Tail Call Optimization

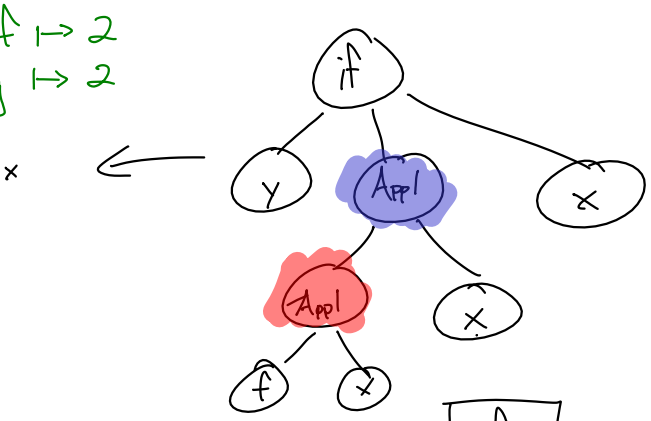
What is? When performing a call, we can remove a stack frame before creating a new one.

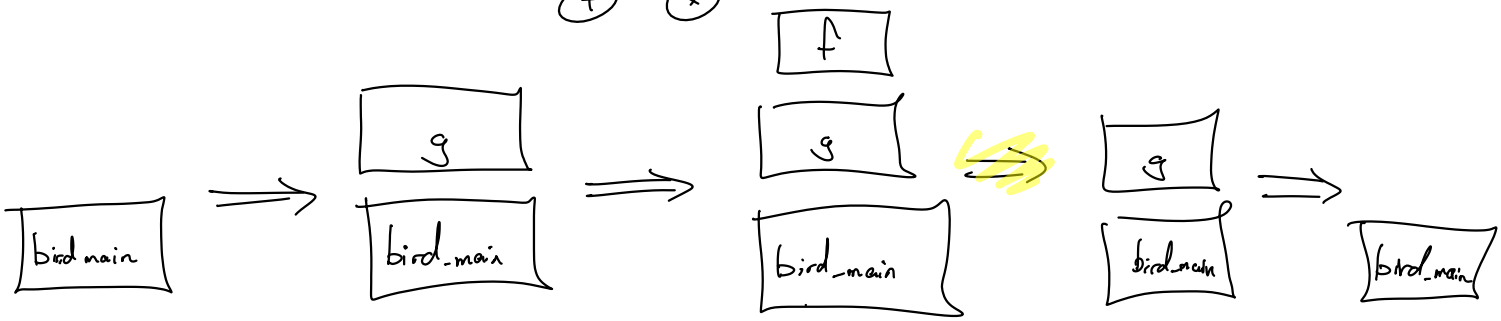
Why? Loops = Recursion, recursion eats stack memory
$\Rightarrow$ Limit on size loop

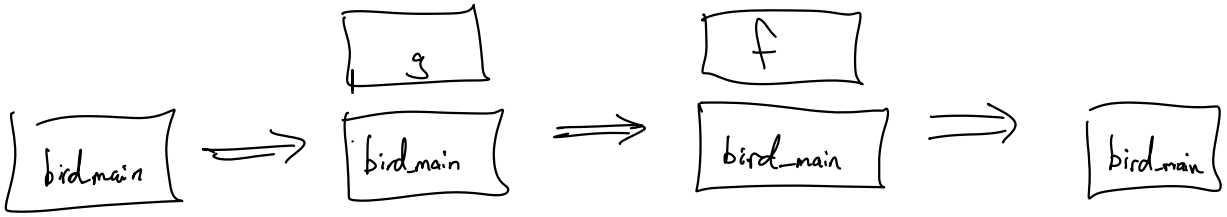
Save stack memory using TCO; run programs that can't run otherwise

```
def f a b          # 2 params
  a + b
end
def g x y =        # 2 params
  if y then (f x) x else x
end
g 5 true
```

$f \mapsto 2$
$g \mapsto 2$





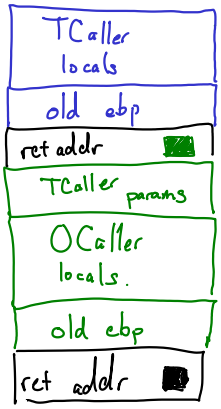


Tail Callee

Tail Caller

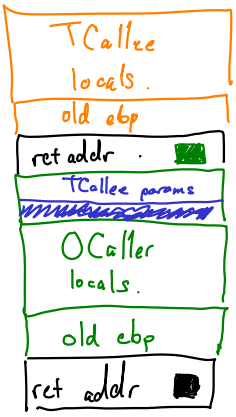
Original Caller

TCaller # params ≥ TCallee # params



Before Tail Call

If Call is not tail

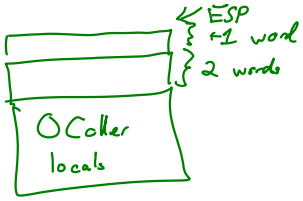


If we tail call

- Replace my own params w/ params of TCallee
- Tear down my stack (exit 1/2 of C callee) (pop "old ebp" into EBP)
- Jmp (NOT Call) the TCallee fn

Hypothetical: TCaller has 2 params & TCallee has 3

!! Just normal call



TCO info

- Is this a tail pos?
  Static info (compile-time)
- TCallee # params ≥? TCaller # params
  Dynamic info (run-time)

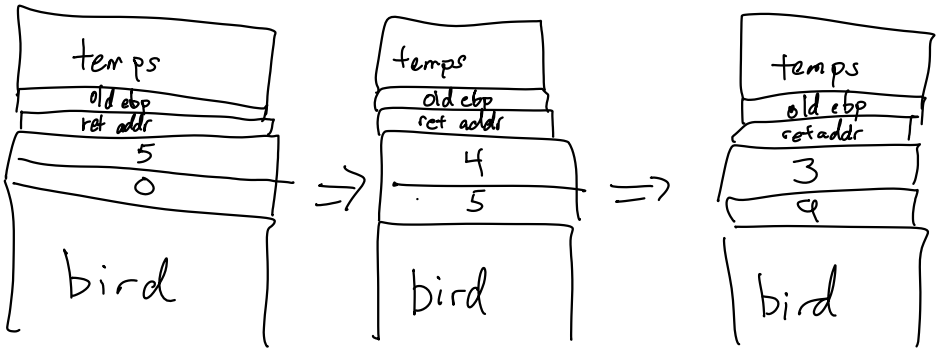
```
def f x y =
  x - y
end
def g x =
  x
end
def h x b q =
  (if b then f 2 else q) x
end
```

?

```
def summate n acc =
  if n=0 then acc else
    summate (n-1) (acc+n)
end
```

```
int summate (int n, int acc) {
  while (n>0) {
    acc+=n;
    n-=1;
  }
  return acc;
}
```

summate 5 0



TCallee params > # TCaller params

Callee: set up locals tear down locals tear down params

Caller: set up params