Programming Paradigms

CT331 Week 7 Lecture 2

Finlay Smith

Finlay.smith@universityofgalway.ie

Example recursive function: Factorial

The factorial of a non-negative integer n, written n!, is the product of all integers less than or equal to n. The factorial of 0 is 1

Factorial

1. What is the base case? 0
2. What should the answer be when we are at the base case?
3. How do you reduce to get to this base case ?
4. What other work needs to be done for each function call?
5. (e.g., creating a new list, etc.)?
6. How can these steps be put together?

Factorial

1. What is the base case? 0 or 1?
2. What should the answer be when we are at the base case? The factorial of 0 is 1
3. How do you reduce to get to this base case ?
4. What other work needs to be done for each function call?
5. (e.g., creating a new list, etc.)?
6. How can these steps be put together?

Factorial

(fact 0) => 1

(fact 1) => 1 * (fact 0)

(fact 2) => 2 * 1 * (fact 0)

Factorial

(fact 0) => 1

(fact 1) => 1 * (fact 0)

(fact 2) => 2 * 1 * (fact 0)

This is the same as (fact 1)

Factorial

(fact 0) => 1

(fact 1) => 1

(fact 2) => 2 * 1

(fact 3) => 3 * 2 * 1

This is the same as (fact 2)

(fact 4) => 4 * 3 * 2 * 1

This is the same as (fact 3)

Factorial

(fact 0) => 1

(fact 1) => 1

(fact 2) => 2 * 1

(fact 3) => 3 * 2 * 1

This is the same as (fact 2)

(fact 4) => 4 * 3 * 2 * 1

This is the same as (fact 3)

(fact 3) => 3 * (fact 2)

(fact 4) => 4 * (fact 3)

Factorial

1. What is the base case? 0
2. What should the answer be when we are at the base case? The factorial of 0 is 1
3. How do you reduce to get to this base case ? n * (n-1) * (n-2) * (n-3) ...
4. What other work needs to be done for each function call?
5. (e.g., creating a new list, etc.)?
6. How can these steps be put together?

Factorial

1. What is the base case? 0
2. What should the answer be when we are at the base case? The factorial of 0 is 1
3. How do you reduce to get to this base case ? n * (n-1) * (n-2) * (n-3) ...
4. What other work needs to be done for each function call?
5. Subtract 1 from n, multiply result
6. How can these steps be put together? Simpler example first

Example recursive function: sum

Write a scheme function that takes a number n and returns the sum of the numbers up to n.

Example `(sum 4)` *will return 10*

Recursive sum function

```
(define (sum num)
        (if (> num 1)
                (+ num (sum (- num 1)))
                1))
```

Recursive factorial function

```
(define (factorial num)
        (if (> num 1)
                (* num (factorial (- num 1)))
                1))
```

Very similar to sum

Example recursive function: all atom?

Write a recursive function `all_atoms?` which checks if all elements in a list are symbols returning #t or #f, e.g.,

```
> (all_atoms? '(x y z))
#t
> (all_atoms? '())
 #t
> (all_atoms? '(a (b) c))
#f
```

Hint:
you will need to make use of the built-in predicate atom?

Other functions/predicates needed: define cond null? car cdr not

Example recursive function: all atom?

Base cases:

- (null? lst) stop and output #t
- (car lst) not an atom, stop and output #f
- One element in list and it is a atom, stop and output #t

Reduce:

- if (car lst) is a atom, continue and check (cdr lst).

Work through this problem – we can look at it in tutorial this week

Example recursive function: sequence

Write a (recursive) function `sequence` which is passed a number num and creates a list of numbers from 1 to num.

Note: Can write this function only using functions/predicates we have seen:

define
cond
append
list
=
'()

Example recursive function: sequence

Base case: If num is 0?

Reduce: - 1 from num

Work: append num to end of new list

Write the code for this function – again we can look at it in the tutorial

Is it the most efficient solution?

Example recursive function: max

The built-in function `max` returns the maximum of a set of numbers, e.g., `(max 3 5 4 19 7)` returns 19.

Write your own version of a function to find the maximum of a list of numbers, e.g., `(maxlst '(3 5 4 19 7))` returns 19

Example recursive function: max

Base case:

one element in list - it’s the maximum

Reduce:

Keep checking pairs of items in list, removing the smaller one, until we have reduced to a list with one element

Example recursive function: max

Base case:

One element in list?

```
(null? (cdr lst))
```

Return that element

```
(car lst)
```

Example recursive function: max

Recursive case: Checking pairs of elements

If first element bigger than second one

```
(> (car lst) (cadr lst))
```

remove (cadr lst) from list which will be sent to recursive call:

```
(cons (car lst) (cddr lst))
```

If second element is bigger, ignore (car lst) for recursive call.

```
(cdr lst)
```

Example recursive function: max

```
(define   (maxlst lst)
          (cond     [  (null? lst) (display '(empty list)) ]
                                   [  (null? (cdr lst)) (car lst) ]
                                   [  (> (car lst) (cadr lst))
                                                (maxlst (cons (car lst) (cddr lst))) ]
                                   [ else (maxlst (cdr lst))] ))
```