Loops and recursion

By tail recursion removal these commands are equivalent

$f(a)$
where f is defined by
procedure $f(p)$
  if e then S $f(b)$
  else T
  end if
end f
	var $p := a$
	while e do
	  S
	  $p := b$
	end do
	T

Loops without invariants

So we can use techniques for reasoning about recursive procedures to reason about loops.

Consider (again) the binary search problem

{ x is sorted }
? without changing x or t, find i such that
{ $(t \in x\{0, ..x.\text{length}\} \Rightarrow x(i) = t)$ and
$(t \notin x\{0, ..x.\text{length}\} \Rightarrow\ i = -1)$ }

We can solve this by calling a procedure that searches only part of the array

$search(0, x.\text{length}, t, x, i)$

where

procedure $search(p, r : \text{Int}, t : T, x : \text{array}\ \langle T \rangle\ , \text{var}\ i)$
precondition $0 \leq p \leq r \leq x.\text{length}$ and x is sorted
postcondition $t \in x\{p, ..r\} \Rightarrow x(i) = t$
and $t \notin x\{p, ..r\} \Rightarrow\ i = -1$

Now we can implement the procedure using recursion

procedure $search(p, r : \mathrm{Int},\ t : T,\ x : \mathrm{array}\ \langle T \rangle\ ,\ \mathrm{var}\ i)$
precondition $0 \le p \le r \le x.\mathrm{length}$ and x is sorted
postcondition $t \in x\{p, ..r\} \Rightarrow x(i) = t$
and $t \notin x\{p, ..r\} \Rightarrow\ i = -1$
  if $r - p > 1$ then
    val $q := \left\lfloor \frac{p+r}{2} \right\rfloor$
    if $x(q) \le t$ then $search(q, r, t, x, i)$
    else $search(p, q, t, x, i)$
    end if
  else // $r - p = 1 \vee p = r$
    if $p = r$ then $i := -1$ elseif $x(p) = t$ then $i := p$ else
    $i := -1$ end if
  end if
end $search$

By tail recursion removal, our problem can be solved by

$p := 0$
$r := x.\mathrm{length}$
while $r - p > 1$ do
  val $q := \left\lfloor \frac{p+r}{2} \right\rfloor$
  if $x(q) \le t$ then $p := q$ else $r := q$ end if
end while
if $p = r$ then $i := -1$ elseif $x(p) = t$ then $i := p$ else
$i := -1$ end if

This is the same result we got in slide set 3.

But we did not need an invariant!

(The invariant from slide set 3 was

$$I: \quad \begin{array}{l} 0 \leq p \leq r \leq x.\text{length} \\ \wedge\ (t \in x\,\{0, ..x.\text{length}\} \Rightarrow t \in x\,\{p, ..r\}) \end{array}$$

and nothing like it makes an appearance in this development.)

- The invariant looks backward. It summarizes what we have learned/accomplished so far.

  "If t is anywhere in the array, it must be in the region $x\,\{p, ..r\}$"

- The recursion based approach to loops, which we just used, looks forward.

  The specification of the procedure explains what remains to be accomplished each time we (re)start the loop.

  "Determine whether $x\,\{p, ..r\}$ contains a t and, if so, indicate it's location"