Finding Invariants

Part 2: Deleting Conjuncts; Adding Disjuncts

CS 536: Science of Programming, Fall 2022

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

- Know how to generate possible invariants using the techniques “Drop a conjunct” and “Add a disjunct”.

C. Problems

1. Consider the postcondition $x^2 \le n < (x+1)^2$, which is short for $x^2 \le n \wedge n < (x+1)^2$. List the possible invariant/loop test combinations you can get for this postcondition using the technique “Drop a conjunct.”

2. Why is the technique “Drop a conjunct” a special case of “Add a disjunct”?

3. One way to view a search is as follows:

   *{**inv** found ∨ not found}*
   while *not found*
   do
       Remove something or somethings from the things to look at
   od

   For this problem, try to recast (a) linear search and (b) binary search of an array using this framework: What parts of that program correspond to “we have found it”, “we haven’t found it”, and “Remove something…”?

4. In Example 7 (integer square root), in the false branch of the ***if-else*** statement, can we replace the assignment $y := y - y÷2$ with $y := y÷2$? If not, why not?

5. Complete the annotation of Binary Search version 1 (Example 2).

6. Complete the annotation of Binary Search version 2 (Example 3).

Solution to Activity 20 (Finding Invariants; Examples)

1. $\{\textbf{inv}\ n < (x+1)^2\}\ \textbf{while}\ x^2 > n\ \ldots$

   $\{\textbf{inv}\ x^2 \le n\}\ \textbf{while}\ n \ge (x+1)^2\ \ldots$

2. Dropping a conjunct is like adding the difference between the dropped conjunct and the rest of the predicate. E.g., dropping p_1 from $p_1 \wedge p_2 \wedge p_3$ is like adding $(\neg p_1 \wedge p_2 \wedge p_3)$ to $(p_1 \wedge p_2 \wedge p_3)$.

3. (Rephrasing searches)
   a. We can rephrase linear search through an array with
      We have found it: $k < n \wedge b[k] = x$
      We haven't found it: $k < n \wedge b[k] \ne x$
      Remove what we're looking at from the things to look at: $k := k+1$
   b. We can rephrase binary search through an array with
      We have found it: $R = L+1$
      We haven't found it: $R > L+1$
      Remove the left or right half from the things to look at: Either $L := m$ or $R := m$

4. We can't replace $y := y - y÷2$ by $y := y÷2$ because for y odd, $y÷2 = y - y÷2 - 1$, which is not strong enough to re-establish $n < (x+y)^2$.

5. (Binary search, version 1) *[*Not included: The intermediate conditions within loop initialization*]*

$\{q_0 \equiv Sorted(b, n) \wedge n \ge 1 \wedge b[0] \le x < b[n]\}$
$L := 0\ ;\ R := n\ ;\ found := F\ ;$
$\{Sorted(b, n) \wedge n \ge 1 \wedge b[0] \le x < b[n] \wedge L = 0 \wedge R = n \wedge \neg found\}$
$\{\textbf{inv}\ p \equiv 0 \le L < R \le n \wedge b[L] \le x < b[R] \wedge (found \rightarrow x = b[L])\}\ \{\textbf{bd}\ R-L\}$
$\textbf{while}\ \neg found \wedge R \ne L+1\ \textbf{do}$
    $\{p \wedge \neg found \wedge R \ne L+1 \wedge R-L = t_0\}$
    $m := (L+R)/2\ ;$
    $\{p_1 \equiv p \wedge \neg found \wedge R \ne L+1 \wedge R-L = t_0 \wedge m = (L+R)/2\}$
    $\textbf{if}\ b[m] = x\ \textbf{then}$
        $\{p_1 \wedge b[m] = x$
            $\equiv 0 \le L < R \le n \wedge b[L] \le x < b[R] \wedge (found \rightarrow x = b[L])$
                $\wedge \neg found \wedge R \ne L+1 \wedge R-L = t_0 \wedge m = (L+R)/2 \wedge b[m] = x\}$
        $\{p[T/found][m/L] \wedge R-m < t_0$
            $\equiv 0 \le m < R \le n \wedge b[m] \le x < b[R] \wedge (T \rightarrow x = b[m]) \wedge R-m < t_0\}$
        $found := T\ ;\ L := m$
        $\{p \wedge R-L < t_0\}$
    $\textbf{else if}\ b[m] < x\ \textbf{then}$

$\{p_1 \wedge b[m] < x$ // *technically, should include* $b[m] \neq x$

$\equiv 0 \le L < R \le n \wedge b[L] \le x < b[R] \wedge (found \rightarrow x < b[L])$

$\wedge \neg found \wedge R \neq L+1 \wedge R-L = t_0 \wedge m = (L+R)/2 \wedge b[m] < x\}$

$\{p[m/L] \wedge R-m < t_0$

$\equiv 0 \le m < R \le n \wedge b[m] \le x < b[R] \wedge (found \rightarrow x = b[m]) \wedge R-m < t_0\}$

$L := m$

$\{p \wedge R-L < t_0\}$

else // $b[m] > x$

$\{p_1 \wedge b[m] > x$ // *technically, should include* $b[m] \neq x \wedge b[m] \not< x$

$\equiv 0 \le L < R \le n \wedge b[L] \le x < b[R] \wedge (found \rightarrow x < b[L])$

$\wedge \neg found \wedge R \neq L+1 \wedge R-L = t_0 \wedge m = (L+R)/2 \wedge b[m] > x\}$

$\{p[m/R] \wedge m-L < t_0$

$\equiv 0 \le L < m \le n \wedge b[L] \le x < b[m] \wedge (found \rightarrow x = b[L]) \wedge m-L < t_0\}$

$R := m$

$\{p \wedge R-L < t_0\}$

fi fi

$\{p \wedge R-L < t_0\}$

od

$\{p \wedge (found \vee R = L+1)\}$

$\{0 \le L < n \wedge (found \leftrightarrow x = b[L])\}$

6. (Binary search, version 2) *[*Not included: The intermediate conditions within loop initialization*]*

$\{n > 0 \wedge Sorted(b, n) \wedge b[0] \le x < b[n-1]\}$

$L := 0;\ R := n-1;\ found := F;$

$\{n > 0 \wedge Sorted(b, n) \wedge b[0] \le x < b[n-1] \wedge L = 0 \wedge R = n-1 \wedge \neg found\}$

$\{\textbf{inv}\ q \equiv -1 \le L-1 \le R < n \wedge (found \rightarrow b[L] = x) \wedge (x \in b[0..n-1] \leftrightarrow x \in b[L..R])\}$

$\{\textbf{bd}\ R-L+1+|\neg found|\}$

while $\neg found \wedge L \le R$ ***do***

$\{q \wedge \neg found \wedge L \le R \wedge R-L+1+|\neg found| = t_0\}$

$m := (L+R)/2;$

$\{q_1 \equiv q \wedge \neg found \wedge L \le R \wedge R-L+1+|\neg found| = t_0 \wedge m = (L+R)/2\}$

if $b[m] = x$ ***then***

$\{q_1 \wedge b[m] = x$

$\equiv -1 \le L-1 \le R < n \wedge (found \rightarrow b[L] = x) \wedge (x \in b[0..n-1] \leftrightarrow x \in b[L..R])$

$\wedge \neg found \wedge L \le R \wedge R-L+1+|\neg found| = t_0 \wedge m = (L+R)/2 \wedge b[m] = x\}$

$\{q[T/found]\ [m/L] \wedge R-(m+1)+1+|\neg T| < t_0$

$\equiv -1 \le m-1 \le R < n \wedge (T \rightarrow b[m] = x)$

$\wedge (x \in b[0..n-1] \leftrightarrow x \in b[m..R]) \wedge R-m+1+|\neg T| < t_0\}$

$found := T\ ;\ L := m$

$\{q \wedge R-L+1+|\neg found| < t_0\}$

else if $b[m] < x$ ***then***

$\{q_1 \wedge b[m] < x$ // technically, should include $b[m] \neq x$

$\equiv -1 \leq L-1 \leq R < n \wedge (found \rightarrow b[L] = x) \wedge (x \in b[0..n-1] \leftrightarrow x \in b[L..R])$

$\wedge \neg found \wedge L \leq R \wedge R-L+1+|\neg found| = t_0 \wedge m = (L+R)/2 \wedge b[m] < x\}$

$\{q[m+1/L] \wedge R-(m+1)+1+|\neg found| < t_0$

$\equiv -1 \leq (m+1)-1 \leq R < n \wedge (found \rightarrow b[m+1] = x)$

$\wedge (x \in b[0..n-1] \leftrightarrow x \in b[m+1..R]) \wedge R-(m+1) +1+|\neg found| < t_0\}$

$L := m+1$

$\{q \wedge R-L+1+|\neg found| < t_0\}$

else // $b[m] > x$// technically, should include $b[m] \neq x \wedge b[m] \nless x$

$\{q_1 \wedge b[m] > x$

$\equiv -1 \leq L-1 \leq R < n \wedge (found \rightarrow b[L] = x) \wedge (x \in b[0..n-1] \leftrightarrow x \in b[L..R])$

$\wedge \neg found \wedge L \leq R \wedge R-L+1+|\neg found| = t_0 \wedge m = (L+R)/2 \wedge b[m] > x\}$

$\{q[m-1/R] \wedge (m-1)-L+1+|\neg found| < t_0\}$

$R := m-1$

$\{q \wedge R-L+1+|\neg found| < t_0\}$

fi fi $\{q \wedge R-L+1+|\neg found| < t_0\}$

od

$\{q \wedge (found \vee L > R)$

$\equiv -1 \leq L-1 \leq R < n \wedge (found \rightarrow b[L] = x) \wedge (x \text{ in } b[0..n-1] \leftrightarrow x \text{ in } b[L..R])$

$\wedge (found \vee L > R) \}$

$\{-1 \leq L-1 \leq R < n \wedge (found \rightarrow b[L] = x) \wedge (\neg found \rightarrow x \notin b[0..n-1])\}$