CSci 117 10/18/18 Review

```
local
   Key = {NewName}
   fun {Wrap X}
      fun {$ K} if K==Key then X end
   end
   fun {Unwrap P}
      {P Key}
   end
   data structures, operations

   fun {Push S X}
     {Wrap X|{Unwrap S}}
   end

in
   stack(push:Push pop:Pop ...)
end

   declare Z
   Z = 2
   X = proc {$ Y1 .... Yn} <S> end
                          ↖ free variable Z

proc {X Y1 ... Yn} <S> end

   {X = (proc {$ Y1 ... Yn} <S> end, Z → z)}
                                  context Env.

{X 1 2 3 4 5}
```

```
fun lazy {Generate N} N|{Generate N+1} end
```

```
fun {Generate N}
   {ByNeed fun {$} N|{Generate N+1} end}
end
```

```
proc {Generate N ?S}
   {ByNeed proc {$ X} local Y in
              X = N|Y
              {Generate N+1 Y}
           end
      S}
```

List Accum Return value

```
proc {Sumlist L A ?S}
   case L of nil then S = A
   [] X|Xr then {Sumlist Xr X+A S}
   end
end
```

Invariant $S = (\text{sum elts of } L) + A$

Proof of invariant. By induction on list

Base case: $L = nil$. Sum elts of nil is 0

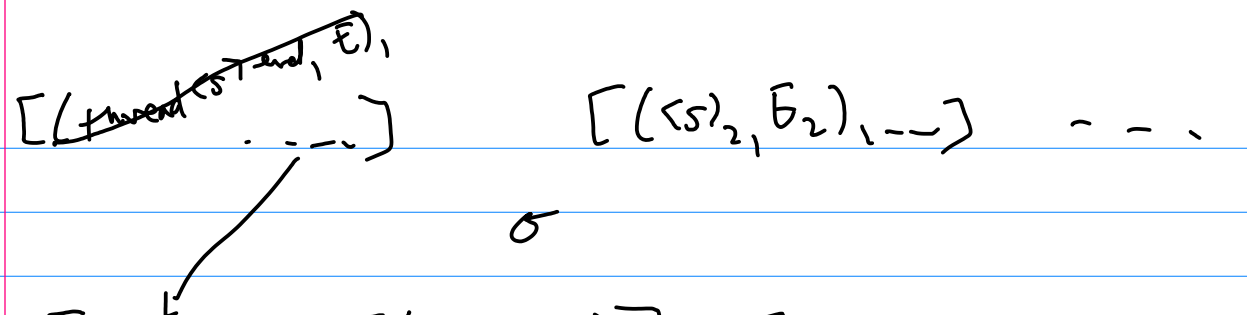
$S = 0 + S$ ✓

Inductive case $L = X|Xr$. By the IH,

$S = (\text{sum elts of } Xr) + (X + A)$

goal: $S = (\text{sum elts of } L) + A$ ✓

$X + (\text{sum elts of } Xr) + A$



Difference List

$L = (1|2|3|4|X) \# X$

$X = 5|Y$

$\longrightarrow L \# Y$

$L1 \# X$

$L2 \# Y$

$X = L2$ O(1)

$L1 \# Y$

```
local W in
   W = 1
   X = proc {$ Y Z} Z = Y + W end
   :
   {W}  §X = p = (proc §Y Z} Z = Y+W end, {W → w})}
   {X A B}
in
   rec(X)
```

$\left(Z = Y + W, \{Y \to a, Z \to b, W \to w\}\right)$

```
local W in
   W = 5
   {X 1 2}
```

$(\{X\ 1\ 2\}, \{W \to 5\})$