Tail recursion and invariants

Ziyan Maraikar

July 1, 2014

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?

Table of Contents

Memory use during recursion

```
fact 4
≡ 4 * fact 3
≡ 4 * 3 * fact 2
≡ 4 * 3 * 2 * fact 1
≡ 4 * 3 * 2 * 1
```

- ★ Observe how the list of intermediate values grows longer as the computation progresses.
- ★ Each intermediate value must be retained until the recursion terminates.
- ★ This corresponds to the stack space used during the recursive evaluation.

Forms of recursion

Do you notice anything different about the way that evaluation of functions **gcd** and **fact** occur?

```
fact 4
≡ 4 * fact 3
≡ 4 * 3 * fact 2
≡ 4 * 3 * 2 * fact 1
≡ 4 * 3 * 2 * 1
```

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

Fibonnacci

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

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

Redundant computations in fib

```
fib 3
≡ if 3=0 then ... else fib 2 + fib 1
≡ fib 2 + fib 1
≡ (if 2=0 then ... else fib 1 + fib 0) + fib 1
≡ fib 1 + fib 0 + fib 1
≡ (if 1=0 then 0 else if 1=1 then 1 else ...) + fib 0 +
   fib 1
≡ 1 + fib 0 + fib 1
≡ 1 + (if 0=0 then 0 else ...) + fib 1
≡ 1 + 0 + fib 1
≡ 1 + 1 + (if 1=0 then 0 else if 1=1 then 1 else ...)
≡ 1 + 0 + 1
```