COL100: Lec 4

Iterative Factorial Alg.

[Via tail-recursion]

$$factorial(n) = fact\_iter(n, 1, 0)$$

$$: \mathbb{N} \times \mathbb{P} \times \mathbb{N} \rightarrow \mathbb{P}$$

where

$$fact\_iter(m, f, c) = \begin{cases} f & \text{if } c = m \\ fact\_iter(m, f*(c+1), c+1) & \text{otherwise} \end{cases}$$

What is that **invariant condition** for factorial function:

- We know

$$0 \leq c \leq m$$

- for any step c

$$f = f_0 * \prod_{i=c_0+1}^{c} i$$

- finally $f_0 * \prod_{i=c_0+1}^{m} i = f * \prod_{i=c+1}^{m} i$

factorial 5)

= fact_iter (5, 1, 0)

= fact_iter (5, 1, 1)

= fact_iter (5, 2, 2)

= fact_iter (5, 6, 3)

= fact_iter (5, 24, 4)

= fact_iter (5, 120, 5)

= 120

Proof of Correctness (to show $fact\_iter(m,f,c) = f * \prod_{c+1}^{m} i$)

Basis : when $m = c$ [induct on $m - c$]

$$fact\_iter(m,f,c) = f * \prod_{i=c+1}^{m} i$$

$$= f * 1 = f$$

I.H. : for some $K = m - c \geq 0$

$$fact\_iter(m,f,c) = f * \prod_{i=c+1}^{m} i$$

I.S : Let $(m-c) = K+1 > 0$

then $fact\_iter(m,f,c) = fact\_iter(m, f*(c+1), c+1)$

$$= f * (c+1) * \prod_{c+2}^{m} i \quad [\text{By I.H.}]$$

$$= f * \prod_{c+1}^{m} i$$

Now we can show

$$fact(n) = fact\_iter(n, 1, 0) = n!$$

$$fact\_iter(n, 1, 0) = 1 * \prod_{i=1}^{n} i$$

$$= n!$$

3 steps for being a good programmer:

1. Avoid recursion
2. Repeat steps 1 and 2
3. Always have an exit condition

More Examples

1. Computing x^n given $x \neq 0$ and $n \geq 0$

Mathematically,

$$x^n = \begin{cases} 1 & \text{if } n = 0 \\ x \cdot x^{n-1} & \text{otherwise} \end{cases}$$

Algorithm : $\mathbb{R} \times \mathbb{N} \longrightarrow \mathbb{R}$

$$\text{power}(x, n) = \begin{cases} 1 & \text{if } n = 0 \\ x \cdot \text{power}(x, n-1) & \text{otherwise} \end{cases}$$

Correctness

- to establish power $(x,n) = x^n$ for all $x \neq 0$ and $n \geq 0$
  - proof similar to the factorial function

Alt. implementation

Mathematically

$$x^n = \begin{cases} 1 & \text{if } n = 0 \\ (x^2)^{n \text{ div } 2} & \text{if } n > 0 \text{ is even} \\ x \cdot (x^2)^{n \text{ div } 2} & \text{if } n > 0 \text{ is odd} \end{cases}$$

Again
- define an algorithm fast_pow (x,n) and show its equivalence to the above mathematical formulation for correctness

Iterative version of power

pow-iter (x, n, f, c)

$$= \begin{cases} f & \text{if } c = n \\ \text{pow-iter}(x, n, f * x, c+1) & \text{otherwise} \end{cases}$$

power (x, n) = pow-iter (x, n, 1, 0)

pow-iter(5, 3, 1, 0) = pow-iter (5, 3, 5, 1)

= pow-iter (5, 3, 5*5, 2)

= pow-iter (5, 3, 5*5*5, 3)

= 125

Alternate imp_

$$\text{fast\_pow\_iter}(x, n, p) = \begin{cases} p & \text{if } n = 0 \\ \text{fast\_pow\_iter}(x^2, n \text{ div } 2, p) & \text{if } n > 0 \text{ and even} \\ \text{fast\_pow\_iter}(x^2, n \text{ div } 2, p \cdot x) & \text{if } n > 0 \text{ and odd} \end{cases}$$

$$\text{fast\_pow}(x, n) = \begin{cases} 1 & \text{if } n = 0 \\ \text{fast\_pow\_iter}(x, n, 1) & \text{if } n > 0 \end{cases}$$

Correctness

$\mathcal{L}1$: $\forall\ x, p > 0$ and $n : int \geq 0$

$$fast\_pow\_iter\ (x, n, p) = p \cdot x^n$$

Proof: induct on n

Basis: $n = 0$

$$fast\_pow\_iter\ (x, 0, p) = p$$
$$= p \cdot x^0$$

IH: $\forall\ x : real,\ p : real > 0$
$\wedge\ \kappa : int,\ 0 \leq \kappa < n$
$fast\_pow\_iter\ (x, \kappa, p) = p \cdot x^\kappa$

I.S: Case 1 Assume $n > 0$ and n is even

i.e. $n = 2j > 0$

fast_pow_iter $(x, 2j, p)$

$= $ fast_pow_iter (x^2, j, p)

$\therefore$ $2j \text{ div } 2 = j$ [and $j < n$]

by I.H. we know

fast_pow_iter (x^2, j, p)

$= p \cdot x^{2 \cdot j} = p \cdot x^n$

Similarly for the odd case.

final correctness

$$fastpow(x,n) = fast\_pow\_iter(x,n,1)$$

Case $n=0$:

$$L.H.S = 1 = RHS = x^0$$

Case $n>0$:

By definition & $\lambda 1$ we have

$$fastpow(x,n) = fast\_pow\_iter(x,n,1)$$

$$= 1 \circ x^n = x^n$$

Assign1: Q4

int_sq_root (n)

Mathematically

Let k be $\lfloor \sqrt{n} \rfloor$

then

$$k^2 \leq n < k+1$$

Let $m = n \text{ div } 4$ [i.e. $n \div 4 = m * 4 + n \bmod 4$]

Let $i = \lfloor m \rfloor$

then $i^2 \leq m < (i+1)^2$

$\Rightarrow \quad m+1 \leq (i+1)^2$

$i^2 \leq m < (i+1)^2$

$\Rightarrow \quad m+1 \leq (i+1)^2$

$(2i)^2 \leq 4m < (2i+2)^2$

But we know that

$4m \leq n$

$n = 4m + \underbrace{n \bmod 4}_{\in \{0,1,2,3\}}$

$\Downarrow$

So

$(2i)^2 \leq 4m \leq n < 4m+4 \leq (2i+2)^2$

$\therefore \lfloor n \rfloor$ is either $2i$ or $2i+1$

Eg: $n = 56$

$\downarrow$

$ndiv4 = 14$

$i = \lfloor \sqrt{14} \rfloor \longrightarrow$ 2 · int_sqrt(14), 2 · int_sqrt(14) +1

$\downarrow$

$n=14,\ ndiv4 = 3$

$\downarrow$

$i = \lfloor 3 \rfloor \longrightarrow$ 2 · int_sqrt(1), 2 · int_sqrt(1) +1

$\downarrow$

....

SML Tutorial

Boolean Conditions in SML

- (Sml) not = ¬

  Eg: val not (5=5);

- (Sml) p andalso q = $p \wedge q$ ≡ if p then q else false

  Eg: - val n=10;
  - (n>=10) andalso (n=10);

- (sml) p orelse q = $p \vee q$ ≡ if p then true else q

Conditions

if $Cond_1$ then $Expr_1$ else $Expr_2$

$Expr_1$ → if $cond_2$ then $Expr_3$ else $expr_4$

$Expr_2$ → if $cond_3$ then $Expr_5$ else $expr_6$

Structuring

if Cond
then Expr
Else Expr'

Eg:

```
fun leap(y) = if y mod 400 =0
              then true
              else if y mod 100 =0
                   then false
                   else y mod 4 =0
```

let Keyword & Scoping rules

Scope :

- The scope of a name begins from its definition and ends where the corresponding scope ends.
- Scopes end with definition of functions
- Scopes end with the keyword end in any let ---- in ---- end

Scope Rules

- Scopes may be disjoint → a scope can't span two disjoint scopes
- Scopes may be nested
  - → no partial overlaps allowed

Eg:

```
fun sum_of_squares (x, y)
  = let fun sq(a) = a*a;
    in
    sq(x) + sq(y)
    End;
```

Scope of x

Scope of a?