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

History:

- Turing 1949
- 1960's:
  McCarthy, Naur, Floyd
- Hoare 1969
- many others to follow
  (see: Apt 1981)

Correctness judgements:

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

- S is a statement of TINY
- the *precondition* φ and the *postcondition* ψ are first-order formulae with variables in **Var**

Partial correctness:
termination not guaranteed!

Intended meaning:

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

Formal definition

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

We add now a new syntactic category:

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

with the corresponding semantic function:

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

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

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

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

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

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:

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

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

Step by step

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

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

an instance of the assignment rule:

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

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

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

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

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

BTW: another version of the assignment rule:

$$\frac{}{\{\varphi\}\ x := e\ \{\exists x'.(\varphi[x \mapsto x'] \wedge x = e[x \mapsto x'])\}}$$

Loop invariant

an instance of the assignment rule:

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

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

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

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

Finishing up

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

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

QED

A fully specified program

Practical representation of a complete proof tree

$\{n \geq 0\}$

$rt := 0;$

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

$sqr := 1;$

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

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**

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

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

The above proof shows:

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

Soundness

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

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

Proof: *in due course...*

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

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

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 fails to terminate 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:

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