Hoare Logic

An Annotated Language

$$
\begin{array}{llll}
\mathbf{Exp_{int}} & \ni & e & ::= & \ldots \mid -1 \mid 0 \mid 1 \mid \ldots \mid x \\
& & & & \mid -e \mid e + e \mid e - e \mid e * e \mid e \ \mathbf{div}\ e \mid e \ \mathbf{mod}\ e \\
\\
\mathbf{Exp_{bool}} & \ni & b & ::= & \mathbf{true} \mid \mathbf{false} \mid !b \mid b \,\&\&\, b \mid b \,\|\, b \\
& & & & \mid e == e \mid e < e \mid e <= e \mid e > e \mid e >= e \mid e != e \\
\\
\mathbf{Comm} & \ni & C & ::= & \mathbf{skip} \mid C \,;\, C \mid x := e \mid \mathbf{if}\ b\ \mathbf{then}\ C\ \mathbf{else}\ C \mid \\
& & & & \mid \mathbf{while}\ b\ \mathbf{do}\ \{A\}\, C \\
\\
\mathbf{Term} & \ni & t & ::= & \ldots \mid -1 \mid 0 \mid 1 \mid \ldots \mid x \\
& & & & \mid -t \mid t + t \mid t - t \mid t * t \mid t \ \mathbf{div}\ t \mid t \ \mathbf{mod}\ t \\
& & & & \mid f(t_1, \ldots, t_{\alpha(f)}) \\
\\
\mathbf{Assert} & \ni & A & ::= & \mathbf{true} \mid \mathbf{false} \mid !A \mid A \,\&\&\, A \mid A \,\|\, A \\
& & & & \mid A \rightarrow A \mid \mathtt{Forall}\ x.\ A \mid \mathtt{Exists}\, x.\, A \\
& & & & \mid t == t \mid t < t \mid t <= t \mid t > t \mid t >= t \mid t != t \\
& & & & \mid p(t_1, \ldots, t_{\alpha(p)})
\end{array}
$$

State and Semantics

$$\Sigma = \mathcal{V} \rightarrow \mathbb{Z}$$

$$\begin{aligned} [\![\cdot]\!]_{\mathbf{Exp_{int}}} &: \mathbf{Exp_{int}} \rightarrow \Sigma \rightarrow \mathbb{Z} \\ [\![\cdot]\!]_{\mathbf{Exp_{bool}}} &: \mathbf{Exp_{bool}} \rightarrow \Sigma \rightarrow \{\mathit{true}, \mathit{false}\} \end{aligned}$$

- Expressions are interpreted as functions from states to the corresponding domain of interpretation
- Operators have the obvious interpretation
- Free of side effects

Command Semantics

- *Natural*, or *big-step* semantics
- Evaluation relation
- See last lecture for definition!

$$\Downarrow \subseteq \mathbf{Comm} \times \Sigma \times \Sigma$$

Assertion Semantics

$$\begin{aligned} [\![\cdot]\!]_{\mathbf{Term}} &: \quad \mathbf{Term} \rightarrow \Sigma \rightarrow \mathbb{Z} \\ [\![\cdot]\!]_{\mathbf{Assert}} &: \quad \mathbf{Assert} \rightarrow \Sigma \rightarrow \{true, false\} \end{aligned}$$

- Terms are interpreted very similarly to (program) integer expressions (but functions remain uninterpreted)
- Assertions are interpreted similarly to (program) boolean expressions (but predicates remain uninterpreted)

$$
\begin{array}{rcl}
[\![\mathbf{true}]\!](s) & = & \mathit{true} \\
[\![\mathbf{false}]\!](s) & = & \mathit{false} \\
 & & \\
[\![e_1 \Box e_2]\!](s) & = & [\![e_1]\!](s) \boxdot [\![e_2]\!](s), \text{ where } \boxdot = I(\Box) \\
 & & \\
[\![\,!A]\!](s) & = & \neg [\![A]\!](s) \\
[\![A_1 \mathbin{\&\&} A_2]\!](s) & = & [\![A_1]\!](s) \wedge [\![A_2]\!](s), \\
[\![A_1 \parallel A_2]\!](s) & = & [\![A_1]\!](s) \vee [\![A_2]\!](s), \\
[\![A_1 \;\rightarrow\; A_2]\!](s) & = & [\![A_1]\!](s) \Longrightarrow [\![A_2]\!](s), \\
 & & \\
[\![\texttt{Forall}\ x.\ A]\!](s) & = & \forall v \in \mathbb{Z}.\ [\![A]\!](s[x \mapsto v]) \quad \text{, with } v \text{ fresh} \\
[\![\texttt{Exists}\, x.\ A]\!](s) & = & \exists v \in \mathbb{Z}.\ [\![A]\!](s[x \mapsto v]) \quad \text{, with } v \text{ fresh}
\end{array}
$$

Validity

$$\models A, \text{ if } [\![A]\!](s) = \mathit{true} \text{ for all states } s \in \Sigma$$

$$\models \mathcal{M} \text{ if } \models A \text{ holds for every } A \in \mathcal{M}$$

In logical terms one considers a model (Z, I), where I is the interpretation function that maps constants to their obvious interpretations as integer numbers, and maps functions and predicates to the corresponding arithmetic functions and comparison predicates (see Logic lectures!)

- We assume the existence of "external" means for checking validity of assertions (see Logic lectures)
- These tools should allow us to define theories, i.e. to write axioms concerning the behaviour of the uninterpreted functions and predicates
- The following axiomatisation of factorial with a binary predicate is an example of this (not satisfying – what is missing?)

$$isfact(0, 1)$$

$$\texttt{Forall}\ n, r.\ n > 0 \ \rightarrow\ isfact(n-1, r)\ \rightarrow\ isfact(n, n * r)$$

Correctness Properties

- A *total correctness* property for a program C relative to specification (P, Q) has the following meaning:

  if P holds in a given state and C is executed in that state, then execution of C will stop, and moreover Q will hold in the final state of execution.

- A *partial correctness* property for a program C relative to specification (P, Q) has the meaning:

  if P holds in a given state and C is executed in that state, then either execution of C does not stop, or if it does, Q will hold in the final state.

Specifications and Hoare triples

- new syntactic classes for partial and total correctness

$$\mathbf{Spec} \ni S ::= \{A\}\, C\, \{A\} \mid [A]\, C\, [A]$$

$$\llbracket\cdot\rrbracket_{\mathbf{Spec}} : \mathbf{Spec} \to \{\mathit{true}, \mathit{false}\}$$

$$\llbracket\{P\}\, C\, \{Q\}\rrbracket = \forall s, s' \in \Sigma.\ \llbracket P\rrbracket(s) \wedge (C, s) \Downarrow s' \implies \llbracket Q\rrbracket(s')$$

$$\llbracket[P]\, C\, [Q]\rrbracket = \forall s \in \Sigma.\ \exists s' \in \Sigma.(C, s) \Downarrow s' \wedge (\llbracket P\rrbracket(s) \implies \llbracket Q\rrbracket(s')$$

Loop Invariants

- Any property whose validity is preserved by executions of the loop's body.
- Since these executions may only take place when the loop condition is true, an invariant of the loop
  `while (b) do C`
  is any assertion `I` such that `{I && b} C {I}` is valid, in which case of course it also holds that
  `{I} while (b) do C {I}` is valid

  (WHY?)

- The validity of `[I && b] C [I]` does not however imply the validity of
  `[I] while (b) do C [I]`

  (WHY?)

- The required notion here is a quantitative one: a loop variant is any positive integer expression that is strictly decreasing from iteration to iteration.

- The existence of a valid variant for a given loop implies the termination of all possible executions of the loop

Example

$$\{n \geq 0\}\, \mathbf{fact}\, \{f == fact(n)\}$$

$$
\begin{array}{l}
\mathbf{fact} \doteq \\
\quad f := 1\,;\, i := 1\,; \\
\quad \mathbf{while}\ i \leq n\ \mathbf{do}\, \{f == fact(i-1)\ \&\&\ i \leq n+1\} \\
\quad\quad f := f * i\,; \\
\quad\quad i := i + 1
\end{array}
$$

- How can this be proved?
- What is the role of the invariant?

Hoare Logic

$$(skip) \quad \frac{}{\{P\}\,\mathbf{skip}\,\{P\}} \qquad (assign) \quad \frac{}{\{Q[x \mapsto e]\}\; x := e\, \{Q\}}$$

$$(seq) \quad \frac{\{P\}\,C_1\,\{R\} \qquad \{R\}\,C_2\,\{Q\}}{\{P\}\,C_1\,;\,C_2\,\{Q\}}$$

$$(while) \quad \frac{\{I \;\&\&\, b\}\,C\,\{I\}}{\{I\}\,\mathbf{while}\ b\ \mathbf{do}\,\{I\}\,C\,\{I \;\&\&\; !b\}}$$

$$(if) \quad \frac{\{P \;\&\&\, b\}\,C_t\,\{Q\} \qquad \{P \;\&\&\; !b\}\,C_f\,\{Q\}}{\{P\}\,\mathbf{if}\ b\ \mathbf{then}\ C_t\ \mathbf{else}\ C_f\,\{Q\}}$$

The Consequence Rule

$$\frac{\{P\}\, C\, \{Q\}}{\{P'\}\, C\, \{Q'\}} \quad \text{if} \ \models P' \to P \ \text{and} \ \models Q \to Q'$$

- Side conditions must be met in order for this rule to be applied in a derivation, corresponding to the validity of certain first-order implication formulas.
- If some of the rules applied in a tree have side conditions that are not satisfied, then this is not a proof-tree and the triple at its root is not inferred.

Total Correctness

The identification of a decreasing variant expression is necessary to guarantee that every loop terminates

$$\frac{[I \ \&\& \ B \ \&\& \ V == n] \, C \, [I \ \&\& \ V < n] \qquad\qquad I \ \&\& \ B \ \rightarrow \ V >= 0}{[I] \, \mathbf{while} \, (B) \, \mathbf{do} \, C \, [I \ \&\& \ \neg B]}$$

Exercise

Prove the validity of the following Hoare triple

$$\{x \geq -100 \ \&\& \ x \leq 100\}$$
$$\textbf{if } x < 0 \textbf{ then } x = x + 100 \textbf{ else skip} \ ; \ y = 2 * x$$
$$\{y \geq 0 \ \&\& \ y \leq 300\}$$

$\{x \geq -100 \ \&\& \ x \leq 100\}$
$\textbf{if } x < 0 \textbf{ then } x = x + 100 \textbf{ else skip} \, ; \; y = 2 * x$
$\{y \geq 0 \ \&\& \ y \leq 300\}$

(seq)

1. $\{x \geq -100 \ \&\& \ x \leq 100\} \textbf{ if } x < 0 \textbf{ then } x = x + 100 \textbf{ else skip} \{x \geq 0 \ \&\& \ x \leq 150\}$ (if)

   1.1 $\{x \geq -100 \ \&\& \ x \leq 100 \ \&\& \ x < 0\} \, x = x + 100 \, \{x \geq 0 \ \&\& \ x \leq 150\}$ $(conseq)$

      1.1.1. $\{x \geq -100 \ \&\& \ x \leq 50\} \, x = x + 100 \, \{x \geq 0 \ \&\& \ x \leq 150\}$ $(assign)$

   1.2. $\{x \geq -100 \ \&\& \ x \leq 100 \ \&\& \ !(x < 0)\} \textbf{ skip} \{x \geq 0 \ \&\& \ x \leq 150\}$ $(conseq)$

      1.2.1. $\{x \geq 0 \ \&\& \ x \leq 150\} \textbf{ skip} \{x \geq 0 \ \&\& \ x \leq 150\}$ $(skip)$

2. $\{x \geq 0 \ \&\& \ x \leq 150\} \, y = 2 * x \, \{y \geq 0 \ \&\& \ y \leq 300\}$ $(assign)$

What are the side conditions?

The next slide contains an alternative derivation

$\{x \geq -100 \; \&\& \; x \leq 100\}$
$\textbf{if } x < 0 \textbf{ then } x = x + 100 \textbf{ else skip} \; ; \; y = 2 * x$
$\{y \geq 0 \; \&\& \; y \leq 300\}$

$(conseq)$

1. $\{x \geq -100 \; \&\& \; x \leq 100\}$
   $\textbf{if } x < 0 \textbf{ then } x = x + 100 \textbf{ else skip} \; ; \; y = 2 * x$
   $\{y \geq 0 \; \&\& \; y \leq 200\}$
   (seq)

   1.1. $\{x \geq -100 \; \&\& \; x \leq 100\} \textbf{ if } x < 0 \textbf{ then } x = x + 100 \textbf{ else skip } \{x \geq 0 \; \&\& \; x \leq 100\}$ (if)

      1.1.1. $\{x \geq -100 \; \&\& \; x \leq 100 \; \&\& \; x < 0\} \; x = x + 100 \; \{x \geq 0 \; \&\& \; x \leq 100\}$ $(conseq)$

         1.1.1.1. $\{x \geq -100 \; \&\& \; x \leq 0\} \; x = x + 100 \; \{x \geq 0 \; \&\& \; x \leq 100\}$ $(assign)$

      1.1.2. $\{x \geq -100 \; \&\& \; x \leq 100 \; \&\& \; !(x < 0)\} \textbf{ skip } \{x \geq 0 \; \&\& \; x \leq 100\}$ $(conseq)$

         1.1.2.1. $\{x \geq 0 \; \&\& \; x \leq 100\} \textbf{ skip } \{x \geq 0 \; \&\& \; x \leq 100\}$ $(skip)$

   1.2. $\{x \geq 0 \; \&\& \; x \leq 100\} \; y = 2 * x \; \{y \geq 0 \; \&\& \; y \leq 200\}$ $(assign)$

Correctness of Hoare Logic

$$\text{If} \vdash \{P\}\, C\, \{Q\}, \text{ then } [\![\{P\}\, C\, \{Q\}]\!] = \mathit{true}$$

<u>Proof</u>

By induction on the derivation of $\vdash$ `{P}C{Q}`.
For the while case we also proceed by induction on the definition of the evaluation relation.

Problems with HL System

- Two desirable properties for backward proof construction are missing:
  - Sub-formula property
  - Unambiguous choice of rule
- The consequence rule causes ambiguity. Its presence is however necessary to make possible the application of rules for *skip*, *assignment*, and *while*
- An alternative is to *distribute* the side conditions among the different rules

HL without Consequ. Rule

$$\frac{}{\{P\}\,\textbf{skip}\,\{Q\}} \quad \text{if } \models P \to Q \qquad\qquad \frac{}{\{P\}\,x := e\,\{Q\}} \quad \text{if } \models P \to Q[x \mapsto e]$$

$$\frac{\{P\}\,C_1\,\{R\} \qquad \{R\}\,C_2\,\{Q\}}{\{P\}\,C_1\,;\,C_2\,\{Q\}}$$

$$\frac{\{I\ \&\&\ b\}\,C\,\{I\}}{\{P\}\,\textbf{while}\ b\ \textbf{do}\,\{I\}\,C\,\{Q\}} \quad \text{if } \models P \to I \ \text{ and } \models I\ \&\&\ !b \to Q$$

$$\frac{\{P\ \&\&\ b\}\,C_t\,\{Q\} \qquad \{P\ \&\&\ !b\}\,C_f\,\{Q\}}{\{P\}\,\textbf{if}\ b\ \textbf{then}\ C_t\ \textbf{else}\ C_f\,\{Q\}}$$

Factorial Example

$\{n \geq 0\}\, \textbf{fact}\, \{f == fact(n)\}$

1. $\{n \geq 0\}\, f := 1\,;\, i := 1\, \{n \geq 0 \,\&\&\, f == 1 \,\&\&\, i == 1\}$

    1.1 $\{n \geq 0\}\, f := 1\, \{n \geq 0 \,\&\&\, f == 1\}$

    1.2 $\{n \geq 0 \,\&\&\, f == 1\}\, i := 1\, \{n \geq 0 \,\&\&\, f == 1 \,\&\&\, i == 1\}$

2. $\{n \geq 0 \,\&\&\, f == 1 \,\&\&\, i == 1\}\, \textbf{while}\ i \leq n\ \textbf{do}\, \{f == fact(i-1) \,\&\&\, i \leq n+1\}\, C_b\, \{f == fact(n)\}$

    2.1. $\{f == fact(i-1) \,\&\&\, i \leq n\}\, C_b\, \{f == fact(i-1) \,\&\&\, i \leq n+1\}$

        2.1.1. $\{f == fact(i-1) \,\&\&\, i \leq n\}\, f := f * i\, \{f == fact(i-1) * i \,\&\&\, i \leq n\}$

        2.1.2. $\{f == fact(i-1) * i \,\&\&\, i \leq n\}\, i := i+1\, \{f == fact(i-1) \,\&\&\, i \leq n+1\}$

with side conditions:

$$1.1 \ \models n \geq 0 \ \rightarrow \ (n \geq 0 \ \&\& \ f == 1)[f \mapsto 1]$$

$$1.2 \ \models n \geq 0 \ \&\& \ f == 1 \ \rightarrow \ (n \geq 0 \ \&\& \ f == 1 \ \&\& \ i == 1)[i \mapsto 1]$$

$$2. \ \models n \geq 0 \ \&\& \ f == 1 \ \&\& \ i == 1 \ \rightarrow \ f == \mathit{fact}(i-1) \ \&\& \ i \leq n+1 \text{ and}$$
$$\models f == \mathit{fact}(i-1) \ \&\& \ i \leq n+1 \ \&\& \ !i \leq n \ \rightarrow \ f == \mathit{fact}(n)$$

$$2.1.1. \ \models f == \mathit{fact}(i-1) \ \&\& \ i \leq n \ \rightarrow \ (f == \mathit{fact}(i-1) * i \ \&\& \ i \leq n)[f \mapsto f * i]$$

$$2.1.2. \ \models f == \mathit{fact}(i-1) * i \ \&\& \ i \leq n \ \rightarrow \ (f == \mathit{fact}(i-1) \ \&\& \ i \leq n+1)[i \mapsto i+1]$$

Exercise

- Show that a triple is provable in this system iff it is provable in the original system of Hoare logic.

Auxiliary Variables

- How to specify formally a program that computes the minimum and maximum of a pair of numbers?

$$\textbf{if } x \leq y \textbf{ then skip else } z := y\,;\, y := x\,;\, x := z$$

Solution

- Employ *auxiliary variables*, forbidden to occur in the program, to record initial values of variables

$$(x == x_0 \;\&\&\; y == y_0,$$
$$x \leq y \;\&\&\; ((x == x_0 \;\&\&\; y == y_0) \,\|\, (x == y_0 \;\&\&\; y == x_0)))$$

Factorial Spec. Updated

$$(n \geq 0 \;\&\&\; n == n_0, f == fact(n_0))$$

or

$$(n \geq 0 \;\&\&\; n == n_0, f == fact(n) \;\&\&\; n == n_0)$$

A Strategy for Proofs

- When an intemediate assertion is required, if possible choose the *weakest* precondition (for the given postcondition)

Example:

$$\{P\}\, x := e_1\, ;\, y := e_2\, ;\, z := e_3\, \{Q\}$$

$$\boxed{1.}\quad \{P\}\, x := e_1\, ;\, y := e_2\, \{R\}$$

$$\boxed{2.}\quad \{R\}\, z := e_3\, \{Q\}$$

A Strategy for Proofs

$$\{P\}\, x := e_1 \,;\, y := e_2 \,;\, z := e_3 \,\{Q\}$$

$\boxed{1.}$ $\{P\}\, x := e_1 \,;\, y := e_2 \,\{Q[z \mapsto e_3]\}$

2. $\{Q[z \mapsto e_3]\}\, z := e_3 \,\{Q\}$

A Strategy for Proofs

$$\{P\}\, x := e_1 \,;\, y := e_2 \,;\, z := e_3 \,\{Q\}$$

$\boxed{1.}$ $\{P\}\, x := e_1 \,;\, y := e_2 \,\{Q[z \mapsto e_3]\}$

1.1. $\{P\}\, x := e_1 \,\{Q[z \mapsto e_3][y \mapsto e_2]\}$

$\boxed{1.2.}$ $\{Q[z \mapsto e_3][y \mapsto e_2]\}\, y := e_2 \,\{Q[z \mapsto e_3]\}$

2. $\{Q[z \mapsto e_3]\}\, z := e_3 \,\{Q\}$

A Strategy for Proofs

$$\{P\}\, x := e_1\, ;\, y := e_2\, ;\, z := e_3\, \{Q\}$$

1. $\{P\}\, x := e_1\, ;\, y := e_2\, \{Q[z \mapsto e_3]\}$
   1.1. $\{P\}\, x := e_1\, \{Q[z \mapsto e_3][y \mapsto e_2]\}$,
   1.2. $\{Q[z \mapsto e_3][y \mapsto e_2]\}\, y := e_2\, \{Q[z \mapsto e_3]\}$
2. $\{Q[z \mapsto e_3]\}\, z := e_3\, \{Q\}$

In this last step we are not free to choose the precondition and thus a *side condition* must be satisfied:

$$\models P \;\rightarrow\; Q[z \mapsto e_3][y \mapsto e_2][x \mapsto e_1]$$

Exercise

Use the weakest precondition strategy to verify Factorial