SICP Exercise 3.27

Yuchong Pan

September 16, 2016

Initially, the environment structure created by **(define (memoize f) ...)** and by **(define memo-fib ...)**, where **table** is empty, is given as follows.

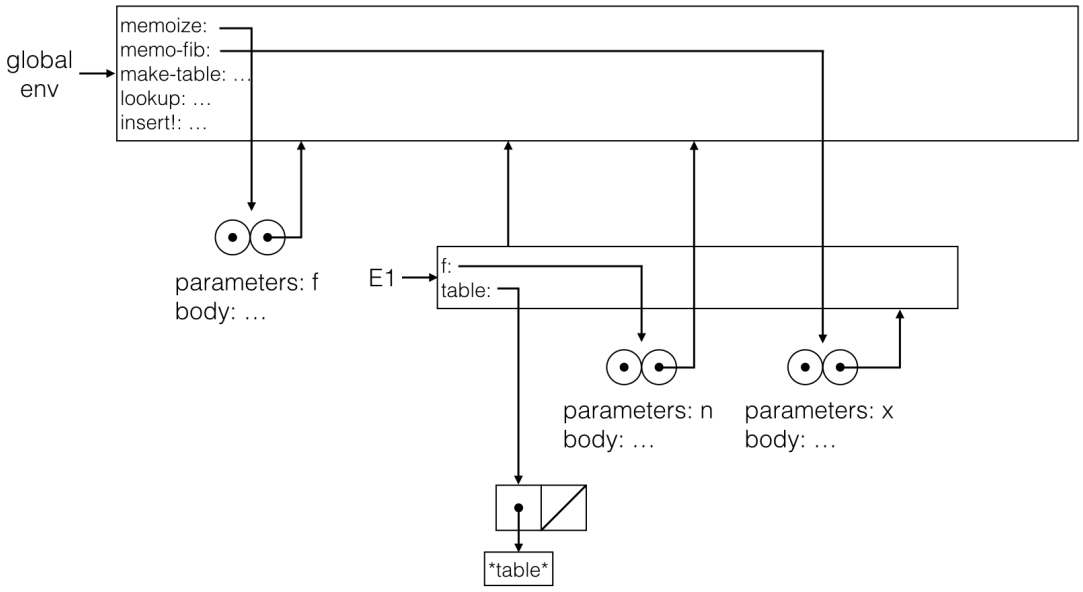

Figure 1: Effect of **(define (memoize f) ...)** and of **(define memo-fib ...)**.

When **(memo-fib 3)** is called, a new environment is created, where **previously-computed-result** is computed to be **false** by **(lookup 3 table)**. Thus, **(f 3)** is called, and **(memo-fib 2)** and **(memo-fib 1)** is subsequently required.

Then, **(memo-fib 2)** is called. Similarly, **previously-computed-result** is computed to be **false**, and hence, **(f 2)** is called. The interpreter will subsequently compute **(memo-fib 1)** and **(memo-fib 0)**.

Thereafter, **(memo-fib 1)** is called, and **previously-computed-result** is still **false**, so **(f 1)** is called, where **(= n 1)** is a boundary condition and gives 1. Therefore, the record

(1, 1) is added to **table**.

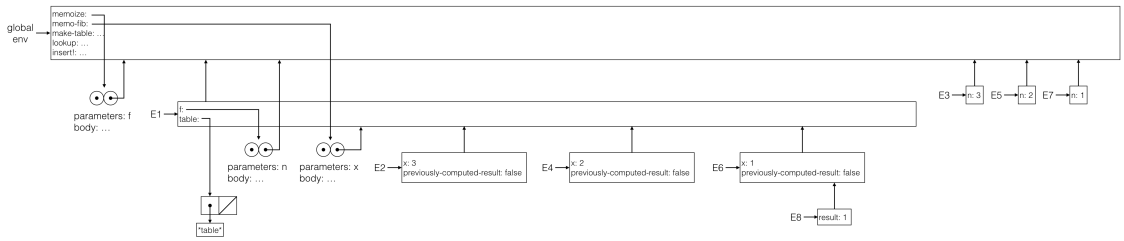

Figure 2: Effect of **(memo-fib 3)**.

Likewise, **(memo-fib 0)** is called, where **previously-computed-result** is also **false**, so **(f 0)** is called, where **(= n 0)** is a boundary condition and gives 0. Therefore, the record **(0, 0)** is added to **table**.

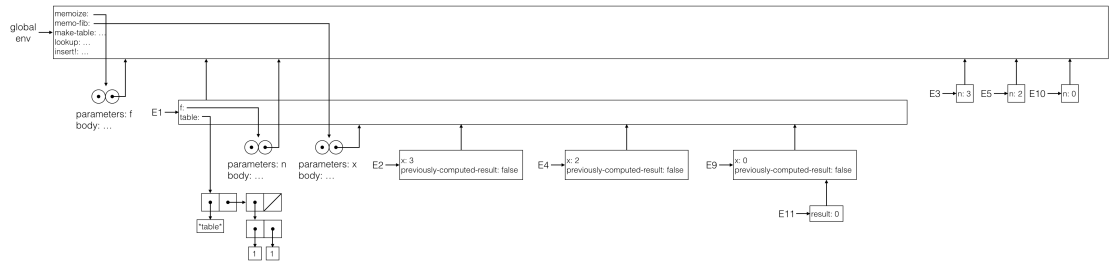

Figure 3: Effect of **(memo-fib 3)**.

Having **(memo-fib 1)** and **(memo-fib 0)** computed, the result of **(memo-fib 2)** is the sum of the two computed results, i.e. 1. Thus, the record **(2, 1)** is added to **table**.

As mentioned above, **(f 3)** also requires **(memo-fib 1)**. At this point, the value 1 is searched in **table** with the key 1. Hence, the result of **(memo-fib 1)** is 1, and the result of **(f 3)** is the sum of **(memo-fib 2)** and **(memo-fib 1)**, which gives 2. The record **(3, 2)** is added to **table**. Therefore, the computation of **(memo-fib 3)** gives 2.

Since **table** is maintain in which values of previous calls are stored using as keys the arguments that produces the values, **(memo-fib** i**)** where $1 \leq i \leq n$ will be called at most once when **(memo-fib** n**)** computes the nth Fibonacci number. Therefore, the number of steps required by the computation of **(memo-fib** n**)** is proportional to n; i.e., the number of steps grows as $\Theta(n)$.

If **memo-fib** is simply defined to be **(memoize fib)**, the scheme would not work. If the key n is not found in **table** when **(memo-fib** n**)** is called and when $n > 1$, then the interpreter will subsequently call **(fib (- n 1))** and **(fib (- n 2))**, which cannot use

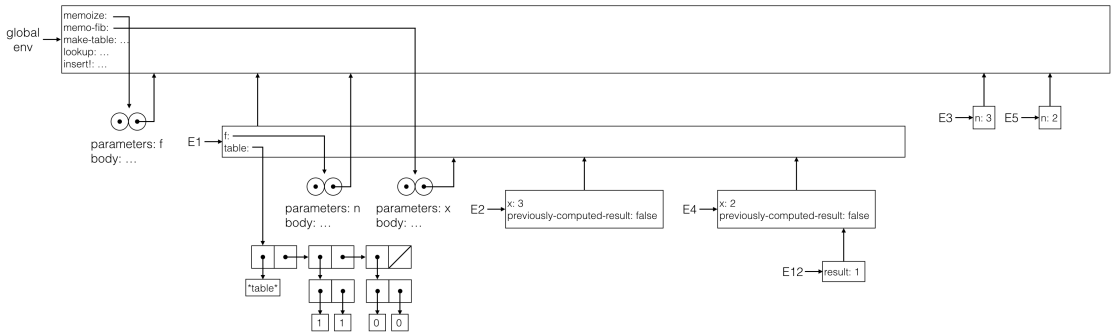

Figure 4: Effect of **(memo-fib 3)**.

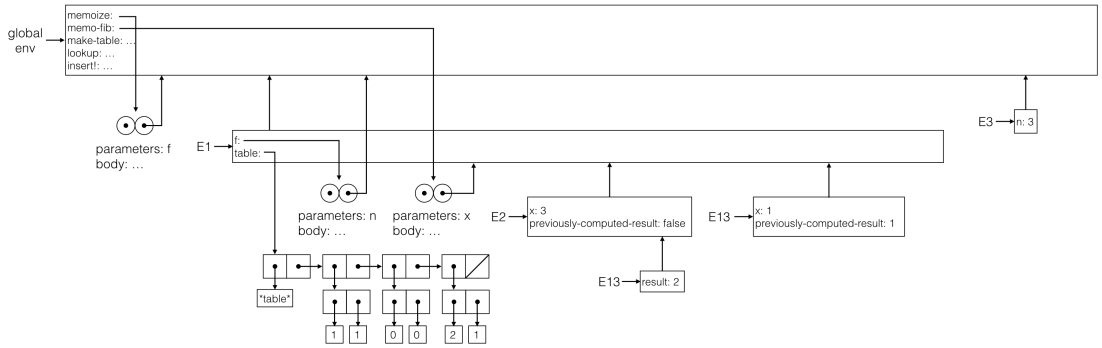

Figure 5: Effect of **(memo-fib 3)**.

memoization any longer and will perform as the original **fib**. Thus, the number of steps required by **(memo-fib** n**)** will grows as $\Theta(Fib(n))$ rather than $\Theta(n)$.

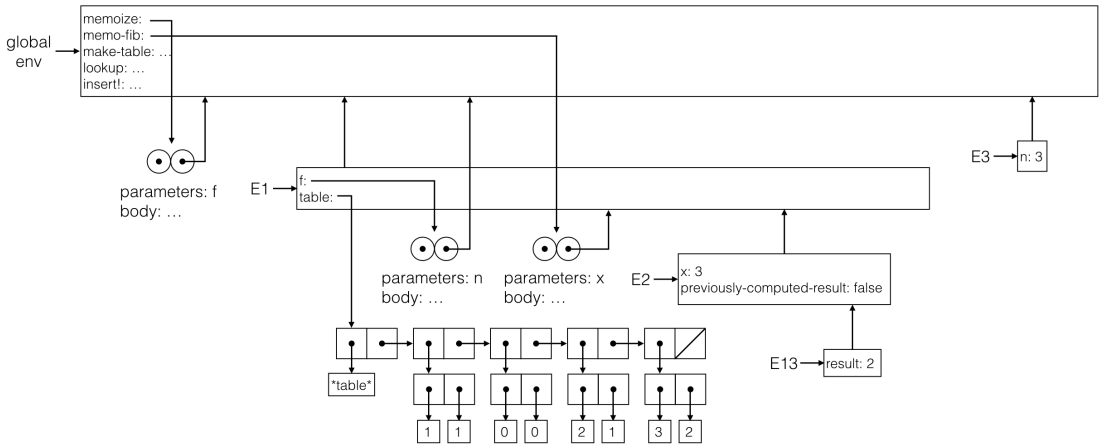

Figure 6: Effect of **(memo-fib 3)**.