Lecture #42: More on Program Verification

• Rules so far:

\[
\begin{align*}
\{P\} & \text{ pass } \{P\} \\
\{P\} & \Rightarrow R, \{R\} \Rightarrow \{Q\} \\
R & \Rightarrow Q, \{P\} \Rightarrow \{R\}, \text{ } \{P\} & \Rightarrow \{Q\} \\
\{P\} & \Rightarrow \{R\}, \{R\} \Rightarrow \{Q\} \\
\{P\} & \Rightarrow x = E, \{P\} \Rightarrow \{Q\} \\
\{P\} & \Rightarrow \{Q\} \\
\{P\} & \Rightarrow \{Q\} \\
\{P\} & \Rightarrow \{Q\} \\
\end{align*}
\]

• We’re assuming that expressions (as opposed to statements) have no side-effects.

• And now come the hard ones…

Loops

• Simple while loops take some invention.

• Classical technique is to invent a loop invariant—an assertion that describes how things look at the start of an arbitrary iteration.

• For example,

\[
\begin{align*}
\text{while } i < \text{len}(A): & \quad (A[0..i] \text{ is sorted}) \iff \text{Loop invariant} \\
& \quad \text{code to swap } A[i] \text{ with largest element in } A[0..i+1] \\
i & \leftarrow 1 \\
\end{align*}
\]

• End up with a rule like this:

\[
\begin{align*}
P & \Rightarrow \text{while } C: S \Rightarrow \{Q\} \\
\{P\} & \Rightarrow I \wedge C \Rightarrow \{Q\} \\
\{P\} & \Rightarrow I \wedge \text{not } C \Rightarrow \{Q\} \\
\end{align*}
\]

Example

• We want demonstrate the following ("The Russian Peasant's Algorithm"), ignoring integer overflow:

\[
\begin{align*}
\{ n \geq 0 \land x \neq 0 \} \\
i = n; \ p = x; \ y = 1 \\
\text{while } i > 0: & \\
& \quad \text{if } i \mod 2 \equiv 1: \ y = y \times p \\
& \quad \text{ } \quad i = i/2; \ p = p \times p \\
\{ y = x^n \} \\
\end{align*}
\]

• Using the assignment rule three times, must show that

\[
\begin{align*}
n \geq 0 \land x \neq 0 & \Rightarrow n = n \geq 0 \land x = x \neq 0 \wedge 1 = 1 \\
\end{align*}
\]

…which is pretty obvious.

Getting to the Loop

• First, let’s get an assertion at the beginning of the while loop to make it more convenient to use the rule:

\[
\begin{align*}
\{ n \geq 0 \land x \neq 0 \} \\
i = n; \ p = x; \ y = 1 \\
\text{while } i > 0: & \\
& \quad \text{if } i \mod 2 \equiv 1: \ y = y \times p \\
& \quad \quad i = i/2; \ p = p \times p \\
\{ y = x^n \} \\
\end{align*}
\]
Invent a Loop Invariant

- The idea behind this loop was to use the fact that \( a^{2b} = a \cdot (a^2)^b \).
- At the beginning of each iteration, we are left with computing \( p^i \).
- Suggesting this invariant:

\[
\begin{align*}
\{ & n \geq 0 \land x \neq 0 \\
i & = n; p = x; y = 1 \\
i & = n \geq 0 \land p = x \neq 0 \land y = 1 \\
\text{while } & i > 0: \\& \quad \{ i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \} \\
& \quad \text{if } i \mod 2 = 1; y = y \cdot p \\
& \quad i = i/2; p = p \cdot p \\
& \{ y = x^n \}
\end{align*}
\]

Establish the invariant

\[
\begin{align*}
\{ & n \geq 0 \land x \neq 0 \\
i & = n; p = x; y = 1 \\
i & = n \geq 0 \land p = x \neq 0 \land y = 1 \\
\text{while } & i > 0: \\& \quad \{ i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \} \\
& \quad \text{if } i \mod 2 = 1; y = y \cdot p \\
& \quad i = i/2; p = p \cdot p \\
& \{ y = x^n \}
\end{align*}
\]

According to the while rule, must first show

\[
\begin{align*}
i & = n \geq 0 \land p \neq 0 \land y = 1 \\
\Rightarrow & \quad i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \\
\equiv & \quad i = n \geq 0 \land p = x \neq 0 \land y = 1 \Rightarrow n \geq 0 \land x \neq 0 \land x^n \cdot 1 = x^n
\end{align*}
\]

...which is obvious.

Does the Invariant give us the desired result?

\[
\begin{align*}
\{ & n \geq 0 \land x \neq 0 \\
i & = n; p = x; y = 1 \\
i & = n \geq 0 \land p = x \neq 0 \land y = 1 \\
\text{while } & i > 0: \\& \quad \{ i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \} \\
& \quad \text{if } i \mod 2 = 1; y = y \cdot p \\
& \quad i = i/2; p = p \cdot p \\
& \{ y = x^n \}
\end{align*}
\]

Last part of while rule requires that we show that when the loop condition is false, invariant implies conclusion:

\[
i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \land i \leq 0 \Rightarrow y = x^n
\]

\[
\equiv \quad \text{since only 0 is both } \geq 0 \text{ and } \leq 0
\]

\[
i = 0 \land p \neq 0 \land p^0 \cdot y = x^n \Rightarrow y = x^n
\]

Again, pretty clear (if tedious).

Invariance of the Invariant

\[
\begin{align*}
\{ & n \geq 0 \land x \neq 0 \\
i & = n; p = x; y = 1 \\
i & = n \geq 0 \land p = x \neq 0 \land y = 1 \\
\text{while } & i > 0: \\& \quad \{ i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \} \\
& \quad \text{if } i \mod 2 = 1; y = y \cdot p \\
& \quad i = i/2; p = p \cdot p \\
& \{ y = x^n \}
\end{align*}
\]

- Finally, need to show that if \( i > 0 \), executing the loop preserves the invariant.
- Applying assignment and if rules, we end up having to demonstrate the following (which I leave to you (:-)):

\[
i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \land i > 0 \land i \mod 2 = 1
\]

\[
\Rightarrow |i/2| \geq 0 \land p^{2[i/2]} \cdot y \cdot p = x^n
\]

and

\[
i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \land i > 0 \land i \mod 2 \neq 1
\]

\[
\Rightarrow |i/2| \geq 0 \land p^{2[i/2]} \cdot y = x^n
\]
An Application

- Very tedious, as you can see, and therefore error-prone.
- Even worse, turns programs into really complicated theorems, but theorem provers are not up to the task.
- **Proof checkers**, on the other hand, are pretty easy to build (give me an alleged formal proof, and I'll check it).
- One application: **proof-carrying code**:
  - Untrusted party provides, e.g., device driver, with an alleged proof that the driver adheres to your system's requirements of what not to mess with.
  - Your (trusted) proof checker checks that their proof actually comes from their program and is correct.