2-1 The Correctness of Algorithms[1]

Hengfeng Wei (hfwei@nju.edu.cn)

13 March 2018 ∼ March 15, 2018

[1] Wish all your correctness proofs were CORRECT! May God bless you!

We show how to prove the correctness of two algorithms. One is an iterative algorithm called $\textsc{Equal}(S_1, S_2)$ for comparing two strings. The other is the classic recursive Euclid algorithm for computing the greatest common divisor (gcd) of two natural numbers.

DH Problem 5.9: $\textsc{Equal}(S_1, S_2)$

Construct a function $\textsc{Equal}(S_1, S_2)$ that tests whether the strings X and Y are equal. It should return true or false accordingly.

You may use the following operations:

- `head`(X)
- `tail`(X)
- `last`(X)
- `all-but-last`(X)
- `eq`(s, t)

Solution

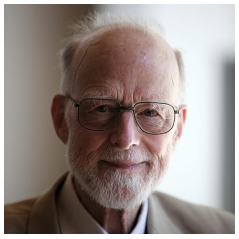
The algorithm $\textsc{Equal}(S_1, S_2)$ is shown in Algorithm 1, with appropriate assertions attached.

The partial correctness of Equal can be denoted as

$$P \, \{\textsc{Equal}\} \, Q,$$

meaning that if the input satisfies the precondition P and the algorithm Equal eventually terminates, then the postcondition Q must hold. To this end, we show that:

We are using the notations of Hoare logic developed by Tony Hoare.



C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969

(i) I is a loop invariant. That is,

$$P \, \{\texttt{Lines 2-4}\} \, I$$

$$\big(I \wedge (X \neq \epsilon \wedge Y \neq \epsilon \wedge E = \top)\big) \, \{\texttt{Lines 6-10}\} \, I$$

(ii) (2) is an invariant. It suffices to show that (2) can be derived from

This is a rather formal way. Be patient.

$$\begin{aligned} & I \wedge \neg(X \neq \epsilon \wedge Y \neq \epsilon \wedge E = \top) \\ = \; & I \wedge (X = \epsilon \vee Y = \epsilon \vee E = \bot) \\ = \; & \big(I \wedge (X = \epsilon)\big) \vee \big(I \wedge (Y = \epsilon)\big) \vee \big(I \wedge (E = \bot)\big) \end{aligned} \tag{a}$$

Algorithm 1 Comparing two strings.

```
1:  procedure EQUAL(S_1, S_2)
      ▷ P : S_1, S_2 are strings
2:    X ← S_1
3:    Y ← S_2
4:    E ← ⊤
      ▷ (1) I : S_1 = S_2 ⟺ (X = Y ∧ E = ⊤)
5:    while X ≠ ϵ ∧ Y ≠ ϵ ∧ E = ⊤ do
6:      if eq(head(X), head(Y)) then
7:        X ← tail(X)
8:        Y ← tail(Y)
9:      else
10:       E ← ⊥
      ▷ (2) S_1 = S_2 ⟺ ((X = ϵ ∧ Y = ϵ) ∧ E = ⊤)
11:   if ¬(X = ϵ ∧ Y = ϵ) then
12:     E ← ⊥
        ▷ (3.1) S_1 ≠ S_2 ∧ E = ⊥
13:   else
14:     DoNothing              ▷ Just for inserting an assertion here.
        ▷ (3.2) S_1 = S_2 ⟺ E = ⊤
      ▷ (4) Q : S_1 = S_2 ⟺ E = ⊤
15:   return E
```

in propositional logic by showing that each disjunct of (a) implies (2).

Consider the first disjunct

$$\begin{aligned} & I \wedge (X = \epsilon) \\ = \; & (S_1 = S_2 \iff X = Y \wedge E = \top) \wedge (X = \epsilon) \end{aligned} \tag{a1}$$

It is required to prove both

$$(\text{a1}) \implies \Big(S_1 = S_2 \implies \big((X = \epsilon \wedge Y = \epsilon) \wedge E = \top\big)\Big)$$

and

$$(\text{a1}) \implies \Big(\big((X = \epsilon \wedge Y = \epsilon) \wedge E = \top\big) \implies S_1 = S_2\Big)$$

Now it is your (YES, it is YOU) job! It is also your job to deal with the second and the third disjuncts in a similar way.

Do you still remember how to prove an implication? What is the given? And what is the conclusion? Furthermore, when you derive (2) from the third disjunct, keep in mind that *falsity implies everything*.

(iii) (3.1) is an invariant. It suffices to show that

$$\big((2) \wedge \neg(X = \epsilon \wedge Y = \epsilon) \wedge E = \bot\big) \implies (3.1).$$

Stop reading! Take your pencil and paper! Write down your

Writing is nature's way of letting you know how sloppy your thinking is.
— Richard Guindon (cartoon), 1989

arguments to convince yourself that the formula above indeed holds.

(iv) (3.2) is an invariant. It suffices to show that

$$\big((2) \wedge (X = \epsilon \wedge Y = \epsilon)\big) \implies (3.2).$$

You already know how to prove this formula in propositional logic. Don't you? We now show you a more intuitive way to reason about programs:

At point (2), we know that $S_1 = S_2$ if and only if $(X = \Lambda \wedge Y = \Lambda \wedge E = \top)$. At point (3.2), we further know that $(X = \Lambda \wedge Y = \Lambda)$. Thus, at this point, we only need to check whether $E = \top$ to decide whether $S_1 = S_2$ holds.

It proceeds *semantically* rather than *syntactically* as we have done before.

(v) Q is an invariant. It suffices to show that

$$\big((3.1) \implies Q\big) \wedge \big((3.2) \implies Q\big).$$

It should be easy now for you to prove this formula.

Extra Problem: EUCLID(m, n)

Prove the recursive Euclid algorithm as shown in Algorithm 2 for computing the greatest common divisor (gcd) of two natural numbers are totally correct.

Algorithm 2 The Euclid Algorithm

```
1: procedure EUCLID(m, n)
2:     if n > 0 then
3:         return m
4:     else
5:         return EUCLID(n, m mod n)
```

Proof

We prove the partial correctness of EUCLID by strong mathematical induction on n, with m any fixed natural number.

Also pay attention to the way how to write a mathematical induction proof.

Basis: $n = 0$. We have that

$$\gcd(m, n) = \gcd(m, 0) = m = \text{EUCLID}(m, 0).$$

Make sure you understand each of these three "="'s:

(1) By $n = 0$;

(2) By the property of gcd;

(3) By the EUCLID algorithm.

Inductive Hypothesis: Suppose that $n \geq 1$ and

$$\gcd(m, k) = \text{EUCLID}(m, k),\ \forall 0 \leq k \leq n - 1.$$

Inductive Step: We need to prove that ($n \geq 1$)

$$\gcd(m, n) = \textsc{Euclid}(m, n).$$

According to EUCLID, we have

$$\textsc{Euclid}(m, n) = \textsc{Euclid}(n, m \bmod n).$$

Since $(m \bmod n) < n$, by the inductive hypothesis, we have

$$\textsc{Euclid}(n, m \bmod n) = \gcd(n, m \bmod n).$$

Therefore, it suffices to prove that

$$\boxed{\gcd(m, n) = \gcd(n, m \bmod n).}$$

For notational convenience, we denote

$$d = \gcd(m, n), \quad d' = \gcd(n, m \bmod n).$$

Because $d, d' \geq 0$, it is sufficient to obtain $d = d'$ by showing that $d \mid d'$ and $d' \mid d$:

- $d \mid d'$.
- $d' \mid d$.

References

C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.