Chapter 5: Deriving Efficient Programs

Two examples are used to show how one can reason about programs in a non-operational way.

Integer Division

Design efficient *divmod* meeting the specification:

$$
\begin{array}{l}
|[\\
\textbf{con}\ A, B : \mathit{int}\ \{A \geq 0 \wedge B > 0\}\\
\textbf{var}\ q, r : \mathit{int}\\
\mathit{divmod}\\
\{q = A\ \textbf{div}\ B \wedge r = A\ \textbf{mod}\ B\}\\
]|
\end{array}
$$

Note that according to the definitions of **div** and **mod**, the post-condition R is

$$R: \; A = q * B + r \wedge 0 \leq r \wedge r < B.$$

We have seen (Lecture 4) that by choosing as invariant

$$P: \; A = q * B + r \;\wedge\; 0 \leq r$$

we can obtain the following solution to *divmod*:

$$\begin{array}{l}
q, r, := 0, A; \\
\{\text{invariant: } A = q * B + r \;\wedge\; 0 \leq r, \text{ bound: } r\} \\
\mathbf{do}\ r \geq B \rightarrow q, r := q + 1, r - B\ \mathbf{od} \\
\{R\}
\end{array}$$

This program takes $\mathcal{O}(A\ \mathbf{div}\ B)$ steps.

Could we do better?

Yes! We can have a program using about half of the steps by doubling B.

$$
\begin{array}{l}
S_1; \\
\{R_1 :\ A = q * \underline{2 * B} + r \wedge 0 \leq r \wedge r < \underline{2 * B}\} \\
S_2; \\
\{R :\ A = q * B + r \wedge 0 \leq r \wedge r < B\}
\end{array}
$$

What are S_1 and S_2?

For S_1, just replace B by $2 * B$ in the previous program:

$$
\begin{array}{l}
q, r, := 0, A; \\
\{\text{invariant: } A = p * 2 * B + r \ \wedge\ 0 \leq r, \text{ bound: } r\} \\
\mathbf{do}\ r \geq 2 * B \rightarrow q, r := q + 1, r - 2 * B\ \mathbf{od} \\
\{R_1 :\ A = q * \underline{2 * B} + r \wedge 0 \leq r \wedge r \leq \underline{2 * B}\}
\end{array}
$$

For S_2, we simply have

$$
\begin{array}{l}
q := 2 * q; \\
\mathbf{if}\ B \leq r \rightarrow q, r := q + 1, r - B \\
\ [] \ r < B \rightarrow \mathit{skip} \\
\mathbf{fi} \\
\{R :\ A = q * B + r \wedge 0 \leq r \wedge r < B\}
\end{array}
$$

Could we do much better?

Yes! Repeat the better method, by replacing constant B by variable b.

So our invariants are:

$$\begin{array}{ll} P_0: & A = q * b + r \wedge 0 \leq r \wedge r < b \\ P_1: & b = 2^k * B \wedge 0 \leq k \end{array}$$

which are established by the following repetition:

$$\begin{array}{l} q, r, b, k := 0, A, B, 0; \\ \mathbf{do}\ r \geq b \rightarrow b, k := b * 2, k + 1\ \mathbf{od}. \end{array}$$

Next, we investigate the effect of $b := b \textbf{ div } 2$ on the invariants.

$$
\begin{array}{ll}
 & P_0 \wedge P_1 \\
= & \quad \{ \text{ definitions of } P_0 \text{ and } P_1\text{, substitution } \} \\
 & A = q * b + r \ \wedge\ 0 \leq r \ \wedge\ r < b \\
 & \qquad \wedge\ b = 2^k * B \ \wedge\ 0 \leq k \\
= & \quad \{ \text{ heading for } b : b \textbf{ div } 2 \ \} \\
 & A = (q * 2) * (b \textbf{ div } 2) + r \ \wedge\ 0 \leq r \ \wedge\ r < 2 * (b \textbf{ div } 2) \\
 & \qquad \wedge\ (b \textbf{ div } 2) = 2^{k-1} * B \ \wedge\ 0 \leq k \\
= & \quad \{ \text{ assume } b \neq B \ \} \\
 & A = (q * 2) * (b \textbf{ div } 2) + r \ \wedge\ 0 \leq r \ \wedge\ r < 2 * (b \textbf{ div } 2) \\
 & \qquad \wedge\ (b \textbf{ div } 2) = 2^{k-1} * B \ \wedge\ 0 \leq k - 1
\end{array}
$$

Hence,

$$\{P_0 \wedge P_1 \wedge b \neq B\}$$
$$q, b, k := q * 2, b \ \mathbf{div}\ 2, k - 1;$$
$$\{A = q * b + r \ \wedge \ 0 \leq r \ \wedge \ r < \underline{2 * b} \ \wedge \ b = 2^k * B \ \wedge \ 0 \leq k\}$$

It is easy to establish $P_0 \wedge P_1$ by

$$\{A = q * b + r \ \wedge\ 0 \leq r \ \wedge\ r < \underline{2 * b} \ \wedge\ b = 2^k * B \ \wedge\ 0 \leq k\}$$

$$\begin{array}{l} \textbf{if } b \leq r \rightarrow q, r := q + 1, r - b \\ \ [] \ r < b \rightarrow \mathit{skip} \\ \textbf{fi} \end{array}$$

$$\{A = q * b + r \ \wedge\ 0 \leq r \ \wedge\ r < \underline{b} \ \wedge\ b = 2^k * B \ \wedge\ 0 \leq k\}$$

Final program:

$$
\begin{array}{l}
|[\\
\mathbf{var}\ b, k : \mathit{int}; \\
q, r, b, k := 0, A, B, 0; \\
\mathbf{do}\ r \geq b \rightarrow b, k := b * 2, k + 1\ \mathbf{od}; \\
\mathbf{do}\ b \neq B \rightarrow \\
\qquad q, b, k := q * 2, b\ \mathbf{div}\ 2, k - 1; \\
\qquad \mathbf{if}\ b \leq r \rightarrow q, r := q + 1, r - b \\
\qquad\ [\,]\ r < B \rightarrow \mathit{skip} \\
\qquad \mathbf{fi} \\
\mathbf{od} \\
]|
\end{array}
$$

What is its time complexity? What is k for?

We could not need to introduce k if we change the invariants to

$$\begin{array}{ll} P_0: & A = q * b + r \wedge 0 \leq r \wedge r < b \\ P_1: & (\exists k : 0 \leq k : b = 2^k * B) \end{array}$$

Can you calculate your efficient program according to these invariants?

Fibonacci

Derive an $\mathcal{O}(\log N)$ program for *fibonacci* specified by

$$
\begin{array}{l}
|[\\
\textbf{con } N : \mathit{int} \ \{N \geq 0\}; \\
\textbf{var } x : \mathit{int}; \\
\mathit{fibonacci} \\
\{x = fib.N\} \\
]|
\end{array}
$$

where fib is defined by

$$
\begin{array}{lcl}
fib.0 & = & 0 \\
fib.1 & = & 1 \\
fib.(n+2) & = & fib.n + fib.(n+1)
\end{array}
$$

We have shown that by choosing

$$\begin{array}{ll} P_0 & x = fib.n \\ P_1 & 0 \leq n \leq N \\ Q & y = fib.(n+1) \end{array}$$

as invariants, we can arrive at the program

$$\begin{array}{l} |[\\ \textbf{var}\ n, y : int; \{N \geq 0\} \\ n, x, y := 0, 0, 1; \\ \{\text{invariant: } P_0 \wedge P_1 \wedge Q, \text{ bound: } N - n\} \\ \textbf{do}\ n \neq N \rightarrow x, y, n := y, x + y, n + 1\ \textbf{od} \\ \{x = fib.N\ \wedge\ y = fib.(N+1)\} \\ ]| \end{array}$$

which has the complexity of $\mathcal{O}(N)$.

In fact, we can obtain the following $\mathcal{O}(\log N)$ program:

$\{N > 0\}$

$|[$

$\textbf{var}\ a, b, n, y : \mathit{int};$

$a, b, x, y, n := 0, 1, 0, 1, N;$

$\textbf{do}\ n \neq 0 \rightarrow$

$\quad \textbf{if}\ n\ \textbf{mod}\ 2 = 0 \rightarrow a, b, n\ := \ a * a + b * b, a * b + b * a + b * b, n\ \textbf{div}\ 2$

$\quad [\!]\ n\ \textbf{mod}\ 2 = 1 \rightarrow x, y, n\ := \ a * x + b * y, b * x + a * y + b * y, n - 1$

$\quad \textbf{fi}$

$\textbf{od}$

$\{x = fib.N\}$

$]|$

Can you understand it, and say it is correct?

Recall that we have obtained:

$$
\begin{array}{l}
|[\\
\mathbf{var}\ n, y : \mathit{int}; \{N \geq 0\} \\
n, x, y := 0, 0, 1; \\
\mathbf{do}\ n \neq N \rightarrow x, y, n := y, x + y, n + 1\ \mathbf{od} \\
\{x = fib.N\ \wedge\ y = fib.(N+1)\} \\
]|
\end{array}
$$

and observe that $x, y := y, x + y$ is a linear combination of x and y:

$$\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

We thus have

$$
\begin{array}{l}
|[\\
\textbf{var } n, y : \mathit{int}; \{N \geq 0\} \\
n, x, y := 0, 0, 1; \\
\textbf{do } n \neq N \rightarrow \\
\qquad \begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}; \\
\qquad n := n + 1 \\
\textbf{od} \\
\{\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^N \begin{pmatrix} 0 \\ 1 \end{pmatrix}\} \\
]|
\end{array}
$$

Following our derivation for computing exponentiation, we have

$$
\begin{array}{l}
|[\\
\mathbf{var}\ n, y : \mathit{int}; \{N \geq 0\} \\
n, x, y := N, 0, 1; \\
A := \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}; \\
\mathbf{do}\ n \neq 0 \rightarrow \\
\qquad \mathbf{if}\ n\ \mathbf{mod}\ 2 = 0 \rightarrow A := A * A; n := n\ \mathbf{div}\ 2 \\
\qquad [\,]\ n\ \mathbf{mod}\ 2 = 1 \rightarrow \begin{pmatrix} x \\ y \end{pmatrix} := A \begin{pmatrix} x \\ y \end{pmatrix};\ n := n - 1; \\
\qquad \mathbf{fi} \\
\mathbf{od}
\end{array}
$$

We can go further by eliminating matrix operations, with the fact that A is always in the form $\begin{pmatrix} a & b \\ b & a+b \end{pmatrix}$. Indeed,

$$\begin{pmatrix} a & b \\ b & a+b \end{pmatrix}\begin{pmatrix} a & b \\ b & a+b \end{pmatrix} = \begin{pmatrix} p & q \\ q & p+q \end{pmatrix}$$

where

$$\begin{aligned} p &= a^2 + b^2 \\ q &= ab + ba + b^2 \end{aligned}$$

So $A := A * A$ corresponds to

$$a, b := a^2 + b^2, ab + ba + b^2$$

and $\begin{pmatrix} x \\ y \end{pmatrix} := A \begin{pmatrix} x \\ y \end{pmatrix}$ corresponds to

$$x, y := a * x + b * y, b * x + a * y + b * y.$$

And we thus abtain the program shown before.

Exercises

[Problem 6-1] Derive a program that has time complexity $\mathcal{O}(\log N)$ for

$$
\begin{array}{l}
|[\\
\textbf{con } N : int\ \{N \geq 1\};\ f : \textbf{array } [0..N] \textbf{ of } int\ \{f.0 < f.N\}; \\
\textbf{var } x : int; \\
S \\
\{0 \leq x < N \wedge f.x < f.(x+1)\} \\
]|
\end{array}
$$

by introducing variable y and invariants

$$
\begin{array}{ll}
P_0 : & f.x < f.y \\
P_1 : & 0 \leq x < y \leq N
\end{array}
$$

[Problem 6-2] Derive an $\mathcal{O}(\log N)$ algorithm for *square root*:

$$
\begin{array}{l}
|[\\
\textbf{con } N : \mathit{int} \ \{N \geq 0\}; \\
\textbf{var } x : \mathit{int}; \\
\mathit{square\ root} \\
\{x^2 \leq N \wedge (x+1)^2 > N\} \\
]|
\end{array}
$$

by introducing variables y and k and invariants:

$$
\begin{array}{ll}
P_0 : & x^2 \leq N \wedge (x+y)^2 > N \\
P_1 : & y = 2^k \wedge 0 \leq k
\end{array}
$$

[Problem 6-3] Solve

$$
\begin{array}{l}
|[\\
\textbf{con}\ A, B, N : int\ \{N \geq 0\}; \\
\textbf{var}\ x : int; \\
S \\
\{x = (\Sigma i : 0 \leq i \leq N : A^{N-i} * B^{i})\} \\
]|
\end{array}
$$

[Problem 6-4] Solve

$$
\begin{array}{l}
|[\\
\mathbf{con}\ N : int\ \{N \geq 0\}; \\
\mathbf{var}\ x : int; \\
Fibolucci \\
\{x = (\Sigma i : 0 \leq i \leq N : fib.i * fib.(N-i)\} \\
]|
\end{array}
$$

where fib is defined by

$$
\begin{array}{lcl}
fib.0 & = & 0 \\
fib.1 & = & 1 \\
fib.(n+2) & = & fib.n + fib.(n+1).
\end{array}
$$