CSE 451, Fall 1999

GRIES'S STRATEGIES FOR DEVELOPING PROGRAMS

Notes: 'B_i' stands for a *Boolean guard*
'P' stands for a *loop invariant*
'Q' stands for a *precondition*
'R' stands for a *postcondition*
'S_i' stands for the *command* of a *guarded command*
't' stands for a *bound function*
Numerals in parentheses correspond to strategy numbering from lecture.

1. Only introduce a variable when there is a good reason,
   and always state (as part of P, Q, and/or R) its bounds. (15.2, 16)

2. Determine Q and R. (4)

   (a) Use initial & final values of variables to help determine Q and R. (4.1)

   (b) Refine R until it's usable by replacing high-level concepts with their definitions. (1, 2)

3. To develop an IF-program: (3)

   (a) **repeat:**

      i. Find S_i such that $S_i\{R\}$, at least sometimes:

         i. Choose S_i that (i) is suggested by R
            & (ii) is simple (3.1)

         ii. Look at Q for clue to find S_i such that $S_i\{R\}$ (5)

            - (α) if Q is part of R, then use *skip* (5.1)

      ii. Find B_i such that $B_i \Rightarrow wp(S_i, R)$; then use $B_i \rightarrow S_i$

         i. Compute $wp(S_i, R)$ & try $B_i = wp(S_i, R)$ (3.2)

         ii. By Thm 10.5, each B_i is such that $Q \wedge B_i \Rightarrow wp(S_i, R)$
            $\therefore$ if Q and $wp(S_i, R)$ are known,
            then B_i := whatever must be conjoined to Q to imply $wp(S_i, R)$ (6)

      or:

      i$'$**.** Suppose you have guards $B_1, \ldots, B_{n-1}$;
      find B_n such that $Q \Rightarrow (B_1 \vee \ldots \vee B_n)$.

      ii$'$**.** Find S_n such that $Q \wedge B_n \Rightarrow wp(S_n, R)$

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

   (b) 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). (7)

4. To develop a DO-program:

   (a) Determine P:

      i. Try $P := R$ with a conjunct deleted. (12)

         A. Delete a conjunct that makes P easy to establish by initialization. (12.1)

      ii. Try $P :=$ (R with a constant replaced by a variable) $\wedge$ $(l.b. \leq var \leq u.b.)$ (15, 16)

         A. Choose a constant that: makes P easy to establish (15.1.1)
            & yields simple guards (15.1.2)
            & makes S_i easy to determine (15.1.3)

      iii. Try $P := R$ with the range of a variable enlarged. (18)

      iv. Both Q and R should imply P.

   (b) Determine $t \geq 0$ as a function of variables and/or constants in Q and R.

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

   (d) Develop guarded commands $B_i \rightarrow S_i$:

      i. **repeat:**

         A. Develop the guards:

            - (α) Find X such that $P \wedge X \Rightarrow R$; then let $B := \neg X$ (9.1)
            - (β) If $P := R$ with conjunct C deleted, then let $B := \neg C$ (13)
            - (γ) Find B_i such that $P \wedge B_i \Rightarrow wp(S_i, P)$ (17)

         B. For each B_i, develop body S_i that (a) decrements t
            (i.e., makes progress towards halting)
            while (b) preserving (i.e., restoring) P (9.2)

            - (α) To find a $\left\{ \begin{array}{c} \text{minimum} \\ \text{maximum} \end{array} \right\}$ value $\left\{ \begin{array}{c} \geq \text{ some l.b. } L \\ \leq \text{ some u.b. } U \end{array} \right\}$ that satisfies some property, look at values starting at $\left\{ \begin{array}{c} L \\ U \end{array} \right\}$ in $\left\{ \begin{array}{c} \text{increasing} \\ \text{decreasing} \end{array} \right\}$ order. (Linear Search Principle 14)

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

      ii. All other things being equal, use *lots* of guarded commands. (19)

      iii. 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, *and* so that it is easier to prove $P \wedge \neg(B_1 \vee \ldots \vee B_n) \Rightarrow R$). (10, 19)

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

         - (α) If a guard can be strengthened to F, then delete it. (11.2.1)