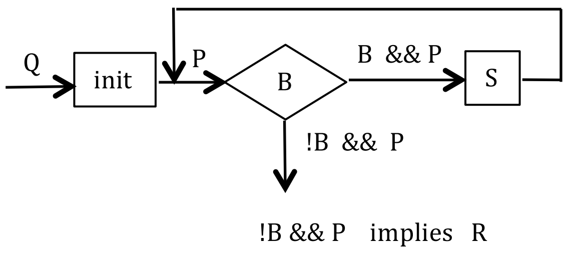
Practice with the first loopy question

Introduction

We practice developing the initialization of a loop by finding initialization to truthify {Q} init {P}.



Here's the precondition Q and invariant P for our first example. Q is the usual restriction on a range m..k-1.

Q: m ≤ n
P: s is the sum of m..k–1 and m ≤ k ≤ n

We can use simple assignments to calculate the sum of a range of values only if the range is empty or contains one value. In this case, range m..k-1 is empty if k = m, that is, the range is m..m-1. So we have the initialization:

k= m; s= 0; // The sum of no values is 0.

Note that the range m..k-1 *could* be empty. So we *have* to start off with k = m.

In any situation in which invariant P contains a range, the initialization usually makes that range contain 0 or 1 values. Be aware of this situation; it happens over and over again.

A case where the range cannot be empty

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

Q: 0 < n
P: v = minimum of b[0..k–1] and 0 ≤ k ≤ n

Because the minimum of an empty set is not defined, the initialization *cannot* make range 0..k-1 empty. Thus, initialization will make the range contain a single value, b[0]:

k= 1; v= b[0]; // the min value is the only value in the segment b[0..0]

We concluded the first example by showing you that making the range empty is easy. The second example reminds you that you have to think. You can't just blindly apply what we did in the first example everywhere.

Pattern matching

A lot of problem soving in computer science and mathematics deals with recognizing patterns and learning from them or using them. This example illustrates this idea of pattern matching in a simple way. Consider calculating b^c (b to the power c) for c ≥ 0. Upon termination, we want z = b^c. Here are the precondition and invariant:

Q: c ≥ 0
P: b^c = z * x^y and y ≥ 0

The initialization must store values in z, x, and y to make invariant P true —the constants b and c of the algorithm don't change. Both sides of the equality in the invariant contain exponentiation —b^c on one side and x^y on the other side. Thus, we are encouraged to assign b to x and c to y:

x= b; y= c; // then b^c = x^y

Also, this makes y ≥ 0 because of precondition Q. But, to have the first term of the invariant true, z must be 1. Thus, the initialization is:

{Q} z= 1; x= b; y= c {P}

Exercises

At this time, please find the initialization for the following two situations. The answers can be found at the end of the pdf script for this video. Be careful.

1. Q: m ≤ n
P: s is the sum of k..n-1 and m ≤ k ≤ n

2. Q: 0 < n
P: v = minimum of b[k..n] and 0 ≤ k ≤ n

Practice with the first loopy question

Answer to exercises

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

To make the range k..n-1 empty, set k to n. The initialization is then: k= n; s= 0;

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

Since the minimum of b[k..n] is desired, range k..n needs to have at least one value. Therefore, set k to n and then v to b[n]. So, the initialization is: k= n; v= b[n];