Homework 6: Fixed Point Combinator

Gabe Giancarlo

October 5, 2025

1 Problem

Compute fact 3 using the fixed point combinator:

$$\text{let rec fact } = \lambda n.\ \text{if } n = 0 \text{ then } 1 \text{ else } n * \text{fact}(n-1) \text{ in fact } 3$$

2 Solution

Let $F = \lambda f.\lambda n.\ \text{if } n = 0 \text{ then } 1 \text{ else } n * f(n-1)$ and $G = \text{fix } F$.

$$\text{let rec fact } = \lambda n.\ \text{if } n = 0 \text{ then } 1 \text{ else } n * \text{fact}(n-1) \text{ in fact } 3 \tag{1}$$
$$\rightsquigarrow \text{let fact } = (\text{fix } F) \text{ in fact } 3 \quad \text{(def of let rec)} \tag{2}$$
$$\rightsquigarrow (\lambda \text{fact.fact } 3)(\text{fix } F) \quad \text{(def of let)} \tag{3}$$
$$\rightsquigarrow G\ 3 \quad \text{(beta rule)} \tag{4}$$
$$\rightsquigarrow F(\text{fix } F)\ 3 \quad \text{(def of fix)} \tag{5}$$
$$\rightsquigarrow (\lambda n.\ \text{if } n = 0 \text{ then } 1 \text{ else } n * G(n-1))\ 3 \quad \text{(beta rule)} \tag{6}$$
$$\rightsquigarrow \text{ if } 3 = 0 \text{ then } 1 \text{ else } 3 * G(3-1) \quad \text{(beta rule)} \tag{7}$$
$$\rightsquigarrow 3 * G(2) \quad \text{(def of if, arithmetic)} \tag{8}$$
$$\rightsquigarrow 3 * F(\text{fix } F)(2) \quad \text{(def of fix)} \tag{9}$$
$$\rightsquigarrow 3 * (\lambda n.\ \text{if } n = 0 \text{ then } 1 \text{ else } n * G(n-1))\ 2 \quad \text{(beta rule)} \tag{10}$$
$$\rightsquigarrow 3 * (\text{ if } 2 = 0 \text{ then } 1 \text{ else } 2 * G(2-1)) \quad \text{(beta rule)} \tag{11}$$
$$\rightsquigarrow 3 * (2 * G(1)) \quad \text{(def of if, arithmetic)} \tag{12}$$
$$\rightsquigarrow 3 * (2 * F(\text{fix } F)(1)) \quad \text{(def of fix)} \tag{13}$$
$$\rightsquigarrow 3 * (2 * (\lambda n.\ \text{if } n = 0 \text{ then } 1 \text{ else } n * G(n-1))\ 1) \quad \text{(beta rule)} \tag{14}$$
$$\rightsquigarrow 3 * (2 * (\text{ if } 1 = 0 \text{ then } 1 \text{ else } 1 * G(1-1))) \quad \text{(beta rule)} \tag{15}$$
$$\rightsquigarrow 3 * (2 * (1 * G(0))) \quad \text{(def of if, arithmetic)} \tag{16}$$
$$\rightsquigarrow 3 * (2 * (1 * F(\text{fix } F)(0))) \quad \text{(def of fix)} \tag{17}$$
$$\rightsquigarrow 3 * (2 * (1 * (\lambda n.\ \text{if } n = 0 \text{ then } 1 \text{ else } n * G(n-1))\ 0)) \quad \text{(beta rule)} \tag{18}$$
$$\rightsquigarrow 3 * (2 * (1 * (\text{ if } 0 = 0 \text{ then } 1 \text{ else } 0 * G(0-1)))) \quad \text{(beta rule)} \tag{19}$$
$$\rightsquigarrow 3 * (2 * (1 * 1)) \quad \text{(def of if)} \tag{20}$$
$$\rightsquigarrow 6 \quad \text{(arithmetic)} \tag{21}$$

3 Step-by-Step Explanation

1. **let rec expansion:** Convert recursive definition to use fixed point combinator
2. **let expansion:** Convert let binding to function application
3. **fix application:** Apply fixed point combinator to create recursive function
4. **Recursive evaluation:** Function evaluates recursively, expanding $\text{fact}(n)$ to $F(\text{fix } F)(n)$
5. **Base case:** When $n = 0$, conditional returns 1, terminating recursion
6. **Arithmetic:** Final result is $3 \times 2 \times 1 \times 1 = 6$

The fixed point combinator enables recursion in λ-calculus by providing a way to define self-referential functions.