Lecture 11
More Hoare Logic
Overview

Last time
- Hoare Logic

Today
- “Decorated” programs
- Weakest Preconditions
Review: Hoare Logic

\[ \vdash \{ P \} \text{skip} \{ P \} \text{ Skip} \quad \vdash \{ P[a/x] \} x := a \{ P \} \text{ Assign} \]

\[ \vdash \{ P \} c_1 \{ R \} \quad \vdash \{ R \} c_2 \{ Q \} \quad \vdash \{ P \} c_1 ; c_2 \{ Q \} \text{ Seq} \]

\[ \vdash \{ P \land b \} c_1 \{ Q \} \quad \vdash \{ P \land \neg b \} c_2 \{ Q \} \quad \vdash \{ P \} \text{ if } b \text{ then } c_1 \text{ else } c_2 \{ Q \} \text{ If} \]

\[ \vdash \{ P \land b \} c \{ P \} \quad \vdash \{ P \} \text{ while } b \text{ do } c \{ P \land \neg b \} \text{ While} \]

\[ \models P \Rightarrow P' \quad \vdash \{ P' \} c \{ Q' \} \quad \models Q' \Rightarrow Q \text{ Consequence} \]
Decorated Programs

Observation: Once we’ve identified loop invariants and uses of consequence, the structure of a Hoare logic is determined!

Notation: Can write proofs by “decorating” programs with:

- A precondition ($\{P\}$)
- A postcondition ($\{Q\}$)
- Invariants ($\{l\} \textbf{while } b \textbf{ do } c$)
- Uses of consequence $\{R\} \Rightarrow \{S\}$
- Assertions between sequences $c_1; \{T\}c_2$

A decorated program describes a valid Hoare logic proof if the rest of the proof tree’s structure is implied. (Caveats: Invariants are constrained, etc.)
Example: Decorated Factorial

\[
\{ x = n \land n > 0 \}
\]

\[
y := 1;
\]

**while** \( x > 0 \) **do** { 
\[
y := y \times x;
\]
\[
x := x - 1
\]
}

\[
\{ y = n! \}
\]
Example: Decorated Factorial

\[
\{ x = n \land n > 0 \} \Rightarrow
\{ 1 = 1 \land x = n \land n > 0 \}
\]

\[
y := 1;
\{ y = 1 \land x = n \land n > 0 \} \Rightarrow
\{ y \times x! = n! \land x \geq 0 \}
\]

**while** \( x > 0 \) do

\[
\{ y \times x! = n! \land x > 0 \land x \geq 0 \} \Rightarrow
\{ y \times x \times (x - 1)! = n! \land (x - 1) \geq 0 \}
\]

\[
y := y \times x;
\{ y \times (x - 1)! = n! \land (x - 1) \geq 0 \}
\]

\[
x := x - 1
\]

\[
\{ y \times x! = n! \land x \geq 0 \}
\]

\[
\}
\]

\[
\{ y \times x! = n! \land (x \geq 0) \land \neg(x > 0) \} \Rightarrow
\{ y = n! \}\]
Informal Rules for Decoration

Check whether a decorated program represents a valid proof using local consistency checks.
Informal Rules for Decoration

Check whether a decorated program represents a valid proof using local consistency checks.

For `skip`, the precondition and postcondition should be the same:

```
{P}
skip
{P}
```
Informal Rules for Decoration

For sequences, \( \{P\} c_1 \{R\} \) and \( \{R\} c_2 \{Q\} \) must be (recursively) locally consistent:

\[
\{P\}
\]

\[
c_1;
\]

\[
\{R\}
\]

\[
c_2
\]

\[
\{Q\}
\]
Informal Rules for Decoration

Assignment should use the substitution from the rule:

\[
\{P[a/x]\}
\]

\[
x := a
\]

\[
\{P\}
\]
Informal Rules for Decoration

An **if** is locally consistent when both branches are locally consistent after adding the branch condition to each:

\[
\begin{align*}
\text{if } & x \neq 5 \\
\{ P \} & \\
\text{then} & \\
\{ P \land b \} & \\
c_1 & \\
\{ Q \} & \\
\text{else} & \\
\{ P \land \neg b \} & \\
c_2 & \\
\{ Q \} & \\
\{ Q \} & \\
\end{align*}
\]
Informal Rules for Decoration

Decorate a **while** with the loop invariant:

\[
\begin{align*}
\{ P \} \\
\textbf{while } b \textbf{ do} \\
\{ P \land b \} \\
c \\
\{ P \} \\
\{ P \land \neg b \}
\end{align*}
\]
Informal Rules for Decoration

To capture the CONSEQUENCE rule, you can always write a (valid) implication:

\[
\{P\} \Rightarrow \{Q\}
\]
```plaintext
Example

{ }

while (0 < y) do (  
    x := x + 1;
    y := y - 1
)

{ }
```

The code snippet demonstrates a `while` loop that increments `x` by 1 and decrements `y` by 1 as long as `y` is greater than 0.
Example

\[ \{ x = m \land y = n \land 0 \leq n \} \]

**while** \((0 < y)\) **do** (  
  \[ x := x + 1; \]
  \[ y := y - 1 \]
)

\[ \{ x = m + n \} \]
\{x = m \land y = n \land 0 \leq n\} \Rightarrow \\
\{l\}

while \ (0 < y) \ do \ ( \\
\{l \land 0 < y\} \Rightarrow \\
\{l[y - 1/y][x + 1/x]\}
\ x := x + 1; \\
\{l[y - 1/y]\}
\ y := y - 1 \\
\{l\}
)

\{l \land 0 \not< y\} \Rightarrow \\
\{x = m + n\}

Where \ l \ is \ (x = m + n - y) \land 0 \leq y.
Example

\{
\}

**while** (x \neq 0) do ( 
  x := x - 1 
)

\{
\}
Example

\{ \textbf{true} \}

\textbf{while} (x \neq 0) \textbf{do} (\n\quad x := x - 1
\)

\{ x = 0 \}
Example

{ }

y := 1

while (0 < x) do ( x := x - 1; y := y * 2 )

{ }

{ }

{ }
Example

\{x = n \land 0 \leq n\}

y := 1

while (0 < x) do (  
    x := x - 1;
    x := x - 1;
    y := y \ast 2
)

\{y = 2^n\}