More examples of loops

Exponentiation

We will look at the problem of computing x^y where y is an integer and is initially nonnegative

$$f = \langle y \geq 0 \Rightarrow z' = x^y \rangle$$

We need to generalize this to get a specification for a loop.

Suppose that we have a partially computed answer already in z.

Then the remaining problem is

$$g = \langle y \geq 0 \Rightarrow z' = z \times x^y \rangle$$

Now

$$f \sqsubseteq z := 1; g$$

Note that when $y = 0$ the problem is easy

$$\begin{aligned} & \langle y = 0 \Rightarrow z' = z \times x^y \rangle \\ = \; & \text{One-point} \\ & \left\langle y = 0 \Rightarrow z' = z \times x^0 \right\rangle \\ = \; & \text{Since } x^0 \text{ is 1 and 1 is the identity of multiplication} \\ & \langle y = 0 \Rightarrow z' = z \rangle \\ \sqsubseteq \; & \text{Erasure law} \\ & \mathbf{skip} \end{aligned}$$

When $y > 0$ we can use the fact that $x^y = x \times x^{y-1}$

$$\begin{aligned}
& \langle y \neq 0 \rangle \Rightarrow g \\
= \; & \text{Shunting} \\
& \langle y > 0 \Rightarrow z' = z \times x^y \rangle \\
= \; & \text{Above fact about exponentiation} \\
& \left\langle y - 1 \geq 0 \Rightarrow z' = z \times x \times x^{y-1} \right\rangle \\
= \; & \text{Substitution law} \\
& y, z := y - 1, z \times x; g
\end{aligned}$$

By the alternation law we have the recursive refinement

$$g \sqsubseteq \textbf{if } y \neq 0 \textbf{ then } (y, z := y - 1, z \times x; g) \textbf{ else skip}$$

We can use y as a bound to justify that

$$g \sqsubseteq \textbf{while } y \neq 0 \textbf{ do } y, z := y - 1, z \times x$$

A tremendous improvement

Let’s revisit the “then” branch.

We need to implement

$$\langle y > 0 \Rightarrow z' = z \times x^y \rangle$$

When y is even, we can apply the fact that $x^y = \left(x^2\right)^{y/2}$

$$\begin{aligned}
& \langle y > 0 \wedge \text{even}(y) \Rightarrow z' = z \times x^y \rangle \\
= \; & \text{Above fact about exponents} \\
& \left\langle y > 0 \wedge \text{even}(y) \Rightarrow z' = z \times \left(x^2\right)^{y/2} \right\rangle \\
\sqsubseteq \; & \text{Strengthening by weakening the precondition} \\
& \left\langle y/2 \geq 0 \Rightarrow z' = z \times \left(x^2\right)^{y/2} \right\rangle \\
= \; & \text{Substitution law} \\
& x, y := x \times x, y/2; g
\end{aligned}$$

For the case where y is odd, we can use the same refinement as before

This gives us an implementation the "then part" of

$$
\begin{array}{ll}
& \langle y > 0 \Rightarrow z' = z \times x^y \rangle \\
\sqsubseteq & \text{Alternation law and above derivations} \\
& \quad \mathbf{if}\ \mathrm{even}(y) \\
& \quad \mathbf{then}\ (x, y := x \times x, y/2; g) \\
& \quad \mathbf{else}\ (y, z := y - 1, z \times x; g) \\
= & \text{Distributivity} \\
& \left(\begin{array}{l} \mathbf{if}\ \mathrm{even}(y) \\ \mathbf{then}\ x, y := x \times x, y/2 \\ \mathbf{else}\ y, z := y - 1, z \times x \end{array} \right) ; g
\end{array}
$$

The last step uses the following distributivity law

$$(\mathbf{if}\ \mathcal{A}\ \mathbf{then}\ (f_0; g)\ \mathbf{else}\ (f_1; g)) = (\mathbf{if}\ \mathcal{A}\ \mathbf{then}\ f_0\ \mathbf{else}\ f_1\ ;\ g)$$

Exercise: Prove this law from the definitions.

This gives a solution of

$$
\begin{array}{ll}
g \sqsubseteq & \mathbf{while}\ y \neq 0 \\
& \mathbf{do}\ \ \mathbf{if}\ \mathrm{even}(y) \\
& \qquad \mathbf{then}\ x, y := x \times x, y/2 \\
& \qquad \mathbf{else}\ y, z := y - 1, z \times x
\end{array}
$$

Exercise: How many iterations does this loop take if y is 2, 4, 8, 16, 32, etc.? How many if y is 3, 7, 15, 31, etc.? As a function of y, how many iterations does the loop take? [Hint, think about the binary representation of y.]

This algorithm represents a tremendous improvement over the previous. If y is, for example $1,000,000$ then the first algorithm requires $1,000,000$ multiplications, whereas this algorithm requires roughly 30.

A search

Suppose that B is a constant[1] boolean function and we know that

$$\exists i \in \{0, ..N\} \cdot B(i)$$

The goal is to find the smallest argument for which B is true

$$f = \langle k' = \min \{i \in \{0, ..N\} \mid B(i)\}\rangle$$

We need to generalize f to get a specification suitable for a loop.

Suppose that the first k items of B have already been searched, the remaining task is to search the remaining items

$$g = \langle 0 \leq k < N \Rightarrow k' = \min \{i \in \{k, ..N\} \mid B(i)\}\rangle$$

So

$$f \sqsubseteq k := 0; g$$

Exercise: Derive an implementation for g.

[1] By constant, I mean that the value of B does not change when the state changes.