Assignment rule:

\[
\{ A[E/V] \} \quad V = E \quad \{ A \}
\]

where \( A \) is the assignment statement that tries to assign the value of the expression \( E \) to the variable \( V \). The rule says that “the value of \( V \) after the assignment must be equal to the evaluated value of the expression \( E \)”. Formally, if the statement \( A \) is true before the assignment, then the statement obtained by substituting \( V \) by the evaluated value of \( E \) must be true after the statement execution. The precondition ensures that the validity of substitution must be checked before the assignment. One such validation constraint is type checking.

**Example 1:** \{ \( x == 0 \land y == 10 \) \} \( x = y \); \( \{ x == 10 \land y == 10 \} \)

In this example, it asserts that the value of \( x \) is changed by the assignment statement while the value of \( y \) remains the same. Types of \( x \) and \( y \) are the same as indicated by the pre-condition.

Is the following assignment statement true?
\{ \( y == 10 \) \} \( x = y \); \( \{ x == 10 \land y == 10 \} \)

Explain.

**Example 2:** \{ \( x + 1 == n + 1 \) \} \( x = x + 1 \); \( \{ x == n + 1 \} \)

In this example, the precondition helps evaluating the value of \( x \) before the assignment and the postcondition asserts what its value will be after the statement execution.

The assignment rule itself may look simple, but it is mainly intended to find the precondition for an assignment statement. So, given a statement \( S \) and an expected postcondition \( Q \),
what will be the precondition $P$ such that the Hoare’s triplet $\{P\} S \{Q\}$ holds.

There may be (and usually there are) several $P$ that satisfy this triplet. A verifier is more interested in finding the weakest precondition that satisfies the triplet. There is only one weakest precondition for any such triplet.

Once a weakest precondition is found, it can be strengthened to make it more suitable for the current application. The next section discusses more about strengthening the precondition.

**Precondition strengthening:**

\[
\begin{align*}
P & \Rightarrow P', \quad \{P'\} S \{Q\} \\
\{P\} S \{Q\} & 
\end{align*}
\]

The precondition strengthening rule describes that a precondition for a statement could have been derived from another condition. This happens when the preconditon for a procedure or a method (which is $P'$) can be derived from the statement that calls this procedure or method (which is identified as $P$ in the rule given above).

In example 2 above, you can write the assignment statement as

\[
\begin{align*}
\{x + 1 == n + 1 \Rightarrow x == n\}, & \quad \{x == n\} x = x + 1 \{x == n + 1\} \\
\{x + 1 == n + 1\} x = x + 1 \{x == n + 1\} & 
\end{align*}
\]

Example 3: Given that $x == abs(x)$ and the assignment statement

$\{x \in \mathbb{N}\} y = factorial(x) \{y == x!\}$, one can write the following proof:

\[
\begin{align*}
x == abs(x) \Rightarrow x \in \mathbb{N}, & \quad \{x \in \mathbb{N}\} y = factorial(x) \{y == x!\} \\
x == abs(x) & \quad \{x \in \mathbb{N}\} y = factorial(x) \{y == x!\} 
\end{align*}
\]

This example shows that if one can guarantee that the parameter for the factorial function is always its absolute value, then it is guaranteed that it will compute the factorial value correctly. Stated otherwise, the condition “the input parameter for the factorial function must be a non-negative number” is satisfied when the parameter is in its absolute value.

**Postcondition weakening**

\[
\begin{align*}
\{P\} S \{Q'\}, & \quad Q' \Rightarrow Q \\
\{P\} S \{Q\} & 
\end{align*}
\]

The postcondition weakening rule asserts that if you can derive a postcondition $Q'$ through the execution of an assignment statement and the postcondition $Q'$ implies another condition $Q$, then you eventually prove $Q$ through the execution of this assignment statement.
Example 4: Given the assignment statement
\[
\{ r == x \} \quad q = 0 \{ r == x \land q == 0 \}, \]
one can prove that \( r == x \land q == 0 \) \( r == x + y \ast q \) for any \( y \).
The proof follows from postcondition weakening if we can show that \( r == x \land q == 0 \Rightarrow r == x + y \ast q \).

The proof follows from basic arithmetic rule that “the multiplication of any number with zero is zero”. Since \( q == 0 \) (from the left side of the implication), one can see that \( y \ast q == 0 \) on the right side of the implication. Therefore, \( r == x + y \ast q \) reduces to \( r == x \) which is already true from the left side.

Here is an interesting question[1]:
Given the assignment statement \( \{ y == 3 \} \ x = y \ \{ x > 2 \} \), is this provable from the axiom/rule for assignment statement? If Yes, how? If No, why not?

**Specification conjunction**
\[
\begin{align*}
\{ P_1 \} & \ S \ \{ Q_1 \}, \\
\{ P_2 \} & \ S \ \{ Q_2 \} \\
\{ P_1 \land P_2 \} & \ S \ \{ Q_1 \land Q_2 \}
\end{align*}
\]
The above rule asserts that the execution of a statement may have multiple consequences, depending on multiple scenarios. See the example below:

Example 5: Given the following assignment statements
\[
\{ x \in \mathbb{Z} \land y \in \mathbb{Z} \} \ x = \text{abs}(x) \ast \text{abs}(y) \ \{ x \in \mathbb{N} \}
\]
and
\[
\{ x \in \mathbb{Z} \land y == 0 \} \ x = \text{abs}(x) \ast \text{abs}(y) \ \{ x == 0 \}
\]
one can deduce that
\[
\begin{align*}
\{ x \in \mathbb{Z} \land y \in \mathbb{Z} \} \ x & = \text{abs}(x) \ast \text{abs}(y) \ \{ x \in \mathbb{N} \}, \\
\{ x \in \mathbb{Z} \land y == 0 \} \ x & = \text{abs}(x) \ast \text{abs}(y) \ \{ x == 0 \} \\
\{ x \in \mathbb{Z} \land y == 0 \} \ x & = \text{abs}(x) \ast \text{abs}(y) \ \{ x \in \mathbb{N} \land x == 0 \}
\end{align*}
\]

**Specification disjunction**
\[
\begin{align*}
\{ P_1 \} & \ S \ \{ Q_1 \}, \\
\{ P_2 \} & \ S \ \{ Q_2 \} \\
\{ P_1 \lor P_2 \} & \ S \ \{ Q_1 \lor Q_2 \}
\end{align*}
\]
The above rule is similar to specification conjunction rule except the conjunction is replaced by a disjunction.
Example 6: Given the following statements

\[ \{ x > 0 \land y \in \mathbb{Z} \} \ x = \max(x, y) \ \{ x > 0 \} \]

and

\[ \{ x \in \mathbb{Z} \land y == 0 \} \ x = \max(x, y) \ \{ x \geq 0 \} \]

one can deduce that

\[ \{ x > 0 \land y \in \mathbb{Z} \} \ x = \max(x, y) \ \{ x > 0 \}, \quad \{ x \in \mathbb{Z} \land y == 0 \} \ x = \max(x, y) \ \{ x \geq 0 \} \]

\[ \{ x > 0 \lor y == 0 \} \ x = \max(x, y) \ \{ x > 0 \lor x \geq 0 \} \]

Actually, in this example, the conclusion can be still further reduced to \( x \geq 0 \).

Sample problems

The following problems and their solutions were taken from[1].

1. Consider the code \( x = x + 1 \) and the postcondition \( x > 0 \). Find, if possible, the weakest precondition \( P \) such that \( \{ P \} \ x = x + 1 \ \{ x > 0 \} \) holds.

One possible precondition is \( x > 0 \) such that \( \{ x > 0 \} \ x = x + 1 \ \{ x > 0 \} \) holds.

Another precondition is \( x > -1 \) such that \( \{ x > -1 \} \ x = x + 1 \ \{ x > 0 \} \) holds.

\( x > -1 \) is weaker than \( x > 0 \) because \( x > 0 \Rightarrow x > -1 \). In fact, \( x > -1 \) is the weakest precondition.

2. Consider the code \( a = a + 1; \ b = b - 1; \) and the postcondition \( a \ast b = 0 \). Find, if possible, the weakest precondition \( P \) such that \( \{ P \} \ a = a + 1; \ b = b - 1; \ \{ a \ast b == 0 \} \) holds.

One possible precondition is \( a == -1 \).

Another possible precondition is \( b == 1 \).

Since neither precondition implies another, both are equally capable to satisfy the triplet. So, the weakest precondition in this case is \( (a == -1) \lor (b == 1) \) such that \( \{(a == -1) \lor (b == 1)\} \ a = a + 1; \ b = b - 1; \ \{ a \ast b == 0 \} \) holds.

Some exercises

1. Given integers \( x \) and \( y \), find, if possible, the weakest precondition \( P \) such that \( \{ P \} \ x = x + y \ \{ y > x \} \) holds.

2. Given integer \( y \), find, if possible, the weakest precondition \( P \) such that \( \{ P \} \ y = 2 \ast y \ \{ y < 5 \} \) holds.
Sequencing rule

\[
\begin{align*}
\{P\} \ S_1 \ \{Q\}, \\
\{P\} \ S_1; \ S_2 \ \{R\}
\end{align*}
\]

Informally, the sequence rule describes that the execution of two consecutive statements can be combined into one composite statement as long as the postcondition of the first statement becomes or implies the precondition of the second statement.

Derived Sequencing rule

\[
\begin{align*}
&P \Rightarrow P_1, \\
&P_1 \ S_1 \ \{Q_1\}, \\
&Q_1 \Rightarrow P_2, \\
&P_2 \ S_2 \ \{Q_2\}, \\
&Q_2 \Rightarrow P_3, \\
&\ldots \\
&P_n \ S_n \ \{Q_n\}, \\
&Q_n \Rightarrow Q
\end{align*}
\]

The derived sequencing rule is an extension of the sequencing rule and is applicable for a block of statements. In fact, this rule is used to prove that if the individual statements of a program are correct, and their sequencing is correct, then the whole program is correct.

Finding weakest preconditions for sequence of statements

In a sequence of statements, the effect on any one statement \( S_i \) depends on the effects of the statements preceding this statement. Based on this, the following rule is applied to find the weakest precondition of a sequence of statements:

Given \( \{P\} \ S_1; \ S_2 \ \{Q\} \),
Let \( P_1 \) be the precondition applied to \( S_2 \) such that \( \{P_1\} \ S_2 \ \{Q\} \) holds. In other words, the effect of \( S_2 \) resulting in postcondition \( Q \) depends on its precondition \( P_1 \). This precondition is actually the result of executing \( S_1 \). Therefore, \( \{P\} \ S_1 \ \{P_1\} \) should hold. Summarizing this,

\[
\{P\} \ S_1 \ \{P_1\} \land \{P_1\} \ S_2 \ \{Q\}
\]

So, in order to find the weakest precondition of a sequence of statements \( S_1; \ S_2; \ldots; \ S_n \), we iterate through the statements backwards such that the precondition of a statement \( S_n \) resulting in \( Q \) is found first which is then used as the postcondition of the statement \( S_{n-1} \) and so on.

Example 7: Consider the code \( x = x + 2; \ y = y - 2 \) with the postcondition \( x + y == 0 \). Find the weakest precondition \( P \) such that \( \{P\} \ x = x + 2; \ y = y - 2; \ \{x + y == 0\} \) holds. (taken from [1])
Consider the sub-problem \( \{P_1\} \quad y = y - 2 \quad \{x + y \equiv 0\} \). We know that value of \( x \) must be an integer in order to satisfy the postcondition. To make \( x + y \equiv 0 \), \( x \) must be equal to \( y \) but with opposite sign. i.e., \( x \equiv -y \). But this is the value of \( y \) after executing the statement \( y = y - 2 \). Hence we arrive at a precondition \( x \equiv -(y - 2) \) such that \( \{x \equiv -(y - 2)\} \quad y = y - 2 \quad \{x + y \equiv 0\} \) holds.

Based on the precondition calculation for sequence of statements, \( x \equiv -(y - 2) \) is the postcondition of \( x = x + 2 \). So, a precondition for that statement would be \( x + 2 \equiv -(y - 2) \). Solving this, we get \( x \equiv -y \). This will be the weakest precondition such that \( \{x \equiv -y\} \quad x = x + 2; \quad y = y - 2 \quad \{x + y \equiv 0\} \) holds.

References