Sample solution to Exercise 9: Generalizing a given subprogram

Refer to the statement of Exercise 9 in the 4/6L03 MRSD course materials.

0. Preliminary definitions

We begin by defining the following mathematical functions, which may be used in preconditions, postconditions and intermediate conditions (assertions), but not in expressions in the program itself:

Define: length(a1) is the length of the sequence (string) a1. The length may be zero.

Define: substring(a1, a2, a3) is the substring of the string a1 beginning in position a2 and a3 elements long. The values of a2 and a3 must be integers. Positions in the string a1 are numbered beginning with 1. The null string is allowed, both as a1 and as the result (a3=0). Therefore, meaningful values of a1, a2 and a3 satisfy the condition

$1 \le a2 \wedge 0 \le a3 \wedge a2+a3-1 \le length(a1)$

which is equivalent to

$1 \le a2 \le length(a1)+1 \wedge 0 \le a3 \le length(a1)-a2+1$

This condition will be used later, so we define it as another mathematical function:

Define: $C1(a1, a2, a3) = [1 \le a2 \le length(a1)+1 \wedge 0 \le a3 \le length(a1)-a2+1]$

Observation: The expression max(a, min(x, b)) limits x to a below and b above, provided that a≤b. I.e.:

max(a, min(x, b)) = a, if x<a,

= x, if a≤x≤b and

= b, if b<x.

This type of bounding the values of variables will be used below. Therefore, in order to make expressions shorter and more readable, we

Define: lim(a, x, b) = max(a, min(x, b))

Provided that a≤b it is always true that a≤lim(a, x, b)≤b.

1. Question 1: Specification of the given subprogram mid

From the description of the subprogram mid in Exercise 9 it follows that part of the precondition for mid is

$1 \le p2 \le p2+p3-1 \le length(p1)$ [abbreviated V1 below]

Note that V1 is equivalent to

$1 \leq p2 \leq \text{length}(p1) \wedge 1 \leq p3 \leq \text{length}(p1)\text{-}p2+1$

which gives lower and upper bounds on both p2 and p3.

It also follows from the description of mid in Exercise 9 that mid is called without formal parameter passing. The parameters are passed to and from mid via the program variables p1, p2, p3 and rmid. The input parameters p1, p2 and p3 must obviously exist in the data environment before mid is called. In Exercise 9 it is explicitly stated that the subprogram mid does not declare rmid, i.e. rmid must also exist in the data environment before mid is called.

The obvious purpose of mid is to determine a value for rmid. There is no reason for mid to change any other aspect of the data environment, so we will assume that it does not.

Therefore, we conclude that the specification for the subprogram mid is:

{Vmid} call mid {Pmid} strictly and

for all d satisfying the strict precondition Vmid: mid.d = d except for the value of rmid

where Vmid is defined to be

$p1 \in \text{Strings} \wedge p2 \in \mathbf{Z} \wedge p3 \in \mathbf{Z} \wedge \text{Strings} \subseteq \text{Set.“rmid”}$ [abbreviated V0 below]

$\wedge$

$1 \leq p2 \leq p2+p3\text{-}1 \leq \text{length}(p1)$ [V1, see above]

i.e., Vmid = (V0 $\wedge$ V1), and where by definition

Pmid = [rmid = substring(p1, p2, p3)]

2. Question 2: Specification of the subprogram midgen to be designed

The specification for the new subprogram midgen should be stronger than that for mid, i.e. the precondition for midgen should be weaker than the precondition for mid and/or the postcondition for midgen should be stronger than the postcondition for mid. Because we do not wish to restrict the values of p1, p2 and p3 any more than absolutely necessary, we require that the precondition for midgen be V0 alone. I.e., we drop the V1 term. We will model the postcondition for midgen on the postcondition for mid, but with the additional consideration that the effective values of p2 and p3 must be “meaningful” values in the sense of condition C1 (see section 1 above). By “effective” values here we mean the values actually used to determine the substring to be returned by midgen (not necessarily the actual values of p2 and p3 input to midgen).

The specification for midgen becomes, then:

{V0} call midgen {Pmidgen} strictly and

for all d satisfying the strict preconditionV0: midgen.d = d except for the value of rmid

where (by definition)

Pmidgen = [rmid = substring(p1,

lim(1, p2, length(p1)+1),

lim(0, p3, length(p1)-lim(1, p2, length(p1)+1)+1))]

3. Question 3: Proof that midgen is more general than mid

To prove is that midgen as specified in section 2 above satisfies the specification for mid (see section 1 above), i.e. that

{Vmid} call midgen {Pmid} strictly and

for all d satisfying the strict precondition Vmid: midgen.d = d except for the value of rmid

The second line above is true; see the corresponding part of the specification of midgen in section 2 above and note that Vmid = V0 $\wedge$ V1, i.e. that Vmid$\subseteq$V0.

The proof of the first line above, i.e. {Vmid} call midgen {Pmid} strictly, is sketched below:

{Vmid}

{V0 $\wedge$ V1} [= Vmid]

call midgen

{Pmidgen $\wedge$ V1}

{Pmid} strictly

The specification of midgen (see section 2 above) states that

{V0} call midgen {Pmidgen} strictly.

Because midgen does not modify any variable referenced in V1, it is also true that

{V1} call midgen {V1}.

Therefore, {V0 $\wedge$ V1} call midgen {Pmidgen $\wedge$ V1} strictly.

By examining V1 (= 1$\leq$p2$\leq$p2+p3-1$\leq$length(p1)), it is easily verified that when V1 is true

p2 = lim(1, p2, length(p1)+1)

and

p3 = lim(0, p3, length(p1)-lim(1, p2, length(p1)+1)+1)

in which case Pmidgen becomes Pmid. I.e. Pmidgen$\wedge$V1$\Rightarrow$ Pmid.

Thus, any program that satisfies the specification for midgen also satisfies the specification for mid. I.e., if midgen is called when the precondition for mid is satisfied, then the result of executing midgen will be defined and it will satisfy the postcondition for mid.

4. Question 4: Design of midgen

We begin by noting the relationship between the condition C1 (for "meaningful" values of p1, p2 and p3) and the condition V1 (part of the precondition for the subprogram mid):

$[C1(p1, p2, p3) \wedge p3 \neq 0] = V1$

The design of midgen is straightforward after the several analyses above have been completed. The subprogram midgen calculates effective values for p2 and p3 (so that condition C1(p1, p2, p3) is satisfied) and if the results satisfy V1, midgen calls mid to determine rmid. If the resulting effective values for p2 and p3 do not satisfy V1, midgen assigns the null string as the value of rmid. Once condition C1 is satisfied, V1 will be satisfied if and only if p3≠0 (see above).

When calculating effective values for p2 and p3, new variables must be declared because midgen may not change the original values of p2 and p3 (see the specification for midgen). The new variables must be released after executing mid.

The formulae for effective values for p2 and p3 refer to the length of the string p1. The given subprogram len will calculate this value, provided that p1 is not the null string. The subprogram midgen must set rlen to 0 if p1 is the null string, otherwise, midgen may call len.

The subprogram midgen becomes, then:

```
{V0}
    declare (rlen, Z, 0)
    if p1≠"" then call len endif

    declare (p2, Z, max(1, min(p2, rlen+1)))
    declare (p3, Z, max(0, min(p3, rlen-p2+1)))
    if p3=0
    then rmid := ""
    else call mid
    endif
    release p3
    release p2
    release rlen
{Pmidgen}
```

5. Question 4: Scheme for a proof of correctness for midgen

The postcondition Pmidgen refers to the variables p2 and p3, which are released at the end of midgen. In this form and with the notation and proof rules we have covered, there is no convenient way to work Pmidgen backwards over the statements releasing p2 and p3. We therefore define P1midgen, a modified (but equivalent) form of the postcondition which refers not to the program variables p2 and p3, but to the specification variables p2' and p3' instead:

P1midgen = [rmid = substring(p1,

lim(1, p2', length(p1)+1),

lim(0, p3', length(p1)-lim(1, p2', length(p1)+1)+1))]

Because P1midgen does not refer to any of the program variables rlen, p2 or p3, P1midgen is a precondition of itself with respect to the three release statements at the end of the program.

The following scheme for proving the correctness of midgen shows the main intermediate conditions which would arise in a formal proof.

{V0 ∧ p2=p2' ∧ p3=p3'} [strict precondition, definition of specification variables]
declare (rlen, Z, 0)
if p1≠"" then call len endif
{V0 ∧ p2=p2' ∧ p3=p3' ∧ rlen=length(p1)}
declare (p2, Z, max(1, min(p2, rlen+1))) [lim(1, p2, length(p1)+1)]
declare (p3, Z, max(0, min(p3, rlen-p2+1))) [lim(0, p3, length(p1)-p2+1)]
{V0 ∧ p2=lim(1, p2', length(p1)+1)
∧ p3=lim(0, p3', length(p1)-p2+1)}
if p3=0
then
{V0 ∧ p2=lim(1, p2', length(p1)+1) ∧ 0=p3=lim(0, p3', length(p1)-p2+1)}
rmid := ""
{P1midgen}
else
{V0 ∧ p2=lim(1, p2', length(p1)+1)
∧ 1≤p3=lim(0, p3', length(p1)-p2+1)}

{Vmid ∧ p2=lim(1, p2', length(p1)+1) [Vmid = (V0 ∧ V1)]
∧ 1≤p3=lim(0, p3', length(p1)-p2+1)}
call mid
{Pmid ∧ p2=lim(1, p2', length(p1)+1) ∧ p3=lim(0, p3', length(p1)-p2+1)}

{P1midgen}
endif
{P1midgen}
release p3
release p2
release rlen
{P1midgen ∧ p2=p2' ∧ p3=p3'}
{Pmidgen}

and for all d satisfying the strict preconditionV0: midgen.d = d except for the value of rmid

{p2=p2' ∧ p3=p3'} is a precondition of itself with respect to the entire program midgen because neither p2 nor p3 is changed by midgen. The release statements at the end of midgen restore the original variables named p2, p3 and rlen. The value of rmid is set to a new value. No other

variable is modified by the program, therefore the last part of the specification (the statement about midgen.d) is true.

Because p2=p2' and p3=p3' after midgen has executed, one may replace p2' by p2 and p3' by p3 in P1midgen. The result is Pmidgen.

(end)