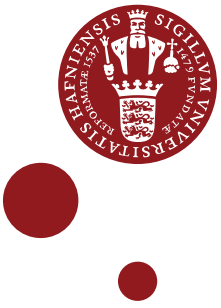
Advanced Programming

Assignment 1: Arithmetic expressions

Gilles Souton

September 16, 2022



1 Simple arithmetic expressions

1.1 Printing expressions

Not much to note here, except that *showExp* use unnecessary parenthesis.

1.2 Evaluating expressions

When using the *Pow* and raising any expression to the power of 0 operation and raising a number to the power of 0: $Pow\ (Exp)\ (Cst\ 0)$ it seems that the operator does not even try to evaluate the inner expression and therefore returns 1 by default.

So in case of an error in an inner expression of Pow the evaluation does not crash i.e: $Pow\ (Div\ (Cst4)\ (Cst0)\ )\ (Cst\ 0)$ here $(Div\ (Cst4)\ (Cst0))$ returns an error when evaluated

2 Extended arithmetic expressions

Here to evaluate the operator sum the chosen implementation was the following:

1. Generate a list containing all the numbers from the given range.
2. map to each element of the list the body of the sum
3. uses the sum function to sum each element in the list.

3 Returning explicit errors

Here the heavy use of Either showed a good way to propagate error since that the problem that was present with the operator *Pow* with the previous implementations of *evalSimple* and *evalFull* is inherently avoided.

Remarks two extras functions were added to try to factorize the code.

1. `evalBranchErr :: Exp -> Exp -> Env -> Either ArithError (Integer, Integer)`

   This function evaluate tow expressions in a given environment and returns each value for each expression. It is mainly used for to factorize the code for the operators *Add Sub Mul...etc*

   Note it does not evaluate the second expression if the evaluation of the first one fails.

2. `evalSum :: [] Integer -> VName -> Exp -> Env -> Integer -> Either ArithError Integer`

   This function takes an array and sums each element of the array while applying to each element the given expression in the given environment. It is used only in *evalErr* when a *Sum* needs to be evaluated

   Note Same as the previous function when an error occurs the rest of the list is not evaluated.

Most of the design and implementation choices were made to have the simplest approach possible.