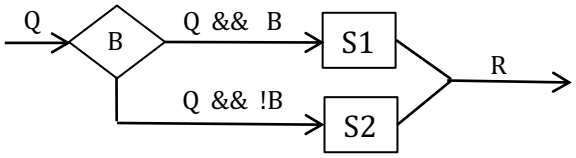
Using the Hoare triple to define statements

We see how Hoare triples are used to define other parts of a programming language. We won't use this material formally, but this *does* formalize the informal thought processes that one goes through when programming.

The if-else statement

We start off with the if-else statement. How can we know that executing the if-else statement beginning with precondition Q true will terminate with postcondition R true?

```
{Q}
if (B) S1
else S2
{R}
```



To the right is a flowchart of this if-else-statement, with an assertion on each edge. Q is on the incoming edge, R is on the outgoing edge, Q && B is on the edge with B being true, and Q && !B is on the edge with B being false. Evidently, we have to show that executing S1 with Q && B true terminates with R true, and similarly for the case when B is false. Thus, from this flowchart, we can say the following:

> **Hoare triple for if-else**:
> If {Q && B} S1{R} and {Q && !B} S2 {R}
> then {Q} **if** (B) S1 **else** S2 {R}

This should be self-evident, common sense: If R is to be true after execution of the if-else statement, we have to be sure that it is true whether B is true and S1 is executed or B is false and S2 is executed.

Sequencing

Suppose we want to prove {Q} S1; S2 {R}. This requires showing that each of S1 and S2 does its job, but we have to find out what their jobs are! We have to find an assertion A that acts as a postcondition for S1 and a precondition for S2:

> {Q} S1; {A} S2 {R}

We state this as follows:

> **Hoare-triple for S1; S2**
> If {Q} S1 {A} and {A} S2 {R},
> then {Q} S1; S2 {R}.

Implication

Let A and B be boolean expressions. A implies B, written A => B , is true if whenever A is true B is also true. Here is an example:

> $x = 5 \Rightarrow x \geq 5$

We sometimes put two assertions in a row, as in the expression below, meaning that the first implies the second

> $\{x = 5\}\ \{x \geq 5\}\ x = x+1;\ \{x \geq 6\}$

This may seem silly, but there *are* real cases where such assertions arise. In such situations, we have to show that the first assertion implies the second. In this case it is obvious.

Here are two general forms for this situation, one before a stateement and the other after a statement:

> {Q} {Q1} S {R}
>
> {Q} S {R1} {R}

And here is how we define what this means:

> **Hoare triple including implication**
> If Q => Q1 and {Q1} S {R},
> then {Q} S {R}
>
> **Hoare triple including implication**
> If {Q1} S {R1} and R1 => R
> then {Q} S {R}