Loop invariants

A loop in an algorithm or program is a sequence of one or more steps that is performed repeatedly as long as a specified condition is met. If \( g \) is the “guard” condition that controls entry into the loop and \( S \) is the sequence of steps, called the body, to be performed, we may represent this control structure as \( \textbf{while } g \textbf{ do } S \). Note that after the body \( S \) is performed, the guard \( g \) is checked to see if \( S \) is to be repeated. When \( g \) is not satisfied, the body \( S \) is skipped and control passes to the next step after the body.

A loop invariant is any condition \( P \) that is true \textit{before} the loop is encountered and which remains true \textit{after} the loop has been executed. It follows that the condition \( P \) will be true after each iteration (pass through) of the loop until the loop terminates with the guard \( g \) not met. The proof uses the Well-ordering Principle, that every nonempty subset of \( \mathbb{N} \) has a smallest element.

Example: Using a loop to find the power \( a^b \) for natural numbers \( a, b \), where \( a>0, b \geq 0 \).

\[
\begin{align*}
{a>0 \land b\geq0} \\
\quad i := 0; \\
\quad p := 1; \\
\textbf{while } i < b \textbf{ do} \\
\quad p := p\times a; \\
\quad i := i+1.
\end{align*}
\]

\{p = a^b\}

To show that the condition \{p = a^i \land i \leq b\} is a loop invariant, we use an informal proof:

If \( p = a^i \) before the loop, then multiplying by \( a \) produces \( p\times a = a^i \times a = a^{i+1} \). But this is the statement that the “new \( p \)” equals \( a \) to the power “new \( i \)”. A formal proof could take up to 30 steps [James Hein, Discrete Structures, Logic, and Computability, Jones & Bartlett, 1995, pp425-426].

Example: The condition: \{Product is even \text{ and sum is odd, for natural numbers } m \text{ and } n.\} is preserved by incrementing both factors by 1.

Given \( m\times n = 2p \) for some natural \( p \) and \( m+n = 2r+1 \) for some natural \( r \), then:

“new product” = \((m+1)(n+1) = m\times n + (m+n) + 1 = 2p + (2r+1) + 1 = 2q \) for some \( q \);
and “new sum” = \((m+1)+(n+1) = (m+n) + 2 = (2r+1) + 2 = 2s+1 \) for some \( s \).

Note that the preceding condition could be replaced by the “simpler” condition: \{Sum is odd\}, because the sum of two naturals being odd implies their product is even. [Can you provide a proof?]

Note also that if the condition \{...\} is not true before the loop is executed, there is no point in checking for loop invariance.

What is the invariant condition suggested by the following additions?

\[
1+3 = 2^2 \\
1+3+5=3^2 \\
1+3+5+7=4^2 \\
1+3+...+9=5^2
\]

[The sum of \( n \) odd naturals equals ...]

Is an invariant suggested by the following calculations? [Is it true for \( n=41 \)]

\[
1^2 - 1 + 41 = 41 \\
2^2 - 2 + 41 = 43 \\
3^2 - 3 + 41 = 47 \\
4^2 - 4 + 41 = 53
\]