Short Communication / Kurze Mitteilung

Comments on a Paper by Neuhold and Studer

P. Pritchard, Brisbane

Received August 20, 1978

Abstract — Zusammenfassung

Comments on a Paper by Neuhold and Studer. The authors of [5] described their experience in applying Hoare's method of program verification to an existing program. They reported two difficulties. The first was that certain Boolean expressions occurring in conditional or while statements needed to be strengthened to enable verification. This claim is false, and their three putative examples are refuted. The second difficulty was that the program had to be modified to use very restricted forms of jumps and procedures. Re this, we document recent work which enables these restrictions to be lifted.


We stick to the notation of [5] and address each of the claimed difficulties in turn.

1. Boolean Expressions

It is firstly claimed that the "Rule of Alternation"

\[ P_1 \{ Q_1 \} R \]
\[ P_2 \{ Q_2 \} R \]

\[ \text{if } B \text{ then } P_1 \text{ else } P_2 \{ \text{if } B \text{ then } Q_1 \text{ else } Q_2 \} R \]  (1)

fails to permit the deduction of the following correctness claims:

(a) \( x = a \land y = b \) \{if \( x > y \) then \( x := x + 1 \) else \( x := x - 1 \)\} \( y = b \land \text{Rel} (x, a, y) \)

where \( \text{Rel} (x, a, y) \) is the relation attaining between \( x, a \) and \( y \).
(b) \(x = a \land y = b\) \(\{\text{bool} := \text{false}\}
\)

\[
\begin{align*}
\text{if } x > y \text{ then } \text{bool} := \text{true}; \\
\text{if } \text{bool} \text{ then } x := x - y \text{ else } x := y - x \}
\end{align*}
\]

\(x \geq 0\)

Ad (a): it is granted that \(P_1 \{Q_1\} R_1\) and \(P_2 \{Q_2\} R_2\), where

\[
\begin{align*}
P_1 & \equiv (x=a \land y=b \land x>y), \\
P_2 & \equiv (x=a \land y=b \land x \leq y), \\
R_1 & \equiv (x>a \land y=b \land x>y), \\
R_2 & \equiv (x<a \land y=b \land x<y).
\end{align*}
\]

But it is then stated that the \(R\) of (1) must be \(y=b\), because "only the common elements of the different assertions \(R_1\) and \(R_2\) which are deducible from \(P_1\) and \(P_2\) can be used for \(R\)". No! \(R\) can be chosen to be \(R_1 \lor R_2\), since it is easy to show \(P_1 \{Q_1\} R_1 \lor R_2\) and \(P_2 \{Q_2\} R_1 \lor R_2\) (by the Rule of Consequence, for example). Thus (a) can be proven with \(\text{Rel}(x, a, y) \equiv (x > a \land x > y) \lor (x < a \land x < y)\).

Ad (b): it is correctly stated that the preconditions \(P_1 \equiv (x-y \geq 0)\) and \(P_2 \equiv (y-x \geq 0)\) are required for the respective assignments in the last statement; i.e. if \(\text{bool} \text{ then } x-y \geq 0 \text{ else } y-x \geq 0\) must hold before this statement. Let us continue to work backwards by using this as our postcondition \(R\) for the penultimate statement, using (1) with \(Q_2 = \text{null}\) (the "do nothing" statement). By the Axiom of Assignment we have \(x-y \geq 0 \\{\text{bool} := \text{true}\} R\), which with \(R \{\text{null}\} R\) gives the precondition if \(x>y\) then \(x-y \geq 0\) else \(y-x \geq 0\). Finally, pushing this back through the assignment \(\text{bool} := \text{false}\) gives the initial required precondition if \(x>y\) then \(x-y \geq 0\) else \(y-x \geq 0\). Clearly, this is a theorem and therefore is implied by \(x=a \land y=b\). This completes the proof of (b). The mistake in [5] was not to realize that \(\text{bool} \equiv x>y\) is available as a precondition for the final statement. Concerning both (a) and (b): no difficulties arise if one works backwards from the desired postcondition, as, indeed, the form of the rules encourages.

The other claimed difficulty with Boolean expressions concerns the "Rule of Iteration".

\[
\frac{P \land B \{Q\} P}{P \{\text{while } B \text{ do } Q\} P \land \neg B}
\]

It is wrongly asserted that "the Boolean expressions of while-loops may have to be strengthened solely for allowing the verification process to infer the required assertions". As evidence, Neuhold and Studer claim that the assertion \(m \neq a\) is not available as a precondition for the statement \(m := 1/(m-a)\) in the program part below.

\[
m := a; \\
n := b; \\
\text{bool} := \text{false}; \\
\text{while } (m < 100 \land \neg \text{bool}) \text{ do} \begin{align*}
\text{begin} \\
m := m + 1; \\
\text{if } m = n \text{ then } \text{bool} := \text{true} \end{align*} \text{ end; } \\
\text{if } \text{bool} \text{ then } m := 1/(m-a)
\]