CS 512, Spring 2018, Handout 17

Hoare Logic (Continued)

Assaf Kfoury

15 March 2018

examples using Hoare triples

- write a program P satisfying the specification

  $$\{\, x > 0 \,\}\; P\; \{\, y \cdot y < x \,\}$$

- let P be the program "$y := 0$"

  (not very interesting!!)

- let P be the program

  $$\begin{array}{ll} y := 0; & \\ \textbf{while} & (y * y < x)\ \textbf{do} \\ & y := y + 1; \\ & \textbf{od}; \\ y := y - 1 & \end{array}$$

  (y is the largest integer whose square is less than x)

examples using Hoare triples

- write a program P satisfying the specification

$$\{ x > 0 \} \; P \; \{ \; y^2 < x \; \wedge \; (y+1)^2 \geqslant x \; \}$$

- how about a program P satisfying the following specification?

$$\{ x > 0 \} \; P \; \{ \; x - 2y - 1 \leqslant y^2 < x \; \}$$

- how about a program P satisfying the following specification?

$$\{ x > 0 \} \; P \; \{ \; y^2 < x \; \wedge \; \forall z \, (\neg(z = 0) \rightarrow (y+z)^2 \geqslant x \; \}$$

program P (on preceding page) satisfies 3 specifications above

examples using Hoare triples

- how about a program P satisfying the following specification?

$$\{\, x > 0 \,\}\; P \;\{\; y^2 < x \;\wedge\; (y+2)^2 \geqslant x \;\}$$

a different program P has to be written for this Hoare specification

examples using Hoare triples

- should the following Hoare triple hold?

  $$\{\, x+1 = 43 \,\} \quad y := x+1 \quad \{\, y = 43 \,\}$$

  Yes!

- should the following Hoare triple hold?

  $$\{\, x+1 \leqslant n \,\} \quad x := x+1 \quad \{\, x \leqslant n \,\}$$

  Yes!

partial correctness vs. total correctness

- partial correctness, [LCS, Definition 4.5, p 265], denoted:

  $$\models_{\mathsf{par}} \{\, \varphi \,\}\ P\ \{\, \psi \,\}$$

- total correctness, [LCS, Definition 4.6, p 266], denoted:

  $$\models_{\mathsf{tot}} \{\, \varphi \,\}\ P\ \{\, \psi \,\}$$

- we write $\models_{\mathsf{par}} \{\, \varphi \,\}\ P\ \{\, \psi \,\}$ instead of $\mathcal{M}, \ell \models_{\mathsf{par}} \{\, \varphi \,\}\ P\ \{\, \psi \,\}$, keeping $\mathcal{M}$ (always the same!) and ℓ implicit

  similarly for $\models_{\mathsf{tot}} \{\, \varphi \,\}\ P\ \{\, \psi \,\}$ instead of $\mathcal{M}, \ell \models_{\mathsf{tot}} \{\, \varphi \,\}\ P\ \{\, \psi \,\}$

examples of PCA's (partial correctness assertions) and TCS's (total correctness assertions)

- consider program Fac1(x)
  which computes the factorial of x and stores the result in y
  (from Handout 16, also [LCS, Example 4.2, page 262])

- $\models_{\text{par}} \{\, x \geqslant 0 \,\}$ Fac1 $\{\, y = x! \,\}$

- $\models_{\text{tot}} \{\, x \geqslant 0 \,\}$ Fac1 $\{\, y = x! \,\}$

- $\models_{\text{par}} \{\, \top \,\}$ Fac1 $\{\, y = x! \,\}$ ("$\top$" is the same as "**`true`**")

- $\not\models_{\text{tot}} \{\, \top \,\}$ Fac1 $\{\, y = x! \,\}$

proof rules for PCA's

$$\frac{\{\,\varphi\,\}\ C_1\ \{\,\theta\,\} \qquad \{\,\theta\,\}\ C_2\ \{\,\psi\,\}}{\{\,\varphi\,\}\ C_1; C_2\ \{\,\psi\,\}}$$

composition

$$\frac{}{\{\,\psi[E/x]\,\}\ x := E\ \{\,\psi\,\}}$$

assignment

$$\frac{\{\,\varphi \wedge B\,\}\ C_1\ \{\,\psi\,\} \qquad \{\,\varphi \wedge \neg B\,\}\ C_2\ \{\,\psi\,\}}{\{\,\varphi\,\}\ \textbf{if}\ B\ \textbf{then}\ C_1\ \textbf{else}\ C_2\ \{\,\psi\,\}}$$

if-statement

proof rules for PCA's

$$\frac{\{\,\psi \wedge B\,\}\ C\ \{\,\psi\,\}}{\{\,\psi\,\}\ \textbf{while}\ B\ \textbf{do}\ C\ \textbf{od}\ \{\,\psi \wedge \neg B\,\}} \quad \text{partial-while}$$

$$\frac{\vdash_{\mathsf{AR}} \varphi' \to \varphi \qquad \{\,\varphi\,\}\ C\ \{\,\psi\,\} \qquad \vdash_{\mathsf{AR}} \psi \to \psi'}{\{\,\varphi'\,\}\ C\ \{\,\psi'\,\}} \quad \text{implied}$$

("AR" stands for "arithmetic")

proof rules for TCA's

- rules "composition", "assignment", "if-statement", and "implied" are used again unchanged

- rule "partial-while" needs to be adapted into new rule "total-while"

$$\frac{\{\, \psi \wedge B \ \wedge (0 \leqslant E = z) \,\}\ C\ \{\, \psi \ \wedge (0 \leqslant E < z) \,\}}{\{\, \psi \ \wedge (0 \leqslant E) \,\}\ \textbf{while}\ B\ \textbf{do}\ C\ \textbf{od}\ \{\, \psi \wedge \neg B \,\}} \quad \text{total-while}$$

where z is a logical variable (not appearing anywhere in B and C

an imperative language + nondeterminism + concurrency

- integer expressions

$$E \quad ::= \quad \ldots \quad \text{(as before)}$$

- boolean expressions

$$B \quad ::= \quad \ldots \quad \text{(as before)}$$

- program expressions (or commands)

$$\begin{array}{rcll} C & ::= & x := E \mid C; C \mid \textbf{if } B \textbf{ then } C \textbf{ else } C \mid \textbf{while } B \textbf{ do } C \textbf{ od} \\ & \mid & C \cup C & \text{(nondeterminism)} \\ & \mid & C \parallel C & \text{(concurrency)} \end{array}$$

- execution of program $(x := 1) \cup (x := 2)$ nondeterministically sets x either to 1 or to 2
- execution of program $(x := 1; x := x + 1) \parallel (x := 2; x := x + 2)$ interleaves the 4 assignments in any order, as long as x is set to 1 before being incremented by 1, and set to 2 before being incremented by 2. possible final values of x are 2, 4, and 5.

(THIS PAGE INTENTIONALLY LEFT BLANK)