

Programming Calculus

“Proof of Correctness”

Dijkstra and Gries

Notation

- Assignment $\{P[e \rightarrow x]\} x := e \{P\}$
- Composition
 $\{P1\} S1 \{P2\}, \{P2\} S2 \{P3\}$
 $\{P1\} S1 S2 \{P3\}$
- Alternation $B1 \text{ or } \dots \text{ or } Bn$
 $\{P \wedge B1\} Si \{R\} 1 \leq i < n$
 $\{P\} \text{ if } B1 \rightarrow S1 \dots Bn \rightarrow Sn \text{ fi } \{R\}$
-one B_i must be true, S_i is executed

Notation 2

- Iteration $\{P \text{ and } B\} S \{P\}$
 - P is invariant relation (does not change)
 - $\{P \wedge B \wedge t \leq t_0 + 1\} S \{t \leq t_0\}$
 - S must reduce t by at least 1
 - $(P \wedge B) \Rightarrow t > 0$
 - $\{P\} \text{ do } B \rightarrow S \text{ or } \{P \wedge \neg B\}$
 - When $t \leq 0$ then $B \leftarrow \text{false}$

Notation 3

WP (S,R)

Statement S – give a rule for developing the weakest precondition for which execution of s will established the post condition R

- must derive the precondition P from S and R
- S is a predicate transformer which transforms P to R

- Step 1 – Input and Output Assertions
 - pre and post condition
- Step 2 – Loops
 - develop invariant relation
 - initialize, loop body, termination
- Step 3 – Think in terms of “While” loop not For
- Step 4 – Work backwards – start at post condition
- Step 5 – Limit nested statements
- Step 6 – Document program

Let B_i be initial values in $b[i]$

Input assertion

$$(3) s \geq 0 \wedge b[i] = B_i \quad 1 \leq i \leq n$$

Form of line (w_1 – word 1)

$W_1[1] \ W_2[1] \ \dots \ W_n[s]$

$$b[1] = 1; \ b[2] = b[1] + \text{len}(W_1) + 1$$

Output form

$$(4) W_1[p+1] \ W_2[p+1] \ \dots [p+1] \ W_t[q+1] \ \dots [q+1] \ W_n$$

Restrictions p,q,t

$$1 \leq t \leq n$$

$$p \geq 0, q \geq 0$$

$$p(t-1) + q(n-t) = s$$

$$(5) ((\text{odd } z) \wedge q=p+1) \vee ((\text{even } Z) \wedge p=q+1)$$

from 3

$$(6) \left\{ \begin{array}{l} b[i] = B_i + p(i-1) \quad 1 \leq i \leq t \\ b[i] = B_i + p(t-1) + q(i-t) \quad t < i \leq n \end{array} \right\}$$

post condition – 6

Algorithm

{(3)}

Calculate p,q,t from (5)

{(3) ^ (5)}

Calculate b[1:t] from (6)

$$\left\{ \begin{array}{l} (5) \wedge b[i] = B_i + p(i-1) \quad 1 \leq i \leq t \\ \wedge b[i] = B_i \quad t < i \leq n \end{array} \right\}$$

Calculate new b[t+1 : n] from (6)

(7) {(5) ^ (6)}

Algorithm 2

Refine Calculations $b[1:t]$

- need loop counter k
- invariant relation $P1$

$P1: 1 \leq k \leq t \text{ incr} = p * (k-1) \quad (5)$

$$b[i] = B_i + p * (i-1) \quad 1 \leq i \leq k$$

(8) $b[i] = B_i \quad k < i \leq n$

Algorithm 3

I initial $k = 1; \text{incr} = 0$

loop S: $k=k+1; \text{incr} = \text{incr} + P; b[k] = b[k] + \text{incr}$

Choose $B \ni P1 \wedge >B$ is postcondition

$\neg B \equiv \text{incr} = p * (t-1) \quad \left. \vphantom{\text{use } B} \right\} \text{ since}$
use $B \equiv \text{incr} <> p * (t-1) \quad \left. \vphantom{\text{use } B} \right\} \text{ p may be 0}$

```

loop:
{calculate b[1:t] from (6)}
k := 1
incr = 0
While incr <> p * (t-1)
    k=k+1
    incr = incr + p
    b[k] = b[k] + incr

P2 t < k ≤ n incr = p * (t-1) + q * (k-t) (5)
b[i] = Bi + p * (i-1) 1 ≤ i < t
b[i] = Bi + p * (t-1) + q * (i-t) t < i ≤ k
b[i] = Bi k < i ≤ n

```

```

{Calculate b[t+1:n]}
loop2 k = t
While k <> n do
    k = k + 1
    incr = incr + q
    b[k] = b[k] + incr

{Calculate p,q,t}
even (z) → p = q + 1
(q+1) * (t-1) + q * (n-t) = s (5)
0 ≤ t-1 ≤ n-1

(10) {
    q = s div (n-1)
    t = s + 1 - q * (n-1)
    p = q + 1

```

Derive Wp execution of (10)
will establish
 $\text{even}(z) \wedge s \geq 0 \wedge (s \text{ div } (n-1) \geq 0)$
(must have at least 2 words)

```
if n ≤ 1 → skip
else if n > 1 → if (even z) → q = s div (n-1)
                    t = s + 1 - q * (n-1)
                    p = q + 1
                    if (odd z) → p = s div (n-1)
                                q = p + 1
                                t = p * (n-1) + n - s
                                loop 1
                                loop 2
                    fi
fi
```