Progetto di Linguaggi e Compilatori 1 – Parte 1
A.A. 2015/16

Gruppo 14

Marco Bucchiarone
Emanuele Tonetti
Francesco Zilli

Esercizio 1

La funzione Haskell `boundedMaximum(n [BST t])`

`boundedMaximum(n[BSTt])`

data una lista di BST

`t`

`t` trova, se esiste

$$max_n t = max(\forall x \in t \mid x < n).$$

La soluzione all'esercizio è implementata nel modulo BoundMax (`BoundMax.hs`) che viene importato nel `Main`. Nel `Main` la funzione viene chiamata fornendole gli argomenti inseriti a tastiera a runtime (usando le funzioni importate tramite `System.Environment`).

La `boundedMaximum` viene costruita sulla `foldl1`: `MaxNT` individua il massimo tra i minoranti nella lista costruita dalla funzione di visita del BST `getmin` che sfrutta la struttura dati considerata. Il modulo `BoundMax` testabile nell'interprete di GHC tramite i test case forniti nel file `queryEs1.txt`, il quale comprende anche una rappresentazione grafica di alcuni degli alberi usati per una maggiore leggibilità.

Esercizio 2

a

Dalla sintassi concreta per alberi pesati con numero arbitrario di figli fornita è stata costruita la sintassi astratta *polimorfa*. L'implementazione proposta suddivisa tra i seguenti file, di cui si da una breve descrizione:

Data.hs contiene la definizione dei tipi di dato per la sintassi astratta polimorfa e per i token

Lexer.x analizzatore lessico-grafico che produce token per dati di tipo Integer e Double;

ParserI.y parser dedicato alla restituzione di alberi pesati di numeri soli Integer;

ParserD.y parser dedicato alla restituzione di alberi pesati di numeri Double;