AN INTRODUCTION TO SEPARATION LOGIC

3. Specifications

John C. Reynolds
Carnegie Mellon University

February 3, 2011

Hoare Triples

- A *partial correctness specification*

$$\{p\}\ c\ \{q\}$$

  is *valid* iff, starting in any state in which p holds:
  - No execution of c aborts.
  - When some execution of c terminates in a final state, then q holds in the final state.

—

- A *total correctness specification*

$$[\ p\ ]\ c\ [\ q\ ]$$

  is *valid* iff, starting in any state in which p holds:
  - No execution of c aborts.
  - Every execution of c terminates.
  - When some execution of c terminates in a final state, then q holds in the final state.

—

Examples of Valid Specifications

$$\{\mathsf{x} - \mathsf{y} > 3\}\ \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}$$

$$\{\mathsf{x} + \mathsf{y} \geq 17\}\ \mathsf{x} := \mathsf{x} + 10\ \{\mathsf{x} + \mathsf{y} \geq 27\}$$

$$\{\mathbf{emp}\}\ \mathsf{x} := \mathbf{cons}(1, 2)\ \{\mathsf{x} \mapsto 1, 2\}$$

$$\{\mathsf{x} \mapsto 1, 2\}\ \mathsf{y} := [\mathsf{x}]\ \{\mathsf{x} \mapsto 1, 2 \wedge \mathsf{y} = 1\}$$

$$\{\mathsf{x} \mapsto 1, 2 \wedge \mathsf{y} = 1\}\ [\mathsf{x} + 1] := 3\ \{\mathsf{x} \mapsto 1, 3 \wedge \mathsf{y} = 1\}$$

$$\{\mathsf{x} \mapsto 1, 3 \wedge \mathsf{y} = 1\}\ \mathbf{dispose}\ \mathsf{x}\ \{\mathsf{x} + 1 \mapsto 3 \wedge \mathsf{y} = 1\}$$

$$\{\mathsf{x} \leq 10\}\ \mathbf{while}\ \mathsf{x} \neq 10\ \mathbf{do}\ \mathsf{x} := \mathsf{x} + 1\ \{\mathsf{x} = 10\}$$

$$\{\mathbf{true}\}\ \mathbf{while}\ \mathsf{x} \neq 10\ \mathbf{do}\ \mathsf{x} := \mathsf{x} + 1\ \{\mathsf{x} = 10\} \quad (*)$$

$$\{\mathsf{x} > 10\}\ \mathbf{while}\ \mathsf{x} \neq 10\ \mathbf{do}\ \mathsf{x} := \mathsf{x} + 1\ \{\mathbf{false}\} \quad (*)$$

$$\begin{array}{l}
\{\mathsf{x} \mapsto -\ *\ \mathsf{y} \mapsto -\} \\
\mathbf{if}\ \mathsf{y} = \mathsf{x} + 1\ \mathbf{then\ skip} \\
\quad \mathbf{else\ if}\ \mathsf{x} = \mathsf{y} + 1\ \mathbf{then}\ \mathsf{x} := \mathsf{y} \\
\quad \mathbf{else}\ (\mathbf{dispose}\ \mathsf{x}\ ;\ \mathbf{dispose}\ \mathsf{y}\ ;\ \mathsf{x} := \mathbf{cons}(1, 2)) \\
\{\mathsf{x} \mapsto -, -\}
\end{array}$$

(All except the examples marked $(*)$ are also valid total specifications.)

—

Inference Rules and Proofs

- An *inference rule* for Hoare logic consists of zero or more *premisses* (either specifications or assertions) and a single *conclusion* (a specification), separated by a horizontal line:

$$\frac{\mathcal{P}_1 \qquad \cdots \qquad \mathcal{P}_n}{\mathcal{C}}$$

- The premisses and conclusion are schemata, i.e., they may contain *metavariables*, each of which ranges over some set of phrases, such as expressions, commands, or assertions.
- An instance of an inference rule is obtained by replacing each metavariable by a phrase in its range. These replacements must satisfy the *side-conditions* (if any) of the rule. (Since this is replacement of metavariables rather than substitution for variables, there is never any renaming.)
- A *formal proof* in Hoare logic is a sequence of assertions and/or specifications, each of which is either a valid assertion or the conclusion of some instance of a sound inference rule whose premisses occur earlier in the sequence.

—

Soundness

An inference rule of Hoare logic is *sound* iff, for all instances,

- if the premisses of the instance are all valid,
- then the conclusion is valid.

Since we require the assertions in a formal proof to be valid and the inference rules used in the proof to be sound, it follows that the specifications in a formal proof must all be valid.

—

Hoare's Inference Rules for Specifications:

Assignment (AS)

$$\frac{}{\{q/v \to e\}\ v := e\ \{q\}}$$

Instances

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

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

—

Sequential Composition (SQ)

$$\frac{\{p\}\ c_1\ \{q\} \qquad \{q\}\ c_2\ \{r\}}{\{p\}\ c_1 ; c_2\ \{r\}}$$

An Instance

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

—

Strengthening Precedent (SP)

$$\frac{p \Rightarrow q \qquad \{q\}\ c\ \{r\}}{\{p\}\ c\ \{r\}}$$

An Instance

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

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

—

Since they are applicable to arbitrary commands, the rules (SP) and (WC) (to be introduced later) are called *structural rules*. One premiss of each of these two rules is an assertion, which is called a *verification condition* (VC). The verification conditions are used to introduce mathematical facts about data types into proofs of specifications.

We will usually omit formal proofs of verification conditions.

—

Partial Correctness of **while** (WH)

$$\frac{\{i \wedge b\}\ c\ \{i\}}{\{i\}\ \mathbf{while}\ b\ \mathbf{do}\ c\ \{i \wedge \neg\, b\}}$$

Here i is the *invariant*.

An Instance

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

—

Weakening Consequent (WC)

$$\frac{\{p\}\ c\ \{q\} \qquad q \Rightarrow r}{\{p\}\ c\ \{r\}}$$

An Instance

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

—

A Proof

1. $y = 2^k \wedge k \leq n \wedge k \neq n \Rightarrow 2 \times y = 2^{k+1} \wedge k + 1 \leq n$ (VC)
2. $\{2 \times y = 2^{k+1} \wedge k + 1 \leq n\}\ k := k + 1$
   $\{2 \times y = 2^k \wedge k \leq n\}$ (AS)
3. $\{2 \times y = 2^k \wedge k \leq n\}\ y := 2 \times y\ \{y = 2^k \wedge k \leq n\}$ (AS)
4. $\{2 \times y = 2^{k+1} \wedge k + 1 \leq n\}\ k := k + 1\ ;\ y := 2 \times y$
   $\{y = 2^k \wedge k \leq n\}$ (SQ 2,3)
5. $\{y = 2^k \wedge k \leq n \wedge k \neq n\}\ k := k + 1\ ;\ y := 2 \times y$
   $\{y = 2^k \wedge k \leq n\}$ (SP 1,4)
6. $\{y = 2^k \wedge k \leq n\}$ (WH 5)
   $\mathbf{while}\ k \neq n\ \mathbf{do}\ (k := k + 1\ ;\ y := 2 \times y)$
   $\{y = 2^k \wedge k \leq n \wedge \neg\, k \neq n\}$
7. $y = 2^k \wedge k \leq n \wedge \neg\, k \neq n \Rightarrow y = 2^n$ (VC)
8. $\{y = 2^k \wedge k \leq n\}$ (WC 6,7)
   $\mathbf{while}\ k \neq n\ \mathbf{do}\ (k := k + 1\ ;\ y := 2 \times y)$
   $\{y = 2^n\}$

—

skip (SK)

$$\frac{}{\{q\}\ \mathbf{skip}\ \{q\}}$$

An Instance

$$\{\mathsf{y} = 2^{\mathsf{k}} \wedge \neg\,\mathsf{odd}(\mathsf{k})\}\ \mathbf{skip}\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \neg\,\mathsf{odd}(\mathsf{k})\}$$

—

Conditional (CD)

$$\frac{\{p \wedge b\}\ c_1\ \{q\} \qquad \{p \wedge \neg\ b\}\ c_2\ \{q\}}{\{p\}\ \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2\ \{q\}}$$

An Instance

$$\frac{\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{odd}(\mathsf{k})\}\ \mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \neg\,\mathsf{odd}(\mathsf{k})\} \qquad \{\mathsf{y} = 2^{\mathsf{k}} \wedge \neg\,\mathsf{odd}(\mathsf{k})\}\ \textbf{skip}\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \neg\,\mathsf{odd}(\mathsf{k})\}}{\begin{array}{l}\{\mathsf{y} = 2^{\mathsf{k}}\} \\ \textbf{if}\ \mathsf{odd}(\mathsf{k})\ \textbf{then}\ \mathsf{k} := \mathsf{k} + 1\,;\ \mathsf{y} := 2 \times \mathsf{y}\ \textbf{else}\ \textbf{skip} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \neg\,\mathsf{odd}(\mathsf{k})\}\end{array}}$$

Variable Declaration (DC)

$$\frac{\{p\}\ c\ \{q\}}{\{p\}\ \mathbf{newvar}\ v\ \mathbf{in}\ c\ \{q\}}$$

when v does not occur free in p or q.

An Instance

$$\frac{\begin{array}{l}\{1 = 2^0 \wedge 0 \leq \mathsf{n}\} \\ \mathsf{k} := 0\ ;\ \mathsf{y} := 1\ ; \\ \mathbf{while}\ \mathsf{k} \neq \mathsf{n}\ \mathbf{do}\ (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}) \\ \{\mathsf{y} = 2^{\mathsf{n}}\}\end{array}}{\begin{array}{l}\{1 = 2^0 \wedge 0 \leq \mathsf{n}\} \\ \mathbf{newvar}\ \mathsf{k}\ \mathbf{in} \\ \quad \big(\mathsf{k} := 0\ ;\ \mathsf{y} := 1\ ; \\ \quad \mathbf{while}\ \mathsf{k} \neq \mathsf{n}\ \mathbf{do}\ (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y})\big) \\ \{\mathsf{y} = 2^{\mathsf{n}}\}\end{array}}$$

Here the requirement on the declared variable v formalizes the concept of *locality*, i.e., that the value of v when c begins execution has no effect on this execution, and that the value of v when c finishes execution has no effect on the rest of the program.

—

Notice that locality is context-dependent: In

$$\{\mathbf{true}\}\ \mathsf{t} := \mathsf{x} + \mathsf{y}\ ;\ \mathsf{y} := \mathsf{t} \times 2\ \{\mathsf{y} = (\mathsf{x} + \mathsf{y}) \times 2\},$$

t is local, and can be declared at the beginning of the command being specified, but in

$$\{\mathbf{true}\}\ \mathsf{t} := \mathsf{x} + \mathsf{y}\ ;\ \mathsf{y} := \mathsf{t} \times 2\ \{\mathsf{y} = (\mathsf{x} + \mathsf{y}) \times 2 \wedge \mathsf{t} = (\mathsf{x} + \mathsf{y})\},$$

t is not local, and cannot be declared.

—

An Alternative Forward Assignment Rule (ASalt)

$$\frac{}{\{p\}\ v := e\ \{\exists v'.\ v = e' \wedge p'\}}$$

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

The problem with this rule is the accumulation of quantifiers. For example:

1. $\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} < \mathsf{n}\}\ \mathsf{k} := \mathsf{k} + 1$ (ASalt)
   $\{\exists \mathsf{k}'.\ \mathsf{k} = \mathsf{k}' + 1 \wedge \mathsf{y} = 2^{\mathsf{k}'} \wedge \mathsf{k}' < \mathsf{n}\}$
2. $\{\exists \mathsf{k}'.\ \mathsf{k} = \mathsf{k}' + 1 \wedge \mathsf{y} = 2^{\mathsf{k}'} \wedge \mathsf{k}' < \mathsf{n}\}\ \mathsf{y} := 2 \times \mathsf{y}$ (ASalt)
   $\{\exists \mathsf{y}'.\ \mathsf{y} = 2 \times \mathsf{y}' \wedge (\exists \mathsf{k}'.\ \mathsf{k} = \mathsf{k}' + 1 \wedge \mathsf{y}' = 2^{\mathsf{k}'} \wedge \mathsf{k}' < \mathsf{n})\}$
3. $\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} < \mathsf{n}\}\ \mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}$ (SQ 1)
   $\{\exists \mathsf{y}'.\ \mathsf{y} = 2 \times \mathsf{y}' \wedge (\exists \mathsf{k}'.\ \mathsf{k} = \mathsf{k}' + 1 \wedge \mathsf{y}' = 2^{\mathsf{k}'} \wedge \mathsf{k}' < \mathsf{n})\}$
4. $(\exists \mathsf{y}'.\ \mathsf{y} = 2 \times \mathsf{y}' \wedge (\exists \mathsf{k}'.\ \mathsf{k} = \mathsf{k}' + 1 \wedge \mathsf{y}' = 2^{\mathsf{k}'} \wedge \mathsf{k}' < \mathsf{n})) \Rightarrow$
   $(\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n})$ (VC)
5. $\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} < \mathsf{n}\}\ \mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}$ (WC 3,4)

—

An Alternative Rule for Conditionals (CDalt)

$$\frac{\{p_1\}\ c_1\ \{q\} \qquad \{p_2\}\ c_2\ \{q\}}{\{(b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2)\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{q\}}$$

An Instance

$$\frac{\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{odd(k)}\}\ \mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \neg\,\mathsf{odd(k)}\} \qquad \{\mathsf{y} = 2^{\mathsf{k}} \wedge \neg\,\mathsf{odd(k)}\}\ \mathbf{skip}\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \neg\,\mathsf{odd(k)}\}}{\begin{array}{l}\{(\mathsf{odd(k)} \Rightarrow \mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{odd(k)}) \wedge \\ \qquad (\neg\,\mathsf{odd(k)} \Rightarrow \mathsf{y} = 2^{\mathsf{k}} \wedge \neg\,\mathsf{odd(k)})\} \\ \mathbf{if}\ \mathsf{odd(k)}\ \mathbf{then}\ \mathsf{k} := \mathsf{k} + 1; \mathsf{y} := 2 \times \mathsf{y}\ \mathbf{else}\ \mathbf{skip} \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \neg\,\mathsf{odd(k)}\}\end{array}}$$

—

The Rule of Consequence (CONSEQ)

$$\frac{p \Rightarrow p' \qquad \{p'\}\ c\ \{q'\} \qquad q' \Rightarrow q}{\{p\}\ c\ \{q\}}$$

An Instance

$$\frac{\begin{array}{l}(\mathsf{n} \geq 0) \Rightarrow (1 = 2^0 \wedge 0 \leq \mathsf{n}) \\ \{1 = 2^0 \wedge 0 \leq \mathsf{n}\} \\ \mathsf{k} := 0\ ;\ \mathsf{y} := 1\ ; \\ \mathbf{while}\ \mathsf{k} \neq \mathsf{n}\ \mathbf{do}\ (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}) \\ \{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n} \wedge \neg\, \mathsf{k} \neq \mathsf{n}\} \\ (\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n} \wedge \neg\, \mathsf{k} \neq \mathsf{n}) \Rightarrow (\mathsf{y} = 2^{\mathsf{n}})\end{array}}{\begin{array}{l}\{\mathsf{n} \geq 0\} \\ \mathsf{k} := 0\ ;\ \mathsf{y} := 1\ ; \\ \mathbf{while}\ \mathsf{k} \neq \mathsf{n}\ \mathbf{do}\ (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}) \\ \{\mathsf{y} = 2^{\mathsf{n}}\}\end{array}}$$

—

Why Annotations Are Needed

Without annotations, it is not straightforward to construct a proof of a specification from the specification itself. For example, if we try to use the rule for sequential composition,

$$\frac{\{p\}\ c_1\ \{q\} \qquad \{q\}\ c_2\ \{r\}}{\{p\}\ c_1\ ;\ c_2\ \{r\},}$$

to obtain the main step of a proof of the specification

$$\begin{array}{l}\{\mathsf{n} \geq 0\} \\ (\mathsf{k} := 0\ ;\ \mathsf{y} := 1)\ ; \\ \mathbf{while}\ \mathsf{k} \neq \mathsf{n}\ \mathbf{do}\ (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y}) \\ \{\mathsf{y} = 2^{\mathsf{n}}\},\end{array}$$

there is no indication of what assertion should replace the metavariable q.

—

Why Annotations Are Needed (continued)

But if we were to change the rule to

$$\frac{\{p\}\ c_1\ \{q\} \qquad \{q\}\ c_2\ \{r\}}{\{p\}\ c_1\ ;\ \underline{\{q\}}\ c_2\ \{r\},}$$

then the new rule would require the annotation q to occur in the conclusion:

$\{\mathsf{n} \geq 0\}$
$(\mathsf{k} := 0\ ;\ \mathsf{y} := 1)\ ;$
$\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}$
$\mathbf{while}\ \mathsf{k} \neq \mathsf{n}\ \mathbf{do}\ (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y})$
$\{\mathsf{y} = 2^{\mathsf{n}}\}.$

Then, once q has been determined, the premisses must be

$\{\mathsf{n} \geq 0\}$
$(\mathsf{k} := 0\ ;\ \mathsf{y} := 1)\ ;$
$\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}$

and

$\{\mathsf{y} = 2^{\mathsf{k}} \wedge \mathsf{k} \leq \mathsf{n}\}$
$\mathbf{while}\ \mathsf{k} \neq \mathsf{n}\ \mathbf{do}$
$\quad (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{y} := 2 \times \mathsf{y})$
$\{\mathsf{y} = 2^{\mathsf{n}}\}.$

The basic trick is to add annotations to the conclusions of the inference rules so that the conclusion of each rule completely determines its premisses.

—

A Surprise

Fortunately, it isn't necessary to annotate every semicolon. Indeed, an annotated specification may sometimes 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

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

—

Weakest Preconditions

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.

—

Annotation Descriptions

We will write the *annotation description*

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

and say that $\mathcal{A}$ *establishes* $\{p\}\ c\ \{q\}$, when $\mathcal{A}$ is an annotated specification that determines the specification $\{p\}\ c\ \{q\}$ and shows how to obtain a formal proof of this specification.

(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.

—

Assignment (ASan)

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

Instances

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

and

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

—

Completeness

We say that an annotated specification is:

- *right-complete* if it ends with a postcondition,
- *left-complete* if it begins with a precondition,
- *complete* if it is both right- and left-complete.

Then

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

—

Sequential Composition (SQan)

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

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 (SPan)

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

For instance,

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

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

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

—

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 quantification rule (EQ) or the frame rule (FR), do not fit well into the framework of weakest assertions.
3. Intermediate assertions are often needed to simplify verification conditions.

—

Partial Correctness of **while** (WHan)

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

For instance,

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

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

—

Weakening Consequent (WCan)

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

For instance,

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

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

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

—

Skip (SKan)

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

Conditional (CDan)

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

Variable Declaration (DCan)

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

when v does not occur free in p or q.

—

More Structural Inference Rules

Vacuity (VAC)

$$\frac{}{\{\mathbf{false}\}\ c\ \{q\}}$$

For example,

1. $(\mathsf{s} = 0 \wedge \mathsf{a} - 1 \geq \mathsf{b} \wedge \mathsf{k} \geq \mathsf{b} \wedge \mathsf{k} < \mathsf{b}) \Rightarrow \mathbf{false}$ (VC)
2. $\{\mathbf{false}\}$ (VAC)
   $\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{s} := \mathsf{s} + \mathsf{k}$
   $\{\mathsf{s} = 0 \wedge \mathsf{a} - 1 \geq \mathsf{b} \wedge \mathsf{k} \geq \mathsf{b}\}$
3. $\{\mathsf{s} = 0 \wedge \mathsf{a} - 1 \geq \mathsf{b} \wedge \mathsf{k} \geq \mathsf{b} \wedge \mathsf{k} < \mathsf{b}\}$ (SP)
   $\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{s} := \mathsf{s} + \mathsf{k}$
   $\{\mathsf{s} = 0 \wedge \mathsf{a} - 1 \geq \mathsf{b} \wedge \mathsf{k} \geq \mathsf{b}\}$
4. $\{\mathsf{s} = 0 \wedge \mathsf{a} - 1 \geq \mathsf{b} \wedge \mathsf{k} \geq \mathsf{b}\}$ (WH)
   $\mathbf{while}\ \mathsf{k} < \mathsf{b}\ \mathbf{do}\ (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{s} := \mathsf{s} + \mathsf{k})$
   $\{\mathsf{s} = 0 \wedge \mathsf{a} - 1 \geq \mathsf{b} \wedge \mathsf{k} \geq \mathsf{b} \wedge \neg\, \mathsf{k} < \mathsf{b}\}$.

—

Disjunction (DISJ)

$$\frac{\{p_1\}\ c\ \{q\} \qquad \{p_2\}\ c\ \{q\}}{\{p_1 \vee p_2\}\ c\ \{q\}}$$

For example, from

$\{\mathsf{a} - 1 \leq \mathsf{b}\}$
$\mathsf{s} := 0\ ;\ \mathsf{k} := \mathsf{a} - 1\ ;$
$\{\mathsf{s} = \sum_{\mathsf{i}=\mathsf{a}}^{\mathsf{k}} \mathsf{i} \wedge \mathsf{k} \leq \mathsf{b}\}$
$\mathbf{while}\ \mathsf{k} < \mathsf{b}\ \mathbf{do}$
$\quad (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{s} := \mathsf{s} + \mathsf{k})$
$\{\mathsf{s} = \sum_{\mathsf{i}=\mathsf{a}}^{\mathsf{b}} \mathsf{i}\}$

$\{\mathsf{a} - 1 \geq \mathsf{b}\}$
$\mathsf{s} := 0\ ;\ \mathsf{k} := \mathsf{a} - 1\ ;$
$\{\mathsf{s} = 0 \wedge \mathsf{a} - 1 \geq \mathsf{b} \wedge \mathsf{k} \geq \mathsf{b}\}$
$\mathbf{while}\ \mathsf{k} < \mathsf{b}\ \mathbf{do}$
$\quad (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{s} := \mathsf{s} + \mathsf{k})$
$\{\mathsf{s} = 0 \wedge \mathsf{a} - 1 \geq \mathsf{b}\}$
$\{\mathsf{s} = \sum_{\mathsf{i}=\mathsf{a}}^{\mathsf{b}} \mathsf{i}\}.$

we can obtain the main step in

$\{\mathbf{true}\}$
$\{\mathsf{a} - 1 \leq \mathsf{b} \vee \mathsf{a} - 1 \geq \mathsf{b}\}$
$\mathsf{s} := 0\ ;\ \mathsf{k} := \mathsf{a} - 1\ ;$
$\mathbf{while}\ \mathsf{k} < \mathsf{b}\ \mathbf{do}$
$\quad (\mathsf{k} := \mathsf{k} + 1\ ;\ \mathsf{s} := \mathsf{s} + \mathsf{k})$
$\{\mathsf{s} = \sum_{\mathsf{i}=\mathsf{a}}^{\mathsf{b}} \mathsf{i}\}.$

—

Conjunction (CONJ)

$$\frac{\{p_1\}\ c\ \{q_1\} \qquad \{p_2\}\ c\ \{q_2\}}{\{p_1 \wedge p_2\}\ c\ \{q_1 \wedge q_2\}}$$

Existential Quantification (EQ)

$$\frac{\{p\}\ c\ \{q\}}{\{\exists v.\ p\}\ c\ \{\exists v.\ q\},}$$

where v is not free in c (i.e., v is a ghost variable).

Universal Quantification (UQ)

$$\frac{\{p\}\ c\ \{q\}}{\{\forall v.\ p\}\ c\ \{\forall v.\ q\},}$$

where v is not free in c (i.e., v is a ghost variable).

—

Substitution (SUB)

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

where

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

For 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

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

—

Renaming (RN)

$$\frac{\{p\}\ \mathbf{newvar}\ v\ \mathbf{in}\ c\ \{q\}}{\{p\}\ \mathbf{newvar}\ v'\ \mathbf{in}\ (c/v \to v')\ \{q\},}$$

where v' does not occur free in c. The only time it is necessary to use this rule is when one must prove a specification of a variable declaration that violates the proviso that the variable being declared must not occur free in the pre- or postcondition. For example,

1. $\{\mathsf{x} = 0\}\ \mathsf{y} := 1\ \{\mathsf{x} = 0\}$ (AS)
2. $\{\mathsf{x} = 0\}\ \mathbf{newvar}\ \mathsf{y}\ \mathbf{in}\ \mathsf{y} := 1\ \{\mathsf{x} = 0\}$ (DC 1)
3. $\{\mathsf{x} = 0\}\ \mathbf{newvar}\ \mathsf{x}\ \mathbf{in}\ \mathsf{x} := 1\ \{\mathsf{x} = 0\}.$ (RN 2)

—

The Frame Rule (O'Hearn) (FR)

$$\frac{\{p\}\ c\ \{q\}}{\{p\ *\ r\}\ c\ \{q\ *\ r\}},$$

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

For instance,

$$\frac{\{\mathbf{list}\ \alpha\ \mathsf{i}\}\ \text{“Reverse List”}\ \{\mathbf{list}\ \alpha^{\dagger}\ \mathsf{j}\}}{\{\mathbf{list}\ \alpha\ \mathsf{i}\ *\ \mathbf{list}\ \gamma\ \mathsf{x}\}\ \text{“Reverse List”}\ \{\mathbf{list}\ \alpha^{\dagger}\ \mathsf{j}\ *\ \mathbf{list}\ \gamma\ \mathsf{x}\},}$$

(assuming “Reverse List” does not modify x or γ).

—

The Delicacy of the Frame Rule

Suppose

$$\{\mathbf{emp}\}\ \mathbf{dispose}\ \mathsf{x}\ \{\mathbf{emp}\}.$$

Then the frame rule would give

$$\{\mathbf{emp}\ *\ \mathsf{x} \mapsto 10\}\ \mathbf{dispose}\ \mathsf{x}\ \{\mathbf{emp}\ *\ \mathsf{x} \mapsto 10\},$$

and therefore

$$\{\mathsf{x} \mapsto 10\}\ \mathbf{dispose}\ \mathsf{x}\ \{\mathsf{x} \mapsto 10\},$$

which is patently false.

—

Why the Frame Rule is Sound

We define:

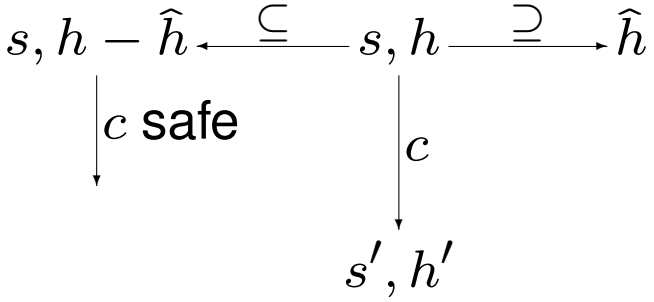
If, starting in the state s, h, no execution of a command c aborts, then c is *safe at* s, h.

If, starting in the state s, h, every execution of c terminates without aborting, then c *must terminate normally* at s, h.

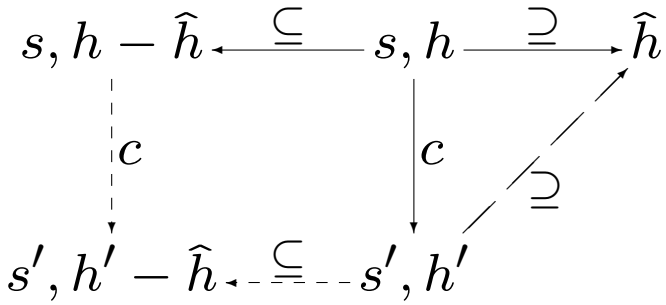
Then our programming language satisfies:

Safety Monotonicity If $\widehat{h} \subseteq h$ and c is safe at $s, h - \widehat{h}$, then c is safe at s, h. If $\widehat{h} \subseteq h$ and c must terminate normally at $s, h - \widehat{h}$, then c must terminate normally at s, h.

The Frame Property If $\widehat{h} \subseteq h$, c is safe at $s, h - \widehat{h}$, and some execution of c starting at s, h terminates normally in the state s', h',



then $\widehat{h} \subseteq h'$ and some execution of c starting at $s, h - \widehat{h}$, terminates normally in the state $s', h' - \widehat{h}$.



Why the Frame Rule is Sound (continued)

Proposition 11 *If the programming language satisfies safety monotonicity and the frame property, then the frame rule is sound for both partial and total correctness.*

PROOF Assume $\{p\}\ c\ \{q\}$ is valid, and $s, h \models p * r$. Then there is an $\widehat{h} \subseteq h$ such that $s, h - \widehat{h} \models p$ and $s, \widehat{h} \models r$.

From $\{p\}\ c\ \{q\}$, we know that c is safe at $s, h - \widehat{h}$ (and in the total-correctness case it must terminate normally). Then, by safety monotonicity, we know that c is safe at s, h (and in the total-correctness case it must terminate normally).

Suppose that, starting in the state s, h, there is an execution of c that terminates in the state s', h'. Since c is safe at $s, h - \widehat{h}$, we know by the frame property that $\widehat{h} \subseteq h'$ and that, starting in the state $s, h - \widehat{h}$, there is some execution of c that terminates in the state $s', h' - \widehat{h}$. Then $\{p\}\ c\ \{q\}$ and $s, h - \widehat{h} \models p$ imply that $s', h' - \widehat{h} \models q$.

Since an execution of c carries s, h into s', h', we know that $s\,v = s'v$ for all v that are not modified by c. Then, since these include the free variables of r, $s, \widehat{h} \models r$ implies that $s', \widehat{h} \models r$. Thus $s', h' \models q * r$. END OF PROOF

—

More Rules for Annotated Specifications

We now give annotated-specification versions of the additional structural rules we have just introduced.

In all of these rules, we assume that the annotated specifications in the premisses will often be sequences of several lines.

- In the unary rules (VACan), (EQan), (UQan), (FRan), and (SUBan), braces are used to indicate the vertical extent of the single operand.
- In the binary rules (DISJan), and (CONJan), the two operands are placed symmetrically around the indicator DISJ or CONJ.

—

Vacuity (VACan)

$$\frac{}{\{c\}\mathsf{VAC}\,\{q\} \gg \{\mathbf{false}\}\ c\ \{q\}.}$$

Here c contains no annotations, since no reasoning about its subcommands is used. For example, using (VACan), (SPan), (WHan), and (WCan):

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

—

Disjunction (DISJan)

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

For example,

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

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

$$\{\mathsf{s} = 0 \wedge \mathsf{a} - 1 \geq \mathsf{b}\}.$$

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

—

In the remaining structural rules, the annotated specification in the conclusion need not be left- or right-complete; it simply contains the annotated specifications of the premisses.

Conjunction (CONJan)

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

Existential Quantification (EQan)

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

where v is not free in c.

Universal Quantification (UQan)

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

where v is not free in c.

In using the two rules above, we will often abbreviate

$$\Big\{\cdots\Big\{\mathcal{A}\Big\}\exists v_1\cdots\Big\}\exists v_n \text{ by } \Big\{\mathcal{A}\Big\}\exists v_1\ldots,v_n$$

$$\Big\{\cdots\Big\{\mathcal{A}\Big\}\forall v_1\cdots\Big\}\forall v_n \text{ by } \Big\{\mathcal{A}\Big\}\forall v_1\ldots,v_n.$$

—

Frame (FRan)

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

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

An Example

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

—

Substitution (SUBan)

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

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

—

Inference Rules for Mutation

The local form (MUL):

$$\frac{}{\{e \mapsto -\}\ [e] := e'\ \{e \mapsto e'\}}.$$

The global form (MUG):

$$\frac{}{\{(e \mapsto -)\ *\ r\}\ [e] := e'\ \{(e \mapsto e')\ *\ r\}}.$$

The backward-reasoning form (MUBR):

$$\frac{}{\{(e \mapsto -)\ *\ ((e \mapsto e') \mathrel{-\!\!*} p)\}\ [e] := e'\ \{p\}}.$$

—

The local form (MUL):

$$\frac{}{\{e \mapsto -\}\ [e] := e'\ \{e \mapsto e'\}}.$$

The global form (MUG):

$$\frac{}{\{(e \mapsto -)\ *\ r\}\ [e] := e'\ \{(e \mapsto e')\ *\ r\}}.$$

One can derive (MUG) from (MUL) by using the frame rule:

$$\begin{array}{l} \{(e \mapsto -)\ *\ r\} \\ \left.\begin{array}{l} \{e \mapsto -\} \\ [e] := e' \\ \{e \mapsto e'\} \end{array}\right\} *\ r \\ \{(e \mapsto e')\ *\ r\}, \end{array}$$

—

The local form (MUL):

$$\frac{}{\{e \mapsto -\}\ [e] := e'\ \{e \mapsto e'\}}.$$

The global form (MUG):

$$\frac{}{\{(e \mapsto -)\ *\ r\}\ [e] := e'\ \{(e \mapsto e')\ *\ r\}}.$$

To go in the opposite direction it is only necessary to take r to be $\mathbf{emp}$:

$$\begin{array}{l}\{e \mapsto -\}\\ \{(e \mapsto -)\ *\ \mathbf{emp}\}\\ [e] := e'\\ \{(e \mapsto e')\ *\ \mathbf{emp}\}\\ \{e \mapsto e'\}.\end{array}$$

—

The global form (MUG):

$$\frac{}{\{(e \mapsto -) \ * \ r\} \ [e] := e' \ \{(e \mapsto e') \ * \ r\}.}$$

The backward-reasoning form (MUBR):

$$\frac{}{\{(e \mapsto -) \ * \ ((e \mapsto e') \mathrel{-\!\!*} p)\} \ [e] := e' \ \{p\}.}$$

One can derive (MUBR) from (MUG) by taking r to be $(e \mapsto e') \mathrel{-\!\!*} p$ and using the law $q \ * \ (q \mathrel{-\!\!*} p) \Rightarrow p$:

$\{(e \mapsto -) \ * \ ((e \mapsto e') \mathrel{-\!\!*} p)\}$
$[e] := e'$
$\{(e \mapsto e') \ * \ ((e \mapsto e') \mathrel{-\!\!*} p)\}$
$\{p\}.$

—

The global form (MUG):

$$\frac{}{\{(e \mapsto -) \ * \ r\}\ [e] := e'\ \{(e \mapsto e') \ * \ r\}.}$$

The backward-reasoning form (MUBR):

$$\frac{}{\{(e \mapsto -) \ * \ ((e \mapsto e') \mathrel{-\!\!*} p)\}\ [e] := e'\ \{p\}.}$$

One can go in the opposite direction by taking p to be $(e \mapsto e') \ * \ r$ and using $(p \ * \ r) \Rightarrow (p \ * \ (q \mathrel{-\!\!*} (q \ * \ r)))$:

$\{(e \mapsto -) \ * \ r\}$
$\{(e \mapsto -) \ * \ ((e \mapsto e') \mathrel{-\!\!*} ((e \mapsto e') \ * \ r))\}$
$[e] := e'$
$\{(e \mapsto e') \ * \ r\}.$

—

Inference Rules for Deallocation

The local form (DISL):

$$\frac{}{\{e \mapsto -\}\ \mathbf{dispose}\ e\ \{\mathbf{emp}\}}.$$

The global (and backward-reasoning) form (DISBR):

$$\frac{}{\{(e \mapsto -)\ *\ r\}\ \mathbf{dispose}\ e\ \{r\}}.$$

One can derive (DISBR) from (DISL) by using (FR); one can go in the opposite direction by taking r to be $\mathbf{emp}$.

—

Inference Rules for Allocation and Lookup

These are *generalized assignment commands*, but they don't obey the assignment rule, e.g.

$$\{\mathbf{cons}(1,2) = \mathbf{cons}(1,2)\}\ \mathsf{x} := \mathbf{cons}(1,2)\ \{\mathsf{x} = \mathsf{x}\}$$

$\uparrow$

syntactically illegal

$\downarrow$

$$\{[\mathsf{y}] = [\mathsf{y}]\}\ \mathsf{x} := [\mathsf{y}]\ \{\mathsf{x} = \mathsf{x}\}$$

—

Inference Rules for Nonoverwriting Allocation

We abbreviate the sequence $e_1, \ldots, e_n$ of expressions by $\overline{e}$:

- The local form (CONSNOL)

$$\frac{}{\{\mathbf{emp}\}\ v := \mathbf{cons}(\overline{e})\ \{v \mapsto \overline{e}\}},$$

  where $v \notin \mathsf{FV}(\overline{e})$.

- The global form (CONSNOG)

$$\frac{}{\{r\}\ v := \mathbf{cons}(\overline{e})\ \{(v \mapsto \overline{e})\ *\ r\}},$$

  where $v \notin \mathsf{FV}(\overline{e}, r)$.

Again, one can derive the global form from the local by using the frame rule, and the local from the global by taking r to be $\mathbf{emp}$.

—

Inference Rules for General Allocation

- The local form (CONSL)

$$\frac{}{\{v = v' \wedge \mathbf{emp}\}\ v := \mathbf{cons}(\overline{e})\ \{v \mapsto \overline{e}'\},}$$

  where v' is distinct from v, and $\overline{e}'$ denotes $\overline{e}/v \rightarrow v'$ (i.e., each e_i' denotes $e_i/v \rightarrow v'$).

- The global form (CONSG)

$$\frac{}{\{r\}\ v := \mathbf{cons}(\overline{e})\ \{\exists v'.\ (v \mapsto \overline{e}')\ *\ r'\},}$$

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

- The backward-reasoning form (CONSBR)

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

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

—

An Instance of (CONSG)

$$\frac{}{\{r\}\ v := \mathbf{cons}(\overline{e})\ \{\exists v'.\ (v \mapsto \overline{e}')\ *\ r'\},}$$

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

An Instance:

$$\{\mathsf{list}\ \alpha\ \mathsf{i}\}\ \mathsf{i} := \mathbf{cons}(3, \mathsf{i})\ \{\exists \mathsf{j}.\ \mathsf{i} \mapsto 3, \mathsf{j}\ *\ \mathsf{list}\ \alpha\ \mathsf{j}\}.$$

—

Deriving (CONSG) from (CONSNOG)

We can define overwriting general assignments in terms of nonoverwriting ones by

$$v := \mathbf{cons}(\overline{e}) \cong \mathbf{newvar}\ \widehat{v}\ \mathbf{in}\ (\widehat{v} := \mathbf{cons}(\overline{e})\ ;\ v := \widehat{v}),$$

where $\widehat{v}$ does not occur in $\overline{e}$ — and can be chosen not to occur in other specified phrases.

Then from (CONSNOG):

$$\frac{}{\{r\}\ v := \mathbf{cons}(\overline{e})\ \{(v \mapsto \overline{e})\ *\ r\}},$$

where $v \notin \mathsf{FV}(\overline{e}, r)$,

we can derive (CONSG):

$$\frac{}{\{r\}\ v := \mathbf{cons}(\overline{e})\ \{\exists v'.\ (v \mapsto \overline{e}')\ *\ r'\}},$$

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

by

$$\begin{array}{l}
\{r\} \\
\mathbf{newvar}\ \widehat{v}\ \mathbf{in} \\
\quad \big(\widehat{v} := \mathbf{cons}(\overline{e})\ ; \\
\quad \{(\widehat{v} \mapsto \overline{e})\ *\ r\} \\
\quad \{\exists v.\ (\widehat{v} \mapsto \overline{e})\ *\ r\} \\
\quad \{\exists v'.\ (\widehat{v} \mapsto \overline{e}')\ *\ r'\} \\
\quad v := \widehat{v}\big) \\
\{\exists v'.\ (v \mapsto \overline{e}')\ *\ r'\}
\end{array}$$

—

An Inadequate Local Rule

From (CONSG):

$$\frac{}{\{r\}\ v := \mathbf{cons}(\overline{e})\ \{\exists v'.\ (v \mapsto \overline{e}')\ *\ r'\}},$$

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

we can derive

$$\frac{}{\{\mathbf{emp}\}\ v := \mathbf{cons}(\overline{e})\ \{\exists v'.\ (v \mapsto \overline{e}')\}},$$

where v' is distinct from v and $v' \notin \mathsf{FV}(\overline{e})$.

by taking r to be $\mathbf{emp}$.

But this rule is too weak. For instance,

$$\{\mathbf{emp}\}\ \mathsf{i} := \mathbf{cons}(3, \mathsf{i})\ \{\exists \mathsf{j}.\ \mathsf{i} \mapsto 3, \mathsf{j}\}$$

gives no information about the second component of the new record.

—

An Adequate Local Rule

- The local form (CONSL)

$$\frac{}{\{v = v' \wedge \mathbf{emp}\}\ v := \mathbf{cons}(\overline{e})\ \{v \mapsto \overline{e}'\},}$$

  where v' is distinct from v, and $\overline{e}'$ denotes $\overline{e}/v \to v'$.

Here the existential quantifier is dropped and v' becomes a variable that is not modified by $v := \mathbf{cons}(\overline{e})$. For example,

$$\{\mathsf{i} = \mathsf{j} \wedge \mathbf{emp}\}\ \mathsf{i} := \mathbf{cons}(3, \mathsf{i})\ \{\mathsf{i} \mapsto 3, \mathsf{j}\}$$

shows that the value of j in the postcondition is value of i before the assignment.

—

From (CONSG) to (CONSL)

From (CONSG):

$$\frac{}{\{r\}\ v := \mathbf{cons}(\overline{e})\ \{\exists v'.\ (v \mapsto \overline{e}')\ *\ r'\}},$$

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

we derive (CONSL):

$$\frac{}{\{v = v' \wedge \mathbf{emp}\}\ v := \mathbf{cons}(\overline{e})\ \{v \mapsto \overline{e}'\}},$$

where v' is distinct from v, and $\overline{e}'$ denotes $\overline{e}/v \to v'$.

by replacing r by $v = v' \wedge \mathbf{emp}$ and v' by v'', and using the fact that $v'' = v'$ is pure, plus simple properties of equality and the existential quantifier:

$$\begin{array}{l}
\{v = v' \wedge \mathbf{emp}\} \\
v := \mathbf{cons}(\overline{e}) \\
\{\exists v''.\ (v \mapsto \overline{e}'')\ *\ (v'' = v' \wedge \mathbf{emp})\} \\
\{\exists v''.\ ((v \mapsto \overline{e}'') \wedge v'' = v')\ *\ \mathbf{emp}\} \\
\{\exists v''.\ (v \mapsto \overline{e}'') \wedge v'' = v'\} \\
\{\exists v''.\ (v \mapsto \overline{e}') \wedge v'' = v'\} \\
\{v \mapsto \overline{e}'\}.
\end{array}$$

(Here v'' is chosen to be distinct from v, v', and the free variables of $\overline{e}$.)

—

From (CONSL) to (CONSG)

From (CONSL):

$$\frac{}{\{v = v' \wedge \mathbf{emp}\}\ v := \mathbf{cons}(\overline{e})\ \{v \mapsto \overline{e}'\},}$$

where v' is distinct from v, and $\overline{e}'$ denotes $\overline{e}/v \to v'$.

we derive (CONSG):

$$\frac{}{\{r\}\ v := \mathbf{cons}(\overline{e})\ \{\exists v'.\ (v \mapsto \overline{e}')\ *\ r'\},}$$

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

by using the frame rule and (EQ):

$$\begin{array}{l}
\{r\} \\
\{\exists v'.\ v = v' \wedge r\} \\
\left.\begin{array}{l}
\quad \{v = v' \wedge r\} \\
\quad \{v = v' \wedge r'\} \\
\quad \{v = v' \wedge (\mathbf{emp}\ *\ r')\} \\
\quad \{(v = v' \wedge \mathbf{emp})\ *\ r'\} \\
\qquad \left.\begin{array}{l} \{(v = v' \wedge \mathbf{emp})\} \\ v := \mathbf{cons}(\overline{e}) \\ \{(v \mapsto \overline{e}')\} \end{array}\right\} *\ r' \\
\quad \{(v \mapsto \overline{e}')\ *\ r'\}
\end{array}\right\} \exists v' \\
\{\exists v'.\ (v \mapsto \overline{e}')\ *\ r'\}
\end{array}$$

—

Soundness of the Backward Reasoning Rule (CONSBR)

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

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

Suppose the precondition holds in the state s, h, i.e., that

$$s, h \vDash \forall v''.\ (v'' \mapsto \overline{e}) \mathrel{-\!\!*} p''.$$

Then the semantics of universal quantification gives

$$\forall \ell.\ [\, s \mid v'' \colon \ell \,], h \vDash (v'' \mapsto \overline{e}) \mathrel{-\!\!*} p'',$$

and the semantics of separating implication gives

$\forall \ell, h'.\ h \perp h'$ and $\underline{[\, s \mid v'' \colon \ell \,], h' \vDash (v'' \mapsto \overline{e})}$ implies

$$[\, s \mid v'' \colon \ell \,], h \cdot h' \vDash p'',$$

where the underlined formula is equivalent to

$$h' = [\, \ell \colon [\![e_1]\!]_{\mathsf{exp}} s \mid \ldots \mid \ell + n - 1 \colon [\![e_n]\!]_{\mathsf{exp}} s \,].$$

Thus

$\forall \ell.\ \big(\ell, \ldots, \ell + n - 1 \notin \mathsf{dom}\, h$ implies

$$[\, s \mid v'' \colon \ell \,], [\, h \mid \ell \colon [\![e_1]\!]_{\mathsf{exp}} s \mid \ldots \mid \ell + n - 1 \colon [\![e_n]\!]_{\mathsf{exp}} s \,] \vDash p''\big).$$

—

The Soundness of (CONSBR) (continued)

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

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

Then, by the substitution law for assertions, since p'' denotes $p/v \to v''$, we have

$$[\, s \mid v'' \colon \ell \,], h{\cdot}h' \vDash p'' \text{ iff } \widehat{s}, h{\cdot}h' \vDash p,$$

where

$$\widehat{s} = [\, s \mid v'' \colon \ell \mid v \colon [\![v'']\!]_{\mathsf{exp}} [\, s \mid v'' \colon \ell \,] \,] = [\, s \mid v'' \colon \ell \mid v \colon \ell \,].$$

Moreover, since v'' does not occur free in p, we can simplify $\widehat{s}, h{\cdot}h' \vDash p$ to $[\, s \mid v \colon \ell \,], h{\cdot}h' \vDash p$. Thus

$$\forall \ell.\ \big(\ell, \ldots, \ell + n - 1 \notin \mathsf{dom}\, h \text{ implies}$$
$$[\, s \mid v \colon \ell \,], [\, h \mid \ell \colon [\![e_1]\!]_{\mathsf{exp}} s \mid \ldots \mid \ell + n - 1 \colon [\![e_n]\!]_{\mathsf{exp}} s \,] \vDash p\big).$$

Now execution of the allocation command $v := \mathbf{cons}(\overline{e})$, starting in the state s, h, will never abort, and will always termininate in a state

$$[\, s \mid v \colon \ell \,], [\, h \mid \ell \colon [\![e_1]\!]_{\mathsf{exp}} s \mid \ldots \mid \ell + n - 1 \colon [\![e_n]\!]_{\mathsf{exp}} s \,]$$

for some ℓ such that $\ell, \ldots, \ell + n - 1 \notin \mathsf{dom}\, h$. Thus the condition displayed above insures that all possible terminating states satisfy the postcondition p.

—

From (CONSG) to (CONSBR)

From (CONSG):

$$\frac{}{\{r\}\ v := \mathbf{cons}(\overline{e})\ \{\exists v'.\ (v \mapsto \overline{e}')\ *\ r'\}},$$

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

we derive (CONSBR):

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

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

by choosing $v' \notin \mathsf{FV}(\overline{e}, p)$ to be distinct from v and v'', taking r to be $\forall v''.\ (v'' \mapsto \overline{e}) \mathrel{-\!\!*} p''$, and using predicate-calculus properties of quantifiers, and $q\ *\ (q \mathrel{-\!\!*} p) \Rightarrow p$:

$$\begin{array}{l}
\{\forall v''.\ (v'' \mapsto \overline{e}) \mathrel{-\!\!*} p''\} \\
v := \mathbf{cons}(\overline{e}) \\
\{\exists v'.\ (v \mapsto \overline{e}')\ *\ (\forall v''.\ (v'' \mapsto \overline{e}') \mathrel{-\!\!*} p'')\} \\
\{\exists v'.\ (v \mapsto \overline{e}')\ *\ ((v \mapsto \overline{e}') \mathrel{-\!\!*} p)\} \\
\{\exists v'.\ p\} \\
\{p\}.
\end{array}$$

—

From (CONSBR) to (CONSG)

From (CONSBR):

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

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

we derive (CONSG):

$$\frac{}{\{r\}\ v := \mathbf{cons}(\overline{e})\ \{\exists v'.\ (v \mapsto \overline{e}')\ *\ r'\}},$$

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

by choosing $v'' \notin \mathsf{FV}(\overline{e}, r)$ to be distinct from v and v', taking p to be $\exists v'.\ (v \mapsto \overline{e}')\ *\ r'$, and using properties of quantifiers, as well as $r \Rightarrow (q \mathrel{-\!\!*} (q\ *\ r))$:

$$\begin{array}{l}
\{r\} \\
\{\forall v''.\ r\} \\
\{\forall v''.\ (v'' \mapsto \overline{e}) \mathrel{-\!\!*} ((v'' \mapsto \overline{e})\ *\ r)\} \\
\{\forall v''.\ (v'' \mapsto \overline{e}) \mathrel{-\!\!*} (((v'' \mapsto \overline{e}')\ *\ r')/v' \to v)\} \\
\{\forall v''.\ (v'' \mapsto \overline{e}) \mathrel{-\!\!*} (\exists v'.\ (v'' \mapsto \overline{e}')\ *\ r')\} \\
v := \mathbf{cons}(\overline{e}) \\
\{\exists v'.\ (v \mapsto \overline{e}')\ *\ r'\}.
\end{array}$$

—

Inference Rules for Nonoverwriting Lookup

- The local nonoverwriting form (LKNOL)

$$\frac{}{\{e \mapsto v''\}\ v := [e]\ \{v = v'' \wedge (e \mapsto v)\}},$$

  where $v \notin \mathsf{FV}(e)$.

- The global nonoverwriting form (LKNOG)

$$\frac{}{\{\exists v''.\ (e \mapsto v'')\ *\ p''\}\ v := [e]\ \{(e \mapsto v)\ *\ p\}},$$

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

—

In (LKNOG):

$$\frac{}{\{\exists v''.\ (e \mapsto v'')\ *\ p''\}\ v := [e]\ \{(e \mapsto v)\ *\ p\},}$$

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

there is no restriction preventing v'' from being the same variable as v. Thus, as a special case,

$$\frac{}{\{\exists v.\ (e \mapsto v)\ *\ p\}\ v := [e]\ \{(e \mapsto v)\ *\ p\},}$$

where $v \notin \mathsf{FV}(e)$. For example, if we take

v to be j
e to be i + 1
p to be $\mathsf{i} \mapsto 3\ *\ \mathbf{list}\ \alpha\ \mathsf{j}$,

(and remember $\mathsf{i} \mapsto 3, \mathsf{j}$ abbreviates $(\mathsf{i} \mapsto 3) * (\mathsf{i} + 1 \mapsto \mathsf{j})$), then we obtain the instance

$$\{\exists \mathsf{j}.\ \mathsf{i} \mapsto 3, \mathsf{j}\ *\ \mathbf{list}\ \alpha\ \mathsf{j}\}\ \mathsf{j} := [\mathsf{i} + 1]\ \{\mathsf{i} \mapsto 3, \mathsf{j}\ *\ \mathbf{list}\ \alpha\ \mathsf{j}\}.$$

—

Inference Rules for General Lookup

- The local form (LKL)

$$\frac{}{\{v = v' \wedge (e \mapsto v'')\}\ v := [e]\ \{v = v'' \wedge (e' \mapsto v)\},}$$

where v, v', and v'' are distinct, and e' denotes $e/v \rightarrow v'$.

- The global form (LKG)

$$\frac{}{\{\exists v''.\ (e \mapsto v'')\ *\ (r/v' \rightarrow v)\}\ v := [e]\ \{\exists v'.\ (e' \mapsto v)\ *\ (r/v'' \rightarrow v)\},}$$

where v, v', and v'' are distinct, v', $v'' \notin \mathsf{FV}(e)$, $v \notin \mathsf{FV}(r)$, and e' denotes $e/v \rightarrow v'$.

- The first backward-reasoning form (LKBR1)

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

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

- The second backward-reasoning form (LKBR2)

$$\frac{}{\{\exists v''.\ (e \hookrightarrow v'') \wedge p''\}\ v := [e]\ \{p\},}$$

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

The Soundness of the Local Rule (LKL)

$$\frac{}{\{v = v' \wedge (e \mapsto v'')\}\ v := [e]\ \{v = v'' \wedge (e' \mapsto v)\},}$$

where v, v', and v'' are distinct, and e' denotes $e/v \to v'$.

Suppose that the precondition holds in the state s_0, h, i.e., that

$$s_0, h \vDash v = v' \wedge (e \mapsto v'').$$

Then $s_0\, v = s_0\, v'$ and $h = [\, [\![e]\!]_{\mathsf{exp}} s_0 \colon s_0\, v'' \,]$.

Starting in the state s_0, h, the execution of $v := [e]$ will not abort (since $[\![e]\!]_{\mathsf{exp}} s_0 \in \mathsf{dom}\, h$), and will terminate with the store

$$s_1 = [\, s_0 \mid v \colon s_0\, v'' \,]$$

and the unchanged heap h. To see that this state satisfies the postcondition, we note that $s_1\, v = s_0\, v'' = s_1\, v''$ and, since e' does not contain v, $[\![e']\!]_{\mathsf{exp}} s_1 = [\![e']\!]_{\mathsf{exp}} s_0$. Then applying the substitution law for assertions, with

$$\widehat{s} = [\, s_0 \mid v \colon s_0\, v' \,] = [\, s_0 \mid v \colon s_0\, v \,] = s_0,$$

we obtain $[\![e']\!]_{\mathsf{exp}} s_0 = [\![e]\!]_{\mathsf{exp}} s_0$. Thus

$$h = [\, [\![e']\!]_{\mathsf{exp}} s_1 \colon s_1\, v \,] \quad \text{and} \quad s_1, h \vDash v = v'' \wedge (e' \mapsto v).$$

—

From (LKL) to (LKG)

From (LKL):

$$\frac{}{\{v = v' \wedge (e \mapsto v'')\}\ v := [e]\ \{v = v'' \wedge (e' \mapsto v)\}},$$

where v, v', and v'' are distinct, and e' denotes $e/v \to v'$.

we derive (LKG):

$$\frac{}{\begin{array}{l}\{\exists v''.\ (e \mapsto v'')\ *\ (r/v' \to v)\}\ v := [e] \\ \qquad\qquad \{\exists v'.\ (e' \mapsto v)\ *\ (r/v'' \to v)\},\end{array}}$$

where v, v', and v'' are distinct, $v', v'' \notin \mathsf{FV}(e)$, $v \notin \mathsf{FV}(r)$, and e' denotes $e/v \to v'$.

by using the frame rule and two applications of (EQ):

$$\begin{array}{l}
\{\exists v''.\ (e \mapsto v'')\ *\ (r/v' \to v)\} \\
\{\exists v', v''.\ (v = v' \wedge (e \mapsto v''))\ *\ (r/v' \to v)\} \\
\left.\begin{array}{l}
\{(v = v' \wedge (e \mapsto v''))\ *\ (r/v' \to v)\} \\
\left.\begin{array}{l}\{v = v' \wedge (e \mapsto v'')\} \\ v := [e] \\ \{v = v'' \wedge (e' \mapsto v)\}\end{array}\right\}\ *\ r \\
\{(v = v'' \wedge (e' \mapsto v))\ *\ (r/v'' \to v)\}
\end{array}\right\}\ \exists v', v'' \\
\{\exists v', v''.\ (v = v'' \wedge (e' \mapsto v))\ *\ (r/v'' \to v)\} \\
\{\exists v'.\ (e' \mapsto v)\ *\ (r/v'' \to v)\}.
\end{array}$$

—

An Instance of (LKG)

$$\frac{}{\{\exists v''.\ (e \mapsto v'')\ *\ (r/v' \to v)\}\ v := [e]\ \{\exists v'.\ (e' \mapsto v)\ *\ (r/v'' \to v)\}},$$

where v, v', and v'' are distinct, $v', v'' \notin \mathsf{FV}(e)$, $v \notin \mathsf{FV}(r)$, and e' denotes $e/v \to v'$.

As an example of an instance, if we take

v to be j
v' to be m
v'' to be k

e to be $\mathsf{j} + 1$
r to be $\mathsf{i} + 1 \mapsto \mathsf{m}\ *\ \mathsf{k} + 1 \mapsto \mathbf{nil}$,

then we obtain (using the commutivity of $*$)

$$\{\exists \mathsf{k}.\ \mathsf{i} + 1 \mapsto \mathsf{j}\ *\ \mathsf{j} + 1 \mapsto \mathsf{k}\ *\ \mathsf{k} + 1 \mapsto \mathbf{nil}\}$$
$$\mathsf{j} := [\mathsf{j} + 1]$$
$$\{\exists \mathsf{m}.\ \mathsf{i} + 1 \mapsto \mathsf{m}\ *\ \mathsf{m} + 1 \mapsto \mathsf{j}\ *\ \mathsf{j} + 1 \mapsto \mathbf{nil}\}.$$

—

From (LKG) to (LKL)

From (LKG):

$$\frac{}{\{\exists v''.\ (e \mapsto v'')\ *\ (r/v' \to v)\}\ v := [e]\ \{\exists v'.\ (e' \mapsto v)\ *\ (r/v'' \to v)\}}$$

where v, v', and v'' are distinct, $v', v'' \notin \mathsf{FV}(e)$, $v \notin \mathsf{FV}(r)$, and e' denotes $e/v \to v'$.

we derive (LKL):

$$\frac{}{\{v = v' \wedge (e \mapsto v'')\}\ v := [e]\ \{v = v'' \wedge (e' \mapsto v)\},}$$

where v, v', and v'' are distinct, and e' denotes $e/v \to v'$.

by first renaming the variables v' and v'' in (LKG) to be $\widehat{v}'$ and $\widehat{v}''$, chosen not to occur free in e, and then replace r in (LKG) by $\widehat{v}' = v' \wedge \widehat{v}'' = v'' \wedge \mathbf{emp}$. Then:

$\{v = v' \wedge (e \mapsto v'')\}$
$\{\exists \widehat{v}''.\ v = v' \wedge \widehat{v}'' = v'' \wedge (e \mapsto \widehat{v}'')\}$
$\{\exists \widehat{v}''.\ (e \mapsto \widehat{v}'')\ *\ (v = v' \wedge \widehat{v}'' = v'' \wedge \mathbf{emp})\}$
$\{\exists \widehat{v}''.\ (e \mapsto \widehat{v}'')\ *\ ((\widehat{v}' = v' \wedge \widehat{v}'' = v'' \wedge \mathbf{emp})/\widehat{v}' \to v)\}$
$v := [e]$
$\{\exists \widehat{v}'.\ (\widehat{e}' \mapsto v)\ *\ ((\widehat{v}' = v' \wedge \widehat{v}'' = v'' \wedge \mathbf{emp})/\widehat{v}'' \to v)\}$
$\{\exists \widehat{v}'.\ (\widehat{e}' \mapsto v)\ *\ (\widehat{v}' = v' \wedge v = v'' \wedge \mathbf{emp})\}$
$\{\exists \widehat{v}'.\ \widehat{v}' = v' \wedge v = v'' \wedge (\widehat{e}' \mapsto v)\}$
$\{v = v'' \wedge (e' \mapsto v)\}$

(where $\widehat{e}'$ denotes $e/v \to \widehat{v}'$).

—

From (LKG) to (LKBR1)

From (LKG):

$$\frac{}{\{\exists v''.\ (e \mapsto v'')\ *\ (r/v' \to v)\}\ v := [e]\ \{\exists v'.\ (e' \mapsto v)\ *\ (r/v'' \to v)\},}$$

where v, v', and v'' are distinct, v', $v'' \notin \mathsf{FV}(e)$, $v \notin \mathsf{FV}(r)$, and e' denotes $e/v \to v'$.

we derive (LKBR1):

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

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

by taking r to be $(e' \mapsto v'') \mathrel{-\!\!*} p''$ and using the axiom schema $q\ *\ (q \mathrel{-\!\!*} p) \Rightarrow p$. (In the first line, we can rename the quantified variable v'' to be any variable not in $\mathsf{FV}(e) \cup (\mathsf{FV}(p) - \{v\})$.)

$$\begin{array}{l}\{\exists v''.\ (e \mapsto v'')\ *\ ((e \mapsto v'') \mathrel{-\!\!*} p'')\}\\ v := [e]\\ \{\exists v'.\ (e' \mapsto v)\ *\ ((e' \mapsto v) \mathrel{-\!\!*} p)\}\\ \{\exists v'.\ p\}\\ \{p\}.\end{array}$$

—

From (LKBR1) to (LKBR2)

From (LKBR1):

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

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

we derive (LKBR2):

$$\frac{}{\{\exists v''.\ (e \hookrightarrow v'') \wedge p''\}\ v := [e]\ \{p\},}$$

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

by using the axiom schema

$$(e \hookrightarrow e') \wedge p \Rightarrow (e \mapsto e')\ *\ ((e \mapsto e') \mathrel{-\!\!*} p).$$

to obtain

$$\begin{array}{l}\{\exists v''.\ (e \hookrightarrow v'') \wedge p''\}\\ \{\exists v''.\ (e \mapsto v'')\ *\ ((e \mapsto v'') \mathrel{-\!\!*} p'')\}\\ v := [e]\\ \{p\}.\end{array}$$

—

From (LKBR2) to (LKL)

From (LKBR2):

$$\frac{}{\{\exists v''.\ (e \hookrightarrow v'') \wedge p''\}\ v := [e]\ \{p\},}$$

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

we derive (LKL):

$$\frac{}{\{v = v' \wedge (e \mapsto v'')\}\ v := [e]\ \{v = v'' \wedge (e' \mapsto v)\},}$$

where v, v', and v'' are distinct, and e' denotes $e/v \to v'$.

by renaming v'' to $\widehat{v}$ in the precondition of (LKBR2), taking p to be $v = v'' \wedge (e' \mapsto v)$, and using properties of $\hookrightarrow$, equality, and the existential quantifier:

$$\begin{array}{l}
\{v = v' \wedge (e \mapsto v'')\} \\
\{v = v' \wedge (e \hookrightarrow v'') \wedge (e' \mapsto v'')\} \\
\{(e \hookrightarrow v'') \wedge (e' \mapsto v'')\} \\
\{\exists \widehat{v}.\ (e \hookrightarrow v'') \wedge \widehat{v} = v'' \wedge (e' \mapsto v'')\} \\
\{\exists \widehat{v}.\ (e \hookrightarrow \widehat{v}) \wedge \widehat{v} = v'' \wedge (e' \mapsto \widehat{v})\} \\
v := [e] \\
\{v = v'' \wedge (e' \mapsto v)\}.
\end{array}$$

—

From (LKG) to (LKNOG)

From (LKG):

$$\frac{}{\{\exists v''.\ (e \mapsto v'')\ *\ (r/v' \to v)\}\ v := [e]\ \{\exists v'.\ (e' \mapsto v)\ *\ (r/v'' \to v)\},}$$

where v, v', and v'' are distinct, v', $v'' \notin \mathsf{FV}(e)$, $v \notin \mathsf{FV}(r)$, and e' denotes $e/v \to v'$.

we derive (LKNOG):

$$\frac{}{\{\exists v''.\ (e \mapsto v'')\ *\ p''\}\ v := [e]\ \{(e \mapsto v)\ *\ p\},}$$

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

Suppose v and v'' are distinct variables, $v \notin \mathsf{FV}(e)$, and $v'' \notin \mathsf{FV}(e) \cup \mathsf{FV}(p)$. We take v' to be a variable distinct from v and v'' that does not occur free in e or p, and r to be $p'' = p/v \to v''$. (Note that $v \notin \mathsf{FV}(e)$ implies that $e' = e$.) Then

$$\begin{array}{l}
\{\exists v''.\ (e \mapsto v'')\ *\ p''\} \\
\{\exists v''.\ (e \mapsto v'')\ *\ (p''/v' \to v)\} \\
v := [e] \\
\{\exists v'.\ (e' \mapsto v)\ *\ (p''/v'' \to v)\} \\
\{\exists v'.\ (e \mapsto v)\ *\ p)\} \\
\{(e \mapsto v)\ *\ p)\}.
\end{array}$$

(In the first line, we can rename v'' to be v.)

—

Annotated Specifications for the Heap Commands

- Mutation (MUBRan)

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

- Disposal (DISBRan)

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

- Allocation (CONSBRan)

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

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

- Lookup (LKBR1an)

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

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 (MUBRan) to be $e \mapsto e'$, and using the valid verification condition

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

we may use (SPan) to obtain a proof

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

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

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

—

A Final Example

$\{\mathbf{emp}\}$

$\mathsf{x} := \mathbf{cons}(\mathsf{a}, \mathsf{a})$; (CONSNOL)

$\{\mathsf{x} \mapsto \mathsf{a}, \mathsf{a}\}$ i.e., $\{\mathsf{x} \mapsto \mathsf{a} * \mathsf{x} + 1 \mapsto \mathsf{a}\}$

$\mathsf{y} := \mathbf{cons}(\mathsf{b}, \mathsf{b})$; (CONSNOG)

$\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{a}) * (\mathsf{y} \mapsto \mathsf{b}, \mathsf{b})\}$
i.e., $\{\mathsf{x} \mapsto \mathsf{a} * \mathsf{x} + 1 \mapsto \mathsf{a} * \mathsf{y} \mapsto \mathsf{b} * \mathsf{y} + 1 \mapsto \mathsf{b}\}$

$\{(\mathsf{x} \mapsto \mathsf{a}, -) * (\mathsf{y} \mapsto \mathsf{b}, \mathsf{b})\}$ $(p/v \rightarrow e \Rightarrow \exists v.\ p)$
i.e., $\{\mathsf{x} \mapsto \mathsf{a} * (\exists \mathsf{a}.\ \mathsf{x} + 1 \mapsto \mathsf{a}) * \mathsf{y} \mapsto \mathsf{b} * \mathsf{y} + 1 \mapsto \mathsf{b}\}$

$[\mathsf{x} + 1] := \mathsf{y} - \mathsf{x}$; (MUG)

$\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{y} - \mathsf{x}) * (\mathsf{y} \mapsto \mathsf{b}, \mathsf{b})\}$
i.e., $\{\mathsf{x} \mapsto \mathsf{a} * \mathsf{x} + 1 \mapsto \mathsf{y} - \mathsf{x} * \mathsf{y} \mapsto \mathsf{b} * \mathsf{y} + 1 \mapsto \mathsf{b}\}$

$\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{y} - \mathsf{x}) * (\mathsf{y} \mapsto \mathsf{b}, -)\}$ $(p/v \rightarrow e \Rightarrow \exists v.\ p)$
i.e., $\{\mathsf{x} \mapsto \mathsf{a} * \mathsf{x} + 1 \mapsto \mathsf{y} - \mathsf{x} * \mathsf{y} \mapsto \mathsf{b} * (\exists \mathsf{b}.\ \mathsf{y} + 1 \mapsto \mathsf{b})\}$

$[\mathsf{y} + 1] := \mathsf{x} - \mathsf{y}$; (MUG)

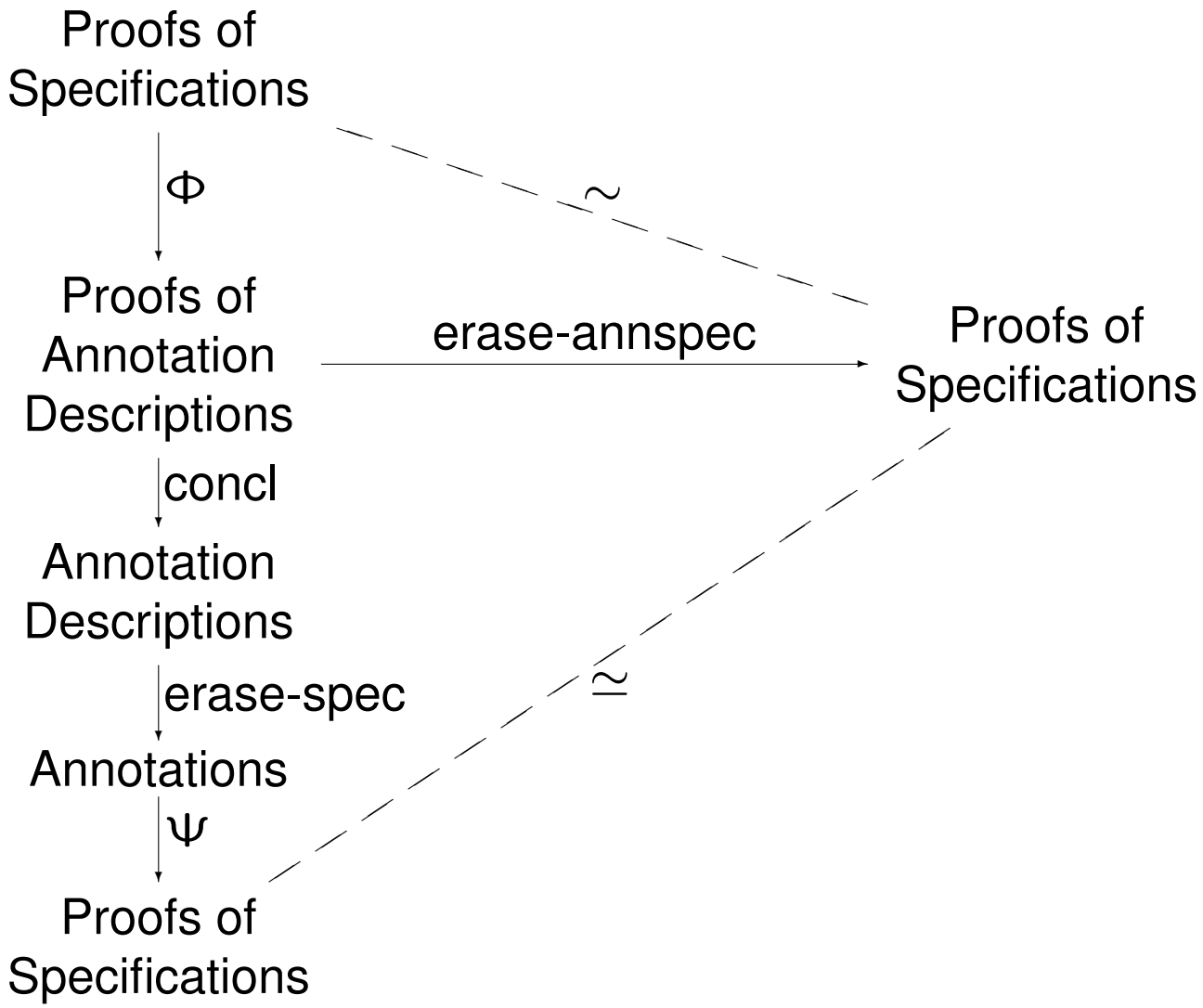
$\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{y} - \mathsf{x}) * (\mathsf{y} \mapsto \mathsf{b}, \mathsf{x} - \mathsf{y})\}$
i.e., $\{\mathsf{x} \mapsto \mathsf{a} * \mathsf{x} + 1 \mapsto \mathsf{y} - \mathsf{x} * \mathsf{y} \mapsto \mathsf{b} * \mathsf{y} + 1 \mapsto \mathsf{x} - \mathsf{y}\}$

$\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{y} - \mathsf{x}) * (\mathsf{y} \mapsto \mathsf{b}, -(\mathsf{y} - \mathsf{x}))\}$ $(x - y = -(y - x))$
i.e., $\{\mathsf{x} \mapsto \mathsf{a} * \mathsf{x} + 1 \mapsto \mathsf{y} - \mathsf{x} * \mathsf{y} \mapsto \mathsf{b} * \mathsf{y} + 1 \mapsto -(\mathsf{y} - \mathsf{x})\}$

$\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{y} - \mathsf{x}) * (\mathsf{x} + (\mathsf{y} - \mathsf{x}) \mapsto \mathsf{b}, -(\mathsf{y} - \mathsf{x}))\}$
$(y = x + (y - x))$
i.e., $\{\mathsf{x} \mapsto \mathsf{a} * \mathsf{x} + 1 \mapsto \mathsf{y} - \mathsf{x}$
$* \mathsf{x} + (\mathsf{y} - \mathsf{x}) \mapsto \mathsf{b} * \mathsf{x} + (\mathsf{y} - \mathsf{x}) + 1 \mapsto -(\mathsf{y} - \mathsf{x})\}$

$\{\exists \mathsf{o}.\ (\mathsf{x} \mapsto \mathsf{a}, \mathsf{o}) * (\mathsf{x} + \mathsf{o} \mapsto \mathsf{b}, -\mathsf{o})\}$ $(p/v \rightarrow e \Rightarrow \exists v.\ p)$

Why Annotated Specifications Work



Exercise 1

Fill in the postconditions in

$$\{(e_1 \mapsto -) * (e_2 \mapsto -)\}\ [e_1] := e_1' ; [e_2] := e_2'\ \{?\}$$

$$\{(e_1 \mapsto -) \wedge (e_2 \mapsto -)\}\ [e_1] := e_1' ; [e_2] := e_2'\ \{?\}.$$

to give two sound inference rules describing a sequence of two mutations. Your postconditions should be as strong as possible.

Give a derivation of each of these inference rules, exhibited as an annotated specification.

—

Exercise 2

The alternative inference rule for conditional commands (CDalt), leads to the following rule for annotated specifications:

- Alternative Rule for Conditionals (CDaltan)

$$\frac{\mathcal{A}_1\,\{q\} \gg \{p_1\}\; c_1\; \{q\} \qquad \mathcal{A}_2\,\{q\} \gg \{p_2\}\; c_2\; \{q\}}{(\mathbf{if}\ b\ \mathbf{then}\ \mathcal{A}_1\ \mathbf{else}\ \mathcal{A}_2)\,\{q\} \gg \{(b \Rightarrow p_1) \wedge (\neg\, b \Rightarrow p_2)\}\ (\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2)\ \{q\},}$$

Examine the annotated specifications in this and the following chapters, and determine how they would need to be changed if (CDan) were replaced by (CDaltan).

—

Exercise 3

The following are alternative global rules for allocation and lookup that use unmodified variables (v' and v''):

- The unmodified-variable global form for allocation (CONSGG)

$$\frac{}{\{v = v' \wedge r\}\ v := \mathbf{cons}(\overline{e})\ \{(v \mapsto \overline{e}')\ *\ r'\}},$$

  where v' is distinct from v, $\overline{e}'$ denotes $\overline{e}/v \rightarrow v'$, and r' denotes $r/v \rightarrow v'$.

- The unmodified-variable global form for lookup (LKGG)

$$\frac{}{\{v = v' \wedge ((e \mapsto v'')\ *\ r)\}\ v := [e]\ \{v = v'' \wedge ((e' \mapsto v)\ *\ r)\}},$$

  where v, v', and v'' are distinct, $v \notin \mathsf{FV}(r)$, and e' denotes $e/v \rightarrow v'$.

Derive (CONSGG) from (CONSG), and (CONSL) from (CONSGG). Derive (LKGG) from (LKG), and (LKL) from (LKGG).

—

Exercise 4

Derive (LKNOL) from (LKNOG) and vice-versa.

Hints:

- To derive (LKNOL) from (LKNOG), use the version of (LKNOG) where $v'' = v$.
- To derive (LKNOG) from (LKNOL), assume v and v'' are distinct, and then apply renaming of v'' in the precondition to cover the case where $v = v''$.

—

Exercise 5

Use the following equivalence of meaning between two lookup commands:

$$v := [e] \cong \mathbf{newvar}\ \widehat{v}\ \mathbf{in}\ (\widehat{v} := [e]\ ;\ v := \widehat{v})$$

to derive (LKG) from (LKNOG).

—

Exercise 6

Suppose that the *total* correctness specification [p] c [q] holds, and the assertion q is precise. Show, by an informal metaproof, that p is precise.

Hint: Use safety monotonicity, the frame property, and the definition of [p] c [q].

—