Chapter 7: Formalizing Programming Principles

The Basic Technique: Slope Search

Programming Problem:

Given an array

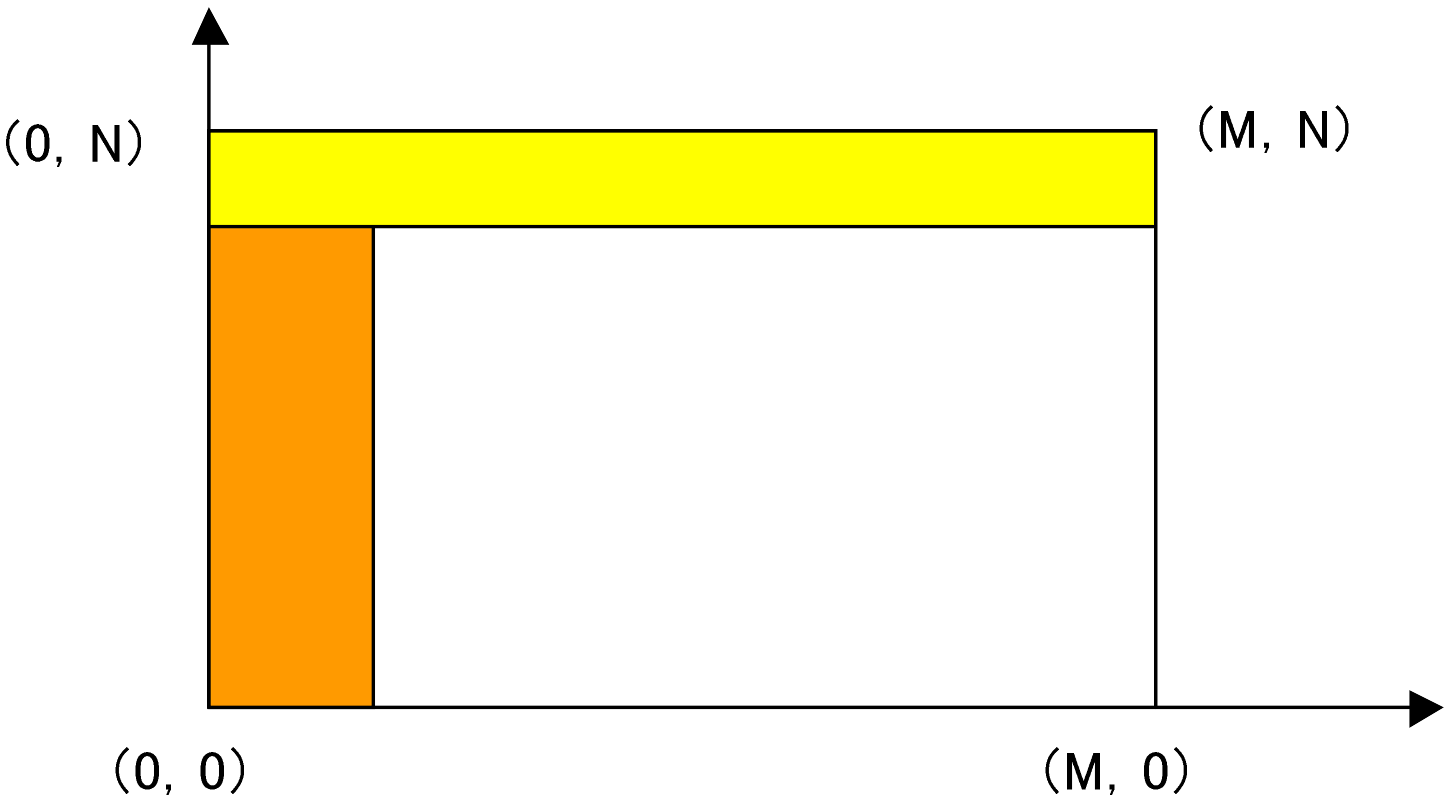
$$f : [0..M] \times [0..N] \to \mathcal{Z}$$

which is ascending in both arguments, and assuming X occurs in f, derive a program that establishes for integer a and b such that

$$f.a.b = X.$$

The post condition is:

$$R: \ 0 \leq a \leq M \ \wedge \ 0 \leq b \leq N \ \wedge \ f.a.b = X$$



Choosing (0,N) for inspection:

$$
\begin{array}{lcl}
f.0.N > X & \Rightarrow & (\forall i : 0 \leq i \leq M : f.i.N > X \\
f.0.N < X & \Rightarrow & (\forall j : 0 \leq j \leq N : f.0.j < X
\end{array}
$$

To be formal, we define (I, J) be the point satisfying

$$R:\ 0 \le I \le M \ \wedge\ 0 \le J \le N \ \wedge\ f.I.J = X$$

and choose the following invariant:

$$P:\ 0 \le a \le I \ \wedge\ J \le b \le N$$

which is established by $a, b := 0, N$.

$$\begin{array}{ll} & f.a.b < X \\ \Rightarrow & \quad \{\ f \text{ is ascending in its second argument, } J \le b\ \} \\ & f.a.J < X \\ \Rightarrow & \quad \{\ f.I.J = X\ \} \\ & a \neq I \\ \equiv & \quad \{\ a \le I\ \} \\ & a + 1 \le I \end{array}$$

So

$$P \wedge f.a.b < X \Rightarrow P(a := a + 1)$$

$$
\begin{array}{ll}
 & f.a.b > X \\
\Rightarrow & \quad \{ \ f \text{ is ascending in its second argument, } a \leq I \ \} \\
 & f.I.b > X \\
\Rightarrow & \quad \{ \ f.I.J = X \ \} \\
 & b \neq J \\
\equiv & \quad \{ \ a \leq I \ \} \\
 & J \leq b - 1
\end{array}
$$

So

$$P \wedge f.a.b > X \Rightarrow P(b := b - 1)$$

This yields the following slope searching solution:

$$\begin{array}{l} a, b := 0, N; \\ \textbf{do}\ f.a.b < X \rightarrow a := a + 1 \\ \ [\,]\ f.a.b > X \rightarrow b := b - 1 \\ \textbf{od} \end{array}$$

It has time complexity of $\mathcal{O}(M + N)$. A similar program can be obtained by choosing $(M, 0)$ as starting point.

Application 1: Searching

Given an array

$$f : [0..M] \times [0..N] \to \mathcal{Z}$$

which is ascending in both arguments, determine whether value X occurs in f.

$|[$
con $M, N, X : int;$
$f :$ **array** $[0..M) \times [0..N)$ **of** $int;$
$\{f$ is ascending in both arguments$\}$
var $r : bool;$
S
$\{r \equiv (\exists i, j : 0 \leq i < M \wedge 0 \leq j < N : f.i.j = X)\}$
$]|$

Following the approach of "slope searching", we define a "tail" invariant:

$$G.a.b \equiv (\exists i, j : a \leq i < M \wedge 0 \leq j < b : f.i.j = X)$$

So:

$$R : \; r \equiv G.0.N$$

$$P_0 : \; r \vee G.a.b \equiv G.0.N$$
$$P_1 : \; 0 \leq a \leq M \wedge 0 \leq b \leq N$$

It is not difficult to prove that

$$P \wedge (a = M \vee b = 0 \vee r) \; \Rightarrow R$$

What is the guard?

Investigating an increase of a by 1.

$$
\begin{array}{ll}
 & G.a.b \\
\equiv & \quad \{ \text{ definition of } G \text{ } \} \\
 & (\exists i,j : a \le i < M \wedge 0 \le j < b : f.i.j = X) \\
\equiv & \quad \{ \text{ split off } i = a \text{ } \} \\
 & G.(a+1).b \vee (\exists j : 0 \le j < b : f.a.j = X) \\
\equiv & \quad \{ \text{ assuming } f.a.(b-1) < X, \ f \text{ is ascending in its second argument } \} \\
 & G.(a+1).b \vee \mathit{false} \\
\equiv & \quad \{ \text{ predicate calculus } \} \\
 & G.(a+1).b
\end{array}
$$

Hence,

$$f.a.(b-1) < X \;\Rightarrow\; (G.a.b = G.(a+1).b)$$

Similarly(?), we have

$$f.a.(b-1) > X \;\Rightarrow\; (G.a.b = G.a.(b-1))$$

And for the case $f.a.(b-1) = X$, we have

$$P_0 \wedge (f.a.(b-1) = X) \;\Rightarrow\; P_0(r := \mathit{true})$$

Our final program:

$$
\begin{array}{l}
|[\\
\textbf{var } a, b : \mathit{int}; \\
a, b, r := 0, N, \mathit{false}; \\
\textbf{do } a \neq M \wedge b \neq 0 \wedge \neg r \rightarrow \\
\quad \textbf{if } f.a.(b-1) < X \rightarrow a := a+1; \\
\quad [] \; f.a.(b-1) > X \rightarrow b := b-1; \\
\quad [] \; f.a.)b-1) = X \rightarrow r := \mathit{true} \\
\quad \textbf{fi} \\
\textbf{od} \\
]|
\end{array}
$$

The time complexity is $\mathcal{O}(M+N)$. How to show this algorithm is optimal?

Application 2: Decomposition in a sum of two squares

Derive a program for the computation of the number of ways in which a natural number N can be written as the sum of two squares.

$$
\begin{array}{l}
|[\\
\textbf{con}\ N : \mathit{int}\ \{N \geq 0\}; \\
\textbf{var}\ r : \mathit{int}; \\
S \\
\{r = (\#x, y : 0 \leq x \leq y : x^2 + y^2 = N)\} \\
]|
\end{array}
$$

Since $x^2 + y^2$ is increasing in both arguments, we define

$$G.a.b = (\#x, y : a \leq x \leq y \leq b : x^2 + y^2 = N)$$

Hence

$$\begin{aligned} R: &\quad r = G.0.N \\ P_0: &\quad r + G.a.b = G.0.N \\ P_1: &\quad 0 \leq a \end{aligned}$$

The invariants can be established by:

$$a, b, r := 0, \sqrt{N}, 0$$

Investigate the effect of increase of a by 1.

$$
\begin{array}{ll}
 & G.a.b \\
= & \quad \{ \text{ definition of } G \ \} \\
 & (\#x, y : a \leq x \leq y \leq b : x^2 + y^2 = N) \\
= & \quad \{ \text{ split } x = a \ \} \\
 & G.(a+1).b + (\#y.a \leq y \leq b : a^2 + y^2 = N) \\
= & \quad \{ \ a \leq b \ \} \\
 & G.(a+1).b + 0, \quad \text{if } a^2 + b^2 < N \\
 & G.(a+1).b + 1, \quad \text{if } a^2 + b^2 = N
\end{array}
$$

Investigate the effect of decrease of b by 1.

$$
\begin{array}{ll}
 & G.a.b \\
= & \quad \{ \text{ definition of } G \ \} \\
 & (\#x, y : a \le x \le y \le b : x^2 + y^2 = N) \\
= & \quad \{ \text{ split } y = b \ \} \\
 & G.a.(b-1) + (\#x.a \le x \le b : x^2 + b^2 = N) \\
= & \quad \{ \ a \le b \ \} \\
 & G.a.(b-1) + 0, \quad \text{if } a^2 + b^2 > N \\
 & G.a.(b-1) + 1, \quad \text{if } a^2 + b^2 = N
\end{array}
$$

Our final program:

$$
\begin{array}{l}
|[\\
\textbf{var}\ a, b : \mathit{int}; \\
r, a := 0, 0; b := 0;\ \ \textbf{do}\ b * b < N \rightarrow b := b + 1\ \textbf{od}; \\
\textbf{do}\ a \leq b \rightarrow \\
\qquad \textbf{if}\ a * a + b * b < N \rightarrow a := a + 1 \\
\qquad [\,]\ a * a + b * b > N \rightarrow b := b - 1 \\
\qquad [\,]\ a * a + b * b = N \rightarrow r, a := r + 1, a + 1 \\
\qquad [\,]\ a * a + b * b = N \rightarrow r, b := r + 1, b - 1 \\
\quad \textbf{fi} \\
\textbf{od} \\
]|
\end{array}
$$

The time complexity is $\mathcal{O}(\sqrt{N})$.

Application 3: Minimal distance

Derive a program for the computation of the minimal distance of two ascending sequences.

$$
\begin{array}{l}
|[\\
\textbf{con } M, N : \mathit{int}\ \{M \geq 0 \wedge N \geq 0\}; \\
f :\ \textbf{array } [0..M)\ \textbf{of } \mathit{int}\ \{f \text{ is ascending}\}; \\
g :\ \textbf{array } [0..N)\ \textbf{of } \mathit{int}\ \{g \text{ is ascending}\}; \\
\textbf{var } r : \mathit{int}; \\
S \\
\{r = (\textbf{min } x, y : 0 \leq x < M \wedge 0 \leq y < N : |f.x - g.y|)\} \\
]|
\end{array}
$$

Note that $f.x - g.y$ is increasing in x and decreasing in y, but $|f.x - g.y|$ does not have this property. Anyway, slope search is still possible.

Define $G.a.b$ by

$$G.a.b = (\mathbf{min}\ x,y : a \le x < M \wedge b \le y < N : |f.x - g.y|)$$

So

$$\begin{aligned} R :&\quad r = G.0.0 \\ P_0 :&\quad r\ \mathbf{min}\ G.a.b = G.0.0 \\ P_1 :&\quad 0 \le a \le M \wedge 0 \le b \le N \end{aligned}$$

The invariants can be established by

$$a, b, r = 0, 0, \infty$$

Furthermore

$$P_0 \wedge (a = M \vee b = N) \ \Rightarrow\ R$$

$$
\begin{array}{cl}
 & G.a.b \\
= & \quad \{ \text{ definition of } G \ \} \\
 & (\mathbf{min}\ x, y : a \le x < M \wedge b \le y < N : |f.x - g.y|) \\
= & \quad \{ \text{ split off } x = a \ \} \\
 & G.(a+1).b \ \mathbf{min}\ (\mathbf{min}\ y : b \le y < N : |f.a - g.y|) \\
= & \quad \{ \text{ assuming } g.b \ge f.a,\ g \text{ is increasing } \} \\
 & G.(a+1).b \ \mathbf{min}\ (\mathbf{min}\ y : b \le y < N : g.y - f.a) \\
= & \quad \{ \ g \text{ is increasing } \} \\
 & G.(a+1).b \ \mathbf{min}\ (g.b - f.a)
\end{array}
$$

Hence,

$$g.b \ge f.a \ \Rightarrow \ G.a.b = G.(a+1).b \ \mathbf{min}\ (g.b - f.a)$$

Symmetrically,

$$f.a \ge g.b \ \Rightarrow \ G.a.b = G.a.(b+1) \ \mathbf{min}\ (f.a - g.b)$$

Longest Segments

Determine the length of a longest segment of $X[0..N)$ that satisfies $\mathcal{A}$.

$$\begin{array}{l}
|[\\
\textbf{con}\ N : \mathit{int}\ \{N \geq 0\};\ X : \textbf{array}\ [0..N)\ \textbf{of}\ \mathit{int}; \\
\textbf{var}\ r : \mathit{int}; \\
\mathit{all\ zeros} \\
\{r = (\textbf{max}\ p, q : 0 \leq p \leq q \leq N \wedge \mathcal{A}.p.q :\ q - p)\} \\
]|
\end{array}$$

Examples of $\mathcal{A}$ are

$$\mathcal{A}.p.q = (\forall i, j : p \le i < q \land p \le j < q : X.i = X.j)$$
$$\mathcal{A}.p.q = (\forall i, j : p \le i \le j < q : X.i \le X.j)$$
$$\mathcal{A}.p.q = (\# i : p \le i < q : X.i = 0) \le 60$$

For these examples, $\mathcal{A}$ has the following properties.

- A holds for empty segments: $A.n.n$
- A is prefix-closed: $A.p.q \Rightarrow (\forall i : p \le i \le q : A.p.i)$
- A is postfix-closed: $A.p.q \Rightarrow (\forall i : p \le i \le q : A.i.q)$

for the function $q - p$ which is ascending in q and descending in p, we may apply "slope search" approach.

$$G.a.b = (\mathbf{max}\ p, q : a \le p \le q \le N \wedge b \le q \le N \wedge \mathcal{A}.p.q :\ q - p)$$

And

$$\begin{aligned} R :& \quad r = G.0.0 \\ P_0 :& \quad r\ \mathbf{max}\ G.a.b = G.0.0 \\ P_1 :& \quad 0 \le a \le b \le N \end{aligned}$$

The invariants can be established by

$$a, b, r := 0, 0, 0$$

What is a guard: $P_0 \wedge ? \Rightarrow R$

To find p,

$$(r\ \mathbf{max}\ G.a.b = G.0.0)\ \wedge\ p?\ \Rightarrow r = G.0.0$$

we calculate as follows.

$$\begin{array}{ll}
 & G.a.N \\
= & \quad \{\ \text{definition of } G\ \} \\
 & (\mathbf{max}\ p,q : a \le p \le q \le N \wedge N \le q \le N \wedge \mathcal{A}.p.q :\ \ q - p) \\
= & \quad \{\ \text{calculation}\ \} \\
 & (\mathbf{max}\ p : a \le p \le q \le N \wedge \mathcal{A}.p.N :\ \ N - p) \\
= & \quad \{\ \text{assume } \mathcal{A}.a.N,\ N - p \text{ is descending in } p\ \} \\
 & N - a
\end{array}$$

Hence

$$\begin{array}{lll}
(r\ \mathbf{max}\ G.a.b = G.0.0)\ \wedge\ b = N\ \wedge\ \mathcal{A}.a.b & \Rightarrow & r\ \mathbf{max}\ (N - a) = G.0.0 \\
 & \equiv & R(r := r\ \mathbf{max}\ (N - a))
\end{array}$$

We investigate the effect of increase of a and b on the invariants, and obtain

$$\begin{aligned} \mathcal{A}.a.b &\Rightarrow G.a.b = G.a.(b+1) \ \mathbf{max} \ (b-a) \\ \neg\mathcal{A}.a.b &\Rightarrow G.a.b = G.(a+1).b \end{aligned}$$

The final program:

$$
\begin{array}{l}
|[\\
\textbf{var } a, b : \mathit{int}; \\
a, b, r := 0, 0, 0; \\
\textbf{do } b \neq N \vee \neg \mathcal{A}.a.b \rightarrow \\
\qquad \textbf{if } \mathcal{A}.a.b \rightarrow r, b := r \textbf{ max } (b - a), b + 1 \\
\qquad [] \ \neg \mathcal{A}.a.b \rightarrow a := a + 1 \\
\qquad \textbf{fi} \\
\textbf{od} \\
r := r \textbf{ max } (N - a) \\
]|
\end{array}
$$

Exercises

Problem 8:

Derive an efficient program S satisfying the following specification:

$$
\begin{array}{l}
|[\\
\mathbf{con}\ N : int\ \{N \geq 0\};\ \mathbf{var}\ r : bool; \\
S \\
\{r \equiv (\exists x, y : 0 \leq x \wedge 0 \leq y : N = 2^x + 3^y)\} \\
]|
\end{array}
$$