INF214 - Assignment 2

Viet Hoang Nguyen

October 16, 2018

Question 2: Greatest Common Divisor

```
int gcd(int n, int m) {
    A<int> x = n;
    A<int> y = m;
    while (x != y) {
        processes ps;
        ps += [&] { ATO if (x > y) x = x - y; MIC; };
        ps += [&] { ATO if (y > x) y = y - x; MIC; };
    }
    return x;
}
```

Weakly Fair Scheduler

With a weakly fair scheduler, we are guaranteed that a process will run if the condition it is waiting for becomes true and stays true. In our greatest common divisor program, we see that in each iteration there is at least one condition becoming true.

- If $x == y$, the program will never enter the loop and just terminates.
- If $x > y$, the program will execute the first thread, decreasing n.
- If $x < y$, the program will execute the second thread, decreasing m.

In other words, only one of the threads will execute for each iteration. We however still have to prove that the program will jump out of the loop in order to terminate. The program terminates if the two numbers are equal because it will never enter the loop. If the two numbers are not equal, one is greater than the other, and we end up subtracting the smaller number from the bigger number. This in turn ensures us that we will never get a negative number. The numbers will keep decreasing getting closer 1, because we know that one of the threads will always execute each iteration. At some point, the program hits the base case which means the termination of the program. This is because the program can only stay in the loop as long the numbers are positive.

$$x > y \iff x - y > 0$$

$$y < x \iff y - x > 0$$

Correctness Rules

The correctness of our program can be proven through the following Hoare triplet

$$\{x > 0 \land y > 0\}\ \ while(x \neq y)\,\{\ \ldots\ \}\ \ \{x = gcd(n,m)\}$$

From the properties of the greatest common divisor, we know the following:

$$x > y \implies gcd(x,y) = gcd(x-y,y) \tag{1}$$

$$gcd(x,x) = x \tag{2}$$

$$gcd(x,y) = gcd(y,x) \tag{3}$$

We also consider the following invariant, I, which holds for every iteration of the program:

$$gcd(x,y) = gcd(n,m)$$

Lastly, we also consider the following inference rules:

$$\frac{\{P\}\,S_1\,\{R\} \qquad \{R\}\,S_2\,\{Q\}}{\{P\}\,S_1;S_2\,\{Q\}} \quad (4) \qquad \frac{\{P \land B\}\,S\,\{Q\} \qquad P \land \neg B \Rightarrow Q}{\{P\}\ \ if\ (B)\ S;\ \{Q\}} \quad (5)$$

$$\frac{\{I \land B\}\ \ S\ \ \{I\}}{\{I\}\ \ while\ (B)\ S;\ \{I \land \neg B\}} \quad (6) \qquad \frac{\{P\}\,S\,\{Q\} \qquad \{P'\}\,S\,\{Q'\}}{\{P \land P'\}\,S\,\{Q \land Q'\}} \quad (7)$$

Proof

```
int gcd(int n, int m) {
  /* 1. $\{n > 0 \wedge m > 0\}$
        Assuming the input is valid, i.e. $n$ and $m$ are greater than 0.
        */
  A<int> x = n;
  A<int> y = m;
  /* 2. $\{x = n > 0 \wedge y = m > 0\}$
        By using the assignment rule, $x$ and $y$ are now equal to $n$ and $m$ respectively.
        */
  /* 3. $\{x = n \wedge y = m \implies gcd(x, y) = gcd(n, m)\}$
        This is the invariant for the loop, so it holds before the program
        enters the loop. This means the invariant will still hold if the
        program never enters the loop, because nothing changes. If the
        program does enter, we have to make sure the invariant still holds
        after an iteration. Refer to equation 6 for the inference rules of
        the while loop.
        */
  while (x != y) {
    /* 4. $\{x \neq y \implies x > y \oplus y > x\}$
          One of the numbers is greater than the other one if they are not equal.
          */
    processes ps;
    ps += [&] { ATO
      /* 5. $\{I \wedge \neg(x > y) \implies I\}$
            By simplification we can see that nothing has changed. And when
            nothing has changed, we know that the invariant still holds.
            The invariant still holds even if the condition is false.
            Refer to equation 5 for the inference rules of the
            conditional statement.
            */
      if (x > y) {
        // 6. $\{x > y\}$ by equation 5
        x = x - y; // $\{x > y \implies gcd(x, y) = gcd(x - y, y)\}$ by equation 1
        /* Equation 1 tells us that
           $\{x > y \wedge x > y \implies gcd(x, y) = gcd(x - y, y)\}$
           This gives us
           gcd(x,y) = gcd(x-y,y)
           */
        /* 7. $\{gcd(x - y, y) = gcd(x, y) = gcd(n, m)\}$
              The transitive property gives us
              $\{gcd(x - y, y) = gcd(x, y) = gcd(n, m) \implies gcd(x - y, y) = gcd(n, m)\}$
              In other words, the invariant $I$ still holds.
              */
      } MIC;
      /* 8. $\{I\}$
             The invariant $I$ still holds after the conditional statement is complete
             The next conditional statement is more or less the same as the one we just
             looked at, but $x$ and $y$ are swapped.
             */
  };

  /* 9. $gcd(x,y) = gcd(y,x)$
        by the third equation, we can see that there are no consequences from swapping
        our two variables, which means that the same rules apply and our invariant
        also still holds.
        */
  ps += [&] { ATO
    /* 10. $\{I \land \neg(y > x) \implies I\}$
           Same as point 5.
           After simplifying, we see nothing has changed, so invariant still holds.
           Also holds even if the condition is false, because the program never jumps
           into the conditional statement, so nothing changes here either.
           */
    if (y > x) {
      // 11. $\{y > x\}$ by equation 5
      y = y - x; // $y > x \implies gcd(y,x) = gcd(y-x,x)$ by equation 1
      /*Same as point 6.
      $y > x$ must be true if the program enters the conditional statement.
      Even with swapped variables, we know from earlier that the same rules apply,
      and from the first eqution we get:
      $\{y > x \land y > x \implies gcd(y,x) = gcd(y-x,x)\}$
      */
      /* 12. $\{gcd(y-x,y) = gcd(y,x) = gcd(n,m)\}$
             Same as step 7. Invariant $I$ still holds.
             $\{gcd(y-x,x) = gcd(y,x) = gcd(n,m) \implies gcd(y-x,x) = gcd(n,m)\}$
             */
    } MIC;
    /* 13. $\{I\}$
           Same as step 8. The invariant $\{I\}$ still holds
           */
  };

  /* 14. $\{I\}$
         The invariant will hold after each conditional statement, no matter how many.
         This is to ensure that the invariant holds for every iteration.
         */
}
/* 15. The loop has ended, and we know that the invariant was held all the time
       until the loop condition was no longer met.

       $\{gcd(x,y) = gcd(n,m) \land x = y\}$

       By applying the second equation, we get
        gcd(x,x) = x

       Derived from elementary logic, we get
       $setx = y \implies gcd(x,y) = gcd(x,x) = x$

       Combining the fact that $x = y$ and the transitive property, we get
        gcd(n,m) = gcd(x,y) = gcd(x,x) = x
```

```
         Simplify and we have proven the correctness of our program.
          gcd(n,m) = x
         */
  return x;
}
```