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

[consegna entro il 14 gennaio 2019]

Il progetto ha l'obiettivo di applicare a casi specifici i concetti e le tecniche di implementazione dei linguaggi di programmazione esaminate durante la seconda parte del corso. Il progetto consiste nella progettazione e realizzazione di un interprete per un semplice linguaggio di programmazione.

Descrizione: Progettazione e sviluppo di un interprete in OCaml

Si consideri un'estensione del linguaggio didattico funzionale presentato a lezione che permetta di manipolare come dati primitivi dizionari di elementi, ovvero una collezione di coppie (chiave, valore). Assumiamo che la chiave sia un identificatore.

Introduciamo brevemente la sintassi concreta per operare con dizionari.

```
/* creazione di un dizionario vuoto */
my_dict = {}
:- my_dict {}
/* creazione di un dizionario con valori */
my_dict = {'name': 'Giovanni', 'matricola': 123456}
:- my_dict = {'name': 'Giovanni', 'matricola': 123456}
/* accedere a un elemento di un dizionario */
my_dict['name']
:- 'Giovanni'
my_dict.get('matricola')
:- 123456
/* operazioni (dizionari sono immutable) */
my_dict1 = my_dict['età'] = 22
-: my_dict1{'età': 22,'name': 'Giovanni', 'matricola': 123456}
my_dict2 = rm(my_dict1, 'name')
-: my_dict2{'età': 22,'matricola': 123456}
my_dict3 = clear(my_dict2)
-: my_dict3{}
ApplyOver((fun x -> x+1), my_dict2)
:- {'età': 23,'matricola': 123457}
```

Il significato di `ApplyOver(f, dict)` diventa ovvio: si tratta di applicare la funzione denotata dal primo parametro `f` al valore associato a ogni elemento del dizionario denotato dal secondo parametro `dict`, creando di conseguenza un nuovo dizionario.

Parte obbligatoria:

1. Si definisca la sintassi concreta del linguaggio e la sintassi astratta tramite una opportuna definizione di tipo in OCaML.
2. Si definisca l'interprete OCaMl del linguaggio funzionale assumendo la regola di scoping statico.
3. Si verifichi la correttezza dell'interprete progettando ed eseguendo una quantità di casi di test sufficiente a testare tutti gli operatori aggiuntivi.

La sintassi concreta suggerita (e conseguentemente quella astratta) può essere modificata e, se ritenuto necessario, estesa.

Parte opzionale:

1. Estendere il linguaggio con un nuovo operatore rt_eval(exp) che prende in ingresso una espressione e restituisce il valore ottenuto eseguendo exp con la regola di scoping dinamico.
2. Si verifichi la correttezza dell'interprete progettando ed eseguendo una quantità di casi di test sufficiente a testare tutti gli operatori aggiuntivi.

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 casi 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 entro il 14 Gennaio 2018. Per il corso A, inviare l'email al Prof. Ferrari con oggetto "[PR2A] Consegna progetto 2". Per il corso B, inviare l'email alla Prof.ssa Levi con oggetto contenente la stringa "[PR2B] Consegna progetto 2".

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.