Exercise 1.11.

A function f is defined by the rule that $f(n) = n$ if $n < 3$ and $f(n) = f(n-1) + 2f(n-2) + 3f(n-3)$ if $n > 3$. Write a procedure that computes f by means of a recursive process. Write a procedure that computes f by means of an iterative process.

Answer. We can immediately translate this definition into a recursive procedure for computing function of f:

```
(define (f n)
  (if (< n 3)
      n
      (+ (f (- n 1))
         (* 2 (f (- n 2)))
         (* 3 (f (- n 3))))))
```

On the other hand, by tracking the process generated while computing `(f 5)` in the recursive procedure above, one might easily figure out that the ‹*alternative*› of `if` conditional split into three at each level (except at the bottom), as is shown in Figure 1. This general pattern strongly offers us an

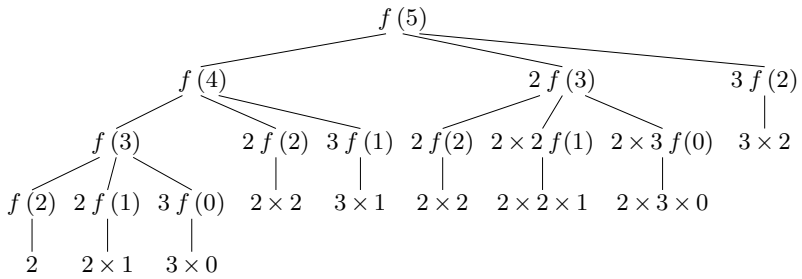

Figure 1.

intuition for expressing the process with a triple of integers a, b and c on every level of computation. More precisely, we can start with an initialization where $f(2) = 2$, $f(1) = 1$ and $f(0) = 0$, and repeatedly apply the simultaneous transformations

$$\begin{aligned} a &\leftarrow a + 2b + 3c \\ b &\leftarrow a \\ c &\leftarrow b \end{aligned}$$

It is not hard to show that, after applying this transformation n times, a, b and c will be equal, respectively, to $f(n+2)$, $f(n+1)$ and $f(n)$. Thus, we can compute $f(n)$ iteratively using the procedure

```
(define (f n)
  (define (f-iter a b c count)
    (if (= count 0)
        c
        (f-iter (+ a (* 2 b) (* 3 c))
                a
                b
                (- count 1))))
  (f-iter 2 1 0 n))
```