CSE 451, Fall 1999

The Strategies So Far, version III (more to come!)

Notes:

1. Numbering corresponds to my lectures of 11, 16, and 18 November.
2. Strategies are listed in "quasi-logical" order.

4: Determine precondition Q and postcondition R (if not already known)

4.1: Use initial and final values of variables to help determine pre- and postconditions.

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

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

3: Suppose we know a precondition Q for an IF.
Then: to develop a guarded command for IF, do:

repeat:

(a) find a command S_i such that $S_i\{R\}$, at least sometimes

3.1: choose an S_i that: (i) is suggested by R (cf. strategy 1)
and (ii) is simple

5: Look at the precondition for clues to find a command S_i such that $S_i\{R\}$.

5.1: If precondition is part of postcondition, use *skip*.

(b) find a Boolean guard B_i such that $B_i \Rightarrow wp(S_i, R)$ (then use: $B_i \rightarrow S_i$)

3.2: compute $wp(S_i, R)$ and try $B_i = wp(S_i, R)$

6: By Thm 10.5, each guard B_i must satisfy $Q \wedge B_i \Rightarrow wp(S_i, R)$
$\therefore$ if Q and $wp(S_i, R)$ are known, can determine B_i as whatever must be conjoined to Q to imply $wp(S_i, R)$

or

(a′) suppose you have guards $B_1, \ldots, B_{i-1}$;
find B_i such that $Q \Rightarrow (B_1 \vee \ldots \vee B_i)$

(b′) find S_i such that $Q \wedge B_i \Rightarrow wp(S_i, R)$

until $Q \Rightarrow (B_1 \vee \ldots \vee B_n)$ (recall Thm 10.5!)

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

To develop a DO-program, given:

- precondition Q,
- invariant P,
- bound function t,
- and postcondition R:

8. Find an initialization that makes P true before the loop.

Develop guarded commands $B_i \rightarrow S_i$:

repeat:

9.1 Find a guard B such that $P \wedge \neg B \Rightarrow R$;

9.2 develop body S that (a) decrements t (i.e., makes progress towards halting) while (b) preserving P

until: 11.1 $P \wedge \neg(B_1 \vee \ldots \vee B_n) \Rightarrow R$ can be proved.

10. All other things being equal, make the guards of a DO-loop as *weak* as possible (so that errors will cause ∞ loops rather than going undetected).

11.2 If possible, *strengthen* the guards
(as long as $P \wedge \neg(B_1 \vee \ldots \vee B_n) \Rightarrow R$ can still be proved)
in order to derive a shorter or more efficient loop.

11.2.1 If a guard can be strengthened to F, then delete it.