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

Proving correctness

We need:

- A formal system to prove correctness of programs w.r.t. specifications

  a logical calculus to prove judgments of program correctness

- A (meta-)proof that the logic proves only true correctness judgements

  soundness of the logical calculus

- A (meta-)proof that the logic proves all true correctness judgements

  completeness of the logical calculus

  under acceptable technical conditions

A specified program

$$\{n \geq 0\}$$

$$rt := 0; sqr := 1;$$

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

$$\quad (rt := rt + 1; sqr := sqr + 2 * rt + 1)$$

$$\{rt^2 \leq n < (rt + 1)^2\}$$

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

Hoare's logic

Correctness judgements:

$$\boxed{\{\varphi\}\, S\, \{\psi\}}$$

- S is a statement of $\textsc{Tiny}$
- the *precondition* φ and the *postcondition* ψ are first-order formulae with variables in $\mathbf{Var}$

Intended meaning:

Partial correctness:
termination not guaranteed!

Whenever the program S starts in a state satisfying the precondition φ and terminates successfully, then the final state satisfies the postcondition ψ

Formal definition

Recall the simplest semantics of TINY, with $\boxed{\mathcal{S} : \mathbf{Stmt} \to \mathbf{State} \rightharpoonup \mathbf{State}}$

We add now a new syntactic category:

$$\boxed{\varphi \in \mathbf{Form} ::= b \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \neg\varphi' \mid \exists x.\varphi' \mid \forall x.\varphi'}$$

with the corresponding semantic function:

$$\boxed{\mathcal{F} : \mathbf{Form} \to \mathbf{State} \to \mathbf{Bool}}$$

and standard semantic clauses.

Also, the usual definitions of *free variables* of a formula and *substitution* of an expression for a variable

More notation

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

$$\boxed{\{\varphi\} = \{s \in \mathbf{State} \mid \mathcal{F}[\![\varphi]\!]\ s = \mathbf{tt}\}}$$

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

$$\boxed{A\ [\![S]\!] = \{s \in \mathbf{State} \mid \mathcal{S}[\![S]\!]\ a = s, \text{for some } a \in A\}}$$

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 $s \in \mathbf{State}$

if $\mathcal{F}[\![\varphi]\!]\ s = \mathbf{tt}$ and $\mathcal{S}[\![S]\!]\ s \in \mathbf{State}$

then $\mathcal{F}[\![\psi]\!]\ (\mathcal{S}[\![S]\!]\ s) = \mathbf{tt}$

Hoare's logic: proof rules

$$\frac{}{\{\varphi[x \mapsto e]\}\ x := e\ \{\varphi\}}$$

$$\frac{}{\{\varphi\}\ \mathbf{skip}\ \{\varphi\}}$$

$$\frac{\{\varphi\}\ S_1\ \{\theta\} \qquad \{\theta\}\ S_2\ \{\psi\}}{\{\varphi\}\ S_1 ; S_2\ \{\psi\}}$$

$$\frac{\{\varphi \wedge b\}\ S_1\ \{\psi\} \qquad \{\varphi \wedge \neg b\}\ S_2\ \{\psi\}}{\{\varphi\}\ \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \{\psi\}}$$

$$\frac{\{\varphi \wedge b\}\ S\ \{\varphi\}}{\{\varphi\}\ \mathbf{while}\ b\ \mathbf{do}\ S\ \{\varphi \wedge \neg b\}}$$

$$\frac{\varphi' \Rightarrow \varphi \qquad \{\varphi\}\ S\ \{\psi\} \qquad \psi \Rightarrow \psi'}{\{\varphi'\}\ S\ \{\psi'\}}$$

Example of a proof

We will prove the following partial correctness judgement:

$$
\begin{array}{l}
\{n \geq 0\} \\
\quad rt := 0; \\
\quad sqr := 1; \\
\quad \textbf{while } sqr \leq n \textbf{ do} \\
\qquad rt := rt + 1; \\
\qquad sqr := sqr + 2 * rt + 1 \\
\{rt^2 \leq n \wedge n < (rt+1)^2\}
\end{array}
$$

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

Step by step

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

EUREKA!!!
We have just invented
the *loop invariant*

Loop invariant

- $\{(sqr = (rt+1)^2 \wedge rt^2 \leq n) \wedge sqr \leq n\}\ rt := rt + 1\ \{sqr = rt^2 \wedge sqr \leq n\}$
- $\{sqr = rt^2 \wedge sqr \leq n\}\ sqr := sqr + 2 * rt + 1\ \{sqr = (rt+1)^2 \wedge rt^2 \leq n\}$
- $\{(sqr = (rt+1)^2 \wedge rt^2 \leq n) \wedge sqr \leq n\}$

  $\quad rt := rt + 1; sqr := sqr + 2 * rt + 1$

  $\{sqr = (rt+1)^2 \wedge rt^2 \leq n\}$

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

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

  $\qquad rt := rt + 1; sqr := sqr + 2 * rt + 1$

  $\{(sqr = (rt+1)^2 \wedge rt^2 \leq n) \wedge \neg(sqr \leq n)\}$

Finishing up

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

  $\quad$ **while** $sqr \leq n$ **do**

  $\qquad rt := rt + 1;\ sqr := sqr + 2 * rt + 1$

  $\{rt^2 \leq n \wedge n < (rt+1)^2\}$

- $\boxed{\begin{array}{l} \{n \geq 0\} \\ \quad rt := 0;\ sqr := 1; \\ \quad \textbf{while}\ sqr \leq n\ \textbf{do} \\ \qquad rt := rt + 1;\ sqr := sqr + 2 * rt + 1 \\ \{rt^2 \leq n \wedge n < (rt+1)^2\} \end{array}}$

QED

A fully specified program

$\{n \geq 0\}$

$rt := 0;$

$\{n \geq 0 \wedge rt = 0\}$

$sqr := 1;$

$\{n \geq 0 \wedge rt = 0 \wedge sqr = 1\}$

while $\{sqr = (rt+1)^2 \wedge rt^2 \leq n\}$ $sqr \leq n$ **do**

$\quad rt := rt + 1;$

$\quad \{sqr = rt^2 \wedge sqr \leq n\}$

$\quad sqr := sqr + 2 * rt + 1$

$\{rt^2 \leq n < (rt+1)^2\}$

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, **Int** with the operations and relations built into the syntax of TINY. Indeed, each use of the consequence rule requires such facts.

Define the *theory* of **Int**

$$\boxed{\mathcal{TH}(\mathbf{Int})}$$

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

The above proof shows:

$$\mathcal{TH}(\mathbf{Int}) \vdash \boxed{\begin{array}{l} \{n \geq 0\} \\ \quad rt := 0;\ sqr := 1; \\ \quad \mathbf{while}\ sqr \leq n\ \mathbf{do}\ rt := rt + 1;\ sqr := sqr + 2 * rt + 1 \\ \{rt^2 \leq n \wedge n < (rt+1)^2\} \end{array}}$$

Soundness

Fact: *Hoare's proof calculus (given by the above rules) is sound, that is:*

$$\boxed{\textit{if}\ \boxed{\mathcal{TH}(\mathbf{Int}) \vdash \{\varphi\}\, S\, \{\psi\}}\ \textit{then}\ \boxed{\models \{\varphi\}\, S\, \{\psi\}}}$$

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

$$\models \boxed{\begin{array}{l} \{n \geq 0\} \\ \quad rt := 0;\ sqr := 1; \\ \quad \mathbf{while}\ sqr \leq n\ \mathbf{do}\ rt := rt + 1;\ sqr := sqr + 2 * rt + 1 \\ \{rt^2 \leq n \wedge n < (rt+1)^2\} \end{array}}$$

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 $s \in \mathbf{State}$, if $s \in \{\varphi[x \mapsto e]\}$ then $\mathcal{S}[\![x := e]\!]\ s = s[x \mapsto \mathcal{E}[\![e]\!]\ s] \in \{\varphi\}$.

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 $\varphi_1 \Rightarrow \varphi_2 \in \mathcal{TH}(\mathbf{Int})$.

Soundness of the loop rule

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

$$\Phi(F) = cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S]\!];F, id_{\mathbf{State}})$$

satisfies

$$\mathit{fix}(\Phi)(\{\varphi\}) \subseteq \{\varphi \wedge \neg b\}$$

Proceed by *fixed point induction*. Suppose that $F(\{\varphi\}) \subseteq \{\varphi \wedge \neg b\}$ for some $F : \mathbf{State} \rightharpoonup \mathbf{State}$, and consider $s \in \{\varphi\}$ with $s' = \Phi(F)(s) \in \mathbf{State}$. Two cases are possible:

- If $\mathcal{B}[\![b]\!]\ s = \mathbf{ff}$ then $s' = s \in \{\varphi \wedge \neg b\}$.
- If $\mathcal{B}[\![b]\!]\ s = \mathbf{tt}$ then $s' = F(\mathcal{S}[\![S]\!]\ s)$. We get $s' \in \{\varphi \wedge \neg b\}$ by the assumption on F, since $\{\varphi \wedge b\}\ [\![S]\!] \subseteq \{\varphi\}$ by the assumption on S, which implies $\mathcal{S}[\![S]\!]\ s \in \{\varphi\}$.

So, $\Phi(F)(\{\varphi\}) \subseteq \{\varphi \wedge \neg b\}$, and the proof is completed.

Problems with completeness

- If $\mathcal{T} \subseteq \mathbf{Form}$ is r.e. then the set of all Hoare's triples derivable from $\mathcal{T}$ is r.e. as well.
- $\models \{\mathbf{true}\}\, S\, \{\mathbf{false}\}$ iff S loops for all initial states.
- Since the halting problem is not decidable for TINY, the set of all judgements of the form $\{\mathbf{true}\}\, S\, \{\mathbf{false}\}$ such that $\models \{\mathbf{true}\}\, S\, \{\mathbf{false}\}$ is not r.e.

Nevertheless:

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