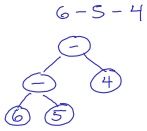
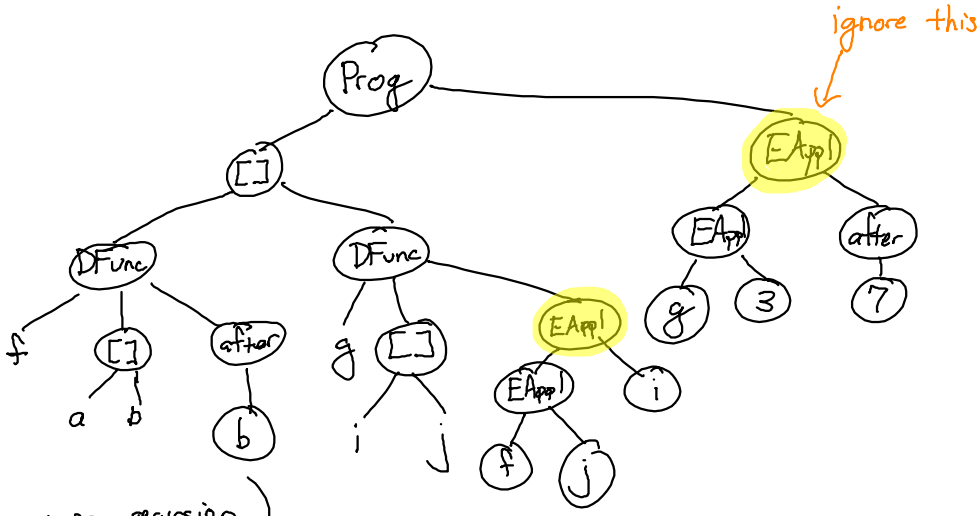
Tail Call Optimization

goal: eliminate stack frames that won't be used

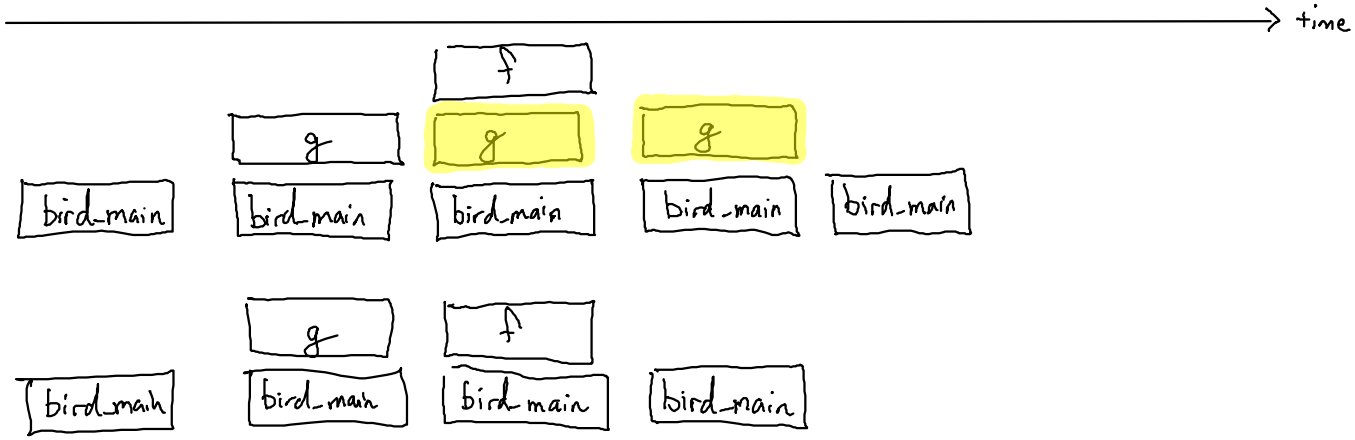


- an expression e_1 of another expression e_2 is a tail expression if ① e_1 is a descendant of e_2, ② the result of e_2 is the result of e_1, and ③ computing e_1 is the last step of e_2
- tail call expression is an application expression which is a tail expression of a function body

```
def f a b =
  after(b)
end
def g i j =
  (f j) i
end
g 3 (after(7))
```



(Real goal: save stack memory on recursion)



Tail Callee (TCE) f

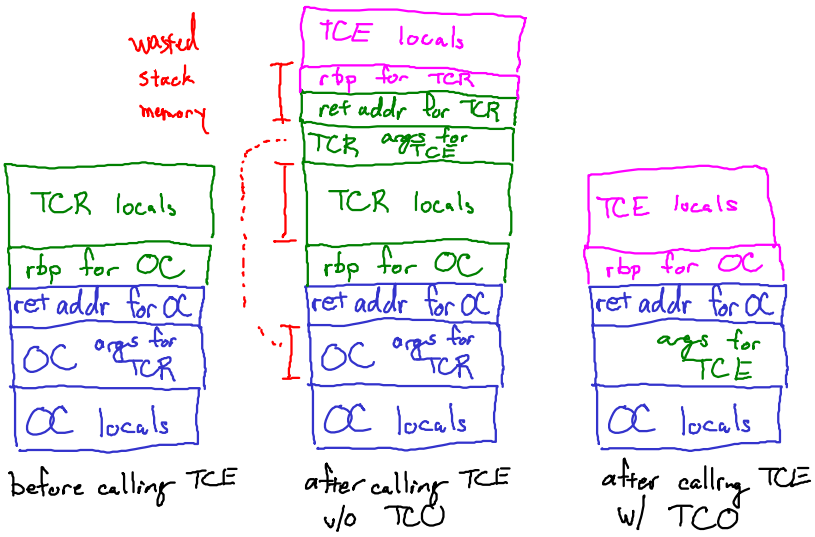
Tail Caller (TCR) g

Original Caller (OC) bird_main

Only the TCR knows this is happening!

call foo ≡	push rip jmp foo

(for now: pretend we know TCE and TCR have same # of args)



- Replace TCR args w/ TCE args
- Tear down TCR stack frame
- pop rbp
- jmp to TCE

① Static: is tail call

② dynamic: ←

args for TCR > # args for TCE : ok

args for TCR < # args for TCE : no tail call