MORE FUEL FOR THE GOTO CONTROVERSY

B. N. Dickman
Bell Telephone Laboratories
Piscataway, New Jersey

Structured, goto-free, coding in a PL/I-like language with the BREAK, ITERATE, and CASE statements has been a requirement placed on several programming groups. In examining a large sample of programs, structured and with gotos, several points have been noted. First, recall that it has been shown in several papers that a program with gotos may be mechanically transformed into a goto-free program (assuming appropriate language constructs) by means of "auxiliary variables" (flags, switches) and by means of replicating blocks of code. Of course this is presented as a theoretical result; no one would call the programs thus transformed good structured programs. Unfortunately this is exactly how some programmers program to avoid the discipline of structured programming. Structured programming is encouraged to provide top-to-bottom flow of control and locality of control structure and data use. These goals are effectively defeated if programmers simulate gotos by setting and testing flags and switches everywhere. Maintainability is hurt by the replication of blocks of code.

In fact, given the usual structured programming features, the opportunity for truly horrendous programming is increased. Consider the following sequence of statements.

```
DO LINK1 = 0 TO 3
   DO CASE LINK1
      CASE 0
         A = 73
      CASE 1
         B = 74
      CASE 2
         C = 23
      CASE 3
         D = 55
   DOEND
DOEND
```

which is equivalent to

```
A = 73
B = 74
C = 23
D = 55
```

The code at each CASE was simplified here for the purposes of the example, but, otherwise is as taken from a working production program. (In the original program too there was only straight line code at each CASE.) The same construct occurred at least twice in a random sample of fourteen programs. This is not to say that structured programming should not be encouraged, but only that it provides as much "leverage" for bad programming as for good programming.