SICP: Exercise 2.5

September 6, 2024

Show that we can represent pairs of nonnegative integers using only numbers and arithmetic operations if we represent the pair a and b as the integer that is the product $2^a 3^b$. Give the corresponding definitions of the procedures `cons`, `car`, *and* `cdr`.

To show this, we need to show that, given $x = 2^a 3^b$, we can extract both a and b again using only arithmetic operations (and without using either a or b, and assuming $a, b \geq 0$).

A basic algorithmic idea I have is that we could repeatedly divide x by 2 (for a) until it's not divisible by 2 anymore, which indicates that the a component is now gone, at which point we can just apply the corresponding logarithm. However, this is very inefficient as a way of storing pairs, leading to $\mathcal{O}(a)$, and I'm wondering if there's a clever mathematical way of doing this instead of just dividing repeatedly. Hm, I've thought for some time now, but can't think of anything. Maybe modular arithmetic can be used here by some means, but I don't want to put too much time into this.

For now, let's just prove that, if we just divide x by 2 until the number is no longer divisible by 2, the number of times we've divided x must be equal to a:

$$\begin{aligned}
(2^a 3^b) \cdot 2^{-i} &\equiv 1 \pmod 2 \\
\Leftrightarrow 2^{a-i} 3^b &\equiv 1 \pmod 2 \\
\Leftrightarrow (2^{a-i} \bmod 2) \cdot (3^b \bmod 2) &\equiv 1 \pmod 2 \\
\Leftrightarrow (2^{a-i} \bmod 2) \cdot (3 \bmod 2) &\equiv 1 \pmod 2 && \text{Fermat's little theorem} \\
\Leftrightarrow (2^{a-i} \bmod 2) \cdot 1 &\equiv 1 \pmod 2 \\
\Leftrightarrow 2^{a-i} &\equiv 1 \pmod 2
\end{aligned}$$

In other words, 2^{a-i} must not be divisible by 2. The only solution for this where a and i are non-negative is $i = a$, since $2^0 = 1$. Now that we've shown this, we can hence retrieve a by dividing until the result is congruent to 1 modulo 2. Once we have a, retrieving b becomes very simple, we just divide by 2^a and take the ternary logarithm:

$$\log_3\left(\frac{x}{2^a}\right) = \log_3\left(\frac{2^a 3^b}{2^a}\right) = \log_3(3^b) = b$$

Implementing this in Scheme (or rather, Racket), we get:

```
(define (cons a b)
  (* (expt 2 a) (expt 3 b)))

(define (car x)
  (define (car-iter r i)
    (if (= (modulo (/ x r) 2) 1)
        i
        (car-iter (* r 2) (inc i))))
  (car-iter 1 0))

(define (cdr x)
  (log (/ x (expt 2 (car x))) 3))
```