Computability Assignment
Year 2012/13 - Number 5

Please keep this file anonymous: do not write your name inside this file.

More information about assignments at http://disi.unitn.it/∼zunino/teaching/computability/assignments

Please do not submit a file containing only the answers; edit this file, instead, filling the answer sections.

1 Question

(I am re-proposing this exercise since only a few students solved it. This exercise is rather important, since it involves a reasoning which frequently appeared in past exam questions. While we shall see more examples of these concepts in class, it would be useful to start exercising on that. If you have already sumbitted an answer, skip this and do *not* resubmit your answer please.)

Let $\mathcal{F}$ be the set of partial functions $\{f \in (\mathbb{N} \rightsquigarrow \mathbb{N}) | \forall x \in \mathbb{N}.\ f(2 \cdot x) = x\}$.

- Define two distinct partial functions f_1, f_2 which belong to $\mathcal{F}$. (I.e, provide two such examples.)
- Define two distinct partial functions g_1, g_2 which do *not* belong to $\mathcal{F}$. (I.e, provide two such examples.)
- Define a partial function $f \in \mathcal{F}$, and consider the set of its *finite* restrictions $\mathcal{G} = \{g \in (\mathbb{N} \rightsquigarrow \mathbb{N}) | g \subseteq f \ \wedge\ \mathsf{dom}(g) \text{ finite}\}$.
  - Define two distinct partial functions h_1, h_2 which belong to $\mathcal{G}$. (I.e, provide two such examples.)
  - Prove whether $\mathcal{F} \cap \mathcal{G} = \emptyset$.

1.1 Answer

...

I'm sorry, I've already done it.

2 Question

Consider the following function:

$$f(n) = \sum_{i=0}^{n} i^2 + i$$

Write a FOR loop implementing f, then translate it in the λ-calculus as program F_1.

Then, write a recursive Java-like function implementing f, and translate it in the λ-calculus as program F_2.

2.1 Answer

...

The function can be calculated with the loop:

```
[c++-like]
uint64_t loopFun () {
    uint64_t sum = 0;
    for (uint64_t i = 0; i <= n; ++i)
        sum += i*i+i;
    return sum;
}
```

...

The function can be rewritten in λ-calculus as:
$F_1 = \lambda n.Scnd(n(\lambda p.Cons(SuccFirst(p))(AddScnd(p)(AddFirst(p)(MulFirst(p)First(p)))))Cons(\ulcorner 0 \urcorner \ulcorner 0 \urcorner)$
(RZ: you should loop n+1 times. Watch out for parentheses.)
Brief verbose and uninformal explanation:

- Start with a couple $(0, 0)$, at each step increment the first as an index and accumulate in the second the result of $sum+ = i \times i + i$.
- The n is used to repeatedly apply the body of the inner function to the result of the previous application.
- At the end, the value of the sum is returned (second argument).

...

The recursive implementation may be:

```
[Java-like]
int recFun(int n) {
    return n*n+n+(n > 0 ? recFun(n - 1) : 0);
}
```

...

A recursive implementation can be:
$G = \lambda gn.IsZero\, n\, \ulcorner 0 \urcorner\, (Add\, (Mul\, n\, n)\, (Add\, n\, (g\, g\, Pred(n))))$
$F_2 = GG$
Brief verbose and uninformal explanation:

- Start with value n, compute $n \times n + n$ and then perform a recursive call over $n - 1$.
- Continue through this decreasing chain of values until $n = 0$ which returns 0. [termination condition guaranteed!]
- Sum up all the values, return them.

3 Question

Consider the following function:

$$f(n) = \begin{cases} x^2 + y & \text{if } n = \mathsf{pair}(\mathsf{inL}(x), y) \\ x + 4 \cdot y & \text{if } n = \mathsf{pair}(\mathsf{inR}(x), y) \end{cases}$$

Convince yourself that f is defined for all naturals n, i.e. it is total.

Write a λ-term implementing function f, exploiting the programs $Pair, Proj1, Proj2, InL, InR, Case, \ldots$ we saw in class (also defined in the notes).

3.1 Answer

...

TRIVIA: $pair(n, m) = [(n + m) \times (n + m + 1)/2] + n$
TRIVIA: $inL(n) = 2 \times n$
TRIVIA: $inR(n) = 2 \times n + 1$
TRIVIA: $Case = \lambda nlr.Even(n)\, (l\, (Div\, n\, \ulcorner 2 \urcorner))\, (r\, (Div\, n\, \ulcorner 2 \urcorner))$
SELF-CONVINCING: Obvious.
...
Let's define two help functions:
$G_1 = \lambda xy.Add\, (Mul\, x\, x)\, y$
$G_2 = \lambda xy.Add\, (Mul\, \ulcorner 4 \urcorner\, y)\, x$
Then we need the inverse of function pair that maps a natural number k into the couple of values $n, m \in \mathbb{N}$ such that $pair(n, m) = k$. Luckily this function is already defined by means of a couple of projecting functions *[page 122, notes]*. Therefore function f can be expressed as:
$F = \lambda n.(Case\, Proj_1(n)\, G_1\, G_2)\, Proj_2(n)$
(RZ: ok, it's subtle in that G1,2 take an extra argument, but it looks correct. Minor thing: write $(Proj_2\ n)$ instead of $Proj_2(n)$)
Brief verbose and uninformal explanation:

- The first programs G_1, G_2 are straightforward.

- The latter definition is pretty intuitive, once one gets ridden of the belief that an equality test over n has to be done. In fact, given the nature of function *pair*, it is sufficient to test the $proj_1(n)$, using the *Case* program. If it's found even, then *Case* takes care of calculating $x = n/2$ and pass it to the function G_1. The other case is similar.
- Finally, the second parameter is given to the function, so that a result can be returned.

V'Ger