Partial Correctness of GCD
Matthew Stone
Review Lecture
April 2004

These notes tie together the course material we've done so far: logic, functions, numbers and induction. We'll be studying the following computer program, called GCD:

procedure gcd(a,b)

$x := a$
$y := b$

while $y \neq 0$
begin
    $r := x \; mod \; y$
    $x := y$
    $y := r$
end

We will assume that all the variables a, b, x and y must take integer values. We also assume that a and b are positive and that $a \geq b$. So our *initial assertion* about this program will be:

$$a \geq b > 0$$

The *final assertion*, meanwhile, just says

$$x = \gcd(a, b)$$

In other words, when the loop terminates, x is the greatest common divisor of a and b. We will show that when the initial statement is true and we run GCD, the final assertion becomes true.

Let's review division and the logical ideas behind the greatest common divisor.

1. If a is a positive integer and b is an integer, we say that *a divides b*, or a is a divisor of b, if there is an integer c such that $ac = b$.

   We indicate that a divides b with the notation $a|b$.

2. A *common divisor* of two integers a and b is an integer d such that $d|a$ and $d|b$.

3. The *greatest common divisor* of two integers a and b is exactly that—of the set of common divisors, it's the largest.

   Formally, $x = \gcd(a, b)$ is equivalent by definition to:

   $$x|a \wedge x|b \wedge \forall d(d|a \wedge d|b \rightarrow d \leq x)$$

   The first two conjuncts of this formula say that x is a common divisor of a and b. The rest of the formula says that any other common divisor is less than or equal to x.

On the basis of Euclid's algorithm we can show something stronger about the greatest common divisor. We can show that (if a and b are positive integers):

$$x = \gcd(a,b) \leftrightarrow x > 0 \wedge x|a \wedge x|b \wedge \forall d(d|a \wedge d|b \rightarrow d|x)$$

In other words, exactly when every divisor of a and b divides their positive common divisor x, x is the greatest common divisor of a and b. For now, all we need is that when x, a and b are all positive integers, if we know

$$x|a \wedge x|b \wedge \forall d(d|a \wedge d|b \rightarrow d|x)$$

(that every common divisor of a and b divides x) then it follows that $x = \gcd(a,b)$.

To prove this, we follow the form of the logical statement of the definition of $\gcd(a,b)$. We consider an arbitrary hypothetical common divisor of a and b. Call it d_0. We need to show that $d_0 \leq x$. By universal instantiation of our main assumption, we know that $d_0|x$. In other words, there is some integer, which we will call c_0 for the purposes of argument, such that $c_0 d_0 = x$. Now since $x > 0$, $c_0 \neq 0$. So $c_0 d_0 \geq d$. Since $c_0 d_0 = x$, that's just another way of saying $x \geq d$. Since d_0 was an arbitrary common divisor of a and b, we can conclude that x is greater than any common divisor of a and b. So $x = \gcd(a,b)$.

The other fact that we will want to remember about division is this. If a, b, s and t are all integers then

$$d|a \wedge d|b \rightarrow d|sa + tb$$

To show this we just unpack the definitions. $d|a$ means $c_0 d = a$ for some unspecified integer we will call c_0. $d|b$ means $c_1 d = b$ for some unspecified integer we will call c_1. That means that $sa + tb = sc_0 d + tc_1 d = (sc_0 + tc_1)d$. Since $sc_0 + tc_1$ is some integer c_2, this means $c_2 d = sa + tb$ so by definition $d|sa + tb$.

This has the further consequence that

$$d|a \wedge d|b \rightarrow d|a \bmod b$$

The reason is that $a \bmod b = 1a - qb$, where q is the quotient of a and b.

On the basis of these definitions and arguments, we now define and the *loop invariant* of the **while** loop of GCD. Each time the **while** loop starts, we need to know the following thing:

$$x > 0 \wedge x \geq y \geq 0 \wedge \forall d(d|a \wedge d|b \rightarrow d|x \wedge d|y)$$

The two conjuncts say that x is positive and no smaller than y which is no smaller than 0. The third conjunct says that only the common divisors of x and y are common divisors of a and b.

The key final element is our assumptions about how programs work. The state of a program is an function mapping values to variables. Executing an assignment changes this function. If e is a mathematical expression in terms of variables and v is the value of e in the current program state, then after running the statement $x := e$, then the value of x is v. This is called the *assignment* clause. If v' is the value of some other variable y before $x := e$, then v' is still the value of y afterwards. This is called the *inertia* clause.

Here then are the key things we will show to be true at the different points in the execution of GCD, given the assumption that $a \geq b > 0$ is the initial assertion:

procedure gcd(a,b)

$(A.)\ a \geq b > 0$

$x := a$
$y := b$

$(B.)\ x = a \wedge y = b \wedge a \geq b > 0$
$(C.)\ x > 0 \wedge x \geq y \geq 0 \wedge \forall d(d|a \wedge d|b \leftrightarrow d|x \wedge d|y)$

while $y \neq 0$
begin
$r := x \bmod y$

$(D.)\ x \geq y > r \geq 0 \wedge \forall d(d|a \wedge d|b \leftrightarrow d|y \wedge d|r)$

$x := y$
$y := r$

$(E. \equiv C.)\ x > 0 \wedge x \geq y \geq 0 \wedge \forall d(d|a \wedge d|b \leftrightarrow d|x \wedge d|y)$

end

$(F.)\ x > y = 0 \wedge \forall d(d|a \wedge d|b \leftrightarrow d|x \wedge d|y)$
$(G.)\ x > 0 \wedge x|a \wedge x|b \wedge \forall d(d|a \wedge d|b \rightarrow d|x)$
$(H.)\ x = \gcd(a,b)$

We organize the proof into an outline. The items follow the structure of the program; the displayed formulas show how we gradually establish progressively more powerful mathematical statements about the program.

1. First we show
$$A\{x := a; y := b\}B$$
This is an obvious consequence of assignment and inertia.

2. Then we show
$$B \rightarrow C$$
by logic. B lets us substitute a for x and b for y in C, which makes the inequalities in C follow immediately from B and gives the universal statement the form $\forall d(p \leftrightarrow p)$. We conclude that
$$A\{x := a; y := b\}C$$
by putting these two things together.

3. We next show
$$C \wedge y \neq 0\{r := x \bmod y\}D$$
This step mixes logical and mathematical reasoning. The key thing is that after we execute $r := x \bmod y$, $r = x \bmod y$, but x and y are unchanged. That means we still have $x \geq y > 0$. It also means $y > r \geq 0$, by the definition of *mod*. So to show D is true, we only need to show $\forall d(d|a \wedge d|b \leftrightarrow d|y \wedge d|r)$. So consider d_0 such that $d_0|a \wedge d_0|b$. By C, we know that $d_0|x$ and $d_0|y$. But we noted earlier that this means $d_0|x \bmod y$. So $d_0|y \wedge d_0|r$. Conversely, suppose $d_0|y \wedge d_0|r$. Then again $d_0|x$ because $x = qy + 1r$. So by C, we know $d_0|a$ and $d_0|b$. This completes the argument.

4. We then show that
$$D\{x := y; y := r\}E$$
using assignment and inertia. Now we conclude that
$$C \wedge y \neq 0\{r := x \bmod y; x := y; y := r\}C$$
by putting these two together (noting that $C \equiv E$).

5. We can now use the induction principle for **while** loops to get:
$$C\{\textbf{while} \ldots \textbf{end}\}C \wedge y = 0$$
That means by sequence
$$A\{x := a; y := b; \textbf{while} \ldots \textbf{end}\}C \wedge y = 0$$

6. Now we do a last little bit of logic. We know $C \wedge y = 0 \equiv F$. We need to show $F \rightarrow G$. We have $x > 0$ immediately. Why does $x|a$ and $x|b$? We know $\forall d(d|x \wedge d|y \rightarrow d|a \wedge d|b)$. We instantiate x for d and recall that $y = 0$: $x|x \wedge x|0 \rightarrow x|a \wedge x|a$. But of course since $x = 1x$ and $x = 00$, $x|x$ and $x|0$. So $x|a$ and $x|b$. Finally we get $\forall d(d|a \wedge d|b \rightarrow d|x)$ by logic from C, which entails that $\forall d(d|a \wedge d|b \rightarrow d|x \wedge d|y)$. Thus we know $F \rightarrow G$. But we gave an argument earlier that $G \rightarrow H$:
$$x > 0 \wedge x|a \wedge x|b \wedge \forall d(d|a \wedge d|b \rightarrow d|x) \rightarrow x = \gcd(a,b)$$
Putting these observations together gives:
$$A\{x := a; y := b; \textbf{while} \ldots \textbf{end}\}H$$
This concludes the proof that the program is correct.