Esame di Linguaggi di Programmazione - 7 febbraio 97

Ricordare di indicare nome e cognome sul compito.

Esercizio 1 (Punti 10)

Dire se le seguenti espressioni ML-like sono tipabili (in ML) e, in caso affermativo, derivarne il tipo più generale.

- $\lambda x.\lambda y.((xy)y)$
- $\lambda x.\lambda y.(x(yy))$
- $\lambda x.\lambda y.(x,(y,y))$
- $\lambda x.\lambda y.[x,(y,y)]$
- $\lambda x.\lambda y.[x,[y],y]$

Nota: La regola ML per derivare il tipo di un'espressione della forma $[e_1, e_2, \ldots, e_n]$ (lista i cui elementi sono $e_1, e_2, \ldots, e_n$) è:

$$\frac{r \vdash e_1 : \alpha \quad r \vdash e_2 : \alpha \ldots r \vdash e_n : \alpha}{r \vdash [e_1, e_2, \ldots, e_n] : \alpha\ list}$$

Esercizio 2 (Punti 6)

Per ognuna delle seguenti espressioni di tipo, trovare, se esiste, un'espressione dell' ML puro che abbia tale tipo, e dire se è il suo tipo più generale.

- $\alpha \to \beta \to \alpha \to \beta$
- $\alpha \to \beta \to \gamma$
- $\alpha \to \beta \to (\alpha \times \alpha)$

Esercizio 3 (Punti 4)

Scrivere in ML una funzione *curry* che, data una funzione $f : (\alpha \times \beta) \to \gamma$, la trasforma nella sua forma curried $f' : \alpha \to \beta \to \gamma$, definita da $f'xy = f(x,y)$. Scrivere inoltre la sua inversa *uncurry*, che trasforma f' in f. Ricapitolando, deve valere: *curry* $f = f'$ e *uncurry* $f' = f$.

Esercizio 4 (Punti 10)

Si consideri il tipo di dato α *list list ... list* dove "*list*" è ripetuto n volte, con n costante intera positiva. Definire, in uno dei linguaggi dichiarativi visti a lezione (ML, Prolog, o Lambda Prolog), una funzione (o un predicato) che effettui il "flattening" di una tale lista, trasformandola nella lista di tutti gli elementi (di tipo α) in essa contenuti, nell'ordine. Ad esmpio, per $n = 3$, se la lista di partenza è $[[[a,b][c]],[[d][e]]]$, la lista che deve essere restituita è $[a,b,c,d,e]$. Discutere se sia possibile o meno definire una tale funzione in ciascuno dei tre linguaggi sopra citati, e, in caso affermativo, se siano tutti ugualmente convenienti. E nel caso che n non sia una costante prefissata, ma bensí un dato del problema?

Esercizio 5 (Punti 8)

Si vuole definire in C^{++}like un tipo di dato "lista bidirezionale" (non circolare). La classe di base *node* avrà campi *info* (informazione, supponiamo di tipo intero), *next* (puntatore all'avanti) e *pred* (puntatore all'indietro). I metodi da definire sono quelli standard delle liste bidirezionali, cioè:

- un costruttore con parametro n che setta *info* a n e i puntatori a *nil*
- un metodo con parametro n che aggiunge in testa un nuovo elemento con *info* settata a n, ed un analogo metodo per aggiungere in coda
- un metodo senza parametri che restituisce la prima informazione (la *info* dell'elemento di testa), ed un analogo metodo che restituisce l'ultima informazione (la *info* dell'elemento di coda)
- un metodo senza parametri che restituisce la lista senza l'elemento di testa, ed un analogo metodo che elimina l'elemento di coda.

Trascurare eventuali problemi di condivisione. Si possono definire altre classi (eventualmente derivate).