Design Refinement & Verification

\[ f = g \rightarrow h \]

Does \( g; h \) accomplish \( f \)?

\[ f = c \rightarrow g \rightarrow h \]

If \( c \) is true does \( g \) accomplish \( f \), and if \( c \) is false does \( h \) accomplish \( f \)?

\[ f = c \rightarrow g \]

Is termination guaranteed?

If \( c \) is true does \( g; f \) accomplish \( f \), and if \( c \) is false is \( f \) a no-op ?

Diagrams above show structured programming control structures (we assume test \( c \) does not change state) and their corresponding verification conditions.\(^1\)

- Suppose functionality \( f \) is implemented as: \( g \) followed by \( h \). That implementation is correct provided:
  \[ g \text{ followed by } h \text{ transforms state the same way as } f \text{ transforms state.} \]

- Suppose functionality \( f \) is implemented as: if \( c \) then \( g \) else \( h \). That implementation is correct provided:
  
  If \( c \) is true, then \( g \) transforms state the same way as \( f \) transforms state.
  
  If \( c \) is false, then \( h \) transforms state the same way as \( f \) transforms state.

- Suppose functionality \( f \) is implemented as: while \( c \) do \( g \). That implementation is correct provided:

  If \( c \) is true, then \( g \) followed by \( f \) transforms state the same way as \( f \) transforms state.
  
  If \( c \) is false, then \( f \) does not change state.

\(^1\)The notation \( g; h \) denotes \( g \) followed by \( h \).
Consider the following example; the code is intended to transform initial state \( \langle x, y \rangle \) into final state \( \langle x, \lfloor \sqrt{x} \rfloor \rangle \). One might conjecture that loop \( f \) transforms state as follows:

- If \( y^2 \leq x \), then initial state \( \langle x, y \rangle \) is transformed into final state \( \langle x, \lfloor \sqrt{x} \rfloor \rangle \).
- If \( y^2 > x \), then initial state \( \langle x, y \rangle \) is transformed into final state \( \langle x, y \rangle \).

\[
\text{The notation } g; f \text{ denotes } g \text{ followed by } f. \]

To affirm that the conjectured way in which the loop transforms state — the first equality beneath the diagram above — represents how \( f \) actually transforms state, the following verification conditions (from the previous page) must be established:

1. If \( (y + 1)^2 \leq x \), then \( g; f \) and \( f \) transform state in the same way.
2. If \( (y + 1)^2 > x \), then \( f \) does not change state.

\[2\text{The notation } g; f \text{ denotes } g \text{ followed by } f.\]
Consider the first verification condition above: assume \((y + 1)^2 \leq x\).

If \(y^2 \leq x\) then \(f\) transforms state \(\langle x, y \rangle\) into state \(\langle x, \lfloor \sqrt{x} \rfloor \rangle\), which is identical with how \(g; f\) changes state (that is the desired conclusion).

If \(y^2 > x\) then \(f\) does not change state. However,

\[
x < y^2 < (y + 1)^2 \leq x
\]

\[
\Rightarrow x < x
\]

Thus this case cannot occur.

*This argument assumes \(y \geq 0\), because otherwise (3) is false; \(y^2 > (y + 1)^2\) when \(y\) is negative.*

Consider the second verification condition above (previous page): assume \((y + 1)^2 > x\).

If \(y^2 > x\) then \(f\) does not change state (that is the desired conclusion).

If \(y^2 \leq x\) then \(f\) transforms state \(\langle x, y \rangle\) into state \(\langle x, \lfloor \sqrt{x} \rfloor \rangle\). Moreover,

\[
y^2 \leq x < (y + 1)^2
\]

\[
\Rightarrow y \leq \sqrt{x} < y + 1
\]

\[
\Rightarrow y = \lfloor \sqrt{x} \rfloor
\]

Thus \(\langle x, y \rangle = \langle x, \lfloor \sqrt{x} \rfloor \rangle\) and \(f\) does not change state (that is the desired conclusion).

Whereas this argument may seem to assume \(y\) is non-negative, \(y < 0\) is not possible because that would contradict (4).

Although desired conclusions hold when \(y \geq 0\), the attempt to establish the required verification conditions has failed because the case \(y < 0\) has not been handled successfully. Moreover, even if that were not an issue, verification would be incomplete because loop termination has not yet been considered.

To illustrate, let \(y = -1\) and \(x = 0\). Condition \(c\) on the initial state \(\langle 0, -1 \rangle\) is true, thus \(g\) executes transforming state to \(\langle 0, 0 \rangle\). Loop \(f\) is entered again and \(c\) on state \(\langle 0, 0 \rangle\) is false, so the loop terminates and \(f\) produces final state \(\langle 0, 0 \rangle\). This contradicts the conjectured behavior

\[
f(\langle 0, -1 \rangle) = \langle 0, -1 \rangle
\]

*Homework: the example above demonstrates \(f\) is not how the loop transforms state; how does \(f\) transform state? Prove the correctness of your answer using verification conditions (include loop termination).*

---

3Whereas the initialization \(y := 0\) — it is called \(h\) on the previous page — guarantees non-negative \(y\), that is irrelevant to the current discussion which is *not* analyzing how \(h; f\) transforms state, the current discussion is instead concerned with how \(f\) transforms state.