PROGRAMMAZIONE II – A.A. 2018-19: Secondo Progetto

Sessione Estiva-Autunnale

Il progetto ha l'obiettivo di applicare a casi specifici i concetti e le tecniche di programmazione esaminate durante la seconda parte del corso, e consiste nella progettazione e realizzazione di alcuni moduli software.

Descrizione: Progettazione e sviluppo di un interprete in OCaml

Si consideri un'estensione del linguaggio didattico funzionale presentato a lezione che permetta di manipolare alberi binari di espressioni. L'estensione minimale dei tipi è riportata di seguito

```
type exp = ... | ETree of tree              (* gli alberi sono anche espressioni *)
 |ApplyOver of (ide list) exp * exp         (* applicazione di funzione ai nodi *)
 |Select of ide * exp                       (* selezione di un nodo *)
and tree = Empty                            (* albero vuoto *)
 | Node of ide * exp * tree * tree          (* albero binario *)
```

Ogni nodo di un albero, oltre ai figli, ha associato un identificatore (tag) e un'espressione. Quando un albero è definito, le espressioni dei nodi devono essere valutate, e solo quelle. I tag servono a caratterizzare (in maniera univoca) i nodi dell 'albero.

"Select restituisce l'albero associato al tag indicato" => tree è un tipo di dato esprimibile. Ovvero può essere il risultato della valutazione di una espressione.

Ad esempio, l'espressione

```
ETree(Node("a", EInt 1, Node("b", EInt 2, Empty, Empty),
                        Node("c", EInt 2, Empty, Empty))
```

denota un albero con un nodo radice (che ha associati il tag "a" e il valore Int 1) e due foglie che hanno associato il tag "b" e il valore Int 2 e il tag "c" e il valore Int 2, rispettivamente.

Operazioni

L'operazione `Select` restituisce l'albero associato al tag indicato, oppure l'albero vuoto se nessun nodo corrisponde al tag. Invece `ApplyOver(lst, exf, ext)` applica la funzione denotata dal secondo parametro `exf` al valore associato ai nodi dell'albero denotato dal secondo parametro `ext`, aggiornandolo di conseguenza. Vengono aggiornati solo i nodi che hanno tag nella lista lst.

Si estenda l'interprete OCaML del linguaggio funzionale assumendo la regola di scoping statico.

1. Si verifichi la correttezza dell'interprete progettando ed eseguendo una quantità di casi di test sufficiente a testare tutti gli operatori aggiuntivi.

La sintassi astratta suggerita può essere modificata e, se ritenuto necessario, estesa.

Modalità di consegna

- Il progetto deve essere svolto e discusso col docente individualmente. Il confronto con colleghi mirante a valutare soluzioni alternative durante la fase di progetto è incoraggiato.
- Il progetto deve essere costituito da
  - i file sorgente contenenti il codice sviluppato e le corrispondenti batterie di test, ove tutto il codice deve essere adeguatamente commentato;

    - una relazione di massimo una pagina che descrive le principali scelte progettuali ed eventuali istruzioni per eseguire il codice.
- La consegna va fatta inviando per email tutti i file in un archivio. Per il corso A, inviare l'email al Prof. Ferrari con oggetto la stringa "[PR2A] Consegna progetto 2". Per il corso B, inviare l'email al Prof. Levi con oggetto la stringa "[PR2B] Consegna progetto 2". Il progetto deve essere consegnato entro il 21 Giugno 2019 (per il primo appello estivo) ed il 11 Luglio 2019 (per il secondo appello estivo) mentre entro il 25 Agosto 2019 per la sessione autunnale.
-

Altre informazioni

- Per quanto riguarda il progetto, i docenti risponderanno solo a eventuali domande riguardanti l'interpretazione del testo, e non commenteranno soluzioni parziali prima della consegna.