DSL in Clojure

Dimitry Gashinsky (@digash)

Pico Quantitative Trading LLC.

17 August 2010

Outline

Who am I? vanity

Used to be by day

: Coder for financial industry.

Used to be at night

: Lisp enthusiast and computational philosopher.

Now in twilight

: VP of engineering for Pico Quantitative Trading, which includes all of the above.

R-LISP

REDUCE

$$\int_0^y cos(2x)\,dx. \tag{1}$$

to solve this integral type at the REPL

```
int(cos(2x),x,y,2y);
```

get this output

```
 SIN(4*Y) - SIN(2*Y)
---------------------
          2
```

Figure: R-LISP Book

definition by listing

parasitic language

targeted towards specific problem

Quick dip into Clojure features useful for DSLs

unquote and unquote-splicing

code as data

macrology

Overview of DSLs in Clojure

TODO search article about clojure DSLs

TODO search for blog about DSLs

Common usage

configuration language

Every application needs a DSL

Questions

Questions?

mailto:dimitry@gashinsky.com

twitter:@digash

xmpp:i@digash.com

http://blog.digash.com

Structure and Interpretation of Computer Programs

When in need of deep wisdom use the book[2].

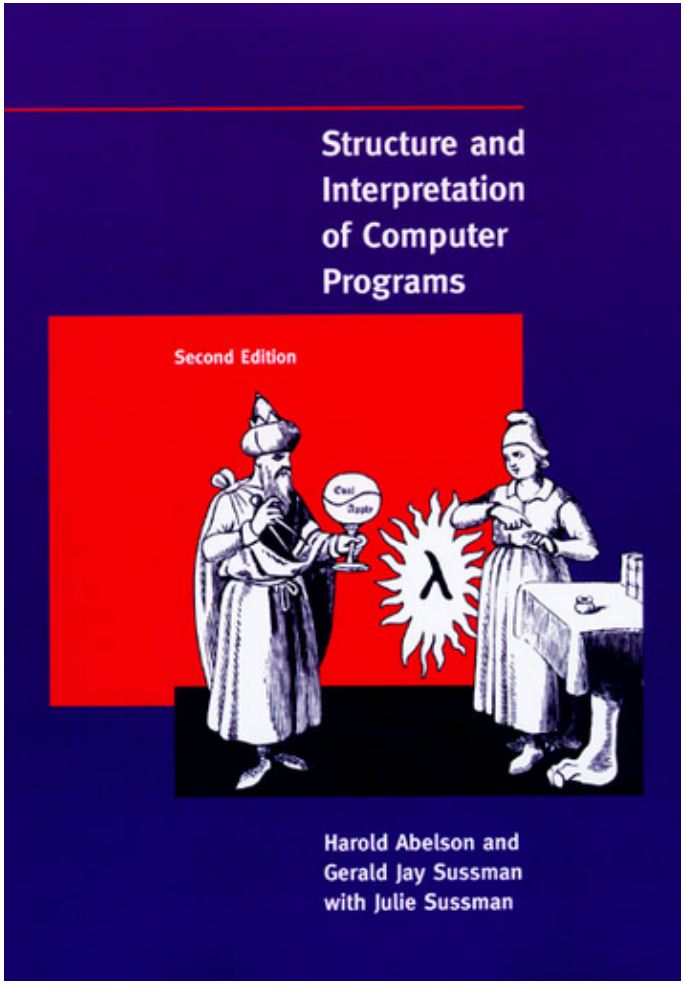

Figure: SICP

Metalinguistic Abstraction, A.K.A. "DSL"

Establishing new languages

- a powerful strategy for controlling complexity
- particularly important to computer programming, because we can implement these languages

The book[2] *tellsus*

> "The evaluator [or compiler], which determines the meaning of expressions in a programming language, is just another program."

"Maxwell's Equations of Software!" – Alan Kay

if either side of the equation is defined at all.

Example

```
f:      λ[[x;y];cons[car[x];y]]
fn:       (LAMBDA (X Y) (CONS (CAR X) Y))
arg1:     (A B)
arg2:     (C D)
args:     ((A B) (C D))

evalquote[(LAMBDA (X Y) (CONS (CAR X) Y)); ((A B) (C D))] =
      λ[[x;y];cons[car[x];y]][(A B);(C D)]=
      (A C D)
```

evalquote is defined by using two main functions, called eval and apply. apply handles a function and its arguments, while eval handles forms. Each of these functions also has another argument that is used as an association list for storing the values of bound variables and function names.

```
evalquote[fn;x] = apply[fn;x;NIL]
```

where

```
apply[fn;x;a] =
    [atom[fn] → [eq[fn;CAR] → caar[x];
                 eq[fn;CDR] → cdar[x];
                 eq[fn;CONS] → cons[car[x];cadr[x]];
                 eq[fn;ATOM] → atom[car[x]];
                 eq[fn;EQ] → eq[car[x];cadr[x]];
                 T → apply[eval[fn;a];x;a]];
     eq[car[fn];LAMBDA] → eval[caddr[fn];pairlis[cadr[fn];x;a]];
     eq[car[fn];LABEL] → apply[caddr[fn];x;cons[cons[cadr[fn];
                                                caddr[fn]];a]]]
eval[e;a] = [atom[e] → cdr[assoc[e;a]];
     atom[car[e]] →
                [eq[car[e];QUOTE] → cadr[e];
                 eq[car[e];COND] → evcon[cdr[e];a];
                 T → apply[car[e];evlis[cdr[e];a];a]];
     T → apply[car[e];evlis[cdr[e];a];a]]
```

pairlis and assoc have been previously defined.

```
evcon[c;a] = [eval[caar[c];a] → eval[cadar[c];a];
              T → evcon[cdr[c];a]]
and
evlis[m;a] = [null[m] → NIL;
              T → cons[eval[car[m];a];evlis[cdr[m];a]]]
```

13

Figure: LISP 1.5 Programmer Manual page 13

Why Rich should have all the fn?

"We come to see ourselves as designers of languages, rather than only users of languages designed by others."

"... computer science itself becomes no more (and no less) than the discipline of constructing appropriate descriptive languages."

```
(defn slice [^ByteBuffer b n]
  (vec (for [p (range 0 (.capacity b) (length MarketDataM
         (let [^ByteBuffer b (.position b p)
               ^ByteBuffer s (.slice b)]
           (.limit s (length MarketDataMmap))))))
```

```
(+ 1 4)
```

```
ls
```