Today’s Lecture

Lab Policy

Recursive function definitions in Scheme

Sets

Lab Policy

Attendance is mandatory.

If you are absent without permission, your score is 0.

Exceptions: My prior permission, or else a doctor’s note if you were unwell and did not email me in advance.

Grading your Scheme assignment:

If your file does not load or does not compile, then 0.

If any function is incorrect then 0 for that function.

If you do not implement a function, leave the ”implement” string alone.

Otherwise, your file may not compile!

Late Submission: same penalties as for late problem sets

Evaluating Expressions

Every Scheme expression has a value.

ATOMS: The value of a number or boolean is itself.

The expression `(define x 5)` binds x to 5

LISTS: To evaluate the list `(f a b c)` the interpreter

1. Checks if the first element is the name of a defined function.
   If not, the interpreter gives an error.
2. Evaluates every argument (if any undefined, then return an error).
3. Apply the function `f` (as defined) to the values returned in Step 2.

CAR, CDR, & CONS

The function `car` returns the first element of a list.

`(car '(a b c))` returns `a`

The function `cdr` returns the rest of the list, (i.e. the list minus its first element).

`(cdr '(a (b c) d))` returns `((b c) d)`

The function `cons` inserts the first argument into the second argument which is a list.

`(cons 'x '(a b c))` returns `(x a b c)`

SIMPLE CONDITIONALS

The expression

`(null? x)` returns `#t` if `x` is the null list, `#f` otherwise

`(list? x)` returns `#t` if `x` is a list, `#f` otherwise

`(number? x)` returns `#t` if `x` is numeric, `#f` otherwise

`(boolean? x)` returns `#t` if `x` is a boolean, `#f` otherwise

`(string? x)` returns `#t` if `x` is a string, `#f` otherwise

`(eq? x y)` returns `#t` if `x` and `y` have the same value, `#f` otherwise

CONDITIONAL EXPRESSIONS

```
(if  cond  expr1 expr2)
```

Evaluate `cond`

if true, return value of `expr1`

if false, return value of `expr2`

Does not evaluate `expr1` (`expr2`) unless `cond` is true (false)

CONDITIONAL EXPRESSIONS

```
(cond  (cond1  expr1)
       (cond2  expr2)
       (cond3  expr3)
       (else   expr))
```

Evaluate cond1, cond2, … in sequence.

Return expr$k$ corresponding to the first cond$k$ that evaluates to #t

If none evaluate to #t return expr

Does not evaluate expr$k$ unless cond$k$ is true and cond$i$ is false for all $i < k$.

Recursive Function Definitions

$$f(n) = 1 + f(n-1)$$
$$f(0) = 0$$

$$\begin{aligned}
f(4) &= 1 + f(3) \\
&= 1 + \big(1 + f(2)\big) \\
&= 1 + \Big(1 + \big(1 + f(1)\big)\Big) \\
&= 1 + \bigg(1 + \Big(1 + \big(1 + f(0)\big)\Big)\bigg) \\
&= 1 + \Big(1 + \big(1 + (1 + 0)\big)\Big) \\
&= 1 + \big(1 + (1 + 1)\big) \\
&= 1 + (1 + 2) \\
&= 1 + 3 \\
&= 4
\end{aligned}$$

unwind

evaluate

Defining New Functions

$$\texttt{(define }\underbrace{\texttt{(f x y)}}_{\substack{\text{function name and}\\ \text{list of arguments}}}\ \underbrace{\texttt{(+ (* x x) (* y y))}}_{\text{function body}}\texttt{)}$$

This expression binds the function name `f` to the function body.

Recursive Function Definitions

Define function `(findlast L)` that returns the last element of a list.

`(findlast '(a b c))` returns `c`

`The last element of '(a b c)`

`is the last element of (b c)`

which is `(cdr '(a b c))`

So `(findlast L) is the same as (findlast (cdr L))`

Recursive Function Definitions

A first attempt:

```
(define (findlast L) (findlast (cdr L)))
(findlast '(a b c) )
    (findlast (b c) )
        (findlast (c) )
            (findlast () )
            ERROR!  cdr of null list undefined
```

Recursive Function Definitions

Here's the fix:

```
(define (findlast L)
    (if (null? (cdr L)) (car L)
                        (findlast (cdr L))

(findlast '(a b c) )
    (findlast (b c) )
        (findlast (c) )
            c
```

But what about `(findlast  '() ) ?`

Recursive Function Definitions

Finally:

```
(define (findlast L)
    (cond ((null? L) 'ERROR_EMPTY_LIST )
          ((null? (cdr L)) (car L))
          (else (findlast (cdr L)))))
```

List Length

```
(define (mylength L)
    (if ((null? L) 0
                (+ 1 (mylength (cdr L) ))))
```

Exercises

```
(define (select k L)   … )
```

return the element with index k (first element has index 0)

```
(define (myappend X Y)   … )
```

return a list containing the elements of X followed by elements of Y

```
(define (myreverse X)   … )
```

return the list of elements of X in reverse order

Solutions

```
(define (select k L)
  (cond ((null? L) 'LIST_IS_TOO_SHORT )
        ((< k 0)  ‘NO_SUCH_INDEX )
        ((= k 0) (car L))
        (else (select (- k 1) (cdr L)))))

(define (myappend X Y) (cond
                         ((null? X) Y)
                         ((null? Y) X)
                         (else (cons (car X) (myappend (cdr X) Y)))))
(define (myreverse X)
  (cond ((null? X) X)
        (else (myappend (myreverse (cdr X))(list (car X))))))
```

Sets

A set is an unordered collection of objects, called members or elements of the set.

$x \in S$ represents the proposition "x is a member of S."

$x \notin S \equiv \neg(x \in S)$ (x is not a member of S).

Sets can contain numbers, letters, people, strings, trees, birds, … as members.

$$\{1, 2, Jack, Jill, elm, sparrow, USA\}$$

Can a set contain no members?

Sure, the *empty set* contains no members.

There is a unique empty set, denoted Φ

Is the proposition $\forall x \in \Phi: x = x$ true? Yes

Is the proposition $\forall x \in \Phi: x \neq x$ true? Yes!

Is the proposition $\exists x \in \Phi: x = x$ true? No

Can a set contain sets as members?

Sure!

$$X = \left\{1, 2, \{Jack, Jill\}, \{elm, beech\}\right\}$$

$$Y = \{\Phi, 1,2\}$$

Is $\{\Phi\}$ different from Φ?

Yes, $\{\Phi\}$ contains one member (the set Φ), but Φ contains nothing!

How many members does $\{\{\Phi\}\}$ contain?

One, its only member is the set $\{\Phi\}$.

The set $\left\{\Phi, \{\Phi\}, \{Jack, Jill\}, \left\{a, \{b, c\}\right\}\right\}$ contains 4 elements.

Can a set contain itself as a member?

Let's see what happens if we allow that.

Now consider all the sets that don't contain themselves:

$$S = \{X : X \notin X\}$$

Is $S \in S$? Or is $S \notin S$?

$$(\boldsymbol{S} \in \boldsymbol{S}) \Leftrightarrow (\boldsymbol{S} \notin \boldsymbol{S})\ !$$

Defining sets precisely is extremely tricky!

We'll just agree that sets cannot contain themselves.

If A contains B then B cannot contain A.

Subsets

$A \subseteq B$ means that every member of A is also a member of B

or, $\forall x$: $(x \in A \Rightarrow x \in B)$

$A \subset B$ means that every member of A is a member of B, and B has members that are not members of A

or, $\forall x$: $(x \in A \Rightarrow x \in B) \wedge (\exists x$: $x \in B \wedge x \notin A)$

Set Notation

$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$: sets of natural numbers, integers, rationals, real numbers

Sets can be represented by:

- Listing elements in the set {1, 2, 3}
- By a predicate that describes properties of elements (Set builder notation)

$$\{x: P(x)\}$$

$$\{x \in \mathbb{N} : \exists y \in \mathbb{N}, x = 2y\}$$

This is the set of even numbers.