Specification as a development task

Given precondition $\varphi$ and postcondition $\psi$

Develop a program $S$ such that

$$\{\varphi\} S \{\psi\}$$
For instance

Find $S$ such that

$$\{n \geq 0\} S \{rt^2 \leq n \land n < (rt + 1)^2\}$$

One correct solution:

$$\{n \geq 0\}
\begin{align*}
rt &:= 0; \text{ sqr } := 1; \\
\textbf{while} \text{ sqr } \leq n \textbf{ do} \; &rt := rt + 1; \text{ sqr } := \text{ sqr } + 2 \ast rt + 1 \\
\{rt^2 \leq n \land n < (rt + 1)^2\}
\end{align*}$$
Another correct solution:

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

while true do skip

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

since \[ \vdash \]

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

while \{ true \} true do skip

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

**Partial correctness**: termination not guaranteed, and hence **not requested**!
Total correctness

\[ \text{Total correctness} = \text{partial correctness} + \text{successful termination} \]

Total correctness judgements:

\[ [\varphi] S [\psi] \]

Intended meaning:

\[ \text{Whenever the program } S \text{ starts in a state satisfying the precondition } \varphi \]
\[ \text{then it terminates successfully in a final state that satisfies the postcondition } \psi \]
Total correctness: semantics

\[ \models [\varphi] S [\psi] \]

iff

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

where for \( S \in \text{Stmt}, \ A \subseteq \text{State} \):

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

{Spelling this out:}

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

\[ \text{if } \mathcal{F}[\varphi] s = \text{tt} \text{ then } S[S] s \in \text{State and } \mathcal{F}[\psi] (S[S] s) = \text{tt} \]
### Total correctness: proof rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( [\varphi[x \mapsto e]] x := e \varphi )</td>
<td>Assignment</td>
</tr>
<tr>
<td>( [\varphi] \text{skip} [\varphi] )</td>
<td>Skip statement</td>
</tr>
<tr>
<td>( [\varphi] S_1 [\theta] \quad \theta S_2 [\psi] )</td>
<td>Sequence of statements</td>
</tr>
<tr>
<td>( [\varphi] \text{if } b \text{ then } S_1 \text{ else } S_2 [\psi] )</td>
<td>Conditional statement</td>
</tr>
<tr>
<td>( ???? ) while ( b ) do ( S ) [????]</td>
<td>Loop statement</td>
</tr>
</tbody>
</table>

Adjustments are necessary if expressions may generate errors!
Total-correctness rule for loops

\[
\begin{align*}
(nat(l) \land \varphi(l + 1)) & \Rightarrow b & \quad [nat(l) \land \varphi(l + 1)] \; S \; [\varphi(l)] & \quad \varphi(0) \Rightarrow \neg b \\
\exists l. \; nat(l) \land \varphi(l) & \quad \textbf{while } b \; \textbf{do } S \; [\varphi(0)]
\end{align*}
\]

where

- \( \varphi(l) \) is a formula with a free variable \( l \) that does not occur in \textbf{while } b \; \textbf{do } S,
- \( nat(l) \) stands for \( 0 \leq l \), and
- \( \varphi(l + 1) \) and \( \varphi(0) \) result by substituting, respectively, \( l + 1 \) and \( 0 \) for \( l \) in \( \varphi(l) \).

Informally: \( l \) is a \emph{counter} that indicates the number of iterations of the loop body.
Soundness

(of the proof rules for total correctness for the statements of Tiny)

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

**Proof:** By induction on the structure of the proof tree: all the cases are as for partial correctness, except for the rule for loops.

**loop rule:** Consider \( s \in \{ \text{nat}(l) \land \varphi(l) \} \). By induction on \( s(l) \) (which is a natural number) show that \( S[\text{while } b \text{ do } S] s = s' \) for some \( s' \in \{ \varphi(0) \} \) (easy!). To complete the proof, notice that if a variable \( x \) does not occur in a statement \( S' \in \text{Stmt} \) and two states differ at most on \( x \), then whenever \( S' \) terminates successfully starting in one of them, then so it does starting in the other, and the result states differ at most on \( x \).
Completeness

(of the proof system for total correctness for the statements of Tiny)

It so happens that:

\( \mathcal{TH}(\text{Int}) \vdash [\varphi] S [\psi] \iff \models [\varphi] S [\psi] \)

Proof (idea): Only loops cause extra problems: here, for \( \varphi(l) \) take the conjunction of the (partial correctness) loop invariant with the formula

“the loop terminates in exactly \( l \) iterations”

It so happens that the latter can indeed be expressed here (since finite tuples of integers and their finite sequences can be coded as natural numbers)!
For example

To prove:

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

use the following invariant with the iteration counter \( l \):

\[
sqr = (rt + 1)^2 \land rt^2 \leq n \land l = \lfloor \sqrt{n} \rfloor - rt
\]

Cheating here, of course:

"\( l = \lfloor \sqrt{n} \rfloor - rt \)" has to be captured by a first-order formula in the language of TINY.

Luckily: this can be done!

Here, this is quite easy:

\[
(rt + l)^2 \leq n < (rt + l + 1)^2
\]
Well-founded relations

A relation \( R \subseteq W \times W \) is well-founded if there is no infinite chain

\[
a_0 \succ a_1 \succ \ldots \succ a_i \succ a_{i+1} \succ \ldots
\]

Typical example:

\[ \langle \text{Nat}, \succ \rangle \]

BTW: For well-founded \( \succ \subseteq W \times W \), its transitive and reflexive closure \( \succ^* \subseteq W \times W \) is a partial order on \( W \).

BUT: subtracting identity from an arbitrary partial order on \( W \) need not in general yield a well-founded relation.

Few other examples:

- \( \text{Nat}^n \) with component-wise (strict) ordering;
- \( A^* \) with proper prefix ordering;
- \( \text{Nat}^n \) with lexicographic (strict) ordering generated by the usual ordering on \( \text{Nat} \);
- any ordinal with the natural (strict) ordering; etc.
Total correctness = partial correctness + successful termination

Proof method

To prove

\[
\text{while } b \text{ do } S [\varphi \land \neg b]
\]

- show “partial correctness”: \([\varphi \land b] S [\varphi]\)

- show “termination”: find a set \(W\) with a well-founded relation \(\succ \subseteq W \times W\) and a function \(w: State \rightarrow W\) such that for all states \(s \in \{\varphi \land b\}\),

\[
w(s) \succ w(S[S] s)
\]

BTW: \(w: State \rightarrow W\) may be partial as long as it is defined on \(\{\varphi\}\).
Example

Prove:

\[x \geq 0 \land y \geq 0\]

while \(x > 0\)
do

\[
\begin{align*}
\text{if } y > 0 & \text{ then } y := y - 1 \text{ else } (x := x - 1; y := f(x)) \\
\text{[true]}
\end{align*}
\]

where \(f\) yields a natural number for any natural argument.

- If one knows nothing more about \(f\), then the previous proof rule for the total correctness of loops is useless here.

- **BUT:** termination can be proved easily using the function \(w: \text{State} \rightarrow \text{Nat} \times \text{Nat}\), where \(w(s) = \langle s \, x, s \, y \rangle\):

  after each iteration of the loop body the value of \(w\) decreases w.r.t. the (well-founded) lexicographic order on pairs of natural numbers.
A fully specified program

\[
\begin{align*}
[x \geq 0 \land y \geq 0] \\
\text{while } [x \geq 0 \land y \geq 0] \ x > 0 \text{ do decren } \langle x, y \rangle \text{ in } \textbf{Nat} \times \textbf{Nat} \text{ wrt } \\
\quad \text{if } y > 0 \text{ then } y := y - 1 \text{ else } (x := x - 1; y := f(x)) \\
[\text{true}]
\end{align*}
\]

... with various notational variants assuming some external definitions for the well-founded set and function into it
Hoare's logic: trouble #2

Find $S$ such that

$$\{n \geq 0\} S \{rt^2 \leq n \land n < (rt + 1)^2\}$$

Another correct solution:

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

OOOOPS?!

A number of techniques to avoid this:

- variables that are required not to be used in the program;
- binary postconditions;
- various forms of algorithmic/dynamic logic, with program modalities.
• New syntactic category **BForm** of *binary formulae*, which are like the usual formulae, except they can use both the usual variables $x \in \text{Var}$ and their “past” copies $\hat{x} \in \hat{\text{Var}}$.

For any syntactic item $\omega$, we write $\hat{\omega}$ for $\omega$ with each variable $x$ replaced by $\hat{x}$.

• Semantic function: $\mathcal{BF} : \text{BForm} \to \text{State} \times \text{State} \to \text{Bool}$

$\mathcal{BF}[\psi](s_0, s)$ is defined as usual, except that the state $s_0$ is used to evaluate “past” variables $\hat{x} \in \hat{\text{Var}}$ and $s$ is used to evaluate the usual variables $x \in \text{Var}$. 
Correctness judgements

\[ \text{pre } \varphi; \ S \ \text{post } \psi \]

where \( \varphi \in \text{Form} \) is a (unary) precondition; \( S \in \text{Stmt} \) is a statement (as usual); and \( \psi \in \text{BForm} \) is a binary postcondition.

**Semantics:**

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

\[
\begin{align*}
\text{if } & \mathcal{F}[\varphi] \ s = \text{tt} \ \text{then } S[S] \ s \in \text{State} \ \text{and } B\mathcal{F}[\psi] \langle s, S[S] \ s \rangle = \text{tt}
\end{align*}
\]
Proof rules

\[
\text{pre } \varphi; \ x := e \ \text{post} (\hat{\varphi} \land x = \hat{e} \land \vec{y} = \hat{\vec{y}})
\]

where \(\vec{y}\) are variables other than \(x\).

\[
\text{pre } \varphi; \ \text{skip} \ \text{post} (\varphi \land \vec{y} = \hat{\vec{y}})
\]

\[
\begin{align*}
\text{pre } \varphi_1; \ S_1 \ \text{post} (\psi_1 \land \varphi_2) & \quad \text{pre } \varphi_2; \ S_2 \ \text{post} \psi_2 \\
\text{pre } \varphi_1; \ S_1; S_2 \ \text{post} \psi_1 \ast \psi_2
\end{align*}
\]

where \(\psi_1 \ast \psi_2\) is \(\exists \vec{z}. (\psi_1[\vec{x} \mapsto \vec{z}] \land \psi_2[\hat{\vec{x}} \mapsto \vec{z}])\), with all the variables free in \(\psi_1\) or \(\psi_2\) are among \(\vec{x}\) or \(\hat{\vec{x}}\), and \(\vec{z}\) are new variables.
Further rules

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

\[
\begin{align*}
&\text{pre } \varphi \land b; \; S \; \text{post } (\psi \land \hat{e} \succeq e) \quad \psi \Rightarrow \varphi \quad (\psi \ast \psi) \Rightarrow \psi \\
&\text{pre } \varphi; \; \text{while } b \; \text{do } S \; \text{post } ((\psi \lor (\varphi \land \vec{y} = \hat{y})) \land \neg b)
\end{align*}
\]

where $\succeq$ is well-founded, and all the free variables are among $\vec{y}$ or $\hat{y}$.

\[
\begin{align*}
&\varphi' \Rightarrow \varphi \quad \text{pre } \varphi; \; S \; \text{post } \psi \quad \psi \Rightarrow \psi' \\
&\text{pre } \varphi'; \; S \; \text{post } \psi' \\
&\text{pre } \varphi; \; S \; \text{post } (\hat{\varphi} \land \psi)
\end{align*}
\]

The rules can (have to?) be polished...
We have now:

\[
\begin{align*}
\text{pre } n & \geq 0; \\
rt & := 0; \text{ sqr } := 1; \\
\textbf{while } s & qr \leq n \textbf{ do } rt := rt + 1; \text{ sqr } := s & qr + 2 * rt + 1 \\
\text{post } rt^2 & \leq \hat{n} \land \hat{n} < (rt + 1)^2
\end{align*}
\]

\textbf{BUT} : \quad \not\models \quad \begin{align*}
\{ n & \geq 0 \} \\
rt & := 0; \quad n := 0 \\
\{ rt^2 & \leq \hat{n} \land \hat{n} < (rt + 1)^2 \}
\end{align*}
Algorithmic/dynamic logic

**Sketch**

**Overall idea:**

*Extend the logical formulae so that they are closed under the usual logical connectives and quantification, as well as under program modalities*

**Syntax:** For any formula $\varphi$ and a statement $S \in Stmt$, build a new formula:

$\langle S \rangle \varphi$

**Semantics:**

$\mathcal{F}[\langle S \rangle \varphi] s = \begin{cases} 
\mathcal{F}[\varphi] s' & \text{if } S[S] s = s' \in State \\
\text{ff} & \text{if } S[S] s \notin State
\end{cases}$
Proof system

...axioms and rules to handle the standard connectives and quantification...

Plus axioms and rules to deal with program modalities — interaction between modalities and propositional connectives; (de)composition of modalities — for instance:

\[
\langle S \rangle (\varphi \land \psi) \iff (\langle S \rangle \varphi \land \langle S \rangle \psi)
\]

\[
\langle S \rangle \neg \varphi \implies \neg \langle S \rangle \varphi
\]

\[
\langle S \rangle \text{true} \implies (\neg \langle S \rangle \varphi \implies \langle S \rangle \neg \varphi)
\]

\[
\langle S_1; S_2 \rangle \varphi \iff \langle S_1 \rangle (\langle S_2 \rangle \varphi)
\]

etc.

Key to the completeness results here: \textit{infinitary rules for loops}