Documentazione secondo progetto di PR2

Anno 2020-2021

Fabio Bocci 580100

# Interprete in Ocaml

Per le definizioni base del linguaggio ho usato quelle gia viste a lezione, quindi dopo aver copiato l’implementazione base del linguaggio l’ho esteso aggiungendo gli insiemi e le cose rischieste. (NB: non ho aggiunto l’implementazione delle stringe solo per un fattore di tempo e perché non avrebbe cambiato nulla nell’implementazione oltre a fare altri test).

# Implentazione degli Insiemi

Set of setList \* TIPO.

Date le richieste che ogni insieme doveva essere insieme ad un Tipo ho creato una nuova tipo per definire il tipo dell’insieme ( TIPO = |TInt |TBool | TUnbound ) inoltre essendo un insieme non ci possono essere valori doppi, quindi subito dopo aver controllato che l’insieme rispetti il tipo vado a controllare che non abbia elementi doppi.

Successivamente ho implementato le funzioni richieste come:  
***Setempty of TIPO*** -> che restituisce un insieme vuoto di Tipo= TIPO  
***SetInsert SetRemove*** -> che fanno l’operazione di Insert e rispettivametne di Remove da un insieme dopo aver controllato il tipo   
***Singleton of exp \* tipo*** -> che restituisce un insieme di un solo elemento di tipo =TIPO  
***Of of tipo \* explist*** -> implementazione identica a Set, restituisce un insieme di Tipo= TIPO con all’interno la valutazione nell’ambiente delle espressioni.

# Funzioni : Insiemi -> Bool

Come richiesto ho implementato le funzioni:  
IsEmpty of exp -> che restituisce un True se l’insieme passata come parametro è vuota, false altrimenti

IsIn of exp \* exp -> che restituisce True se l’elemento passato è presente nella l’insieme passata come parametro, false altrimenti

Subset of exp \* exp -> che restituisce True se il primo insieme è un sotto insieme del secondo, false altrimenti

# Funzioni insiemistiche : Insiemi -> Insiemi

Come richiesto ho implementato le funzioni insiemistiche come:  
Union of exp \* exp -> che effettua l’unione fra insiemi dopo aver controllato che i tipi corrispondano

Intersection of exp \* exp -> che effettua l’intersezione fra insiemi dopo aver controllato che i tipi corrispondano

Diff of exp \* exp -> che effettua la differenza insiemistica fra il secondo insieme ed il primo, dopo aver controllato che i tipi corrispondano

# Funzioni con LAMBDA : Insieme \* funzione ->

For\_all of exp \* exp -> restituisce True se ogni elemento dell’insieme aplicato alla funzione restituisce True, altrimenti False

Exists of exp \* exp -> implementazione come la For\_all, restituisce True se esiste almeno un elemento dell’insieme che applicato alla funzione restituisce True , altrimenti False

Filter of exp \* exp -> restituisce un sottoinsieme dell’insieme passato come parametro tale che soddisfi la funzione passata come parametro

Map of exp \* exp -> restituisce un insieme dello stesso tipo in modo tale che gli elementi V = f(x) dove x era l’elemento prima dell’applicazione della funzione

# Conclusioni

Ho dovuto implementare un nuovo Typechecker per i Tipi degli insiemi e farli matchare con i tipi gia creati da evT ( guardare funzione “newTypecheck x y”)

Era possibile anche un implementazione delle Stringhe però c’era bisogno di modficare a quel punto anche TIPO aggiungendo il tipo TString e modificare newTypecheck