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 $0 \leqslant N$?

   > **Solution:** The specification could be:
   >
   > **con** $N : Int \ \{0 \leqslant N\}$
   > **con** $f : $ **array** $[0..N)$ **of** Int
   > S
   > $\{r = \langle \uparrow p\ q : 0 \leqslant p < q \leqslant N : sum\ p\ q \rangle\}$
   >
   > When $N = 0$, $0 \leqslant p < q \leqslant N$ 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
   >
   > $P_0 \equiv r = \langle \uparrow p\ q : 0 \leqslant p < q \leqslant n : sum\ p\ q \rangle$,
   > $Q \equiv 0 \leqslant n \leqslant N$,
   >
   > and use $N - n$ as bound. To find out what we can do with r before $n := n + 1$, we calculate:
   >
   > $\quad \langle \uparrow p\ q : 0 \leqslant p < q \leqslant n : sum\ p\ q \rangle[n \backslash n + 1]$
   > $= \langle \uparrow p\ q : 0 \leqslant p < q \leqslant n + 1 : sum\ p\ q \rangle$
   > $= \quad \{$ split off $q = n + 1$ (safe since $0 \leqslant n$) $\}$
   > $\quad \langle \uparrow p\ q : 0 \leqslant p < q \leqslant n : sum\ p\ q \rangle \uparrow \langle \uparrow p : 0 \leqslant p < n + 1 : sum\ p\ (n + 1) \rangle$.
   >
   > Therefore we add another invariant:
   >
   > $P_0 \equiv s = \langle \uparrow p : 0 \leqslant p < n : sum\ p\ n \rangle$.
   >
   > Variables r, s, n can be initialised by $-\infty, -\infty, 0$. To find out how to update s, we calculate:
   >
   > $\quad \langle \uparrow p : 0 \leqslant p < n : sum\ p\ n \rangle[n \backslash n + 1]$
   > $= \langle \uparrow p : 0 \leqslant p < n + 1 : sum\ p\ (n + 1) \rangle$
   > $= \quad \{$ split off $p = n$ (safe since $0 \leqslant n$) $\}$
   > $\quad \langle \uparrow p : 0 \leqslant p < n : sum\ p\ (n + 1) \rangle \uparrow sum\ n\ (n + 1)$
   > $= \quad \{$ definition of sum, one-point rule $\}$
   > $\quad \langle \uparrow p : 0 \leqslant p < n : sum\ p\ (n + 1) \rangle \uparrow f[n]$
   > $= \quad \{$ split off $i = n$ in definition of sum, (safe since $0 \leqslant n$) $\}$
   > $\quad \langle \uparrow p : 0 \leqslant p < n : sum\ p\ n + f[n] \rangle \uparrow f[n]$
   > $= \quad \{ p$ not free in $f[n] \}$
   > $\quad (\langle \uparrow p : 0 \leqslant p < n : sum\ p\ n \rangle + f[n]) \uparrow f[n]$.

The constructed program is:

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

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

$$P_0 \equiv r = \langle \uparrow p\ q : 0 \leqslant p \leqslant q \leqslant n : sum\ p\ q \rangle \ ,$$
$$P_1 \equiv s = \langle \uparrow p : 0 \leqslant p \leqslant n + 1 : sum\ p\ (n+1) \rangle \ ,$$
$$Q \equiv 0 \leqslant n \leqslant N \ .$$

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

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

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

and the value of s should be

$\quad P_1[r, s, n \backslash 0, ?, 0]$
$\equiv ? = \langle \uparrow p : 0 \leqslant p \leqslant 1 : sum\ p\ 1 \rangle$
$\equiv ? = sum\ 0\ 1 + sum\ 1\ 1$
$\equiv ? = f[0] + 0$
$\equiv ? = f[0] \ .$

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

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

$\{P_0 \wedge P_1 \wedge 0 \leqslant n \leqslant N \wedge n \neq N\}$
$r := r \uparrow s$
$\{P_0[n \backslash n + 1] \wedge P_1 \wedge 0 \leqslant n + 1 \leqslant N\}$
$s := (s + f[n + 1]) \uparrow 0$
$\{(P_0 \wedge P_1 \wedge 0 \leqslant n \leqslant N)[n \backslash n + 1]\}$
$n := n + 1$
$\{P_0 \wedge P_1 \wedge 0 \leqslant n \leqslant N\}$

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

$$s = \langle \uparrow p : 0 \leqslant p \leqslant N + 1 : sum\ p\ (N + 1) \rangle \ ,$$

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 $1 \leqslant N$. The resulting program would not be as clean as the one in the handouts, though.

3. Derive a solution for:

> **con** $N : Int\{N \geqslant 0\}; a : $ **array** $[0..N)$ **of** Int
> **var** $r : Int$
> S
> $\{r = \langle \uparrow i, j : 0 \leqslant i < j < N : a[i] - a[j] \rangle\}$.

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

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

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

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

So we strengthen the invariant by adding a variable s satisfying

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

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

The program:

> **con** $N : Int \ \{0 \leqslant N\}$
> **con** $a : $ **array** $[0..N)$ **of** Int
> **var** $r, s, n : Int$
> $r, s, n := -\infty, -\infty, 0$ -- Pf0
> $\{P \wedge Q \wedge 0 \leqslant n \leqslant N, bnd : N - n\}$ -- Pf1
> **do** $n \neq N \rightarrow$
> $r, s, n := r \uparrow (s - a[n]), s \uparrow a[n], n + 1$ -- Pf2
> **od**
> $\{r = \langle \uparrow i\ j : 0 \leqslant i < j < N : a[i] - a[j] \rangle\}$ -- Pf3

Here I am omitting other proofs and presenting only Pf2:

Pf2

$$(P \wedge Q \wedge 0 \leqslant n \leqslant N)[r, s, n \backslash r \uparrow (s - a[n]),\ s \uparrow a[n],\ n + 1]$$

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

4. Derive a solution for:

> **con** $N : Int\{N \geqslant 1\}; a : $ **array** $[0..N)$ **of** Int
> **var** $r : Int$
> S
> $\{r = \langle \#i, j : 0 \leqslant i < j < N : a[i] \times a[j] \geqslant 0 \rangle \}$.

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

$$P \;\equiv\; r = \langle \#i, j : 0 \leqslant i < j < n : a[i] \times a[j] \geqslant 0 \rangle).$$

We first attempt to use $P \wedge 0 \leqslant n \leqslant N$ 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 $0 \leqslant n$):

$$\langle \#i, j : 0 \leqslant i < j < n + 1 : a[i] \times a[j] \geqslant 0 \rangle$$
$$= \; \{ \text{ since } 0 \leqslant n, \text{ splitting off } j = n \}$$
$$\langle \#i, j : 0 \leqslant i < j < n : a[i] \times a[j] \geqslant 0 \rangle + \langle \#i : 0 \leqslant i < n : a[i] \times a[n] \geqslant 0 \rangle.$$

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

$$\langle \#i : 0 \leqslant i < n : a[i] \times a[n] \geqslant 0 \rangle = \begin{array}{ll} \langle \#i : 0 \leqslant i < n : a[i] \geqslant 0 \rangle, & \text{if } a[n] > 0; \\ n, & \text{if } a[n] = 0; \\ \langle \#i : 0 \leqslant i < n : a[i] \leqslant 0 \rangle, & \text{if } a[n] < 0. \end{array}$$

Thus we strengthen the invariant by adding two more variables:

$$Q_1 \;:\; s_1 = \langle \#i : 0 \leqslant i < n : a[i] \geqslant 0 \rangle \;,$$
$$Q_2 \;:\; s_2 = \langle \#i : 0 \leqslant i < n : a[i] \leqslant 0 \rangle \;.$$

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

The program:

```
con N : Int{N ⩾ 1}; a : array [0..N) of Int
var r, s₁, s₂ : Int

r, s₁, s₂, n := 0, 0, 0, 0
{P ∧ Q₁ ∧ Q₂ ∧ 0 ⩽ n ⩽ N,  bnd : N − n}
do n < N →
   if a[n] > 0 → r, s₁, n := r + s₁, s₁ + 1, n + 1
   |  a[n] = 0 → r, s₁, s₂, n := r + n, s₁ + 1, s₂ + 1, n + 1
   |  a[n] < 0 → r, s₂, n := r + s₂, s₂ + 1, n + 1
   fi
od
{r = ⟨#i, j : 0 ⩽ i < j < N : a[i] × a[j] ⩾ 0⟩} .
```