Program correctness and verification

Programs should be:

- clear; efficient; robust; reliable; user friendly; well documented; …
- but first of all, CORRECT
- don’t forget though: also, executable…

Correctness

Program correctness makes sense only w.r.t. a precise specification of the requirements.
Defining correctness

We need:

- A formal definition of the programs in use
  
  syntax and semantics of the programming language

- A formal definition of the specifications in use
  
  syntax and semantics of the specification formalism

- A formal definition of the notion of correctness to be used
  
  what does it mean for a program to satisfy a specification
We need:

- A formal system to prove correctness of programs w.r.t. specifications
  
  \textit{a logical calculus to prove judgments of program correctness}

- A (meta-)proof that the logic proves only true correctness judgements
  
  \textit{soundness of the logical calculus}

- A (meta-)proof that the logic proves all true correctness judgements
  
  \textit{completeness of the logical calculus}

under acceptable technical conditions
If we start with a non-negative $n$, and execute the program successfully, then we end up with $rt$ holding the integer square root of $n$. 

A specified program

\[
\{ n \geq 0 \}
\]

\[
rt := 0; sqr := 1;
\]

while $sqr \leq n$ do

\[
(rt := rt + 1; sqr := sqr + 2 \times rt + 1)
\]

\[
\{ rt^2 \leq n < (rt + 1)^2 \} \]
Hoare’s logic

Correctness judgements:

\[
\{ \varphi \} S \{ \psi \}
\]

- \( S \) is a statement of \textsc{Tiny}
- the \textit{precondition} \( \varphi \) and the \textit{postcondition} \( \psi \) are first-order formulae with variables in \textsc{Var}

Intended meaning:

\textit{Partial correctness:} termination not guaranteed!

\textit{Whenever the program} \( S \) \textit{starts in a state satisfying the precondition} \( \varphi \) \textit{and terminates successfully, then the final state satisfies the postcondition} \( \psi \)
Recall the simplest semantics of \textsc{Tiny}, with

\[ S : \text{Stmt} \rightarrow \text{State} \rightarrow \text{State} \]

We add now a new syntactic category:

\[ \varphi \in \text{Form} ::= b \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \neg \varphi \mid \exists x.\varphi \mid \forall x.\varphi \]

with the corresponding semantic function:

\[ \mathcal{F} : \text{Form} \rightarrow \text{State} \rightarrow \text{Bool} \]

and standard semantic clauses.

Also, the usual definitions of \textit{free variables} of a formula and \textit{substitution} of an expression for a variable...
More notation

For $\varphi \in \text{Form}$:

$$\{\varphi\} = \{s \in \text{State} \mid \mathcal{F}[\varphi] s = \text{tt}\}$$

For $S \in \text{Stmt}, A \subseteq \text{State}$:

$$A[S] = \{s \in \text{State} \mid S[S] a = s, \text{for some } a \in A\}$$
Hoare’s logic: semantics

\[ \models \{ \varphi \} S \{ \psi \} \]
iff
\[ \{ \varphi \} [S] \subseteq \{ \psi \} \]

Spelling this out:

The partial correctness judgement \( \{ \varphi \} S \{ \psi \} \) holds, written \( \models \{ \varphi \} S \{ \psi \} \), if for all states \( s \in \text{State} \)

if \( \mathcal{F}[\varphi] s = \text{tt} \) and \( S[S] s \in \text{State} \)

then \( \mathcal{F}[\psi] (S[S] s) = \text{tt} \)
Hoare's logic: proof rules

\[
\begin{align*}
\{ \varphi[x \rightarrow e] \} & \quad x := e \{ \varphi \} \\
\{ \varphi \} & \quad S_1 \{ \theta \} \quad \{ \theta \} \quad S_2 \{ \psi \} \\
& \quad \{ \varphi \} \quad S_1; \quad S_2 \{ \psi \} \\
\{ \varphi \} & \quad S \{ \varphi \} \\
& \quad \{ \varphi \} \quad \text{while } b \text{ do } S \{ \varphi \land \neg b \} \\
\end{align*}
\]

\[
\begin{align*}
\{ \varphi \land b \} & \quad S_1 \{ \psi \} \quad \{ \varphi \land \neg b \} \quad S_2 \{ \psi \} \\
& \quad \{ \varphi \} \quad \text{if } b \text{ then } S_1 \text{ else } S_2 \{ \psi \} \\
\end{align*}
\]

\[
\begin{align*}
\varphi' \Rightarrow \varphi & \quad \{ \varphi \} \quad S \{ \psi \} \quad \psi \Rightarrow \psi' \\
& \quad \{ \varphi' \} \quad S \{ \psi' \}
\end{align*}
\]
Example of a proof

We will prove the following partial correctness judgement:

\[
\{ n \geq 0 \}
\]

\[
rt := 0;
\]

\[
sqr := 1;
\]

\[
\textbf{while} \ sqr \leq n \ \textbf{do}
\]

\[
rt := rt + 1;
\]

\[
sqr := sqr + 2 \times rt + 1
\]

\[
\{ rt^2 \leq n \land n < (rt + 1)^2 \}
\]

Consequence rule will be used implicitly to replace assertions by equivalent ones of a simpler form.
Step by step

- \( \{ n \geq 0 \} \; rt := 0 \{ n \geq 0 \land rt = 0 \} \)
- \( \{ n \geq 0 \land rt = 0 \} \; sqr := 1 \{ n \geq 0 \land rt = 0 \land sqr = 1 \} \)
- \( \{ n \geq 0 \} \; rt := 0; \; sqr := 1 \{ n \geq 0 \land rt = 0 \land sqr = 1 \} \)
- \( \{ n \geq 0 \} \; rt := 0; \; sqr := 1 \{ sqr = (rt + 1)^2 \land rt^2 \leq n \} \)

EUREKA!!!
We have just invented the \textit{loop invariant}
Loop invariant

- \{ (sqr = (rt + 1)^2 \land rt^2 \leq n) \land sqr \leq n \} \quad rt := rt + 1 \quad \{ sqr = rt^2 \land sqr \leq n \}

- \{ sqr = rt^2 \land sqr \leq n \} \quad sqr := sqr + 2 \cdot rt + 1 \quad \{ sqr = (rt + 1)^2 \land rt^2 \leq n \}

- \{(sqr = (rt + 1)^2 \land rt^2 \leq n) \land sqr \leq n \}
  \quad rt := rt + 1; \quad sqr := sqr + 2 \cdot rt + 1

- \{ sqr = (rt + 1)^2 \land rt^2 \leq n \}

  \textbf{while} \quad sqr \leq n \quad \textbf{do}

  \quad rt := rt + 1; \quad sqr := sqr + 2 \cdot rt + 1

  \{ (sqr = (rt + 1)^2 \land rt^2 \leq n) \land \neg(sqr \leq n) \}
• \{sqr = (rt + 1)^2 \land rt^2 \leq n\}

\begin{align*}
\textbf{while} \; sqr \leq n \; \textbf{do} \\
\quad rt := rt + 1; \; sqr := sqr + 2 \times rt + 1 \\
\{rt^2 \leq n \land n < (rt + 1)^2\}
\end{align*}

• \{n \geq 0\}

\begin{align*}
\textbf{rt} := 0; \; sqr := 1; \\
\textbf{while} \; sqr \leq n \; \textbf{do} \\
\quad rt := rt + 1; \; sqr := sqr + 2 \times rt + 1 \\
\{rt^2 \leq n \land n < (rt + 1)^2\}
\end{align*}
A fully specified program

\[
\{ n \geq 0 \}
\]

\[ rt := 0; \]

\[
\{ n \geq 0 \land rt = 0 \}
\]

\[ sqr := 1; \]

\[
\{ n \geq 0 \land rt = 0 \land sqr = 1 \}
\]

\begin{algorithm}
\textbf{while} \{ sqr = (rt + 1)^2 \land rt^2 \leq n \} \ sqr \leq n \textbf{ do}
  \begin{align*}
  rt &:= rt + 1; \\
  \{ sqr = rt^2 \land sqr \leq n \}
  \\
  sqr &:= sqr + 2 \ast rt + 1
  \\
  \{ rt^2 \leq n < (rt + 1)^2 \}
\end{align*}
\end{algorithm}
The first-order theory in use

In the proof above, we have used quite a number of facts concerning the underlying data type, that is, \textbf{Int} with the operations and relations built into the syntax of \textsc{Tiny}. Indeed, each use of the consequence rule requires such facts.

Define the \textit{theory} of \textbf{Int} 

\[ \mathcal{TH}(\textbf{Int}) \]

to be the set of all formulae that hold in all states.

The above proof shows:

\[ \mathcal{TH}(\textbf{Int}) \vdash \begin{cases} n \geq 0 \\ rt := 0; \ sqr := 1; \\ \textbf{while} \ sqr \leq n \ \textbf{do} \ rt := rt + 1; \ sqr := sqr + 2 \times rt + 1 \\ \{ rt^2 \leq n \land n < (rt + 1)^2 \} \end{cases} \]
Fact: Hoare’s proof calculus (given by the above rules) is sound, that is:

\[
\text{if } \mathcal{H}(\text{Int}) \vdash \{\varphi\} S \{\psi\} \text{ then } \models \{\varphi\} S \{\psi\}
\]

So, the above proof of a correctness judgement validates the following semantic fact:

\[
\begin{align*}
\{n \geq 0\} \\
rt := 0; sqr := 1; \\
\textbf{while} \; \texttt{sqr} \leq n \; \textbf{do} \; \texttt{rt} := \texttt{rt} + 1; \; \texttt{sqr} := \texttt{sqr} + 2 \times \texttt{rt} + 1 \\
\{rt^2 \leq n \land n < (rt + 1)^2\}
\end{align*}
\]
Proof

(of soundness of Hoare’s proof calculus)

By induction on the structure of the proof in Hoare’s logic:

**assignment rule:** Easy, but we need a lemma (to be proved by induction on the structure of formulae):

\[ \mathcal{F}[\varphi[x \mapsto e]] \ s = \mathcal{F}[\varphi] \ s[x \mapsto \mathcal{E}[e]] \ s \]

Then, for \( s \in \text{State} \), if \( s \in \{ \varphi[x \mapsto e] \} \) then \( S[x := e] \ s = s[x \mapsto \mathcal{E}[e]] \ s \in \{ \varphi \} \).

**skip rule:** Trivial.

**composition rule:** Assume \( \{ \varphi \} [S_1] \subseteq \{ \theta \} \) and \( \{ \theta \} [S_2] \subseteq \{ \psi \} \). Then

\[ \{ \varphi \} [S_1; S_2] = (\{ \varphi \} [S_1]) [S_2] \subseteq \{ \theta \} [S_2] \subseteq \{ \psi \} \].

**if-then-else rule:** Easy.

**consequence rule:** Again the same, given the obvious observation that \( \{ \varphi_1 \} \subseteq \{ \varphi_2 \} \)

iff \( \varphi_1 \Rightarrow \varphi_2 \in \mathcal{T}\mathcal{H}(\text{Int}) \).
Soundness of the loop rule

**Loop rule:** We need to show that the least fixed point of the operator

\[
\Phi(F') = \text{cond}(B[b], S[S]; F, id_{\text{State}})
\]

satisfies

\[
\text{fix}(\Phi)(\{\varphi\}) \subseteq \{\varphi \land \neg b\}
\]

Proceed by fixed point induction (this is an admissible property!). Suppose that \(F(\{\varphi\}) \subseteq \{\varphi \land \neg b\}\) for some \(F: \text{State} \rightarrow \text{State}\), and consider \(s \in \{\varphi\}\) with \(s' = \Phi(F)(s) \in \text{State}\). Two cases are possible:

- If \(B[b] s = \text{ff}\) then \(s' = s \in \{\varphi \land \neg b\}\).
- If \(B[b] s = \text{tt}\) then \(s' = F(S[S] s)\). We get \(s' \in \{\varphi \land \neg b\}\) by the assumption on \(F\), since \(\{\varphi \land b\} [S] \subseteq \{\varphi\}\) by the inductive hypothesis, which implies \(S[S] s \in \{\varphi\}\).

So, \(\Phi(F)(\{\varphi\}) \subseteq \{\varphi \land \neg b\}\), and the proof is completed.
Problems with completeness

- If $\mathcal{T} \subseteq \text{Form}$ is r.e. then the set of all Hoare’s triples derivable from $\mathcal{T}$ is r.e. as well.

- $\models \{\text{true}\} S \{\text{false}\}$ iff $S$ fails to terminate for all initial states.

- Since the halting problem is not decidable for $\text{TINY}$, the set of all judgements of the form $\{\text{true}\} S \{\text{false}\}$ such that $\models \{\text{true}\} S \{\text{false}\}$ is not r.e.

Nevertheless:

$$
\mathcal{TH} (\text{Int}) \vdash \{\varphi\} S \{\psi\} \quad \text{iff} \quad \models \{\varphi\} S \{\psi\}
$$