Notes on Program Verification - Part 2

Conditional rules

\[
\begin{align*}
\{ P \land C \} &\rightarrow S_1 \{ Q \}, & \{ P \land \neg C \} &\rightarrow S_2 \{ Q \} \\
\{ P \} &\text{ if } C \text{ then } S_1 \text{ else } S_2 \{ Q \}
\end{align*}
\]

In the above rule, \( C \) refers to the condition in the if statement, \( P \) refers to the precondition just before the if statement and \( Q \) refers to the postcondition right after the if statement. So, if the precondition \( P \) is true, the execution enters into the if block. Depending on the condition \( C \), either the statement \( S_1 \) is executed or \( S_2 \) is executed, but not both (mutual exclusion). Notice that the precondition is common for the execution of \( S_1 \) or \( S_2 \) and both statements result in the same postcondition \( Q \).

Example 1: Given \( x \in \mathbb{Z} \) and \( y \in \mathbb{N} \), consider the code

\[ \text{if } x > y \text{ then } y = x - y \text{ else } y = x \]

with the postcondition \( y \geq 0 \).

Rewriting this in the rule format, assuming \( P \) is the precondition,

\[
\begin{align*}
\{ P \land (x > y) \} &\rightarrow y = x - y \{ y \geq 0 \}, & \{ P \land \neg (x > y) \} &\rightarrow y = x \{ y \geq 0 \} \\
\{ P \} &\text{ if } x > y \text{ then } y = x - y \text{ else } y = x \{ Q \}
\end{align*}
\]

What will be the weakest precondition \( P \) for the above example?

Let us consider the first part

\( \{ P \land (x > y) \} \rightarrow y = x - y \{ y \geq 0 \} \)

In order for the postcondition to be true after successfully executing the statement \( y = x - y \), we need to assert that \( x - y \geq 0 \) which is equivalent to \( x \geq y \). The condition \( C \) of the if statement \( x > y \) is a stronger condition (i.e., \( x > y \Rightarrow x \geq y \)). So, the weakest precondition for this part is \( x \geq y \).

Now, consider the second part

\( \{ P \land \neg (x > y) \} \rightarrow y = x \{ y \geq 0 \} \)

In order for the postcondition to be true after successfully executing the statement \( y = x \), we need to assert that \( x \geq 0 \). This implies that \( x \) should be a natural number (even though it is declared as an integer, as given in the problem). But this statement will be executed only if the condition in if statement is false (i.e., \( \neg (x > y) \equiv x \leq y \)). Combining the two, we need to assert that \((x \geq 0) \land (x \leq y)\), in order for the statement to arrive at the postcondition after successfully executing the statement. Since \( y \) is a natural number, this condition thus requires that \( x \) should be a natural number.

Combining the two parts together, the weakest precondition that satisfies both parts is \( x \) is a natural number (i.e. \( x \geq 0 \)).
Sometimes, we use the if statement without the else clause. The rule for this situation can be derived from the rule for if ... then ... else ...

\[
\frac{\{P \land C\} \quad S \quad \{Q\}, \quad \{P \land \neg C\} \Rightarrow \{Q\}}{\{P\} \text{ if } C \text{ then } S \quad \{Q\}}
\]

In this rule, the second part shows how the postcondition \( Q \) is implied directly by the conjunction of the precondition and the condition in the if statement without executing any code.

Example 2: Given \( x \in \mathbb{Z} \), consider the code

\[
\text{if } x < 0 \text{ then } x = -x
\]

with the postcondition \( x \geq 0 \) (this code represents the ‘absolute’ function for integers).

Rewriting this in the rule format, assuming \( P \) is the precondition,

\[
\frac{\{P \land (x < 0)\} \quad x = -x \quad \{x \geq 0\}}{\{P\} \text{ if } x < 0 \text{ then } x = -x \quad \{x \geq 0\}}
\]

Let us consider the first part

\[
\{P \land (x < 0)\} \quad x = -x \quad \{x \geq 0\}
\]

In order for the postcondition to be true after successfully executing the statement \( x = -x \), we need to assert that \( x < 0 \) to start with. This happens to be the condition \( C \) of the if statement as well in this code. So, the precondition for this part is the same as the condition \( C \) of the if statement.

Now consider the second part

\[
\{P \land \neg (x < 0)\} \Rightarrow \{x \geq 0\}
\]

Simplifying the negation, we get

\[
\{P \land (x \geq 0)\} \Rightarrow \{x \geq 0\}
\]

This indicates that whatever \( P \) is, it must be simply true. So, the weakest precondition is simply TRUE.

Combining the two parts, we get the weakest precondition being simply TRUE. This means that this code can be executed in any situation because the postcondition does not relay on the precondition.

Apart from finding the precondition of an if statement, we could use these rules to verify whether two code fragments are equivalent. This is illustrated by the following case studies.

1. Show that, using the rules for conditional statements, the following two code fragments are equivalent:

   Code fragment 1:

   \[
   \text{if } (n >= '0' \&\& n <= '9') \{
   \]
print ("n is a digit");
else
    print ("n is not a digit");

Code fragment 2:
if (n >= '0') {
    if (n <= '9') {
        print ("n is a digit");
        print ("n is not a digit");
    } else print ("n is not a digit");
else print ("n is not a digit");

Let
P - the precondition for the fragment of code (same for both)
Q - the post-condition for the fragment of code (same for both)
of the second if block.
C_1 - the condition n ≥ '0'
C_2 - the condition n ≤ '9'
S_1 - the statement print ("n is a digit")
S_2 - the statement print ("n is not a digit")

Notice that both code fragments must have the same precondition P and postcondition Q; otherwise, they cannot be equivalent.

Rewriting the code in the rules format, the first code fragment will be written as

\[
\begin{align*}
&P \land (C_1 \land C_2) \rightarrow S_1(Q) \\
&P \land \neg (C_1 \land C_2) \rightarrow S_2(Q)
\end{align*}
\]

\[
\begin{align*}
&P \land C_1 \rightarrow S_1(Q) \\
&P \land \neg C_1 \rightarrow S_2(Q)
\end{align*}
\]

For the proof, we only need the axioms in the top and so hereafter we only represent the axioms. So, extracting the axioms of the first code fragment,
\[
\begin{align*}
&P \land (C_1 \land C_2) \rightarrow S_1(Q) \\
&P \land \neg (C_1 \land C_2) \rightarrow S_2(Q)
\end{align*}
\]

For the second code fragment,
\[
\begin{align*}
&P \land C_1 \rightarrow S_1(Q) \\
&P \land \neg C_1 \rightarrow S_2(Q)
\end{align*}
\]

The inner block can be written separately as
\[
\begin{align*}
&(P \land C_1) \land C_2 \rightarrow S_1(Q) \\
&(P \land C_1) \land \neg C_2 \rightarrow S_2(Q)
\end{align*}
\]

Substituting the inner block expression in the axioms for the second code fragment, we get
\[
\begin{align*}
&P \land C_1 \rightarrow [(P \land C_1) \land C_2 \rightarrow S_1(Q)] \\
&P \land C_1 \rightarrow [(P \land C_1) \land \neg C_2 \rightarrow S_2(Q)] \\
&P \land \neg C_1 \rightarrow S_2(Q)
\end{align*}
\]
\{P \land \neg C_1\} S_2\{Q\}

The precondition \{P \land C_1\} is required to enter into the inner block and so it is conjoined with the precondition of the inner block. After simplification,
\{P \land C_1 \land C_2\} S_1\{Q\} \quad (3)
\{P \land C_1 \land \neg C_2\} S_2\{Q\} \quad (4)
\{P \land \neg C_1\} S_2\{Q\} \quad (5)

To prove that both code fragments are equivalent, we need to show that the set of axioms (1) and (2) describes the same situation by the set of axioms (3), (4) and (5). Equations (1) and (3) are equivalent and describe the same situation (same precondition, same statement executed resulting in the same post-condition). We will now prove that equation (2) is equivalent to the combination of equations (4) and (5).
Equation (2) has the precondition \neg (C_1 \land C_2) \equiv \neg C_1 \lor \neg C_2 \quad \text{DeMorgan’s law.}
Since equations (4) and (5) indicate that the same statement is executed and result in the same post-condition, it becomes obvious that one or both of the preconditions can be true in order to execute the statement $S_2$. That is,
\begin{align*}
(C_1 \land \neg C_2) \lor \neg C_1 \\
\equiv (C_1 \lor \neg C_1) \land (\neg C_2 \lor \neg C_1) \quad \text{Distributive law} \\
\equiv \text{true} \land (\neg C_2 \lor \neg C_1) \\
\equiv (\neg C_2 \lor \neg C_1) \quad \text{DeMorgan’s law} \\
\equiv \neg (C_2 \land C_1) \quad \text{Commutativity of } \land
\end{align*}
Therefore, equation (2) describes the same situation as the combined effects of equations (4) and (5). Hence, both code fragments are equivalent.

2. Show that, using the rules for conditional statements, the following two fragments of code are equivalent:

```java
if (x >= 0) {
    if (y < MAX) {
        System.out.print ("'1'");
    }
    else
        System.out.print ("'2'");
} else
    System.out.print ("'2'");
```

```java
if (x >= 0 && y < MAX) {
    System.out.print ("'1'");
} else {
    System.out.print ("'2'");
}
```

For simplicity, we write only the hypotheses (axioms in the top portion) in the rules ignoring the bottom portion. This simplification is mainly because the bottom portion in the rule actually shows the code.
Let us use the following symbols in the code:

- **P** - the precondition for the fragment of code (same for both)
- **Q** - the post-condition for the fragment of code (same for both)
- **P'** - in code fragment 2, the post-condition of the first if block
- which is the same as the precondition of the second if block.

- **C1** - the condition \( x \geq 0 \)
- **C2** - the condition \( y < \text{MAX} \)
- **S1** - the statement `System.out.println("1")`
- **S2** - the statement `System.out.println("2")`

For the first code fragment, the rule would be

\[
\{ P \land C_1 \} \{ \text{inner block} \} \{ Q \} \quad \{ P \land \neg C_1 \} \quad S_2 \quad \{ Q \}
\]

The inner block axioms are written as

\[
\{ (P \land C_1) \land C_2 \} \quad S_1 \quad \{ Q \}
\]

\[
\{ (P \land C_1) \land \neg C_2 \} \Rightarrow \{ Q \}
\]

\[
\{ P \land \neg C_1 \} \quad S_2 \quad \{ Q \}
\]

This can be simplified as

\[
\{ P \land C_1 \land C_2 \} \quad S_1 \quad \{ Q \} \quad (1)
\]

\[
\{ P \land C_1 \land \neg C_2 \} \Rightarrow \{ Q \} \quad (2)
\]

\[
\{ P \land \neg C_1 \} \quad S_2 \quad \{ Q \} \quad (3)
\]

For the second code fragment, the rule would be

\[
\{ P \land C_1 \land C_2 \} \quad S_1 \quad \{ P' \} \quad (4)
\]

\[
\{ P \land \neg (C_1 \land C_2) \} \Rightarrow \{ P' \} \quad (5)
\]

\[
\{ P' \land \neg C_1 \} \quad S_2 \quad \{ Q \} \quad (6)
\]

\[
\{ P' \land C_1 \} \Rightarrow \{ Q \} \quad (7)
\]

In order to show that the two code fragments are equivalent, we show that the set of axioms (1) through (3) describe the same situations by the axioms (4) through (7).

We do this base analyzing all four possible scenarios of the two conditions \( C_1 \) and \( C_2 \) on a case-by-case basis.

**Case 1:** \( C_1 \) and \( C_2 \) are both true.

For first code fragment,

By equation (1), statement \( S_1 \) will be executed and post-condition \( Q \) is reached.

For second code fragment,
By equation (4), statement $S_1$ will be executed resulting in post-condition $P'$.
By equation (7), the postcondition $Q$ is arrived.

Case 2: $C_1$ and $C_2$ are both false.
For first code fragment,

By equation (3), statement $S_2$ will be executed resulting in post-condition $Q$.

For second code fragment,

By equation (5), we first arrive at the intermediate postcondition $P'$.
By equation (6), statement $S_2$ will be executed resulting in post-condition $Q$.

Case 3: $C_1$ is true and $C_2$ is false.
For first code fragment,

By equation (2), the code will lead to post-condition $Q$ straight without executing any statement.

For second code fragment,

By equation (5), we will arrive at the intermediate postcondition without executing any statement.
By equation (7), the code will reach post-condition $Q$ without executing any statement.

Case 4: $C_2$ is true and $C_1$ is false.
For first code fragment,

By equation (3), $S_2$ will be executed resulting in post-condition $Q$.

For second code fragment,

By equation (5), the code will reach the post-condition $P'$ without executing any statement.
By equation (6), statement $S_2$ will be executed resulting post-condition $Q$.

We have thus proved that both code fragments result in the same post-condition for the same precondition and the conditions in the if statements, and hence they are equivalent.

This exercise also illustrates how the basic rules for the conditional statements can be used for nested statements, a situation that occurs so common in practice.