CS 424 – Lecture 2 – 25/09/2014

Scheme schema:

* Procedure Calls
* Strings
* Functions

Lambda expressions:

* A Lambda expression is an expression that returns a function
* Takes the form (lambda (var\*) expr) e.g.
* ((lambda (x) (\* x x)) 7) ... This takes in 7, binds it to x, then squares x and prints the result
* Note; Arguments do not always get called from left to right, so scheme requires that arguments do not intermingle

Let:

* Let is a keyword(i.e. not a function) in scheme for binding e.g.
* (let ((x (+ 3 4 ))) (\* x x))
* Let\* is syntactic sugar for binding nested lists

Data Structures

Definition: A higher-order function is a function that takes in a function as an argument or returns a function as a result

What do we have to make data structures?

* Numbers
* Strings
* Lambdas

When we return a lambda a data structure is passed back with the values that were passed in.

Example: Church coded pairs

(define kons

( lambda (a d)

(lambda (z)

(z a d))))

... This is a constructor of a pair, it takes in 2 values, and returns a lambda that allows us to run functions on that pair

(define kar

( lambda (p)

(p (lambda (a d) a))))

... This is a right accessor method on the pair shown above, it returns a function, that we can pass into our pair data structure

(define kdr

(lambda (p)

(p (lambda (a d) d))))

... This is a left accessor method on the pair shown above, it returns a function, that we can pass into our pair data structure