Finish Auklet material

Start Bluebird*

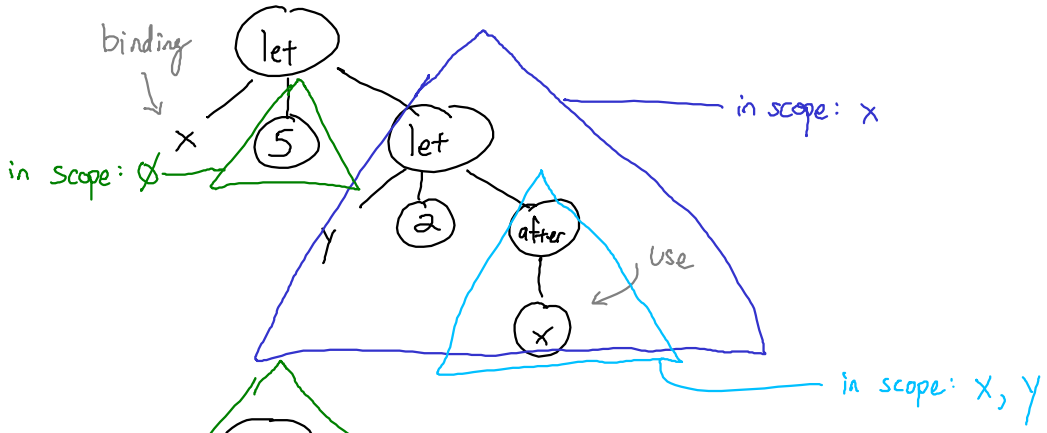
- Integers
- Arithmetic — binary operators (2 args)
- after, before — unary (1 arg)
- vars — let, vars

Environment: keeps track of where we're storing variable values

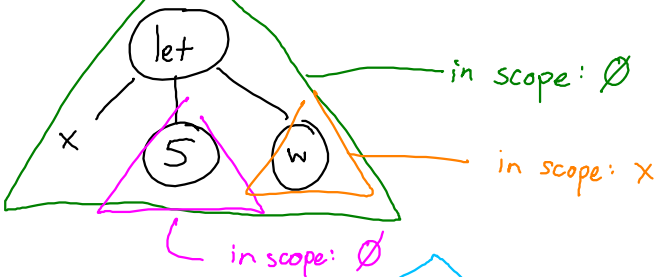
$$(-24, \{ x \mapsto [rsp-8],\ y \mapsto [rsp-16] \})$$

Scope: places where variable is available to be used

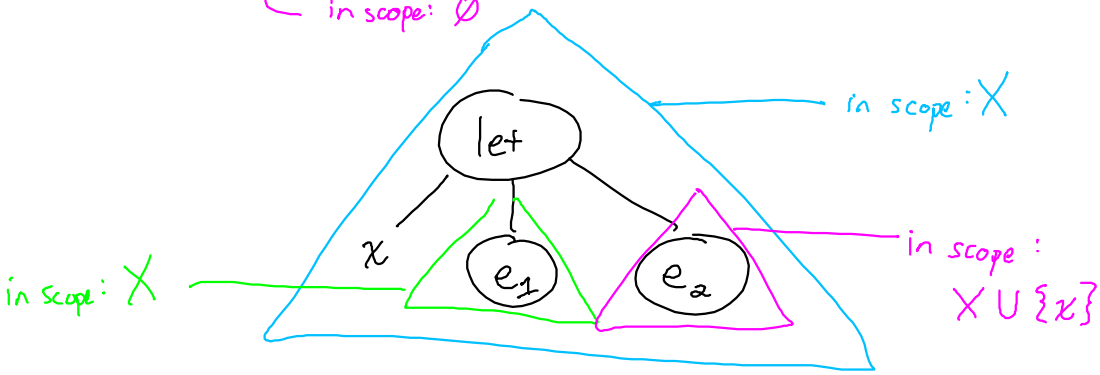
```
let x=5 in
let y=2 in
after(x)
```



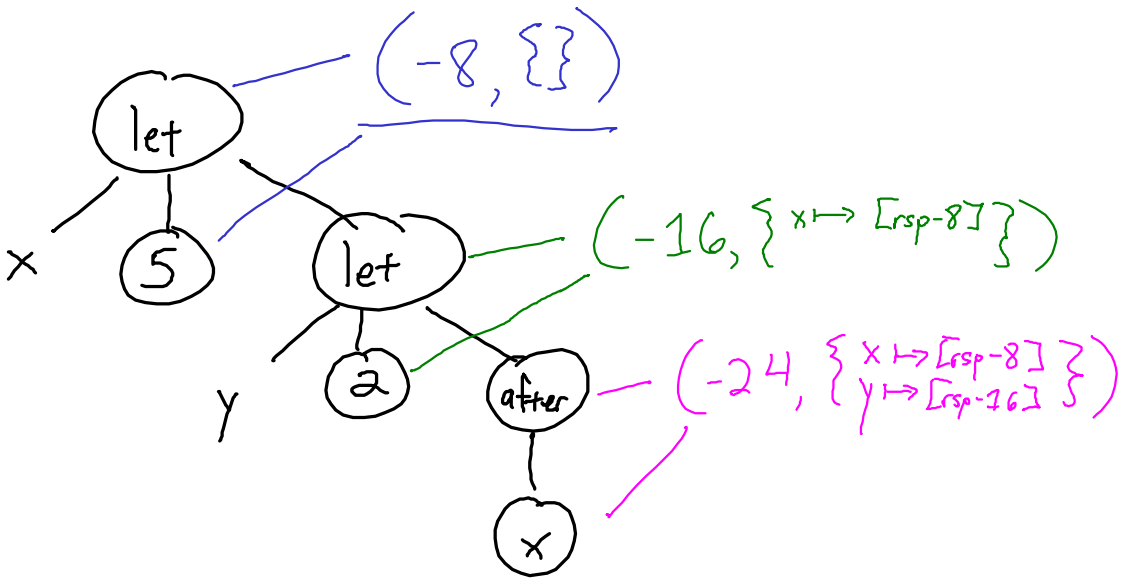

```
let x=5 in w
```



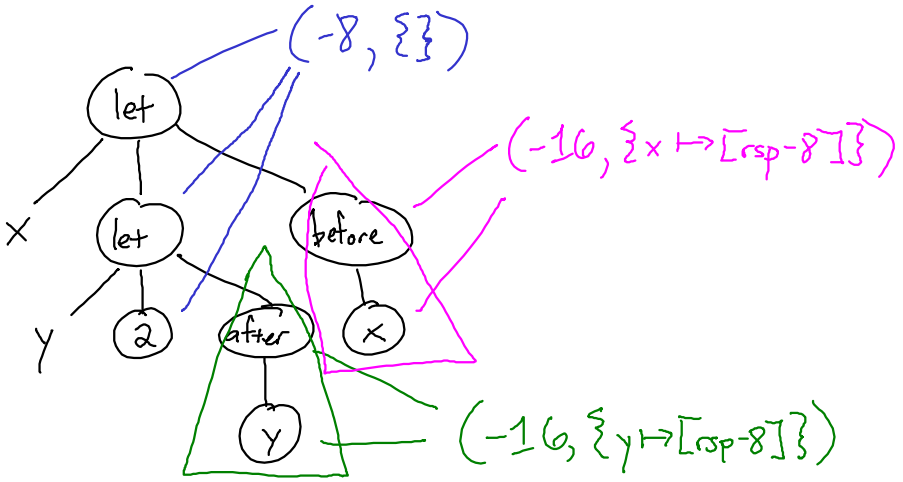

```
let x=x in x
```



1. Compile e_1
2. Store it
3. Compile e_2 w/ an env referring to loc of 1st



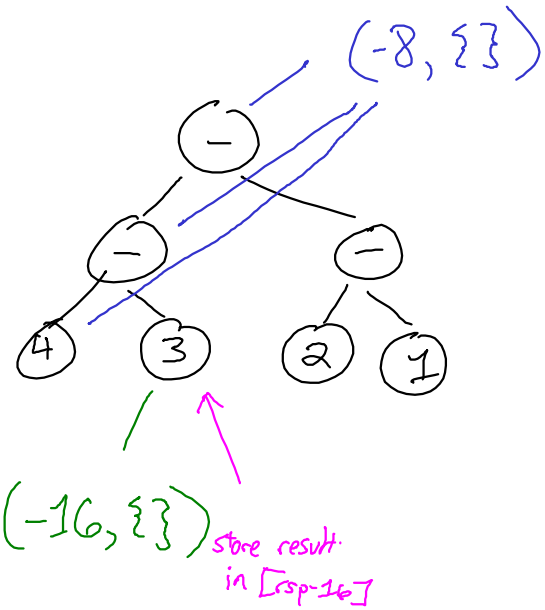

```
let x =
    let y = 2 in
    after(y)
in
before(x)
```




```
mov rax, 2
mov [rsp-8], rax
mov rax, [rsp-8]
inc rax
```
} y is in scope

```
mov [rsp-8], rax
mov rax, [rsp-8]
dec rax
```
} x is in scope

$(4-3)-(2-1)$




```
mov rax, 4
mov [rsp-8], rax
mov rax, 3
mov [rsp-16], rax
mov rax, [rsp-8]
sub rax, [rsp-16]
```

store result in [rsp-8]

```
mov [rsp-8], rax
```

```
mov [rsp-16], rax
mov rax, [rsp-8]
sub rax, [rsp-16]
```

C-bird

<expr> ::= ... | ifnz <expr> then <expr> else <expr>

ifnz after(2) then 6 else 8 ⟹ 6

Label

mylabel:

Comparison

cmp rax, 5

Jumps

jmp mylabel

jnz jl jle jne
je jg jge

```
    mov rax, 2
    inc rax
    cmp rax, 0
    jne then_2
    mov rax, 8
    jmp end_3
then_2:
    mov rax, 6
end_3:
```