DIJKSTRA78
Dijkstra E.W.; Guarded Commands, Nondeterminacy and Formal Derivation of Programs; CACM 18, 8 (August 1978).

SYNOPSIS

Conceptually the paper is divided into two parts, in the first the author introduces the concept of a 'guarded command', and alternative and repetitive language constructs using these guarded commands which allow nondeterministic program execution. The author then formalizes the semantics of these constructs to allow the formal derivation of programs. He gives the following BNF grammar for guarded commands:

```
<statement> := <alternative> | <repetitive> | "other statements"
<alternative> := if <guarded command set> fi
<repetitive> := do <guarded command set> od
<guarded command set> := <guarded command> {[] <guarded command>}
<guarded command> := <guard> --> <guarded list>
<guard> := <boolean expression>
<guarded list> := <statement> { ; <statement>}
```

Informally the semantics are as follows:

A ';' means that the statements that the semicolon separates will be executed sequentially.

A 'guarded list' will only be executed if its guard is true.

An 'alternative' (if statement) will abort if none of its guards are true otherwise the guarded list associated with one of the true guards will be arbitrarily selected for execution.

A 'repetitive' (do statement) will terminate when all its guards are false otherwise on each iteration the guarded list associated with one of the true guards will be arbitrarily selected for execution.

The author then gives several examples, concluding with the following example which shows that not only is the order of computation nondeterministic but also the final state of the computation is not uniquely determined.

"Determine k such that for a fixed value n > 0 and a function f(i) defined for 0 <= i < n, k will eventually satisfy
(0 <= k < n and (for all 0 <= i < n : f(k) >= f(i)))". With solution

```
k := 0;   j := 1;
do
  j != n --> if
                 f(j) <= f(k)  --> j := j+1
              [] f(k) >= f(j)  --> k := j; j := j+1
             fi
od
```

The next sections of the paper define the formal semantics in a manner directly inspired by Hoare's axiomatic method but differing in the fact that Dijkstra's pre-conditions guarantee a correct result and program termination whereas Hoare's do not guarantee termination. He then presents a formal derivation of two simple programs from their post conditions.

```
"( m=x or m=y ) and m >= x and m >= y"
     /* m = max(x,y) */  derives

if
      x >= y -->  m := x
   [] y >= x -->  m := y
fi

and " gcd(X,Y) = gcd(x,y) and x > 0 and y > 0 " derives

x := X;    y := Y;
do
      x > y --> x := x - y
   [] y > x --> y := y - x
od

which computes x = gcd(X,Y)
```

COMMENTS

The idea of guarded commands has allready been adopted by other languages (cf. Hoare's CSPs) and represents a viable synthesis of a selection mechanism and concurrency. The method for the derivation of programs however seems to be on the one hand too complex to be done by the programmer for a large program and on the other hand to be too intuitive to be automated efficiently.