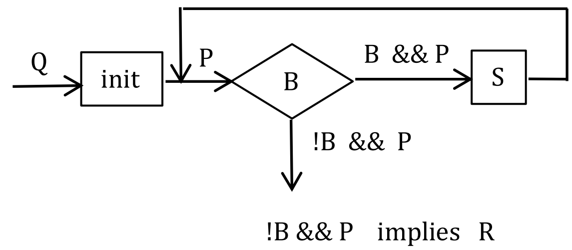
Practice with the second loopy question

Introduction

We practice finding a loop condition B by using the second loopy question: Is !B && P => R true? Thus, we look for !B that makes !B && P => R true and complement !B to get B.



Here's the invariant P and postcondition for our first example.

P: s is the sum of m..k–1 and $m \le k \le n$
R: s is the sum of m..n-1

Knowing that P is true, and doing some pattern matching with P and R, we see that R will be true if k = n. Therefore, !B is k = n, so the loop condition B is k != n. Looking at the restriction on k in invariant P, we can write the loop condition at k < n if we want. Thus, we use either

while (k != n) { … } or **while** (k < n) { … }

A second example

Here are the invariant and postcondition for a loop to calculate the minimum value in array segment b[0..n-1]:

P: v = minimum of b[0..k–1] and $0 \le k \le n$
R: v = minimum of b[0..n–1]

Using reasoning like we did the first example, you can see that we get the same answer for B as in the previous example.

while (k != n) { … } or **while** (k < n) { … }

Computing z = b^c

Here are the invariant and postcondition for a loop to store b^c in z, given $c \ge 0$:

P: b^c = z * x^y and $y \ge 0$
R: z = b^c

Again doing pattern matching, we see that R will be true when P is true and x^y = 1. That last formula, x^y = 1, is true, when y = 0. So our loop condition is $y \ne 0$:

while (y != 0) { … }

Exercises

In the two examples below, find the loop condition. Answers are at the end of the pdf script for this video.

1. P: s is the sum of k..n–1 and $m \le k \le n$
R: s is the sum of m..n-1

2. P: v = minimum of b[k..n] and $0 \le k \le n$
R: v = minimum of b[0..n] and $0 \le k \le n$

Answers

In the first exercise, doing pattern matching on P and R, we see that k = m is needed. Therefore the loop condition is k != m. This can be written as m < k if you want, since $m \le k \le n$:

while (k != m) { … } or **while** (m < k) { … }

In the second exercise, pattern matching on P and R, we see that k = 0 is needed. Therefore, the loop condition condition is k != 0. This can be written as 0 < k if you want, since $0 \le k \le n$:

while (k != 0) { … } or **while** (m < k) { … }