Guarded Commands, Nondeterminancy, and Formal Derivation of Programs

E.W. Dijkstra, CACM 18,8 pp.453-457, (Aug. 1975).

A Discipline of Programming, *Prentice-Hall, 1976.*

Concurrent assignment *vs* ***Sequential assignment***

$x, y := y, x$	$x := y; y := x$
	$y := x; x := y$
$t := x; x := y; y := t$	

$x_1, x_2, \ldots, x_n := E_1, E_2, \ldots, E_n$	*Concurrent assignment statement(S)*
$S_1; S_2; \ldots, S_n$	*Statements List(SL)*

Two ***separators*** *; and , has different meanings*

Alternative statement and Nondeterminancy

if $x \geq y \rightarrow m := x$
$|$ $x \leq y \rightarrow m := y$
fi.

Nondeterminancy of alternative statement

if $x{=}y \rightarrow m := x \vee m := y$

if fi $\equiv$ ***abort***

Repeatative statement

$q_1, q_2, q_3, q_4 := Q_1, Q_2, Q_3, Q_4;$

do $q_1 > q_2 \rightarrow q_1, q_2 := q_2, q_1$
$|$ $q_2 > q_3 \rightarrow q_2, q_3 := q_3, q_2$
$|$ $q_3 > q_4 \rightarrow q_3, q_4 := q_4, q_3$
od.

do od $\equiv$ ***skip***

Syntax of ***alternatve*** *and* ***repeatative*** *statements*

if $B_1 \rightarrow SL_1 \mid B_2 \rightarrow SL_2 \mid \ldots \mid B_n \rightarrow SL_n$ ***fi***.

do $B_1 \rightarrow SL_1 \mid B_2 \rightarrow SL_2 \mid \ldots \mid B_n \rightarrow SL_n$ ***od***.

P	*Loop invariance condition*
BB	*There exists at least one guard that is true*

$P = 1 \leq \forall i \leq 4$: q_i's are permutation of Q_i

$\neg BB = \neg((q_1 > q_2) \vee (q_2 > q_3) \vee (q_3 > q_4))$

$= \neg(q_1 > q_2) \wedge \neg(q_2 > q_3) \wedge \neg(q_3 > q_4))$

$= (q_1 \leq q_2) \wedge (q_2 \leq q_3) \wedge (q_3 \leq q_4)$

$= (q_1 \leq q_2 \leq q_3 \leq q_4)$

*P is true before the loop(**initialization** of P),*
*P remains true in the loop(loop **invariance**), and*
*P is also true after the loop terminates(loop **terminating**),*
then $P \wedge \neg BB$ is true after the loop.
for (init; BB; update) ... in C

Given $n > 0$ and $0 \le \forall i < n$: f(i) is defined.
Determine k .э. $0 \le k < n \wedge (\forall i: 0 \le i < n: f(k) \ge f(i))$.

$k, j := 0, 1;$
***do** $j \neq n \rightarrow$ **if** $f(j) \le f(k) \rightarrow j := j+1$*
$\mid f(j) \ge f(k) \rightarrow k := j; j := j+1$ vs $k, j := j, j+1$
fi
***od**.*

Formal Derivation of Programs

$m = max(x, y)$

$R: (m = x \vee m = y) \wedge m \geq x \wedge m \geq y.$

"$m := x$" $R_m^x = (x = x \vee x = y) \wedge x \geq x \wedge x \geq y \equiv x \geq y.$

$\textbf{if } x \geq y \rightarrow m := x \textbf{ fi}$

"$m := y$" $R_m^y = (y = x \vee y = y) \wedge y \geq x \wedge y \geq y \equiv y \geq x.$

$\textbf{if } y \geq x \rightarrow m := y \textbf{ fi}$

$\textbf{if } x \geq y \rightarrow m := x$
$\mid y \geq x \rightarrow m := y$
$\textbf{fi}.$

Given two positive numbers X and Y, find x .э. x = gcd(X, Y)
Loop invariance P: $gcd(X, Y) = gcd(x, y) \wedge x > 0 \wedge y > 0$.
Initialization of the loop invariance P

$x := X; y := Y$ or $x, y := X, Y$

Do something under the loop invariance of P

$gcd(x, y) = gcd(x–y, y)$ or $gcd(x, y–x)$

"$x := x–y$" $P_x^{x-y} = gcd(X, Y) = gcd(x–y, y) \wedge x–y > 0 \wedge y > 0 = x > y$.

do $x > y \rightarrow x := x - y$ **od**.

"$y := y–x$" $P_y^{y-x} = gcd(X, Y) = gcd(x, y–x) \wedge x > 0 \wedge y–x > 0 = y > x$.

do $y > x \rightarrow y := y - x$ **od**.

$x, y := X, Y;$
do $x > y \rightarrow x := x - y$
 $| \; x < y \rightarrow y := y - x$
od.

*Does the loop **terminate**?*

$|x - y|$ is a non-negative monotonically decreasing function.

Loop terminates when $x=y$, i.e., $|x - y|$ becomes zero.

if $X > 0$ ***and*** $Y > 0 \rightarrow$
    $x, y := X, Y;$
    do $x > y \rightarrow x := x - y$
      $| \; x < y \rightarrow y := y - x$
    od
fi.