Loop Invariants

In the while-rule

$$\frac{\vdash \{ P \wedge B \}\ S\ \{ P \}}{\vdash \{ P \}\ \textbf{while}\ B\ \textbf{do}\ S\ \{ P \wedge \neg B \}}$$

the formula P is called the **loop invariant** because each iteration of the body of the loop preserves the truth of P (assuming the loop guard B is also true). The primary challenge in proving loops is in the formulation of a suitable loop invariant.

This approach to understanding loops is *much* different from the usual repetitive execution concept. In fact, there is nothing in the while-rule that explicitly requires the loop body to be executed repeatedly! And the loop invariant focuses on what stays the same rather than what changes — it is static not dynamic.

With the loop invariant approach, it is helpful to think of a loop as producing a series of improving approximations to the desired result. Each iteration improves an approximation until the series “converges” to the desired condition and the loop terminates.

David Gries says one should *never* write a loop without writing down the loop invariant. It is often difficult to come up with a loop invariant, but this may be taken as a signal that we do not understand the loop as well as we might. Loops provide one of the most common sources of programming errors, so any aid to improving our understanding of loops should lead to programs that are more reliable.