Software Verification

Partial and Total Correctness (II)

Ahmed Rezine

IDA, Linköpings Universitet

Spring 2021

Outline

Outline

Function Specifications and Correctness

- Contract between the caller and the implementation. **Total Correctness** requires that:
  - if the pre-condition (`0 < x && 0 < y`) holds
  - then the implementation terminates,
  - after termination, the following post-condition holds (`x < sum && y < sum`)
- **Partial Correctness** does not require termination

```
method add(x: int, y: int) returns (sum: int)
    requires 0 < x && 0 < y
    ensures  x < sum && y < sum
{
    sum := x + y;
}
```

Hoare Triples and Partial Correctness

- a Hoare triple $\{P\}$ *stmt* $\{R\}$ consists in:
  - a predicate pre-condition P
  - an program *stmt*,
  - a predicate post-condition R
- intuitively, $\{P\}$ *stmt* $\{R\}$ holds if whenever P holds and *stmt* is executed and terminates (**partial correctness**), then R holds after *stmt* terminates.
- examples:
  - $\{\mathit{true}\}\ x := y\ \{x = y\}$
  - $\{x = 1 \wedge y = 2\}\ x := y\ \{x = 2\}$
  - $\{x \geq 1\}\ y := 2\ \{x = 0 \vee y \leq 10\}$
  - $\{x \geq 1\}\ \text{if}(y = 2)\ \text{then}\ x := 0\ \{x \geq 0\}$
  - $\{\mathit{false}\}\ x := 1\ \{x = 2\}$
  - $\{\mathit{true}\}\ \texttt{abort}\ \{\mathit{false}\}$
  - $\{x = 10\}\ \texttt{skip}\ \{x = 10\}$

Hoare Triples and Partial Correctness

$$\frac{}{\{\mathit{true}\}\ \texttt{abort}\ \{\mathit{false}\}} \qquad \frac{}{\{P\}\ \texttt{skip}\ \{P\}}$$

$$\frac{P' \implies P \quad \{P\}\ \mathit{stmt}\ \{Q\} \quad Q \implies Q'}{\{P'\}\ \mathit{stmt}\ \{Q'\}} \qquad \frac{\{P\}\ \mathit{stmt}\ \{Q\} \qquad \{Q\}\ \mathit{stmt}'\ \{R\}}{\{P\}\ \mathit{stmt}; \mathit{stmt}'\ \{R\}}$$

$$\frac{\{P \wedge B\}\ \mathit{stmt}\ \{Q\} \qquad \{P \wedge \neg B\}\ \mathit{stmt}'\ \{Q\}}{\{P\}\ \textbf{if}\ B\ \textbf{then}\ \mathit{stmt}\ \textbf{else}\ \mathit{stmt}'\ \{Q\}} \qquad \frac{}{\{P[e/x]\}\ x := e\ \{P\}}$$

$$\frac{\{P \wedge B\}\ \mathit{stmt}\ \{P\}}{\{P\}\ (\textbf{while}\ (B)\ \{\mathit{stmt}\})\ \{P \wedge \neg B\}}$$

Outline

Weakest precondition

- if $\{P\}$ *stmt* $\{R\}$ and $P' \Rightarrow P$ for any P' s.t. $\{P'\}$ *stmt* $\{R\}$, then P is the **weakest precondition** of R wrt. *stmt*, written $\mathbf{wp}\,(stmt, R)$. It is unique.
- $\mathbf{wp}\,(stmt, R)$ transforms predicate R wrt. *stmt*. It is said to be a **predicate transformer**.
- $\mathbf{wp}\,(x := x + 1, x \geq 1) = (x \geq 0)$. Observe $(x \geq 5)$, $(x = 6)$, $(x \geq 0 \wedge y = 8)$ are all valid preconditions, but they are not weaker than $x \geq 0$.
- Intuitively $\mathbf{wp}\,(stmt, R)$ is the weakest predicate P for which $\{P\}$ *stmt* $\{R\}$ holds

Weakest precondition of assignments

- $\mathbf{wp}\,(x := e, R) = R[e/x]$ replaces occurrences of x in R by e.
- examples:
  - $\mathbf{wp}\,(x := 3, x = 5) = (x = 5)[x/3] = (3 = 5) = \mathit{false}$
  - $\mathbf{wp}\,(x := 3, x \geq 0) = (x \geq 0)[x/3] = (3 \geq 0) = \mathit{true}$
  - $\mathbf{wp}\,(x := y + 5, x \geq 0) = (x \geq 0)[x/y + 5] = (y + 5 \geq 0)$
  - $\mathbf{wp}\,(x := 5 * y + 2 * z, x + y \geq 0) = (x + y \geq 0)[x/5 * y + 2 * z] = (6 * y + 2 * z \geq 0)$

Weakest precondition of sequences

- Assume a sequence of two instructions $stmt; stmt'$, for example $x := 2 * y; y := x + 3 * y;$
- the weakest precondition is given by:
  $\mathbf{wp}\,(stmt; stmt', R) = \mathbf{wp}\,(stmt, \mathbf{wp}\,(stmt', R))$,

$$
\begin{aligned}
& \mathbf{wp}\,(x = 2 * y; y = x + 3 * y, y > 10) \\
= \;& \mathbf{wp}\,(x = 2 * y, \mathbf{wp}\,(y = x + 3 * y, y > 10)) \\
= \;& \mathbf{wp}\,(x = 2 * y, (y > 10)[y/x + 3 * y]) \\
= \;& \mathbf{wp}\,(x = 2 * y, x + 3 * y > 10) \\
= \;& (x + 3 * y > 10)[x/2 * y] \\
= \;& (2 * y + 3 * y > 10) \\
= \;& y > 2
\end{aligned}
$$

Weakest precondition of conditionals

- Assume a conditional $(\text{if}(B)\ \text{then}\ stmt\ \text{else}\ stmt')$, for example $(\text{if}(x > y)\ \text{then}\ z := x\ \text{else}\ z := y)$
- The weakest precondition is given by:

$$\left(\begin{array}{rl} & \mathbf{wp}\left((\text{if}(B)\ \text{then}\ stmt\ \text{else}\ stmt'), R\right) \\ = & (B \Rightarrow \mathbf{wp}\left(stmt, R\right)) \wedge (\neg B \Rightarrow \mathbf{wp}\left(stmt', R\right)) \end{array}\right)$$

- For example,

$$\begin{array}{rl} & \mathbf{wp}\left((\text{if}(x > y)\ \text{then}\ z := x\ \text{else}\ z := y), z \leq 10\right) \\ = & (x > y \Rightarrow \mathbf{wp}\left(z := x, z \leq 10\right)) \wedge (x \leq y \Rightarrow \mathbf{wp}\left(z := y, z \leq 10\right) \\ = & (x > y \Rightarrow x \leq 10) \wedge (x \leq y \Rightarrow y \leq 10) \end{array}$$

- More general:

$$\mathbf{wp}\left(\left(\begin{array}{llll} \mathbf{if} & B_1 & \rightarrow & stmt_1 \\ \square & B_2 & \rightarrow & stmt_2 \\ \mathbf{fi} & & & \end{array}\right), R\right)$$

$$=$$

$$(B_1 \vee B_2) \wedge (B_1 \Rightarrow \mathbf{wp}\left(stmt, R\right)) \wedge (B_2 \Rightarrow \mathbf{wp}\left(stmt', R\right))$$

Outline

Hoare Triples for Loops, Partial Correctness

- In order to establish $\{P\}$ $(\texttt{while}(B)\texttt{do}\{stmt\})$ $\{R\}$, you will need to find an invariant Inv such that:
  - $P \Rightarrow Inv$
  - $\{Inv \wedge B\}$ $stmt$ $\{Inv\}$
  - $(Inv \wedge \neg B) \Rightarrow R$

Hoare Triples for Loops, Example

Show:

$$
\begin{array}{l}
\{Q : true\} \\
i := 0; \\
j := 0; \\
\{P : i = j = 0\} \\
\textbf{while}(i < 10)\textbf{do} \\
\{ \\
\quad i := i + 1; \\
\quad j := j + 1; \\
\} \\
\{R : j = 10\}
\end{array}
$$

- First, we show $\{Q : true\} i := 0; j := 0 \{P : i = j = 0\}$
- Then for the loop:
  1. $P \Rightarrow Inv$
  2. $\{Inv \wedge B\}\ stmt\ \{Inv\}$
  3. $(Inv \wedge \neg B) \Rightarrow R$

Hoare Triples for Loops, Example

- To show $\{Q : true\} i := 0; j := 0 \{P : i = j = 0\}$:
  - compute $\mathbf{wp}\,(i := 0; j := 0, i = j = 0)$
  - show $true \implies \mathbf{wp}\,(i := 0; j := 0, i = j = 0)$
- Then for the loop you can use $Inv : i = j \wedge i \leq 10$ and prove each one of
  1. $P \Rightarrow Inv$ is shown be proving $(i = j = 0) \implies (i = j \wedge j \leq 10)$
  2. $\{Inv \wedge B\}\ stmt\ \{Inv\}$ is shown by proving
     $(i = j \wedge i \leq 10 \wedge i < 10)$ implies
     $\mathbf{wp}\,(i := i + 1; j := j + 1, i = j \leq 10)$
  3. $(Inv \wedge \neg B) \Rightarrow R$ is shown by proving that
     $(i = j \wedge j \leq 10 \wedge \neg(i < 10)) \implies (j = 10)$

Hoare Triples for Loops, Total Correctness

- $\{P\}$ $(\texttt{while}(B)\texttt{do}\{stmt\})$ $\{R\}$
- Partial correctness: if we start from P and $(\texttt{while}(B)\texttt{do}\{stmt\})$ terminates, then R terminates.
  - $P \Rightarrow Inv$
  - $\{Inv \wedge B\}$ $stmt$ $\{Inv\}$
  - $(Inv \wedge \neg B) \Rightarrow R$
- Total correctness: the loop does terminate: find a **variant function** v such that:
  - $(Inv \wedge B) \Rightarrow (v > 0)$
  - $\{Inv \wedge B \wedge v = v_0\}$ $stmt$ $\{v < v_0\}$

Hoare Triples for Loops, Example

Show termination of the loop:

$$
\begin{array}{l}
\{Q : \mathit{true}\} \\
i := 0; j := 0; \\
\{P : i = j = 0\} \\
\textbf{while}(i < 10)\textbf{do}\{ \\
\quad i := i + 1; \\
\quad j := j + 1; \\
\} \\
\{R : j = 10\}
\end{array}
$$

- we can use the invariant used for the first three rules (partial correctness) $(i = j \leq 10)$ and the variant $(v = 10 - i)$
- $(\mathit{Inv} \wedge B) \Rightarrow (v > 0)$ is established by proving $(i = j \leq 10 \wedge i < 10)$ implies $10 - i > 0$
- $\{\mathit{Inv} \wedge B \wedge v = v_0\}\ \mathit{stmt}\ \{v < v_0\}$ is shown by proving $(i = j \leq 10 \wedge i < 10 \wedge 10 - i = v0)$ implies $\mathbf{wp}\,(i := i + 1; j := j + 1, 10 - i < v_0)$

Dutch national flag

Show:

$$\{P : \forall i.0 \leq i < a.Length : (a[i] = red \vee a[i] = white \vee a[i] = blue)\}$$
$$r := 0; w := 0; b := a.Length;$$
while$(w \leq b)$**do**$\{$
  if$(a[w] = blue)$**then**$\{$
    $a[w], a[b-1] := a[b-1], a[w]; b := b-1;$
  $\}$**else if**$(a[w] = red)$**then**$\{$
    $a[w], a[r] := a[r], a[w]; w, r := w+1, r+1;$
  $\}$**else if**$(a[w] = white)$**then**$\{$
    $w := w+1;$
  $\}$
$\}$

$$\{R : \begin{pmatrix} & (\forall i.0 \leq i < r : a[i] = red) \\ \wedge & (\forall i.r \leq i < w : a[i] = white) \\ \wedge & (\forall i.w \leq i < a.Length : a[i] = blue) \end{pmatrix}$$

Examples: Invariants

Show:

$$\{Q : a \geq 0 \wedge b \geq 0\}$$
$$z := 0; x := a; y := b$$
$$\{P : (x \geq 0) \wedge (z + x * y = a * b)\}$$
$$\begin{array}{llllll} \textbf{do} & x \geq 1 & \rightarrow & \textbf{if} & odd(x) & \rightarrow & z := z + y \\ & & & \square & even(x) & \rightarrow & \texttt{skip} \\ & & & \textbf{fi}; & & & \\ & & & x := x/2; & & & \\ & & & y := 2 * y & & & \\ \textbf{od} & & & & & & \end{array}$$
$$\{R : z = a * b\}$$

Examples: Termination

Show:

$$
\begin{array}{l}
\{Q : a \geq 0 \wedge b \geq 0\} \\
z := 0; x := a; y := b \\
\{invP : (x \geq 0) \wedge (z + x * y = a * b)\} \\
\{boundt : x\} \\
\textbf{do} \quad x \geq 1 \quad \rightarrow \quad \textbf{if} \quad odd(x) \quad \rightarrow \quad z := z + y \\
\qquad\qquad\qquad\qquad\quad \Box \quad even(x) \quad \rightarrow \quad \texttt{skip} \\
\qquad\qquad\qquad\qquad\quad \textbf{fi}; \\
\qquad\qquad\qquad\qquad\quad x := x/2; \\
\qquad\qquad\qquad\qquad\quad y := 2 * y \\
\textbf{od} \\
\{R : z = a * b\}
\end{array}
$$

Outline

Further readings

A. R. Bradley and Z. Manna.
(chap 5-6) The calculus of computation: decision procedures with applications to verification.
Springer Science & Business Media, 2007.

R. Leino. et al. `https://github.com/dafny-lang/dafny`.
Dafny. Dafny is a verification-aware programming language., Accessed December 4, 2020.

K. R. M. Leino.
Dafny: An automatic program verifier for functional correctness.
In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 348–370. Springer, 2010.