Relazione progetto Ocaml Programmazione II

Anno Accademico 2020/2021

Studente Luca Rizzo

Matricola 598992

Introduzione

In questo progetto si richiedeva di estendere il linguaggio funzionale visto a lezione con il tipo di dato Set e alcune operazioni di base.

Implementazione del tipo di dato Set

Come prima cosa ho esteso l'albero di sintassi astratta tramite i costruttori EmptySet e Singleton per permettere la definizione del nuovo tipo di dato.

…

| Empty of tval

| Singleton of tval * exp

…

Successivamente ho esteso l'insieme dei valori che il mio interprete può restituire in seguito alla chiamata di valutazione di un albero di sintassi astratta (evT).

…

| Set of tval * evT list

….

Regole operazionali del tipo di dato Set

$$\frac{t \in tval}{env \rhd Empty(t) \Longrightarrow Set(t,\{\})}$$

$$\frac{env \rhd e \Longrightarrow g \quad t \in tval \quad typeof(g) = t}{env \rhd Singleton(t,e) \Longrightarrow Set(t,\{g\})}$$

Osservazione: tval rappresenta l'insieme di tipi che possono essere associate al tipo Set (costruito sempre con i tipi algebrici di Ocaml, in cui ogni tipo possibile è rappresentato da un costruttore).

Implementazione operazioni di base

Successivamente sono passato alla definizione delle operazioni di base estendendo l'albero di sintassi astratta con gli appositi costruttori per permettere l'introduzione di operazioni su insiemi.

La loro valutazione restituisce ovviamente un evT.

Nell'implementazione di alcune operazioni ho preferito restituire eccezioni che avvertono l'utente di un errato utilizzo dell'operazione piuttosto che non avere effetti collaterali.

(Es remove chiamato con un set ed un elemento che hanno tipi diversi, così come Insert ecc.)

…

| Insert of exp * exp

| Remove of exp*exp

| IsEmpty of exp

| IsInSet of exp*exp

| Union of exp*exp

| Intersection of exp*exp

| Difference of exp*exp

| Subset of exp*exp

| Max of exp

| Min of exp

| ForAll of exp*exp

| Exsist of exp*exp

| Filter of exp*exp

| Map of exp*exp

Regole operazionali per le operazioni di base

Nella realizzazione delle regole operazionali ho preferito astrarre sui dettagli implementativi delle operazioni di base e usare la notazione matematica per esprimere operazioni come l'unione o l'intersezione tra insiemi.

Il tipo di dato Set contiene al suo interno il tipo dei valori contenuti e una lista di evT: ho considerato la lista del Set ad un livello di astrazione più elevato che mi permette di trattarla come un insieme matematico.

La descrizione della semantica operazionale risulta così più snella e comprensibile.

$$\frac{env \rhd e1 \Longrightarrow g \quad env \rhd e2 \Longrightarrow Set(t,list) \quad typeof(g) = t}{env \rhd Insert(e1,e2) \Longrightarrow Set(t,\ list \cup \{g\})}$$

$$\frac{env \rhd e1 \Longrightarrow g \quad env \rhd e2 \Longrightarrow Set(t,list) \quad typeof(g) = t}{env \rhd Remove(e1,e2) \Longrightarrow Set(t,\ list \setminus \{g\})}$$

$$\frac{env \rhd e \Longrightarrow Set(t,list)}{env \rhd isEmpty(e) \Longrightarrow list = \{\}}$$

$$\frac{env \rhd e1 \Longrightarrow g \quad env \rhd e2 \Longrightarrow Set(t,list)}{env \rhd IsInSet(e1,e2) \Longrightarrow g \in list}$$

$$\frac{env \rhd e1 \Longrightarrow Set(t1,list1) \quad env \rhd e2 \Longrightarrow Set(t2,list2)}{env \rhd Subset(e1,e2) \Longrightarrow lis1 \subset list2}$$

$$\frac{env \rhd e \Longrightarrow Set(t,list)}{env \rhd MaxSet(e) \Longrightarrow max\{list\}}$$

$$\frac{env \rhd e \Longrightarrow Set(t,list)}{env \rhd MinSet(e) \Longrightarrow min\{list\}}$$

$$\frac{env \rhd e1 \Longrightarrow Set(t1,list1) \quad env \rhd e2 \Longrightarrow Set(t2,list2) \quad t1=t2}{env \rhd Union(e1,e2) \Longrightarrow Set(t1,\ list1 \cup list2)}$$

$$\frac{env \rhd e1 \Longrightarrow Set(t1,list1) \quad env \rhd e2 \Longrightarrow Set(t2,list2) \quad t1=t2}{env \rhd Intersection(e1,e2) \Longrightarrow Set(t1,\ list1 \cap list2)}$$

$$\frac{env \rhd e1 \Longrightarrow Set(t1,list1) \quad env \rhd e2 \Longrightarrow Set(t2,list2) \quad t1=t2}{env \rhd Difference(e1,e2) \Longrightarrow Set(t1,\ list1 \setminus list2)}$$

$$\frac{env \rhd Apply(e1,\_) \Longrightarrow Bool(v) \quad env \rhd e2 \Longrightarrow Set(t1,list1)}{env \rhd ForAll(e1,e2) \Longrightarrow \forall\ x \in list1.\ (Apply(e1,x) = Bool(true))}$$

$$\frac{env \rhd Apply(e1,\_) \Longrightarrow Bool(v) \quad env \rhd e2 \Longrightarrow Set(t1,list1)}{env \rhd Exsist(e1,e2) \Longrightarrow \exists\ x \in list1.\ (Apply(e1,x) = Bool(true))}$$

$$\frac{env \rhd Apply(e1,\_) \Longrightarrow Bool(v) \quad env \rhd e2 \Longrightarrow Set(t1,list1)}{env \rhd Filer(e1,e2) \Longrightarrow\ Set(t1,list2)\ con\ list2 = \{x|\ x \in list1\ ,\ (Apply(e1,x) = Bool(true))\}}$$

$$\frac{env \rhd Apply(e1,\_) \Longrightarrow g \quad typeof(g)=t2 \quad env \rhd e2 \Longrightarrow Set(t1,list1)}{env \rhd Map(e1,e2) \Longrightarrow\ Set(t2,list2)\ con\ list2 = \{y|\ x \in list1\ ,\ y=Apply(e1,x)\}}$$

da notare il tipo del set che cambia in relazione al tipo che ritorna la funzione