Program Synthesis Example 2

Find the quotient $z$ of real numbers $0 \leq x < y \leq 1$ to within tolerance $\Delta$ (without using division, of course).

$$\{0 \leq x < y \leq 1\} S \{z \leq x/y < z + \Delta\}$$

Programming strategy:

$$\{0 \leq x < y \leq 1\}$$

begin

**make a guess**

while $\neg$good-enough

**this is a reasonable guess**

do

**improve the guess**

do

end

$$\{z \leq x/y < z + \Delta\}$$
**Technique:** Generalize a constant ($\Delta$) to a variable ($d$).

Introduce a variable $d$ to represent the known accuracy of the current guess.

$$\text{INV} \equiv z \leq x/y < z + d$$

If $d \leq \Delta$ then the postcondition is satisfied, so let $d > \Delta$ be the loop test.

**Establish the invariant:** Since $0 \leq x < y \leq 1$ initially, we know that $0 < x/y < 1$. Hence, initializing $d$ to 1 and $z$ to 0 assures the invariant.

\[
\begin{align*}
\{0 \leq x < y \leq 1\} \\
\text{begin} \\
z &:= 0; \\
d &:= 1; \\
\text{while } d > \Delta \\
\{z < x/y < z + d\} \\
\text{do} \\
\text{improve } z \text{ and } d \\
\text{end} \\
\{z \leq x/y < z + \Delta\}
\end{align*}
\]
Idea: Let’s try a binary search, that is, reduce $d$ by half each time through the loop. Assuming that $z \leq x/y < z + d$, there are two possibilities:

- If $z + \frac{1}{2}d > x/y$, then $z \leq x/y < z + \frac{1}{2}d$
- otherwise, $z + \frac{1}{2}d \leq x/y < z + d$

Either way, we know the quotient to within $\frac{1}{2}d$, so

\[
\{0 \leq x < y \leq 1\} \\
\text{begin}
\begin{align*}
  z &:= 0; \\
  d &:= 1;
\end{align*}
\text{while } d > \Delta \text{ do}
\begin{align*}
  \{z < x/y < z + d\} \\
  \text{begin}
  \begin{align*}
    \text{if } z + \frac{1}{2}d > x/y \\
    \text{then } &z := z \\
    \text{else } &z := z + \frac{1}{2}d;
  \end{align*}
  \quad d := \frac{1}{2}d
\end{align*}
\text{end}
\{z \leq x/y < z + \Delta\}
Replace the distracting \( z := z \) by \textbf{skip}.

Fix the “cheat” in the \texttt{if}-test. Get rid of the division by multiplying through by \( y \).

\[
\{ 0 \leq x < y \leq 1 \} \\
beg\begin{align*} 
& z := 0; \\
& d := 1; \\
& \text{while } d > \Delta \text{ do} \\
& \quad \{ z < x/y < z + d \} \\
& \quad \text{begin} \\
& \quad \quad \text{if } \textcolor{red}{zy + \frac{1}{2}dy > x} \text{ then skip} \\
& \quad \quad \text{else } z := z + \frac{1}{2}d; \\
& \quad \quad d := \frac{1}{2}d \\
& \text{end} \\
\text{end} \\
\end{align*}
\]
\[
\{ z \leq x/y < z + \Delta \}
\]
The test seems too costly.

**Technique: strength reduction.** Introduce “trailer variables” \( u \) and \( v \) to hold the multiplications, subject to invariants

\[
\begin{align*}
I1: & \quad u = zy \\
I2: & \quad v = \frac{1}{2}dy
\end{align*}
\]

Now we are obligated to maintain these new invariants. Let \( d' \), \( z' \), \( u' \), and \( v' \) denote the “next” values of \( d \), \( z \), \( u \), and \( v \), respectively. Depending on the test,

**Case A: \( u + v > x \):**

\[
\begin{align*}
d' &= \frac{1}{2}d \\
z' &= z \\
u' &= z' \cdot y = z \cdot y \quad (\text{I1}) = u \\
v' &= \frac{1}{2}(d') \cdot y = \frac{1}{2}dy \quad (\text{I1}) = \frac{1}{2}v
\end{align*}
\]

**Case B: \( u + v \leq x \):**

\[
\begin{align*}
d' &= \frac{1}{2}d \\
z' &= z + \frac{1}{2}d = z + d' \\
u' &= z' \cdot y = (z + \frac{1}{2}d)y = zy + \frac{1}{2}dy \quad (\text{I1, I2}) = u + v \\
v' &= \frac{1}{2}(d') \cdot y = \frac{1}{2}dy \quad (\text{I2}) = \frac{1}{2}v
\end{align*}
\]
\{0 \leq x < y \leq 1\}

begin
z := 0;
d := 1;
u := 0;
v := \frac{1}{2}y;

while d > \Delta do
\{z < x/y < z + d \land u = zy \land v = \frac{1}{2}dy\}
begin
d := \frac{1}{2}d;
if u + v > x
    then skip
    else begin z := z + d; u := u + v; end;
v := \frac{1}{2}v
end
\end
\{z \leq x/y < z + \Delta\}

This is called *Wensley’s algorithm* [Wensley 58] for real number division, specifically computing the fractional parts of floating point representations.

Since it reduces integer division to addition and divide-by-two, it is a candidate for use in a typical processor, which will have instructions for these operations.
However, this algorithm is not ideally suited for implementation in hardware because addition takes too long ($O(\log n)$ for $n$-bit operands). Later, we will see a series of optimizations that improve on Wensley’s algorithm that are used in floating point hardware.

\[
\{0 \leq x < y \leq 1\}
\begin{align*}
&\text{begin} \\
&z := 0; \\
d := 1; \\
u := 0; \\
v := \frac{1}{2} y; \\
\text{while } d > \Delta \text{ do} \\
&\{z < x/y < z+d \land u = zy \land v = \frac{1}{2} dy\} \\
&\text{begin} \\
&d := \frac{1}{2} d; \\
&\text{if } u+v > x \\
&\quad \text{then skip} \\
&\quad \text{else begin } z := z+d; u := u+v; \text{end;} \\
&v := \frac{1}{2} v \\
&\text{end} \\
&\text{end} \\
&\{z \leq x/y < z+\Delta\}
\end{align*}
\]