Minicaml, a purely functional, didactical programming language with an interactive REPL.

Alessandro Cheli
Course taught by Prof. Gianluigi Ferrari
and Prof. Francesca Levi

December 23, 2019

Abstract

minicaml is a small, purely functional interpreted programming language with a didactical purpose. It is based on the Prof. Gianluigi Ferrari and Prof. Francesca Levi's minicaml, an evaluation example to show students attending the Programming 2 course at the University of Pisa how interpreters work. It is an interpreted language with a Caml-like syntax, featuring interchangeable eager and lazy evaluation and a didactical REPL that shows each AST expression and each evaluation step.

1 REPL and command line interface

1.1 Installation

minicaml is available in the opam 2.0 repository. (`https://opam.ocaml.org/`). The easiest way to install minicaml is with the OCaml package manager **opam**. To do so, please check that you have a version of opam $\geq$ 2.0.0 and run:

```
opam install minicaml
```

Alternatively, **minicaml** can be installed from source by downloading the source code git repository and building it manually. **minicaml** has been tested only on Linux and macOS systems. It has not been tested yet on Windows and BSD derived systems.

```
# download the source code
git clone https://github.com/0x0f0f0f/minicaml
# cd into the source code directory
cd minicaml
# install dependencies
opam install ANSITerminal dune ppx_deriving menhir cmdliner
# compile
make
# execute
make run
# install
make install
```

2 Lexer

3 Parser

4 AST Optimization

Before being evaluated, AST expressions are analyzed and optimized by an optimizer function that is recursively called over the tree that is representing the expression. To achieve minimization to an unreducible (not optimizable)

form, optimizer calls are iterated on each previous call's result; this way, when a tree representing an expression cannot be optimized again. This is the process represented in pseudo-code:

5 Evaluation

6 Tests

Unit testing is extensively performed using the alcotest testing framework.

7 Thanks

Thanks to Antonio DeLucreziis for helping me introduce lazy evaluation.