An experiment in teaching functional programming to musicology students

Philippe ÉZÉQUEL

CIEREC, Université Jean Monnet, Saint-Étienne

Summary

1. Background and motivations
2. The Peano language
3. A tentative curricula (interleaved)
4. Feedback from the RIM students (2015 and 2016)
5. Technicalities

Background

- Computer Science perspective
- More than 10 years feedback on teaching functional programming
- Adaptation to musicology students : Lisp is the final goal

Motivations

A statement

Functional programming is hard :

- for beginners in programming
- for seasoned programmers used to imperative paradigm

Possible explanations

- "real" functional programs may be rather obfuscated
  "*Unfortunately a Russian spy stole the last meg of a LISP program for controlling our nuclear defense systems. Fortunately, it was all right-parenthesis.*"
- pure functional languages lack well known control structures

What is needed

A simple language

- with minimal syntactic sugar
- Turing-equivalent
- allowing to play with lists (and trees ?)
- providing useful features for learners

The Peano machine : basic model

- evaluates arithmetic expressions
- one can define new functions

Expressions

(i) 0

(ii) $++$ Expr

(iii) if $E_1 = E_2$ then E_3 else E_4

(iv) function call

Where the name Peano comes from

Giuseppe Peano described the integers using 0, successor function and recurrence

The basic Peano machine : curricula

Arithmetic

- $--$ Expr
- addition, subtraction
- comparisons, multiplication
- quotient, remainder

Programming skills

- recurrence, recursion and induction
- purely recursive programming
- iterative recursive programming using accumulators

The Peano machine : DeLuxe model

Basic Peano machine, plus

1. all the arithmetic operators defined so far
2. List data type : cons, car, cdr, null

The DeLuxe Peano machine : curricula

Lists

- length of a list
- append two lists
- reverse a list, in 3 flavors
- playing with lists : shuffle, interleave, lists of lists,...

Programming and Computer Science skills

- there is more than one way to do something
- complexity issues steming from algorithms or programming
- "Elegance is not optional" (R. O'Keefe, *The Craft of Prolog*)

Feedbacks from the RIM students

- years 2015 and 2016
- 3 students of the 2015 class (out of 8), 9 of the 2016 class (out of 10)
- only positive feedbacks :
  - "Syntax and semantics are simple and clear"
  - "Way easier to program with than Lisp"
  - "A good introduction to Lisp, OpenMusic and Faust"
  - "The tools provided by the interpreter are very useful"

Technicalities : how to use the Peano Machine

- edit a source file
- compile it, getting an interpreter
- launch the interpreter
- demo !

Technicalities : features of the interpreter

- read-eval-print loop
- history of commands
- line editing facilities (à la Emacs. . .)
- special commands :
  - `listing` : to get the list of the functions defined
  - `trace` : to have a trace of the functions calls (`notrace` to switch off)
  - `calc` : to show how many internal operations were executed while evaluating an expression (`nocalc` to switch off)
  - `mem` : to show how many memory cells were needed while evaluating an expression (`nomem` to switch off)
- demo !

Technicalities : implementation

- source to source compilation from Peano to C, then gcc
- usual list and binary tree functions
- atoms à la Lisp
- graceful handling of non-terminating function calls
- demo !

Technicalities : what remains to be done

- auto completion of function names
- infinite precision arithmetic
- more pleasant interface
- ...

Downloading the Peano machine

You can freely download the Peano machine at this URL :

`http://webperso.univ-st-etienne.fr/~ezequel/Peano/`