11 Luglio 2014

Linguaggi di Programmazione

Cognome e nome	
Matricola	

1. Dato un alfabeto $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$, si chiede di specificare l'espressione regolare delle stringhe che contengono esattamente tre **b** separati fra loro da altri caratteri.

2. Specificare la grammatica BNF di un linguaggio in cui ogni frase è un programma specificato da un linguaggio funzionale per la manipolazione di booleani, come nel seguente esempio:

```
const a=true, b=false, c=true;

func alfa() = a and not (b or c),
     beta(x, y) = (alfa() or x) and y or false,
     gamma(z) = beta(z, a or false) or not (a and b) or z;

body (a or b and true) and alfa() or not gamma(beta(a,b)).
```

Il programma è composto da tre sezioni, di cui solo la terza è obbligatoria. La prima sezione (introdotta dalla keyword **`const`**) specifica un insieme di costanti. La seconda sezione (introdotta dalla keyword **`func`**) specifica una serie di funzioni, definite da una lista (anche vuota) di parametri formali ed una espressione (corpo della funzione). La terza sezione (introdotta dalla keyword **`body`**) specifica l'espressione del programma. Una espressione coinvolge le operazioni logiche **`and`**, **`or`**, **`not`** (con possibilità di parentesi) e chiamate di funzione.

3. È data la seguente tabella degli operatori (con precedenza decrescente verso il basso):

Operatori	*Associatività*	*Ordine di valutazione*
`+`, **`-`**	sinistra	Da sinistra a destra
`*`, **`/`**	sinistra	Da destra a sinistra
`=`	non associativo	Da sinistra a destra
`or`	destra	Da sinistra a sinistra
`and`	destra	Da destra a sinistra

e la seguente espressione: `x + y - z * v / w = (x + y) * z + v and v = w or x = y`

Assumendo che solo l'operatore **or** sia valutato in corto circuito, si chiede di:

a. Inserire le parentesi nella espressione, indicando così l'associazione degli operandi agli operatori;

b. Rappresentare l'albero della espressione;

c. Specificare la semantica operazionale della valutazione della espressione sulla base dei seguenti vincoli:

- Il linguaggio di specifica include tutti gli operatori indicati nella tabella, l'assegnamento (**`:=`**), la negazione logica (**`not`**), le costanti booleane, la selezione a più vie ed il **return**,
- Ogni operatore (ad eccezione dell'assegnamento) non può essere applicato al risultato di altre operazioni.

4. È dato il seguente frammento di grammatica BNF, relativo alla specifica del ciclo a condizione finale in un linguaggio imperativo:

do-stat → **do** *stat* **while** *expr*
expr → **not** *expr* | **id**
stat → *assign-stat* | *do-stat*

Specificare la semantica denotazionale del corrispondente frammento di linguaggio, assumendo che **id** rappresenti il nome di una variabile logica, il linguaggio di specifica non disponga di operatori logici, siano disponibili le funzioni ausiliarie M_{id}(**id**, `s`), che restituisce il valore di **id** allo stato `s` (eventualmente **errore**), e M_{assign}(*assign-stat*, `s`). In particolare, specificare la funzione semantica M_{stat}(*stat*, `s`).

5. Specificare nel linguaggio *Scheme* la funzione `sumpairs`, avente in ingresso una lista (anche vuota) di numeri, la quale computa la lista delle somme delle coppie, come nei seguenti esempi (se disaccoppiato, l'ultimo numero viene trascritto nel risultato):

```
(sumpairs '()) = ()
(sumpairs '(1)) = (1)
(sumpairs '(1 2 5) = (3 5)
(sumpairs '(1 2 3 4 8 12) = (3 7 20)
```

6. Dopo aver illustrato il significato della forma funzionale `foldr` in *Haskell*, sulla base della seguente definizione,

```
computa :: [[a]] -> [a]
computa = foldr (\x y -> init (x++y)) []
```

indicare sia l'espressione computata dalla seguente applicazione che il corrispondente risultato:

```
computa ["alfa", "beta", "gamma"]
```

7. Specificare in *Prolog* il predicato `sumpairs(L,S)`, in cui `L` è una lista (anche vuota) di numeri, mentre `S` è la lista di somme di coppie, come specificato al punto **5**.

8. Illustrare i concetti di *scope* e *ambiente di referenziazione*, sia nel caso statico che in quello dinamico.