2-1 The Correctness of Algorithms[1]

Hengfeng Wei

13 March 2018

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.

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

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:

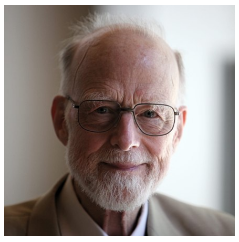
(i) I is a loop invariant. That is,

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

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

Stop reading! Take your pencil and paper! Write down your arguments to convince yourself that the two formulae above indeed hold.

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

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

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
```

(ii) (2) is an invariant. It suffices to show that

$$\big(I \wedge \neg(X \neq \epsilon \wedge Y \neq \epsilon \wedge E = \top)\big) \implies (2).$$

We proceed in two ways: syntactically and semantically. You choose the way you feel comfortable with.

Syntactically, we prove that (2) can be derived from $\big(I \wedge \neg(X \neq \epsilon \wedge Y \neq \epsilon \wedge E = \top)\big)$ in propositional logic. This is a rather formal way.

$$\begin{aligned} & I \wedge (X = \epsilon \vee Y = \epsilon \vee E = \bot) \\ = \ & \big(I \wedge (X = \epsilon)\big) \vee \big(I \wedge (Y = \epsilon)\big) \vee \big(I \wedge (E = \bot)\big) \end{aligned} \tag{1}$$

In the following, we show that each of the three disjuncts implies (2).

$$\begin{aligned} & I \wedge (X = \epsilon) \\ = \ & (S_1 = S_2 \iff X = Y \wedge E = \top) \wedge (X = \epsilon) \end{aligned} \tag{2}$$

We need to show that

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

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

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

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

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

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

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

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

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

Basis: $n = 0$. We have that

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

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 $(n \geq 1)$

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

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

(1) By $n = 0$;

(2) By the property of gcd;

(3) By the EUCLID algorithm.

According to EUCLID, we have

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

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

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

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.