Hoare Semantics

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Objectives

You should be able to ...

- Explain the syntax of Hoare triples and relate them to small step semantics.
- Use a Hoare triple to show the correctness of a simple program.
- Explain the properties of the weakest precondition.

Review of Language Syntax

The Language

$$
\begin{array}{rl}
S ::= & \mathtt{skip} \\
\mid & u := t \\
\mid & S_1 ; S_2 \\
\mid & \mathtt{if}\ B\ \mathtt{then}\ S_1\ \mathtt{else}\ S_2\ \mathtt{fi} \\
\mid & \mathtt{while}\ B\ \mathtt{do}\ S_1\ \mathtt{od}
\end{array}
$$

- The **else** branch can be left off if the subexpression is simply a skip.
- $var(S)$ is the set of variable names appearing in S.
- $change(S)$ is the set of variables appearing on the LHS of `:=`.

Definition of $\to$

Skip and Assignment

$$< \mathtt{skip}\,, \sigma > \to < E, \sigma >$$

$$< u := t, \sigma > \to < E, \sigma[u := \sigma(t)] >$$

$$\frac{< S_1, \sigma > \to < S_2, \tau >}{< S_1; S, \sigma > \to < S_2; S, \tau >}$$

$$E; S \equiv S$$

$$< \mathtt{if}\ B\ \mathtt{then}\ S_1\ \mathtt{else}\ S_2\ \mathtt{fi}\,, \sigma > \to < S_1, \sigma > \text{ where } \sigma \models B$$

$$< \mathtt{if}\ B\ \mathtt{then}\ S_1\ \mathtt{else}\ S_2\ \mathtt{fi}\,, \sigma > \to < S_2, \sigma > \text{ where } \sigma \models \neg B$$

$$< \mathtt{while}\ B\ \mathtt{do}\ S_1\ \mathtt{od}\,, \sigma > \to < S_1; \mathtt{while}\ B\ \mathtt{do}\ S_1\ \mathtt{od}\,, \sigma > \text{ where } \sigma \models B$$

$$< \mathtt{while}\ B\ \mathtt{do}\ S_1\ \mathtt{od}\,, \sigma > \to < E, \sigma > \text{ where } \sigma \models \neg B$$

Hoare Triples

- The $\rightarrow$ semantics gives us exact transformations, but sometimes we want something more general.
- Define $\{p\}S\{q\}$, where p and q are assertions, and S is a program:
  - $\models \{p\}S\{q\}$ – if p is true before the program runs, q will be true afterwards; if the program terminates. "Partial Correctness"
  - $\models_{tot} \{p\}S\{q\}$ – if p is true before the program runs, q will be true afterwards. Termination guaranteed. "Total Correctness"
- These are sometimes called *correctness formulas*.

Examples

- $\{x = 0\}x := x + 1\{x = 1\}$
- $\{x = 0\}x := x + 1\{x > 0\}$
- $\{x = 0\}x := x + 1\{\mathit{true}\}$

False formulas ...

- $\{x = 0\}x := x + 1\{x = 2\}$
- $\{x = 0\}x := x + 1\{x < 0\}$

What does this one mean? $\{x = 0\}x := x + 1\{\mathit{false}\}$

Axiom 1: Skip

$$\{p\}\,\mathtt{\textbf{skip}}\,\{p\}$$

Axiom 2: Assignment

$$\{p[u := t]\}u := t\{p\}$$

- Is this what you expected?

$$\{y > 10\}x := y\{x > 10\}$$

Rule 3: Composition

$$\frac{\{p\}S_1\{r\},\ \{r\}S_2\{q\}}{\{p\}S_1;S_2\{q\}}$$

Rule 4: Conditional

$$\frac{\{p \wedge B\}S_1\{q\}, \{p \wedge \neg B\}S_2\{q\}}{\{p\}\texttt{if } B \texttt{ then } S_1 \texttt{ else } S_2 \texttt{ fi } \{q\}}$$

- See Dijkstra's paper EWD 264.

Rule 5: Loop

$$\frac{\{p \wedge B\} S \{p\}}{\{p\}\texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}\ \{p \wedge \neg B\}}$$

Rule 6: Consequence

- This one you will use a *lot*.

$$\frac{p \to p_1, \{p_1\}S\{q_1\}, q_1 \to q}{\{p\}S\{q\}}$$

Skip, Assignment, and Sequence

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

$$\{p[u := t]\}u := t\{p\}$$

$$\frac{\{p\}S_1^*\{r\},\ \{r\}S_2^*\{q\}}{\{p\}S_1^*;\ \{r\}S_2^*\{q\}}$$

- Example

$$\begin{array}{l}\{y = 20, x = 10\}\\ t := x;\\ \{y = 20, t = 10\}\\ x := y;\\ \{x = 20, t = 10\}\\ y := t\\ \{x = 20, y = 10\}\end{array}$$

If, Consequence

$$\frac{\{p \wedge B\}S_1^*\{q\}, \{p \wedge \neg B\}S_2^*\{q\}}{\{p\}\mathbf{if}\ B\ \mathbf{then}\ \{p \wedge B\}S_1^*\{q\}\ \mathbf{else}\ \{p \wedge \neg B\}S_2^*\{q\}\ \mathbf{fi}\ \{q\}}$$

$$\frac{p \rightarrow p_1, \{p_1\}S^*\{q_1\}, q_1 \rightarrow q}{\{p\}\{p_1\}S^*\{q_1\}\{q\}}$$

Activity

Try to verify the following program.

$$\{true\} \begin{array}{l} \texttt{if } x > y \texttt{ then } m := x \texttt{ fi } ; \\ \texttt{if } x < y \texttt{ then } m := y \texttt{ fi} \end{array} \{m = max(x, y)\}$$

(Hint: actually, it's not true!)

The Verification

$$
\begin{array}{l}
\{true\} \\
\textbf{if } x > y \textbf{ then } m := x \textbf{ fi } ; \\
\textbf{if } x < y \textbf{ then } m := y \textbf{ fi} \\
\{m = max(x, y)\}
\end{array}
$$

The Verification

$$
\begin{array}{l}
\{true\} \\
\mathbf{if}\ x > y\ \mathbf{then}\ m := x\ \mathbf{fi}\ ; \\
\mathbf{if}\ x < y\ \mathbf{then}\ \{y = max(x,y)\} m := y \{m = max(x,y)\} \\
\qquad\qquad \mathbf{else}\ \{m = max(x,y)\} \mathbf{skip}\ \{m = max(x,y)\}\ \mathbf{fi} \\
\{m = max(x,y)\}
\end{array}
$$

The Verification

$$
\begin{array}{l}
\{\mathit{true}\} \\
\mathtt{if}\ x > y\ \mathtt{then}\ m := x\ \mathtt{fi}\ ; \\
\{P \equiv x < y \wedge y = \mathit{max}(x,y) \vee x > y \wedge m = \mathit{max}(x,y)\} \\
\mathtt{if}\ x < y\ \mathtt{then}\ \{y = \mathit{max}(x,y)\} m := y \{m = \mathit{max}(x,y)\} \\
\qquad\qquad \mathtt{else}\ \{m = \mathit{max}(x,y)\} \mathtt{skip}\ \{m = \mathit{max}(x,y)\}\ \mathtt{fi} \\
\{m = \mathit{max}(x,y)\}
\end{array}
$$

The Verification

$$
\begin{array}{l}
\{true\} \\
\textbf{if } x > y \textbf{ then } \{P[x = max(x,y)]\} m := x \{P\} \\
\qquad\qquad \textbf{else } \{P\} \textbf{skip } \{P\} \textbf{ fi };\\
\{P \equiv x < y \wedge y = max(x,y) \vee x > y \wedge m = max(x,y)\} \\
\textbf{if } x < y \textbf{ then } \{y = max(x,y)\} m := y \{m = max(x,y)\} \\
\qquad\qquad \textbf{else } \{m = max(x,y)\} \textbf{skip } \{m = max(x,y)\} \textbf{ fi} \\
\{m = max(x,y)\}
\end{array}
$$

The Verification

$$
\begin{array}{l}
\{true\} \\
\{x > y \wedge x = max(x,y) \vee x < y \wedge y = max(x,y) \vee x = y \wedge m = max(x,y)\} \\
\textbf{if } x > y \textbf{ then } \{P[x = max(x,y)]\} m := x \{P\} \\
\qquad\qquad \textbf{else } \{P\} \textbf{skip } \{P\} \textbf{ fi };\\
\{P \equiv x < y \wedge y = max(x,y) \vee x \geq y \wedge m = max(x,y)\} \\
\textbf{if } x < y \textbf{ then } \{y = max(x,y)\} m := y \{m = max(x,y)\} \\
\qquad\qquad \textbf{else } \{m = max(x,y)\} \textbf{skip } \{m = max(x,y)\} \textbf{ fi} \\
\{m = max(x,y)\}
\end{array}
$$