Relazione secondo progetto intermedio – OCaml

di Francesco Di Luzio

Un insieme è una coppia formata dal tipo dell'insieme e dagli elementi non ordinati, inoltre tra gli elementi non sono presenti duplicati. Ogni elemento ha lo stesso tipo dell'insieme. Nell'implementazione, il tipo esprimibile di un insieme è l'espressione "Set(tipo, lista)" dove gli elementi dell'insieme sono rappresentati da una lista.

Come richiesto dal testo del progetto sono state introdotte tre espressioni per la creazione di un insieme: "EmptySet", che crea un insieme vuoto di un determinato tipo; "Singleton", che crea un insieme contenente un singolo elemento di un determinato tipo; "Of", che crea un insieme contenente una serie di elementi di un determinato tipo.
Una serie di elementi può essere creata tramite l'espressione "valuesList" che ha una definizione ricorsiva. "valuesList" può essere "Empty" oppure una coppia formata da un'espressione e da una "valuesList".

Per i tipi degli insiemi si è optato per una gestione tramite stringhe. I tipi consentiti sono "int", "bool", "string".
Non si è ritenuto adatto consentire la creazione di insiemi di funzioni in quanto bisognerebbe avere una metodologia di confronto tra funzioni. Bisognerebbe controllare che tutti gli elementi dei due dominii, che potrebbero essere infiniti, vengano associati agli stessi elementi del codominio.

Il linguaggio didattico è stato esteso aggiungendo ai tipi "set", che identifica un insieme.

Nelle operazioni primitive, dato che il linguaggio didattico ha typecheck dinamico, viene controllata la correttezza dei tipi. La "Add", per esempio, richiede come primo parametro un insieme e come secondo parametro un valore che abbia lo stesso tipo dell'insieme.

Nelle operazioni per determinare il massimo e il minimo all'interno di un insieme sono consentiti tutti e tre i tipi. La scelta deriva dall'implementazione di OCaml stesso, a cui il linguaggio didattico è ispirato, in cui è presente una nozione di massimo e minimo sia per interi che per stringhe e booleani.

È stata necessaria la definizione di una funzione "check" che permettesse di ricavare il tipo restituito dalle espressioni. Questo è possibile grazie alla notazione prefissa

usata dalla sintassi astratta del linguaggio; si analizza l'espressione più esterna e, se necessario, si effettua l'analisi nelle espressioni più interne.
Grazie a questa funzione, è stato possibile aggiungere il controllo all'espressione "Ifthenelse" per avere lo stesso tipo restituito in entrambi i rami.

La funzione "check" è stata fondamentale per le implementazioni delle operazioni primitive "For_all", "Exists" e "Filter" in quanto è necessario controllare che l'espressione associata al primo parametro sia effettivamente un predicato.
Nell'operazione primitiva "Map", invece, la funzione "check" è stata usata per ricavare il tipo restituito dalla funzione associata al primo parametro in modo tale da restituire l'insieme con il tipo corretto.

Per concludere, il linguaggio è stato esteso anche con le operazioni logiche "And", "Or" e "Not" a cui non è stato possibile applicare regole di corto circuito. È necessaria la valutazione di entrambe le espressioni altrimenti potrebbero essere valide espressioni sintatticamente non corrette.