Hoare Logic examples

Introduction

This document contains some additional examples of Hoare Logic triples. There are also example solutions for the first two, you should be able to do the rest yourself.

Examples

For each of these examples, check whether the triple is valid or not.

Example 1

```
[x = y+1]
x := z-x;
y := z-y
[x < y]
```

Example 2

```
[x >= y]
IF z > 0 THEN
   x := x+z
ELSE
   y := y+z
END
[x > y]
```

Example 3

```
[true]
IF x < y THEN
   min := x
ELSE
   min := y
END
```

```
[min <= x & min <= y]
```

Example 4

```
[x <= y]
x := y;
y := x
[y <= x]
```

Example 5

```
[true]
sum := x+y;
IF x<y THEN
   min := x;
   max := y
ELSE
   min := y;
   max := x
END
[sum = min + max]
```

Example 6

```
[sum >= 0]
IF x > 0 THEN
   sum := sum + x
END;
IF y > 0 THEN
   sum := sum + y
END
[sum >= x+y]
```

Example solutions

This section contains solutions for the first two examples

Example 1

We are given the triple

```
[x = y+1]
x := z-x;
y := z-y
[x < y]
```

In order to check if it is valid, we are going to compute the weakest pre-conditions and check if they follow from the given one. So we will want something like this:

```
[x = y+1]
[Assertion 1]
x := z-x;
[Assertion 2]
y := z-y
[x < y]
```

where we need to find the missing assertions and then see if $x = y + 1$ implies assertion 1. We start at the bottom: assertion 2 is a copy of the post-condition with y replaced by $z - y$.

```
[x = y+1]
[Assertion 1]
x := z-x;
[x < z-y]   // Assertion 2: obtained by replacing y with z-y in x < y
y := z-y
[x < y]
```

Next we get assertion 1 by replacing x with $z - x$ in assertion 2.

```
[x = y+1]
[z-x < z-y]    // Assertion 1: obtained by replacing x with z-x in x < z-y
x := z-x;
[x < z-y]
y := z-y
[x < y]
```

We can simplify assertion to get $x > y$. This does follow from the given pre-condition: if $x = y + 1$ then $x > y$. So the triple is valid.

Example 2

We are given

```
[x >= y]
IF z > 0 THEN
   x := x+z
ELSE
   y := y+z
END
[x > y]
```

We will again need some intermediate assertions:

```
[x >= y]
```

```
[Assertion 1]
IF z > 0 THEN
   [Assertion 2]
   x := x+z
   [x > y]
ELSE
   [Assertion 3]
   y := y+z
   [x > y]
END
[x > y]
```

Here we have already copied the post-condition into each branch.

We compute the pre-conditions for assignments as before to get assertions 2 and 3:

```
[x >= y]
[Assertion 1]
IF z > 0 THEN
   [x+z > y]   // Assertion 2: obtained by replacing x with x+z in x > y
   x := x+z
   [x > y]
ELSE
   [x > y+z]   // Assertion 3: obtained by replacing y with y+z in x > y
   y := y+z
   [x > y]
END
[x > y]
```

Then we combine them to get the pre-condition for the conditional:

```
[x >= y]
[(z > 0 => x+z > y) & (z <= 0 => x > y+z)]
IF z > 0 THEN
   [x+z > y]
   x := x+z
   [x > y]
ELSE
   [x > y+z]
   y := y+z
   [x > y]
END
[x > y]
```

Finally we need to check if the given pre-condition implies the one we have derived:

If $x >= y$ and $z > 0$, is $x + z > y$? Yes.

If $x >= y$ and $z <= 0$, is $x > y + z$? No: A counterexample is $x = 0, y = 0, z = 0$. So this triple is not valid.