Lisp and Recursion

Christopher Buckingham
Aston University

March 19, 2012

The `member` function in Lisp has been explained earlier and works as follows (note that it is the letter 'l' that is being set to the list):

```
> (setf l '(a r c w d h a w))
(A R C W D H A W)

> (member 'c l)
(C W D H A W)

> (member 'x l)
NIL
```

The English steps for defining it are as follows:

1. If item is equal to the head of the list, return the list.
2. If item is not equal to the head of the list, see if it is a member of the rest of the list.

This is a recursive definition because the definition of the function has the function itself as part of the definition. In short, the function "calls" itself.

1 The lisp equivalent

Note that we have called the function `member1` because otherwise we will redefine Lisp's own `member` function!

```
(defun member1 (item l)
  (unless (null l)              ;; Stopping condition
    (if (equal item (first l))  ;; If at head of list
        l                       ;; Return list
      (member1 item (rest l)))));; Else see if member
                                ;; of rest of list
```

A recursive definition is one where the function body (i.e. its definition) has a call to the function itself. We saw this in the earlier lectures on lisp with the recursive definition of s-expressions:

- An atom is an s-expression
- If S1, S2, ..., Sn are s-expressions, then so is the list (S1 S2 ... Sn).

1.1 Recursive definition of a tree

A tree is a branch that has two or more branches attached where each branch may also be a tree.