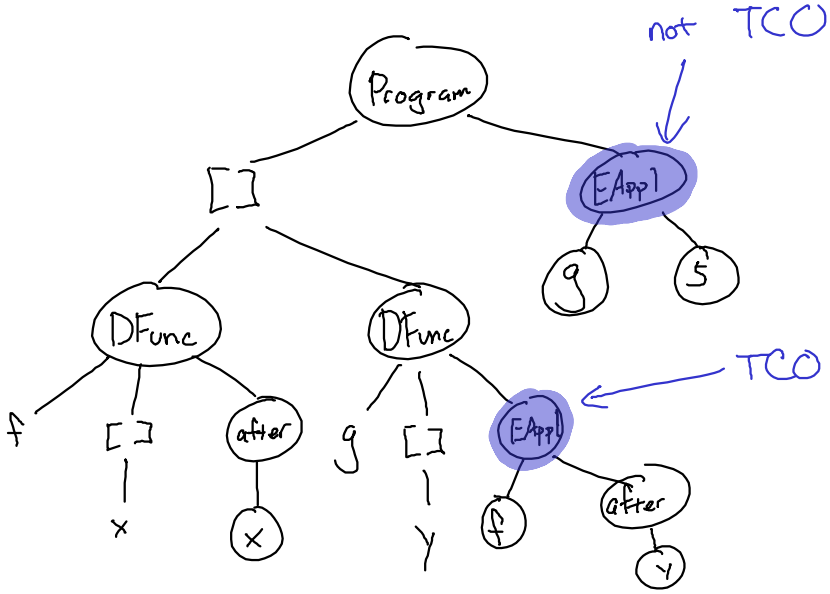
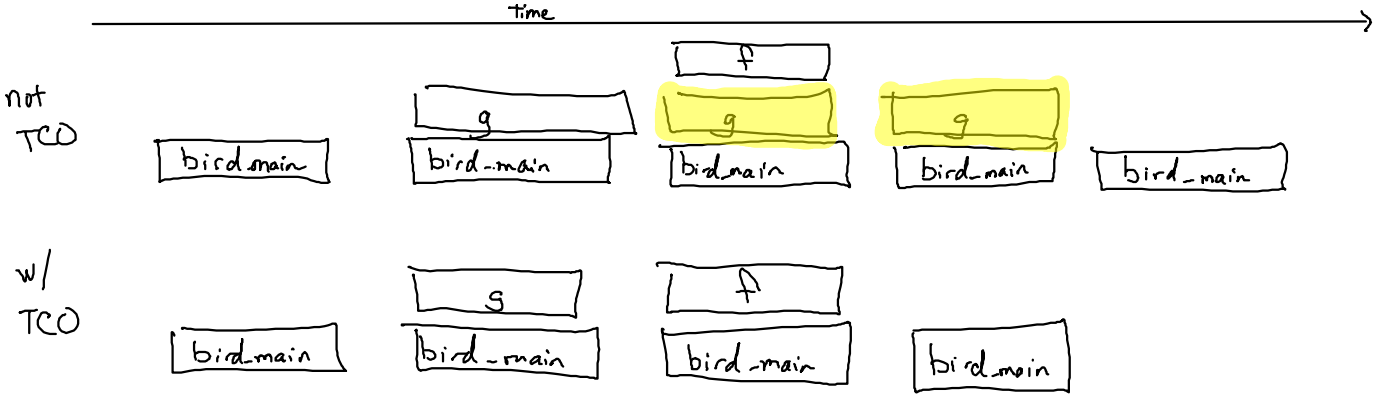
Tail Call Optimization

- Tail expression of a larger expression ①-produces result of larger expression. and ② is last thing larger expr does
- Tail call is a call/application which is a tail expression of overall func.
- Tail call optimization reclaims stack memory during a tail call
  - Remove stack frame before calling b/c it will not be used again

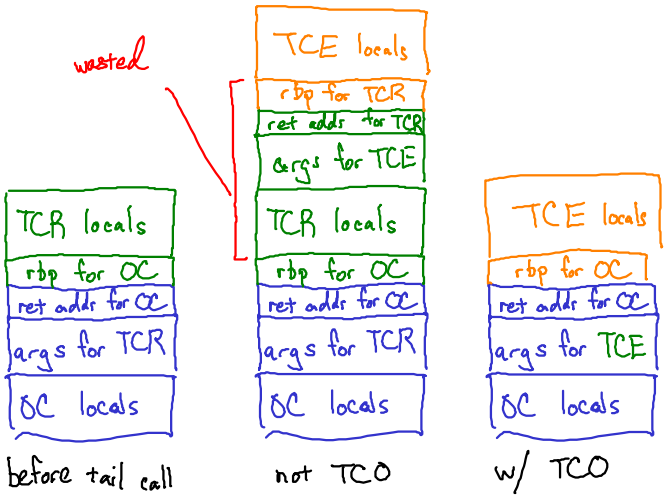
```
def f x =
  after(x)
end
def g y =
  f(after(y))
end
g 5
```

TCO





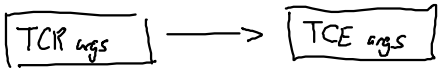
Tail Callee (TCE) (f)
Tail Caller (TCR) (g) ← only one that knows what's happening
Original Caller (OC) (bird_main)



Assume TCR #args = TCE #args:

1. Replace TCR args w/ TCE args
2. Tear down stack frame (including pop rbp)
3. jmp to TCE

TCR args # = TCE args #



TCR args # > TCE args #

TCR args # < TCE args #

don't TCO

1. Static: saturated application in a tail call position
2. Dynamic: check # args of TCE ≤ # args of TCR

or change calling conventions

1. add to stack # of args and caller must use that # when deallocating
2. using registers would help
3. args to fn call become a heap object
4. callee's job to remove args