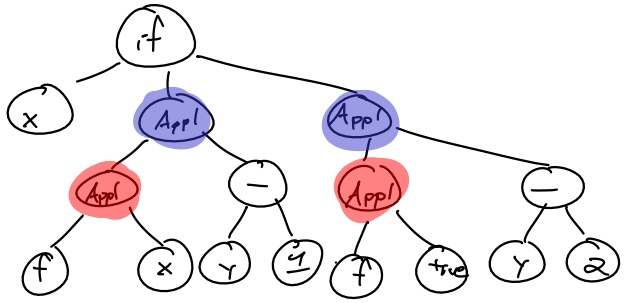
Tail Call Optimization

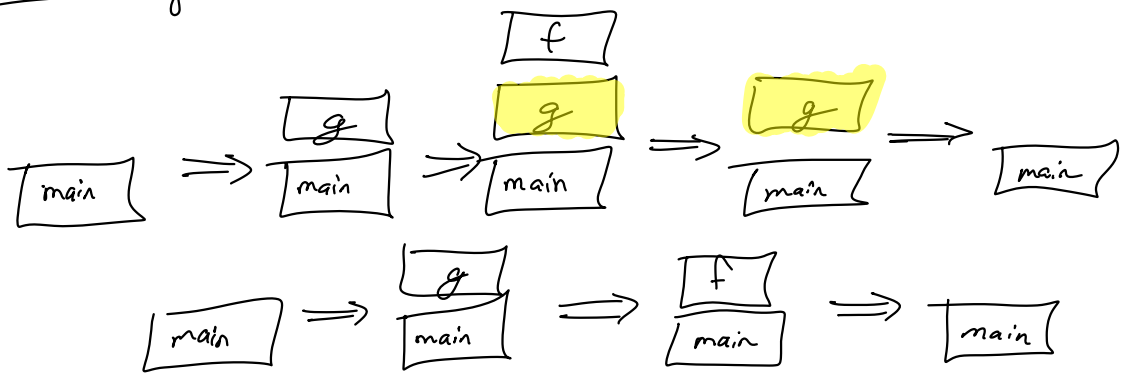
- Tail expression: last thing a larger expression does (e.g. function body)

```
def f x y =
    if x then (f x)(y-1) else (f true)(y-2)
end
```

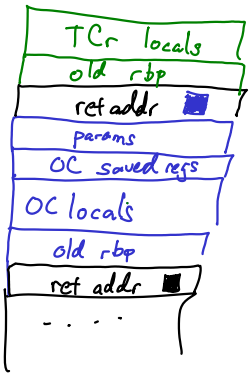


- Any call in a tail position can reclaim current stack frame before calling

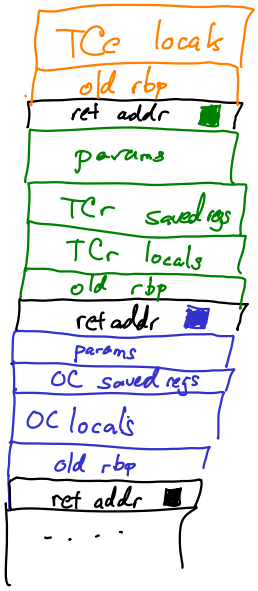
```
def f x
    x + 1
end
def g y
    f (y + 1)
end
g 2
```



Tail Callee
Tail Caller
Original Caller



before call

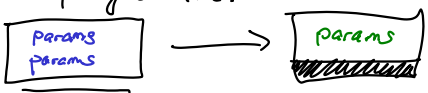


w/o TCO

w/ TCO

1. Store TCe params over our own params (careful not to destroy our own data) (assuming we have enough room)
2. Tear down TCr stack frame (including pop rbp)
3. Jmp (not call) to the TCe

- If # params for TCr = # params TCe then everything's fine.
- If # params for TCr > # params TCe



- If # params for TCr < # params TCe

don't tail call !!

To tail call:

1. Call is in tail pos (compile-time)
2. Callee wants # args ≤ Caller's (run time)

```
def f g h b =
    if b then g 1 else h 2 3
end
```

1. Overallocate params (experimental data)
2. Change calling convs to store args on heap (extra deref)
3. Change calling convs to either
   a. change who is responsible for arg cleanup
   b. use a register to communicate # params