Homework, Week 5, part A

November 22, 2013

This week, you will implement the code for the Wizard game in Chapter 5. It's best for you to copy the code manually, that is, type it yourself with te keyboard. If you need to, you can copy it from the web site.

There are five pieces of code that we will look at today. The first is the list `*nodes*` on page 70. This is nothing more than the questions and answers that we did for the test. If you like, you can type this in the same way we did for the tester program. The second list is `*edges*` on page 72. Again, you have seen this before.

There are also three functions. The first is `(describe-location)`. Please understand what `(assoc)` does, what it returns, and why we take the `(cadr)` of the return value of `(assoc)`. The second function is`(describe-path)`. Again, you should understand what the `(cadr)` and the `(caddr)` of the "edge" that is passed as an argument to the function.

The final function for today is `(describe-paths)`. IT IS VERY, VERY IMPORTANT THAT YOU UNDERSTAND THIS FUNCTION IN DETAIL! If possible, you should tatoo this on the back of your eyelids so you can study it with your eyes closed. As with many functions in Lisp, you should understand this function from the inside out. The first piece is `(assoc)` — understand completely the arguments that are passed to this function, and what it returns. The second piece is `(cdr)` — again, understand why you need to take the remainder of the list returned. What is the return value of `(assoc)`, and what is the `(cdr)` of it? The third and fourth functions are `(describe-path)` and `(mapcar)`. You already know what `(describe-path)` does. What does `(mpacar)` do, and what does it produce as a return value? The last two functions are `(append)` and `(apply)`. Once you understand all the pieces, understand how the function `(describe-paths)` works as a whole. You will be seeing a lot of this kind of coding throughout this book, and the earlier you understand how it works, the more progress you will make.

I don't like to do things the same way Barski does. Here is my version of the definition of `*nodes*` and `*edges*`. I also added a bedroom and a kitchen. If you like, you can add other locations. You can do things however you like.

```lisp
(defparameter aplace 'living-room)
(defparameter adesc '(you are in the living-room. a wizard is
    snoring loudly on the couch.))
(defparameter a (list aplace adesc))

(defparameter bplace 'garden)
(defparameter bdesc '(you are in a beautiful garden. there is
    a well in front of you.))
(defparameter b (list bplace bdesc))

(defparameter cplace 'attic)
(defparameter cdesc '(you are in the attic. there is a giant
    welding torch in the corner.))
(defparameter c (list cplace cdesc))

(defparameter dplace 'bedroom)
(defparameter ddesc '(you are in the bedroom. there is an
    unmade bed in the corner.))
(defparameter d (list dplace ddesc))

(defparameter eplace 'kitchen)
(defparameter edesc '(You are in the kitchen. there are
    strange green things growing from dirty plates on the
    table.))
(defparameter e (list eplace edesc))

(defparameter *nodes* (list a b c d e))

(defun describe-location (location nodes)
   (cadr (assoc location nodes)))

(defparameter qedge 'living-room)
(defparameter qpath-1 '(garden west door))
(defparameter qpath-2 '(attic upstairs ladder))
(defparameter qpath-3 '(kitchen south arch))
(defparameter qpath-4 '(bedroom east door))
(defparameter q (list qedge qpath-1 qpath-2 qpath-3 qpath-4))

(defparameter redge 'garden)
(defparameter rpath-1 '(living-room east door))
(defparameter r (list redge rpath-1))

(defparameter sedge 'attic)
(defparameter spath-1 '(living-room downstairs ladder))
(defparameter s (list sedge spath-1))

(defparameter tedge 'bedroom)
(defparameter tpath-1 '(kitchen west door))
(defparameter tpath-2 '(living-room west door))
;; can't use t as a variable, so use t* instead
```

```
(defparameter t* (list tedge tpath-1 tpath-2))

(defparameter uedge 'kitchen)
(defparameter upath-1 '(living-room north arch))
(defparameter upath-2 '(bedroom east door))
(defparameter u (list uedge upath-1 upath-2))

(defparameter *edges* (list q r s t* u))

(defun describe-path (edge)
  `(there is a ,(caddr edge) going ,(cadr edge) from here.))

(defun describe-paths (location edges)
  (apply #'append (mapcar #'describe-path (cdr (assoc location
      edges)))))
```