2-1 The Correctness of Algorithms[1]

Hengfeng Wei

13 March 2018

[1] Wish all your correctness proofs were CORRECT!

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.

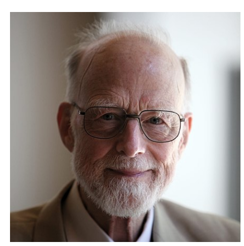
The partial correctness of $\textsc{Equal}$ can be denoted as

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

meaning that if the input satisfies the precondition P, then after $\textsc{Equal}$ the postcondition Q must hold. To this end, we show that:

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.

(ii) (2) is an invariant.

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

(iii) (3.1) is an invariant

(iv) (3.2) is an invariant

(v) Q is an invariant

Algorithm 1 Comparing two strings.

procedure EQUAL(S_1, S_2)
  ▷ $P : S_1, S_2$ are strings
    $X \leftarrow S_1$
    $Y \leftarrow S_2$
    $E \leftarrow \top$

    ▷ (1) $I : S_1 = S_2 \iff X = Y \wedge E = \top$
    while $X \neq \epsilon \wedge Y \neq \epsilon \wedge E = \top$ **do**
        if `eq`(`head`(X), `head`(Y)) **then**
            $X \leftarrow$ `tail`(X)
            $Y \leftarrow$ `tail`(Y)
        else
            $E \leftarrow \bot$

    ▷ (2) $S_1 = S_2 \iff (X = \epsilon \wedge Y = \epsilon) \wedge E = \top$
    if $\neg(X = \epsilon \wedge Y = \epsilon)$ **then**
        $E \leftarrow \bot$
        ▷ (3.1) $S_1 \neq S_2 \wedge E = \bot$
    else
        `DoNothing` ▷ Just for inserting an assertion here.
        ▷ (3.2) $S_1 = S_2 \iff E = \top$

    ▷ (4) $Q : S_1 = S_2 \iff E = \top$
    return E

Extra Problem: EUCLID(m, n)

Prove the following recursive Euclid algorithm for computing the greatest common divisor (gcd) of two natural numbers are totally correct.

Algorithm 2 The Euclid Algorithm

procedure EUCLID(m, n)
  if $n > 0$ **then**
    return m
  else
    return EUCLID($n, m \bmod n$)

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 ($n \geq 1$)

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

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.