

Programming Languages: Imperative Program Construction

Practicals 6: Loop Constuction II

Shin-Cheng Mu

Autumn Term, 2021

1. Recall the maximum segment sum problem. What if we want to compute the maximum sum of *non-empty* segments?

(a) How would you write the specification? Does the specification still make sense with N being constrained only by $0 \leq N$?

Solution: The specification could be:

```
con  $N : \text{Int } \{0 \leq N\}$ 
con  $f : \text{array } [0..N) \text{ of } \text{Int}$ 
 $S$ 
 $\{r = \langle \uparrow p \ q : 0 \leq p < q \leq N : \text{sum } p \ q \rangle\}$  ,
```

where the definition of *sum* is unchanged:

$$\text{sum } p \ q = \langle \sum i : p \leq i < q : f[i] \rangle .$$

When $N = 0$, $0 \leq p < q \leq N$ reduces to *False* and r should be $-\infty$. The specification is fine if $-\infty$ is a value allowed in our program.

(b) Derive a program solving the problem.

Solution: Like in the handouts, we start with

$$P_0 \equiv r = \langle \uparrow p \ q : 0 \leq p < q \leq n : \text{sum } p \ q \rangle ,$$

$$Q \equiv 0 \leq n \leq N ,$$

and use $N - n$ as bound.

To find out what we can do with r before $n := n + 1$, we calculate:

$$\begin{aligned} & \langle \uparrow p \ q : 0 \leq p < q \leq n : \text{sum } p \ q \rangle [n \setminus n + 1] \\ &= \langle \uparrow p \ q : 0 \leq p < q \leq n + 1 : \text{sum } p \ q \rangle \\ &= \{ \text{split off } q = n + 1 \text{ (safe since } 0 \leq n) \} \\ & \quad \langle \uparrow p \ q : 0 \leq p < q \leq n : \text{sum } p \ q \rangle \uparrow \langle \uparrow p : 0 \leq p < n + 1 : \text{sum } p \ (n + 1) \rangle . \end{aligned}$$

Therefore we add another invariant:

$$P_1 \equiv s = \langle \uparrow p : 0 \leq p < n : \text{sum } p \ n \rangle .$$

Variables r, s, n can be initialised by $-\infty, -\infty, 0$. To find out how to update s , we calculate:

$$\begin{aligned}
& \langle \uparrow p : 0 \leq p < n : \text{sum } p \ n \rangle [n \setminus n + 1] \\
&= \langle \uparrow p : 0 \leq p < n + 1 : \text{sum } p \ (n + 1) \rangle \\
&= \{ \text{split off } p = n \text{ (safe since } 0 \leq n) \} \\
& \langle \uparrow p : 0 \leq p < n : \text{sum } p \ (n + 1) \rangle \uparrow \text{sum } n \ (n + 1) \\
&= \{ \text{definition of sum, one-point rule} \} \\
& \langle \uparrow p : 0 \leq p < n : \text{sum } p \ (n + 1) \rangle \uparrow f[n] \\
&= \{ \text{split off } i = n \text{ in definition of sum, (safe since } 0 \leq n) \} \\
& \langle \uparrow p : 0 \leq p < n : \text{sum } p \ n + f[n] \rangle \uparrow f[n] \\
&= \{ p \text{ not free in } f[n] \} \\
& (\langle \uparrow p : 0 \leq p < n : \text{sum } p \ n \rangle + f[n]) \uparrow f[n] .
\end{aligned}$$

The constructed program is:

```

con  $N : \text{Int} \{0 \leq N\}$ 
con  $f : \text{array } [0..N) \text{ of Int}$ 
var  $r, n : \text{Int}$ 
 $r, s, n := -\infty, -\infty, 0$ 
 $\{P_0 \wedge P_1 \wedge Q, \text{bnd} : N - n\}$ 
do  $n \neq N \rightarrow$ 
   $s := (s + f[n]) \uparrow f[n]$ 
   $r := r \uparrow s$ 
   $n := n + 1$ 
od
 $\{r = \langle \uparrow p \ q : 0 \leq p < q \leq N : \text{sum } p \ q \rangle\}$ 

```

2. Recall the derivation of the maximum segment sum problem. Assuming that we had instead used the loop invariant $P_0 \wedge P_1 \wedge Q$, where

$$\begin{aligned}
P_0 &\equiv r = \langle \uparrow p \ q : 0 \leq p \leq q \leq n : \text{sum } p \ q \rangle , \\
P_1 &\equiv s = \langle \uparrow p : 0 \leq p \leq n + 1 : \text{sum } p \ (n + 1) \rangle , \\
Q &\equiv 0 \leq n \leq N .
\end{aligned}$$

Can you construct a program using the invariant above? What if the array is non-empty, that is, $1 \leq N$?

Solution: With $0 \leq N$ we would run into problem initialising s . The initialisation may look like:

$$\begin{aligned}
& r, s, n := 0, ?, 0 \\
& \{P_0 \wedge P_1 \wedge Q\}
\end{aligned}$$

and the value of s should be

$$\begin{aligned}
& P_1[r, s, n \setminus 0, ?, 0] \\
& \equiv ? = \langle \uparrow p : 0 \leq p \leq 1 : \text{sum } p \ 1 \rangle \\
& \equiv ? = \text{sum } 0 \ 1 + \text{sum } 1 \ 1 \\
& \equiv ? = f[0] + 0 \\
& \equiv ? = f[0] .
\end{aligned}$$

However, $f[0]$ does not have a value when $N = 0$.

When we have $1 \leq N$ instead of $0 \leq N$ we will be able to initialize the variables by $r, s, n := 0, f[0], 0$. When we construct the loop body, knowing that $P_0[n \setminus n + 1] \equiv r = r \uparrow \langle \uparrow p : 0 \leq p \leq (n + 1) : \text{sum } p \ (n + 1) \rangle$, the constructed loop body would be:

```

{P0 ∧ P1 ∧ 0 ≤ n ≤ N ∧ n ≠ N}
r := r ↑ s
{P0[n\ n+1] ∧ P1 ∧ 0 ≤ n+1 ≤ N}
s := (s + f[n+1]) ↑ 0
{(P0 ∧ P1 ∧ 0 ≤ n ≤ N)[n\ n+1]}
n := n + 1
{P0 ∧ P1 ∧ 0 ≤ n ≤ N}

```

Note that the order of assignments is different from that in the handouts.

However, upon termination (that is, $n = N$) we would need to establish

$$s = \langle \uparrow p : 0 \leq p \leq N+1 : \text{sum } p(N+1) \rangle ,$$

which we cannot do because $f[N]$ is not defined.

It is possible to fix all these: for example, terminate the loop one step earlier and do some post processing, and put the entire loop under an **if** to ensure that $1 \leq N$. The resulting program would not be as clean as the one in the handouts, though.

3. Derive a solution for:

```

con N : Int {N ≥ 0}; a : array [0..N) of Int
var r : Int
S
{r = ⟨↑ i, j : 0 ≤ i < j < N : a[i] - a[j]⟩} .

```

Solution: Replace constant N by variable n , and use a loop that increments n in each step. Use the following P as a candidate of the loop invariant

$$P : r = \langle \uparrow i, j : 0 \leq i < j < n : a[i] - a[j] \rangle .$$

To find out how to update r such that we may increment n , we calculate (assuming $0 \leq n$):

$$\begin{aligned}
& \langle \uparrow i, j : 0 \leq i < j < n+1 : a[i] - a[j] \rangle \\
&= \{ \text{since } 0 \leq n, \text{ split off } j = n \} \\
& \langle \uparrow i, j : 0 \leq i < j < n : a[i] - a[j] \rangle \uparrow \langle \uparrow i : 0 \leq i < n : a[i] - a[n] \rangle \\
&= \{ n \text{ not bounded} \} \\
& \langle \uparrow i, j : 0 \leq i < j < n : a[i] - a[j] \rangle \uparrow (\langle \uparrow i : 0 \leq i < n : a[i] \rangle - a[n]).
\end{aligned}$$

So we strengthen the invariant by adding a variable s satisfying

$$Q : s = \langle \uparrow i : 0 \leq i < n : a[i] \rangle .$$

The invariant is $P \wedge Q \wedge 0 \leq n \leq N$.

The program:

```

con N : Int {0 ≤ N}
con a : array [0..N) of Int
var r, s, n : Int
r, s, n := -∞, -∞, 0 -- Pf0
{P ∧ Q ∧ 0 ≤ n ≤ N, bnd : N - n} -- Pf1
do n ≠ N →
    r, s, n := r ↑ (s - a[n]), s ↑ a[n], n + 1 -- Pf2
od
{r = ⟨↑ i, j : 0 ≤ i < j < N : a[i] - a[j]⟩} -- Pf3

```

Here I am omitting other proofs and presenting only Pf2:

Pf2

$$\begin{aligned}
& (P \wedge Q \wedge 0 \leq n \leq N)[r, s, n \setminus r \uparrow (s - a[n]), s \uparrow a[n], n + 1] \\
& \equiv r \uparrow (s - a[n]) = \langle \uparrow i, j : 0 \leq i < j < n + 1 : a[i] - a[j] \rangle \wedge \\
& \quad s \uparrow a[n] = \langle \uparrow i : 0 \leq i < n + 1 : a[i] \rangle \wedge 0 \leq n + 1 \leq N \\
& \equiv \{ \text{split off } i = n \} \\
& \quad r \uparrow (s - a[n]) = \langle \uparrow i, j : 0 \leq i < j < n : a[i] - a[j] \rangle \uparrow (\langle \uparrow i : 0 \leq i < n : a[i] \rangle - a[n]) \wedge \\
& \quad s \uparrow a[n] = \langle \uparrow i : 0 \leq i < n : a[i] \rangle + a[n] \wedge 0 \leq n + 1 \leq N \\
& \Leftarrow r = \langle \uparrow i, j : 0 \leq i < j < n : a[i] - a[j] \rangle \wedge \\
& \quad s = \langle \uparrow i : 0 \leq i < n : a[i] \rangle \wedge 0 \leq n \leq N \wedge n \neq N \\
& \equiv P \wedge Q \wedge 0 \leq n \leq N \wedge n \neq N.
\end{aligned}$$

4. Derive a solution for:

```

con  $N : \text{Int}\{N \geq 1\}; a : \text{array}[0..N] \text{ of } \text{Int}$ 
var  $r : \text{Int}$ 
 $S$ 
 $\{r = \langle \#i, j : 0 \leq i < j < N : a[i] \times a[j] \geq 0 \rangle\}$  .

```

Solution: Replace constant N by variable n , and use a loop that increments n in each step. Let

$$P \equiv r = \langle \#i, j : 0 \leq i < j < n : a[i] \times a[j] \geq 0 \rangle.$$

We first attempt to use $P \wedge 0 \leq n \leq N$ as the invariant and, apparently, $N - n$ as the bound. To find out how to update r such that we may increment n , we calculate (assuming $0 \leq n$):

$$\begin{aligned}
& \langle \#i, j : 0 \leq i < j < n + 1 : a[i] \times a[j] \geq 0 \rangle \\
& = \{ \text{since } 0 \leq n, \text{ splitting off } j = n \} \\
& \quad \langle \#i, j : 0 \leq i < j < n : a[i] \times a[j] \geq 0 \rangle + \langle \#i : 0 \leq i < n : a[i] \times a[n] \geq 0 \rangle.
\end{aligned}$$

To further simplify $\langle \#i : 0 \leq i < n : a[i] \times a[n] \geq 0 \rangle$, we do a case analysis on $a[n]$:

$$\begin{aligned}
\langle \#i : 0 \leq i < n : a[i] \times a[n] \geq 0 \rangle = & \langle \#i : 0 \leq i < n : a[i] \geq 0 \rangle, \text{ if } a[n] > 0; \\
& n, \text{ if } a[n] = 0; \\
& \langle \#i : 0 \leq i < n : a[i] \leq 0 \rangle, \text{ if } a[n] < 0.
\end{aligned}$$

Thus we strengthen the invariant by adding two more variables:

$$\begin{aligned}
Q_1 : s_1 &= \langle \#i : 0 \leq i < n : a[i] \geq 0 \rangle, \\
Q_2 : s_2 &= \langle \#i : 0 \leq i < n : a[i] \leq 0 \rangle.
\end{aligned}$$

The invariant is $P \wedge Q_1 \wedge Q_2 \wedge 0 \leq n \leq N$.

The program:

```

con  $N : \text{Int}\{N \geq 1\}; a : \text{array}[0..N] \text{ of } \text{Int}$ 
var  $r, s_1, s_2 : \text{Int}$ 

 $r, s_1, s_2, n := 0, 0, 0, 0$ 
 $\{P \wedge Q_1 \wedge Q_2 \wedge 0 \leq n \leq N, \text{ bnd} : N - n\}$ 
do  $n < N \rightarrow$ 
  if  $a[n] > 0 \rightarrow r, s_1, n := r + s_1, s_1 + 1, n + 1$ 
  |  $a[n] = 0 \rightarrow r, s_1, s_2, n := r + n, s_1 + 1, s_2 + 1, n + 1$ 
  |  $a[n] < 0 \rightarrow r, s_2, n := r + s_2, s_2 + 1, n + 1$ 
fi
od
 $\{r = \langle \#i, j : 0 \leq i < j < N : a[i] \times a[j] \geq 0 \rangle\}$  .

```