1 Fixed-point combinators

1. The most popular encoding of recursion in \(\lambda\)-calculus is using the Y combinator. Another approach is using the simple U combinator (which is not a fixed-point combinator):

\[
U = \text{df} \lambda x. x x
\]

(We use =\text{df} to denote a definition, as opposed to semantic equivalence.) Using the U combinator, we can define the factorial function as follows:

\[
\text{fact} = \text{df} U (\lambda f. \lambda n. \text{if } (n = 0) \text{ then } 1 \text{ else } n \ast ((f f)(n - 1)))
\]

(For clarity, we extend pure lambda calculus with conditionals and arithmetic.) Prove that \(\text{fact}\) satisfies the following \(\lambda\)-calculus equation:

\[
\text{fact} = (\lambda n. \text{if } (n = 0) \text{ then } 1 \text{ else } n \ast (\text{fact}(n - 1)))
\]

2. Consider the following theorem for characterizing fixed-point combinators themselves as fixed points:

Let \(G = \lambda y. \lambda f. f(y f)\). Then \(M\) is a fixed-point combinator if and only if \(M = GM\).

(1)

(Note: Recall that the following \(\lambda\)-calculus transformation is called the \(\eta\)-reduction rule:

\[
(\lambda x. M x) \rightarrow^{\eta} M,
\]

where \(x\) does not occur as one of the free variables of \(M\). You are allowed to use \(\eta\)-reduction in this question.)

**subpart (i)**
Use Theorem 1 to show that $Y = \text{df } \lambda f.((\lambda x.f(xx))(\lambda x.f(xx)))$ is a fixed-point combinator.

**subpart (ii)**

Use Theorem 1 to show that the U combinator $(\lambda x. x x)$ is not a fixed-point combinator.

**subpart (iii)**

Prove Theorem 1. (Note that the theorem involves an “if and only if”; consequently, your proof should have two parts.)

## 2 Hoare Logic

In this question, all variables hold real-world integers—i.e., don’t worry about machine arithmetic and overflows.

1. Using Hoare logic, give a proof of the following Hoare triple:

   $$\{ x \geq 0 \land y > 0 \}$$

   $$r = x;$$
   $$q = 0;$$
   $$\text{while } (r \geq y) \{}$$
   $$r = r - y;$$
   $$q = q + 1;$$
   $$\text{}\}$$

   $$\{ x = y \ast q + r \land 0 \leq r < y \}$$

   You need only provide an annotation of the form $\{P\}$ for every location in the program; you do not need to show a derivation tree. Accompany your answer with an English description.

2. Using Hoare logic, give a proof that the following sequence of statements swaps the values of $x$ and $y$.

   $$x = x + y;$$
   $$y = x - y;$$
   $$x = x - y;$$

   You may use auxiliary variables to denote the initial/final values of $x$ and $y$. Again, you need only supply annotations of the form $\{P\}$ for every location along the program. Accompany your answer with an English description.
3 Transition Relations

Consider the following program where all variables are mathematical integers.

\begin{verbatim}
x = 10;
y = 10;
while (x + y > 0)
  x = x - 1;
y = y - 1;
z = x + y;
assert (z == 0)
\end{verbatim}

Part 1 Prove that the assertion always holds (i.e., can never be violated) by giving an inductive loop invariant that is strong enough to imply the assertion. Show that your answer is indeed an inductive loop invariant.

Part 2 In class, we encoded a program as a transition relation $trans(\vec{x}, \vec{x}')$, along with a precondition $pre(\vec{x})$ and a postcondition $post(\vec{x})$. Give an encoding of the above program in this form. Argue that your encoding is indeed faithful.

Part 3 Suppose you were to use Cartesian predicate abstraction (as presented in class) to prove this program correct. Give a sufficient set of predicates that will enable constructing a safe inductive loop invariant. Demonstrate why your set of predicates is sufficient.