Lecture #42: More on Program Verification

• Rules so far:

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

\[
\begin{align*}
P \Rightarrow R, \{R\} S \{Q\} \\
\{P\} S \{Q\}
\end{align*}
\]

\[
\begin{align*}
P \Rightarrow Q, \{P\} S \{R\} \\
\{P\} \text{ if } C: S1; \text{ else: } S2 \{Q\}
\end{align*}
\]

\[
\begin{align*}
\{P \land C\} S1 \{Q\}, \{P \land \text{not } C\} S2 \{Q\} \\
\{P\} S1;S2 \{Q\}
\end{align*}
\]

• 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,
  ```python
  while i < len(A):
    { A[0..i] is sorted } ⇐ Loop invariant
    code to swap A[i] with largest element in A[0..i+1]
    i += 1
  ```

• End up with a rule like this:

  \[
  P \Rightarrow I, \{ I \land C \} \quad S \quad \{ I \}, \quad I \land \text{not } C \Rightarrow Q
  \]

  \[
  \{ P \} \quad \text{while } C: \quad S \quad \{ Q \}
  \]
Example

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

\[
\{ n \geq 0 \land x \neq 0 \}
\]

\[
i = n; \ p = x; \ y = 1
\]

\[
\text{while } i > 0:\n\]

\[
\text{if } i \% 2 == 1: \ y = y*p
\]

\[
i = i/2; \ p = p*p
\]

\[
\{ y = x^n \}
\]
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:

\[
\{ n \geq 0 \land x \neq 0 \} \\
i = n; \ p = x; \ y = 1 \\
\{ i = n \geq 0 \land p = x \neq 0 \land y = 1 \} \\
\text{while } i > 0:\ \\
\quad \text{if } i \ % \ 2 == 1: \ y = y*p \\
\quad i = i/2; \ p = p*p \\
\{ y = x^n \}
\]

• Using the assignment rule three times, must show that

\[
n \geq 0 \land x \neq 0 \Rightarrow n = n \geq 0 \land x = x \neq 0 \land 1 = 1
\]

... which is pretty obvious.
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:

\[
\begin{array}{l}
\{ n \geq 0 \land x \neq 0 \} \\
i = n; \ p = x; \ y = 1 \\
\{ i = n \geq 0 \land p = x \neq 0 \land y = 1 \} \\
\textbf{while} \ i > 0: \{ i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \} \\
\quad \textbf{if} \ i \mod 2 == 1: \ y = y \cdot p \\
\quad i = i/2; \ p = p \cdot p \\
\{ y = x^n \}
\end{array}
\]
Establish the invariant

\{ 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 \}\}
while i > 0: \{ i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \}\}
   if i \% 2 == 1: y = y*p
   i = i/2; p = p*p
\{ y = x^n \}\}

According to the while rule, must first show

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

\ldots which is obvious.
Does the Invariant give us the desired result?

{ \( n \geq 0 \land x \neq 0 \) }
\( i = n; \ p = x; \ y = 1 \)

{ \( i = n \geq 0 \land p = x \neq 0 \land y = 1 \) }
while \( i > 0 \):
{ \( i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \) }
  if \( i \% 2 == 1 \): \( y = y \cdot p \)
  \( i = i/2; \ p = p^*p \)
{ \( y = x^n \) }

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

\[
i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \land i \leq 0 \implies y = x^n
\]
\[
\equiv \text{since only 0 is both } \geq 0 \text{ and } \leq 0
\]
\[
i = 0 \land p \neq 0 \land p^0 \cdot y = x^n \implies y = x^n
\]

Again, pretty clear (if tedious).
Invariance of the Invariant

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

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

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

and

\[
i \geq 0 \land p \neq 0 \land p^i \cdot y = x^n \land i > 0 \land i \mod 2 \neq 1
\Rightarrow [i/2] \geq 0 \land p^2 \neq 0 \land p^{2[i/2]} \cdot y = x^n
\]
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.