Finding Invariants

Part 1: Adding Parameters by Replacing Constants by Variables

CS 536: Science of Programming, Fall 2021

A. *Why*

- It is easier to write good programs and check them for defects than to write bad programs and then debug them.
- The hardest part of programming is finding good loop invariants.
- There are heuristics for finding them but no algorithms that work in all cases.

B. *Objectives*

At the end of this activity assignment you should

- Be able to how to generate possible invariants using "replace a constant by a variable" or more generally "add a parameter".

C. *Problems*

1. What are the constants in the postcondition $x = max(b[0], b[1], \ldots, b[n-1])$? Using the technique "replace a constant by a variable," list the possible invariants for this postcondition. Also, what would the loop tests be? (Assume $n-1$ is a constant.)
2. Repeat, on the postcondition $x = n!$, where $n!$ is short for a function call $product(1, n)$.
3. Repeat, on the postcondition $\forall i \,.\, 0 \le i < n \rightarrow b[i] = 3$.
4. Repeat, on the postcondition $\forall i \,.\, \forall j \,.\, 0 \le i < K \land K \le j < n \rightarrow b[i] < b[j]$. (Every value in $b[0 \ldots K-1]$ is < every value in $b[K \ldots n-1]$.)

Solution to Practice 19 (Finding Invariants; Examples)

1. Certainly *0* is a constant; if we replace it by a variable *i*, we get

   $\{inv\ x = max(b[i], \ldots, b[n-1]) \land 0 \le i \le n-1\}$ $while\ i \ne 0\ do \ldots$

   As a constant, *n-1* seems better than just *n* or 1 by themselves:

   $\{inv\ x = max(b[0], \ldots, b[j]) \land 0 \le j \le n-1\}$ $while\ j \ne n-1\ do \ldots$

   If you want to treat just *n* as a constant and replace it by a variable *j*, we get

   $\{inv\ x = max(b[0], \ldots, b[j-1]) \land 1 \le j \le n\}$ $while\ j \ne n\ do \ldots$

   Similarly, if you want replace just the *1* in *n-1* by with *j*, we get

   $\{inv\ x = max(b[0], \ldots, b[n-j]) \land 1 \le j \le n\}$ $while\ j \ne 1\ do \ldots$

2. We can replace *n* by a variable and get

   $inv\ x = i! \land 1 \le i \le n\}$ $while\ i \ne n\ do \ldots$

   We can replace *1* and get

   $\{inv\ x = j*(j+1)*\ldots*n \land 1 \le j \le n\}$ $while\ j \ne 1\ do \ldots$

3. For $\forall i\ .\ 0 \le i < n \to b[i] = 3$ as the postcondition, we can replace *0* or *n* or *3*.

   Replace *0* by *k*:

   $\{inv\ 0 \le k \le n-1 \land \forall i\ .\ k \le i < n \to b[i] = 3\}$ $while\ k \ne 0\ do \ldots$

   Replace *n* by *k*

   $\{inv\ 0 \le k \le n \land \forall i\ .\ 0 \le i < k \to b[i] = 3\}$ $while\ k \ne n\ do \ldots$

   Replace *3* by *k* (this doesn't look useful)

   $\{inv\ \forall i\ .\ 0 \le i < n \to b[i] = k\}$ $while\ k \ne 3\ do \ldots$

4. For $\forall i\ .\forall j\ .0 \le i < K \land K \le j < n \to b[i] < b[j]$, we have constants *0*, *n*, the two occurrences of *K*.

   Replace *0* by *k*:

   $\{inv\ 0 \le k < K \land \forall i\ .\ \forall j\ .\ k \le i < K \land K \le j < n \to b[i] < b[j]\}$
   $while\ k \ne 0$

   Replace left *K* by *k*:

   $\{inv\ 0 \le k < K \land \forall i\ .\ \forall j\ .\ 0 \le i < k \land K \le j < n \to b[i] < b[j]\}$
   $while\ k \ne K$

   Replace right *K* by *k*:

   $\{inv\ K \le k \le n \land \forall i\ .\ \forall j\ .\ 0 \le i < K \land k \le j < n \to b[i] < b[j]\}$
   $while\ k \ne K$

   Replace *n* by *k*:

   $\{inv\ K \le k \le n \land \forall i\ .\ \forall j\ .\ 0 \le i < K \land K \le j < k \to b[i] < b[j]\}$
   $while\ k \ne n$

You could argue that the ranges for k could be $0 \le k < n$, $0 \le k < n$, $0 \le k \le n$, and $0 \le k \le n$ for the four cases above; it depends on knowing more about the context of the problem.