

Programming Languages: Imperative Program Construction

Practicals 7: Loop Constuction III

Shin-Cheng Mu

Autumn Term, 2021

1. Solve:

```

con  $A, B : \text{Int}\{A \geq 0 \wedge B \geq 0\};$ 
var  $r : \text{Int};$ 
 $S$ 
 $\{r = A \times B\}$  ,
    
```

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

Solution: Use the invariant

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

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  $A, B : \text{Int}\{A \geq 0 \wedge B \geq 0\}$ 
var  $r, a, b : \text{Int}$ 

 $r, a, b := 0, A, B$ 
 $\{r + a \times b = A \times B \wedge 0 \leq a \wedge 0 \leq b, \text{ bnd} : b\}$ 
do  $b > 0 \wedge \text{even } b \rightarrow a, b := a \times 2, b/2$ 
     $| \ b > 0 \wedge \text{odd } b \rightarrow r, b := r + a, b - 1$ 
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

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

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.

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

4. Solve:

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

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

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

$$\begin{aligned}
 &g\ n \\
 &= \langle \sum i : n \leq i < N : f[i] \times X^{i-n} \rangle \\
 &= \{ \text{since } 0 \leq n < N, \text{ split off } i = n \} \\
 &\quad f[n] \times X^{n-n} + \langle \sum i : n+1 \leq i < N : f[i] \times X^{i-n} \rangle \\
 &= \{ \text{with } n+1 \leq i < N, \text{ arithmetics} \} \\
 &\quad f[n] + \langle \sum i : n+1 \leq i < N : f[i] \times X^{i-(n+1)} \times X \rangle \\
 &= \{ \text{distributivity} \} \\
 &\quad f[n] + X \times \langle \sum i : n+1 \leq i < N : f[i] \times X^{i-(n+1)} \rangle \\
 &= f[n] + X \times g\ (n+1) .
 \end{aligned}$$

Therefore we conclude:

$$\begin{aligned}
 &g\ N = 0 \\
 &g\ n = f[n] + X \times g\ (n+1), \text{ if } 0 \leq n < N.
 \end{aligned}$$

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

Solution: Introduce a new variable n and use $r = g\ n \wedge 0 \leq n \leq N$ 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:

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

The resulting program (supplementary proofs omitted for now):

```

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

```

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

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

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

Solution: Use the invariant

$$a \times \text{fusc } n + b \times \text{fusc } (n + 1) = \text{fusc } N \wedge 0 \leq n \leq N,$$

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

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

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

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

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

The program:

```

con  $N : \text{Int} \{N \geq 0\}$ 
var  $a, b, n : \text{Int}$ 

 $a, b, n := 1, 0, N$ 
 $\{a \times \text{fusc } n + b \times \text{fusc } (n + 1) = \text{fusc } N \wedge 0 \leq n \leq N, \text{ bnd} : n\}$ 
do  $n > 0 \wedge \text{even } n \rightarrow a, n := a + b, n \text{ div } 2$ 
     $| \ n > 0 \wedge \text{odd } n \rightarrow b, n := a + b, n \text{ div } 2$ 
od
 $\{b = \text{fusc } N\}$  .

```

6. Solve:

```

con  $N : \text{Int} \{0 \leq N\}$ 
con  $f : \text{array}[0..N] \text{ of } \text{Int}$ 
var  $r : \text{Bool}$ 
?
 $\{r = \langle \exists i : 0 \leq i < N : f[i] = 0 \rangle\}$ 

```

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

Solution: We have $g\ N = \text{False}$. For $0 \leq n < N$,

$$\begin{aligned}
 g\ n &= \langle \exists i : n \leq i < N : f[i] = 0 \rangle \\
 &= \{ 0 \leq n < N, \text{ split off } i = n \} \\
 &= f[n] = 0 \vee \langle \exists i : n + 1 \leq i < N : f[i] = 0 \rangle \\
 &= f[n] = 0 \vee g\ (n + 1) .
 \end{aligned}$$

Therefore

$$\begin{aligned}
 g\ N &\equiv \text{False} \\
 g\ n &\equiv f[n] = 0 \vee g\ (n + 1), \text{ if } 0 \leq n < N.
 \end{aligned}$$

(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

$$P \equiv (r \vee g\ n) = \langle \exists i : 0 \leq i < N : f[i] = 0 \rangle .$$

and use $P \wedge 0 \leq n \leq N$ as the invariant. It can be established by the initialisation $r, n := \text{False}, 0$.

To allow early termination we use $\neg r \wedge n \neq N$ as the loop guard, since

$$\begin{aligned}
 &\neg (\neg r \wedge n \neq N) \wedge P \wedge 0 \leq n \leq N \\
 &\equiv \{ \text{de Morgan} \} \\
 &= (r \vee n = N) \wedge P \wedge 0 \leq n \leq N \\
 &\equiv \{ \text{distributivity} \} \\
 &= (r \wedge P \wedge 0 \leq n \leq N) \vee (n = N \wedge P) .
 \end{aligned}$$

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

$$\begin{aligned} n &= N \wedge ((r \vee g \ n) = \langle \exists i : 0 \leq i < N : f[i] = 0 \rangle) \\ &\equiv (r \vee g \ N) = \langle \exists i : 0 \leq i < N : f[i] = 0 \rangle \\ &\equiv \{ g \ N = False \} \\ &= \langle \exists i : 0 \leq i < N : f[i] = 0 \rangle . \end{aligned}$$

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

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

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

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

$$\begin{aligned} &(r \vee g \ n)[n \setminus n + 1] \\ &= r \vee g \ (n + 1) \end{aligned}$$

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

$$\begin{aligned} &((r \vee g \ n)[n \setminus n + 1])[r \setminus r \vee f[n] = 0] \\ &= (r \vee g \ (n + 1))[r \setminus r \vee f[n] = 0] \\ &= (r \vee f[n] = 0) \vee g \ (n + 1) \\ &= \{ \text{disjunction associative} \} \\ &= r \vee (f[n] = 0 \vee g \ (n + 1)) \\ &= \{ \text{calculation above} \} \\ &= r \vee g \ n . \end{aligned}$$

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

In conclusion, the program can be

```

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

```

Supplementary proofs omitted for now.