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***

$x, y := y, x$ $\qquad\qquad$ $x := y; y := x$

$\qquad\qquad\qquad\qquad\qquad\quad$ $y := x; x := y$

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

$x_1, x_2, \ldots, x_n := E_1, E_2, \ldots, E_n$ $\qquad$ *Concurrent assignment statement(S)*

$S_1; S_2; \ldots, S_n$ $\qquad\qquad\qquad\qquad$ *Statements List(SL)*

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

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*

Alternative statement and Nondeterminancy

if $x \geq y \rightarrow m := x$

$\mid \; x \leq y \rightarrow m := y$

fi.

Nondeterminancy of alternative statement

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

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

Repeatative construct

$i, S := 1, 0;$ **do** $i \leq 100 \rightarrow S := S + i;\ i := i + 1$ **od**

$P = [1 \leq {}^{\forall} i \leq 101: S = \Sigma_{j=1}^{i-1}\ j]$

$\wedge\ (i = 101) \Rightarrow S = \Sigma_{j=0}^{100}\ j$

$i, S := 0, 0;$ **do** $i \leq 99 \rightarrow i := i + 1;\ S := S + i$ **od**

$P = [0 \leq {}^{\forall} i \leq 100: S = \Sigma_{j=0}^{i}\ j]$

$\wedge\ (i\ = 100) \Rightarrow S = \Sigma_{j=0}^{100}\ j$

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

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 ≡ skip

$P = 1 \le {}^{\forall} i \le 4$: q_i's are permutation of Q_i

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

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

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

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

*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 .ə. $0 \le k < n \wedge ({}^{\forall} i: 0 \le i < n: f(k) \ge f(i))$.

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

Formal Derivation of Programs

$m = max(x, y)$

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

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

if $x \geq y \rightarrow m := x$ **fi**

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

if $y \geq x \rightarrow m := y$ **fi**

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

Given two positive numbers X and Y, find x .э. x = gcd(X, Y)
Loop invariance P: gcd(X, Y) = gcd(x, y) ∧ x > 0 ∧ 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 $x > y \rightarrow x := x - y$
$\quad | \; x < y \rightarrow y := y - x$
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 $x > y \rightarrow x := x - y$

$\mid x < y \rightarrow y := y - x$

od

fi.