MFI 2002/2003 - Compitino n. 3

Cognome e Nome:

Si esibisca una λ-espressione $\mathcal{E}$ che contenga la variabile libera y e renda i seguenti termini α/β- equivalenti, illustrando i passaggi di riduzione che provano che la $\mathcal{E}$ è corretta (punti 5).

$$\begin{array}{rcl} t_1 & = & (\ \lambda\, x\, .\, (\ \lambda\, y\, .\, (\ x\ y\ )\ x\ )\ \mathcal{E}\ ) \\ t_2 & = & (\ \lambda\, z\, .\, (\ \mathcal{E}\ \lambda\, x\, .\, (\ x\ y\ )\ )\ (\ \mathcal{E}\ \lambda\, x\, .\, (\ x\ y\ )\ )\ ) \end{array}$$

Si consideri la seguente grammatica $\mathcal{GF}$:

`Prog`	::=	**with** `Decs` **eval** `Exp`.
`Dec`	::=	`Id` = `Exp`;.
`Exp`	::=	**0** | **1** | **pack** | **select** | **sumR** | **sumC** | **ITE** | π_1(`Exp`) | π_2(`Exp`) | `Exp`(`Exps`) | λ `Id` : `Type`.`Exp`.
`Exps`	::=	`Exp` | `Exp`, `Exps`.
`Decs`	::=	`Dec` | `Dec`, `Decs`.
`Type`	::=	**bit** | **word** | (`Type` $\times$ `Type`) | (`Type` $\to$ `Type`).

Intuitivamente, questo linguaggio serve per descrivere elementi di tipo bit e word (=mezzo byte) e **funzioni su di essi**. In particolare, le constanti **pack**, **select**, **sumR**, **sumC** e **ITE** sono funzioni corrispondenti (rispettivamente) ad "impacchettare" 4 bit per ottenere un word, selezionare il bit in una certa posizione all'interno di un word, ottenere il risultato della somma di 3 bit, ottenere il riporto della somma di 3 bit e eseguire una scelta condizionale a seconda se i primi due argomenti siano o no uguali (ulteriori dettagli nel seguito).

1. (punti 1) Dare i tipi delle funzioni di semantica statica, definendo eventuali insiemi e funzioni ausiliari che si usino in essi.

2. (punti 2) Dare la semantica dinamica per il non terminale `Type`.

3. (punti 2) Dare i tipi delle funzioni di semantica dinamica, definendo eventuali insiemi e funzioni ausiliari che si usino in essi.

4. Dare la/e regola/e della semantica statica e dinamica per il non terminale `Exps` relative alle produzioni

   `Exp` ::= **0** e `Exp` ::= **1** corrispondenti alle costanti di tipo bit (punti 1)

   `Exp` ::= **pack** dove **pack** è una funzione che trasforma 4 bit nel word da essi composto (punti 2).

   `Exp` ::= **select** dove **select** è una funzione che seleziona all'interno del suo primo argomento (che è di tipo word) il bit che si trova nella posizione indicata dagli altri due argomenti (che sono di tipo bit). (punti 2).

   `Exp` ::= **sumR** dove **sumR** è la funzione che produce il bit risultato della somma di 3 bit (punti 2).

   `Exp` ::= π_1(`Exp`) corrispondente alla proiezione sulla prima componente di un elemento di un tipo prodotto (punti 2).

   `Exp` ::= λ `Id` : `Type`.`Exp` corrispondente alla definizione di una funzione anonima (stile lambda-calcolo) con un solo parametro del tipo dichiarato che può essere usato nel corpo della funzione (punti 4).

   `Exp` ::= `Exp(Exps)` corrispondente alla applicazione di una funzione ai suoi argomenti (punti 3).