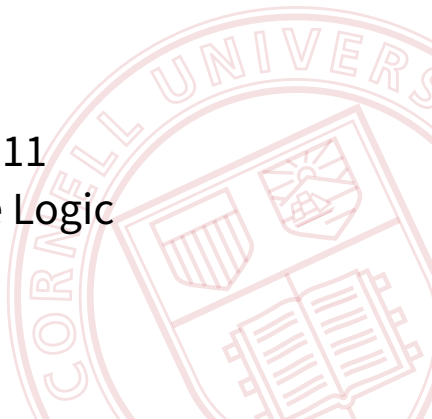# CS 4110

Programming Languages & Logics

Lecture 11
More Hoare Logic

Overview

Last time
- Hoare Logic

Today
- "Decorated" programs
- Weakest Preconditions

$$\overline{\vdash \{P\} \textbf{ skip } \{P\}} \text{ Skip} \qquad \overline{\vdash \{P[a/x]\}\, x := a\, \{P\}} \text{ Assign}$$

$$\frac{\vdash \{P\}\, c_1\, \{R\} \qquad \vdash \{R\}\, c_2\, \{Q\}}{\vdash \{P\}\, c_1; c_2\, \{Q\}} \text{ Seq}$$

$$\frac{\vdash \{P \wedge b\}\, c_1\, \{Q\} \qquad \vdash \{P \wedge \neg b\}\, c_2\, \{Q\}}{\vdash \{P\} \textbf{ if } b \textbf{ then } c_1 \textbf{ else } c_2\, \{Q\}} \text{ If}$$

$$\frac{\vdash \{P \wedge b\}\, c\, \{P\}}{\vdash \{P\} \textbf{ while } b \textbf{ do } c\, \{P \wedge \neg b\}} \text{ While}$$

$$\frac{\models P \Rightarrow P' \qquad \vdash \{P'\}\, c\, \{Q'\} \qquad \models Q' \Rightarrow Q}{\vdash \{P\}\, c\, \{Q\}} \text{ Consequence}$$

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 ($\{I\}$**while** b **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

$$\{x = n \land n > 0\}$$
$$y := 1;$$
$$\textbf{while } x > 0 \textbf{ do } \{$$
$$y := y * x;$$
$$x := x - 1$$
$$\}$$
$$\{y = n!\}$$

Example: Decorated Factorial

$\{x = n \land n > 0\} \Rightarrow$
$\{1 = 1 \land x = n \land n > 0\}$
$y := 1;$
$\{y = 1 \land x = n \land n > 0\} \Rightarrow$
$\{y * x! = n! \land x \geq 0\}$
while $x > 0$ **do** $\{$
    $\{y * x! = n! \land x > 0 \land x \geq 0\} \Rightarrow$
    $\{y * x * (x - 1)! = n! \land (x - 1) \geq 0\}$
    $y := y * x;$
    $\{y * (x - 1)! = n! \land (x - 1) \geq 0\}$
    $x := x - 1$
    $\{y * x! = n! \land x \geq 0\}$
$\}$
$\{y * x! = n! \land (x \geq 0) \land \neg(x > 0)\} \Rightarrow$
$\{y = n!\}$

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:

$$
\begin{array}{l}
\{P\} \\
c_1; \\
\{R\} \\
c_2 \\
\{Q\}
\end{array}
$$

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:

$$\{P\}$$
if b **then**
$$\{P \wedge b\}$$
$$c_1$$
$$\{Q\}$$
else
$$\{P \wedge \neg b\}$$
$$c_2$$
$$\{Q\}$$
$$\{Q\}$$

Informal Rules for Decoration

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

$$\{P\}$$
$$\textbf{while } b \textbf{ do}$$
$$\{P \wedge b\}$$
$$c$$
$$\{P\}$$
$$\{P \wedge \neg b\}$$

Informal Rules for Decoration

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

$$\{P\} \Rightarrow$$
$$\{Q\}$$

Example

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

Example

$\{x = m \ \wedge \ y = n \wedge 0 \le n\}$
while $(0 < y)$ **do** (
  $x := x + 1;$
  $y := y - 1$
)
$\{x = m + n\}$

Example

$\{x = m \land y = n \land 0 \le n\} \Rightarrow$
$\{I\}$
while $(0 < y)$ **do** (
   $\{I \land 0 < y\} \Rightarrow$
   $\{I[y - 1/y][x + 1/x]\}$
   $x := x + 1;$
   $\{I[y - 1/y]\}$
   $y := y - 1$
   $\{I\}$
)
$\{I \land 0 \not< y\} \Rightarrow$
$\{x = m + n\}$

Where I is $(x = m + n - y) \land 0 \le y$.

Example

$\{\quad\}$
while $(x \neq 0)$ **do** (
  $x := x - 1$
)
$\{\quad\}$

Example

$\{\textbf{true}\}$
while $(x \neq 0)$ **do** (
  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;
  y := y ∗ 2
)
$\{y = 2^n\}$