CSE 451, Fall 1999

Strategies for Developing a Correct Program Using IF

(more to come!)

Strategy 1: Look at the *goal* of the program (i.e., the postcondition)

Strategy 2:

Refine the postcondition until it's usable;
e.g., replace high-level concepts with their definitions

Strategy 3:

Suppose we know a *precondition* Q for an IF.
To develop a guarded command for IF:
repeat

1. find a command S_i such that $S_i\{R\}$ at least sometimes
2. find a Boolean guard B_i such that $B_i \Rightarrow wp(S_i, R)$
   $\therefore$ have $B_i \rightarrow S_i$

until $Q \Rightarrow B_1 \vee \ldots \vee B_n$

Strategy 3.1:

To find S such that $S\{R\}$:
choose an S that is (a) suggested by R (cf. Strategy 1)
& (b) simple

Strategy 3.2:

To find B such that $B \Rightarrow wp(S, R)$:
compute $wp(S, R)$, & try $B = wp(S, R)$

Strategy 4: Find pre- & postconditions if not already known

Strategy 4.1: Use initial and final values of variables

Strategy 5: Look at preconditions for clues to find S such that $S\{R\}$

Strategy 5.1:
if precondition is part of postcondition,
use *skip* somewhere

Strategy 6:
By Thm 10.5, each guard B_i must satisfy $Q \wedge B_i \Rightarrow wp(S_i, R)$
$\therefore$ if $Q, wp(S, R)$ are known,
can determine B

Strategy 7:
All other things being equal,
make the guards of an IF as *strong* as possible
(so that errors will cause program to *abort*,
rather than work in a GIGO manner)