CS 512, Spring 2018, Handout 18

Hoare Logic (Continued)

Assaf Kfoury

March 20, 2018

using proof rules for PCA's

- show that $\vdash_{\text{par}} \{ y = 5 \}\, x := y + 1 \,\{ x = 6 \}$

using proof rules for PCA's

- show that $\vdash_{\text{par}} \{\, y = 5 \,\}\, x := y + 1 \,\{\, x = 6 \,\}$

$x := y + 1$

using proof rules for PCA's

- show that $\vdash_{\mathsf{par}} \{ y = 5 \}\, x := y + 1\, \{ x = 6 \}$

$$x := y + 1$$

$$\{x = 6\}$$

using proof rules for PCA's

- show that $\vdash_{\text{par}} \{ y = 5 \}\ x := y + 1\ \{ x = 6 \}$

$\{y + 1 = 6\}$ (assignment)

$x := y + 1$

$\{x = 6\}$

using proof rules for PCA's

- show that $\vdash_{\text{par}} \{ y = 5 \}\ x := y + 1\ \{ x = 6 \}$

$\{y = 5\}$ (implied)

$\{y + 1 = 6\}$ (assignment)

$x := y + 1$

$\{x = 6\}$

using proof rules for PCA's (continued)

- show that $\vdash_{\text{par}} \{ y < 3 \}\ y := y + 1\ \{ y < 4 \}$

using proof rules for PCA's (continued)

- show that $\vdash_{\text{par}} \{ y < 3 \} \; y := y + 1 \; \{ y < 4 \}$

$$y := y + 1$$

using proof rules for PCA's (continued)

- show that $\vdash_{\text{par}} \{\, y < 3 \,\}\; y := y + 1 \;\{\, y < 4 \,\}$

$y := y + 1$

$\{y < 4\}$

using proof rules for PCA's (continued)

- show that $\vdash_{\text{par}} \{ y < 3 \}\ y := y + 1\ \{ y < 4 \}$

$\{y + 1 < 4\}$ (assignment)

$y := y + 1$

$\{y < 4\}$

using proof rules for PCA's (continued)

- show that $\vdash_{\text{par}} \{ y < 3 \}\ y := y + 1\ \{ y < 4 \}$

$\{y < 3\}$ (implied)

$\{y + 1 < 4\}$ (assignment)

$y := y + 1$

$\{y < 4\}$

using proof rules for PCA's (continued)

- show $\vdash_{\text{par}} \{\, \top \,\}\, z := x;\ z := z + y;\ u := z;\ \{\, u = x + y \,\}$

using proof rules for PCA's (continued)

- show $\vdash_{\text{par}} \{\top\}\ z := x;\ z := z + y;\ u := z;\ \{u = x + y\}$

$z := x;$

$z := z + y;$

$u := z;$

using proof rules for PCA's (continued)

- show $\vdash_{\text{par}} \{\,\top\,\}\; z := x;\; z := z + y;\; u := z;\; \{\, u = x + y \,\}$

$z := x;$

$z := z + y;$

$u := z;$

$\{u = x + y\}$

using proof rules for PCA's (continued)

- show $\vdash_{\mathsf{par}}\ \{\ \top\ \}\ z := x;\ z := z + y;\ u := z;\ \{\ u = x + y\ \}$

$z := x;$

$z := z + y;$

$\qquad \{z = x + y\}$ (assignment)

$u := z;$

$\qquad \{u = x + y\}$

using proof rules for PCA's (continued)

- show $\vdash_{\text{par}}\ \{\ \top\ \}\ z := x;\ z := z + y;\ u := z;\ \{\ u = x + y\ \}$

$$
\begin{array}{ll}
z := x; & \\
\qquad \{z + y = x + y\} & \text{(assignment)} \\
z := z + y; & \\
\qquad \{z = x + y\} & \text{(assignment)} \\
u := z; & \\
\qquad \{u = x + y\} &
\end{array}
$$

using proof rules for PCA's (continued)

- show $\vdash_{\text{par}} \{\top\}\ z := x;\ z := z + y;\ u := z;\ \{u = x + y\}$

$\qquad \{x + y = x + y\}$	(assignment)
$z := x;$	
$\qquad \{z + y = x + y\}$	(assignment)
$z := z + y;$	
$\qquad \{z = x + y\}$	(assignment)
$u := z;$	
$\qquad \{u = x + y\}$	

using proof rules for PCA's (continued)

- show $\vdash_{\text{par}} \{ \top \}\; z := x;\; z := z + y;\; u := z;\; \{ u = x + y \}$

$\{\top\}$	(implied)
$\{x + y = x + y\}$	(assignment)
$z := x;$	
$\{z + y = x + y\}$	(assignment)
$z := z + y;$	
$\{z = x + y\}$	(assignment)
$u := z;$	
$\{u = x + y\}$	

wrong uses of the rule (assignment)

wrong uses of the rule (assignment)

$$\frac{}{\{\ \varphi\ \}\ x := E\ \{\ \varphi[x \mapsto E]\ \}} \quad \text{assignment-wrong-1}$$

wrong uses of the rule (assignment)

$$\frac{}{\{\ \varphi\ \}\ x := E\ \{\ \varphi[x \mapsto E]\ \}} \quad \text{assignment-wrong-1}$$

rule (assignment-wrong-1) allows us to show
$\vdash_{\text{par}}\ \{\ x = 0\ \}\ x := 1\ \{\ 1 = 0\ \}$

wrong uses of the rule (assignment)

$$\frac{}{\{\ \varphi\ \}\ x := E\ \{\ \varphi[x \mapsto E]\ \}}\quad \text{assignment-wrong-1}$$

rule (assignment-wrong-1) allows us to show
$\vdash_{\mathsf{par}}\ \{\ x = 0\ \}\ x := 1\ \{\ 1 = 0\ \}$

$$\frac{}{\{\ \varphi\ \}\ x := E\ \{\ \varphi[E \mapsto x]\ \}}\quad \text{assignment-wrong-2}$$

wrong uses of the rule (assignment)

$$\frac{}{\{\ \varphi\ \}\ x := E\ \{\ \varphi[x \mapsto E]\ \}}\quad \text{assignment-wrong-1}$$

rule (assignment-wrong-1) allows us to show
$\vdash_{\text{par}}\ \{\ x = 0\ \}\ x := 1\ \{\ 1 = 0\ \}$

$$\frac{}{\{\ \varphi\ \}\ x := E\ \{\ \varphi[E \mapsto x]\ \}}\quad \text{assignment-wrong-2}$$

rule (assignment-wrong-2) allows us to show
$\vdash_{\text{par}}\ \{\ x = 0\ \}\ x := 1\ \{\ x = 0\ \}$

more proof rules for PCA's?

Are the following proof rules (not in the book) **sound**?

more proof rules for PCA's?

Are the following proof rules (not in the book) **sound**?

$$\frac{\{\ \varphi_1\ \}\ C\ \{\ \psi_1\ \} \qquad \{\ \varphi_2\ \}\ C\ \{\ \psi_2\ \}}{\{\ \varphi_1 \wedge \varphi_2\ \}\ C\ \{\ \psi_1 \wedge \psi_2\ \}} \quad \text{spec conjunction}$$

more proof rules for PCA's?

Are the following proof rules (not in the book) **sound**?

$$\frac{\{\ \varphi_1\ \}\ C\ \{\ \psi_1\ \} \qquad \{\ \varphi_2\ \}\ C\ \{\ \psi_2\ \}}{\{\ \varphi_1 \wedge \varphi_2\ \}\ C\ \{\ \psi_1 \wedge \psi_2\ \}}\quad \text{spec conjunction}$$

$$\frac{\{\ \varphi_1\ \}\ C\ \{\ \psi_1\ \} \qquad \{\ \varphi_2\ \}\ C\ \{\ \psi_2\ \}}{\{\ \varphi_1 \vee \varphi_2\ \}\ C\ \{\ \psi_1 \vee \psi_2\ \}}\quad \text{spec disjunction}$$

more proof rules for PCA's?

Are the following proof rules (not in the book) **sound**?

$$\frac{\{\ \varphi_1\ \}\ C\ \{\ \psi_1\ \} \qquad \{\ \varphi_2\ \}\ C\ \{\ \psi_2\ \}}{\{\ \varphi_1 \wedge \varphi_2\ \}\ C\ \{\ \psi_1 \wedge \psi_2\ \}} \quad \text{spec conjunction}$$

$$\frac{\{\ \varphi_1\ \}\ C\ \{\ \psi_1\ \} \qquad \{\ \varphi_2\ \}\ C\ \{\ \psi_2\ \}}{\{\ \varphi_1 \vee \varphi_2\ \}\ C\ \{\ \psi_1 \vee \psi_2\ \}} \quad \text{spec disjunction}$$

YES!

more proof rules for PCA's?

Are the following proof rules (not in the book) **sound**?

$$\frac{\{\ \varphi_1\ \}\ C\ \{\ \psi_1\ \} \qquad \{\ \varphi_2\ \}\ C\ \{\ \psi_2\ \}}{\{\ \varphi_1 \wedge \varphi_2\ \}\ C\ \{\ \psi_1 \wedge \psi_2\ \}} \quad \text{spec conjunction}$$

$$\frac{\{\ \varphi_1\ \}\ C\ \{\ \psi_1\ \} \qquad \{\ \varphi_2\ \}\ C\ \{\ \psi_2\ \}}{\{\ \varphi_1 \vee \varphi_2\ \}\ C\ \{\ \psi_1 \vee \psi_2\ \}} \quad \text{spec disjunction}$$

YES!

Are these derivable from the rules in Handout 18?

more proof rules for PCA's?

Are the following proof rules (not in the book) **sound**?

$$\frac{\{\ \varphi_1\ \}\ C\ \{\ \psi_1\ \} \qquad \{\ \varphi_2\ \}\ C\ \{\ \psi_2\ \}}{\{\ \varphi_1 \wedge \varphi_2\ \}\ C\ \{\ \psi_1 \wedge \psi_2\ \}} \quad \text{spec conjunction}$$

$$\frac{\{\ \varphi_1\ \}\ C\ \{\ \psi_1\ \} \qquad \{\ \varphi_2\ \}\ C\ \{\ \psi_2\ \}}{\{\ \varphi_1 \vee \varphi_2\ \}\ C\ \{\ \psi_1 \vee \psi_2\ \}} \quad \text{spec disjunction}$$

YES!

Are these derivable from the rules in Handout 18?

ALMOST . . .

more program constructs

more program constructs

- Exercise 4.2.2, page 299, in [LCS]: **for** loops

more program constructs

- Exercise 4.2.2, page 299, in [LCS]: **for** loops
- Exercise 4.2.3, page 299, in [LCS]: **repeat-until** loops

more program constructs (not in the book):
an imperative language + nondeterminism + concurrency

more program constructs (not in the book): an imperative language + nondeterminism + concurrency

- integer expressions

  $E \ ::= \ \ldots$ (as before)

more program constructs (not in the book): an imperative language + nondeterminism + concurrency

- integer expressions

  $E \quad ::= \quad \ldots$ (as before)
- boolean expressions

  $B \quad ::= \quad \ldots$ (as before)

more program constructs (not in the book): 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)

  $$C \quad ::= \quad x := E \;\mid\; C; C \;\mid\; \textbf{if } B \textbf{ then } C \textbf{ else } C \;\mid\; \textbf{while } B \textbf{ do } C \textbf{ od}$$

more program constructs (not in the book): an imperative language + nondeterminism + concurrency

- integer expressions

  $E \quad ::= \quad \ldots$ (as before)

- boolean expressions

  $B \quad ::= \quad \ldots$ (as before)

- program expressions (or commands)

  $C \quad ::= \quad x := E \;\mid\; C; C \;\mid\; \textbf{if } B \textbf{ then } C \textbf{ else } C \;\mid\; \textbf{while } B \textbf{ do } C \textbf{ od}$

  $\quad\quad \mid \quad C \cup C$ (nondeterminism)

more program constructs (not in the book): 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)

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

more program constructs (not in the book): an imperative language + nondeterminism + concurrency

- integer expressions

  $E \quad ::= \quad \ldots$ (as before)

- boolean expressions

  $B \quad ::= \quad \ldots$ (as before)

- program expressions (or commands)

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

- execution of program $(x := 1) \cup (x := 2)$
  nondeterministically sets x either to 1 or to 2

more program constructs (not in the book): an imperative language + nondeterminism + concurrency

- integer expressions

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

- boolean expressions

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

- program expressions (or commands)

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

- 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.

more program constructs (not in the book):
an imperative language + nondeterminism + concurrency

more program constructs (not in the book): an imperative language + nondeterminism + concurrency

- Write proof rules for **concurrency**

more program constructs (not in the book): an imperative language + nondeterminism + concurrency

- Write proof rules for **concurrency**
- Write proof rules for **non-determinism**

(THIS PAGE INTENTIONALLY LEFT BLANK)