

Re: spec 3.1.2.1.2 and the lambda exception

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2004-08/1542.html>

From: Pascal Bourguignon (*spam_at_thalassa.informatimago.com*)

Date: 08/24/04

Date: 24 Aug 2004 00:31:51 +0200

Ross Lippert <lippert@MATHSTATION031.MIT.EDU> writes:

> *I have been trying to learn some lisp. I have had some experience in
 > scheme, which is probably why I even got disturbed by that which is
 > the source of my question.*
 >
 > *Playing around with the age old bank-account example I did something
 > equivalent to*
 > *(defun make-adder (x) (lambda (y) (+ x y)))*
 > *((make-adder 1) 1)*
 > *and of course, I got an error. What I mean to do here is done by
 > using funcall. This is part of that 1-namespace vs 2-namespace
 > issue, which divides scheme and lisp, I know.*
 > *[...]*
 > *I guess that 3.1.2.1.2 allows lambdas and no other expressions
 > because either:*
 > *1) there is some good reason to allow lambda expressions in the car.*
 > *2) there is some good reason to disallow arbitrary function-returning
 > expressions in the car.*
 >
 > *Which is it and why?*

Both.

1: Because a list of the form (lambda (arg...) exp...) IS a function.

I mean, this is a function:

```

+-----+
|
| |
| +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+ |
| | * | * | --> | * | * | --> | * | * | --> | * | NIL | |
| +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+ |
| | | | | | | | |
| v | v v |
| +-----+ | +-----+ +-----+ |
| | LAMBDA | | | EXP0 | | EXP1 | |
| +-----+ | +-----+ +-----+ |
| v |

```

```
|+---+---+ +---+---+ |
| | * | * |--->| * |NIL| |
|+---+---+ +---+---+ |
| | | |
| v v |
|+-----+ +-----+ |
| | ARG0 | | ARG1 | |
|+-----+ +-----+ |
| |
+-----+
```

2: Because there is no difference between a function call and a variable evaluation. Both are executed with EVAL, and since Common-Lisp is a lisp-2, it won't evaluate the first item in a sexp.

Note however that you can still play some tricks:

```
CL-USER> (defun make-adder (x) `(lambda (y) (+ ,x y)))
MAKE-ADDER
CL-USER> (#.(make-adder 1) 1)
2
```

Of course, the adder will be created at read time instead of run time.

```
--
__Pascal Bourguignon__ http://www.informatimago.com/
Our enemies are innovative and resourceful, and so are we. They never
stop thinking about new ways to harm our country and our people, and
neither do we.
```