Hoare Logic 2

slides by Chris Osborn

Hoare Triple

P { …code…} Q

$$\frac{}{P[e/x]\ \{\ x := e\ \}\ P}$$

$$\frac{P\ \{\ C_1\ \}\ R \qquad R\ \{\ C_2\ \}\ Q}{P\ \{\ C_1; C_2\ \}\ Q}$$

$$\frac{P \wedge b\ \{\ C_1\ \}\ Q \qquad P \wedge \neg\, b\ \{\ C_2\ \}\ Q}{P\ \{\ \text{if } b \text{ then } C_1 \text{ else } C_2\ \}\ Q}$$

While Rule

$$\frac{P \wedge b \{ C \} P}{P \{\text{While } b\ C\} P \wedge \neg b}$$

(P is a **loop invariant**)

Rule of Consequence

$$\frac{P \rightarrow P' \qquad P' \{ C \} Q' \qquad Q' \rightarrow Q}{P \{ C \} Q}$$

Sample Proofs

- sum of n
- fibonacci
- list append
- list reverse
- termination

Sum of n

$x = 0 \;\&\; y = 0$

{

  While $y < n$

    $y := y + 1;$

    $x := x + y$

}

$x = 1 + \ldots + n$

$P \equiv x = 1 + \ldots + y \wedge y \leq n$

$x = 0 \wedge y = 0 \Rightarrow x = 1 + \ldots + y \wedge y \leq n$ ✓

$x = 1 + \ldots + y \wedge y \leq n \wedge \neg(y < n) \Rightarrow x = 1 + \ldots + n$ ✓

$x = 1 + \ldots + y \wedge y \leq n \wedge y < n \Rightarrow$? ✓

$x + y + 1 = 1 + \ldots + y + 1 \wedge y + 1 \leq n$

$\{y := y + 1\}$ $x + y = 1 + \ldots + y \wedge y \leq n$

$\{x := x + y\}$ $x = 1 + \ldots + y \wedge y \leq n$

? $\{y := y + 1; x := x + y\}$ $x = 1 + \ldots + y \wedge y \leq n$

$x = 1 + \ldots + y \wedge y \leq n \wedge y < n$ $\{y := y + 1; x := x + y\}$ $x = 1 + \ldots + y \wedge y \leq n$

$x = 1 + \ldots + y \wedge y \leq n$ {While $y < n$ ...} $x = 1 + \ldots + y \wedge y \leq n \wedge \neg(y < n)$

$x = 0 \wedge y = 0$ {While ...} $x = 1 + \ldots + n$

Fibonacci

$x = 0 \;\&\; y = 1 \;\&\; z = 1 \;\&\; 1 \leq n$

{

  While $z < n$

    $y := x + y;$

    $x := y - x;$

    $z := z + 1$

}

$y = \text{fib } n$

$P \equiv y = \text{fib } z \wedge x = \text{fib } (z\text{-}1) \wedge z \leq n$

$x = 0 \wedge y = 1 \wedge z = 0 \wedge 1 \leq n \rightarrow y = fib\ z \wedge x = fib\ (z-1) \wedge z \leq n$ ✓

$y = fib\ z \wedge x = fib\ (z-1) \wedge z \leq n \wedge \neg(z < n) \rightarrow y = fib\ n$ ✓

$y = fib\ z \wedge x = fib\ (z-1) \wedge z \leq n \wedge z < n \rightarrow$? ✓

$x+y = fib\ (z+1) \wedge x+y-x = fib\ (z+1-1) \wedge z + 1 \leq n$

$\{y := x + y\}$ $y = fib\ (z+1) \wedge y-x = fib\ (z+1-1) \wedge z + 1 \leq n$

$\{x := y - x\}$ $y = fib\ (z+1) \wedge x = fib\ (z+1-1) \wedge z + 1 \leq n$

$\{z := z + 1\}$ $y = fib\ z \wedge x = fib\ (z-1) \wedge z \leq n$

? $\{y := x + y;\ x := y - x;\ z := z + 1\}$ $y = fib\ z \wedge x = fib\ (z-1) \wedge z \leq n$

$y = fib\ z \wedge x = fib\ (z-1) \wedge z \leq n \wedge z < n$ $\{y := x + y;\ x := y - x;\ z := z + 1\}$ $y = fib\ z \wedge x = fib\ (z-1) \wedge z \leq n$

$y = fib\ z \wedge x = fib\ (z-1) \wedge z \leq n$ {While $z < n$ ...} $y = fib\ z \wedge x = fib\ (z-1) \wedge z \leq n \wedge \neg(z < n)$

$x = 0 \wedge y = 1 \wedge z = 0 \wedge 1 \leq n$ {While ...} $y = fib\ n$

List length

$x = lst \;\&\; y = 0$

$P \equiv len\ lst = y + len\ x$

```
{
  While x ≠ []
    x := tl x;
    y := y + 1
}
```

$y = len\ lst$

$x = lst \wedge y = 0 \Rightarrow len\ lst = y + len\ x$ ✓

$len\ lst = y + len\ x \wedge \neg(x \neq []) \Rightarrow y = len\ lst$ ✓

$len\ lst = y + len\ x \wedge x \neq [] \Rightarrow$? ✓

$len\ lst = y + 1 + len(tl\ x)$

$\{x := tl\ x\}$ $len\ lst = y + 1 + len\ x$

$\{y := y + 1\}$ $len\ lst = y + len\ x$

? $\{x := tl\ x;\ y := y + 1\}$ $len\ lst = y + len\ x$

$len\ lst = y + len\ x \wedge x \neq []$ $\{x := tl\ x;\ y := y + 1\}$ $len\ lst = y + len\ x$

$len\ lst = y + len\ x$ {While $x \neq []$ ...} $len\ lst = y + len\ x \wedge \neg(x \neq [])$

$x = lst \wedge y = 0$ {While ...} $y = len\ lst$

List reverse

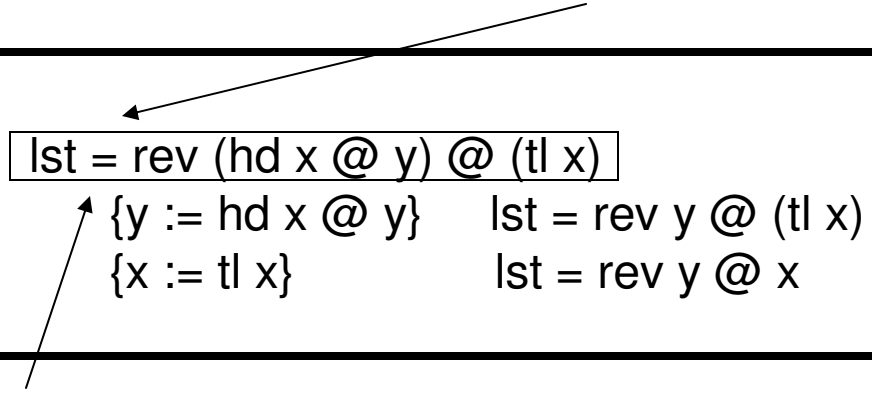
```
x = lst & y = []
{
  While x ≠ []
    y := hd x :: y;
    x := tl x
}
y = rev lst
```

P ≡ lst = rev y @ x

$x = lst \wedge y = [] \Rightarrow lst = rev\ y\ @\ x$ ✓

$lst = rev\ y\ @\ x \wedge \neg(x \neq []) \Rightarrow y = rev\ lst$ ✓

$lst = rev\ y\ @\ x \wedge x \neq [] \Rightarrow$? ✓



? {y := hd x @ y; x := tl x} lst = rev y @ x

---

lst = rev y @ x ∧ x ≠ [] {y := hd x @ y; x := tl x} lst = rev y @ x

---

lst = rev y @ x {While x ≠ [] ...} lst = rev y @ x ∧ ¬(x ≠ [])

---

x = lst ∧ y = [] {While ...} y = rev lst