7 dicembre 2004

Linguaggi di Programmazione

Nome e Cognome	
Matricola	
Anno di corso	
Telefono	

1. Specificare la grammatica BNF di un linguaggio ad eventi in cui ogni programma è costituito da una lista (non vuota) di trigger come nel seguente esempio:

```
define trigger propaga
  event alfa, beta, gamma
  condition alfa = beta, gamma > alfa
  action f1(x, y, z), f2(n), f3(m, alfa)
end propaga.

define trigger controlla
  event delta, epsilon
  condition delta != epsilon
  action g(a, b)
end controlla.
```

Ogni trigger ha un nome ed è definito in termini di regole ECA (*event-condition-action*), in cui le clausole *event* ed *action* sono obbligatorie, mentre la clausola *condition* è opzionale. La clausola *event* specifica una lista (non vuota) di eventi espressi come identificatori. La clusola *condition* specifica uno o più confronti semplici tra due eventi. Possibili operatori di confronto sono **=**, **!=**, **>**, **<**, **>=**, **<=**. La clausola *action* specifica una sequenza (non vuota) di chiamate di funzioni, in cui ogni funzione è applicata ad una lista (non vuota) di argomenti espressi come identificatori. La specifica del trigger si chiude con il nome del trigger definito nell'intestazione.

2. È data la seguente tabella di operatori (con priorità decrescente dall'alto verso il basso e valutazione delle espressioni logiche in corto circuito):

Operatori	*Associatività*	*Ordine valutazione operandi*
+, **–**	sinistra	da sinistra a destra
*****	destra	da sinistra a destra
<, **>**	nonassoc	da destra a sinistra
and, **or**	sinistra	da destra a sinistra

e la seguente istruzione di assegnamento:

```
x := a + b * c – d * e > f + (g * h) – i * m and v > w or c < d
```

a) Rappresentare l'albero della espressione di assegnamento.
b) Specificare la semantica operazionale dell'istruzione di assegnamento.

NB: Il linguaggio di specifica operazionale contempla le seguenti limitazioni:

- Non contiene gli operatori logici **and**, **or**;
- Contiene gli operatori **+**, **-**, *****, **<**, **>** che però non possono essere applicati al risultato di altre operazioni (necessità di variabili temporanee);
- Contiene le istruzioni condizionali **if** *cond* **then** … **endif** e **if** *cond* **then** … **else** … **endif**, in cui *cond* può essere o un temporaneo (ad esempio, **if** `t1` **then** …) oppure la negazione di un temporaneo (ad esempio, **if** `!t1` **then** …).

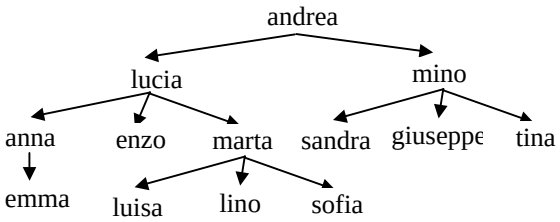
3. Definire nel linguaggio *Haskell* le seguenti strutture tabellari:

```
Libri(Titolo, Autore, Edizione)
Corsi(Nome, Docente, Libro)
Studenti(Cognome, Corso)
```

Quindi, codificare la funzione `studenti_autori` che, ricevendo in ingresso le tabelle `libri`, `corsi` e `studenti`, computa la lista degli studenti che sono autori di testi adottati nei loro corsi.

4. Data una base di fatti *Prolog* relativa alla relazione parentale tra un insieme di persone, come nel seguente esempio:

```
genitore(andrea, [lucia, mino]).
genitore(lucia, [anna, enzo, marta]).
genitore(mino, [sandra, giuseppe, tina]).
genitore(anna, [emma]).
genitore(marta, [luisa, lino, sofia]).
...
```



definire le regole per i seguenti predicati:

a) `nonno(X, Y)`
in cui `X` è nonno (o nonna) di `Y`;

b) `antenato(X, Y)`
in cui `X` è un antenato di `Y`.

5. Illustrare le due varianti del costrutto **`let`** di fattorizzazione delle espressioni in *Scheme*.

6. Definire e giustificare le regole di co-varianza e contro-varianza nel meccanismo di ereditarietà dei linguaggi orientati agli oggetti.