Program Proving — Auxiliary Rules

skip axiom

Diller pursues one other atomic statement — the **skip** statement, which is a no-op. It's contribution to the proving paradigm is the axiom scheme

I— {P} **skip** {P}

where P is any predicate logic formula. Intuitively, since skip does nothing, what is true after it's execution is exactly the same as what was true before.

Auxiliary rules

The axiom of assignment alone is insufficient for the proofs we wish to carry out for assignment statements. Consider the program assertion

{Y = 5} X := 2 {Y >X}.

This is a claim we certainly wish to be able to justify. However, from the axiom of assignment what we can actually prove is

|— {Y > 2} X := 2 {Y>X}

and Y > 2 ≢ Y = 5. Therefore, we cannot prove this program assertion using the axiom of assignment. However, Y = 5 □ Y > 2, and to resolve such proof failures we introduce the first rule of consequence

Strengthening the pre-condition

This is the first of the rules of deduction in our program proving system. The intuitive idea is that if a program assertion can be proven, then the pre-condition can be replaced by any formula that implies it. Schematically,

$$\frac{\text{I— \{Q\} □ \{R\}, I— P □ Q}}{\text{I— \{P\} □ \{R\}}}$$

where □ is any program fragment.

In the previous example, we can now accomplish the proof in two steps, using first the axiom of assignment, then strengthening the pre-condition.

For another instance, consider the program assertion

{ X=1   Y=1} X := 1 { X=1   Y=1}.

Again, this is a claim we definitely wish to be able to justify. However, from the axiom of assignment what we can actually prove is

I— {**true**} X := 1 { X=1   Y=1}.

and now the desired pre-condition implies the pre-condition established by the Axiom of Assignment, so strengthening the pre-condition can be used to complete the desired proof.

Weakening the post-condition

This is the next rule of deduction in our program proving system. The intuitive idea is that if a program assertion can be proven, then the post-condition can be replaced by any formula that it implies. Schematically,

I— {P} □ {Q}, I— Q □ R

I— {P} □ {R}

where □ is any program fragment.

Now to prove { X=1 Y=1} X := 1 { X=1 Y=1}, wecan use the following steps:

1. I— { **true**} X := 1 { X=1} by the axiom of assignment
2. I— X=1 Y=1 □ **true** by logic
3. I— { X=1 Y=1} X := 1 { X=1} by 1 & 2 and strengthening the pre-condition
4. I— X=1 □ X=1 Y=1 by logic
5. I— { X=1 Y=1} X := 1 { X=1 Y=1} by 3 & 4 and weakening the post-condition.