UNIVERSITÀ DI PISA

Dipartimento di Informatica
Corso di Laurea in Informatica

Relazione progetto OCaml

Programmazione II

Prof. Gianluigi Ferrari

Giacomo Trapani
600124 - Corso A

Anno Accademico 2020/2021

Dettagli implementativi

Il progetto consiste nella progettazione e realizzazione di una estensione del linguaggio didattico funzionale presentato a lezione che permetta di manipolare insiemi. Un insieme è una collezione di valori, non ordinati, che non contiene valori duplicati.

Creare un insieme.

Si riporta l'invariante di rappresentazione sugli insiemi (definiti come (**string** str, **evT list** elems)):

$$(\forall evT\ e \in elems.\ typecheck(str, e)\ \wedge\ (\exists! evT\ i \in elems \mid i \equiv e)) \wedge$$
$$str \in \{"int", "bool", "string"\}\ \wedge\ (\#elems = 1 \implies elems = [Unbound]).$$

Si ricorda che due funzioni f e g sono uguali se e solo se $dom(f) = dom(g)\ \wedge\ (\forall x \in dom(f).f(x) = g(x))$, operazione computazionalmente costosa; per questo motivo si sceglie - nonostante si lavori con un linguaggio funzionale - di non permettere l'esistenza di insiemi di funzioni, che non avrebbero comunque trovato alcuna applicazione tra i metodi da implementare all'interno del progetto. Nel caso in cui la lista *elems* coincida con la lista vuota, si inserisce il valore iniziale **evT** *Unbound*. Si implementano i seguenti operatori:

- **Empty**(**string** type), che costruisce un insieme vuoto di tipo *type* nel quale inserisce il valore *Unbound*;
- **Of**(**string** type, **exp list** elems), che costruisce un insieme di tipo *type* e lo popola con i valori esprimibili risultanti dalla valutazione degli elementi di *elems* a patto che siano (tra loro) distinti e di tipo *type*;
- **Singleton**(**exp** elem, **string** type), che costruisce un insieme di tipo *type* contenente soltanto l'elemento risultante dalla valutazione di *elem*.

Operazioni su insiemi.

Si implementano le seguenti operazioni su insiemi:

- **Add**(**exp** aSet, **exp** aElem), che restituisce il **Set** risultante dalla valutazione di *aSet* a cui viene aggiunto l'elemento (risultante dalla valutazione di) *aElem* mantenendo l'invariante su insiemi;
- **Remove**(**exp** aSet, **exp** aElem), che restituisce il **Set** risultante dalla valutazione di *aSet* a cui viene rimosso l'elemento (risultante dalla valutazione di) *aElem* mantenendo l'invariante su insiemi;
- **IsEmpty**(**exp** aSet), che restituisce **Bool(true)** se e solo se il Set risultante dalla valutazione di *aSet* è formato dal solo elemento di valore *Unbound*, **Bool(false)** altrimenti;
- **BelongsTo**(**exp** aSet, **exp** aElem), che restituisce **Bool(true)** se e solo vale la relazione di appartenenza tra i risultati della valutazione di *aSet* (di tipo **Set**) e *aElem*;
- **IsSubset**(**exp** aSet1, **exp** aSet2), che restituisce **Bool(true)** se e solo se vale la relazione di inclusione tra i risultati della valutazione di *aSet1* e *aSet2* (entrambi di tipo **Set**);
- **Maximum**(**exp** aSet), che restituisce il valore massimo nel Set risultante dalla valutazione di *aSet*;
- **Minimum**(**exp** aSet), che restituisce il valore minimo nel Set risultante dalla valutazione di *aSet*;
- **Union**(**exp**, **exp**), che restituisce il **Set** risultante dall'unione dei due insiemi risultanti dalla valutazione di *aSet1* e di *aSet2* formati da elementi dello stesso tipo;
- **Intersection**(**exp**, **exp**), che restituisce il **Set** risultante dall'intersezione dei due insiemi risultanti dalla valutazione di *aSet1* e di *aSet2* formati da elementi dello stesso tipo;

- **Difference**(**exp**, **exp**), che restituisce il **Set** risultante dalla differenza dei due insiemi risultanti dalla valutazione di *aSet1* e di *aSet2*.

Per ulteriori dettagli sulle implementazioni delle operazioni sopracitate si faccia riferimento alla semantica operazionale definita all'interno del file *RegoleOperazionali.pdf*.

Operatori funzionali.

Si implementano i seguenti operatori funzionali:

- **For_all**(**exp** predicate, **exp** aSet), che verifica che la proprietà risultante dalla valutazione di *predicate* valga su ogni elemento dell'insieme risultante dalla valutazione di *aSet*;
- **Exists**(**exp** predicate, **exp** aSet), che verifica che la proprietà risultante dalla valutazione di *predicate* valga su almeno un elemento dell'insieme risultante dalla valutazione di *aSet*;
- **Filter**(**exp** predicate, **exp** aSet), che restituisce il **Set** formato da tutti gli elementi dell'insieme risultante dalla valutazione di *aSet* su cui vale il risultato della valutazione di *predicate*;
- **Map**(**exp** func, **exp** aSet), che restituisce il **Set** formato dai risultati della valutazione di *func* su ogni elemento di *aSet*.

Nel caso in cui gli operatori funzionali vengano applicati su insiemi vuoti si sceglie di non implementare meccanismi di typechecking statico; lavorando con dei meccanismi di typechecking dinamico (scegliendo dunque un approccio à la **JavaScript**) il parametro funzionale necessita di un "valore" su cui essere applicato, cosa che l'insieme vuoto non fornisce in alcun modo. Per questa ragione:

- **For_all** su insieme vuoto restituisce sempre e solo il valore booleano true;
- **Exists** su insieme vuoto restituisce sempre e solo il valore booleano false;
- **Filter** su insieme vuoto restituisce l'insieme vuoto;
- **Map** su insieme vuoto restituisce un errore poiché non si può conoscere il tipo restituito dalla funzione che avrebbe chiamato - nozione fondamentale per la creazione di un insieme.

Metodo di sviluppo.

Il progetto è stato sviluppato con il supporto del text editor *VSCode* senza tuttavia l'utilizzo di strumenti esterni; è stato testato in un ambiente *REPL* e - nello specifico - la batteria di test all'interno del file *tests.ml* necessita di essere riportato linea per linea dopo in un ambiente in cui si è già valutato tutto ciò che è contenuto nel file *eval.ml*. Entrambi i file contengono - tra i commenti - quello che ci si aspetta sia l'output del REPL in modo che si possa facilmente confrontare con l'output effettivo.