READABLE PROOFS IN HOARE LOGIC
(AND SEPARATION LOGIC)

John C. Reynolds
Carnegie Mellon University

ETAPS 2009
York, March 25, 2009 (revised April 6)

Our Thesis

Formal proofs of program specifications (more precisely, proofs that specifications follow from their verification conditions) are best communicated by *annotated specifications* (sometimes called *proof outlines*), in which intermediate assertions and other notations are interspersed within the specification.

These annotated specifications can be defined by inference rules and mechanically translated into conventional formal proofs.

A Program for Fast Division

$$
\begin{array}{l}
\{\mathsf{x} \geq 0 \wedge \mathsf{y} > 0\} \\
\mathbf{newvar}\ \mathsf{n} := 0\ \mathbf{in}\ \mathbf{newvar}\ \mathsf{z} := \mathsf{y}\ \mathbf{in} \\
\big(\mathbf{while}\ \mathsf{z} \leq \mathsf{x}\ \mathbf{do}\ (\mathsf{n} := \mathsf{n} + 1\ ;\ \mathsf{z} = \mathsf{z} \times 2)\ ; \\
\mathsf{q} := 0\ ;\ \mathsf{r} := \mathsf{x}\ ; \\
\mathbf{while}\ \mathsf{n} \neq 0\ \mathbf{do} \\
\quad \big(\mathsf{n} := \mathsf{n} - 1\ ;\ \mathsf{z} := \mathsf{z} \div 2\ ;\ \mathsf{q} := \mathsf{q} \times 2\ ; \\
\quad \mathbf{if}\ \mathsf{z} \leq \mathsf{r}\ \mathbf{then}\ \mathsf{q} := \mathsf{q} + 1\ ;\ \mathsf{r} := \mathsf{r} - \mathsf{z}\ \mathbf{else}\ \mathbf{skip}\big)\big) \\
\{\mathsf{x} = \mathsf{q} \times \mathsf{y} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{y}\}
\end{array}
$$

A Formal Proof for Fast Division

The invariants:

$$I_0 \stackrel{\text{def}}{=} \mathsf{z} = \mathsf{y} \times 2^{\mathsf{n}} \wedge \mathsf{n} \geq 0 \wedge \mathsf{x} \geq 0 \wedge \mathsf{y} > 0$$

$$I_1 \stackrel{\text{def}}{=} \mathsf{x} = \mathsf{q} \times \mathsf{z} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{z} \wedge \mathsf{z} = \mathsf{y} \times 2^{\mathsf{n}} \wedge \mathsf{n} \geq 0$$

The proof:

1. $(\mathsf{x} \geq 0 \wedge \mathsf{y} > 0) \Rightarrow (\mathsf{y} = \mathsf{y} \times 2^{0} \wedge 0 \geq 0 \wedge \mathsf{x} \geq 0 \wedge \mathsf{y} > 0)$

2. $\{\mathsf{x} \geq 0 \wedge \mathsf{y} > 0\}$

   $\mathsf{n} := 0 \,;\, \mathsf{z} := \mathsf{y}$

   $\{I_0\}$ (RAS,1)

3. $(I_0 \wedge \mathsf{z} \leq \mathsf{x}) \Rightarrow$

   $(\mathsf{z} \times 2 = \mathsf{y} \times 2^{\mathsf{n}+1} \wedge \mathsf{n} + 1 \geq 0 \wedge \mathsf{x} \geq 0 \wedge \mathsf{y} > 0)$

4. $\{I_0 \wedge \mathsf{z} \leq \mathsf{x}\}$

   $\mathsf{n} := \mathsf{n} + 1 \,;\, \mathsf{z} = \mathsf{z} \times 2$

   $\{I_0\}$ (RAS,3)

5. $\{I_0\}$

   $\mathbf{while}\ \mathsf{z} \leq \mathsf{x}\ \mathbf{do}\ (\mathsf{n} := \mathsf{n} + 1 \,;\, \mathsf{z} = \mathsf{z} \times 2)$

   $\{I_0 \wedge \neg\ \mathsf{z} \leq \mathsf{x}\}$ (WH,4)

6. $(I_0 \wedge \neg\ \mathsf{z} \leq \mathsf{x}) \Rightarrow$

$$(\mathsf{x} = 0 \times \mathsf{z} + \mathsf{x} \wedge 0 \leq \mathsf{x} < \mathsf{z} \wedge \mathsf{z} = \mathsf{y} \times 2^{\mathsf{n}} \wedge \mathsf{n} \geq 0)$$

7. $\{I_0 \wedge \neg\ \mathsf{z} \leq \mathsf{x}\}$

$\mathsf{q} := 0\ ;\ \mathsf{r} := \mathsf{x}$

$\{I_1\}$ (RAS,6)

8. $(I_1 \wedge \mathsf{n} \neq 0) \Rightarrow$

$$\big(\mathsf{x} = (\mathsf{q} \times 2) \times (\mathsf{z} \div 2) + \mathsf{r} \wedge 0 \leq \mathsf{r} < (\mathsf{z} \div 2) \times 2 \wedge \mathsf{z} \div 2 = \mathsf{y} \times 2^{\mathsf{n}-1} \wedge \mathsf{n} - 1 \geq 0\big)$$

9. $\{I_1 \wedge \mathsf{n} \neq 0\}$

$\mathsf{n} := \mathsf{n} - 1\ ;\ \mathsf{z} := \mathsf{z} \div 2\ ;\ \mathsf{q} := \mathsf{q} \times 2$

$\{\mathsf{x} = \mathsf{q} \times \mathsf{z} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{z} \times 2 \wedge \mathsf{z} = \mathsf{y} \times 2^{\mathsf{n}} \wedge \mathsf{n} \geq 0\}$

(RAS,8)

10. $(\mathsf{x} = \mathsf{q} \times \mathsf{z} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{z} \times 2 \wedge \mathsf{z} = \mathsf{y} \times 2^{\mathsf{n}} \wedge \mathsf{n} \geq 0 \wedge \mathsf{z} \leq \mathsf{r}) \Rightarrow$

$$\big(\mathsf{x} = (\mathsf{q} + 1) \times \mathsf{z} + (\mathsf{r} - \mathsf{z}) \wedge 0 \leq \mathsf{r} - \mathsf{z} < \mathsf{z} \wedge \mathsf{z} = \mathsf{y} \times 2^{\mathsf{n}} \wedge \mathsf{n} \geq 0\big)$$

11. $\{\mathsf{x} = \mathsf{q} \times \mathsf{z} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{z} \times 2 \wedge \mathsf{z} = \mathsf{y} \times 2^{\mathsf{n}} \wedge \mathsf{n} \geq 0 \wedge \mathsf{z} \leq \mathsf{r}\}$

$\mathsf{q} := \mathsf{q} + 1\ ;\ \mathsf{r} := \mathsf{r} - \mathsf{z}$

$\{I_1\}$ (RAS,10)

12. $(\mathsf{x} = \mathsf{q} \times \mathsf{z} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{z} \times 2 \wedge$
$\mathsf{z} = \mathsf{y} \times 2^{\mathsf{n}} \wedge \mathsf{n} \geq 0 \wedge \neg\ \mathsf{z} \leq \mathsf{r}) \Rightarrow I_1$

13. $\{\mathsf{x} = \mathsf{q} \times \mathsf{z} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{z} \times 2 \wedge \mathsf{z} = \mathsf{y} \times 2^{\mathsf{n}} \wedge \mathsf{n} \geq 0$
$\wedge \neg\ \mathsf{z} \leq \mathsf{r}\}$
skip
$\{I_1\}$ (ISK,12)

14. $\{\mathsf{x} = \mathsf{q} \times \mathsf{z} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{z} \times 2 \wedge \mathsf{z} = \mathsf{y} \times 2^{\mathsf{n}} \wedge \mathsf{n} \geq 0\}$
if $\mathsf{z} \leq \mathsf{r}$ **then** $\mathsf{q} := \mathsf{q} + 1\ ;\ \mathsf{r} := \mathsf{r} - \mathsf{z}$ **else skip**
$\{I_1\}$ (CD,11,13)

15. $\{I_1 \wedge \mathsf{n} \neq 0\}$
$\mathsf{n} := \mathsf{n} - 1\ ;\ \mathsf{z} := \mathsf{z} \div 2\ ;\ \mathsf{q} := \mathsf{q} \times 2\ ;$
if $\mathsf{z} \leq \mathsf{r}$ **then** $\mathsf{q} := \mathsf{q} + 1\ ;\ \mathsf{r} := \mathsf{r} - \mathsf{z}$ **else skip**
$\{I_1\}$ (MSQ,9,14)

16. $\{I_1\}$
while $\mathsf{n} \neq 0$ **do**
$\big(\mathsf{n} := \mathsf{n} - 1\ ;\ \mathsf{z} := \mathsf{z} \div 2\ ;\ \mathsf{q} := \mathsf{q} \times 2\ ;$
if $\mathsf{z} \leq \mathsf{r}$ **then** $\mathsf{q} := \mathsf{q} + 1\ ;\ \mathsf{r} := \mathsf{r} - \mathsf{z}$ **else skip**$\big)$
$\{I_1 \wedge \neg\ \mathsf{n} \neq 0\}$ (WH,15)

17. $(I_1 \wedge \neg\ \mathsf{n} \neq 0) \Rightarrow (\mathsf{x} = \mathsf{q} \times \mathsf{y} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{y})$

18. $\{\mathsf{x} \geq 0 \wedge \mathsf{y} > 0\}$

$\mathsf{n} := 0\ ;\ \mathsf{z} := \mathsf{y}\ ;$

$\mathbf{while}\ \mathsf{z} \leq \mathsf{x}\ \mathbf{do}\ (\mathsf{n} := \mathsf{n} + 1\ ;\ \mathsf{z} = \mathsf{z} \times 2)\ ;$

$\mathsf{q} := 0\ ;\ \mathsf{r} := \mathsf{x}\ ;$

$\mathbf{while}\ \mathsf{n} \neq 0\ \mathbf{do}$

$\quad \big(\mathsf{n} := \mathsf{n} - 1\ ;\ \mathsf{z} := \mathsf{z} \div 2\ ;\ \mathsf{q} := \mathsf{q} \times 2\ ;$

$\quad \mathbf{if}\ \mathsf{z} \leq \mathsf{r}\ \mathbf{then}\ \mathsf{q} := \mathsf{q} + 1\ ;\ \mathsf{r} := \mathsf{r} - \mathsf{z}\ \mathbf{else\ skip}\big)$

$\{\mathsf{x} = \mathsf{q} \times \mathsf{y} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{y}\}$ (MSQ,2,5,7,15,17)

19. $\{\mathsf{x} \geq 0 \wedge \mathsf{y} > 0\}$

$\mathsf{n} := 0\ ;\ \mathbf{newvar}\ \mathsf{z} := \mathsf{y}\ \mathbf{in}$

$\big(\mathbf{while}\ \mathsf{z} \leq \mathsf{x}\ \mathbf{do}\ (\mathsf{n} := \mathsf{n} + 1\ ;\ \mathsf{z} = \mathsf{z} \times 2)\ ;$

$\mathsf{q} := 0\ ;\ \mathsf{r} := \mathsf{x}\ ;$

$\mathbf{while}\ \mathsf{n} \neq 0\ \mathbf{do}$

$\quad \big(\mathsf{n} := \mathsf{n} - 1\ ;\ \mathsf{z} := \mathsf{z} \div 2\ ;\ \mathsf{q} := \mathsf{q} \times 2\ ;$

$\quad \mathbf{if}\ \mathsf{z} \leq \mathsf{r}\ \mathbf{then}\ \mathsf{q} := \mathsf{q} + 1\ ;\ \mathsf{r} := \mathsf{r} - \mathsf{z}\ \mathbf{else\ skip}\big)\big)$

$\{\mathsf{x} = \mathsf{q} \times \mathsf{y} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{y}\}$ (DC,18)

20. $\{\mathsf{x} \geq 0 \wedge \mathsf{y} > 0\}$

$\textbf{newvar}\ \mathsf{n} := 0\ \textbf{in}\ \textbf{newvar}\ \mathsf{z} := \mathsf{y}\ \textbf{in}$

$\big(\textbf{while}\ \mathsf{z} \leq \mathsf{x}\ \textbf{do}\ (\mathsf{n} := \mathsf{n} + 1\ ;\ \mathsf{z} = \mathsf{z} \times 2)\ ;$

$\mathsf{q} := 0\ ;\ \mathsf{r} := \mathsf{x}\ ;$

$\textbf{while}\ \mathsf{n} \neq 0\ \textbf{do}$

$\quad \big(\mathsf{n} := \mathsf{n} - 1\ ;\ \mathsf{z} := \mathsf{z} \div 2\ ;\ \mathsf{q} := \mathsf{q} \times 2\ ;$

$\quad \textbf{if}\ \mathsf{z} \leq \mathsf{r}\ \textbf{then}\ \mathsf{q} := \mathsf{q} + 1\ ;\ \mathsf{r} := \mathsf{r} - \mathsf{z}\ \textbf{else}\ \textbf{skip}\big)\big)$

$\{\mathsf{x} = \mathsf{q} \times \mathsf{y} + \mathsf{r} \wedge 0 \leq \mathsf{r} < \mathsf{y}\}$ (DC,19)

An Annotated Specification for Fast Division

$\{x \geq 0 \wedge y > 0\}$

$\textbf{newvar}\ n := 0\ \textbf{in}\ \textbf{newvar}\ z := y\ \textbf{in}$

$\big(\{z = y \times 2^n \wedge n \geq 0 \wedge x \geq 0 \wedge y > 0\}$

$\textbf{while}\ z \leq x\ \textbf{do}\ (n := n + 1\ ;\ z = z \times 2)\ ;$

$q := 0\ ;\ r := x\ ;$

$\{x = q \times z + r \wedge 0 \leq r < z \wedge z = y \times 2^n \wedge n \geq 0\}$

$\textbf{while}\ n \neq 0\ \textbf{do}$

$\quad \big(n := n - 1\ ;\ z := z \div 2\ ;\ q := q \times 2\ ;$

$\quad \{x = q \times z + r \wedge 0 \leq r < z \times 2 \wedge z = y \times 2^n \wedge n \geq 0\}$

$\quad \textbf{if}\ z \leq r\ \textbf{then}\ q := q + 1\ ;\ r := r - z\ \textbf{else}\ \textbf{skip}\big)\big)$

$\{x = q \times y + r \wedge 0 \leq r < y\}$

An Example from Separation of Logic: Relative Pointers

$$\{\mathbf{emp}\}$$
$$\mathsf{x} := \mathbf{cons}(\mathsf{a}, \mathsf{a})\ ;$$
$$\{\mathsf{x} \mapsto \mathsf{a}, \mathsf{a}\}$$
$$\mathsf{y} := \mathbf{cons}(\mathsf{b}, \mathsf{b})\ ;$$
$$\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{a})\ *\ (\mathsf{y} \mapsto \mathsf{b}, \mathsf{b})\}$$
$$\{(\mathsf{x} \mapsto \mathsf{a}, -)\ *\ (\mathsf{y} \mapsto \mathsf{b}, -)\}$$
$$[\mathsf{x} + 1] := \mathsf{y} - \mathsf{x}\ ;$$
$$\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{y} - \mathsf{x})\ *\ (\mathsf{y} \mapsto \mathsf{b}, -)\}$$
$$[\mathsf{y} + 1] := \mathsf{x} - \mathsf{y}\ ;$$
$$\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{y} - \mathsf{x})\ *\ (\mathsf{y} \mapsto \mathsf{b}, \mathsf{x} - \mathsf{y})\}$$
$$\{\exists \mathsf{o}.\ (\mathsf{x} \mapsto \mathsf{a}, \mathsf{o})\ *\ (\mathsf{x} + \mathsf{o} \mapsto \mathsf{b},\ - \mathsf{o})\}.$$

Another Example (O'Hearn): Concurrent Buffering

$$\{\mathbf{emp}\}$$

$$\{\mathbf{emp} * \mathbf{emp}\}$$

$$\begin{array}{lcl}
\{\mathbf{emp}\} & & \{\mathbf{emp}\} \\
\mathsf{x} := \mathbf{cons}(\ldots,\ldots)\ ; & & \mathsf{get}(\mathsf{y})\ ; \\
\{\mathsf{x} \mapsto -,-\} & \| & \{\mathsf{y} \mapsto -,-\} \\
\mathsf{put}(\mathsf{x})\ ; & & \text{“Use } \mathsf{y}\text{”}\ ; \\
\{\mathbf{emp}\} & & \{\mathsf{y} \mapsto -,-\} \\
 & & \mathbf{dispose}\,\mathsf{y}\ ; \\
 & & \{\mathbf{emp}\}
\end{array}$$

$$\{\mathbf{emp} * \mathbf{emp}\}$$

$$\{\mathbf{emp}\}$$

Annotation Descriptions

We will write the *annotation description*

$$\mathcal{A} \gg \{p\}\ c\ \{q\}$$

to indicate that $\mathcal{A}$ is an annotated specification proving the specification $\{p\}\ c\ \{q\}$.

(The letter $\mathcal{A}$, with various decorations, will be a metavariable ranging over annotated specifications and their subphrases.)

We will define the valid annotation descriptions by means of inference rules.

A Surprise

Sometimes an annotated specification may contain *fewer* assertions than its unannotated version. For example, we will regard

$$\mathsf{y} := 2 \times \mathsf{y} \, \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}$$

as an annotated version of

$$\{2 \times \mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \ \mathsf{y} := 2 \times \mathsf{y} \ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\},$$

and

$$\mathsf{k} := \mathsf{k} + 1 \, ; \mathsf{y} := 2 \times \mathsf{y} \, \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}$$

as an annotated version of

$$\{2{\times}\mathsf{y} = 2^{\mathsf{k}+1} {\wedge} \mathsf{k}{+}1 \leq \mathsf{n}\} \ \mathsf{k}{:=}\mathsf{k}{+}1;\mathsf{y}{:=}2{\times}\mathsf{y} \ \{\mathsf{y} = 2^{\mathsf{k}} {\wedge} \mathsf{k} \leq \mathsf{n}\}.$$

The main reason we can allow such incomplete specifications is that, in such cases, for the command c and postcondition q, one can calculate a *weakest (liberal) precondition* p_w, which is an assertion such that $\{p\} \ c \ \{q\}$ holds just when $p \Rightarrow p_w$. In many such cases, we will take p_w as an implicit precondition of the annotated specification.

Assignment (AS)

$$\frac{}{v := e\,\{q\} \gg \{q/v \to e\}\; v := e\; \{q\}.}$$

Instances

$$\left.\begin{array}{l} \mathsf{y} := 2 \times \mathsf{y} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \end{array}\right\} \gg \left\{\begin{array}{l} \{2 \times \mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \\ \mathsf{y} := 2 \times \mathsf{y} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \end{array}\right.$$

and

$$\left.\begin{array}{l} \mathsf{k} := \mathsf{k} + 1 \\ \{2 \times \mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \end{array}\right\} \gg \left\{\begin{array}{l} \{2 \times \mathsf{y} = 2^{\mathsf{k}+1} \wedge \mathsf{k} + 1 \leq \mathsf{n}\} \\ \mathsf{k} := \mathsf{k} + 1 \\ \{2 \times \mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}. \end{array}\right.$$

Sequential Composition (SQ)

$$\frac{\mathcal{A}_1\ \{q\} \gg \{p\}\ c_1\ \{q\} \qquad \mathcal{A}_2 \gg \{q\}\ c_2\ \{r\}}{\mathcal{A}_1\ ;\ \mathcal{A}_2 \gg \{p\}\ c_1\ ;\ c_2\ \{r\}.}$$

For instance,

$$\left.\begin{array}{l} \mathsf{k} := \mathsf{k}+1 \\ \{2\times\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \le \mathsf{n}\} \end{array}\right\} \gg \left\{\begin{array}{l} \{2\times\mathsf{y} = 2^{\mathsf{k}+1} \wedge \mathsf{k}+1 \le \mathsf{n}\} \\ \mathsf{k} := \mathsf{k}+1 \\ \{2\times\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \le \mathsf{n}\} \end{array}\right.$$

$$\left.\begin{array}{l} \mathsf{y} := 2\times\mathsf{y} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \le \mathsf{n}\} \end{array}\right\} \gg \left\{\begin{array}{l} \{2\times\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \le \mathsf{n}\} \\ \mathsf{y} := 2\times\mathsf{y} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \le \mathsf{n}\} \end{array}\right.$$

$$\left.\begin{array}{l} \mathsf{k} := \mathsf{k}+1\ ;\ \mathsf{y} := 2\times\mathsf{y} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \le \mathsf{n}\} \end{array}\right\} \gg \left\{\begin{array}{l} \{2\times\mathsf{y} = 2^{\mathsf{k}+1} \wedge \mathsf{k}+1 \le \mathsf{n}\} \\ \mathsf{k} := \mathsf{k}+1\ ;\ \mathsf{y} := 2\times\mathsf{y} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \le \mathsf{n}\}. \end{array}\right.$$

Strengthening Precedent (SP)

$$\frac{p \Rightarrow q \qquad \mathcal{A} \gg \{q\}\ c\ \{r\}}{\{p\}\ \mathcal{A} \gg \{p\}\ c\ \{r\}.}$$

For instance,

$$(\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}) \Rightarrow (2 \times \mathsf{y} = 2^{\mathsf{k}+1} \wedge \mathsf{k}+1 \leq \mathsf{n})$$

$$\dfrac{\left.\begin{array}{l} \mathsf{k} := \mathsf{k}+1\ ;\ \mathsf{y} := 2 \times \mathsf{y} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \end{array}\right\} \gg \left\{\begin{array}{l} \{2 \times \mathsf{y} = 2^{\mathsf{k}+1} \wedge \mathsf{k}+1 \leq \mathsf{n}\} \\ \mathsf{k} := \mathsf{k}+1\ ;\ \mathsf{y} := 2 \times \mathsf{y} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \end{array}\right.}{\left.\begin{array}{l} \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}\} \\ \mathsf{k} := \mathsf{k}+1\ ;\ \mathsf{y} := 2 \times \mathsf{y} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \end{array}\right\} \gg \left\{\begin{array}{l} \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}\} \\ \mathsf{k} := \mathsf{k}+1\ ;\ \mathsf{y} := 2 \times \mathsf{y} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}. \end{array}\right.}$$

Why Do We Ever Need Intermediate Assertions?

1. $\mathbf{while}$ commands and calls of recursive procedures do not always have weakest preconditions that can be expressed in our assertion language.
2. Certain structural inference rules, such as the existential quantifier rule (or the frame rule), do not fit well into the framework of weakest assertions.
3. Intermediate assertions are often needed to simplify verification conditions.

Partial Correctness of $\mathbf{while}$ (WH)

$$\frac{\{i \wedge b\}\ \mathcal{A}\ \{i\} \gg \{i \wedge b\}\ c\ \{i\}}{\{i\}\ \mathbf{while}\ b\ \mathbf{do}\ \mathcal{A} \gg \{i\}\ \mathbf{while}\ b\ \mathbf{do}\ c\ \{i \wedge \neg b\}.}$$

For instance,

$$\frac{\left.\begin{array}{l}\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}\}\\ \mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}\\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}\end{array}\right\} \gg \left\{\begin{array}{l}\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}\}\\ \mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}\\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}\end{array}\right.}{\left.\begin{array}{l}\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}\\ \mathbf{while}\ \mathsf{k} \neq \mathsf{n}\ \mathbf{do}\\ \quad(\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y})\end{array}\right\} \gg \left\{\begin{array}{l}\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}\\ \mathbf{while}\ \mathsf{k} \neq \mathsf{n}\ \mathbf{do}\\ \quad(\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y})\\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n} \wedge \neg\,\mathsf{k} \neq \mathsf{n}\}.\end{array}\right.}$$

Weakening Consequent (WC)

$$\frac{\mathcal{A} \gg \{p\}\ c\ \{q\} \qquad q \Rightarrow r}{\mathcal{A}\,\{r\} \gg \{p\}\ c\ \{r\}.}$$

For instance,

$$\left.\begin{array}{l} \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \\ \textbf{while}\ \mathsf{k} \neq \mathsf{n}\ \textbf{do} \\ \quad (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}) \end{array}\right\} \gg \left\{\begin{array}{l} \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \\ \textbf{while}\ \mathsf{k} \neq \mathsf{n}\ \textbf{do} \\ \quad (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}) \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}\} \end{array}\right.$$

$$\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n} \ \Rightarrow\ \mathsf{y} = 2^{\mathsf{n}}$$

$$\left.\begin{array}{l} \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \\ \textbf{while}\ \mathsf{k} \neq \mathsf{n}\ \textbf{do} \\ \quad (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}) \\ \{\mathsf{y} = 2^{\mathsf{n}}\} \end{array}\right\} \gg \left\{\begin{array}{l} \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\} \\ \textbf{while}\ \mathsf{k} \neq \mathsf{n}\ \textbf{do} \\ \quad (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}) \\ \{\mathsf{y} = 2^{\mathsf{n}}\}. \end{array}\right.$$

Alternative Axiom Schema (Assignment)

- Backward Reasoning (Hoare)

$$\overline{\mathcal{A}\,\{q\} \gg \{g(q)\}\; c\; \{q\}}$$

e.g. $$\overline{v := e\,\{q\} \gg \{q/v \to e\}\; v := e\; \{q\}}.$$

- Forward Reasoning (Floyd)

$$\overline{\{p\}\mathcal{A} \gg \{p\}\; c\; \{g(p)\}}$$

e.g. $$\overline{\{p\}v := e \gg \{p\}\; v := e\; \{\exists v'.\, v = e' \wedge p'\}},$$

where $v' \notin \{v\} \cup \mathsf{FV}(e) \cup \mathsf{FV}(p)$, e' is $e/v \to v'$, and p' is $p/v \to v'$. The quantifier can be omitted when v does not occur in e or p.

Completeness

We say that an annotated specification is *right-complete* if it ends with a postcondition, *left-complete* if it begins with a precondition, and *complete* if it is both right- and left-complete. Then the patterns

$$\left.\begin{array}{r} \mathcal{A} \\ \mathcal{A}\{q\} \\ \{p\}\mathcal{A} \\ \{p\}\mathcal{A}\{q\} \end{array}\right\} \begin{array}{c} \text{will} \\ \text{match} \\ \text{any} \end{array} \left\{\begin{array}{l} \text{annotated specification.} \\ \text{right-complete annotated specification.} \\ \text{left-complete annotated specification.} \\ \text{complete annotated specification.} \end{array}\right.$$

Inclusions of Patterns

$$\begin{tikzcd}[column sep=small]
 & \mathcal{A}\ \text{(Outward)} \arrow[dl, dash, "\subseteq"'] \arrow[dr, dash, "\supseteq"] & \\
\mathcal{A}\{q\}\ \text{(Backward)} \arrow[dr, dash, "\supseteq"'] & & \{p\}\mathcal{A}\ \text{(Forward)} \arrow[dl, dash, "\subseteq"] \\
 & \{p\}\mathcal{A}\{q\}\ \text{(Inward)} &
\end{tikzcd}$$

Forcing Completeness

We can force any specification to be left-complete by strengthening its precedent with the vacuous verification condition $p \Rightarrow p$:

$$\frac{p \Rightarrow p \qquad \mathcal{A} \gg \{p\}\ c\ \{r\}}{\{p\}\,\mathcal{A} \gg \{p\}\ c\ \{r\}.}$$

For example,

$$\frac{\begin{array}{l} 2 \times \mathsf{y} = 2^{\mathsf{k}} \Rightarrow 2 \times \mathsf{y} = 2^{\mathsf{k}} \\ \qquad \mathsf{y} := 2 \times \mathsf{y}\ \{\mathsf{y} = 2^{\mathsf{k}}\} \gg \{2 \times \mathsf{y} = 2^{\mathsf{k}}\}\ \mathsf{y} := 2 \times \mathsf{y}\ \{\mathsf{y} = 2^{\mathsf{k}}\} \end{array}}{\begin{array}{l} \{2 \times \mathsf{y} = 2^{\mathsf{k}}\}\ \mathsf{y} := 2 \times \mathsf{y}\ \{\mathsf{y} = 2^{\mathsf{k}}\} \gg \\ \qquad \{2 \times \mathsf{y} = 2^{\mathsf{k}}\}\ \mathsf{y} := 2 \times \mathsf{y}\ \{\mathsf{y} = 2^{\mathsf{k}}\}. \end{array}}$$

Similarly, we can force any specification to be right-complete by weakening its consequent with $q \Rightarrow q$.

Alternative Rules with Premisses (Conditionals)

- Backward-Preserving Reasoning (for weakest preconditions)

$$\frac{\begin{array}{c}\mathcal{A}_1\{q\} \gg \{p_1\}\ c_1\ \{q\} \\ \mathcal{A}_2\{q\} \gg \{p_2\}\ c_2\ \{q\}\end{array}}{\begin{array}{l}\mathbf{if}\ b\ \mathbf{then}\ \mathcal{A}_1\ \mathbf{else}\ (\mathcal{A}_2)\{q\} \gg \\ \quad \{(b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2)\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{q\}\end{array}}$$

- Forward-Preserving Reasoning

$$\frac{\begin{array}{c}\{p \wedge b\}\mathcal{A}_1 \gg \{p \wedge b\}\ c_1\ \{q_1\} \\ \{p \wedge \neg\, b\}\mathcal{A}_2 \gg \{p \wedge \neg\, b\}\ c_2\ \{q_2\}\end{array}}{\begin{array}{l}\{p\}\mathbf{if}\ b\ \mathbf{then}\ \mathcal{A}_1\ \mathbf{else}\ (\mathcal{A}_2) \gg \\ \qquad \{p\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{q_1 \vee q_2\}\end{array}}$$

- Inward-Preserving Reasoning (my preference)

$$\frac{\begin{array}{c}\{p \wedge b\}\ \mathcal{A}_1\ \{q\} \gg \{p \wedge b\}\ c_1\ \{q\} \\ \{p \wedge \neg\, b\}\ \mathcal{A}_2\ \{q\} \gg \{p \wedge \neg\, b\}\ c_2\ \{q\}\end{array}}{\begin{array}{l}\{p\}\ \mathbf{if}\ b\ \mathbf{then}\ \mathcal{A}_1\ \mathbf{else}\ (\mathcal{A}_2)\ \{q\} \gg \\ \qquad \{p\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{q\}\end{array}}$$

- Outward-Preserving Reasoning

$$\frac{\begin{array}{c}\mathcal{A}_1 \gg \{p_1\}\ c_1\ \{q_1\} \\ \mathcal{A}_2 \gg \{p_2\}\ c_2\ \{q_2\}\end{array}}{\begin{array}{l}\mathbf{if}\ b\ \mathbf{then}\ \mathcal{A}_1\ \mathbf{else}\ (\mathcal{A}_2) \gg \\ \quad \{(b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2)\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{q_1 \vee q_2\}\end{array}}$$

Variable Declaration (DC) (Inward-Preserving)

$$\frac{\{p\}\ \mathcal{A}\ \{q\} \gg \{p\}\ c\ \{q\}}{\{p\}\ \textbf{newvar}\ v\ \textbf{in}\ \mathcal{A}\ \{q\} \gg \{p\}\ \textbf{newvar}\ v\ \textbf{in}\ c\ \{q\},}$$

when v does not occur free in p or q.

Skip (SK) (Backward)

$$\frac{}{\textbf{skip}\,\{q\} \gg \{q\}\ \textbf{skip}\ \{q\}.}$$

More Structural Rules

In the following structural rules:

- In the unary rules a right brace is used to indicate the extent of the single operand.
- In the binary rules the two operands are placed symmetrically around the indicator DISJ, CONJ, or $\|$.

(We assume that the annotated specifications in the premisses will often be sequences of several lines.)

Vacuity (VAC) (Backward)

$$\frac{}{c\Big\}\text{VAC}\;\{q\} \gg \{\mathbf{false}\}\; c\; \{q\}.}$$

Here c contains no annotations, since no reasoning about its subcommands is used. For example, using (VAC), (SP), (WH), and (WC):

$$\begin{array}{l}
\{\mathsf{s} = 0 \land \mathsf{a} - 1 \geq \mathsf{b} \land \mathsf{k} \geq \mathsf{b}\} \\
\mathbf{while}\; \mathsf{k} < \mathsf{b}\; \mathbf{do} \\
\quad \left.\begin{array}{l} (\mathsf{k} := \mathsf{k} + 1\;; \\ \mathsf{s} := \mathsf{s} + \mathsf{k}) \end{array}\right\}\text{VAC} \\
\{\mathsf{s} = 0 \land \mathsf{a} - 1 \geq \mathsf{b}\}.
\end{array}$$

Disjunction (DISJ) (Backward-Preserving)

$$\frac{\mathcal{A}_1\,\{q\} \gg \{p_1\}\ c\ \{q\} \qquad \mathcal{A}_2\,\{q\} \gg \{p_2\}\ c\ \{q\}}{(\mathcal{A}_1\ DISJ\ \mathcal{A}_2)\,\{q\} \gg \{p_1 \vee p_2\}\ c\ \{q\}.}$$

For example,

$$\{\mathbf{true}\}$$

$$\begin{array}{lcl}
 & & \{\mathsf{a} - 1 \geq \mathsf{b}\} \\
\{\mathsf{a} - 1 \leq \mathsf{b}\} & & \mathsf{s} := 0\ ;\ \mathsf{k} := \mathsf{a} - 1\ ; \\
\mathsf{s} := 0\ ;\ \mathsf{k} := \mathsf{a} - 1\ ; & & \{\mathsf{s} = 0 \wedge \mathsf{a} - 1 \geq \mathsf{b} \wedge \mathsf{k} \geq \mathsf{b}\} \\
\{\mathsf{s} = \sum_{\mathsf{i}=\mathsf{a}}^{\mathsf{k}} \mathsf{i} \wedge \mathsf{k} \leq \mathsf{b}\} & \text{DISJ} & \mathbf{while}\ \mathsf{k} < \mathsf{b}\ \mathbf{do} \\
\mathbf{while}\ \mathsf{k} < \mathsf{b}\ \mathbf{do} & & \ (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{s} := \mathsf{s} + \mathsf{k})\big\}\text{VAC} \\
\ (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{s} := \mathsf{s} + \mathsf{k}) & & \{\mathsf{s} = 0 \wedge \mathsf{a} - 1 \geq \mathsf{b}\}.
\end{array}$$

$$\{\mathsf{s} = \textstyle\sum_{\mathsf{i}=\mathsf{a}}^{\mathsf{b}} \mathsf{i}\}.$$

Conjunction (CONJ)

- Forward-Preserving

$$\frac{\{p\}\mathcal{A}_1 \gg \{p\}\ c\ \{q_1\} \qquad \{p\}\mathcal{A}_2 \gg \{p\}\ c\ \{q_2\}}{\{p\}(\mathcal{A}_1 \ \textsf{CONJ}\ \mathcal{A}_2) \gg \{p\}\ c\ \{q_1 \wedge q_2\},}$$

- Outward-Preserving (Better?):

$$\frac{\mathcal{A}_1 \gg \{p_1\}\ c\ \{q_1\} \qquad \mathcal{A}_2 \gg \{p_2\}\ c\ \{q_2\}}{(\mathcal{A}_1 \ \textsf{CONJ}\ \mathcal{A}_2) \gg \{p_1 \wedge p_2\}\ c\ \{q_1 \wedge q_2\}.}$$

Concurrency (CONC) (Outward-Preserving)

- In Hoare Logic:

$$\frac{\mathcal{A}_1 \gg \{p_1\}\ c_1\ \{q_1\} \qquad \mathcal{A}_2 \gg \{p_2\}\ c_2\ \{q_2\}}{(\mathcal{A}_1\ \|\ \mathcal{A}_2) \gg \{p_1 \wedge p_2\}\ c_1 \,\|\, c_2\ \{q_1 \wedge q_2\}},$$

  where no variable occurring free in p_1, c_1, or q_1 is modified by c_2, and vice-versa.

The above rule is unsound for separation logic; instead we have:

- In Separation Logic (O'Hearn)

$$\frac{\mathcal{A}_1 \gg \{p_1\}\ c_1\ \{q_1\} \qquad \mathcal{A}_2 \gg \{p_2\}\ c_2\ \{q_2\}}{(\mathcal{A}_1\ \|\ \mathcal{A}_2) \gg \{p_1\ *\ p_2\}\ c_1 \,\|\, c_2\ \{q_1\ *\ q_2\}},$$

  where no variable occurring free in p_1, c_1, or q_1 is modified by c_2, and vice-versa.

Existential Quantification (EQ) (Outward-Preserving)

$$\frac{\mathcal{A} \gg \{p\}\ c\ \{q\}}{\mathcal{A}\big\}\, \exists v\ \gg \{\exists v.\ p\}\ c\ \{\exists v.\ q\}},$$

where v is not free in c.

Universal Quantification (UQ) (Outward-Preserving)

$$\frac{\mathcal{A} \gg \{p\}\ c\ \{q\}}{\mathcal{A}\big\}\, \forall v\ \gg \{\forall v.\ p\}\ c\ \{\forall v.\ q\}},$$

where v is not free in c.

Constancy (CONST) (Outward-Preserving)

$$\frac{\mathcal{A} \gg \{p\}\ c\ \{q\}}{\mathcal{A}\big\} \wedge r\ \gg \{p \wedge r\}\ c\ \{q \wedge r\},}$$

where no variable occurring free in r is modified by c.

(In separation logic, r must be pure.)

An Example

$$\begin{array}{l}
\{\mathsf{f} = \mathsf{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathsf{fib}(\mathsf{k}-1) \wedge \mathsf{k} < \mathsf{n}\} \\
\left.\begin{array}{l}
\{\mathsf{f} = \mathsf{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathsf{fib}(\mathsf{k}-1)\} \\
\mathsf{t} := \mathsf{g}\ ; \\
\mathsf{g} := \mathsf{f}\ ; \\
\mathsf{f} := \mathsf{g} + \mathsf{t}\ ; \\
\{\mathsf{f} = \mathsf{fib}(\mathsf{k}+1) \wedge \mathsf{g} = \mathsf{fib}(\mathsf{k})\}
\end{array}\right\} \wedge \mathsf{k} < \mathsf{n} \\
\mathsf{k} := \mathsf{k} + 1 \\
\{\mathsf{f} = \mathsf{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathsf{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\}
\end{array}$$

Frame (FR) (O'Hearn) (Outward-Preserving)

$$\frac{\mathcal{A} \gg \{p\}\ c\ \{q\}}{\left.\mathcal{A}\right\} * r \gg \{p * r\}\ c\ \{q * r\},}$$

where no variable occurring free in r is modified by c.

An Example

$$\begin{array}{l}\{\exists \mathsf{j}.\ \mathsf{x} \mapsto -, \mathsf{j} * \mathsf{list}\ \alpha\ \mathsf{j}\} \\ \quad \left.\left.\begin{array}{l}\{\mathsf{x} \mapsto -\} \\ [\mathsf{x}] := \mathsf{a} \\ \{\mathsf{x} \mapsto \mathsf{a}\}\end{array}\right\} * \mathsf{x} + 1 \mapsto \mathsf{j} * \mathsf{list}\ \alpha\ \mathsf{j}\right\} \exists \mathsf{j} \\ \{\exists \mathsf{j}.\ \mathsf{x} \mapsto \mathsf{a}, \mathsf{j} * \mathsf{list}\ \alpha\ \mathsf{j}\}\end{array}$$

Substitution (SUB) (Outward-Preserving)

$$\frac{\mathcal{A} \gg \{p\}\ c\ \{q\}}{\mathcal{A}\Big\}/\delta \gg \{p/\delta\}\ (c/\delta)\ \{q/\delta\},}$$

where δ is the substitution $v_1 \rightarrow e_1, \ldots, v_n \rightarrow e_n$; $v_1, \ldots, v_n$ are the variables occurring free in p, c, or q, and, if v_i is modified by c, then e_i is a variable that does not occur free in any other e_j.

In the conclusion of this rule, $\mathcal{A}\Big\}/\delta$ denotes an annotated specification in which "/" and the substitution denoted by δ occur literally, i.e., the substitution is not carried out on $\mathcal{A}$.

An Example

In

$$\{\mathsf{x} = \mathsf{y}\}\ \mathsf{x} := \mathsf{x} + \mathsf{y}\ \{\mathsf{x} = 2 \times \mathsf{y}\},$$

one can substitute $\mathsf{x} \to \mathsf{z},\ \mathsf{y} \to 2 \times \mathsf{w} - 1$ to infer

$$\{\mathsf{z} = 2 \times \mathsf{w} - 1\}\ \mathsf{z} := \mathsf{z} + 2 \times \mathsf{w} - 1\ \{\mathsf{z} = 2 \times (2 \times \mathsf{w} - 1)\}.$$

But one cannot substitute $\mathsf{x} \to \mathsf{z},\ \mathsf{y} \to 2 \times \mathsf{z} - 1$ to infer the invalid

$$\{\mathsf{z} = 2 \times \mathsf{z} - 1\}\ \mathsf{z} := \mathsf{z} + 2 \times \mathsf{z} - 1\ \{\mathsf{z} = 2 \times (2 \times \mathsf{z} - 1)\}.$$

Simple Procedures

By "simple" procedures, we mean that the following restrictions are imposed:

- Parameters are variables and expressions, not commands or procedure names.
- There are no "global" variables: All free variables of the procedure body must be formal parameters of the procedure.
- Procedures are proper, i.e., their calls are commands.
- Calls are restricted to prevent aliasing.

An additional peculiarity, which substantially simplifies reasoning about simple procedures, is that we syntactically distinguish parameters that may be modified from those that may not be.

Procedure Definitions

A *simple nonrecursive (or recursive) procedure definition* is a command of the form

$$\mathbf{let}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n) = c\ \mathbf{in}\ c'$$
$$\mathbf{letrec}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n) = c\ \mathbf{in}\ c',$$

where

- h is a binding occurrence of a procedure name, whose scope is c' (or c and c' in the recursive case).
- c and c' are commands.
- $v_1, \ldots, v_m; v'_1, \ldots, v'_n$ is a list of distinct variables, called *formal parameters*, that includes all of the free variables of c. The formal parameters are binding occurrences whose scope is c.
- $v_1, \ldots, v_m$ includes all of the variables modified by c.

Procedure Calls

A *procedure call* is a command of the form

$$h(w_1, \ldots, w_m; e'_1, \ldots, e'_n),$$

where

- h is a procedure name.
- $w_1, \ldots, w_m$ and $e'_1, \ldots, e'_n$ are called *actual parameters*.
- $w_1, \ldots, w_m$ are distinct variables.
- $e'_1, \ldots, e'_n$ are expressions that do not contain occurrences of the variables $w_1, \ldots, w_m$.
- The free variables of the procedure call are

  $$\begin{aligned}&\mathsf{FV}(h(w_1, \ldots, w_m; e'_1, \ldots, e'_n)) = \\ &\qquad\qquad \{w_1, \ldots, w_m\} \cup \mathsf{FV}(e'_1) \cup \cdots \cup \mathsf{FV}(e'_n)\end{aligned}$$

  and the variables modified by the call are $w_1, \ldots, w_m$.

Hypothetical Specifications

The truth of a specification $\{p\}\ c\ \{q\}$ will depend upon an *environment*, which maps the procedure names occurring free in c into their meanings.

We define a *hypothetical specification* to have the form

$$\Gamma \vdash \{p\}\ c\ \{q\},$$

where the *context* Γ is a sequence of specifications of the form

$$\{p_0\}\ c_0\ \{q_0\}, \ldots, \{p_{n-1}\}\ c_{n-1}\ \{q_{n-1}\}.$$

We say that such a hypothetical specification is true iff $\{p\}\ c\ \{q\}$ holds for every environment in which all of the specifications in Γ hold.

Generalizing Old Inference Rules

For example,

- Strengthening Precedent (SP)

$$\frac{p \Rightarrow q \qquad \Gamma \vdash \{q\}\ c\ \{r\}}{\Gamma \vdash \{p\}\ c\ \{r\}.}$$

- Substitution (SUB)

$$\frac{\Gamma \vdash \{p\}\ c\ \{q\}}{\Gamma \vdash \{p/\delta\}\ (c/\delta)\ \{q/\delta\},}$$

  where δ is the substitution $v_1 \to e_1, \ldots, v_n \to e_n$; $v_1, \ldots, v_n$ are the variables occurring free in p, c, or q, and, if v_i is modified by c, then e_i is a variable that does not occur free in any other e_j.

Note that substitutions do not affect procedure names.

Rules for Procedures

- Hypothesis (HYPO)

$$\frac{}{\Gamma, \{p\}\ c\ \{q\}, \Gamma' \vdash \{p\}\ c\ \{q\}.}$$

- Simple Procedures (SPROC)

$$\frac{\begin{array}{l}\Gamma \vdash \{p\}\ c\ \{q\} \\ \qquad \Gamma, \{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\ \{q\} \vdash \{p'\}\ c'\ \{q'\}\end{array}}{\Gamma \vdash \{p'\}\ \mathbf{let}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n) = c\ \mathbf{in}\ c'\ \{q'\},}$$

where h does not occur free in any triple of Γ.

- Simple Recursive Procedures (SRPROC) (partial correctness only)

$$\frac{\begin{array}{l}\Gamma, \{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\ \{q\} \vdash \{p\}\ c\ \{q\} \\ \qquad \Gamma, \{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\ \{q\} \vdash \{p'\}\ c'\ \{q'\}\end{array}}{\Gamma \vdash \{p'\}\ \mathbf{letrec}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n) = c\ \mathbf{in}\ c'\ \{q'\},}$$

where h does not occur free in any triple of Γ.

Some Limitations

To keep our exposition straightforward, we have ignored:

- Simultaneous recursion,
- Multiple hypotheses for the same procedure.

Two Derived Rules

From (HYPO):

- Call (CALL)

$$\frac{}{\begin{array}{l}\Gamma, \{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\ \{q\}, \Gamma' \vdash \\ \qquad\qquad \{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\ \{q\}.\end{array}}$$

and from (CALL) and (SUB):

- General Call (GCALL)

$$\frac{}{\begin{array}{l}\Gamma, \{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\ \{q\}, \Gamma' \vdash \\ \qquad\qquad \{p/\delta\}\ h(w_1, \ldots, w_m; e'_1, \ldots, e'_n)\ \{q/\delta\},\end{array}}$$

where δ is a substitution

$$\begin{aligned}\delta = v_1 \to w_1, \ldots, v_m \to w_m,\\ v'_1 \to e'_1, \ldots, v'_n \to e'_n,\\ v''_1 \to e''_1, \ldots, v''_k \to e''_k,\end{aligned}$$

which acts on all the free variables in

$$\{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\ \{q\},$$

and none of the variables $w_1, \ldots, w_m$ occur free in the expressions $e'_1, \ldots, e'_n$ or $e''_1, \ldots, e''_k$.

Annotated Specifications: Ghosts

In (GCALL):

$$\frac{}{\begin{array}{l}\Gamma, \{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\ \{q\}, \Gamma' \vdash \\ \qquad\qquad \{p/\delta\}\ h(w_1, \ldots, w_m; e'_1, \ldots, e'_n)\ \{q/\delta\},\end{array}}$$

where δ is a substitution

$$\begin{aligned}\delta = v_1 \to w_1, \ldots, v_m \to w_m,& \\ v'_1 \to e'_1, \ldots, v'_n \to e'_n,& \\ v''_1 \to e''_1, \ldots, v''_k \to e''_k,&\end{aligned}$$

which acts on . . . .

there may be ghost variables $v''_1, \ldots, v''_k$ that appear in δ but are not formal parameters.

We will treat $v''_1, \ldots, v''_k$ as formal ghost parameters, and $e''_1, \ldots, e''_k$ as actual ghost parameters.

For example,

$$\left.\begin{array}{l}\{\mathsf{n} \geq 0 \wedge \mathsf{r} = \mathsf{r}_0\} \\ \mathsf{multfact}(\mathsf{r};\mathsf{n}) \\ \{\mathsf{r} = \mathsf{n}! \times \mathsf{r}_0\}\end{array}\right\} \vdash \left\{\begin{array}{l}\{\mathsf{n}-1 \geq 0 \wedge \mathsf{r} = \mathsf{n} \times \mathsf{r}_0\} \\ \mathsf{multfact}(\mathsf{r};\mathsf{n}-1) \\ \{\mathsf{r} = (\mathsf{n}-1)! \times \mathsf{n} \times \mathsf{r}_0\}\end{array}\right.$$

is an instance of (GCALL) using the substitution

$$\mathsf{r} \rightarrow \mathsf{r}, \mathsf{n} \rightarrow \mathsf{n}-1, \mathsf{r}_0 \rightarrow \mathsf{n} \times \mathsf{r}_0.$$

The corresponding annotated specification will be

$$\left.\begin{array}{l}\{\mathsf{n} \geq 0 \wedge \mathsf{r} = \mathsf{r}_0\} \\ \mathsf{multfact}(\mathsf{r};\mathsf{n})\underline{\{\mathsf{r}_0\}} \\ \{\mathsf{r} = \mathsf{n}! \times \mathsf{r}_0\}\end{array}\right\} \vdash \left\{\begin{array}{l}\{\mathsf{n}-1 \geq 0 \wedge \mathsf{r} = \mathsf{n} \times \mathsf{r}_0\} \\ \mathsf{multfact}(\mathsf{r};\mathsf{n}-1)\underline{\{\mathsf{n} \times \mathsf{r}_0\}} \\ \{\mathsf{r} = (\mathsf{n}-1)! \times \mathsf{n} \times \mathsf{r}_0\}.\end{array}\right.$$

Generalizing Annotation Descriptions

An *annotated context* is a sequence of *annotated hypotheses*, which have the form

$$\{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\{v''_1, \ldots, v''_k\}\ \{q\},$$

where $v''_1, \ldots, v''_k$ is a list of formal ghost parameters (and all of the formal parameters, including the ghosts, are distinct).

We write $\widehat{\Gamma}$ to denote an annotated context, and Γ to denote the corresponding ordinary context that is obtained by erasing the lists of ghost formal parameters. Then an annotation description has the form:

$$\widehat{\Gamma} \vdash \mathcal{A} \gg \{p\}\ c\ \{q\},$$

meaning that $\widehat{\Gamma} \vdash \mathcal{A}$ is an annotated hypothetical specification proving the hypothetical specification $\Gamma \vdash \{p\}\ c\ \{q\}$.

Rules for Procedural Annotated Specifications

- General Call (GCALL) (Outward)

$$\frac{}{\begin{array}{r}\widehat{\Gamma}, \{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\{v''_1, \ldots, v''_k\}\ \{q\}, \widehat{\Gamma}' \vdash \quad\quad\quad\quad\quad\quad \\ h(w_1, \ldots, w_m; e'_1, \ldots, e'_n)\{e''_1, \ldots, e''_k\} \gg \\ \{p/\delta\}\ h(w_1, \ldots, w_m; e'_1, \ldots, e'_n)\ \{q/\delta\},\end{array}}$$

where δ is the substitution

$$\begin{aligned}\delta = v_1 \to w_1, \ldots, v_m \to w_m,& \\ v'_1 \to e'_1, \ldots, v'_n \to e'_n,& \\ v''_1 \to e''_1, \ldots, v''_k \to e''_k,&\end{aligned}$$

which acts on all the free variables in

$$\{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\ \{q\},$$

and $w_1, \ldots, w_m$ are distinct variables that do not occur free in the expressions $e'_1, \ldots, e'_n$ or $e''_1, \ldots, e''_k$.

- Simple Procedures (SPROC)

$$\frac{\begin{array}{l}\widehat{\Gamma} \vdash \{p\}\ \mathcal{A}\ \{q\} \gg \{p\}\ c\ \{q\} \\ \widehat{\Gamma}, \{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\{v''_1, \ldots, v''_k\}\ \{q\} \vdash \\ \hfill \mathcal{A}' \gg \{p'\}\ c'\ \{q'\}\end{array}}{\begin{array}{l}\widehat{\Gamma} \vdash \mathbf{let}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\{v''_1, \ldots, v''_k\} = \\ \hfill \{p\}\ \mathcal{A}\ \{q\}\ \mathbf{in}\ \mathcal{A}' \\ \quad \gg \{p'\}\ \mathbf{let}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n) = c\ \mathbf{in}\ c'\ \{q'\},\end{array}}$$

where h does not occur free in any triple of $\widehat{\Gamma}$.

- Simple Recursive Procedures (SRPROC)

$$\frac{\begin{array}{r}\hat{\Gamma}, \{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\{v''_1, \ldots, v''_k\}\ \{q\} \vdash \\ \{p\}\ \mathcal{A}\ \{q\} \gg \{p\}\ c\ \{q\} \\ \hat{\Gamma}, \{p\}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\{v''_1, \ldots, v''_k\}\ \{q\} \vdash \\ \mathcal{A}' \gg \{p'\}\ c'\ \{q'\}\end{array}}{\begin{array}{r}\hat{\Gamma} \vdash \mathbf{letrec}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n)\{v''_1, \ldots, v''_k\} = \\ \mathcal{A}\ \mathbf{in}\ \{p'\}\ \mathcal{A}'\ \{q'\} \\ \gg \{p'\}\ \mathbf{letrec}\ h(v_1, \ldots, v_m; v'_1, \ldots, v'_n) = c\ \mathbf{in}\ c'\ \{q'\},\end{array}}$$

where h does not occur free in any triple of $\hat{\Gamma}$.

An Example

$$
\begin{array}{l}
\{\mathsf{z} = 10\} \\
\textbf{letrec}\ \mathsf{multfact}(\mathsf{r};\mathsf{n})\{\mathsf{r}_0\} = \\
\quad \{\mathsf{n} \geq 0 \land \mathsf{r} = \mathsf{r}_0\} \\
\quad \textbf{if}\ \mathsf{n} = 0\ \textbf{then} \\
\qquad \{\mathsf{n} = 0 \land \mathsf{r} = \mathsf{r}_0\}\ \textbf{skip}\ \{\mathsf{r} = \mathsf{n}! \times \mathsf{r}_0\} \\
\quad \textbf{else} \\
\qquad \{\mathsf{n} - 1 \geq 0 \land \mathsf{n} \times \mathsf{r} = \mathsf{n} \times \mathsf{r}_0\} \\
\qquad \mathsf{r} := \mathsf{n} \times \mathsf{r}\ ; \\
\qquad \{\mathsf{n} - 1 \geq 0 \land \mathsf{r} = \mathsf{n} \times \mathsf{r}_0\} \\
\qquad\quad \left.\begin{array}{l}
\{\mathsf{n} - 1 \geq 0 \land \mathsf{r} = \mathsf{n} \times \mathsf{r}_0\} \\
\mathsf{multfact}(\mathsf{r};\mathsf{n} - 1)\{\mathsf{n} \times \mathsf{r}_0\} \\
\{\mathsf{r} = (\mathsf{n} - 1)! \times \mathsf{n} \times \mathsf{r}_0\}
\end{array}\right\} \land \mathsf{n} - 1 \geq 0 \\
\qquad \{\mathsf{n} - 1 \geq 0 \land \mathsf{r} = (\mathsf{n} - 1)! \times \mathsf{n} \times \mathsf{r}_0\} \\
\quad \{\mathsf{r} = \mathsf{n}! \times \mathsf{r}_0\} \\
\textbf{in} \\
\{5 \geq 0 \land \mathsf{z} = 10\} \\
\mathsf{multfact}(\mathsf{z}; 5)\{10\} \\
\{\mathsf{z} = 5! \times 10\}
\end{array}
$$

How the Annotations Determine a Formal Proof

The application of (SRPROC) to the $\mathbf{letrec}$ definition gives rise to the hypothesis

$$\{\mathsf{n} \geq 0 \wedge \mathsf{r} = \mathsf{r}_0\}\ \mathsf{multfact}(\mathsf{r}; \mathsf{n})\{\mathsf{r}_0\}\ \{\mathsf{r} = \mathsf{n}! \times \mathsf{r}_0\}.$$

By (GCALL), the hypothesis entails

$$\begin{array}{l} \{\mathsf{n} - 1 \geq 0 \wedge \mathsf{r} = \mathsf{n} \times \mathsf{r}_0\} \\ \mathsf{multfact}(\mathsf{r}; \mathsf{n} - 1)\{\mathsf{n} \times \mathsf{r}_0\} \\ \{\mathsf{r} = (\mathsf{n} - 1)! \times \mathsf{n} \times \mathsf{r}_0\}. \end{array}$$

Next, since n is not modified by the call $\mathsf{multfact}(\mathsf{r}; \mathsf{n} - 1)$, the rule of constancy gives

$$\begin{array}{l} \{\mathsf{n} - 1 \geq 0 \wedge \mathsf{r} = \mathsf{n} \times \mathsf{r}_0 \wedge \mathsf{n} - 1 \geq 0\} \\ \mathsf{multfact}(\mathsf{r}; \mathsf{n} - 1)\{\mathsf{n} \times \mathsf{r}_0\} \\ \{\mathsf{r} = (\mathsf{n} - 1)! \times \mathsf{n} \times \mathsf{r}_0 \wedge \mathsf{n} - 1 \geq 0\}. \end{array}$$

Then, we can strengthen the precondition to obtain

$$\begin{array}{l} \{\mathsf{n} - 1 \geq 0 \wedge \mathsf{r} = \mathsf{n} \times \mathsf{r}_0\} \\ \mathsf{multfact}(\mathsf{r}; \mathsf{n} - 1)\{\mathsf{n} \times \mathsf{r}_0\} \\ \{\mathsf{r} = (\mathsf{n} - 1)! \times \mathsf{n} \times \mathsf{r}_0 \wedge \mathsf{n} - 1 \geq 0\}. \end{array}$$

Also, by (GCALL), the hypothesis entails

$$\{5 \geq 0 \wedge \mathsf{z} = 10\}\ \mathsf{multfact}(\mathsf{z}; 5)\{10\}\ \{\mathsf{z} = 5! \times 10\}.$$

Why Annotated Specifications Work

$$
\begin{tikzcd}
\text{Proofs of Specifications} \arrow[d, "\Phi"] \arrow[dr, dash, dashed, "\sim"] & \\
\text{Proofs of Annotation Descriptions} \arrow[r, "\text{erase-annspec}"] \arrow[d, "\text{concl}"] & \text{Proofs of Specifications} \\
\text{Annotation Descriptions} \arrow[d, "\text{erase-spec}"] & \\
\text{Annotated Specifications} \arrow[d, "\Psi"] & \\
\text{Proofs of Specifications} \arrow[uuur, dash, dashed, "\simeq"'] &
\end{tikzcd}
$$

- $\simeq$ relates proofs that contain the same verification conditions.
- $\sim$ relates proofs that contain the same verification conditions, except for vacuous conditions of the form $p \Rightarrow p$.

This proof depends upon the fact that we have not permitted more than one rule for any particular command type or structural situation.

Annotated Specifications for the Heap Commands

- Mutation (MUBR) (Backward)

$$\overline{[e] := e' \, \{p\} \gg \{(e \mapsto -) * ((e \mapsto e') -\!\!* \, p)\} \; [e] := e' \; \{p\}.}$$

- Disposal (DISBR) (Backward)

$$\overline{\mathbf{dispose}\; e \, \{r\} \gg \{(e \mapsto -) * r\} \; \mathbf{dispose}\; e \; \{r\}.}$$

- Allocation (CONSBR) (Backward)

$$\overline{v := \mathbf{cons}(\overline{e}) \, \{p\} \gg \{\forall v''.\ (v'' \mapsto \overline{e}) -\!\!* \, p''\} \; v := \mathbf{cons}(\overline{e}) \; \{p\},}$$

  where v'' is distinct from v, $v'' \notin \mathsf{FV}(\overline{e}, p)$, and p'' denotes $p/v \to v''$.

- Lookup (LKBR1) (Backward)

$$\overline{v := [e] \, \{p\} \gg \{\exists v''.\ (e \mapsto v'') * ((e \mapsto v'') -\!\!* \, p'')\} \; v := [e] \; \{p\},}$$

  where $v'' \notin \mathsf{FV}(e) \cup (\mathsf{FV}(p) - \{v\})$, and p'' denotes $p/v \to v''$.

Deriving Local and Global Rules

By taking p in (MUBR) to be $e \mapsto e'$, and using the valid verification condition

$$\text{VC} = (e \mapsto -) \Rightarrow (e \mapsto -) \ * \ ((e \mapsto e') \mathrel{-\!\!*} (e \mapsto e')),$$

we may use (SP) to obtain a proof:

$$\dfrac{VC \qquad \dfrac{}{\begin{array}{l}[e] := e' \, \{e \mapsto e'\} \gg \\ \quad \{(e \mapsto -) \ * \ ((e \mapsto e') \mathrel{-\!\!*} (e \mapsto e'))\} \ [e] := e' \ \{e \mapsto e'\}\end{array}}}{\{e \mapsto -\} \ [e] := e' \ \{e \mapsto e'\} \gg \{e \mapsto -\} \ [e] := e' \ \{e \mapsto e'\}}$$

of an annotation description corresponding to the local form (MUL). In such a manner, one may derive local and global rules of the form

$$\overline{\{p\} \ c \ \{q\} \gg \{p\} \ c \ \{q\}}.$$