08—Program Verification II

CS 5209: Foundation in Logic and AI

Martin Henz and Aquinas Hobor

March 11, 2010

Generated on Thursday 11th March, 2010, 16:11
1. Review

2. Hoare Triples; Partial and Total Correctness

3. Practical Aspects of Correctness Proofs

4. Correctness of the Factorial Function

5. Proof Calculus for Total Correctness
1 Review

2 Hoare Triples; Partial and Total Correctness

3 Practical Aspects of Correctness Proofs

4 Correctness of the Factorial Function

5 Proof Calculus for Total Correctness
Expressions in Core Language

Expressions come as arithmetic expressions $E$:

$$E ::= n \mid x \mid (\neg E) \mid (E + E) \mid (E - E) \mid (E \times E)$$

and boolean expressions $B$:

$$B ::= \text{true} \mid \text{false} \mid (!B) \mid (B \& B) \mid (B \| B) \mid (E < E)$$

Where are the other comparisons, for example $==$?
Commands in Core Language

Commands cover some common programming idioms. Expressions are components of commands.

\[ C ::= x = E \mid C; C \mid \text{if } B \{ C \} \text{ else } \{ C \} \mid \text{while } B \{ C \} \]
Example

Consider the factorial function:

\[
0! \overset{\text{def}}{=} 1 \\
(n + 1)! \overset{\text{def}}{=} (n + 1) \cdot n!
\]

We shall show that after the execution of the following Core program, we have \( y = x! \).

\begin{verbatim}
y = 1;
z = 0;
while (z != x) { z = z + 1; y = y * z; }
\end{verbatim}
1. Review

2. Hoare Triples; Partial and Total Correctness

3. Practical Aspects of Correctness Proofs

4. Correctness of the Factorial Function

5. Proof Calculus for Total Correctness
Example

```
y = 1;
z = 0;
while (z != x) {
    z = z + 1;
y = y * z;
}
```
Example

```plaintext
y = 1;
z = 0;
while (z != x) {
    z = z + 1;
y = y * z;
}
```

We need to be able to say that at the end, \( y \) is \( x! \), provided that at the beginning, we have \( x \geq 0 \).
Assertions on Programs

Shape of assertions

$(\phi) \ P \ (\psi)$

Informal meaning

If the program $P$ is run in a state that satisfies $\phi$, then the state resulting from $P$’s execution will satisfy $\psi$. 
Partial Correctness

Definition

We say that the triple \((\phi)\ P \ (\psi)\) is satisfied under partial correctness if, for all states which satisfy \(\phi\), the state resulting from \(P\)'s execution satisfies \(\psi\), provided that \(P\) terminates.

Notation

We write \(\models_{\text{par}} (\phi) \ P \ (\psi)\).
Total Correctness

Definition
We say that the triple $(\phi) \quad P \quad (\psi)$ is satisfied under total correctness if, for all states which satisfy $\phi$, $P$ is guaranteed to terminate and the resulting state satisfies $\psi$.

Notation
We write $\models_{\text{tot}} (\phi) \quad P \quad (\psi)$. 
Consider Fac1:

\[
y = 1; \\
z = 0; \\
\textbf{while} \ (z \neq x) \ \{ \ z = z + 1; \ y = y \ast z; \ \} 
\]

\[
\vdash_{\text{tot}} (x \geq 0) \ Fac1 (y = x!) \\
\not\vdash_{\text{tot}} (\top) \ Fac1 (y = x!) 
\]
Back to Factorial

Consider $\text{Fac1}$:

$y = 1$
$z = 0$

$\textbf{while} \ (z \neq x) \ \{ \ z = z + 1; \ y = y \ast z; \ \}$

- $\vdash_{\text{tot}} (x \geq 0) \ \text{Fac1} (y = x!)$
- $\vdash_{\text{par}} (\top) \ \text{Fac1} (y = x!)$
Rules for Partial Correctness

\[
(\phi) \; C_1 \; (\eta) \quad (\eta) \; C_2 \; (\psi)
\]

\[
\frac{}{\phi) \; C_1 \; ; \; C_2 \; (\psi)} \quad \text{[Composition]}
\]
Rules for Partial Correctness (continued)

\[ ([x \rightarrow E] \psi) \ x = E \ (\psi) \]
Rules for Partial Correctness (continued)

\[
(\phi \land B) \ C_1 \ (\psi) \quad (\phi \land \neg B) \ C_2 \ (\psi)
\]

\[
\frac{}{(\phi) \text{ if } B \{ \ C_1 \} \text{ else } \{ \ C_2 \} \ (\psi)} \quad \text{[If-statement]}
\]

\[
(\psi \land B) \ C \ (\psi)
\]

\[
\frac{}{(\psi) \text{ while } B \{ \ C \} \ (\psi \land \neg B)} \quad \text{[Partial-while]}
\]
Rules for Partial Correctness (continued)

\[ \vdash_{AR} \phi' \rightarrow \phi \quad \langle \phi \rangle \ C \ (\psi) \quad \vdash_{AR} \psi \rightarrow \psi' \]

\[ \frac{\vdash_{AR} \phi' \rightarrow \phi \quad \langle \phi \rangle \ C \ (\psi) \quad \vdash_{AR} \psi \rightarrow \psi'}{\langle \phi' \rangle \ C \ (\psi')} \]
Proof Tableaux

Proofs have tree shape
All rules have the structure

```
something
_____
something else
```

As a result, all proofs can be written as a tree.

Practical concern
These trees tend to be very wide when written out on paper. Thus we are using a linear format, called *proof tableaux*.
Interleave Formulas with Code

\[
\begin{align*}
\langle \phi \rangle C_1 \langle \eta \rangle & \quad \langle \eta \rangle C_2 \langle \psi \rangle \\
\hline
\langle \phi \rangle C_1 \; C_2 \langle \psi \rangle \\
\end{align*}
\]

[Composition]

Shape of rule suggests format for proof of \( C_1; C_2; \ldots; C_n \):
\[
\begin{align*}
\langle \phi_0 \rangle \\
C_1; \quad \text{justification} \\
\langle \phi_1 \rangle \\
C_2; \quad \text{justification} \\
\vdots \\
\langle \phi_{n-1} \rangle \\
C_n; \quad \text{justification} \\
\langle \phi_n \rangle \\
\end{align*}
\]
Working Backwards

Overall goal
Find a proof that at the end of executing a program $P$, some condition $\psi$ holds.

Common situation
If $P$ has the shape $C_1; \ldots; C_n$, we need to find the weakest formula $\psi'$ such that

$$(\psi') \ C_n \ (\psi)$$

Terminology
The weakest formula $\psi'$ is called *weakest precondition*. 
Example

\( (y < 3) \)
\( (y + 1 < 4) \) Implied
\( y = y + 1; \)
\( (y < 4) \) Assignment
Another Example

Can we claim $u = x + y$ after $z = x; z = z + y; u = z;$?

$(\top)$

$(x + y = x + y)$  Implied

$z = x;$

$(z + y = x + y)$  Assignment

$z = z + y;$

$(z = x + y)$  Assignment

$u = z;$

$(u = x + y)$  Assignment
An Alternative Rule for If

We have:

\[
\left( \phi \land B \right) C_1 \left( \psi \right) \quad \left( \phi \land \neg B \right) C_2 \left( \psi \right) \\
\hline
\left( \phi \right) \text{if } B \{ C_1 \} \text{ else } \{ C_2 \} \left( \psi \right)
\]

Sometimes, the following derived rule is more suitable:

\[
\left( \phi_1 \right) C_1 \left( \psi \right) \quad \left( \phi_2 \right) C_2 \left( \psi \right) \\
\hline
\left( (B \rightarrow \phi_1) \land (\neg B \rightarrow \phi_2) \right) \text{if } B \{ C_1 \} \text{ else } \{ C_2 \} \left( \psi \right)
\]
Example

Consider this implementation of Succ:

```
a = x + 1;
if (a - 1 == 0) {
y = 1;
}
else {
y = a;
}
```

Can we prove \( (\top) \) Succ \( (y = x + 1) \) ?
Another Example

: if ( a - 1 == 0 ) {
    (1 = x + 1) \text{ If-Statement 2}
    y = 1;
    (y = x + 1) \text{ Assignment}
} else {
    (a = x + 1) \text{ If-Statement 2}
    y = a;
    (y = x + 1) \text{ Assignment}
}
( y = x + 1) \text{ If-Statement 2}
Another Example

\[
(\top)
\]
\[
((x + 1 - 1 = 0 \rightarrow 1 = x + 1) \land
(\neg(x + 1 - 1 = 0) \rightarrow x + 1 = x + 1)) \quad \text{Implied}
\]
\[
a = x + 1;
\]
\[
((a - 1 = 0 \rightarrow 1 = x + 1) \land
(\neg(a - 1 = 0) \rightarrow a = x + 1)) \quad \text{Assignment}
\]
if \( (a - 1 == 0) \) {
    \[
    (1 = x + 1)
    \]
    \[
y = 1;
    \]
    \[
    (y = x + 1)
    \]
} else {
    \[
    (a = x + 1)
    \]
    \[
y = a;
    \]
    \[
    (y = x + 1)
    \]
}
Recall: Partial-while Rule

\[
\frac{(|\psi \land B|) \ C \ (|\psi|)}{(|\psi|) \ \text{while} \ B \ \{ \ C \} \ (|\psi \land \neg B|)} \quad \text{[Partial-while]}
\]
Factorial Example

We shall show that the following Core program $\text{Fac1}$ meets this specification:

$y = 1$;
$z = 0$;
while $(z \neq x)$ {
    $z = z + 1$; $y = y \times z$;
}

Thus, to show:

$(\top) \text{Fac1} (y = x!)$
Partial Correctness of \texttt{Fac1}

\begin{align*}
\textbf{:} \\
\langle y = z! \rangle \\
\text{while} \ ( z \neq x ) \ { \\
\langle y = z! \land z \neq x \rangle } \quad \text{Invariant} \\
\langle y \cdot (z + 1) = (z + 1)! \rangle \quad \text{Implied} \\
z = z + 1; \\
\langle y \cdot z = z! \rangle \quad \text{Assignment} \\
y = y \ast z; \\
\langle y = z! \rangle \quad \text{Assignment} \\
} \\
\langle y = z! \land \neg (z \neq x) \rangle \quad \text{Partial-while} \\
\langle y = x! \rangle \quad \text{Implied}
Partial Correctness of \texttt{Fac1}

\[
\begin{align*}
&((1 = 0!)) \quad \text{Implied} \\
y = 1; \quad \text{Assignment} \\
&(|y = 0!|) \\
z = 0; \quad \text{Assignment} \\
&(|y = z!|) \\
\text{while ( } z \neq x \text{ )} \lbrace \\
\quad : \\
\rbrace \\
&(|y = z! \land \neg(z \neq x)|) \quad \text{Partial-while} \\
&(|y = x!|) \quad \text{Implied}
\end{align*}
\]
Review
Hoare Triples; Partial and Total Correctness
Practical Aspects of Correctness Proofs
Correctness of the Factorial Function
Proof Calculus for Total Correctness
Ideas for Total Correctness

- The only source of non-termination is the `while` command.
- If we can show that the value of an integer expression decreases in each iteration, but never becomes negative, we have proven termination.
  Why? Well-foundedness of natural numbers
- We shall include this argument in a new version of the `while` rule.
Rules for Partial Correctness (continued)

\[
\begin{align*}
(\psi \land B) & \quad C \quad (\psi) \\
\hline
(\psi) \text{ while } B \{ C \} \quad (\psi \land \neg B)
\end{align*}
\]

\[
\begin{align*}
(\psi \land B \land 0 \leq E = E_0) & \quad C \quad (\psi \land 0 \leq E < E_0) \\
\hline
(\psi \land 0 \leq E) \text{ while } B \{ C \} \quad (\psi \land \neg B)
\end{align*}
\]
Factorial Example (Again!)

\[ y = 1; \]
\[ z = 0; \]
\[ \textbf{while} \ (z \neq x) \{ \ z = z + 1; \ y = y \ast z; \ \} \]

What could be a good variant \( E \)?
Factorial Example (Again!)

\begin{verbatim}
y = 1;
z = 0;
while (z != x) {
z = z + 1; y = y * z;
}
\end{verbatim}

What could be a good variant $E$?

$E$ must strictly decrease in the loop, but not become negative.
Factorial Example (Again!)

\[
y = 1; \\
z = 0; \\
\textbf{while} \ (z != x) \ \{ \ z = z + 1; \ y = y \ast z; \ \} \\
\]

What could be a good variant \( E \)?

\( E \) must strictly decrease in the loop, but not become negative.

Answer:

\[ x - z \]
Total Correctness of Fac1

:  
(\(y = z! \land 0 \leq x - z\))
while ( z != x ) {
  (\(y = z! \land z \neq x \land 0 \leq x - z = E_0\)) Invariant
  (\(y \cdot (z + 1) = (z + 1)! \land 0 \leq x - (z + 1) < E_0\)) Implied
  z = z + 1;
  (\(y \cdot z = z! \land 0 \leq x - z < E_0\)) Assignment
  y = y \ast z;
  (\(y = z! \land 0 \leq x - z < E_0\)) Assignment
}
(\(y = z! \land \neg(z \neq x)\)) Total-while
(\(y = x!\)) Implied
Total Correctness of \texttt{Fac1}

\[
\begin{align*}
\{x \leq 0\} & \quad \text{Implied} \\
\{ (1 = 0! \land 0 \leq x - 0) \} & \quad \text{Implied} \\
y = 1; & \\
\{ y = 0! \land 0 \leq x - 0 \} & \quad \text{Assignment} \\
z = 0; & \\
\{ y = z! \land 0 \leq x - z \} & \quad \text{Assignment} \\
\text{while} ( z \neq x ) \{ & \\
\quad : & \\
\} & \\
\{ y = z! \land \neg(z \neq x) \} & \quad \text{Total-while} \\
\{ y = x! \} & \quad \text{Implied}
\end{align*}
\]