n)} ← assioma!!

In questi assiomi *i* è il valore fornito in input. I predicati che indicano la precondizione e la postcondizione sono stati racchiusi tra parentesi graffe e scritti rispettivamente a sinistra e a destra della istruzione; la precondizione è stata indicata con P, la postcondizione è sempre P eventualmente modificato con una sostituzione.

---

4. Scrivere in Pascal l'algoritmo dell'ordinamento per selezione di un array e verificarne la correttezza (parziale e terminazione) (*10 punti*)

DESCRIZIONE DELL'ALGORITMO

Input: array a[1..n] di n elementi

- Mentre ci sono ancora elementi nella parte di a non ordinata esegui:
  a) trova il minimo min e la sua posizione p
  b) scambia il minimo min con il primo elemento della parte di a non ordinata

Output: array a[1..n] ordinato

IMPLEMENTAZIONE IN PASCAL

```
procedure selectsort(var a: nelement; n: integer);
var i {primo elem. nella parte di array non ordinata},
        j {indice x la parte non ordinata}, p {posiz. del min},
        min {minimo corrente}: integer;
begin
        for i := 1 to n-1 do
        begin
                min := a[i]; p := i;
                for j := i+1 to n do
                        if a[j]<min then
                                begin
                                        min :=a[j]; p := j;
                                end;
                a[p] := a[i]; a[i] := min
        end
end
```

Le invarianti cicliche sono

- Ciclo più esterno: $\{\forall k\in[1,i-1]\ a[k]\le a[k+1],\ \forall j\in[i+1,n]\ \ a[i]\le a[j]\}$
- Ciclo più interno: $\{\forall k\in[i,j]\ \min\le a[k],\ i\le n-1,\ i\le j\le n,\ i\le p\le i,\ \min=a[]\}$

- Il programma Pascal termina per via dei cicli ‘for’.

5. Data in input una lista di valori numerici memorizzata su un file, si provveda a scrivere un programma C in grado di acquisire e ordinare, per mezzo dell’algoritmo bubble-sort, tale lista. Tale programma dovrà inoltre stampare la lista ordinata su video o su file in base alle preferenze dell’utente. (*13 punti*) [1]

   Il file di input deve essere conforme al formato definito dal seguente esempio:

   384

   35647

   47783

   27

   8439

   8439

[1] La totalizzazione di un punteggio superiore a 30 punti equivale al *30 con lode*.