Lecture #42: More on Program Verification

- Rules so far:

$$\{P\}\ \textsf{pass}\ \{P\}$$

$$\frac{P \Rightarrow R,\ \{R\}\ \mathcal{S}\ \{Q\}}{\{P\}\ \mathcal{S}\ \{Q\}}$$

$$\frac{R \Rightarrow Q,\ \{P\}\ \mathcal{S}\ \{R\}}{\{P\}\ \mathcal{S}\ \{Q\}}$$

$$\frac{\{P \wedge C\}\ \textsf{S1}\ \{Q\},\ \ \{P \wedge \textbf{not}\ C\}\ \textsf{S2}\ \{Q\}}{\{P\}\ \textsf{if C: S1; else: S2}\ \{Q\}}$$

$$\frac{P \Rightarrow Q[x \leftarrow E]}{\{P\}\ \textsf{x = E}\ \{Q\}}$$

$$\frac{\{P\}\ \textsf{S1}\ \{R\}, \{R\}\ \textsf{S2}\ \{Q\}}{\{P\}\ \textsf{S1;S2}\ \{Q\}}$$

- 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,

```
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:

$$\frac{P \Rightarrow \mathcal{I}, \{\mathcal{I} \wedge C\}\ \ \mathsf{S}\ \ \{\mathcal{I}\},\ \ \mathcal{I} \wedge \textbf{not}\ C \Rightarrow Q}{\{P\}\ \ \text{while } C\text{: } \mathsf{S}\ \ \{Q\}}$$

Example

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

{ $n \geq 0 \wedge x \neq 0$ }

```
i = n; p = x; y = 1
while i > 0:
   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:

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

- Using the assignment rule three times, must show that

$$n \geq 0 \wedge x \neq 0 \Rightarrow n = n \geq 0 \wedge x = x \neq 0 \wedge 1 = 1$$

...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:

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

Establish the invariant

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

According to the **while** rule, must first show

$$i = n \geq 0 \wedge p = x \neq 0 \wedge y = 1 \Rightarrow i \geq 0 \wedge p \neq 0 \wedge p^i \cdot y = x^n$$

$$\equiv$$

$$i = n \geq 0 \wedge p = x \neq 0 \wedge y = 1 \Rightarrow n \geq 0 \wedge x \neq 0 \wedge x^n \cdot 1 = x^n$$

...which is obvious.

Does the Invariant give us the desired result?

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

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

$$
\begin{aligned}
& i \geq 0 \wedge p \neq 0 \wedge p^i \cdot y = x^n \wedge i \leq 0 \Rightarrow y = x^n \\
\equiv \quad & \textit{since only 0 is both } \geq 0 \textit{ and } \leq 0 \\
& i = 0 \wedge p \neq 0 \wedge p^0 \cdot y = x^n \Rightarrow y = x^n
\end{aligned}
$$

Again, pretty clear (if tedious).

Invariance of the Invariant

```
{ n ≥ 0 ∧ x ≠ 0 }
i = n; p = x; y = 1
{ i = n ≥ 0 ∧ p = x ≠ 0 ∧ y = 1 }
while i > 0: { i ≥ 0 ∧ p ≠ 0 ∧ p^i · y = x^n }
   if i % 2 == 1: y = y*p
   i = i/2; p = p*p
{ y = x^n }
```

- 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 (:->)):

$$i \geq 0 \wedge p \neq 0 \wedge p^i \cdot y = x^n \wedge i > 0 \wedge i \bmod 2 = 1$$
$$\Rightarrow \lfloor i/2 \rfloor \geq 0 \wedge p^2 \neq 0 \wedge p^{2\lfloor i/2 \rfloor} \cdot y \cdot p = x^n$$

and

$$i \geq 0 \wedge p \neq 0 \wedge p^i \cdot y = x^n \wedge i > 0 \wedge i \bmod 2 \neq 1$$
$$\Rightarrow \lfloor i/2 \rfloor \geq 0 \wedge p^2 \neq 0 \wedge p^{2\lfloor i/2 \rfloor} \cdot y = x^n$$

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 alledged 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.