Expression Problem

From Wikipedia, the free encyclopedia

The **Expression Problem** is a term used in discussing strengths and weaknesses of various programming paradigms and programming languages. The expression problem can treated as a use case in programming language design.[1] [2] [3] [4] [5]

Philip Wadler coined the term:

> The Expression Problem is a new name for an old problem. The goal is to define a datatype by cases, where one can add new cases to the datatype and new functions over the datatype, without recompiling existing code, and while retaining static type safety (e.g., no casts). [6]

Wadler selected the term as a pun. On the one hand, the programmer is trying to "express" a solution to a problem. On the other hand, the standard illustrative example given is that of an interpreter for expressions in some simple calculator language.

The expression problem is also a fundamental problem in multi-dimensional Software Product Line design and in particular as an application (?) or special case(?) of FOSD Program Cubes (http://en.wikipedia.org /wiki/FOSD_Program_Cubes#Applications) .

News:

A modular solution of the expression problem was given by Lect.Drd. Dan Popa, from Bacau University during the summer of 2008. His idea have at least three important parts:

1)to produce a modular tree by using a sort of special functions called pseudoconstructors

2)to produce a modular language interpreter or a modular language evaluator using pseudoconstructors over monadic values instead of usual monadic semantics

3)to NOT use an **interpret** or an **eval** function but to make every expression an itself evaluator.

Basically, the main idea was to replace a typical case from the evaluator or interpreter which is written in Haskell as

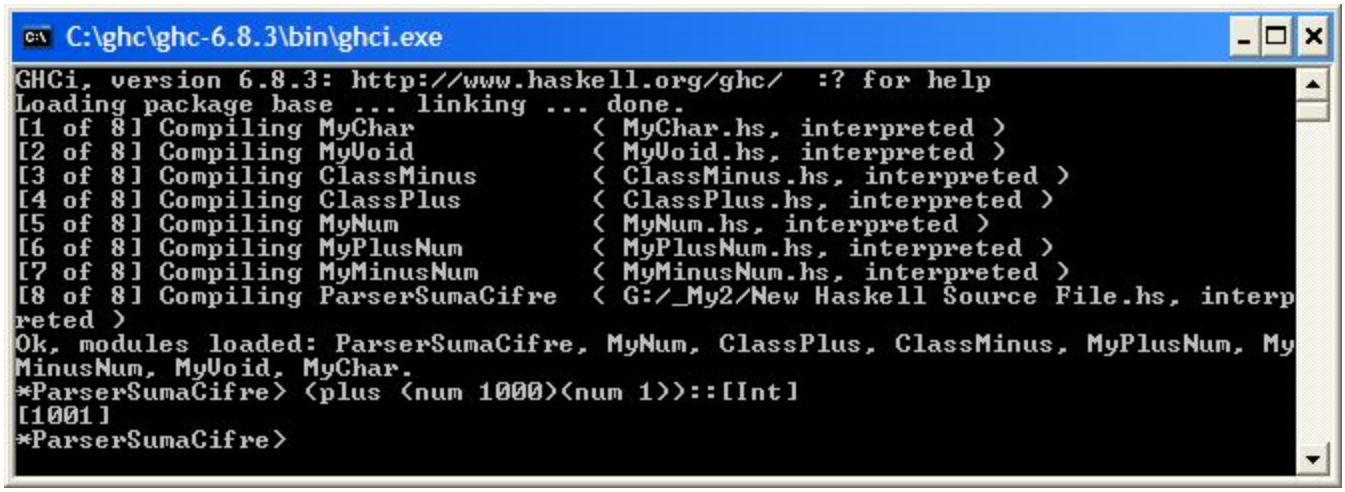
```
do {vx <-interp x env;
    vy <-interp y env;
    return(vx +  vy); }:: M Float
```

by something very simple, a modular simple **itself evaluator**:

```
plus x y  =
do { vx <-x;
     vy <-y;
     return(vx +  vy); }:: M Float
```

supported by some definitions for numbers and classes of operations.Such functions can be spread in various modules.

As a result,an interpreter or an evaluator can be built by simply imported all the required modules in a main client module (program.)



In the figure above you may see GHCI including the modules and being able to evaluate an abstract syntax tree which is built using pseudoconstructors. Note the fact that pseudoconstructors did not use Capitals as the usual constructors from the data declaration.

References

- Direct modular evaluation of expressions using the monads and type classes in Haskell by DAN V. POPA ; UNIVERSITATEA DIN BACĂU ;STUDII ŞI CERCETĂRI ŞTIINŢIFICE ;Seria: MATEMATICĂ ; Nr. 18 (2008), pag. 233 – 248 (http://www.haskell.org/sitewiki/images/7/7d/POPA_D.pdf.)
- Home Page of Dan Popa - The User:Ha$kell (http://www.haskell.org /haskellwiki/User:Ha%24kell)
- Applications of FOSD Program Cubes (http://en.wikipedia.org /wiki/FOSD_Program_Cubes#Applications)

- Generic Programming (http://en.wikipedia.org/wiki/Generic_programming)

1. ^ "User-defined types and procedural data structures as complementary approaches to data abstraction". http://www.google.com/url?sa=U&start=1&q=http://portal.acm.org/citation.cfm%3Fid%3D186680&ei=iM3QSdLVNYLoyAWv3szRCQ&usg=AFQjCNHnntAxyeeK5SzrnmPiJ77Qgr_pSw.
2. ^ "Object-Oriented Programming versues Abstract Data Types". http://www.google.com/url?sa=U&start=1&q=http://www.cs.utexas.edu/users/wcook/papers/OOPvsADT/CookOOPvsADT90.pdf&ei=Ms7QSYOwAqCyyQXPioHKCQ&usg=AFQjCNEx070KUZDcmbDS_x3uslsIYSaZCQ.
3. ^ "Synthesizing Object-Oriented and Functional Design to Promote Re-Use". http://www.google.com/url?sa=U&start=2&q=http://citeseer.ist.psu.edu/krishnamurthi98synthesizing.html&ei=xczQScHyNIvEyQWEv9nWCQ&usg=AFQjCNEU3GvIyhjYeU2JndB3Uu3-1olEBQ.
4. ^ "Extensible Algebraic Datatypes with Defaults". http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.6778.
5. ^ "Independently extensible solutions to the expression problem". http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.107.4449.
6. ^ "The Expression Problem". http://www.daimi.au.dk/~madst/tool/papers/expression.txt.

Retrieved from "http://en.wikipedia.org/wiki/Expression_Problem"
Categories: Computer programming
Hidden categories: Computer science articles needing expert attention | Articles needing expert attention from May 2009 | Articles lacking sources | All articles lacking sources

---