Class Notes for CS 819A3 - Spring 2003

2. Specifications and Their Inference Rules

John C. Reynolds
Department of Computer Science
Carnegie Mellon University

Revised January 28, 2003

Specifications (Hoare Triples)

$$\begin{aligned} \langle \text{spec} \rangle ::= {} & \{\langle \text{assert} \rangle\}\ \langle \text{comm} \rangle\ \{\langle \text{assert} \rangle\} && \text{(partial correctness)} \\ \mid {} & [\ \langle \text{assert} \rangle\ ]\ \langle \text{comm} \rangle\ [\ \langle \text{assert} \rangle\ ] && \text{(total correctness)} \end{aligned}$$

$$[\![-]\!]_{\text{spec}} \in \langle \text{spec} \rangle \to \mathbf{B}.$$

When $[\![sp]\!]_{\text{spec}}$ is true we say that *sp holds* or is *valid*.

Partial correctness:

$$\begin{aligned} & \{p\}\ c\ \{q\} \text{ holds iff } \forall s \in \text{States}.\ s \models p \text{ implies} \\ & \qquad \forall s' \in \text{States}.\ c, s \rightsquigarrow^* s' \text{ implies } s' \models q \end{aligned}$$

Total correctness:

$$\begin{aligned} & [\ p\ ]\ c\ [\ q\ ] \text{ holds iff } \forall s \in \text{States}.\ s \models p \text{ implies} \\ & \qquad \neg\,(c, s \uparrow) \\ & \qquad \text{and } \forall s' \in \text{States}.\ c, s \rightsquigarrow^* s' \text{ implies } s' \models q \end{aligned}$$

Examples of Partial Correctness Specifications

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

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

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

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

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

Examples of Total Correctness Specifications

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

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

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

Inference Rules for Specifications:

Assignment (AS)

$$\frac{}{\{p/v \to e\}\ v := e\ \{p\}} \qquad \frac{}{[\,p/v \to e\,]\ v := e\ [\,p\,]}$$

Instances

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

$$\frac{}{\{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\ \{\mathsf{x} - \mathsf{y} > 3\}}$$

Inference Rules for Specifications:

Sequential Composition (SQ)

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

An Instance

$$\frac{\begin{array}{r}\{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\ \{\mathsf{x} - \mathsf{y} > 3\} \qquad\qquad \\ \{\mathsf{x} - \mathsf{y} > 3\}\ \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}\end{array}}{\{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\,;\,\mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}}$$

Note

Except for **while** commands and recursive procedures, reasoning for partial and total correctness is the same. Henceforth, we will give only rules for partial correctness, except when the rule for total correctness is different.

Inference Rules for Specifications:

Strengthening Precedent (SP)

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

An Instance

$$\frac{\begin{array}{l}\mathsf{y} > 3 \Rightarrow (2 \times \mathsf{y}) - \mathsf{y} > 3 \\ \qquad\qquad \{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\; \mathsf{x} := 2 \times \mathsf{y}\,;\, \mathsf{x} := \mathsf{x} - \mathsf{y}\; \{\mathsf{x} > 3\}\end{array}}{\{\mathsf{y} > 3\}\; \mathsf{x} := 2 \times \mathsf{y}\,;\, \mathsf{x} := \mathsf{x} - \mathsf{y}\; \{\mathsf{x} > 3\}}$$

Weakening Consequent (WC)

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

An Instance

$$\frac{\{\mathsf{y} > 3\}\; \mathsf{x} := 2 \times \mathsf{y}\,;\, \mathsf{x} := \mathsf{x} - \mathsf{y}\; \{\mathsf{x} > 3\} \qquad \mathsf{x} > 3 \Rightarrow \mathsf{x} \geq 4}{\{\mathsf{y} > 3\}\; \mathsf{x} := 2 \times \mathsf{y}\,;\, \mathsf{x} := \mathsf{x} - \mathsf{y}\; \{\mathsf{x} \geq 4\}}$$

Since they are applicable to arbitrary commands, the rules (SP) and (WC) are called *structural rules.* One premiss of each of these 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.

A Simple Proof

1. $\{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\ \{\mathsf{x} - \mathsf{y} > 3\}$ (AS)
2. $\{\mathsf{x} - \mathsf{y} > 3\}\ \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}$ (AS)
3. $\{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\,;\, \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}$ (SQ,1,2)
4. $\mathsf{y} > 3 \Rightarrow (2 \times \mathsf{y}) - \mathsf{y} > 3$ (VC)
5. $\{\mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\,;\, \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}$ (SP,4,3)
6. $\mathsf{x} > 3 \Rightarrow \mathsf{x} \geq 4$ (VC)
7. $\{\mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\,;\, \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} \geq 4\}$ (WC,5,6)

As long as only simple facts about integer arithmetic are involved, we will allow *valid* verification conditions to appear in proofs without giving subproofs for the conditions.

A More Realistic Proof

1. $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{g} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{k} := \mathsf{k}+1$
   $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\}$ (AS)

2. $\{\mathsf{f}+\mathsf{t} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{g} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{f} := \mathsf{f}+\mathsf{t}$
   $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{g} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$ (AS)

3. $\{\mathsf{f}+\mathsf{t} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{g} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{f} := \mathsf{f}+\mathsf{t}\,;\mathsf{k} := \mathsf{k}+1$
   $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\}$ (SQ,1,2)

4. $\{\mathsf{f}+\mathsf{t} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{f} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{g} := \mathsf{f}$
   $\{\mathsf{f}+\mathsf{t} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{g} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$ (AS)

5. $\{\mathsf{f}+\mathsf{t} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{f} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{g} := \mathsf{f}\,;\mathsf{f} := \mathsf{f}+\mathsf{t}\,;\mathsf{k} := \mathsf{k}+1$
   $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\}$ (SQ,3,4)

A More Realistic Proof (continued)

6. $\{\mathsf{f} + \mathsf{g} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{f} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{t} := \mathsf{g}$
   $\{\mathsf{f} + \mathsf{t} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{f} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$ (AS)

7. $\{\mathsf{f} + \mathsf{g} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{f} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{t} := \mathsf{g}\,;\, \mathsf{g} := \mathsf{f}\,;\, \mathsf{f} := \mathsf{f} + \mathsf{t}\,;\, \mathsf{k} := \mathsf{k}+1$
   $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\}$ (SQ,5,6)

8. $(\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}) \Rightarrow$
   $(\mathsf{f} + \mathsf{g} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{f} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n})$
   (VC*)

9. $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}\}$
   $\mathsf{t} := \mathsf{g}\,;\, \mathsf{g} := \mathsf{f}\,;\, \mathsf{f} := \mathsf{f} + \mathsf{t}\,;\, \mathsf{k} := \mathsf{k}+1$
   $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\}$ (SP,7,8)

*since

$$\mathrm{fib}(0) = 0 \qquad \mathrm{fib}(1) = 1 \qquad \mathrm{fib}(i) = \mathrm{fib}(i-1) + \mathrm{fib}(i-2).$$

Inference Rules for Specifications:

An Alternative "Forward" Rule for Assignment (ASALT)

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

where $v' \notin \{v\} \cup \mathrm{FV}(e) \cup \mathrm{FV}(p)$, 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.

The problem with this rule is the accumulation of quantifiers. For example, when we use it to prove

$$\begin{aligned}&\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}\}\\ &\mathsf{t} := \mathsf{g}\ ;\ \mathsf{g} := \mathsf{f}\ ;\ \mathsf{f} := \mathsf{f} + \mathsf{t}\ ;\ \mathsf{k} := \mathsf{k} + 1\\ &\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\},\end{aligned}$$

we get the verification condition

$$\begin{aligned}&(\exists \mathsf{k}'.\ \exists \mathsf{f}'.\ \exists \mathsf{g}'.\ \mathsf{k} = \mathsf{k}' + 1 \wedge \mathsf{f} = \mathsf{f}' + \mathsf{t} \wedge \mathsf{g} = \mathsf{f}' \wedge \mathsf{t} = \mathsf{g}'\\ &\qquad\qquad \wedge\, \mathsf{f}' = \mathrm{fib}(\mathsf{k}') \wedge \mathsf{g}' = \mathrm{fib}(\mathsf{k}'-1) \wedge \mathsf{k}' \leq \mathsf{n} \wedge \mathsf{k}' \neq \mathsf{n}) \Rightarrow\\ &\quad\ \mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}.\end{aligned}$$

Inference Rules for Specifications:

Partial Correctness of **while** (WHP)

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

Here i is the *invariant*.

An Instance

$$\frac{\begin{array}{l}\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \le \mathsf{n} \wedge \mathsf{k} \ne \mathsf{n}\} \\ \mathsf{t} := \mathsf{g}\,;\, \mathsf{g} := \mathsf{f}\,;\, \mathsf{f} := \mathsf{f} + \mathsf{t}\,;\, \mathsf{k} := \mathsf{k} + 1 \\ \{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \le \mathsf{n}\}\end{array}}{\begin{array}{l}\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \le \mathsf{n}\} \\ \textbf{while}\; \mathsf{k} \ne \mathsf{n}\; \textbf{do}\; (\mathsf{t} := \mathsf{g}\,;\, \mathsf{g} := \mathsf{f}\,;\, \mathsf{f} := \mathsf{f} + \mathsf{t}\,;\, \mathsf{k} := \mathsf{k} + 1) \\ \{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \le \mathsf{n} \wedge \neg\, \mathsf{k} \ne \mathsf{n}\}\end{array}}$$

Inference Rules for Specifications:

Total Correctness of **while** (WHT)

$$\frac{[\, i \wedge b \wedge e = v_0 \,]\; c \;[\, i \wedge e < v_0 \,] \qquad (i \wedge b) \Rightarrow e \geq 0}{[\, i \,]\; \textbf{while}\; b\; \textbf{do}\; c\; [\, i \wedge \neg\, b \,]}$$

when v_0 does not occur free in i, b, c, or e.
Here i is the *invariant*, e is the *variant*, and v_0 is a *ghost variable*.

An Instance

$$\frac{\begin{array}{l}[\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n} \wedge \mathsf{n} - \mathsf{k} = \mathsf{v}_0] \\ \mathsf{t} := \mathsf{g}\,;\, \mathsf{g} := \mathsf{f}\,;\, \mathsf{f} := \mathsf{f} + \mathsf{t}\,;\, \mathsf{k} := \mathsf{k} + 1 \\ [\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{n} - \mathsf{k} < \mathsf{v}_0] \\ \\ (\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}) \Rightarrow \mathsf{n} - \mathsf{k} \geq 0\end{array}}{\begin{array}{l}[\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}] \\ \textbf{while}\; \mathsf{k} \neq \mathsf{n}\; \textbf{do}\; (\mathsf{t} := \mathsf{g}\,;\, \mathsf{g} := \mathsf{f}\,;\, \mathsf{f} := \mathsf{f} + \mathsf{t}\,;\, \mathsf{k} := \mathsf{k} + 1) \\ [\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \neg\, \mathsf{k} \neq \mathsf{n}]\end{array}}$$

Other Well-Ordered Sets

Sometimes, the range of variants must be a larger set than the set of natural numbers. For example, the set $\{\, n \mid n \geq 0 \,\} \cup \{\infty\}$ is used in

$$
\begin{array}{l}
[\mathbf{true}] \\
\mathbf{while}\ [\mathbf{vrnt}\colon \mathbf{if}\ \mathsf{x} < 0\ \mathbf{then}\ \infty\ \mathbf{else}\ \mathsf{x}]\ \mathsf{x} \neq 0\ \mathbf{do} \\
\quad \mathbf{if}\ \mathsf{x} < 0\ \mathbf{then}\ (\mathbf{newvar}\ \mathsf{y}\ \mathbf{in}\ \mathsf{x} := |\mathsf{y}|)\ \mathbf{else}\ \mathsf{x} := \mathsf{x} - 1 \\
[\mathsf{x} = 0]
\end{array}
$$

This program contains an infinitary choice that always terminates:

$$\mathbf{newvar}\ \mathsf{y}\ \mathbf{in}\ \mathsf{x} := |\mathsf{y}|.$$

This is called *unbounded nondeterminism.*

Inference Rules for Specifications:

skip (SK)

$$\frac{}{\{p\}\ \mathbf{skip}\ \{p\}}$$

Conditional (CD)

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

An Alternative Rule for Conditionals (CDALT)

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

Inference Rules for Specifications:

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.

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.

Locality is a property of the specification of a command rather than just the command itself, since it depends on the use to which the command may be put. For example, the same command satisfies both of the specifications

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

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

But the variable t is local only in the first case, so that **newvar** t **in** $(\mathsf{t} := \mathsf{x} + \mathsf{y}\ ;\ \mathsf{z} := \mathsf{t} \times \mathsf{t})$ meets only the first specification:

$$\{\mathbf{true}\}\ \mathbf{newvar}\ \mathsf{t}\ \mathbf{in}\ (\mathsf{t} := \mathsf{x} + \mathsf{y}\ ;\ \mathsf{z} := \mathsf{t} \times \mathsf{t})\ \{\mathsf{z} = (\mathsf{x} + \mathsf{y})^2\}.$$

Annotated Specifications

In an *annotated specification*, additional assertions called *annotations* are placed within the command in such a way that the proof of the specification can be reconstructed from the annotations. To determine this reconstruction, we must use rules in which the premises are completely determined by the conclusion. This requires us to change the following rules:

Sequential Composition (SQAN)

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

Strengthening Precedent (SPAN)

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

Weakening Consequent (WCAN)

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

Total Correctness of **while**

$$\frac{[\, i \wedge b \wedge e = v_0 \,]\ c\ [\, i \wedge e < v_0 \,] \qquad (i \wedge b) \Rightarrow e \geq 0}{[\, i \,]\ \mathbf{while}\ [\mathbf{vrnt}\colon e]\ b\ \mathbf{do}\ c\ [\, i \wedge \neg\, b \,]}$$

when v_0 does not occur free in i, b, c, or e.

Annotated Specifications (continued)

To reduce the number of unnecessary annotations, it is useful to use the following derived inference rules:

Assignment (ASAN)

$$\frac{p_0 \Rightarrow (p/v \rightarrow e)}{\{p_0\}\ v := e\ \{p\}} \qquad \frac{\{p_0\}\ c\ \{p/v \rightarrow e\}}{\{p_0\}\ c\,;\, v := e\ \{p\}}$$

skip (SKAN)

$$\frac{p_0 \Rightarrow p}{\{p_0\}\ \mathbf{skip}\ \{p\}}$$

Partial Correctness of **while** (WHPAN)

$$\frac{\{i \wedge b\}\ c\ \{i\} \qquad (i \wedge \neg\, b) \Rightarrow p}{\{i\}\ \mathbf{while}\ b\ \mathbf{do}\ c\ \{p\}}$$

Total Correctness of **while** (WHTAN)

$$\frac{[\, i \wedge b \wedge e = v_0 \,]\ c\ [\, i \wedge e < v_0 \,] \quad (i \wedge b) \Rightarrow e \geq 0 \quad (i \wedge \neg\, b) \Rightarrow p}{[\, i \,]\ \mathbf{while}\ [\mathbf{vrnt}\!: e]\ b\ \mathbf{do}\ c\ [\, p \,]}$$

when v_0 does not occur free in i, b, c, or e.

An Annotated Specification for Fibonacci Numbers

$$
\begin{array}{l}
[\mathsf{n} \geq 0] \\
\textbf{if } \mathsf{n} = 0 \textbf{ then } \mathsf{f} := 0 \textbf{ else} \\
\quad \textbf{newvar } \mathsf{g} \textbf{ in newvar } \mathsf{k} \textbf{ in} \\
\quad\quad \big(\mathsf{k} := 1\,; \mathsf{g} := 0\,; \mathsf{f} := 1\,; \\
\quad\quad [\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}] \\
\quad\quad \textbf{while } [\textbf{vrnt}\!: \mathsf{n} - \mathsf{k}]\; \mathsf{k} \neq \mathsf{n} \textbf{ do} \\
\quad\quad\quad \textbf{newvar } \mathsf{t} \textbf{ in } (\mathsf{t} := \mathsf{g}\,; \mathsf{g} := \mathsf{f}\,; \mathsf{f} := \mathsf{f} + \mathsf{t}\,; \mathsf{k} := \mathsf{k} + 1)\big) \\
[\mathsf{f} = \mathrm{fib}(\mathsf{n})]
\end{array}
$$

Verification Conditions

1. $\mathsf{n} \geq 0 \wedge \mathsf{n} = 0 \Rightarrow 0 = \mathrm{fib}(\mathsf{n})$

2. $\mathsf{n} \geq 0 \wedge \neg\, \mathsf{n} = 0 \Rightarrow 1 = \mathrm{fib}(1) \wedge 0 = \mathrm{fib}(1-1) \wedge 1 \leq \mathsf{n}$

3. $\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n} \wedge \mathsf{n} - \mathsf{k} = \mathsf{v}_0 \Rightarrow$
   $\mathsf{f} + \mathsf{g} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{f} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}$
   $\wedge\, \mathsf{n} - (\mathsf{k}+1) < \mathsf{v}_0$

4. $\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n} \Rightarrow \mathsf{n} - \mathsf{k} \geq 0$

5. $\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \neg\, \mathsf{k} \neq \mathsf{n} \Rightarrow \mathsf{f} = \mathrm{fib}(\mathsf{n})$

An Annotated Specification for Fast Exponentiation

$$
\begin{array}{l}
[\mathsf{n} \geq 0] \\
\textbf{newvar}\ \mathsf{k}\ \textbf{in newvar}\ \mathsf{z}\ \textbf{in} \\
\quad \big(\mathsf{k} := \mathsf{n}\,;\, \mathsf{z} := \mathsf{x}\,;\, \mathsf{y} := 1\,; \\
\quad [\mathsf{y} \times \mathsf{z}^{\mathsf{k}} = \mathsf{x}^{\mathsf{n}} \wedge \mathsf{k} \geq 0] \\
\quad \textbf{while}\ [\textbf{vrnt}\colon \mathsf{k}]\ \mathsf{k} \neq 0\ \textbf{do} \\
\qquad (\textbf{if odd}(\mathsf{k})\ \textbf{then}\ (\mathsf{k} := \mathsf{k} - 1\,;\, \mathsf{y} := \mathsf{y} \times \mathsf{z})\ \textbf{else skip}\,; \\
\qquad \mathsf{k} := \mathsf{k} \div 2\,;\, \mathsf{z} := \mathsf{z} \times \mathsf{z})\big) \\
[\mathsf{y} = \mathsf{x}^{\mathsf{n}}]
\end{array}
$$

Verification Conditions

1. $\mathsf{n} \geq 0 \Rightarrow 1 \times \mathsf{x}^{\mathsf{n}} = \mathsf{x}^{\mathsf{n}} \wedge \mathsf{n} \geq 0$
2. $\mathsf{y} \times \mathsf{z}^{\mathsf{k}} = \mathsf{x}^{\mathsf{n}} \wedge \mathsf{k} \geq 0 \wedge \mathsf{k} \neq 0 \wedge \mathsf{k} = \mathsf{k}_0 \wedge \textbf{odd}(\mathsf{k}) \Rightarrow$
   $$(\mathsf{y} \times \mathsf{z}) \times (\mathsf{z} \times \mathsf{z})^{(\mathsf{k}-1)\div 2} = \mathsf{x}^{\mathsf{n}}$$
   $$\wedge\, (\mathsf{k} - 1) \div 2 \geq 0 \wedge (\mathsf{k} - 1) \div 2 < \mathsf{k}_0$$
3. $\mathsf{y} \times \mathsf{z}^{\mathsf{k}} = \mathsf{x}^{\mathsf{n}} \wedge \mathsf{k} \geq 0 \wedge \mathsf{k} \neq 0 \wedge \mathsf{k} = \mathsf{k}_0 \wedge \neg\, \textbf{odd}(\mathsf{k}) \Rightarrow$
   $$\mathsf{y} \times (\mathsf{z} \times \mathsf{z})^{\mathsf{k}\div 2} = \mathsf{x}^{\mathsf{n}} \wedge \mathsf{k} \div 2 \geq 0 \wedge \mathsf{k} \div 2 < \mathsf{k}_0$$
4. $\mathsf{y} \times \mathsf{z}^{\mathsf{k}} = \mathsf{x}^{\mathsf{n}} \wedge \mathsf{k} \geq 0 \wedge \mathsf{k} \neq 0 \Rightarrow \mathsf{k} \geq 0$
5. $\mathsf{y} \times \mathsf{z}^{\mathsf{k}} = \mathsf{x}^{\mathsf{n}} \wedge \mathsf{k} \geq 0 \wedge \neg\, \mathsf{k} \neq 0 \Rightarrow \mathsf{y} = \mathsf{x}^{\mathsf{n}}$

Interderivability of the Assignment Rules

$$\overline{\{p/v \to e\}\ v := e\ \{p\}}$$

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

where $v' \notin \{v\} \cup \mathrm{FV}(e) \cup \mathrm{FV}(p)$, e' is $e/v \to v'$, and p' is $p/v \to v'$.

$\{p\}$
$\{e = e \wedge p\}$
$\{(e = e' \wedge p')/v' \to v\}$
$\{\exists v'.\ e = e' \wedge p'\}$
$v := e$
$\{\exists v'.\ v = e' \wedge p'\}$

$\{p/v \to e\}$
$v := e$
$\{\exists v'.\ v = e' \wedge ((p/v \to e)/v \to v')\}$
$\{\exists v'.\ v = e' \wedge (p/v \to e')\}$
$\{\exists v'.\ v = e' \wedge p\}$
$\{\exists v'.\ p\}$
$\{p\}$

Interderivability of the Conditional Rules

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

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

Assume the premises of the second rule. Then

$\{(b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2) \wedge b\}$
$\{p_1\}$
c_1
$\{q\}$

and

$\{(b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2) \wedge \neg\ b\}$
$\{p_2\}$
c_2
$\{q\}$,

and the first rule gives the conclusion of the second rule:

$\{(b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2)\}$
$\textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2$
$\{q\}$.

Interderivability of the Conditional Rules (continued)

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

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

Assume the premises of the first rule. Then the second rule gives the main step in

$\{p\}$
$\{(b \Rightarrow p \wedge b) \wedge (\neg\ b \Rightarrow p \wedge \neg\ b)\}$
$\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2$
$\{q\}$,

which is the conclusion of the second rule.

Structural Inference Rules

(In addition to Strengthening Precedent and Weakening Consequent)

Renaming (RN)

$$\frac{\{p\}\ c\ \{q\}}{\{p'\}\ c'\ \{q'\}},$$

where p', c', and q' are obtained from p, c, and q by zero or more renamings of bound variables.

It is sometimes necessary to use renaming in combination with the rule for variable declarations, e.g.

$$\cfrac{\cfrac{\{\mathsf{x} = 0\}\ \mathsf{y} := 1\ \{\mathsf{x} = 0\}}{\{\mathsf{x} = 0\}\ \textbf{newvar}\ \mathsf{y}\ \textbf{in}\ \mathsf{y} := 1\ \{\mathsf{x} = 0\}}}{\{\mathsf{x} = 0\}\ \textbf{newvar}\ \mathsf{x}\ \textbf{in}\ \mathsf{x} := 1\ \{\mathsf{x} = 0\}.}$$

Substitution (SUB)

$$\frac{\{p\}\ c\ \{q\}}{(\{p\}\ c\ \{q\})/v_1 \to e_1,\ \ldots\ ,\ v_n \to e_n,}$$

where $v_1, \ldots, v_n$ are the variables occurring free in p, c, or q, and, if v_i is assigned 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

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

The rule for substitution will become important when we consider procedures.

Structural Inference Rules:

Disjunction (DISJ)

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

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.

Universal Quantification (UQ)

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

where v is not free in c.

Structural Inference Rules:

Constancy (CONST)

$$\frac{\{p\}\ c\ \{q\}}{\{p \wedge r\}\ c\ \{q \wedge r\}}$$

when no variable occurs both free in r and assigned by c.

Structural Rules for Annotated Specifications:

Existential Quantification (EQAN)

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

where v is not free in c.

Universal Quantification (UQAN)

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

where v is not free in c.

Constancy (CONSTAN)

$$\frac{\{p\}\ c\ \{q\}}{\{p \wedge r\}\ \mathbf{coninv}\ r\ \mathbf{in}\ c\ \{q \wedge r\}}$$

when no variable occurs both free in r and assigned by c.

Ghost Variables

$$
\begin{array}{l}
[\mathsf{k} \geq 0 \wedge \mathsf{k} = \mathsf{k}_0] \\
\textbf{newvar } \mathsf{z} \textbf{ in} \\
\quad \big(\mathsf{z} := \mathsf{x} \,;\, \mathsf{y} := 1 \,; \\
\quad [\mathsf{y} \times \mathsf{z}^{\mathsf{k}} = \mathsf{x}^{\mathsf{k}_0} \wedge \mathsf{k} \geq 0] \\
\quad \textbf{while } [\textbf{vrnt}\!: \mathsf{k}]\; \mathsf{k} \neq 0 \textbf{ do} \\
\qquad (\textbf{if odd}(\mathsf{k}) \textbf{ then } (\mathsf{k} := \mathsf{k} - 1 \,;\, \mathsf{y} := \mathsf{y} \times \mathsf{z}) \textbf{ else skip} \,; \\
\qquad \mathsf{k} := \mathsf{k} \div 2 \,;\, \mathsf{z} := \mathsf{z} \times \mathsf{z})\big) \\
[\mathsf{y} = \mathsf{x}^{\mathsf{k}_0}]
\end{array}
$$

Varied Arithmetic

In general, integer division satisfies

$$\mathsf{y} \neq 0 \;\Rightarrow\; \mathsf{x} = (\mathsf{x} \div \mathsf{y}) \times \mathsf{y} + \mathsf{x}\ \mathbf{rem}\ \mathsf{y}$$

$$\mathsf{x} \geq 0 \wedge \mathsf{y} > 0 \;\Rightarrow\; 0 \leq \mathsf{x}\ \mathbf{rem}\ \mathsf{y} < \mathsf{y},$$

However, although most machines provide division that is "odd" in x and y:

$$\mathsf{y} \neq 0 \;\Rightarrow\; (-\mathsf{x}) \div \mathsf{y} = -(\mathsf{x} \div \mathsf{y})$$

$$\mathsf{y} \neq 0 \;\Rightarrow\; \mathsf{x} \div (-\mathsf{y}) = -(\mathsf{x} \div \mathsf{y}),$$

a few machines provide a "number-theoretic" division satisfying

$$\mathsf{y} > 0 \Rightarrow 0 \leq \mathsf{x}\ \mathbf{rem}\ \mathsf{y} < \mathsf{y},$$

so that, for example, $-3 \div 2 = -2$ and $-3\ \mathbf{rem}\ 2 = 1$.

One can add a monotonicity law, since it holds in both situations:

$$\mathsf{y} > 0 \wedge \mathsf{x} \leq \mathsf{x}' \Rightarrow \mathsf{x} \div \mathsf{y} \leq \mathsf{x}' \div \mathsf{y}.$$

Totality is built into Predicate Calculus

$$[\,\mathbf{true}\,]\ \mathsf{y} := \mathsf{x} \div 0\,;\, \mathsf{y} := 7\ [\,\mathsf{y} = 7\,]$$

is sound since the second assignment will overwrite whatever value is possessed by $\mathsf{x} \div 0$, and

$$[\,\mathsf{x} = 7\,]\ \mathbf{if}\ \mathsf{x}{\div}0 = 29\ \mathbf{then}\ \mathsf{x}{:=}\mathsf{x}{+}1\ \mathbf{else}\ \mathsf{x}{:=}\mathsf{x}{-}1\ [\,\mathsf{x} = 8 \vee \mathsf{x} = 6\,]$$

is sound since $\mathsf{x} \div 0 = 29$ must yield true or false.

Moreover, as long as equality is axiomatized in the standard way, since the logic incorporates the assumption that the same operation applied to the same operands yields the same result, one can prove $\mathsf{x} \div 0 = \mathsf{x} \div 0$.