

Proving Program Correctness

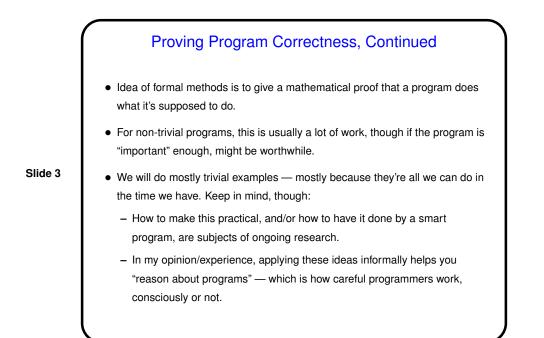
- Once you've written a program, want to have some confidence that "it works".
- What do you mean "it works"? Informally? Formally, "meets its specification" (more later).

Slide 2

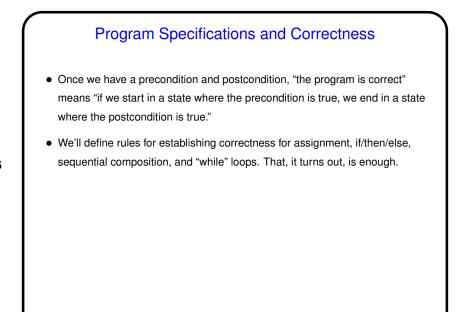
• How do you show it works? As a grad-school colleague wrote:

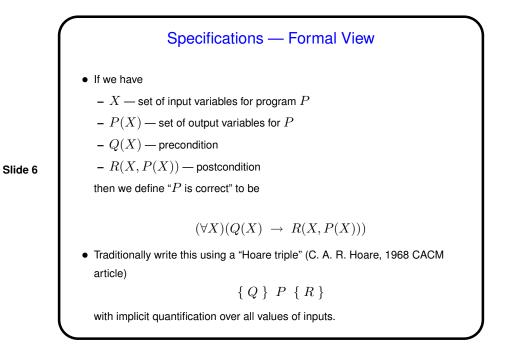
To reduce the number of errors in a program, or to increase one's confidence in a program, one can *test* the program on a given test suite. If the program is observed to behave correctly for these test cases, the program is shipped to the customer. One then hopes there will be other cases that customers try for which the program also behaves correctly.

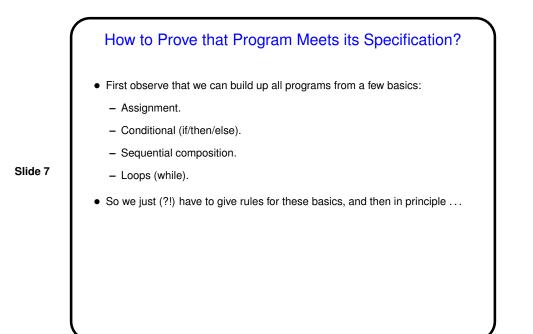
• Is there another way to "increase your confidence" in the program? "Formal methods" . . .

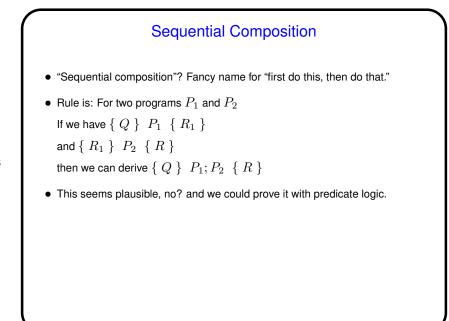


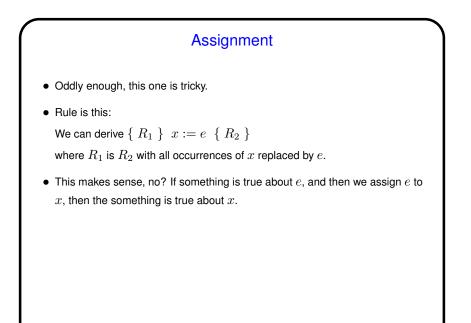
Program Specifications
 Before we can prove that a program "works", we have to define what that means — "specification".
• For many programs (the ones we'll talk about here), we care that the program produces the right output for all allowed inputs. So we can write a specification in terms of "precondition" and "postcondition". E.g., for a function sqrt that takes a double <i>x</i> as input and returns a double, we could have:
Precondition: $x \ge 0$. Postcondition: For return value $y, y \ge 0$ and $y^2 = x$.
• This is trivial? Consider the following proposed specification for a sorting function with two inputs A (array of integers) and n (size of A). Okay? Precondition: A is of size $n, n \ge 0$.
Postcondition: $(\forall i)(((0 < i < n) \rightarrow A[i-1] \le A[i])$



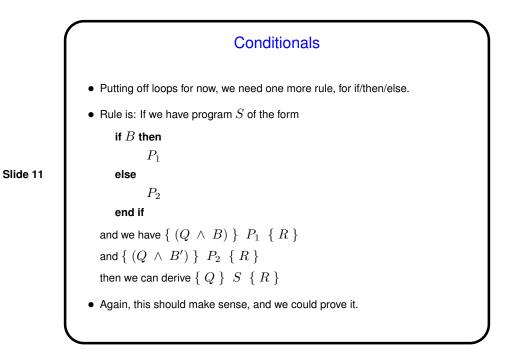








Strengthening Preconditions, Weakening Postconditions • Two more rules: If we have $\{Q\} P \{R\}$ then for "stronger" precondition Q_1 (i.e., $Q_1 \rightarrow Q$) we can derive $\{Q_1\} P \{R\}$ and for "weaker" postcondition R_1 (*i.e.*, $R \rightarrow R_1$) we can derive $\{Q\} P \{R_1\}$ • This also should make sense, and we could prove it. Also, it can be helpful in applying the rule for sequential composition when the postcondition / precondition pairs don't quite match up.



Slide 12 **Example** • Try an example — silly program to compute absolute value (call it *S*): if $x \ge 0$ then y := xelse y := -xend if We want to show that { true } S { y := |x| } • We can do this using rules for conditionals and assignment ...

