Recursion

Ziyan Maraikar

July 8, 2014

Table of Contents

Inductive definitions

In mathematics we often encounter functions that are defined in terms of themselves. For example the factorial function is defined as,

$$n! = \begin{cases} 1 & n < 2 \\ n \times (n-1)! & \text{otherwise} \end{cases}$$

Recursive functions

Recursive definitions in Ocaml are introduced using *let rec*.

```
let rec fact n =
  if n<2 then 1 else n * fact (n-1)
```

If you forget *rec* you get an “unbound value” error.

Termination

To ensure that recursion *terminates*, a recursive definition must

- ★ have a trivial or *base case* whose value is given explicitly.
- ★ the *inductive case* must be defined in terms of the function applied to a value *closer* to the base case value.

Exercise

$$fib\ n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ fib(n-1) + fib(n-2) & \text{otherwise} \end{cases}$$

Table of Contents

- ★ There are no new rules required for evaluating recursive functions.
- ★ Substitute the (recursive) function definition until the base case is reached.
- ★ Don't forget to keep the intermediate vaules during substitution.

Factorial Evaluation

```
fact 3 ≡ if 3<2 then 1 else 3 ∗ fact (3−1)
≡ 3 ∗ fact 2
≡ 3 ∗ (if 2<2 then 1 else 2 ∗ fact (2−1))
≡ 3 ∗ 2 ∗ fact 1
≡ 3 ∗ 2 ∗ (if 1<2 then 1 else 1 ∗ fact (1−1))
≡ 3 ∗ 2 ∗ 1
```

Note how we keep the intermediate values $3 * 2 * \ldots$.

Exercise

Show the evaluation of `fib 3` using the following definition.

```
let rec fib n =
  if n=0 then 0
  else if n=1 then 1
  else fib (n-1) + fib (n-2)
```

Table of Contents

Series sum

Write recursive functions to compute the following functions for a value x, up to n terms.

$$e^x = \sum_0^\infty \frac{x^n}{n!}$$

$$\cos x = \sum_0^\infty \frac{(-1)^n}{(2n)!} x^{2n}$$

Note that the the recursion goes “backwards” from the n^{th} term to the first.

Euclid's GCD algorithm

The *greatest common divisor* of two positive integers can be calculated using the following observation

The GCD of x and y is equal to the GCD of y and $x \mod y$. Repeating the procedure until $y = 0$ gives the GCD of the original numbers in x.

Example

```
gcd 12 33
≡ gcd 33 9
≡ gcd 9 6
≡ gcd 6 3
≡ gcd 3 0
≡ 3
```

- ★ Implement Euclid's GCD algorithm.
- ★ Give an argument to show that the algorithm terminates.

Fast exponentiation

- Write a function to calculate x^n, where x and y are positive integers.
- How many multiplications does your algorithm require?
- Can you implement exponentiation using fewer multiplications?

Summary of Concepts

- Conditional expressions
- Recursive function definitions
- Base case and inductive case
- Ensuring termination
- Evaluating recursive functions