More on Iteration

In order to use while loops in programming, we need to be able to prove statements such as

$$g \sqsubseteq \mathbf{while}\ \mathcal{A}\ \mathbf{do}\ h$$

So we need to know under what conditions this refinement holds.

Earlier we saw an incomplete law

While law (incomplete version): For any g, h, and $\mathcal{A}$, such that ... if

$$g \sqsubseteq \mathbf{if}\ \mathcal{A}\ \mathbf{then}\ (h; g)\ \mathbf{else\ skip}$$

then

$$g \sqsubseteq \mathbf{while}\ \mathcal{A}\ \mathbf{do}\ h$$

But we still need to fill in the "..."

It turns out that what is missing is that we must require the loop to terminate.

Loop Termination

Consider two loops where x is a natural number program variable

$$\mathbf{while}\ x > 0\ \mathbf{do}\ x := x - 1$$

and

$$\mathbf{while}\ x \neq n\ \mathbf{do}\ x := x + 1$$

The first definitely terminates, while the second may not. How can we show that a loop terminates?

In the first case we can state a *bound* on the number of remaining iterations. Namely x. At any point, there can not be more than x iterations left. In the second case we can not do this.

If we can find a bound for a loop, then it always terminates.

A better iteration law

While law (better version): For any g, h, and $\mathcal{A}$, such that $\mathbf{while}\ \mathcal{A}\ \mathbf{do}\ h$ always terminates, if

$$g \sqsubseteq \mathbf{if}\ \mathcal{A}\ \mathbf{then}\ (h; g)\ \mathbf{else\ skip}$$

then

$$g \sqsubseteq \mathbf{while}\ \mathcal{A}\ \mathbf{do}\ h$$

An even better iteration law

In practice, we do not require that loops terminate regardless of the initial state. We only require that they terminate when started in states that matter.

For example, we would expect that for an integer i

$$\langle i \leq n \Rightarrow i' = n \rangle \sqsubseteq \mathbf{while}\ i \neq n\ \mathbf{do}\ i := i + 1$$

even though, the loop does not terminate when started with an initial value of i that is larger than n.

To show this sort of refinement we need an even better law.

We will use a program variable τ of type $\mathbb{Z}$ to count the number of repetitions of a loop.

Let us strengthen the example specification to include information about the maximum number of repetitions allowed

$$g = \langle (i \leq n \Rightarrow i' = n \wedge \tau' \leq \tau + (n - i)) \rangle$$

Now we show

$$g \sqsubseteq \mathbf{if}\ i \neq n\ \mathbf{then}\ (i := i + 1; \tau := \tau + 1; g)\ \mathbf{else}\ \mathbf{skip}$$

While law (final version): For any g, h, and $\mathcal{A}$, where g is of the form $\langle P \Rightarrow Q \wedge \tau' \leq \tau + \mathcal{E} \rangle$ and $\mathcal{E}$ is a natural number expression, if

$$g \sqsubseteq \mathbf{if}\ \mathcal{A}\ \mathbf{then}\ (h; \tau := \tau + 1; g)\ \mathbf{else}\ \mathbf{skip}$$

then

$$g \sqsubseteq \mathbf{while}\ \mathcal{A}\ \mathbf{do}\ h$$

($\mathcal{E}$ is called the *bound* for the loop.)

Practically, what this means is that when $\mathcal{A}$ and $\mathcal{E} > 0$ are true initially, h needs to decrease the value of $\mathcal{E}$ by at least 1.

Summation revisited

In the summation problem above we had to refine

$$\left\langle i \leq n \Rightarrow s' = s + \left(\sum_{k \in \{i,..n\}} a(k) \right) \right\rangle$$

with a loop. In that case the bound can be $n - i$, so we should refine

$$\left\langle i \leq n \Rightarrow s' = s + \left(\sum_{k \in \{i,..n\}} a(k) \right) \wedge \tau' \leq \tau + n - i \right\rangle$$

by a while loop.

GCD revisited

In the GCD problem we had to refine

$$\langle a \neq 0 \vee b \neq 0 \Rightarrow a' = \gcd(a, b) \rangle$$

in this case we can use b as the bound. We can show that

$$\langle a \neq 0 \vee b \neq 0 \Rightarrow a' = \gcd(a, b) \wedge \tau' \leq \tau + b \rangle$$

is implemented by

$$\textbf{while } b \neq 0 \textbf{ do } a, b := b, a \bmod b$$

Defining Iteration

Although I've stated some laws about iteration, we can't prove these laws yet, as we haven't actually defined iteration.

We can define

$$\mathbf{while}\ \mathcal{A}\ \mathbf{do}\ h$$

to be the least refined specification w such that

$$\mathbf{if}\ \mathcal{A}\ \mathbf{then}\ (h; w)\ \mathbf{else}\ \mathbf{skip} \sqsubseteq w$$

That is to say that (0)

$$\mathbf{if}\ \mathcal{A}\ \mathbf{then}\ (h; w)\ \mathbf{else}\ \mathbf{skip} \sqsubseteq w$$

and, (1)

for any f, if $\mathbf{if}\ \mathcal{A}\ \mathbf{then}\ (h; f)\ \mathbf{else}\ \mathbf{skip} \sqsubseteq f$ then $w \sqsubseteq f$

From this definition, we can show the above laws for iteration.