Program Proving — Axiom of Assignment

For this presentation we assume that X := ξ denotes an *assignment statement*, where X is a variable and ξ is a suitable expression. The axiom of assignment underlies the logical system for proving imperative programs. It is actually an *axiom scheme* — a readily recognizable pattern that can be instantiated in a variety of ways.

axiom of assignment: |— {P[X→ξ]} X := ξ {P}

where P is a predicate logic formula, X is a variable, ξ is an expression, and P[X→ξ] is the formula P with each occurrence of X replaced by ξ. Such a proving step is validated by confirming the syntactic formula correspondence. However, we do not distinguish formulas that are logically equivalent.

The orientation of the assignment axiom leads us to think “in reverse” about assignment. This is because the rule identifies the “weakest pre-condition” needed to justify the post-condition P. If P is true of the variable values after the assignment, then P must be true of the same values before the assignment since a logic formula expresses a static relationship. But before the assignment, the only variable that is different is X, and its value after the assignment is obtained by the evaluation of ξ with the variable values prior to the assignment.

Although this back to front orientation may seem unnatural from the more familiar (forward) execution perspective, in proving we typically know a post-condition (the results we seek), and working back from it can be helpful in formulating needed pre-conditions. Some examples should help this seem more natural.

“Strength” of wffs

We will frequently have occasion to refer to the *strength* of a wff. If P and Q are wffs and it is true that P ⇒ Q, then we say that P is a **stronger** assertion than Q, and Q is **weaker** than P. A stronger condition is one that is more restrictive — fewer values satisfy a stronger condition; conversely, a weaker condition is one that is more permissive — more values satisfy a weaker condition. For instance, | x>0 ⇒ x≥0.

For an individual pair of assertions, it may be that neither is stronger than the other, for instance x>0 and y>0. But for an individual formulas, there will be a span of strengthening/weakening assertions from one extreme to the other.

false	strongest, |— **false** ⇒ Q for all Q
…	
x>5	
…	
x=2	
…	
x=2 v x=1	↑ stronger (more restrictive)
…	
x>0	↓ weaker (less restrictive)
…	
x≥0	
…	
x>-5	
…	
true	weakest, |— **true** ⇒ Q only for Q≡**true**