Correctness ("Hoare") Triples, v1.1

Part 2: Sequencing, Assignment, Strengthening, and Weakening

CS 536: Science of Programming, Fall 2022

A. Examples of Partial and Total Correctness With Loops

- For the following examples, let $W \equiv$ *while* $k \neq 0$ { $k := k\text{-}1$ }.
  - Example 1: $\models[k \geq 0]\ W\ [k = 0]$. If we start in a state where k is ≥ 0, then the loop is guaranteed to terminate in a state satisfying $k = 0$.
  - Example 2: $\models \{k = -1\}\ W\ \{k = 0\}$ but $\not\models$ [k = -1] W [k = 0]. The triple is partially correct but not totally correct because it diverges if $k = -1$. Also note that partial correctness would hold if we substitute any predicate for $k = 0$.
  - Example 3: $\models \{T\}\ W\ \{k = 0\}$ but $\not\models$ [T] W [k = 0]. The triple is partially correct but not totally correct because it diverges for at least one value of k.
- For the following examples, let $W' \equiv$ *while* $k > 0$ $\{k := k\text{-}1\}$. (We're changing the loop test of W so that it terminates immediately when k is negative.)
  - Example 4: $\models$[T] W′ [k ≤ 0].
  - Example 5: $\models$ [k = c_0] W′ [($c_0 \leq 0 \rightarrow$ k = c_0) ∧ ($c_0 \geq 0 \rightarrow$ k = 0)]. This is Example 4 with the "strongest" (most precise) postcondition possible.

B. More Correctness Triple Examples

Same Code, Different Conditions

- The same piece of code can be annotated with conditions in different ways, and there's not always a "best" annotation. An annotation might be the most general one possible (we'll discuss this concept soon), but depending on the context, we might prefer a different annotation.
- Below, let $sum(x, y) = x + (x+1) + (x+2) + \ldots + y$. (If $x > y$, let $sum(x, y) = 0$.) In Examples 9 – 12, we have the same program annotated (with preconditions and postconditions) of various strengths (strength = generality).
- Example 9: $\{T\}\ i := 0;\ s := 0\ \{i = s = 0\}$.
  - This is the strongest (most precise) annotation for this program.
- Example 10: $\{T\}\ i := 0;\ s := 0\ \{i \geq 0\ \wedge$ s ≥ 0$\}$.
  - This is a strictly weaker (and therefore also correct) annotation.

- Example 10: $\{T\}\ i := 0; s := 0\ \{i = s = 0 = sum(0, i)\}$.
  - This adds a summation relationship to i and s when they're both zero.
- Example 11: $\{n \geq 0\}\ i := 0; s := 0\ \{0 = i \leq n \wedge s = 0 = sum(0, i)\}$
  - This limits i to a range of values $0, \ldots, n$. There's no way in the postcondition to know $n \geq 0$ unless we assume it in the precondition.
- Example 12: $\{n \geq 0\}\ i := 0; s := 0\ \{0 \leq i \leq n \wedge s = sum(0, i)\}$
  - The postcondition no longer includes $i = s = 0$ and is therefore weaker, which might seem like a disadvantage but will turn out to be an advantage later.
- The next two examples relate to calculating the midpoint in binary search. Though the code is the same, whether the midpoint is strictly between the left and right endpoints depends on whether or not the endpoints are nonadjacent.
  - Example 13: $\{L < R \wedge L \neq R - 1\}\ M := (L + R)/2\ \{L < M < R\}$
  - Example 14: $\{L < R\}\ M := (L + R)/2\ \{L \leq M < R\}$
- DeMorgan's laws can apply when a test is a conjunction or disjunction.
- Example 15: Here we search downward for a nonnegative x where $f(x)$ is $\leq y$; we stop if x goes negative or we find an x with $f(x) \leq y$.

  $\{x \geq 0\}$
  while $x \geq 0 \wedge f(x) > y$ *do* $x := x-1$ *od*
  $\{x < 0 \vee f(x) \leq y\}$
- Example 16: This is Example 15 rephrased as an array search; as long as we have a legal index i and $b[i]$ isn't $\leq y$, we move left. We stop if the index becomes illegal or we find an index with $b[i] \leq y$.

  $\{0 \leq i\}$
  while $i \geq 0 \wedge b[i] > y$ *do* $i := i-1$ *od*
  $\{i < 0 \vee b[i] \leq y\}$

