Programming Languages: Imperative Program Construction
Practicals 8: Case Studies

Shin-Cheng Mu

Autumn Term, 2024

1. For $r, b : \mathit{Int}$, verify the following program.

$$
\begin{array}{l}
\{0 \leqslant r < b \wedge \mathit{even}\ b\} \\
b := b \mathbin{/} 2 \\
\textbf{if}\ r < b \rightarrow \mathit{skip} \\
\ \mid\ b \leqslant r \rightarrow r := r - b \\
\textbf{fi} \\
\{0 \leqslant r < b\}
\end{array}
$$

Solution: We try to establish

$$
\begin{array}{l}
\{0 \leqslant r < b \wedge \mathit{even}\ b\} \\
b := b \mathbin{/} 2 \\
\{0 \leqslant r < 2 \times b\} \\
\textbf{if}\ r < b \rightarrow \mathit{skip} \\
\ \mid\ b \leqslant r \rightarrow r := r - b \\
\textbf{fi} \\
\{0 \leqslant r < b\}\ .
\end{array}
$$

The first Hoare triple is established by

$$
\begin{array}{ll}
& (0 \leqslant r < 2 \times b)[b \backslash b \mathbin{/} 2] \\
\equiv & (0 \leqslant r < 2 \times (b \mathbin{/} 2)) \\
\Leftarrow & 0 \leqslant r < b \wedge \mathit{even}\ b\ .
\end{array}
$$

Note that without *even b*, the fragment

$$
\begin{array}{l}
\{0 \leqslant r < b\} \\
b := b \mathbin{/} 2 \\
\{0 \leqslant r < 2 \times b\}
\end{array}
$$

would not be true. (A thank you to the student who pointed this out.)

The **if** statement is total because $r < b \vee b \leqslant r \equiv \mathit{True}$. For the first branch, certainly

$$0 \leqslant r < 2 \times b \wedge r < b \Rightarrow 0 \leqslant r < b\ .$$

For the second branch,

$$
\begin{array}{ll}
& (0 \leqslant r < b)[r \backslash r - b] \\
\equiv & 0 \leqslant r - b < b \\
\equiv & b \leqslant r < 2 \times b \\
\Leftarrow & 0 \leqslant r < 2 \times b \wedge b \leqslant r\ .
\end{array}
$$

The proof above looks trivial now. In 1972, Dijkstra needed textual proof that was about one page long, which was not that easy to comprehend. It shows the advances of symbolic reasoning since then.

2. Derive a $O(\log N)$ algorithm for computing square root:

   con $N : Int\ \{0 \leqslant N\}$
   var $x : Int$
   ?
   $\{x^2 \leqslant N < (x+1)^2\}$,

   by introducing a variable y and use $P_0 \wedge P_1$ as the invariant, where

   $P_0 \equiv x^2 \leqslant N < (x+y)^2$,
   $P_1 \equiv 0 \leqslant k \wedge y = 2^k$.

Solution: We can establish $P_1 \wedge N < y^2$ by a loop:

$y, k := 1, 0$
do $y^2 \leqslant N \to y, k := 2 \times y, k+1$ **od**

Afterwards, the invariant $P_0 \wedge P_1$ can be established by $x := 0$.

We consider the effect of $y := y/2$ on P_0:

$$\begin{aligned} & (x^2 \leqslant N < (x+y)^2)[y \backslash (y/2)] \\ \equiv\ & x^2 \leqslant N < (x + y/2)^2 \ . \end{aligned}$$

(Note that we mean $x + (y/2)$, not $(x+y)/2$!)

There are two possibilities $x^2 \leqslant N < (x+y/2)^2$ can be established. It could be the case that, luckily, $N < (x+y/2)^2$ already holds, hinting at using

$$N < (x+y/2)^2 \to y := y/2$$

Otherwise, when $(x+y/2)^2 \leqslant N$, we let $x := x + y/2$. The hint is that $(x+y/2)[x \backslash (x+y/2)] = x + y/2 + y/2 = x + y$, therefore we regain the right side of P_0. Let us see whether it works:

$$\begin{aligned} & (x^2 \leqslant N < (x+y/2)^2)[x \backslash (x+y/2)] \\ \equiv\ & (x+y/2)^2 \leqslant N < (x + y/2 + y/2)^2 \\ \Leftarrow\ & (x+y/2)^2 \leqslant N \wedge x^2 \leqslant N < (x+y)^2 \ . \end{aligned}$$

In summary, the program we get is:

con $N : Int\ \{0 \leqslant N\}$
var $x, y, k : Int$

$y, k := 1, 0$
do $\cdot\ y \leqslant N \to y, k := 2 \times y, k+1$ **od**
$\{P_1 \wedge N < y^2\}$
$x := 0$
$\{x^2 \leqslant N < (x+y)^2 \wedge 0 \leqslant k \wedge y = 2^k \wedge 0 \leqslant x, bnd : y\}$
do $y \neq 1 \to$
  if $N < (x+y/2)^2 \to y, k := y/2, k-1$
  $\mid (x+y/2)^2 \leqslant N \to x, y, k := x + y/2, y/2, k-1$
  fi
od
$\{x^2 \leqslant N < (x+1)^2\}$.

Alternatively, the loop could be:

$\{x^2 \leqslant N < (x+y)^2 \wedge 0 \leqslant k \wedge y = 2^k \wedge 0 \leqslant x, bnd : y\}$
do $y \neq 1 \rightarrow$
  $y, k := y/2, k-1$
  $\{x^2 \leqslant N < (x + 2 \times y)^2 \wedge 0 \leqslant k+1 \wedge 2 \times y = 2^{k+1} \wedge 0 \leqslant x\}$
  if $N < (x+y)^2 \rightarrow skip$
  $\mid (x+y)^2 \leqslant N \rightarrow x := x + y$
  fi
od .

You may find the second program easier to come up with.

3. Derive, again, a $O(\log N)$ algorithm for computing square root:

con $N : Int\ \{0 \leqslant N\}$
var $x : Int$
?
$\{x^2 \leqslant N < (x+1)^2\}$.

(a) This time, construct an algorithm using binary search. What Φ will you use? Does your program rely on the fact that x^2 is monotonic on x (that is, $x \geqslant y \Rightarrow x^2 \geqslant y^2$)?

Solution: Choose $\Phi\ x\ y = \neg\ (Q\ x) \wedge Q\ y$ where $Q\ z = N < z^2$, that is, $\Phi\ x\ y = x^2 \leqslant N < y^2$. It can be established by $x, y := 0, N+1$, since $N < (N+1)^2$.

con $N : Int\ \{0 \leqslant N\}$
var $x, y, m : Int$
$x, y := 0, N+1$
$\{\Phi\ x\ y \wedge 0 \leqslant x < y \leqslant N+1, bnd : y - x\}$
do $x + 1 \neq y \rightarrow$
  $m := (x+y) \mathbin{/} 2$
  if $m^2 \leqslant N \rightarrow x := m$
  $\mid N < m^2 \rightarrow y := m$
  fi
od
$\{x^2 \leqslant N < (x+1)^2\}$.

We do not need monotonicity for the program to be correct.

(b) Knowing that $x \geqslant y \Rightarrow x^2 \geqslant y^2$, after the loop terminates, what can you conclude in addition to $x^2 \leqslant N < (x+1)^2$?

Solution: Having the monotonicity, one can check whether $x^2 = N$ and conclude that

$$x^2 = N \vee \neg \langle \exists n : n \in Nat : n^2 = N \rangle \ .$$

That is, if $x^2 \neq N$, we may conclude that N is not a square number (an integer that is the square of an integer).