Programming Languages: Imperative Program Construction
Practicals 7: Loop Constuction III

Shin-Cheng Mu

Autumn Term, 2021

1. Solve:

    > **con** $A, B : Int\{A \geqslant 0 \wedge B \geqslant 0\}$;
    > **var** $r : Int$;
    > S
    > $\{r = A \times B\}$,

    using only $(/2)$ (integral division by two), $(\times 2)$, *even*, *odd*, addition, and subtraction.

    Solution: Use the invariant

    $$r + a \times b = A \times B \wedge 0 \leqslant a \wedge 0 \leqslant b,$$

    with initialisation $r, a, b := 0, A, B$. If b is even:

    $$
    \begin{aligned}
    & r + a \times b \\
    = \; & r + a \times 2 \times (b/2) \\
    = \; & r + (a \times 2) \times (b/2).
    \end{aligned}
    $$

    If b is odd:

    $$
    \begin{aligned}
    & r + a \times b \\
    = \; & r + a \times (1 + (b - 1)) \\
    = \; & (r + a) + a \times (b - 1).
    \end{aligned}
    $$

    The program:

    > **con** $A, B : Int\{A \geqslant 0 \wedge B \geqslant 0\}$
    > **var** $r, a, b : Int$
    >
    > $r, a, b := 0, A, B$
    > $\{r + a \times b = A \times B \wedge 0 \leqslant a \wedge 0 \leqslant b, \; bnd : b\}$
    > **do** $b > 0 \wedge even\ b \rightarrow a, \; b \; := \; a \times 2, \; b/2$
    > $\quad | \;\; b > 0 \wedge odd\ b \rightarrow r, \; b \; := \; r + a, \; b - 1$
    > **od**
    > $\{r = A \times B\}$.

    ---

2. The sum of all digits of a natural number can be computed by

$sd\ 0 = 0$
$sd\ x = x\ \%\ 10 + sd\ (x\ /\ 10)$,

where ($/$) is integral division and $a\ \%\ b$ computes the remainder of $a\ /\ b$. Solve

con N : *Int* $\{0 \leqslant N\}$
var r : *Int*
?
$\{r = sd\ N\}$

3. Given a natural number N, derive a program that computes the number of factors 3 of N. For example, when $N = 945 = 3^3 \times 5 \times 7$ we output 3.

con N : *Int* $\{0 \leqslant N\}$
var r : *Int*
?
$\{r = $ how do you write the post condition?$\}$

4. Solve:

con N, X : *Int* $\{0 \leqslant N\}$
con f : **array** $[0..N)$ **of** *Int*
var r : *Int*
?
$\{r = \langle \Sigma i : 0 \leqslant i < N : f[i] \times X^i \rangle\}$

We have seen this problem before but let us do it slightly differently this time. (This problem is not that much about associativity, but a practice constructing and using recursive function definition.)

(a) Define $g\ n = \langle \Sigma i : n \leqslant i < N : f[i] \times X^{i-n} \rangle$ for $0 \leqslant n \leqslant N$, derive a recursive definition of g.

Solution: For an easy base case, $g\ N = \langle \Sigma i : N \leqslant i < N : f[i] \times X^{i-n} \rangle = 0$. For $0 \leqslant n < N$ we calculate:

$g\ n$
$= \langle \Sigma i : n \leqslant i < N : f[i] \times X^{i-n} \rangle$
$=$ { since $0 \leqslant n < N$, split off $i = n$ }
$\quad f[n] \times X^{n-n} + \langle \Sigma i : n+1 \leqslant i < N : f[i] \times X^{i-n} \rangle$
$=$ { with $n+1 \leqslant i < N$, arithmetics }
$\quad f[n] + \langle \Sigma i : n+1 \leqslant i < N : f[i] \times X^{i-(n+1)} \times X \rangle$
$=$ { distributivity }
$\quad f[n] + X \times \langle \Sigma i : n+1 \leqslant i < N : f[i] \times X^{i-(n+1)} \rangle$
$= f[n] + X \times g\ (n+1)$.

Therefore we conclude:

$g\ N = 0$
$g\ n = f[n] + X \times g\ (n+1)$, if $0 \leqslant n < N$.

(b) Use $r = g\ n$ as the main invariant, construct a program that solves the problem.

Solution: Introduce a new variable n and use $r = g\ n \wedge 0 \leqslant n \leqslant N$ as the main invariant, and use $n \neq 0$ as the loop guard since $r = g\ 0$ is the postcondition we want, which can be satisfied by initialising $r, n := 0, N$.

We decrease the bound by $n := n - 1$. To find out how to update r we calculate:

$$
\begin{aligned}
& (g\ n)[n \backslash n - 1] \\
=\ & f[n - 1] + X \times g\ n \\
=\ & \quad \{\ r = g\ n \wedge 0 \leqslant n \leqslant N \wedge n \neq 0\ \} \\
& f[n - 1] + X \times r\ .
\end{aligned}
$$

The resulting program (supplementary proofs omitted for now):

> **con** $N, X : Int\ \{0 \leqslant N\}$
> **con** $f : $ **array** $[0..N)$ **of** Int
> **var** $r, n : Int$
>
> $r, n := 0, n$
> $\{r = g\ n \wedge 0 \leqslant n \leqslant N, bnd : n\}$
> **do** $n \neq 0 \rightarrow$
> $\quad r := f[n - 1] + X \times r$
> $\quad n := n - 1$
> **od**
> $\{r = \langle \Sigma i : 0 \leqslant i < N : f[i] \times X^i \rangle\}$

5. The function *fusc* is defined on natural numbers by:

$$
\begin{aligned}
fusc\ 0 \quad &= \ 0 \\
fusc\ 1 \quad &= \ 1 \\
fusc\ (2 \times n) \quad &= \ fusc\ n \\
fusc\ (2 \times n + 1) \quad &= \ fusc\ n + fusc\ (n + 1).
\end{aligned}
$$

Derive a program computing *fusc* N for $N \geqslant 0$. Hint: try *fusc* 78.

Solution: Use the invariant

$$a \times fusc\ n + b \times fusc\ (n + 1) = fusc\ N\ \wedge\ 0 \leqslant n \leqslant N,$$

which can be established by $a, b, n := 1, 0, N$. When n is even (let $n = 2 \times m$):

$$
\begin{aligned}
& a \times fusc\ n + b \times fusc\ (n + 1) \\
=\ & a \times fusc\ (2 \times m) + b \times fusc\ (2 \times m + 1) \\
=\ & a \times fusc\ m + b \times fusc\ m + b \times fusc\ (m + 1) \\
=\ & (a + b) \times fusc\ m + b \times fusc\ (m + 1) \\
=\ & (a + b) \times fusc\ (n\ \mathbf{div}\ 2) + b \times fusc\ (n\ \mathbf{div}\ 2 + 1).
\end{aligned}
$$

When n is odd (let $n = 2 \times m + 1$):

$$
\begin{aligned}
& a \times fusc\ n + b \times fusc\ (n + 1) \\
=\ & a \times fusc\ (2 \times m + 1) + b \times fusc\ (2 \times m + 2) \\
=\ & a \times fusc\ m + a \times fusc\ (m + 1) + b \times fusc\ (m + 1) \\
=\ & a \times fusc\ m + (a + b) \times fusc\ (m + 1) \\
=\ & a \times fusc\ (n\ \mathbf{div}\ 2) + (a + b) \times fusc\ (n\ \mathbf{div}\ 2 + 1).
\end{aligned}
$$

When $n = 0$, we have $b = fusc\ N$.

The program:

> **con** $N : Int\ \{N \geqslant 0\}$
> **var** $a, b, n : Int$
>
> $a, b, n := 1, 0, N$
> $\{a \times fusc\ n + b \times fusc\ (n + 1) = fusc\ N \ \wedge\ 0 \leqslant n \leqslant N,\ \ bnd : n\}$
> **do** $n > 0 \wedge even\ n \rightarrow a,\ n\ :=\ a + b,\ n\ \textbf{div}\ 2$
> $\mid\ n > 0 \wedge odd\ n \rightarrow b,\ n\ :=\ a + b,\ n\ \textbf{div}\ 2$
> **od**
> $\{b = fusc\ N\}$.

6. Solve:

> **con** $N : Int\ \{0 \leqslant N\}$
> **con** $f : \textbf{array}\ [0..N)\ \textbf{of}\ Int$
> **var** $r : Bool$
> ?
> $\{r = \langle \exists i : 0 \leqslant i < N : f[i] = 0\rangle\}$

(a) Define, for $0 \leqslant n \leqslant N$, $g\ n = \langle \exists i : n \leqslant i < N : f[i] = 0\rangle$. Come up with a recursive definition of g.

> **Solution:** We have $g\ N = \textit{False}$. For $0 \leqslant n < N$,
>
> $\qquad g\ n$
> $= \langle \exists i : n \leqslant i < N : f[i] = 0\rangle$
> $= \quad \{\ 0 \leqslant n < N,\ \text{split off } i = n\ \}$
> $\quad f[n] = 0 \vee \langle \exists i : n + 1 \leqslant i < N : f[i] = 0\rangle$
> $= f[n] = 0 \vee g\ (n + 1)$.
>
> Therefore
>
> $\qquad g\ N \equiv \textit{False}$
> $\qquad g\ n \equiv f[n] = 0 \vee g\ (n + 1),\ \ \text{if } 0 \leqslant n < N.$

(b) Try come up with a program that, as soon as a zero is found in the array, terminates without having to scan the entire list. What invariant would you choose?

> **Solution:** Define
>
> $\qquad P \equiv (r \vee g\ n) = \langle \exists i : 0 \leqslant i < N : f[i] = 0\rangle$.
>
> and use $P \wedge 0 \leqslant n \leqslant N$ as the invariant. It can be established by the initialisation $r, n := \textit{False}, 0$.
> To allow early termination we use $\neg\ r \wedge n \neq N$ as the loop guard, since
>
> $\qquad \neg\ (\neg\ r \wedge n \neq N) \wedge P \wedge 0 \leqslant n \leqslant N$
> $\equiv \quad \{\ \text{de Morgan}\ \}$
> $\quad (r \vee n = N) \wedge P \wedge 0 \leqslant n \leqslant N$
> $\equiv \quad \{\ \text{distributivity}\ \}$
> $\quad (r \wedge P \wedge 0 \leqslant n \leqslant N) \vee (n = N \wedge P)$.

Consider the branch $n = N \wedge P$:

$$n = N \wedge ((r \vee g\ n) = \langle \exists i : 0 \leqslant i < N : f[i] = 0 \rangle)$$
$$\equiv (r \vee g\ N) = \langle \exists i : 0 \leqslant i < N : f[i] = 0 \rangle$$
$$\equiv \quad \{ g\ N = \mathit{False} \}$$
$$r = \langle \exists i : 0 \leqslant i < N : f[i] = 0 \rangle \quad .$$

Consider the branch $r \wedge P \wedge 0 \leqslant n \leqslant N$:

$$r \wedge P \wedge 0 \leqslant n \leqslant N$$
$$\Rightarrow r \wedge ((r \vee g\ n) = \langle \exists i : 0 \leqslant i < N : f[i] = 0 \rangle)$$
$$\equiv \quad \{ \text{ replace } r \text{ by } \mathit{True}, \text{ and } \mathit{True} \vee g\ n = \mathit{True} \}$$
$$r \wedge \langle \exists i : 0 \leqslant i < N : f[i] = 0 \rangle$$
$$\Rightarrow r = \langle \exists i : 0 \leqslant i < N : f[i] = 0 \rangle \quad .$$

Therefore $\neg\,(\neg\,r \wedge n \neq N) \wedge P \wedge 0 \leqslant n \leqslant N$ implies $r = \langle \exists i : 0 \leqslant i < N : f[i] = 0 \rangle$.

Let the bound bd $N - n$ and let the last statement of the loop be $n := n + 1$. Consider:

$$(r \vee g\ n)[n \backslash n + 1]$$
$$= r \vee g\ (n + 1)$$

If we apply a substitution $[r \backslash r \vee f[n] = 0]$ we get

$$((r \vee g\ n)[n \backslash n + 1])[r \backslash r \vee f[n] = 0]$$
$$= (r \vee g\ (n + 1))[r \backslash r \vee f[n] = 0]$$
$$= (r \vee f[n] = 0) \vee g\ (n + 1)$$
$$= \quad \{ \text{ disjunction associative } \}$$
$$r \vee (f[n] = 0 \vee g\ (n + 1))$$
$$= \quad \{ \text{ calculation above } \}$$
$$r \vee g\ n \quad .$$

Therefore we get $(P[n \backslash n + 1])[r \backslash r \vee f[n] = 0] = P$.

In conclusion, the program can be

> **con** $N : \mathit{Int}\ \{0 \leqslant N\}$
> **con** $f : \mathbf{array}\ [0..N)\ \mathbf{of}\ \mathit{Int}$
> **var** $r : \mathit{Bool}$
> **var** $n : \mathit{Int}$
>
> $r, n := \mathit{False}, 0$
> $\{P \wedge 0 \leqslant n \leqslant N, \mathit{bnd} : N - n\}$
> **do** $\neg\, r \wedge n \neq N \rightarrow$
> $r := r \vee f[n] = 0$
> $n := n + 1$
> **od**
> $\{r = \langle \exists i : 0 \leqslant i < N : f[i] = 0 \rangle\}$

Supplementary proofs omitted for now.