8 aprile 2008

Linguaggi di Programmazione

Nome e Cognome	
Corso di laurea	
Telefono	
Email	

1. Specificare la grammatica EBNF di un linguaggio per la specifica di un programma basato sul paradigma funzionale. Un programma si compone di tre sezioni, di cui le prime due sono opzionali, mentre la terza è obbligatoria. Nella prima sezione vengono dichiarate le variabili mediante un tipo (intero o booleano) ed una costante corrispondente. Nella seconda sezione vengono definite le funzioni in termini di protocollo (lista dei parametri formali) e corpo (espressione). La terza sezione specifica l'espressione del programma. Ogni espressione può coinvolgere operatori aritmetici (**+**, **-**, *****, **/**) o logici (**and**, **or**), nomi di variabili e/o parametri formali, costanti (intere o booleane) e chiamate di funzione (nome della funzione seguita da una lista di espressioni, i parametri attuali). Ecco un esempio di programma:

```
int a = 3, b = 5;
bool c = true, d = false;

function alfa(x) = (x and c) or d;
function beta(x, y) = ((a + x) * (b – y)) / 2;

(beta(a, b+1) + beta(b, a-b)) – 10;
```

2. È data la seguente tabella di operatori relativi and un linguaggio **L**, per la quale si assume priorità decrescente dall'alto verso il basso,

Operatore	*Associatività*	*Ordine valutazione*
*****, **and**	sinistra	da sinistra a destra
+, **or**	destra	da destra a sinistra

e la seguente espressione:

```
x := a + b * c and d or f + g and h
```

Si assume che il linguaggio **L** manipoli unicamente tipi interi. In particolare, le operazioni booleane **and** ed **or** su tali interi sono definite dalle seguenti regole:

- Il risultato di un **and** è 1 se e solo se entrambi gli operandi sono diversi da 0, altrimenti il risultato è 0.
- Il risultato di un **or** è 0 se e solo se entrambi gli operandi sono uguali a 0, altrimenti il risultato è 1.

Si assume inoltre che gli operatori **and**, **or**, **+** e ***** siano valutati in corto circuito. In particolare, ecco le regole di corto circuito per i due operatori aritmetici:

- Se l'operando valutato per primo nella la somma (**+**) è 0, allora il risultato coincide con il secondo operando.
- Se l'operando valutato per primo nel prodotto (*****) è 1, allora il risultato coincide con il secondo operando.

Si chiede di:

a) Rappresentare l'albero della espressione.

b) Specificare la semantica operazionale della espressione.

NB: Il linguaggio di specifica operazionale è così caratterizzato:

- Contiene gli operatori aritmetici (**+**, **-**, *****, **/**), uguaglianza (**==**), disuguaglianza (**!=**) ed assegnamento (**=**);
- Non contiene gli operatori **and**, **or**;
- Ogni operatore non può essere applicato ad espressioni, ma solo a variabili o costanti;
- Contiene le istruzioni condizionali (*if-then-endif* ed *if-then-else-endif*) i cui predicati possono essere solo confronti tra una variabile ed una costante.

3. Specificare nel linguaggio *Scheme* la funzione binaria **indexing** che, ricevendo in ingresso un array associativo **A** (rappresentato da una lista di coppie chiave-valore) ed una chiave di accesso **key**, computa il valore corrispondente alla chiave **key** nell'array associativo **A**. Ogni chiave di accesso è una stringa di caratteri. Nel caso di chiave inesistente, la funzione restituisce l'atomo **error**. Ecco alcuni esempi:

A	**key**	**(indexing A key)**
((alfa 3)(beta 4)(gamma 5))	beta	4
((x (1 2))(y (3 4))(z (5 6)) (w ()))	y	(3 4)
((alfa 3)(beta 4)(gamma 5))	delta	error

4. Sono date le seguenti due strutture tabellari:

- **Cliente(nomecliente, indirizzo, cittacliente)**
- **Filiale(nomefiliale, cittafiliale, deposito)**

Si chiede di specificare nel linguaggio *Haskell* protocollo e corpo della funzione **stessacitta** che, ricevendo in ingresso una lista di clienti ed una lista di filiali, computa la lista delle coppie (**nomecliente, nomefiliale**) per le quali il cliente **nomecliente** vive nella stessa città in cui si trova la filiale **nomefiliale**.

5. E' data una base di fatti *Prolog* relativa alla specifica di una grammatica BNF, in termini di assioma, simboli terminali, simboli nonterminali e produzioni, come nel seguente esempio:

$S \rightarrow \mathbf{a}\ A \mid \mathbf{b}$
$A \rightarrow \mathbf{c}\ S \mid \mathbf{d}$

⇨

```
assioma('S').
terminali([a,b,c,d]).
nonterminali(['S','A']).
produzione('S',[a,'A']).
produzione('S',[b]).
produzione('A',[c,'S']).
produzione('A',[d]).
```

Assumendo che la parte destra ogni produzione BNF inizi sempre con un terminale (come nell'esempio), si chiede di specificare i seguenti predicati:

- **deriva(Simboli,Frase)**, vero qualora sia possibile derivare la frase **Frase** dalla lista **Simboli** di simboli grammaticali.
- **corretta(Frase)**, vero qualora **Frase** sia una frase del linguaggio specificato dalla BNF.

Ad esempio, **deriva([a,'A'],[a,c,b])** risulta vero (in quanto è possibile derivare la frase mediante i seguenti passi di derivazione: **a** $A \Rightarrow$ **a c** $S \Rightarrow$ **a c b**). Invece, **deriva(['S'],[a,b])** risulta falso.

6. Illustrare (con l'ausilio di semplici esempi) le differenze sostanziali che sussistono tra la programmazione funzionale e la programmazione logica.